


Thèse présentée pour l'obtention du grade de

DOCTEUR de SORBONNE UNIVERSITÉ

Spécialité

Ingénierie / Systèmes Informatiques

École doctorale

Informatique, Télécommunication et Électronique Paris (ED130)

Model-based Reinforcement Learning in the Era of Foundation Models

---

Abdelhakim Benechehab

Soutenue publiquement le : *13 Mai 2026*

Devant un jury composé de :

Yann Chevaleyre, Professeur, Dauphine PSL	*Président du jury*
Sylvain Lamprier, Professeur, Université d'Angers	*Rapporteur*
Yingzhen Li, Professeure, Imperial College London	*Rapporteure*
Olivier Sigaud, Professeur, Sorbonne Université, ISIR	*Examinateur*
Claire Vernade, Professeure, University of Technology Nuremberg	*Examinatrice*
Giuseppe Paolo, Research scientist, Cognizant AI	*Encadrant industriel*
Albert Thomas, Research scientist, Huawei Noah's Ark Lab	*Encadrant industriel*
Maurizio Filippone, Professeur, KAUST	*Directeur de thèse*
Balázs Kégl, Research scientist, Meta	*Invité*

To all of the people that light up my life. 🐾

Abstract

Reinforcement Learning (RL) provides a general framework for sequential decision-making, enabling agents to learn through interaction with an environment. Despite successes in domains such as games and robotics, it remains challenging to deploy in real-world settings, notably due to limited sample efficiency. Model-based Reinforcement Learning (MBRL) addresses part of this issue by learning a model of the environment dynamics for planning, thereby reducing costly interactions. However, learned models introduce new challenges, including compounding errors and model-policy objective mismatch. In parallel, machine learning has shifted toward Foundation Models (FMs): large-scale pre-trained models that learn transferable representations from massive datasets. Initially developed for natural language processing, they are now emerging in areas such as time series modeling, raising questions about their role in RL.

This thesis investigates how integrating FMs can strengthen MBRL. Adopting a sequence modeling perspective, it first revisits dynamics modelling by directly optimizing a multi-step objective to improve long-term accuracy. Building on this view, it then formulates next-state prediction as an in-context learning problem and introduces a latent projection mechanism that enables zero-shot use of pre-trained Large Language Models (LLMs) within MBRL. Extending these representation learning ideas further, the thesis addresses the adaptation of univariate Time Series Foundation Models (TSFMs) to multivariate and probabilistic forecasting through learnable encoder–decoder adapters. Finally, shifting the focus from dynamics to rewards, it proposes a bilevel optimization approach to learn implicit reward functions from supervised data, broadening the applicability of RL to settings where explicit reward design is impractical.

Overall, this work demonstrates how combining MBRL with FMs leads to more sample-efficient and generalizable decision-making systems.

Keywords:

Model-based Reinforcement Learning, Foundation Models, Adaptation, Time Series

Résumé

L'apprentissage par renforcement (RL) fournit un cadre général pour la prise de décision séquentielle, permettant à des agents d'apprendre par interaction avec un environnement. Malgré des succès dans des domaines tels que les jeux et la robotique, son déploiement dans des contextes réels demeure difficile, notamment en raison d'une efficacité limitée en termes d'échantillons.

L'apprentissage par renforcement basé sur des modèles (MBRL) répond en partie à ce problème en apprenant un modèle de la dynamique de l'environnement afin de planifier, réduisant ainsi les interactions coûteuses. Cependant, les modèles appris introduisent de nouveaux défis, notamment l'accumulation d'erreurs et le décalage entre les objectifs du modèle et de la politique.

Parallèlement, l'apprentissage automatique a évolué vers les modèles de fondation (FM) : des modèles pré-entraînés à grande échelle qui apprennent des représentations transférables à partir de jeux de données massifs. Initialement développés pour le traitement automatique du langage naturel, ils émergent désormais dans des domaines tels que la modélisation de séries temporelles, soulevant des questions quant à leur rôle en apprentissage par renforcement.

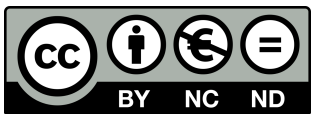
Cette thèse étudie comment l'intégration de modèles de fondation peut renforcer le MBRL. En adoptant une perspective de modélisation séquentielle, elle revisite d'abord la modélisation des dynamiques en optimisant directement un objectif multi-pas afin d'améliorer la précision à long terme. S'appuyant sur cette vision, elle reformule ensuite la prédiction de l'état suivant comme un problème d'apprentissage en contexte (in-context learning) et introduit un mécanisme de projection latente permettant l'utilisation en zero-shot de grands modèles de langage (LLM) pré-entraînés au sein du MBRL. En prolongeant ces idées d'apprentissage de représentations, la thèse aborde l'adaptation de modèles de fondation de séries temporelles univariées (TSFM) vers des prévisions multivariées et probabilistes grâce à des adaptateurs encodeur–décodeur apprenables. Enfin, en déplaçant l'attention des dynamiques vers les récompenses, elle propose une

approche d'optimisation bi-niveau pour apprendre des fonctions de récompense implicites à partir de données supervisées, élargissant ainsi l'applicabilité de l'apprentissage par renforcement aux contextes où la conception explicite de la récompense est impraticable.

Dans l'ensemble, ce travail montre que la combinaison du MBRL et des modèles de fondation conduit à des systèmes de prise de décision plus efficaces en données et plus généralisables.

Mots-clés :

Apprentissage par renforcement basé sur des modèles, Modèles de fondation, Adaptation, Séries temporelles



Acknowledgements

I would like to express my deepest gratitude to everyone who has supported me throughout this PhD journey.

First and foremost, I want to thank my advisors, who made this journey incredible from day one and helped me in every possible way they could. Maurizio, for giving me the freedom to explore various research directions, for always being available to discuss ideas, and for your continuous trust and encouragement. Balázs, for answering my cold email back in 2021, for believing in me at a time when I knew close to nothing about research, and for your constant support, guidance, and mind-blowing philosophical takes. Albert, for your unmatched kindness, for always being there to listen and help, for being a great mentor in research, engineering, and life in general, and for being a great friend. Giuseppe, for your constant presence and support, for your brilliant insights and ideas, for all the technical tricks you taught me, and for being a great friend.

I extend my sincere thanks to Prof. Sylvain Lamprier and Prof. Yingzhen Li for accepting to review this thesis, your feedback and insights contributed to improving the quality of this work. I also want to thank the jury members, Prof. Olivier Sigaud, Prof. Yann Chevaleyre, and Prof. Claire Vernade, for your time and effort in evaluating this thesis.

This work was carried out as part of a collaboration between Huawei Noah's Ark Lab and the EURECOM Data Science department, I am grateful to both institutions for providing the best possible conditions for my research. I am thankful to all the colleagues and friends I met during my four years at Huawei Noah's Ark Lab in Paris: Vasilii, Ievgen, Jonas, Malik, Aladin, Hamza, Chunchun, Keli, Lei, Shifeng, Gabriel, Corentin, Ludovic, and others. I also thank fellow PhD students, with whom I shared the ups and downs of PhD life, worries about the future, ping pong games, and LeetCode preparation tips: Youssef, Ambroise, Paul, Alexandre, Ba Hien, Yassine, Yasser, Naoufal, and others.

I want to thank my family and friends, to whom I owe everything, and to whom I dedicate this work, starting with my beloved parents نبيلة و عبد العزيز.

رغم أنني لست حاضرًا معكما كما ينبغي، فإنني أحبكما كثيرا، وأسعى لأن أكون مصدر فخر لكما، كما أنني فخور بكما.
أهديكما هذا العمل، وأعبر لكما عن امتناني العميق لكل ما قدمتماه لي.

To my brother Redouan, for being an inspiration throughout my life, to my niece Jood and nephew Taha, for bringing so much joy to the family. To my late uncle, Said, and his family, for their invaluable support during my early days in France and afterward. To all my family members in Morocco and in France, for the memories we shared together.

To my significant other, Lamia, for your unconditional love and support, for being patient with me in the toughest moments, and for sharing with me the joys of this journey.

To Nohaila & Mahdi, for supporting me (and my laptop) during this journey, for all the activities we shared together, and for all the trips I experienced from the backseat of your car. To Asmae and Chaimaa, for the meals we shared on weekends and the Catan games we played. To my friend Ichraq, for sharing part of this journey with me and for the fun times we had together. To my friends Omar, Anass, Younes, Othman, Gyros, Zineddine, Ayoub, Nabil, and others with whom I shared unforgettable moments.

Finally, I would like to thank volleyball, CrossFit, padel, Climb'Up Porte d'Italie, my guitar, Radio Mars, Lofi Girl, Moroccan rap, Italian courses, chess, coffee, juggling, the 40 countries I visited in previous years, Champions League nights, the UFC, Too Good To Go, Vélib', Obsidian, ChatGPT, Claude, and Gemini for making my life easier.

Contents

Chapter 1

Introduction

Nowadays, activities such as writing a book, recognizing items in an image, and predicting stock prices, can hardly be spoken about without mention of Artificial Intelligence (AI). The common denominator between the evoked activities is the fact that they rely on massive analoguous data to drive decision. A writer can't publish a best-seller without having read thousands of pages from fellow writers. The highly skilled technician and the experienced physician use years of pattern recognition training to hope achieving high accuracy in image recognition tasks. Similarly, predicting stock prices (and analoguously any signal that is changing over time) is a tedious task requiring massive data in the form of past observations. Within the broader field of AI, methods that learn patterns from data fall under the umbrella of Machine Learning (ML) (also called *statistical learning*) [Bis07]. ML has been the main engine behind the recent advances in the field, mainly thanks to the massive compute power that accompanied access to massive and diverse datasets.

Among the principal ML paradigms that drive this revolution, we mention *supervised learning* where examples of labeled pairs are shown to an algorithm in order to learn the distribution of targets given samples from an input distribution. Supervised learning constitutes the go-to framework for tasks such as classification, regression, and time series forecasting. Unlike supervised learning, Reinforcement Learning (RL) [SB18] is a paradigm where supervision doesn't stem from labeled pairs, but rather from a reward signal observed during interaction with an environment. RL is a natural way to model problems where the accomplishment of a task can be completely defined by a numerical signal: the *reward* function. Beyond ML related applications, RL can be thought off as being a learning paradigm that is closer to how humans and animals learn from their environment [Niv09; NDD]. Indeed, humans and animals don't learn from labeled pairs, but rather from the consequences of their actions, observed as the positive or negative feedback of the environment. This makes RL a very appealing framework for solving

sequential decision-making problems, where an agent has to take a series of actions in order to maximize the cumulative reward or the final outcome.

1.1 The rise and fall (and rise again) of RL

Historically, RL experienced a rise in popularity roughly a decade ago, driven by high-profile successes in domains previously thought to be beyond the reach of AI. This era was defined by the conquest of complex games, starting with deep RL agents mastering the Atari suite purely from pixels [Mni+15], and culminating in AlphaGo's victory over the world champion in the game of Go [Sil+16] . Beyond games, robotics benefited immensely from these advancements, enabling agents to learn complex manipulation and locomotion tasks through trial and error [Lev+16].

Nevertheless, the field subsequently faced a period that can be described as a *RL winter*. With the advent of the Transformer architecture [Vas+17], the research community's attention shifted toward sequence modeling problems such as Natural Language Processing (NLP). During this period, RL was legitimately criticized for its sample inefficiency and lack of reproducibility compared to the rapidly advancing language models [Hen+18].

Recently, RL has *risen from its ashes*, finding new relevance through integration with these very architectures. On one hand, RL has benefited from architectural advancements, treating decision-making as a sequence modeling problem (e.g., Decision Transformers [Che+21], Vision-Language-Action (VLA) models [Zit+23; Int+25a]). On the other, it has found a critical new application as a fine-tuning mechanism for LLMs. Techniques such as Reinforcement Learning from Human Feedback (RLHF) [Ouy+22] and Reinforcement Learning from Verifiable Rewards (RLVR) [Dee+25] have become standard for aligning foundation models, proving that RL remains an indispensable component of modern AI.

1.2 The role of MBRL

While much of the early success in deep RL relied on *model-free* approaches which learn a policy directly from interaction, Model-based Reinforcement Learning (MBRL) [SB18] offers a distinct paradigm. MBRL is defined by the agent's ability to learn and use a transition dynamics model (also called *world model*) to plan into the future. This dynamics model can be seen as an internal approximation of how the state evolves given a certain action.

The MBRL approach addresses a critical limitation in real-world deployment: the unrealistic assumption that an RL agent can access a perfect, fast, and cheap simulator. Indeed, in many physical applications interactions are costly, dangerous, or slow. By learning a model of the world, MBRL achieves significantly higher sample efficiency compared to model-free counterparts, as the agent can "*dream*" or simulate thousands of trajectories without interacting with the real environment [HS18b; Haf+21a]. Consequently, MBRL has found applications in robotics and industrial control systems, where minimizing real-world trials is key.

1.3 The era of Foundation Models

Currently, we are witnessing a paradigm shift toward FMs, which can be defined as large-scale, pre-trained models that can be adapted to a wide variety of downstream tasks. Moving away from the "one model per task" approach, this era is characterized by generalist systems capable of zero-shot or few-shot generalization.

While this revolution began in NLP [Rad+19; Bro+20], it has rapidly expanded to other modalities. In computer vision, foundation models now handle segmentation and recognition with unprecedented robustness [Car+26]; in time series forecasting, large pre-trained models are beginning to outperform traditional statistical methods on unseen data [Ans+25].

The relationship between RL and foundation models is becoming increasingly symbiotic: foundation models provide broad, diverse priors that can be leveraged to enhance RL agents, while RL provides the mechanism to optimize these models for specific behaviors. The goal of this PhD thesis is to explore this intersection, focusing on how MBRL can benefit from the integration of foundation models.

1.4 Outline and Contributions of the Thesis

This thesis investigates how recent advances in foundation models can be leveraged to address long-standing challenges in model-based reinforcement learning. Across its chapters, the thesis progressively builds from improving the training objectives of learned dynamics models, to integrating large pre-trained sequence models into the MBRL pipeline, and finally to broadening the scope of reinforcement learning through implicit reward learning. The unifying theme is the design of learning and adaptation mechanisms that improve generalization, sample efficiency, and applicability of reinforcement learning methods in realistic settings.

More specifically, the thesis is structured around a set of interrelated research questions, each addressed by a dedicated chapter as detailed in the following:

- Chapter 2 defines the notions, concepts, and algorithms at play in the rest of the document. This spans RL, MBRL, time series, and FM adaptation methods.
- Chapter 3 addresses the research question of **how to perform accurate long-horizon predictions with learned dynamics models?** To tackle this question, we study the impact of augmenting traditional MBRL with a multi-step loss function that directly optimizes for long-term predictive accuracy. The research question treated in this chapter is related to the *objective mismatch* problem in MBRL, where the dynamics model is trained to minimize one-step prediction error, while being used for policy optimization in reality. To derive intuitions about the impact of this multi-step loss, we analyze its closed-form solutions in a controlled linear setting, along with the bias and variance of their respective estimators. Furthermore, we conduct extensive experiments to evaluate different weighting schemes for the multi-step loss, especially in a noisy setting. The conclusions of this chapter include the efficiency of the multi-step loss in improving long-term predictions, all while suffering from the objective mismatch when evaluating the final return.
- Chapter 4 kicks off the integration of FMs and MBRL, specifically addressing the question **how to leverage LLMs for dynamics modelling in the context of MBRL?** The contributions made in this chapter include the formalization of the dynamics modeling problem as an in-context learning problem, therefore enabling the exploitation of emerging capabilities of LLMs. Within this formalization, two challenges that are specific to MBRL data emerge: 1) Dealing with multidimensional state spaces, and 2) integrating the control signal (actions) with state trajectories in the context. We propose an adaptation approach destined to solving both challenges: The state-action trajectories are first projected into a latent space of linearly uncorrelated features, prior to being processed in a zero-shot way by a pre-trained LLM. The resulting dynamics modeling algorithm, is analyzed theoretically in terms of return gap it induces, and practically integrated with popular RL algorithms to improve their sample efficiency. Finally, we establish the properties of the proposed LLM-based dyanmics model, which include better prediction quality with longer context sizes, and quantile calibration.
- Chapter 5 follows-up on FMs integration, extending the adaptation idea from Chapter 4 to more general families of neural network-based adapters in the context of time series forecasting. It specifically tackles the question **how to optimally adapt time series foundation models for multivariate probabilistic forecasting?** In this chapter, we start from a common architectural design found in many TSFM: to avoid dealing with size-changing inputs, TSFMs are trained on diverse datasets of *univariate* time series. Consequently, these TSFMs are called on each dimension

independently when dealing with multivariate data at inference time. To address this issue, we propose an adaptation approach based on learnable *encoder-decoder* modules that can be plugged in any TSFM to capture feature dependencies in a multivariate setting. Furthermore, inspired by partially stochastic Bayesian neural networks, we extend these adapters to enable probabilistic forecasting for natively deterministic TSFMs. This is done by inserting stochasticity in the encoder part of the adapter, allowing probabilistic forecasting through sampling in the adapter's latent space. We then theoretically analyze linear adapters, and show that in practice, our approach improves forecasting accuracy compared to the vanilla application of TSFMs.

- Chapter 6 shifts the focus from dynamics models to reward models in the context of LLM fine-tuning. Indeed, RL fine-tuning methods have witnessed incredible success in reasoning tasks. Nevertheless, they remain restricted to verifiable tasks. Therefore, in this chapter we question **how can we learn implicit reward functions from supervised data to broaden the application scope of RL?** The contributions made in this chapter include the formulation of maximum likelihood estimation as a bilevel optimizmation problem, where the outer level maximizes the likelihood over reward functions, and the inner levels constitues a typical RL objective. In terms of reward parametrization, we propose a family of reward functions based on a negative Mahalanobis distance in an embedding space. We theoretically analyse this parametrization in the case of a Gaussian linear model, providing a closed form solution to the aforementioned bilevel problem, along with its interpretation. Furthermore, we consider implicit differentiation-based solvers to provide more general solutions, validating the intuitions derived from the theoretical analysis. Practically, we validate our approach in an LLM finetuning experiment, where we show that we are able to achieve the same results by substitiuting the true reward in a reasoning task with the reward derived from our framework. To close the loop with MBRL, we further attempt our approach in a dynamic modeling setting, where we find that models pretrained with RL and our reward functions achieve comparable or better next state prediction accuracy.

Taken together, the chapters of this thesis attempt to answer the broader research question: **how can foundation models and principled learning objectives be leveraged to improve the modeling, planning, and optimization components of MBRL, leading to more sample-efficient and generalizable decision-making systems?**

Publications

The contributions presented in this thesis were done in collaboration with colleagues, and have been presented in the following list of preprints and peer-reviewed publica-

tions:

Chapter 3 is based on the following workshop paper:

- **Benechehab, A.**, Thomas, A., Paolo, G., Filippone, M., & Kégl, B. "A Study of the Weighted Multi-step Loss Impact on the Predictive Error and the Return in MBRL". In *I Can't Believe It's Not Better Workshop: Failure Modes of Sequential Decision-Making in Practice* (RLC 2024).

which is an extension of the following preprint:

- **Benechehab, A.**, Paolo, G., Thomas, A., Filippone, M., & Kégl, B. "Multi-timestep models for Model-based Reinforcement Learning". *ArXiv* (2023).

Chapter 4 is based on the following conference paper:

- **Benechehab, A.**, El Hili, Y. A., Odonnat, A., Zekri, O., Thomas, A., Paolo, G., Redko, I., Filippone, M., & Kégl, B. "Zero-shot Model-based Reinforcement Learning using Large Language Models". In *The Thirteenth International Conference on Learning Representations* (ICLR 2025).

Chapter 5 is based on the following conference paper:

- **Benechehab, A.**, Feofanov, V., Paolo, G., Thomas, A., Filippone, M., & Kégl, B. "AdaPTS: Adapting Univariate Foundation Models to Probabilistic Multivariate Time Series Forecasting". In *Forty-second International Conference on Machine Learning* (ICML 2025).

Chapter 6 is based on the following under review preprint:

- **Benechehab, A.**, Singer, G., Léger, C., Hili, Y. A. E., Paolo, G., Thomas, A., Filippone, M., & Kégl, B. "Bridging SFT and RL: Learning implicit rewards from supervised data". (*Under review*) (2026).

which is an extension of the following workshop paper:

- **Benechehab, A.**, Hili, Y. A. E., Thomas, A., Paolo, G., & Filippone, M. "Embedding Distance as a Reward Signal can replace Verifiers for LLM Reasoning". In *Workshop on Logical Reasoning of Large Language Models* (ICLR 2026).

Publications not included in the thesis

The following publications were made during the course of this PhD, but are not included in this thesis:

- **Benechehab, A.**, Thomas, A., & Kégl, B. "Deep autoregressive density nets vs neural ensembles for model-based offline reinforcement learning". *ArXiv* (2023).
- Thomas, A., **Benechehab, A.**, Paolo, G., & Kégl, B. "Fair Model-Based Reinforcement Learning Comparisons with Explicit and Consistent Update Frequency". *The Third Blogpost Track at ICLR* (2024).
- Zekri, O., **Benechehab, A.**, & Redko, I. "Can LLMs predict the convergence of Stochastic Gradient Descent?". *ICML Workshop on In-Context Learning* (2024).
- Odonnat, A. & Zekri, O., **Benechehab, A.**, Bleistein, L., Boullé, N., & Redko, I. "Large language models as markov chains." (*Under review*) (2024).
- Paolo, G., **Benechehab, A.**, Cherkaoui, H., Thomas, A., & Kégl, B. "Tag: A decentralized framework for multi-agent hierarchical reinforcement learning." *ArXiv* (2025).
- El Hili, Y. A., Thomas, A., **Benechehab, A.**, Léger, C., Ancourt, C., & Kégl, B. "In-Context Meta-Learning with Large Language Models for Automated Model and Hyperparameter Selection." *NeurIPS Workshop LLM-eval* (2025).

Open-source Code Contributions

In addition to the publications listed above, the contributions of this thesis are accompanied by publicly available open-source implementations. These repositories were developed to ensure reproducibility, facilitate adoption by the research community, and provide modular research frameworks for further extensions.

- **multi_step_mbrl** (`https://github.com/abenechehab/multi_step_mbrl`): Official implementation of the multi-timestep modeling approach for MBRL presented in Chapter 3. The repository provides experimental pipelines, weighted multi-step loss implementations, and evaluation scripts for studying predictive error and return trade-offs.
- **dicl** (`https://github.com/abenechehab/dicl`): Official codebase for the framework introduced in Chapter 4. It implements zero-shot MBRL using LLMs, including modular interfaces for environment interaction, state-action trajectories discretization, and in-context dynamics modeling.
- **AdaPTS** (`https://github.com/abenechehab/AdaPTS`): Implementation of the AdaPTS method described in Chapter 5. The repository provides adapters for univariate time-series foundation models, probabilistic forecasting utilities, and benchmarking scripts for multivariate time-series evaluation.
- **rlft4rl** (`https://github.com/abenechehab/rlft4rl`): Research framework developed during this PhD for fine-tuning LLMs to act as controllers/policies for RL

environments.

- **nll_to_po** (https://github.com/abenechehab/nll_to_po): Official implementation accompanying Chapter 6. The repository provides code for studying maximum likelihood estimation from a bilevel optimization perspective, including experimental scripts for synthetic data generation, tabular classification and regression, and LLM fine-tuning.

Chapter 2

Background & Related work

Contents

In this chapter, we establish the foundational ideas and background material that will underpin the entire thesis. We provide concise introductions to key concepts, including RL, MBRL, and FMs. In most chapters of this thesis, we include a deeper discussion of the related literature relevant to that material.

2.1 Reinforcement Learning

Unlike classical learning paradigms where the model learns from a fixed dataset, RL [SB18] focuses on learning optimal behaviors through interactions with an environment. Although this approach remains data-driven at its core, it emphasizes *sequential decision making* and *long-term* outcomes rather than immediate predictions. On a high-level, the core components of a RL system include an agent, an environment, and a reward signal that reflects the accomplishment of a given task.

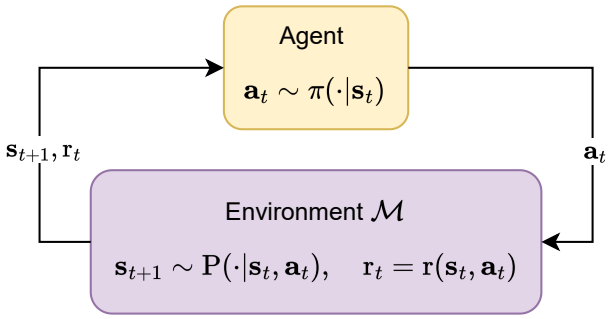


Figure 2.1: The Reinforcement Learning loop.

As depicted in Figure 2.1, the interaction loop between an agent and an environment is as follows: at each discrete time step t, the agent selects an action $\mathbf{a}_t$, applies it to the environment, and observes the corresponding reward r_t and next state $\mathbf{s}_{t+1}$ where the state is a representation of the environment. The space of all possible environement states $\mathbf{s}$ is denoted as the state space $\mathcal{S}$, while the space of all possible actions is denoted as the action space $\mathcal{A}$. The *learning* in RL involves the policy improving the quality of the selected actions through experience, with the goal of maximizing cumulative rewards over time. Meanwhile, the *reinforcement* aspect refers to the feedback mechanism provided by the reward signal, which reinforces desirable behaviors and discourages undesirable ones.

Formally, we define an environment as a first-order Markov Decision Process (MDP) (i.e., it satisfies the Markov property $\mathrm{P}(\mathbf{s}_{t+1}|\mathbf{s}_t, \ldots, \mathbf{s}_0) = \mathrm{P}(\mathbf{s}_{t+1}|\mathbf{s}_t)$) $\mathcal{M}$ [Bel57a], a policy π as a mapping from states to actions (precisely, a stochastic policy; we can also define a deterministic policiy as a function $\pi : \mathcal{S} \to \mathcal{A}$), and the learning objective of RL as maximizing the expected return $\eta[\pi]$:

Definition 2.1.1 (Markov Decision Process)**.**

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mu_0, \gamma, \mathrm{P}, \mathrm{r})$$

where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, μ_0 the initial state distribution, $\gamma \in [0, 1)$

the discount factor, $\mathrm{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ *the Markovian transition probability where* $\Delta(\mathcal{S})$ *denotes distributions over the state space* $\mathcal{S}$*, and* $\mathrm{r} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ *the reward function.*

Definition 2.1.2 (Policy)**.**

$$\pi : \mathcal{S} \to \Delta(\mathcal{A})$$

where $\Delta(\mathcal{A})$ *denotes distributions over the action space* $\mathcal{A}$*.*

Definition 2.1.3 (Return)**.**

$$\eta[\pi] = \mathbb{E}_{\mu_0,\pi,\mathrm{P}}\left[\sum_{t=0}^{\infty} \gamma^t \mathrm{r}(\mathbf{s}_t, \mathbf{a}_t)\right]$$

The return as defined in definition 2.1.3, is computed by following the environment dynamics P and the policy π for all $t \geq 0$ up to an infinite horizon, with the discount factor γ playing the role of prioritizing short-term vs long-term rewards. Additionally, the discount factor guarantees that the infinite-horizon return is well-defined: for $\gamma \in [0, 1)$, the discounted rewards form a geometrically weighted series, which converges and thus yields a finite return under bounded rewards. For infinite horizons, the discount factor γ implicitly defines the effective horizon as $\mathrm{H}_{\text{eff}} \approx \frac{1}{1-\gamma}$. In practice, an instance of agent-environment interactions (called *episode*) runs up to a pre-defined MDP horizon H, or when certain termination conditions are met.

The first step of solving a problem using RL typically involves modelling the problem as a MDP by defining its elements as stated in definition 2.1.1. Examples of MDPs include robotic control [Lil+16; Lev+16], game playing [Mni+15; Sil+16], and language modeling [Chr+17; Ouy+22]. Table 2.1 summarizes the mapping between these real-world problems and their respective MDP components.

Anatomy of RL algorithms. An RL algorithm can be typically defined as an optimization procedure that aims at finding policies that maximize the expected return $\eta[\pi]$. Directly optimizing the expected return is difficult for many reasons. First of all the return depends on full trajectories, and the expectation is taken over a distribution that involves all future states and actions. This means that the trajectory-level return offers no local information about which actions were actually good. Secondly, this return tends to have extremely high variance when estimated from samples. For these reasons, we find ourselves faced with the question of *How do we measure which states or actions are good so we can improve the policy?* To answer this question, we mention *Value functions* and *Q-functions*, which represent measurements of the expected sum of rewards respectively, from a state $\mathbf{s}$ and taking an action $\mathbf{a}$ from a state $\mathbf{s}$, both under the policy π:

Environment	$\mathcal{S}$ (states)	$\mathcal{A}$ (actions)	r (reward)	P (transition)	H (horizon)
 Robotic control	$\mathbb{R}^{\mathrm{d}}$ positions, velocities	$\mathbb{R}^{\mathrm{d}}$ motor torques	$\|\dot{x}_t\|, \mathbf{s}_t = \mathbf{s}_{\text{goal}}$ running, goal-reaching	$\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t)$ physics-based simulator f	Finite episodic
 Games	$\mathbb{R}^{\mathrm{h}\times\mathrm{w}\times\mathrm{c}}$ images, game stats	Discrete joystick	+1 for win, -1 for loss, game score	$\mathbf{s}_{t+1} = f(\mathbf{s}_t, \mathbf{a}_t)$ game engine f	Finite episodic
 Language	$\mathbb{R}^{\mathrm{T}\times\mathrm{d}}$ token sequences	Discrete $\in$ vocabulary	Verifiable tasks mathematics, coding	$\mathbf{s}_{t+1} = \mathrm{cat}(\mathbf{s}_t, \mathbf{a}_t)$ concatenation	Until end of sentence

Table 2.1: Mapping examples of real-world problems to their corresponding MDP components. A *token* represents a unit of language, and $\mathrm{d}, \mathrm{h}, \mathrm{w}, \mathrm{c}$ are integers representing arbitrary dimensions. $\|\dot{x}_t\|$ represents the velocity along an arbitrary axis x.

Definition 2.1.4 (Value function)**.**

$$V^\pi(\mathbf{s}) = \mathbb{E}_{\pi,\mathrm{P}}\left[\sum_{t=0}^{\infty} \gamma^t \mathrm{r}(\mathbf{s}_t, \mathbf{a}_t) \mid \mathbf{s}_0 = \mathbf{s}\right]$$

Definition 2.1.5 (Q-function)**.**

$$Q^\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_{\pi,\mathrm{P}}\left[\sum_{t=0}^{\infty} \gamma^t \mathrm{r}(\mathbf{s}_t, \mathbf{a}_t) \mid \mathbf{s}_0 = \mathbf{s}, \mathbf{a}_0 = \mathbf{a}\right]$$

These functions satisfy a recursive relationship known as the *Bellman equation* [Bel57b], which expresses the value of a state (or state-action pair) in terms of the immediate reward and the value of subsequent states and actions. Below is the Bellman equation for the Q-function, which will be a central element in many RL algorithms we discuss later.

$$Q^\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_{\mathbf{s}'\sim\mathrm{P}(\cdot|\mathbf{s},\mathbf{a})}\Big[\mathrm{r}(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{a}'\sim\pi(\cdot|\mathbf{s}')}[Q^\pi(\mathbf{s}', \mathbf{a}')]\Big]$$

Similarly, one can write the Bellman equation for the value function, and relate it to the Q-function by taking an expectation with respect to the initial action:

$$\begin{aligned} V^\pi(\mathbf{s}) &= \mathbb{E}_{\mathbf{a}\sim\pi(\cdot|\mathbf{s})}[Q^\pi(\mathbf{s}, \mathbf{a})] \\ &= \mathbb{E}_{\mathbf{a}\sim\pi(\cdot|\mathbf{s}),\mathbf{s}'\sim\mathrm{P}(\cdot|\mathbf{s},\mathbf{a})}\Big[\mathrm{r}(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{a}'\sim\pi(\cdot|\mathbf{s}')}[Q^\pi(\mathbf{s}', \mathbf{a}')]\Big] \\ &= \mathbb{E}_{\mathbf{a}\sim\pi(\cdot|\mathbf{s}),\mathbf{s}'\sim\mathrm{P}(\cdot|\mathbf{s},\mathbf{a})}[\mathrm{r}(\mathbf{s}, \mathbf{a}) + \gamma V^\pi(\mathbf{s}')] \end{aligned}$$

It's important to highlight the connection between the value of current states and actions and that of subsequent states and actions, a relationship made possible by the Bellman equation. Indeed, this recursive formulation enables the use of *Dynamic Programming* (DP) [Bel57b] to propagate values across visited states, thereby refining estimates of Q-functions through iteration. Before delving into how accurate estimates of value

functions and Q-functions can be used to improve policies, let's first caracterize optimal policies, the object at the core of all RL algorithms.

Let's denote Π the set of all possible policies. An optimal policy $\pi^\star \in \Pi$ is defined as a policy that achieves the highest expected return among all policies: $\eta[\pi^\star] \geq \eta[\pi], \forall \pi \in \Pi$. Regarding the existence of optimal policies, it can be shown that at least one optimal policy always exists for any MDP with finite state-action spaces [Put94]. Notably, there can be multiple optimal policies that achieve the same maximum expected return. Moreover, all optimal policies share the same optimal value function $V^\star$ and optimal Q-function $Q^\star$. The latter satisfies a special case of the Bellman equation, known as the *Bellman optimality equation*, which characterizes the relationship between the optimal Q-values of states and actions:

$$Q^\star(\mathbf{s}, \mathbf{a}) = \mathbb{E}_{\mathbf{s}' \sim \mathrm{P}(\cdot|\mathbf{s},\mathbf{a})}\Big[\mathrm{r}(\mathbf{s}, \mathbf{a}) + \gamma \max_{\mathbf{a}'} Q^\star(\mathbf{s}', \mathbf{a}')\Big]$$

The introduced Bellman equations naturally lead to the definition of the *Bellman operator* $\mathcal{T}^\pi$ and the *Bellman optimality operator* $\mathcal{T}^\star$ as in definition 2.1.6. These operators are a key stone in the definition of many RL algorithms, as it will be covered in the next sections of this chapter.

A key property of these operators is that they are γ-*contractions* (that is, $\forall Q, Q',\ \|\mathcal{T}[Q] - \mathcal{T}[Q']\|_\infty \leq \gamma \|Q - Q'\|_\infty$, where $\|\cdot\|_\infty$ denotes the supremum norm), a feature that ensures the existence and uniqueness of their respective fixed points by virtue of the Banach fixed-point theorem [Ban22].

Definition 2.1.6 (Bellman Operator)**.**

$$\mathcal{T}^\pi[Q](\mathbf{s}, \mathbf{a}) = \mathbb{E}_{\mathbf{s}' \sim \mathrm{P}(\cdot|\mathbf{s},\mathbf{a})}\Big[\mathrm{r}(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{a}' \sim \pi(\cdot|\mathbf{s}')}[Q(\mathbf{s}', \mathbf{a}')]\Big] \qquad (\textit{classical})$$

$$\mathcal{T}^\star[Q](\mathbf{s}, \mathbf{a}) = \mathbb{E}_{\mathbf{s}' \sim \mathrm{P}(\cdot|\mathbf{s},\mathbf{a})}\Big[\mathrm{r}(\mathbf{s}, \mathbf{a}) + \gamma \max_{\mathbf{a}'} Q(\mathbf{s}', \mathbf{a}')\Big] \qquad (\textit{optimal})$$

for any function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$*, policy* $\pi \in \Pi$*, and state-action pair* $(\mathbf{s}, \mathbf{a}) \in \mathcal{S} \times \mathcal{A}$*.*

A key property of these operators is that they are γ-*contractions*[1], a feature that ensures the existence and uniqueness of their respective fixed points by virtue of the Banach fixed-point theorem [Ban22]. This means that sequences obtained by iteratively applying these operators: $\forall Q_0, \quad Q_{n+1} = \mathcal{T}^\pi[Q_n]$ (respectively $Q_{n+1} = \mathcal{T}^\star[Q_n]$) will converge to the corresponding Q-function Q^π (respectively $Q^\star$). Using these tools, we can now describe how different RL algorithms approach the problem of finding optimal policies.

[1] That is, $\forall Q, Q',\ \|\mathcal{T}[Q] - \mathcal{T}[Q']\|_\infty \leq \gamma \|Q - Q'\|_\infty$, where $\|\cdot\|_\infty$ denotes the supremum norm.

2.1.1 Dynamic Programming

DP methods [Bel57b] are a class of algorithms that directly leverage the Bellman equations to iteratively improve estimates of value functions and policies. These methods rely on the assumption that the full model of the environment (i.e., transition probabilities P and reward function r at all states and actions) is known to the agent. To ensure that such an assumption holds, DP methods are restricted to the class of MDPs with finite state and action spaces, the so called *Tabular* setting. Below we show the pseudo-code of two fundamental DP algorithms: *Q-Value Iteration* (VI) [Bel57b] and *Policy Iteration* (PI) [How60]. These algorithms enjoy theoretical guarantees of convergence to the optimal policy $\pi^\star$ thanks to the aforementioned Bellman operators' properties.

Algorithm 1 Q-Value Iteration

Input: MDP $(\mathcal{S}, \mathcal{A}, \mu_0, \gamma, \mathrm{P}, \mathrm{r})$, Q_0
for $k = 0, 1, 2, \ldots$ until convergence **do**
  for $(\mathbf{s}, \mathbf{a}) \in \mathcal{S} \times \mathcal{A}$ **do**
    $Q_{k+1}(\mathbf{s}, \mathbf{a}) = \mathcal{T}^\star[Q_k](\mathbf{s}, \mathbf{a})$
  end for
end for
$\pi^*(\mathbf{s}) = \arg\max_{\mathbf{a}} Q_k(\mathbf{s}, \mathbf{a}),\ \forall \mathbf{s}$
return π^*

Algorithm 2 Policy Iteration

Input: MDP $(\mathcal{S}, \mathcal{A}, \mu_0, \gamma, \mathrm{P}, \mathrm{r})$, π_0, Q_0
for $n = 0, 1, 2, \ldots$ until convergence **do**
  for $k = 0, 1, 2, \ldots$ until convergence **do**
    $Q_{k+1}(\mathbf{s}, \mathbf{a}) = \mathcal{T}^{\pi_n}[Q_k](\mathbf{s}, \mathbf{a}),\ \forall (\mathbf{s}, \mathbf{a})$
  end for
  $\pi_{n+1}(\mathbf{s}) = \arg\max_{\mathbf{a}} Q_k(\mathbf{s}, \mathbf{a}),\ \forall \mathbf{s}$
end for
return π_n

As it can be seen in Algorithm 1, the Q-Value Iteration algorithm iteratively applies the Bellman optimality operator $\mathcal{T}^\star$ to update the Q-function estimates until convergence. In practice, we assume convergence when the maximum change in Q-values across all state-action pairs falls below a predefined threshold. The optimal policy is then derived as the corresponding *greedy* policy, selecting the action that maximizes the Q-value for each state. On the other hand, Algorithm 2 describes the Policy Iteration algorithm, which alternates between *policy evaluation* (iterating the Bellman operator $\mathcal{T}^{\pi_n}$) and *policy improvement* (greedy policy) steps until convergence. Next we are going to relax the assumption of knowing the full environment model, and introduce RL algorithms that learn optimal policies through direct interactions with the environment.

2.1.2 Value-based methods

In more realistic settings, agents only have access to realizations of the transition probability P and reward function r corresponding to states and actions visited during past interactions. Value-based methods aim at learning optimal value functions or Q-functions solely from these interactions. In this type of algorithms, we distinguish between two

major phases: i) *acting* where the agent chooses actions to interact with the environment, and ii) *learning* where the agent updates its value function or Q-function estimates based on the collected experience. One of the most well-known value-based algorithms is *Q-Learning* [Wat89; WD92], which uses the Bellman optimality equation to update its Q-function estimates. For a transition $(\mathbf{s}_t, \mathbf{a}_t, \mathrm{r}_t, \mathbf{s}_{t+1})$ the Q-Learning update rule using the so-called *Temporal Difference* (*TD*) learning can be expressed as follows:

$$Q(\mathbf{s}_t, \mathbf{a}_t) \leftarrow Q(\mathbf{s}_t, \mathbf{a}_t) + \alpha \left(\mathrm{r}_t + \gamma \max_{\mathbf{a}} Q(\mathbf{s}_{t+1}, \mathbf{a}) - Q(\mathbf{s}_t, \mathbf{a}_t) \right),$$

where α is a learning rate that controls the step size of the update.

On-policy vs off-policy RL. To compute the TD target, Q-Learning uses the maximum Q-value over all possible actions from the next state $\mathbf{s}_{t+1}$, as shown in the green line of Algorithm 4. However, the action that realises this maximum $(\arg\max_{\mathbf{a}} Q(\mathbf{s}_{t+1}, \mathbf{a}))$ do not necessarily correspond to the action $\mathbf{a}_{t+1}$ used to interact with the environment. The fact of learning the optimal policy independently of the actions taken by the current policy is referred to as *off-policy* RL, which is the case for Q-learning (Algorithm 4). An alternative algorithm called *SARSA* (State-Action-Reward-State-Action) [RN94] uses the Q-value of the action $\mathbf{a}_{t+1}$ actually taken in the next state $\mathbf{s}_{t+1}$, as shown in the orange line of Algorithm 4. The latter approach is referred to as *on-policy* RL.

In the tabular setting, and similarly to the policy improvement phase in PI (Algorithm 2), the acting policy can be chosen as the greedy policy with respect to the current estimate of the Q-function. However, because the initial Q-estimates are inaccurate, it is standard practice to use exploration strategies (e.g., ϵ-greedy[2] as in Algorithm 3) to encourage the agent to sample diverse actions and states early on.

Algorithm 3 ϵ-greedy policy $\pi_{Q,\epsilon}$

Input: Q-function Q, exploration rate ϵ, state $\mathbf{s}_t$
if random number $\in [0, 1] < \epsilon$ **then**
  $\mathbf{a}_t \sim \mathcal{U}(\mathcal{A})$ ▷ uniform distribution
else
  $\mathbf{a}_t = \arg\max_{\mathbf{a}} Q(\mathbf{s}_t, \mathbf{a})$ ▷ greedy policy
end if
return $\mathbf{a}_t$

Analoguous to the difference between VI and PI, Q-Learning's update rule is based on the Bellman optimality operator $\mathcal{T}^\star$, whereas SARSA's is based on the Bellman operator

[2]To ensure convergence to the optimal policy, the exploration rate ϵ must decay to 0 over time so that $Q^{\pi_{Q,\epsilon}}$ converges to $Q^\star$ asymptotically.

Algorithm 4 Q-Learning (off-policy) / SARSA (on-policy)

Input: MDP $(\mathcal{S}, \mathcal{A}, \mu_0, \gamma)$, learning rate α, exploration rate ϵ, initial Q-function Q
for episode = 0, 1, 2, ... **do**
  Initialize $\mathbf{s}_0 \sim \mu_0$, $\mathbf{a}_0 \sim \pi_{Q,\epsilon}(\cdot|\mathbf{s}_0)$
  for t = 0, 1, 2, ... until episode ends **do**
    $\mathbf{s}_{t+1}, \mathrm{r}_t \leftarrow \mathrm{P}(\cdot|\mathbf{s}_t, \mathbf{a}_t), \mathrm{r}(\mathbf{s}_t, \mathbf{a}_t)$ ▷ agent-environment interaction
    $Q(\mathbf{s}_t, \mathbf{a}_t) \leftarrow Q(\mathbf{s}_t, \mathbf{a}_t) + \alpha \underbrace{\left(\mathrm{r}_t + \gamma \max_{\mathbf{a}} Q(\mathbf{s}_{t+1}, \mathbf{a}) - Q(\mathbf{s}_t, \mathbf{a}_t)\right)}_{\text{TD with } \mathcal{T}^\star}$
    $\mathbf{a}_{t+1} \sim \pi_{Q,\epsilon}(\cdot|\mathbf{s}_{t+1})$ ▷ ϵ-greedy policy
    $Q(\mathbf{s}_t, \mathbf{a}_t) \leftarrow Q(\mathbf{s}_t, \mathbf{a}_t) + \alpha \underbrace{(\mathrm{r}_t + \gamma Q(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) - Q(\mathbf{s}_t, \mathbf{a}_t))}_{\text{TD with } \mathcal{T}^\pi}$
  end for
end for
return $\pi_{Q,\epsilon=0}$

$\mathcal{T}^\pi$. However, unlike DP methods, the expectations in these operators are not computed exactly but are instead approximated using the single transition $(\mathbf{s}_t, \mathbf{a}_t, \mathrm{r}_t, \mathbf{s}_{t+1}, \mathbf{a}_{t+1})$ observed at each timestep t.

One major drawback of the Q-Learning algorithm (and tabular methods in general) is the fact that it stores the Q-function as a table, which becomes impractical for large or continuous state spaces. To address this limitation, function approximation techniques can be used to cast learning the Q-function as a *regression* problem based on samples collected from the environment. The learned regression model can then generalize, and provide estimated values that can be used to select actions in previously unseen states. In this setting, the agent-environment interactions are typically stored in a *replay buffer* $\mathcal{D} = \{(\mathbf{s}_i, \mathbf{a}_i, \mathrm{r}_i, \mathbf{s}'_i)\}_{i=1}^N$ that contains tuples of states, actions, rewards, and next states. The Q-function parameters (denoted θ) are then updated by gradient descent in the direction minimizing the Mean Squared Error (MSE) to the TD targets. Algorithm 5 shows the variant when the function approximator is a neural network, resulting in the well-known *Deep Q-Network* (DQN) [Mni+15] algorithm, famous for achieving human-like capabilities in the Atari benchmark [Bel+13].

The two main ingredients that contributed to the success of DQN are: i) *experience replay* through the use of a replay buffer $\mathcal{D}$ to store and reuse past experiences, and ii) the use of a *target network* for which the parameters (copy(θ)) update frequency is reduced relative to that of the current parameters θ. More recently, other DQN variants have been proposed to further improve performance and stability. This includes but is not limited to Double DQN [HGS16], Dueling DQN [Wan+16], Prioritized replay [Sch+16a],

Algorithm 5 Deep Q-Network (DQN)

Input: MDP $(\mathcal{S}, \mathcal{A}, \mu_0, \gamma)$, learning rate α, exploration rate ϵ, replay buffer $\mathcal{D}$, initial Q-network parameters θ
for episode = 0, 1, 2, ... **do**
  Initialize state $\mathbf{s}_0 \sim \mu_0$
  for t = 0, 1, 2, ... until episode ends **do**
    $\mathbf{a}_t \sim \pi_{Q_\theta,\epsilon}(\cdot|\mathbf{s}_t)$ ▷ ϵ-greedy policy
    $\mathbf{s}_{t+1}, \mathrm{r}_t \leftarrow \mathrm{P}(\cdot|\mathbf{s}_t, \mathbf{a}_t), \mathrm{r}(\mathbf{s}_t, \mathbf{a}_t)$ ▷ agent-environment interaction
    Store $(\mathbf{s}_t, \mathbf{a}_t, \mathrm{r}_t, \mathbf{s}_{t+1})$ in $\mathcal{D}$
    Sample mini-batch $\{(\mathbf{s}_i, \mathbf{a}_i, \mathrm{r}_i, \mathbf{s}'_i)\}_{i=1}^B$ from $\mathcal{D}$ ▷ batch size B
    $\mathcal{L}(\theta) = \frac{1}{B}\sum_{i=1}^B \left(\underbrace{\mathrm{r}_i + \gamma \max_{\mathbf{a}'} Q_{\text{copy}(\theta)}(\mathbf{s}'_i, \mathbf{a}')}_{\text{TD target}} - \underbrace{Q_\theta(\mathbf{s}_i, \mathbf{a}_i)}_{\text{prediction}} \right)^2$ ▷ loss function
    $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta)$ ▷ gradient descent step
    if targte network update frequency **then**
      copy$(\theta) \leftarrow \theta$ ▷ target network update
    end if
  end for
end for
return $\pi_{Q_\theta,\epsilon=0}$

Multi-step targets [Sut88], and Distributional RL [BDM17]. The combination of these techniques has led to one of the most powerful RL algorithms for discrete action spaces: Rainbow DQN [Hes+18], capable of achieving super human-level performance on the Atari benchmark [Bel+13]. Although being impactful, DQN and its variants are limited to discrete action spaces. Indeed, computing the $\max$ over all possible actions becomes intractable when the action space is continuous. To address this limitation, *Policy-based methods* have been developed to handle continuous action spaces by directly parameterizing the policy.

2.1.3 Policy-based methods

These methods focus on directly optimizing the policy π_ϕ parameterized by ϕ (e.g., using neural networks) to maximize the expected return $\eta[\pi_\phi]$. We have mentioned at the beginning of this section that directly optimizing η is challenging due to its dependence on full trajectories and the high variance of its estimates. Throughout this section, we will explore how policy-based methods tackle these challenges to effectively learn optimal policies. This is typically achieved using gradient ascent techniques, where the policy parameters are updated in the direction maximizing the expected return. For this to

happen, we need to compute the gradient of the expected return with respect to the policy parameters ϕ. Expanding the expectation and using the logarithmic trick[3], the *policy gradient theorem* [Sut+00] provides a simple way to estimate this gradient:

Theorem 2.1.1 (Policy Gradient Theorem)**.**

$$\nabla_\phi \eta[\pi_\phi] = \mathbb{E}_{\pi_\phi, \mathrm{P}}\left[\sum_{t=0}^{\infty} \underbrace{\left(\sum_{t'=t}^{\infty} \gamma^{t'-t} \mathrm{r}(\mathbf{s}_{t'}, \mathbf{a}_{t'})\right)}_{G_t} \nabla_\phi \log \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)\right]$$

Computing the Expectation in theorem 2.1.1 is generally intractable as it involves an integration over potentially infinite state and action spaces. In practice, the gradient can be estimated using Monte-Carlo sampling [MU49] by collecting trajectories from the environment using the current policy π_ϕ. This results in the well-known *REINFORCE* algorithm [Wil92b], which pseudo-code is shown in Algorithm 6.

Algorithm 6 REINFORCE

Input: MDP $(\mathcal{S}, \mathcal{A}, \mu_0, \gamma)$, learning rate α, initial policy parameters ϕ
for episode = 0, 1, 2, ... **do**
  Generate episode $(\mathbf{s}_0, \mathbf{a}_0, \mathrm{r}_0, \mathbf{s}_1, \mathbf{a}_1, ..., \mathbf{s}_T)$ using π_ϕ ▷ agent-environment interaction
  for t = 0, 1, 2, ... T **do**
    $G_t = \sum_{t'=t}^{T} \gamma^{t'-t} \mathrm{r}_{t'}$ ▷ return from time step t
    $\phi \leftarrow \phi + \alpha G_t \nabla_\phi \log \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$ ▷ policy update
  end for
end for
return π_ϕ

Practically, the return term G_t in Algorithm 6 has been shown to have high variance [Sei17], which can lead to unstable policy optimization. As a remedy, we can subtract a *baseline* function $b(\mathbf{s}_t)$ from the return, which does not introduce bias in the gradient estimate while reducing its variance [GBB04; WT01]. A common choice for the baseline is the value function $V^{\pi_\phi}(\mathbf{s}_t)$, leading to the return taking the form: $G_t = \sum_{t'=t}^{T} \gamma^{t'-t} \mathrm{r}_{t'} - V^{\pi_\phi}(\mathbf{s}_t)$. Furthermore, it has been shown that the policy gradient can be written in terms of multiple estimates of the return G_t, all being instances of the so-called Generalized

[3]The logarithmic trick refers to removing the derivation variable ϕ from the expectation term by observing that $\nabla_\phi \log \pi_\phi(\mathbf{a}_t|\mathbf{s}_t) = \frac{\nabla_\phi \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)}{\pi_\phi(\mathbf{a}_t|\mathbf{s}_t)}$.

Advantage Estimation (GAE) [Sch+16b]. For instance, we can use the Q-function estimate with a baseline in-place of the return, resulting in the so-called *Advantage function*. This leads to the following expression of the policy gradient:

$$\nabla_\phi \eta[\pi_\phi] = \mathbb{E}_{\pi_\phi, \mathrm{P}}\left[\sum_{t=0}^{\infty} \gamma^t \underbrace{\left(Q^{\pi_\phi}(\mathbf{s}_t, \mathbf{a}_t) - V^{\pi_\phi}(\mathbf{s}_t)\right)}_{\text{Advantage function: } A^{\pi_\phi}(\mathbf{s}_t, \mathbf{a}_t)} \nabla_\phi \log \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)\right]$$

With the Q-function now being part of the policy gradient expression, we can combine value-based and policy-based methods to form *Actor-Critic* algorithms. These methods combine the low-variance value estimation of value-based approaches with the direct policy optimization of policy-based methods, resulting in more stable and sample-efficient learning.

2.1.4 Actor-critic methods

These methods consist of two main components: the *actor*, which represents the policy π_ϕ, and the *critic*, which estimates the value function or Q-function Q_θ. The underlying mechanism consists in the critic evaluating the actions taken by the actor, thus providing feedback to improve the policy. The next algorithm (Algorithm 7) shows a simple actor-critic method where the critic estimates the Q-function using TD learning, and the actor updates its policy using the estimated Q-values plugged in the policy gradient expression.

Corollary 2.1.1 (Off-policy Policy Gradient)**.**

$$\nabla_\phi \eta[\pi_\phi] = \mathbb{E}_{\mu, \mathrm{P}}\left[\sum_{t=0}^{\infty} \gamma^t \frac{\pi_\phi(\mathbf{a}_t|\mathbf{s}_t)}{\mu(\mathbf{a}_t|\mathbf{s}_t)} Q^{\pi_\phi}(\mathbf{s}_t, \mathbf{a}_t) \nabla_\phi \log \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)\right]$$

In Algorithm 7, the actor and the critic are updated using either an off-policy or on-policy fashion, similar to respectively Q-Learning and SARSA. One key difference however, is the fact that the off-policyness in this case doesn't stem from taking the maximum over actions (as in Q-Learning), but rather from using transitions collected under a different (older) policy $\pi_{\phi_{\text{old}}}$. To account for this discrepancy in the off-policy setting, an *importance sampling* [KM53; PSS00; Mun+16] weight $\frac{\pi_\phi(\mathbf{a}_i|\mathbf{s}_i)}{\pi_{\phi_{\text{old}}}(\mathbf{a}_i|\mathbf{s}_i)}$ is introduced in the actor update. This importance weight comes from the derivation of the off-policy policy gradient [DWS12], which follows similar derivation steps to theorem 2.1.1. As shown in corollary 2.1.1, the only difference is the expectation being taken with respect to an arbitrary *behavior policy*[4] μ instead of the current policy π_ϕ.

[4] In Algorithm 7, the behavior policy correspond to older versions of the policy being updated: $\pi_{\phi_{\text{old}}}$.

Algorithm 7 Actor-Critic (off-policy / on-policy)

Input: MDP $(\mathcal{S}, \mathcal{A}, \mu_0, \gamma)$, learning rate α, replay buffer $\mathcal{D}$, initial policy parameters ϕ, initial Q-network parameters θ
for episode = 0, 1, 2, ... **do**
  Initialize $\mathbf{s}_0 \sim \mu_0$, $\mathbf{a}_0 \sim \pi_\phi(\cdot|\mathbf{s}_0)$
  for t = 0, 1, 2, ... until episode ends **do**
    $\mathbf{s}_{t+1}, \mathrm{r}_t \leftarrow \mathrm{P}(\cdot|\mathbf{s}_t, \mathbf{a}_t), \mathrm{r}(\mathbf{s}_t, \mathbf{a}_t)$ ▷ agent-environment interaction
    Store $(\mathbf{s}_t, \mathbf{a}_t, \mathrm{r}_t, \mathbf{s}_{t+1})$ in $\mathcal{D}$
    Sample mini-batch $\{(\mathbf{s}_i, \mathbf{a}_i, \mathrm{r}_i, \mathbf{s}'_i)\}_{i=1}^B$ from $\mathcal{D}$ ▷ batch size B
    Critic update: ▷ minimize TD error
    $\mathbf{a}'_i \sim \pi_\phi(\cdot|\mathbf{s}'_i)$ for all $i = 1, ..., B$
    $\theta \leftarrow \theta - \alpha \nabla_\theta \left(\frac{1}{B} \sum_{i=1}^B (r_i + \gamma Q_{\text{copy}(\theta)}(\mathbf{s}'_i, \mathbf{a}'_i) - Q_\theta(\mathbf{s}_i, \mathbf{a}_i))^2 \right)$
    $\mathbf{a}_{t+1} \sim \pi_\phi(\cdot|\mathbf{s}_{t+1})$ ▷ sample next action from policy
    $\theta \leftarrow \theta - \alpha \nabla_\theta \left(r_t + \gamma Q_{\text{copy}(\theta)}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) - Q_\theta(\mathbf{s}_t, \mathbf{a}_t) \right)^2$
    Actor update: ▷ policy update
    $\phi \leftarrow \phi + \alpha \left(\frac{1}{B} \sum_{i=1}^B \frac{\pi_\phi(\mathbf{a}_i|\mathbf{s}_i)}{\pi_{\phi_{\text{old}}}(\mathbf{a}_i|\mathbf{s}_i)} Q_\theta(\mathbf{s}_i, \mathbf{a}_i) \nabla_\phi \log \pi_\phi(\mathbf{a}_i|\mathbf{s}_i) \right)$ ▷ with importance weight
    $\phi \leftarrow \phi + \alpha Q_\theta(\mathbf{s}_t, \mathbf{a}_t) \nabla_\phi \log \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$
  end for
end for
return π_ϕ

The additional importance weight in corollary 2.1.1 is actually a likelihood ratio that corrects the distribution mismatch between the behavior policy μ and the target policy π_ϕ. To compute this ratio (and also the derivative term $\nabla_\phi \log \pi_\phi$) in practice, the likelihood model defined by the parametrized policy π_ϕ needs to be known and tractable. This is often the case when using stochastic policies parameterized by neural networks that learn parameters of known probability distributions, e.g. Gaussian policies for continuous action spaces: $\pi_\phi(\mathbf{a}|\mathbf{s}) = \mathcal{N}(\mathbf{a}; \mu_\phi(\mathbf{s}), \sigma_\phi(\mathbf{s})\mathrm{I})$ where $\mathcal{N}$ denotes the PDF of a normal distribution with mean $\mu_\phi(\mathbf{s})$ and covariance $\sigma_\phi(\mathbf{s})\mathrm{I}$. Practical implementations of on-policy actor-critic methods with a stochastic policy include Advantage Actor-Critic (A2C) [Mni+16] and its asynchronuous version A3C [Mni+16]. Numerous deterministic policy gradient methods have also been developed, including Deterministic Policy Gradient (DPG) [Sil+14], its deep variant DDPG [Lil+16], the distributional RL extension D4PG [Bar+18], and the overestimation bias-adjusted[5] Twin Delayed DDPG (TD3) [FHM18]. For off-policy actor-critic algorithms, ACER (Actor-Critic with Experience Replay) [Wan+17] is a well-known method that combines experience replay with im-

[5]Overestimation bias denotes the phenomenon whereby, under function approximation, Q-value estimates tend to exceed their true values within the TD learning framework.

portance sampling corrections.

Exploration vs exploitation. In RL, balancing exploration and exploitation is crucial because the agent must try new actions to discover potentially better rewards (*exploration*) while also leveraging known high-reward actions to maximize returns (*exploitation*). For instance, in the Q-Learning and DQN algorithms we use an ϵ-greedy policy to encourage exploration through the selection of uniformly random actions. In the context of actor-critic methods, exploration can be encouraged by adding an *entropy regularization* [Haa+17; Haa+18a] term to the policy objective. This regularization term encourages the policy to maintain high entropy, thus promoting exploration of different actions. The modified policy objective with entropy regularization can be expressed as follows:

Definition 2.1.7 (Max-entropy Return)**.**

$$\eta[\pi] = \mathbb{E}_{\mu_0,\pi,\mathrm{P}}\left[\sum_{t=0}^{\infty} \gamma^t \mathrm{r}(\mathbf{s}_t, \mathbf{a}_t) + \lambda \mathcal{H}(\pi(\cdot|\mathbf{s}_t))\right],$$

where $\mathcal{H}(\pi(\cdot|\mathbf{s})) = -\mathbb{E}_{\mathbf{a}\sim\pi(\cdot|\mathbf{s})}[\log \pi(\mathbf{a}|\mathbf{s})]$ *is the entropy of the policy* π *at state* $\mathbf{s}$*, and* $\lambda \geq 0$ *a temperature parameter.*

Analoguous to the steps we took to define the Q-function and the classical Bellman operator from the return in definition 2.1.3, we can derive the *soft Bellman operator* [Haa+17] corresponding to the max-entropy return:

$$\mathcal{T}^{\pi}_{\text{soft}}[Q](\mathbf{s}, \mathbf{a}) = \mathbb{E}_{\mathbf{s}'\sim\mathrm{P}(\cdot|\mathbf{s},\mathbf{a})}\Big[\mathrm{r}(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{a}'\sim\pi(\cdot|\mathbf{s}')}[Q(\mathbf{s}', \mathbf{a}') - \lambda \log \pi(\mathbf{a}'|\mathbf{s}')]\Big].$$

Using $\mathcal{T}^{\pi}_{\text{soft}}$ to perform policy evaluation in actor-critic methods leads to the Soft Actor-Critic (SAC) [Haa+18a] algorithm, which was shown to obtain state-of-the-art performance on various continuous control benchmarks. Algorithm 8 shows the pseudo-code of the SAC algorithm.

SAC is an off-policy actor-critic algorithm that uses a replay buffer to perform updates using past experiences. An important aspect of SAC is the use of the *reparametrization trick* [Hee+15; KW13] during the actor update. This implies that instead of using the policy gradient theorem and importance sampling weights, one can directly update the policy parameters in the direction maximizing the Q-values, provided that the sampled actions $\mathbf{a}_{i_\phi}$ are differentiable with respect to ϕ. This is made possible by the reparametrization trick, where actions are obtained by transforming samples from a fixed distribution (e.g., standard normal) using a differentiable function parameterized

Algorithm 8 Soft Actor-Critic (SAC)

Input: MDP $(\mathcal{S}, \mathcal{A}, \mu_0, \gamma)$, learning rate α, replay buffer $\mathcal{D}$, initial policy parameters ϕ, initial Q-network parameters θ
for episode = 0, 1, 2, ... **do**
  Initialize state $\mathbf{s}_0 \sim \mu_0$
  for t = 0, 1, 2, ... until episode ends **do**
    $\mathbf{a}_t \sim \pi_\phi(\cdot|\mathbf{s}_t)$ ▷ sample action from policy
    $\mathbf{s}_{t+1}, \mathrm{r}_t \leftarrow \mathrm{P}(\cdot|\mathbf{s}_t, \mathbf{a}_t), \mathrm{r}(\mathbf{s}_t, \mathbf{a}_t)$ ▷ agent-environment interaction
    Store $(\mathbf{s}_t, \mathbf{a}_t, \mathrm{r}_t, \mathbf{s}_{t+1})$ in $\mathcal{D}$
    Sample mini-batch $\{(\mathbf{s}_i, \mathbf{a}_i, \mathrm{r}_i, \mathbf{s}'_i)\}_{i=1}^B$ from $\mathcal{D}$ ▷ batch size B
    Critic update: ▷ minimize soft TD error
    $\mathbf{a}'_i \sim \pi_\phi(\cdot|\mathbf{s}'_i)$ for all $i = 1, ..., B$
    $$\theta \leftarrow \theta - \alpha \nabla_\theta \left(\frac{1}{B} \sum_{i=1}^B (\underbrace{r_i + \gamma(Q_{\text{copy}(\theta)}(\mathbf{s}'_i, \mathbf{a}'_i) - \lambda \log \pi_\phi(\mathbf{a}'_i|\mathbf{s}'_i)) - Q_\theta(\mathbf{s}_i, \mathbf{a}_i)}_{\text{TD with } \mathcal{T}^\pi_{\text{soft}}})^2 \right)$$
    Actor update: ▷ policy update
    $\mathbf{a}_{i_\phi} \sim \pi_\phi(\cdot|\mathbf{s}_i)$ for all $i = 1, ..., B$ ▷ reparametrization trick
    $\phi \leftarrow \phi + \alpha \nabla_\phi \left(\frac{1}{B} \sum_{i=1}^B (Q_\theta(\mathbf{s}_i, \mathbf{a}_{i_\phi}) - \lambda \log \pi_\phi(\mathbf{a}_{i_\phi}|\mathbf{s}_i)) \right)$
  end for
end for
return π_ϕ

by ϕ: $\mathbf{a}_\phi = \tanh(\mu_\phi + \sigma_\phi \odot \epsilon)$ with $\epsilon \sim \mathcal{N}(0, \mathrm{I})$ and $\odot$ denoting element-wise multiplication. The *tanh* (also called *squashing*) function is often used to bound the actions within a certain range, typically [-1, 1]. When used, the *log-likelihood* function is adjusted accordingly.

The introduced entropy term in SAC has an intriguing interpretation as a KL-divergence between the policy and the uniform distribution[6]: $\mathcal{H}(\pi) = -d_{\text{KL}}(\pi||\mathcal{U}(\mathcal{A}))$. Using this perspective, we can see that maximizing the entropy encourages the policy to stay close to the uniform distribution, thus promoting exploration. In general, the idea of policy optimization while remaining close to a certain distribution can be extended to other choices of distributions, depending on the specific application or desired behavior.

Regularized RL. Building upon this idea, we can generalize the concept of regularized RL by introducing a regularization term based on a divergence measure (e.g. KL-divergence) between the current policy and a reference policy π_{ref}. This can be achieved

[6] In this expression, an additional $\log |\mathcal{A}|$ term appears, where $|\mathcal{A}|$ is the volume of the action space. In practice, this term doesn't intervene in optimization and can be set to zero by normalizing $\mathcal{A}$ to have unit volume.

by introducing a hard constraint on the divergence between the two policies [KL02; Sch+15], and solving the following constrained optimization problem:

$$\max_{\phi} \eta[\pi_\phi] \text{ subject to } \max_{\mathbf{s}\in\mathcal{S}} d_{\mathrm{KL}}(\pi_\phi(\cdot|\mathbf{s})||\pi_{\mathrm{ref}}(\cdot|\mathbf{s})) \leq \delta,$$

where $\delta \geq 0$ is a threshold parameter. Approaches such as Trust Region Policy Optimization (TRPO) [Sch+15] solve exactly this problem, with the choice of the reference policy being the previous policy before the last update: $\pi_{\mathrm{ref}} = \pi_{\phi_{\mathrm{old}}}$. The motivation for this choice is to ensure that the updated policy does not deviate too much from the previous one, thereby maintaining stability during training. However, solving the above constrained optimization problem can be challenging in practice, requiring approximate second-order methods or computationally-heavy solvers[7].

As a more practical alternative, *Proximal Policy Optimization* (PPO) [Sch+17] relaxes the hard constraint using a clipped surrogate objective. This objective penalizes large deviations from the reference policy by clipping the probability ratio between the current and reference policies to a value close to 1. As depicted in Algorithm 9, PPO operates in a similar fashion to REINFORCE (6) in the sense that one or many episodes are collected from the environment prior to performing policy and value updates. The return G_t is then estimated as the *return-to-go*: $G_t = \sum_{t'=t}^{T-1} \gamma^{t'-t} \mathrm{r}_{t'}$, and used with a baseline to get an estimate of the advantage $\hat{A}_t$. A key difference between PPO and other actor-critic algorithms is the use of a value network V_θ that estimates the value function, instead of the Q-function. Furthermore, G_t is used as the target to train this value network, without reoccuring to the TD learning that we have seen for value-based methods so far. Indeed, when parameter updates are decoupled from agent-environment interactions, one can estimate the true value of a state from the rewards collected from that state, eliminating the need to bootstrap[8] the value function as done in the Bellman operator.

Regarding the policy update, the clipping objective of PPO consists in a minimum between the unclipped objective $\rho(\phi)\hat{A}$ and the clipped objective $\mathrm{clip}(\rho(\phi), 1-\epsilon, 1+\epsilon)\hat{A}$ which ensures that the policy updates remain within range. Taking the minimum does not simply enforce a smaller update, instead it removes the incentive to increase the objective when $\rho(\phi)$ moves outside the clipping range in the direction favored by $\hat{A}$. Therefore, this surrogate objective helps stabilizing training by preventing excessive policy shifts without explicitly constraining the step size. Thanks to its simplicity[9], PPO has

[7] For example, TRPO uses the conjugate gradient algorithm together with the Fisher information matrix to approximate the solution.

[8] In RL, bootstrapping refers to using the value function itself as a recursive proxy for part of the reward sum appearing in its original definition.

[9] Although simpler than the constrained optimization approaches like TRPO, it has been shown that in practice PPO requires numerous implementation details to ensure optimal performance [Hua+22a].

Algorithm 9 Proximal Policy Optimization (PPO)

Input: MDP $(\mathcal{S}, \mathcal{A}, \mu_0, \gamma)$, learning rate α, clip parameter ϵ,
Initialize policy parameters ϕ, value parameters θ
for episode = 0, 1, 2, ... **do**
  initialize $\mathbf{s}_0 \sim \mu_0$, $\mathcal{D} = \emptyset$
  Generate episode $(\mathbf{s}_0, \mathbf{a}_0, \mathrm{r}_0, \mathbf{s}_1, \mathbf{a}_1, ..., \mathbf{s}_T)$ using π_ϕ ▷ agent-environment interaction
  for t = 0, 1, 2, ... T **do**
    Compute empirical returns $\hat{G}_t = \sum_{t'=t}^{T-1} \gamma^{t'-t} \mathrm{r}_{t'}$
    Compute advantages $\hat{A}_t = \hat{G}_t - V_\theta(\mathbf{s}_t)$ for all t
    Store $(\mathbf{s}_t, \mathbf{a}_t, \hat{G}_t, \hat{A}_t, \pi_\phi(\mathbf{a}_t|\mathbf{s}_t))_{t=0}^{T-1}$ in $\mathcal{D}$ ▷ trajectory buffer
  end for
  for mini-batch $\{(\mathbf{s}_i, \mathbf{a}_i, \hat{G}_i, \hat{A}_i, \pi_{\phi_{\text{old}}}(\mathbf{a}_i|\mathbf{s}_i))\}_{i=1}^{B}$ in $\mathcal{D}$ **do** ▷ batch size B
    $\rho_i(\phi) = \frac{\pi_\phi(\mathbf{a}_i|\mathbf{s}_i)}{\pi_{\phi_{\text{old}}}(\mathbf{a}_i|\mathbf{s}_i)}$ ▷ likelihood ratio
    Critic update:
    $\theta \leftarrow \theta - \alpha \nabla_\theta \frac{1}{B} \sum_{i=1}^{B} \left(V_\theta(\mathbf{s}_i) - \hat{G}_i\right)^2$
    Actor update:
    $\mathcal{L}^{\text{clip}}(\phi) = \frac{1}{B} \sum_{i=1}^{B} \min\left(\rho_i(\phi)\hat{A}_i,\ \text{clip}(\rho_i(\phi), 1-\epsilon, 1+\epsilon)\hat{A}_i\right)$
    $\phi \leftarrow \phi + \alpha \nabla_\phi \mathcal{L}^{\text{clip}}(\phi)$
  end for
end for
return π_ϕ

become one of the most widely used RL algorithms in practice, especially in continuous[10] control tasks. It has also been successfully applied to fine-tune LLMs using human feedback [Ouy+22], leading to significant improvements in alignment on various natural language tasks.

RL for LLM fine-tuning. A variant of PPO called Groupe Relative Policy Optimization (GRPO) [Sha+24] has been proposed to address specific aspects of LLM fine-tuning using RL. These aspects are related to the fact that in this setting, the network inputs (states and actions) are sequences of tokens. Consequently, the networks themselves are constructed using the embedding components of LLMs, combined with a task-specific head for either policy or value estimation. Moreover, in Reinforcement Learning from Human Feedback (RLHF), the reward is derived from another LLM trained to predict a score reflecting human preferences. As a result, at least four distinct LLMs must be maintained in memory during training: the policy, the value function, the reward model, and the

[10] PPO can also be used with discrete action spaces by parameterizing the policy as logits of a softmax categorical distribution.

reference policy. To mitigate the high memory footprint, GRPO proposes to replace the baseline in the advantage estimate $\hat{A}_t$ in PPO by a *group relative average* computed over multiple actions sampled for each example. Additionally, GRPO explicitly includes a KL-regularization term in the objective to further constrain the policy updates and retain the LLM's initial conversational capabilities. This leads to the following modified objective for the actor update:

$$\mathcal{L}^{\text{GRPO}}(\phi) = \frac{1}{B}\sum_{i=1}^{B}\left[\min\left(\rho_i(\phi)\hat{A}_i,\ \text{clip}(\rho_i(\phi), 1-\epsilon, 1+\epsilon)\hat{A}_i\right) - \lambda d_{\text{KL}}(\pi_\phi(\cdot|\mathbf{s}_i)||\pi_{\text{ref}}(\cdot|\mathbf{s}_i))\right]$$

where

$$\hat{A}_i = \sum_{t'=t}^{T-1}\gamma^{t'-t}\frac{\mathrm{r}_{t'} - \text{mean}(\{\mathrm{r}_{t'_1}, \ldots, \mathrm{r}_{t'_K}\})}{\text{std}(\{\mathrm{r}_{t'_1}, \ldots, \mathrm{r}_{t'_K}\})},$$

and $\{\mathrm{r}_{t'_1}, \ldots, \mathrm{r}_{t'_K}\}$ are rewards obtained by sampling K different actions for the same state $\mathbf{s}_i$. GRPO forms the basis of the modern arsenal of RL algorithms used for fine-tuning LLMs on verifiable tasks, such as mathematical theorem proving and code generation. Since GRPO [Dee+25] demonstrated strong performance in LLM reasoning tasks, subsequent works have focused on improving its stability and overall performance. For instance, [Liu+25] introduced Dr GRPO (Distribution Robust GRPO), which eliminates the standard deviation normalization term to correct estimator bias and improve token efficiency. Addressing the instability of long-horizon reasoning, [Yu+25] proposed DAPO, which employs an asymmetric Clip-Higher mechanism and dynamic sampling to filter uninformative training tokens.

When testing some of these algorithms against popular benchmarks such as Atari games [Bel+13] or MuJoCo continuous control tasks [TET12], we can see that in general these so-called *model-free* algorithms require a large number of interactions with the environment to reach satisfactory performance. This is evidenced by Figure 2.2 where we see that even the best algorithms (respectively SAC for MuJoCo HalfCheetah and Rainbow DQN for Atari Breakout) need several million environment steps to reach high returns. The number of interactions required to learn a good policy is often referred to as the *sample efficiency* of an RL algorithm, with more sample-efficient algorithms requiring fewer interactions. One way to improve the *sample efficiency* of RL algorithms is to leverage a model of the environment dynamics, leading to the field of Model-based Reinforcement Learning (MBRL), which we will explore in the next section.

Summary. In this section, we began by introducing the fundamental components of RL, including the MDP framework, value functions, and the Bellman equations. We then examined a range of RL algorithms, starting with dynamic programming methods, followed by value-based approaches such as Q-Learning and DQN, and policy-based methods including REINFORCE and actor-critic algorithms. Finally, we covered more advanced techniques, such as entropy regularization, regularized RL, and the widely

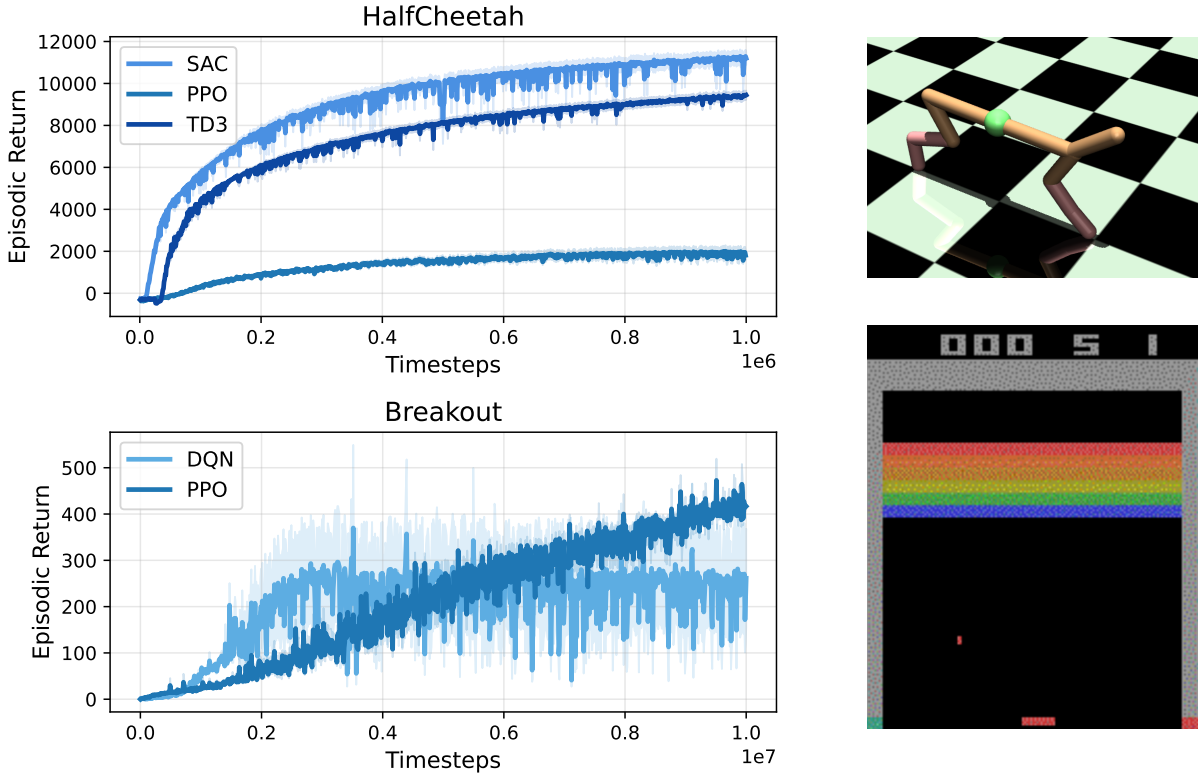


Figure 2.2: Return of model-free RL algorithms.

used PPO algorithm, along with its GRPO variant for LLM fine-tuning. Table 2.2 provides a summary of the main RL algorithms discussed in this section, highlighting their key characteristics.

Connection with the thesis

This section provided a comprehensive review of RL, which constitutes a central concept of the thesis. In Chapter 3, the SAC algorithm is trained on synthetically generated data and leveraged to guide planning within a model-based framework. Building on this foundation, Chapter 4 adapts LLMs as transition models and integrates them with SAC to improve sample efficiency. Finally, Chapter 6 shifts the focus to policy gradient methods in the context of post-training, with particular emphasis on REINFORCE and GRPO for LLMs fine-tuning.

Algorithm	type	$\mathcal{S}$	$\mathcal{A}$	P, r	actor	critic	objective	on-/off- policy
VI [Bel58]	DP	discrete	discrete	✓	greedy	tabular	a: ∅ c: TD($\mathcal{T}^\star$)	∅
PI [How60]	DP	discrete	discrete	✓	greedy	tabular	a: ∅ c: TD($\mathcal{T}^\pi$)	∅
Q-Learning [WD92]	Value-based	discrete	discrete	✗	ϵ-greedy	tabular	a: ∅ c: TD($\mathcal{T}^\star$)	OFF
SARSA [RN94]	Value-based	discrete	discrete	✗	ϵ-greedy	tabular	a: ∅ c: TD($\mathcal{T}^\pi$)	ON
DQN [Mni+15]	Value-based	continuous	discrete	✗	ϵ-greedy	Q_θ	a: ∅ c: TD($\mathcal{T}^\star$)	OFF
REINFORCE [Wil92b]	Policy-based	discrete, continuous	discrete, continuous	✗	π_ϕ	∅	a: $\eta[\pi_\phi]$ c: ∅	ON
A2C / A3C [Mni+16]	Actor-Critic	discrete, continuous	discrete, continuous	✗	π_ϕ	Q_θ	a: $\eta[\pi_\phi]$ c: TD($\mathcal{T}^\pi$)	ON
ACER [Wan+17]	Actor-Critic	discrete, continuous	discrete, continuous	✗	π_ϕ	Q_θ	a: $\eta[\pi_\phi]$ + IS c: TD($\mathcal{T}^\pi$)	OFF
SAC [Haa+18a]	Actor-Critic	discrete, continuous	continuous	✗	π_ϕ	Q_θ	a: $Q_\theta(\mathbf{s}, \mathbf{a}_\phi)$ c: TD($\mathcal{T}^\pi_{\text{soft}}$)	OFF
TRPO [Sch+15]	Actor-Critic	discrete, continuous	discrete, continuous	✗	π_ϕ	V_θ	a: $\eta[\pi_\phi]$ s.t. $d_{\text{KL}} \leq \delta$ c: $(V_\theta - G_t)^2$	ON
PPO [Sch+17]	Actor-Critic	discrete, continuous	discrete, continuous	✗	π_ϕ	V_θ	a: $\mathcal{L}^{\text{clip}}(\phi)$ c: $(V_\theta - G_t)^2$	ON
GRPO [Sha+24]	Policy-based	discrete, continuous	discrete, continuous	✗	π_ϕ	∅	a: $\mathcal{L}^{\text{GRPO}}(\phi)$ c: ∅	ON

Table 2.2: **Summary of RL algorithms.** Each row corresponds to an algorithm, while each column describes one of its key properties. **Type** takes values among: dynamic programming (DP), value-based, policy-based, or actor–critic methods. The state space $\mathcal{S}$ and action space $\mathcal{A}$ take values among discrete (▭▭▭) or continuous (∿). P, r column specifies whether the transition kernel P and reward function r are assumed to be known (✓) or unknown (✗). **Actor** and **Critic** respectively refer to the policy (neural π_ϕ or greedy/ϵ-greedy), and the value model (tabular ▦, neural Q_θ or V_θ, or none ∅). The **objective** column shows the learning objectives of the actor (e.g., $\eta[\pi_\phi]$ using the policy gradient theorem, $Q_\theta(\mathbf{s}, \mathbf{a}_\phi)$ using the reparametrization trick) and the critic (e.g., TD($\mathcal{T}^\pi$), TD($\mathcal{T}^\star$), or $(V_\theta - G_t)^2$ with G_t being any return estimate). Finally, the **on-/off-policy** column denotes whether the algorithm learns from data gathered by its current policy (on-policy ON) or from external/replayed experience (off-policy OFF).

2.2 Model-based Reinforcement Learning

Besides the restricted class of DP algorithms, the classical RL algorithms we have presented so far assume no prior knowledge about the environment dynamics. Indeed, they learn a policy (and/or value function) directly from experience gathered through agent-environment interactions, which are realizations of the true transition dynamics P and reward function r. Another way to use this collected experience is to learn a model of the environment dynamics, which can then be used to further improve policy learning.

Precisely, what we mean by *model* here is a parameterized approximation of either the transition dynamics P (called *dynamics model*) or the reward function r (called *reward model*). This approach is known as MBRL [Moe+23; Luo+22], and Figure 2.3 shows the modification it adds to the original RL loop.

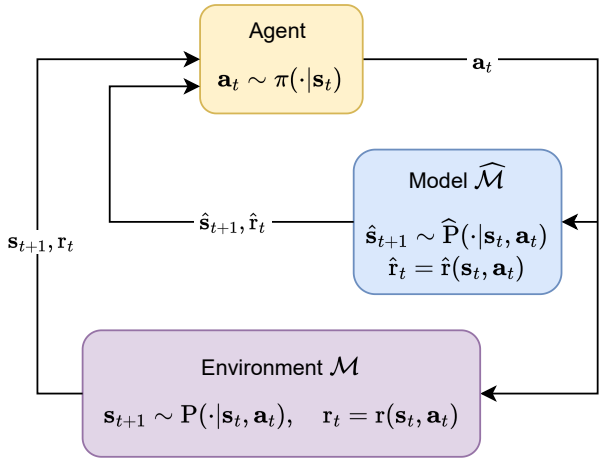


Figure 2.3: The Model-based Reinforcement Learning loop.

Having a dynamics model and a reward model, we can plug-in these learned components into the MDP framework by replacing the true environment dynamics P and reward function r by their respective approximations $\widehat{\mathrm{P}}_\psi$ and $\hat{\mathrm{r}}_\psi$. This results in a *learned* MDP $\widehat{\mathcal{M}}$ defined as follows:

Definition 2.2.1 (Learned MDP)**.**

$$\widehat{\mathcal{M}} = (\mathcal{S}, \mathcal{A}, \mu_0, \gamma, \widehat{\mathrm{P}}_\psi, \hat{\mathrm{r}}_\psi)$$

where $\mathcal{S}$, $\mathcal{A}$, γ, μ_0 are the same as $\mathcal{M}$, and $\widehat{\mathrm{P}}_\psi$ and $\hat{\mathrm{r}}_\psi$ are learned approximations of P and r, respectively.

Given a dataset $\mathcal{D} = \{(\mathbf{s}_t, \mathbf{a}_t, \mathrm{r}_t, \mathbf{s}_{t+1})\}_{i=1}^N$ of past experiences collected from the true environment $\mathcal{M}$ and a data-collecting policy $\pi_\mathcal{D}$, we can learn the parameters ψ of the dynamics and reward models by minimizing supervised losses as in definition 2.2.2. The dynamics model loss corresponds to a Maximum Likelihood Estimation (MLE) of the transition probabilities, while the reward model objective minimizes the MSE between the predicted and true rewards. Depending on the model's characteristics, these objectives may vary; when they do, we will state them explicitly, otherwise references to the dynamics and reward model objectives will refer to those defined in 2.2.2.

Definition 2.2.2 (Model objective)**.**

$$\min_{\psi} \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_{t+1})\sim\mathcal{D}}\Big[-\log \widehat{\mathrm{P}}_\psi(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)\Big] \qquad \textit{(dynamics model)}$$

$$\min_{\psi} \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t,\mathrm{r}_t)\sim\mathcal{D}}\Big[|\widehat{\mathrm{r}}_\psi(\mathbf{s}_t,\mathbf{a}_t)-\mathrm{r}_t|^2\Big] \qquad \textit{(reward model)}$$

where $\mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t,\mathrm{r}_t,\mathbf{s}_{t+1})\sim\mathcal{D}}$ *is a Monte Carlo estimate of the true expectation* $\mathbb{E}_{\mathrm{P},\pi_\mathcal{D}}$*.*

In the next sections, we will focus on dynamics models and assume that the reward function is either known or directly accessible from the environment.

2.2.1 Dynamics models

So far we have defined the dynamics model as being a *Forward model*, meaning that it predicts the next state $\mathbf{s}_{t+1}$ given the current state $\mathbf{s}_t$ and action $\mathbf{a}_t$. We can similarly define two other types of models: *Backward model* [MA93; Lai+20; Lee+20] where the goal is to predict the previous state $\mathbf{s}_{t-1}$ and action $\mathbf{a}_{t-1}$ given the current state $\mathbf{s}_t$, and *Inverse model* [LaV98; PMB21] where the goal is to predict the action $\mathbf{a}_t$ taken to go from state $\mathbf{s}_t$ to state $\mathbf{s}_{t+1}$. Besides this model classification, there exist other ways to differentiate dynamics models [Moe+23], such as *deterministic* vs *probabilistic*, or *parametric* vs *non-parametric* models. In the rest of the thesis, we will continue to assume forward models while exploring these other model types.

Deterministic vs probabilistic. A common choice when designing dynamics models is to decide whether to make them deterministic or probabilistic. A *deterministic* dynamics model directly predicts the next state as a function of the current state and action: $\widehat{\mathbf{s}}_{t+1} = \widehat{f}_\psi(\mathbf{s}_t,\mathbf{a}_t)$. This type of model is especially relevant when the environment dynamics are also deterministic[11], meaning that a given state-action pair always leads to the same next state: $\forall\ \mathbf{s},\mathbf{a},\ \ \exists\ \mathbf{s}'$ s.t. $\mathrm{P}(\mathbf{s}'|\mathbf{s},\mathbf{a})=1$. For deterministic models, the training objective boils down to minimizing the MSE between the predicted next state and the observed next state, akin to a classical regression problem:

$$\min_{\psi} \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_{t+1})\sim\mathcal{D}}\Big[\|\widehat{f}_\psi(\mathbf{s}_t,\mathbf{a}_t)-\mathbf{s}_{t+1}\|^2\Big].$$

On the other hand, a *probabilistic* dynamics model outputs a distribution over the next state, mimicking an inherently stochastic environment. There are many reasons to opt for a probabilistic model instead of a deterministic one. First of all, a deterministic model disregards the randomness stemming from a stochastic environment. Furthermore, even in the case of a deterministic environment, a probabilistic model enables uncertainty

[11] In this case, we write the transition dynamics as a function: $f:\mathcal{S}\times\mathcal{A}\to\mathcal{S}$ where $\mathbf{s}_{t+1}=f(\mathbf{s}_t,\mathbf{a}_t)$.

quantification which helps identify randomness that is stemming from limited data or model capacity. We now cite the two main approaches to learn probabilistic dynamics models: The *frequentist* and the *Bayesian* approaches.

The *frequentist* approach to uncertainty modeling corresponds to finding the optimal parameters that minimize the Negative Log-Likelihood (NLL) (as in definition 2.2.2) of the observed next states under the predicted distribution. When the state and action spaces are finite (the tabular case), the optimal parameters optimizing the MLE objective correspond to the counting estimator[12] [Sut91; BT02] of the transition probabilities:

$$\widehat{\mathrm{P}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) = \frac{\mathrm{count}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})}{\sum_{\mathbf{s}'} \mathrm{count}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}')}$$

where $\mathrm{count}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$ is the number of times the transition $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$ has been observed in the dataset $\mathcal{D}$. In the continuous case, the probabilistic model can be parameterized using any function approximation technique (e.g., neural networks) to output the parameters of a distribution whose likelihood model is tractable. Typically, this can be achieved by outputting the parameters of a Gaussian distribution from which the next state is sampled: $\widehat{\mathrm{P}}_\psi(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) = \mathcal{N}(\mathbf{s}_{t+1}; \mu_\psi(\mathbf{s}_t, \mathbf{a}_t), \sigma_\psi(\mathbf{s}_t, \mathbf{a}_t)\mathrm{I})$.

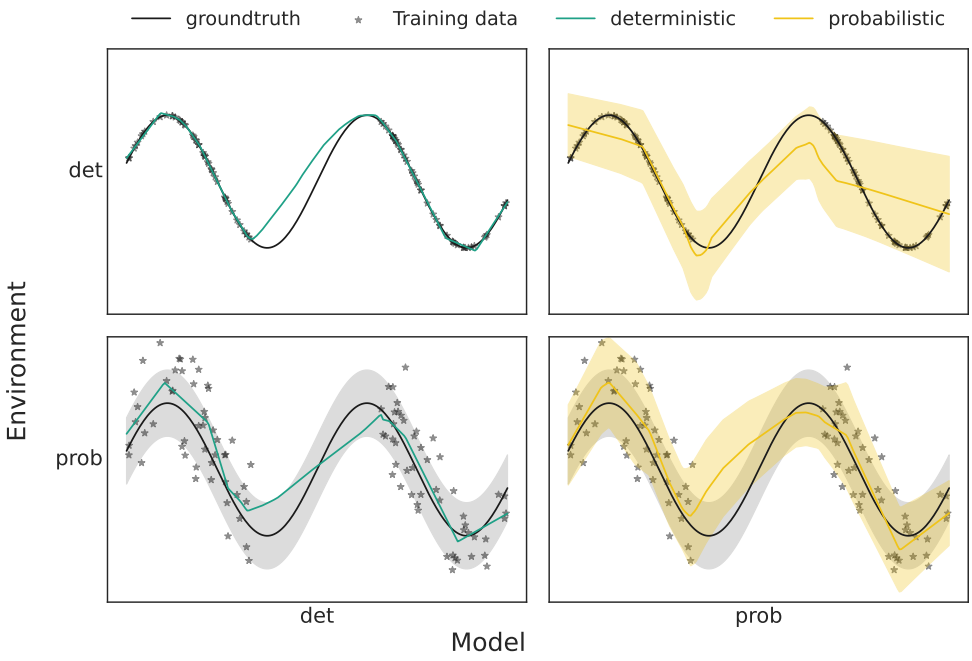


Figure 2.4: Deterministic vs probabilistic models.

Figure 2.4 illustrates the difference between neural network-based deterministic and stochastic (frequentist) models evaluated against determinisitc and stochastic environment dynamics. For deterministic environment dynamics, both models are able to accurately

[12] An instance of the Laplacian smoothing operator, which can be shown [HOP18] to be the minimax optimal estimator in this setting.

fit the function, with the stochastic model correctly estimating the uncertainty of regions with no data (see the center of the x-axis). In stochastic environments, although a stochastic model can correctly capture uncertainty in regions supported by data, it fails to extrapolate that uncertainty to unseen regions, resulting in overconfident and inaccurate predictions.

This issue can be mitigated using the *Bayesian* [Bay63] approach to uncertainty modeling that consists in placing a prior distribution over the model parameters ψ and inferring a posterior distribution given the observed data using Bayes theorem[13]. This results in a predictive transition distribution which integrates over the uncertainty in the model parameters:

$$\widehat{\mathrm{P}}(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) = \int \widehat{\mathrm{P}}_\psi(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) p(\psi|\mathcal{D}) d\psi.$$

This integral is often intractable, and various approximation techniques have been proposed to estimate it, such as variational inference [KW14] or Markov Chain Monte Carlo [Has70; JS96]. When using neural networks as function approximators in the context of Bayesian inference, these models are refered to as *Bayesian Neural Networks* (BNN) [Nea96; Mac95] where an entire distribution is maintained over the network weights instead of scalar values. Another example is Bayesian Linear Regression [Bis07]. In this case, when the prior distribution and the likelihood model are Gaussian, finding the mode of the posterior distribution (the so-called *maximum a posteriori* estimate) is equivalent to minimizing a Ridge-regularized least-squares[14] objective. In the context of MBRL, BNNs have been successfully used to model the environment dynamics in an uncertianty-aware manner [GMR16a; Dep+17; Dep+18].

Parametric vs nonparametric. The previously described Bayesian treatment applies to *parametric* models, where the model structure is fixed and only the parameters are learned. There exist also *nonparametric* models, where the model size can grow with the amount of data. One example is Gaussian Processes (GP) [RW06], which define a distribution over functions and can be used to model the transition dynamics in a Bayesian manner. Formally, a GP is a collection of (infinite) random variables such that any finite subset of them has a joint multivariate normal distribution. A GP is completely specified by two elements: a mean function $m(x)$, and a covariance function (or kernel) $k(x, x')$, which defines the similarity between data points and determines the shape and smoothness of the resulting function. PILCO (Probabilistic Inference for Learning COntrol) [DR11] is a well-known MBRL algorithm that uses GPs to model the environment dynamics and perform policy gradient based on the learned model.

[13]Bayes theorem states the relationship between the posterior $p(\psi|\mathcal{D})$, the likelihood $p(\mathcal{D}|\psi)$, and the prior $p(\psi)$ as $p(\psi|\mathcal{D}) = \frac{p(\mathcal{D}|\psi)p(\psi)}{p(\mathcal{D})}$, where $p(\mathcal{D})$ is called the evidence.

[14]For a linear model: $\hat{Y} = WX$, this corresponds to minimizing $\|Y - WX\|_2^2 + \lambda\|W\|_2^2$, where λ is a parameter controlling the strength of the regularization (equivalently the prior distribution).

Uncertainty quantification. In MBRL, it is crucial to accurately quantify the uncertainty in the learned dynamics model, as this uncertainty directly impacts the quality of the resulting policy. A distinction is made between *aleatoric* uncertainty, arising from inherent stochasticity in the environment dynamics, and *epistemic* uncertainty, which reflects the model's estimation and approximation errors[15]. Models based on Bayesian principles, such as BNNs or GPs, inherently capture both types of uncertainty. Indeed, the posterior distribution over the model parameters accounts for epistemic uncertainty, while the obtained predictive distribution captures aleatoric uncertainty. On the other hand, MLE-based models only capture aleatoric uncertainty through the variance of the learned distribution, in which case this estimate is contaminated by the model's epistemic uncertainty. To equip these models with uncertainty separation [KG17], several techniques have been proposed, such as *ensemble* methods [Die00] and Monte Carlo dropout [GG16b]. For instance, ensembling consists in having $N_{\mathrm{m}} > 1$ models, each initialized randomly and trained on a set $\mathcal{D}_\ell$ for $\ell \in \{1, \dots, N_{\mathrm{m}}\}$ generated by sampling with replacement[16] from the common dataset $\mathcal{D}$. Using ensembles, we can compute a disagreement metric [Lu+22] to capture the epistemic uncertainty, as opposed to the aleatoric uncertainty learned by each member of the ensemble separately:

$$\underbrace{\mathrm{Var}(\mathbf{s}_{t+1})}_{\text{Total Uncertainty}} = \underbrace{\frac{1}{N_{\mathrm{m}}} \sum_{i=1}^{N_{\mathrm{m}}} \sigma_i^2(\mathbf{s}_t, \mathbf{a}_t)}_{\text{Aleatoric}} + \underbrace{\frac{1}{N_{\mathrm{m}}} \sum_{i=1}^{N_{\mathrm{m}}} (\mu_i(\mathbf{s}_t, \mathbf{a}_t) - \bar{\mu}(\mathbf{s}_t, \mathbf{a}_t))^2}_{\text{Epistemic}},$$

where μ_i and σ_i are the mean and standard deviation predicted by model i of the ensemble, and $\bar{\mu}$ is the average mean prediction across all models.

Latent space models. When dealing with high-dimensional state spaces, such as images, learning dynamics models directly in the observation space can be challenging due to the complexity of modeling pixel-level transitions. Furthermore, these observations might be incomplete or noisy representations of the true underlying state of the environment. Partially-observable MDPs (POMDP) [KLC95] are a generalization of MDPs that account for such scenarios by introducing an observation space $\mathcal{O}$ and an observation function $\mathrm{P}_{\mathcal{O}} : \mathcal{S} \to \mathcal{O}$ that defines the probability of observing $\mathbf{o}_t$ given the underlying state $\mathbf{s}_t$. To address these challenges, latent space models have been proposed [HS18b; Haf+19a], which learn a lower-dimensional representation of the state space, known as the latent space $\mathcal{Z}$. These models typically consist of an encoder $\mathrm{enc}_\psi : \mathcal{O} \to \mathcal{Z}$ that maps observations to latent states, a dynamics model $\widehat{\mathrm{P}}_\psi : \mathcal{Z} \times \mathcal{A} \to \mathcal{Z}$ that predicts the next latent state given the current latent state and action, and a decoder $\mathrm{dec}_\psi : \mathcal{Z} \to \mathcal{O}$

[15] Also called bias and variance in empirical risk minimization theory. They indicate respectively the gap between the actual solution and the best in-class solution, and the gap between the found solution and the best in-class solution.

[16] Referred to as bootstrapping in statistics.

that reconstructs observations from latent states. All the components of the latent space model can be learned jointly using a combination of reconstruction and prediction losses. However, reconstruction is not always necessary, and some approaches focus solely on learning reconstruction-free dynamics models in the latent space [Ngu+; DJA; OT22; Gar+24]. To account for partial observability, sequence-modelling architectures such as RNNs (Recurrent Neural Networks) [Chu+15] or Transformers [Vas+17] are often used to maintain a history of past observations, enabling the model to infer the underlying transition more accurately.

2.2.2 Planning using dynamics models

Traditionally, *Planning* refers to any computational process that uses a model of the environment dynamics (either known or learned) to obtain an optimal policy [GB13]. For instance, the classical DP algorithms we have presented in the previous section are examples of planning methods, as they assume access to the true environment dynamics P and reward function r. In general, Planning methods can be categorized into two main classes: *decision-time planning* and *background planning* [SB18]. The former type of planning is performed at each decision step, where the agent uses the model to simulate possible future trajectories and select the best action based on these simulations. On the other hand, background planning involves using the model to generate data "in the background", and improve the policy based on the model-generated experience (as in Figure 2.3). In the following, we will discuss different instances of both decision-time and background planning methods.

Decision-time planning

Model Predictive Control (MPC). MPC [CB99] is a decision-time planning method that uses the model recursively to plan at each timestep t and optimize over action sequences $\{(\mathbf{a}^k_{t:t+H})_{k\in\{1,\dots,K\}}\}$ where $H \in \mathbb{N}^*$ is the planning horizon, and $K \in \mathbb{N}^*$ the number of candidates (*population size*). As opposed to learning a parametrized policy π_ϕ, MPC is simple to implement, has lower computational burden (since no gradients are involved), and is more accurate than optimizing over a single action per timestep. The latter advantage comes at the cost of higher running time per action selection as the number of model calls grows linearly with the planning horizon and the population size. The evaluation function $f_{\widehat{\mathrm{P}}_\psi,\mathrm{r}} : \mathcal{A}^H \times \mathcal{S} \to \mathbb{R}$ is the expected cumulative reward of the action sequence starting from state $\mathbf{s}$:

$$f^{\text{eval}}_{\widehat{\mathrm{P}}_\psi,\mathrm{r}}(\mathbf{a}_{t:t+H}, \mathbf{s}) = \sum_{\tau=t}^{t+H} \mathbb{E}_{\widehat{\mathrm{P}}_\psi}[\mathrm{r}(\mathbf{s}_\tau, \mathbf{a}_\tau) \mid \mathbf{s}_t = \mathbf{s}]$$

where the expectation is taken over the learned dynamics model $\widehat{\mathrm{P}}_\psi$, and r is the true reward function [17]. The MPC controller then performs the first action $\mathbf{a}_t^\star$ of the optimal action sequence $\mathbf{a}_{t:t+H}^\star = \arg\min_{\mathbf{a}_{t:t+H}} f(\mathbf{a}_{t:t+H})$ in the real environment, and restarts the process in the next timestep. Multiple ways to sample the candidate action sequences exist; the most naïve approach being to sample uniformly at random from the action space $\mathcal{A}$. This results in the Random Shooting (RS) algorithm, whose pseudo-code is given in Algorithm 10.

Algorithm 10 Random Shooting (RS)

Input: state $\mathbf{s}_t$, dynamics model $\widehat{\mathrm{P}}_\psi$, reward function r
Sample action sequences $\{\mathbf{a}_{t:t+H-1}^k\}_{k=1}^K \sim \mathcal{U}(\mathcal{A}^H)$ ▷ uniform distribution
$\{R^k\}_{k=1}^K \leftarrow f^{\text{eval}}_{\widehat{\mathrm{P}}_\psi,\mathrm{r}}(\{\mathbf{a}_{t:t+H-1}^k\}_{k=1}^K, \mathbf{s}_t)$ ▷ evaluate action sequences
$k^\star = \arg\max_k R^k$
return $\pi_{\text{RS}}(\mathbf{s}_t) = \mathbf{a}_t^{k^\star}$ ▷ first action of best sequence

Although simple and effective, RS suffers from the curse of dimensionality[18], especially when planning for long horizons or dealing with high-dimensional action spaces. Inspired from the field of evolutionary algorithms [CL18], more sophisticated sampling methods have been proposed to address this issue, such as CEM (Cross-Entropy Method) [de +05]. In CEM, the sampling distribution is iteratively refined over multiple iterations by selecting the top-performing action sequences (the so-called *elite set*) and updating the sampling distribution parameters to better fit these elite samples. The pseudo-code for CEM is given in Algorithm 11. PETS (Probabilistic Ensembles with Trajectory Sampling) [Chu+18] is a MBRL algorithm that uses CEM with an ensemble of probabilistic neural network-based dynamics models. PETS whose pseudo-code is given in Algorithm 12 has shown strong empirical performance on various continuous control benchmarks, achieving high sample efficiency and competitive asymptotic performance compared to model-free algorithms.

PETS performs planning directly in the observation space, meaning that the dynamics model predicts the next observation given the current observation and action. In parallel, it is also possible to apply a similar strategy in a learned latent space when using latent dynamics models, as done in the PlaNet (Deep Planning Network) algorithm

[17] It is also possible to virtually extend the planning horizon to ∞ by bootstrapping with a value function: $\sum_{\tau=t}^{t+H} \mathbb{E}_{\widehat{\mathrm{P}}_\psi}[\mathrm{r}(\mathbf{s}_\tau, \mathbf{a}_\tau) \mid \mathbf{s}_t = \mathbf{s}] + \mathbb{E}_{\widehat{\mathrm{P}}_\psi}[V(\mathbf{s}_{t+H+1})]$

[18] The curse of dimensionality refers to the phenomenon where, as the number of dimensions increases, the volume of the space grows exponentially, causing data points to become increasingly sparse.

Algorithm 11 Cross-Entropy Method

Input: state $\mathbf{s}_t$, dynamics model $\widehat{\mathrm{P}}_\psi$, reward function r, elite fraction ρ, iterations I
Initialize sampling distribution $\mathrm{P}_{\mathrm{CEM}} \in \Delta(\mathcal{A}^H)$ ▷ e.g. Gaussian
for $i = 1, 2, ..., I$ **do**
  Sample $\{\mathbf{a}^k_{t:t+H-1}\}_{k=1}^K \sim \mathrm{P}_{\mathrm{CEM}}$
  $\{R^k\}_{k=1}^K \leftarrow f^{\mathrm{eval}}_{\widehat{\mathrm{P}}_\psi, \mathrm{r}}(\{\mathbf{a}^k_{t:t+H-1}\}_{k=1}^K, \mathbf{s}_t)$ ▷ evaluate action sequences
  Select elite set $\mathcal{E}$ ▷ top ρK sequences
  Update $\mathrm{P}_{\mathrm{CEM}}$ using $\mathcal{E}$ ▷ e.g., fit Gaussian to elite set
end for
return $\pi_{\mathrm{CEM}}(\mathbf{s}_t) = \mathbb{E}_{\mathrm{P}_{\mathrm{CEM}}}[\mathbf{a}_t]$

Algorithm 12 PETS

Input: MDP $(\mathcal{S}, \mathcal{A}, \mathrm{r}, \mu_0, \gamma)$, replay buffer $\mathcal{D}$, CEM parameters, initial ensemble parameters $\psi = \{\phi_i\}_{i=1}^{N_m}$, learning rates α
for episode = 0, 1, 2, ... **do**
  Initialize $\mathbf{s}_0 \sim \mu_0$
  for t = 0, 1, 2, ... until episode ends **do**
    Model learning:
    for model i = 1, 2, ... N_m **do**
      $\mathcal{D}_i = \mathrm{bootstrap}(\mathcal{D})$ ▷ sample with replacement
      $\psi_i \leftarrow \psi_i + \alpha \nabla_{\psi_i} \sum_{(\mathbf{s},\mathbf{a},\mathbf{s}') \in \mathcal{D}_i} \left[\log \widehat{\mathrm{P}}_{\psi_i}(\mathbf{s}'|\mathbf{s}, \mathbf{a})\right]$ ▷ MLE loss
    end for
    Planning:
    $\mathbf{a}_t \leftarrow \pi_{\mathrm{CEM}}(\mathbf{s}_t)$ ▷ MPC
    $\mathbf{s}_{t+1}, \mathrm{r}_t \leftarrow \mathrm{P}(\cdot|\mathbf{s}_t, \mathbf{a}_t), \mathrm{r}(\mathbf{s}_t, \mathbf{a}_t)$ ▷ agent-environment interaction
    Store $(\mathbf{s}_t, \mathbf{a}_t, \mathrm{r}_t, \mathbf{s}_{t+1})$ in $\mathcal{D}$
  end for
end for
return π_{PETS}

[Haf+19a]. In this case, the observations are first encoded into a lower-dimensional representation, while the evaluation function is computed by rolling-out trajectories in this latent space. The PlaNet algorithm further learns a reward model in the latent space to compute the evaluation function, and uses a RSSM (Recurrent State-Space Model) [Chu+15] as the dynamics model to better capture temporal dependencies. Regarding decision-time planning paradigms, several alternatives to MPC exist, such as the more elaborated MPPI (Model-Predictive Path Integral) [WAT17] used in several MBRL algortihms such as MBOP (Model-Based Offline Planning) [AD21]. TD-MPC (Temporal Difference Model Predictive Control) [HWS22] and its scalable variant TD-MPC2 [HSW24]

are one example that combine reconstruction-free latent dynamics models with MPPI to achieve state-of-the-art performance in continuous control environments.

Monte-Carlo Tree Search (MCTS). An alternative planning paradigm called MCTS [Cha+08b] explicitly reasons over long-horizon sequences by building a partial search tree over state action trajectories. When combined with a learned dynamics model, MCTS can be used to simulate future rollouts and selectively expand promising branches according to an exploration–exploitation trade-off, typically guided by upper confidence bounds [LR85; KCG12]. In contrast to trajectory optimization methods such as RS or CEM, which optimize open-loop action sequences, MCTS performs closed-loop planning by conditioning future actions on the visited states. This property makes MCTS particularly attractive in discrete or structured action spaces and in settings where long-term strategic planning is required. Several algorithms have successfully integrated MCTS planning, most notably in AlphaGo [Sil+16] and AlphaZero [Sil+18] which use the true transition function and tackle challenging tasks such as chess and GO. In the context of learned dynamics models, MuZero [Sch+20] is a prominent example that combines MCTS with a learned latent dynamics model to achieve state-of-the-art performance on various environments.

Background planning

Background planning methods use the learned dynamics model to generate synthetic experience offline, which is then combined with real interaction data to improve the value or the policy. In this paradigm, the agent alternates between updating the dynamics model from the replay buffer, generating synthetic transitions using the learned model, and updating the policy or value function using both or either of the real and synthetic data. The so-called *Dyna*-style methods [Sut91] constitute instantiations of this paradigm with RL algorithms such as Q-learning or Actor-Critic methods. From a formal perspective, Dyna-style methods rely on policy evaluation and improvement under the learned MDP $\widehat{\mathcal{M}}$. When training a policy π on model generated data, the corresponding return and value function under the learned dynamics are defined as

$$\hat{\eta}[\pi] = \mathbb{E}_{\mu_0,\pi,\widehat{\mathrm{P}}_\psi}\left[\sum_{t=0}^{\infty}\gamma^t \mathrm{r}(\mathbf{s}_t,\mathbf{a}_t)\right], \quad \hat{V}^\pi(\mathbf{s}) = \mathbb{E}_{\pi,\widehat{\mathrm{P}}_\psi}\left[\sum_{t=0}^{\infty}\gamma^t \mathrm{r}(\mathbf{s}_t,\mathbf{a}_t) \mid \mathbf{s}_0 = \mathbf{s}\right],$$

with an analogous definition for the action-value function $\hat{Q}^\pi$. Using $\hat{\eta}[\pi]$ to evaluate a policy π introduces an error compared to the true return $\eta[\pi]$ due to the inaccuracies in the learned dynamics model. When the model is learned on a dataset $\mathcal{D}$ collected with a data-collecting policy $\pi_\mathcal{D}$, the simulation lemma (lemma 2.2.1) characterizes how com-

pounding model errors and policy mismatch[19] jointly influence the deviation between the true and learned returns [KS02; BT02; KKL03; XLY20b].

Lemma 2.2.1 (Simulation lemma). *Given bounded model error and policy shift*

$$\mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}}\Big[d_{\mathrm{TV}}\Big(\mathrm{P}(.\mid\mathbf{s},\mathbf{a})||\widehat{\mathrm{P}}_\psi(.\mid\mathbf{s},\mathbf{a})\Big)\Big]\leq\epsilon_\psi \quad \textit{and} \quad \max_{\mathbf{s}} d_{\mathrm{TV}}(\pi(.|\mathbf{s})||\pi_{\mathcal{D}}(.|\mathbf{s}))\leq\epsilon_\pi$$

the return error of the policy π is bounded as

$$|\hat{\eta}[\pi]-\eta[\pi]|\leq 2\,\mathrm{r}_{max}\left[\frac{\gamma}{(1-\gamma)^2}\,\epsilon_\psi+\frac{2}{(1-\gamma)^2}\epsilon_\pi\right],$$

where $\mathrm{r}_{max}=\max_{\mathbf{s},\mathbf{a}}|\mathrm{r}(\mathbf{s},\mathbf{a})|$.

Proof. See Lemma 3 in [XLY20a] for a full proof. □

The first term in the bound captures the effect of compounding model errors, which grows quadratically with the effective horizon $1/(1-\gamma)$. In practice, ϵ_ψ can be assimilated to the model's average training loss on the dataset $\mathcal{D}$. The second term captures the effect of policy mismatch, which can be controlled by ensuring that the learned policy does not deviate too much from the data-collecting policy $\pi_{\mathcal{D}}$.

A practical instance of the dyna-style paradigm is the Model-Based Policy Optimization (MBPO) algorithm [Jan+19] which has shown strong empirical performance by combining an ensemble of probabilistic dynamics models with the SAC algorithm. MBPO uses the learned dynamics model to generate *short* rollouts starting from states sampled from the real experience replay buffer, and adds these model-generated transitions to the replay buffer used by SAC. This approach helps mitigating the compounding errors that can arise from long rollouts in the learned model, while still benefiting from the additional data provided by the model. As explicited in Algorithm 13, MBPO also learns a reward function $\widehat{\mathrm{r}}_\psi$ alongside the dynamics model. In the same fashion as MBPO, multiple other dyna-style MBRL algorithms have been proposed such as ME-TRPO (Model Ensemble TRPO) [Kur+18], where the TRPO algorithm is used instead of SAC for policy optimization.

While the simulation lemma provides a general characterization of how model error affects policy evaluation, it does not directly explain the empirical effectiveness of short-horizon model rollouts as used by MBPO. In particular, MBPO relies on *branched rollouts* that first follow the true environment dynamics, then at timestep t with probability γ^t

[19] Both the model error and the policy shift are quantified as a total variation distance (denoted d_{TV}) between their respective input distributions.

Algorithm 13 MBPO

Input: MDP $(\mathcal{S}, \mathcal{A}, \mu_0, \gamma)$, replay buffers $\mathcal{D}, \mathcal{D}_{\text{model}}$, model rollout horizon h, ensemble parameters $\psi = \{\psi_i\}_{i=1}^{N_m}$, policy parameters ϕ, value parameters θ, learning rate α
for episode = 0, 1, 2, ... **do**
  Initialize $\mathbf{s}_0 \sim \mu_0$
  for t = 0, 1, 2, ... until episode ends **do**
    $\mathbf{a}_t \sim \pi_\phi(\cdot|\mathbf{s}_t)$
    $\mathbf{s}_{t+1}, \mathrm{r}_t \leftarrow \mathrm{P}(\cdot|\mathbf{s}_t, \mathbf{a}_t), \mathrm{r}(\mathbf{s}_t, \mathbf{a}_t)$ ▷ agent-environment interaction
    Store $(\mathbf{s}_t, \mathbf{a}_t, \mathrm{r}_t, \mathbf{s}_{t+1})$ in $\mathcal{D}$
    Model learning:
    for model $i = 1, 2, ...N_m$ **do**
      $\mathcal{D}_i = \text{bootstrap}(\mathcal{D})$
      $\psi_i \leftarrow \psi_i - \alpha \nabla_{\psi_i} \sum_{(\mathbf{s},\mathbf{a},\mathrm{r},\mathbf{s}') \in \mathcal{D}_i} \left[-\log \widehat{\mathrm{P}}_{\psi_i}(\mathbf{s}'|\mathbf{s}, \mathbf{a}) + |\widehat{\mathrm{r}}_{\psi_i}(\mathbf{s}, \mathbf{a}) - \mathrm{r}|^2 \right]$
    end for
    Model rollouts:
    Sample initial states $\hat{\mathbf{s}}_0 \sim \mathcal{D}$
    $\{(\hat{\mathbf{s}}_t, \mathbf{a}_t, \hat{\mathrm{r}}_t, \hat{\mathbf{s}}_{t+1})\}_{t=0}^{H} \sim \pi_\phi, \widehat{\mathrm{P}}_\psi$ ▷ Generate model rollouts
    Store $(\hat{\mathbf{s}}_t, \mathbf{a}_t, \hat{\mathrm{r}}_t, \hat{\mathbf{s}}_{t+1})$ in $\mathcal{D}_{\text{model}}$
    Policy optimization:
    Update π_ϕ and Q_θ using SAC on $\mathcal{D} \cup \mathcal{D}_{\text{model}}$ ▷ real + model data
  end for
end for
return π_ϕ

switch to the learned dynamics model for h steps. The branched return bound introduced in MBPO then quantifies the discrepancy between the h-step branched rollout and the true return, relating it to the model error, policy shift, and the rollout horizon h. theorem 2.2.1 highlights the trade-off between the rollout horizon k and the compounding model error. By choosing a smaller h, the impact of model error on the return estimation is reduced, at the cost of less model-generated data to train the policy.

Theorem 2.2.1 (Branched return bound)**.** *Given similar bounds on the model error and policy shift as in lemma 2.2.1, the h-step branched return error is bounded as*

$$|\hat{\eta}[\pi] - \eta[\pi]| \leq 2\, \mathrm{r}_{max} \left[\frac{\gamma^{h+1} \epsilon_\pi}{(1-\gamma)^2} + \frac{\gamma^h + 2}{1-\gamma}\ \epsilon_\pi + \frac{h}{1-\gamma}(\epsilon_\psi + 2\, \epsilon_\pi) \right],$$

where $\mathrm{r}_{max} = \max_{\mathbf{s},\mathbf{a}} |\mathrm{r}(\mathbf{s}, \mathbf{a})|$.

Proof. See Theorem 4.2 in [Jan+19] for a full proof. □

Background planning in latent spaces. Similar to decision-time planning, background planning can also be performed in latent spaces when using latent dynamics models. In this setting, algorithms such as Dreamer [Haf+20] learn an RSSM that captures the dynamics of the environment and generate imagined rollouts entirely within this latent space. These imagined trajectories are then used to train an actor–critic algorithm, allowing both the policy and value function to be optimized without any interaction with the real environment. Thanks to lower dimensionality, operating in a latent space enables long-horizon rollouts with reduced compounding errors and improved sample efficiency, while abstracting away irrelevant observation-level details. Subsequent works have explored a variety of backbone architectures for latent world models, including gated recurrent units as in the Dreamer saga of algorithms [Haf+20; Haf+21a; Haf+25; HYL25], transformers [Che+24; MAF23; BT25; Zha+23c], and more recently structured state-space models [DPA23; Sam+24]. These architectural choices offer improved expressiveness and the ability to model long-range temporal dependencies in complex environments.

In addition to decision-time planning and background planning, learned dynamics models have been used in RL in several other complementary ways. MVE (Model Value Expansion) [Fei+18] and its variants, such as STEVE (STochastic Ensemble Value Expansion) [Buc+18], use short model-based rollouts to improve value estimates while bootstrapping from model-free critics in actor-critic algorithms. In offline RL, algorithms such as MOPO (Model-based Offline Policy Optimization) [Yu+20a], MOReL (Model-Based Offline Reinforcement Learning) [Kid+20], and COMBO (Conservative Offline Model-Based Policy Optimization) [Yu+21a] leverage uncertainty estimates obtained from ensembles of probabilistic dynamics models to penalize policy optimization in regions where the model is unreliable. More broadly, model uncertainty has been used not only for conservatism but also to guide exploration by encouraging actions that reduce epistemic uncertainty [Wan+24]. Regarding MBRL theory, many prior works established performance bounds and sample complexity results for MBRL algorithms especially in the tabular setting [AJO08; AOM17; ZB19]. In more general settings, prior work has established a theoretical framework to optimize lower bounds on the true return resulting in the SLBO (Stochastic Lower Bound Optimization) algorithm [Luo+19]. Furthermore, MBRL methods have been successfully combined with a range of domains, including safe RL [Hua+24; Pao+24] and control theory [Mon+24; Sun+24].

Summary. The main components of MBRL algorithms, namely dynamics model learning and planning with the learned model, have been presented. In Table 2.3, several prominent MBRL algorithms are summarized along with their key properties, including the type of dynamics model used, whether it is deterministic or probabilistic, single-model or ensemble-based, the neural architecture employed, the planning method, and the type of policy or critic. This table provides a comprehensive overview of the diverse

Algorithm	model type	det vs prob	single vs ens	parametric	architecture	planning	policy / critic
RS [KHT21]	P, r	∅	∅	✗	∅	decision-time	RS
PILCO [DR11]	GP, r	prob	single	✗	∅	background	π_ϕ
PETS [Chu+18]	$\{\hat{\mathrm{P}}_{\psi_i}\}_i, \mathrm{r}$	prob	ens	✓	MLP	decision-time	CEM
PlaNet [Haf+19a]	latent	prob	single	✓	RNN	decision-time	CEM
VB [Dep+17]	BNN, r	prob	single	✓	MLP	background	π_ϕ
MBOP [AD21]	$\{\hat{f}_{\psi_i}, \mathrm{r}_{\psi_i}\}_i$	det	ens	✓	MLP	decision-time	MPPI
TD-MPC2 [HSW24]	latent	prob	single	✓	RNN	decision-time	MPPI
AlphaZero [Sil+18]	P, r	∅	∅	✗	∅	decision-time	MCTS
MuZero [Sch+20]	latent	det	single	✓	RNN	decision-time	MCTS
MBPO [Jan+19]	$\{\hat{\mathrm{P}}_{\psi_i}, \mathrm{r}_{\psi_i}\}_i$	prob	ens	✓	MLP	background	SAC
ME-TRPO [Kur+18]	$\{\hat{\mathrm{P}}_{\psi_i}\}_i, \mathrm{r}$	det	ens	✓	MLP	background	TRPO
Dreamer [Haf+25]	latent	prob	single	✓	RNN	background	Actor-Critic
TWM [BT25]	latent	prob	single	✓	Transformer	background	Actor-Critic
R2I [Sam+24]	latent	prob	single	✓	S4	background	Actor-Critic

Table 2.3: **Summary of MBRL algorithms.** Each row corresponds to an algorithm, while each column describes one of its key properties. **Model type** takes values among: P, r for the true dynamics and reward, $\{\hat{\mathrm{P}}_{\psi_i}, \hat{\mathrm{r}}_{\psi_i}\}_i$ for probabilistic ensembles, etc. The following columns show model properties such as **det vs prob** (deterministic or probabilistic), **single vs ens** (single model or ensemble), and **parametric** (✓) vs nonparametric (✗). The **architecture** column describes the neural architecture used for the dynamics model: MLP, RNN, Transformer, etc. **Planning** indicates whether the algorithm uses decision-time planning or background planning. Finally, the **policy / critic** column specifies the type of policy or critic used by the algorithm.

design choices in MBRL and highlights the various approaches used to leverage learned dynamics models for sample-efficient RL.

To conclude this section, we show the aforementioned sample efficiency advantage of MBRL over model-free RL in the same environments used in Figure 2.2. In HalfCheetah, using MBPO which is practically augmenting the SAC algorithm with model generated data improves the sample efficiency of the latter. The Breakout Atari environment exhibits a more pronounced presence of the sample efficiency that can be achieved by means of model-based algorithms. Indeed, the MuZero algorithm achieves orders of

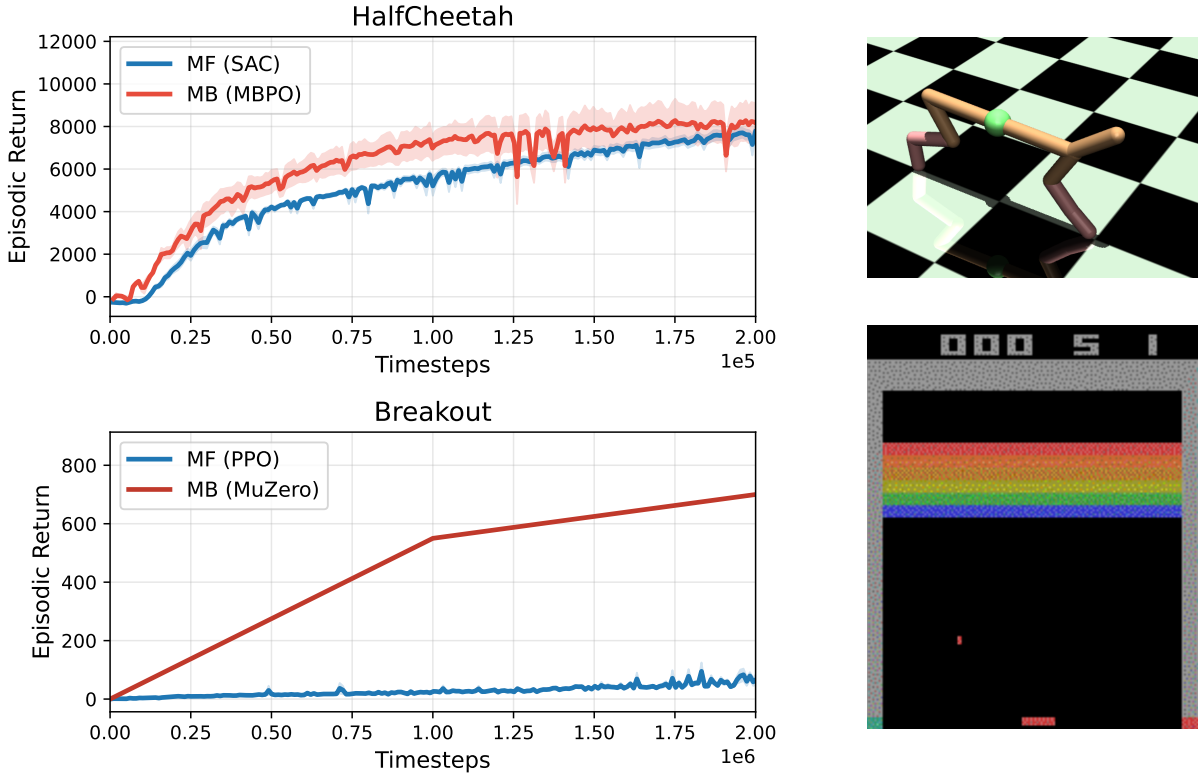


Figure 2.5: Return of model-based (MB) and model-free (MF) RL algorithms.

magnitude better performance than its model-free counterpart PPO. A more comprehensive sample comlexity analysis can be found in [Wan+19].

Connection with the thesis

This section provided a comprehensive review of MBRL, the concept at the core of this thesis. Throughout the next chapters of the thesis, we put a particular emphasis on forward, single, parametric, neural network-based, and probabilistic models. In Chapter 3, we build upon the MBRL framework presented here to develop multi-step dynamics models that better capture long-term dependencies, leveraging a combination of decision-time planning via MPC and background planning through MBPO. In Chapter 4, the focus shifts to background planning, where SAC is provided with model-generated data in the spirit of MBPO. Furthermore, we build on the branched return bound (theorem 2.2.1) in Chapter 4, and provide a generalized version that extends its applicability to multi-branch rollouts.

2.2.3 Evaluation metrics

Although dynamics models in MBRL essentially solve a supervised learning problem, their evaluation goes beyond standard metrics such as log-likelihood or MSE on a held-out test set. Indeed, the ultimate goal of learning a dynamics model is to leverage it for planning or policy optimization, which makes it part of a complex pipeline involving multiple components. In this section, we review the main evaluation metrics used in the MBRL literature. We start by discussing typical regression metrics for dynamics models, called *static* metrics in the context of MBRL, with an emphasis on *single-step* vs *multi-step* evaluation. We then discuss *dynamic* evaluation that assesses the performance of the overall MBRL algorithm when using the learned dynamics model. Finally, we review uncertainty estimation metrics, particularly calibration metrics, which are crucial when using probabilistic dynamics models.

Static metrics. The most straightforward way to evaluate a learned dynamics model $\widehat{\mathrm{P}}_\psi$ is to measure its predictive accuracy on a held-out test set $\mathcal{D}_{\text{test}} = \{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i)\}_{i=1}^{N_{\text{test}}}$. In continuous state spaces (which is assumed for the rest of the manuscript), common metrics include the average NLL or MSE between the predicted and true next states. The choice of either metric depends primarily on the type of model (deterministic vs probabilistic). However, it also depends on how the model is used, for example whether planning relies on sampling from the model or on its mean prediction. For Gaussian likelihood-based models with diagonal covariance[20], there is a direct relationship between NLL and MSE, as can be seen by deriving the log-likelihood function:

$$-\log \widehat{\mathrm{P}}_\psi(\mathbf{s}'|\mathbf{s},\mathbf{a}) = \frac{d_s}{2}\log(2\pi) + \frac{1}{2}\sum_{i=1}^{d_s}\log\hat{\sigma}^2_{\psi,i}(\mathbf{s},\mathbf{a}) + \frac{1}{2}\sum_{i=1}^{d_s}\frac{(s'_i - \hat{\mu}_{\psi,i}(\mathbf{s},\mathbf{a}))^2}{\hat{\sigma}^2_{\psi,i}(\mathbf{s},\mathbf{a})},$$

where d_s is the dimension of the state space, and $\hat{\mu}_\psi, \hat{\sigma}^2_\psi$ are the predicted mean and variance. From this expression, it is clear that minimizing the NLL encourages accurate mean predictions through the last term that resembles a weighted MSE.

Figure 2.6 compares the aformentioned metrics for two Gaussian predictive distributions ($\hat{\mathrm{P}}_1$ and $\hat{\mathrm{P}}_2$) having the same mean ($\hat{\mu}_1 = \hat{\mu}_2$), but different variances ($\hat{\sigma}_1 > \hat{\sigma}_2$). As can be seen in the figure, minimizing NLL also encourages appropriate uncertainty estimates as overconfident predictions get penalized through the amplifying coefficient $1/\hat{\sigma}^2_{\psi,i}$.

When states span a high dimensional space (e.g., $\mathcal{S} := \mathbb{R}^{d_s}$ with $d_s >> 1$), reporting per-dimension metrics such as the average MSE or NLL per state dimension can in general be more informative. Nevertheless, it is more practical to provide aggregated metrics, especially in the context of benchmarks [TET12; Bro+16; Bel+13] involving multiple

[20] Here we assume a diagonal covariance matrix: $\hat{\Sigma}_\psi = \hat{\sigma}^2_\psi \mathrm{I}$ following common practices in MBRL.

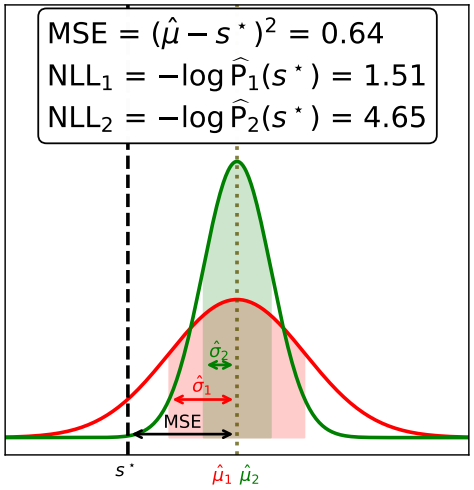


Figure 2.6: Comparing MSE and NLL for predictive distributions with different variances.

environments with different state space dimensions and different dimension semantics. To this end, prior works [KHT21] have proposed to report normalized metrics such as the R-squared (R^2) score which normalizes the MSE by the total variance of each state dimension:

$$R^2 = \frac{1}{d_s}\sum_{j=1}^{d_s} R_j^2, \quad R_j^2 = 1 - \frac{\sum_{i=1}^{N_{\text{test}}}(\mathbf{s}_i'^j - \hat{\mu}_\psi^j(\mathbf{s}_i, \mathbf{a}_i))^2}{\sum_{i=1}^{N_{\text{test}}}(\mathbf{s}_i'^j - \bar{\mathbf{s}}'^j)^2},$$

where $j \in \{1, \ldots, d_s\}$ indexes the state dimensions, and $\bar{\mathbf{s}}'$ is the empirical mean. The R^2 score takes values in $(-\infty, 1]$, with a value of 1 indicating perfect predictions, and a value of 0 indicating that the model performs no better than predicting the empirical mean.

Single-step vs multi-step. When generating long rollouts using the learned dynamics model, errors can compound over time, leading to significant deviations from the true trajectories. This phenomenon is referred to as the *compounding errors* issue [Lam+21b; LPC22], and it is a well-known challenge in MBRL. The paradoxical nature of this issue stems from the fact that the dynamics model is typically trained to minimize one-step prediction errors, which is well-motivated given the Markovian property of the studied MDPs. Indeed, under the Markov assumption, the information required to predict the next state is fully contained in the current state and action, making one-step predictions sufficient for accurate modeling. However, when the learned model is rolled out for multiple steps, a key difference occurs: the model's inputs at each step are no longer drawn from the true state distribution, but rather from the distribution induced by the model itself. This distribution shift can lead to the accumulation of small errors over time, resulting in significant deviations from the true trajectories. To better capture the

model's performance in long-horizon rollouts, multi-step evaluation metrics are essential to avoid any surprises during planning time. Examples of multi-step metrics include the average MSE or NLL over h-step rollouts, defined as

$$\text{MSE}(h) = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \|\mathbf{s}'_{i,h} - \hat{\mu}_\psi(\hat{\mu}_\psi(\dots, \mathbf{a}_{i,h-2}), \mathbf{a}_{i,h-1})\|^2,$$

where $\mathbf{s}'_{i,h}$ is the state obtained after rolling out the learned model for t steps starting from $\mathbf{s}_i$ and following the action sequence $\{\mathbf{a}_{i,0}, \dots, \mathbf{a}_{i,h-1}\}$.

Calibration. When discussing the difference between MSE and NLL metrics for probabilistic models, we highlighted the importance of appropriate uncertainty estimates as captured by Figure 2.6. Indeed, in MBRL, uncertainty estimates are crucial for effective planning and policy optimization, and have numerous applications such as offline RL [Yu+20a; Yu+21a] and enhanced exploration [Wan+24]. *Calibration* is a way to quantify the quality of uncertainty estimates provided by probabilistic models [GBR]. In the context of classification, calibration has a straightforward meaning: the model's confidence score should correspond to its actual error rate [KFF17; Nix+19; Min+21]. This means that for predictions with confidence score $p\%$, the predicted label should be correct $p\%$ of the time for a model to be correctly calibrated. Calibration in classification models is not achieved by default, in fact it has been shown that modern neural networks improve accuracy to the detriment of calibration [Guo+17a]. As a remedy, many post-hoc recalibration methods have been proposed such as Histogram binning [ZE01], isotonic regression [ZE02], and temperature scaling [Pla99; NC05; HVD15].

In the context of dynamics modeling, we are interested in assessing the calibration of regression models. Defining calibration for regression is not a trivial task [Lev+20a; UR20]. In fact, multiple definitions have been proposed in prior work (see distribution calibration [Son+19] for instance). In this work, we adopt the *quantile regression* [KFE18a] definition which admits a similar interpretation as calibration for classification models. Its formal definition goes as follows:

Definition 2.2.3 (Quantile calibration)**.** *Given $\mathcal{X}, \mathcal{Y} := \mathbb{R}$, a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, and a predictor $f : \mathcal{X} \to \mathcal{Y}$ that learns a CDF F_i over y_i, quantile calibration means that y_i (groundtruth) should fall in the $p\%$ quantile $p\%$ of the time:*

$$\frac{\sum_{i=1}^N \mathbb{I}\{y_i \leq \mathrm{F}_i^{-1}(p)\}}{N} \to p \quad \textit{for all} \quad p \in [0,1] \quad \textit{as} \quad N \to \infty,$$

where $\mathrm{F}_i^{-1} : [0,1] \to \mathcal{Y}$ denotes the quantile function $\mathrm{F}_i^{-1}(p) = inf\{y : p \leq \mathrm{F}_i(y)\}$.

In the context of MBRL, calibration metrics and their impact on performance have been studied in prior works [Mal+19]. One way to assess quantile calibration is computing

the Kolmogorov-Smirnov (KS) statistic for each state dimension $j \in \{1, \dots, d_s\}$. Starting from the model predicted CDF $\mathrm{F}^j_\psi(\mathbf{s}'^j|\mathbf{s}, \mathbf{a})$, we define the empirical CDF of the quantiles corresponding to groundtruth values:

$$\mathcal{F}^j_\psi(x) = \frac{\left|\left\{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i) \in \mathcal{D} | \mathrm{F}^j_\psi(\mathbf{s}'^j_i|\mathbf{s}_i, \mathbf{a}_i) \leq x\right\}\right|}{N}, \qquad \text{for } x \in [0, 1].$$

For each sample $(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i)$ indexed by $i \in \{1, \dots, N\}$, we denote the predicted quantile as $q_{i,j} = \mathrm{F}^j_\psi(\mathbf{s}'^j_i|\mathbf{s}_i, \mathbf{a}_i)$. We then define the KS statistic as the largest absolute difference between the empirical and uniform quantiles across the dataset $\mathcal{D}$:

$$\mathrm{KS} = \frac{1}{d_s}\sum_{j=1}^{d_s} \mathrm{KS}_j, \quad \mathrm{KS}_j = \max_{i \in \{1,\dots,N\}} \left|\mathcal{F}^j_\psi(q_{i,j}) - q_{i,j}\right|.$$

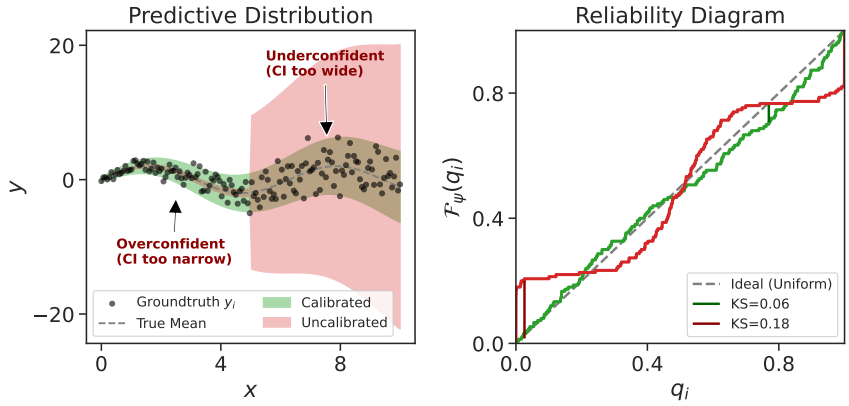


Figure 2.7: An example of calibrated vs uncalibrated predictive distributions (left). The corresponding reliability diagram (right).

Figure 2.7 illustrates the calibration concept by showing a miscalibrated predictive distribution, both in terms of overconfidence and underconfidence (left panel). The right panel then shows the *reliability diagram*, a visualization tool that illustrates the empirical CDF $\mathcal{F}^j_\psi$ against the ideal diagonal line corresponding to perfect calibration. The KS statistic can be interpreted as the maximum deviation from the ideal diagonal line, with lower values indicating better calibration.

Dynamic metrics. Ultimately, the goal of learning a dynamics model in MBRL is to use it for planning. Therefore, it is crucial to evaluate the performance of the overall MBRL algorithm when using a learned dynamics model. This so-called dynamic performance is typically measured by the final return achieved by the learned policy in the true environment after model-based training. At this level of evaluation, another paradoxical phenomenon occurs: a dynamics model with lower prediction error does not

necessarily lead to better returns. This phenomenon is called the *objective mismatch* issue [Lam+21a]. This issue arises from the fact that the dynamics model is typically trained to minimize prediction errors on a dataset collected by some behavior policy, which may not align with the objective of maximizing returns. Prior works have addressed this issue, suggesting multiple value-aware [FBN17; Voe+22] or policy-aware [AGF21] model learning objectives. In the next section, we will explore common benchmarks used to evaluate MBRL algorithms, along with open-source libraries that facilitate their implementation.

Connection with the thesis

Chapter 3 further discusses multi-step evaluation and is dedicated to the compounding errors phenomenon. The same chapter also provides insights into the objective mismatch issue, showing that static evaluation does not necessarily correlate with dynamic performance. In Chapters 4 and 5, we leverage the uncertainty evaluation tools introduced here to assess the calibration of probabilistic models.

2.3 Foundation Models

The field of machine learning is built on *empirical risk minimization* theory [Bac24], where a model is trained on a labeled dataset to minimize a loss, with the goal of generalizing to unseen test data from the same distribution. In contrast, FMs are trained on large collections of datasets (often via self-supervision[21]) and then adapted to various downstream tasks, including those different from the pretraining task. Therefore, the focus is shifted from learning task-specific models to learing versatile representations that capture the underlying structure of the data and that can be adapted to a wide range of tasks. Below are some definitions of a foundation model from various sources:

Definition 2.3.1 (Foundation Model)**.**

- [*Vas+*] *"In generative AI, [Foundation models are] models trained on broad data using self-supervised learning that can be adapted such as through fine-tuning for a variety of downstream tasks."*
- [*Mer25*] *"Foundation models are AI neural networks trained on massive unlabeled datasets to handle a wide variety of jobs from translating text to analyzing medical images."*

[21] Self-supervised learning [Gui+24] refers to a training paradigm where the model learns to predict parts of the input data from other parts, without requiring explicit labels.

- [*aws25*] *"The term foundation model was coined by researchers to describe ML models trained on a broad spectrum of generalized and unlabeled data and capable of performing a wide variety of general tasks."*

These definitions agree on the key characteristics of FMs: large-scale training on broad datasets, self-supervised learning, and the ability to adapt to various downstream tasks. Conceptually, FMs can be seen as applications of *multi-task learning* [ZY21; CZY24] and *transfer learning* [PY10] at scale. Multi-task learning refers to the paradigm of training a model to solve multiple tasks simultaneously by minimizing a joint loss function that combines the losses of individual tasks. Meanwhile, Transfer learning includes all techniques that leverage knowledge learned from one task to advantageously solve another task. From the pretraining of FMs on large datasets using self-supervised learning, FMs can be seen as a form of multi-task learning since completing parts of different inputs can be interpreted as solving multiple tasks. On the other hand, adapting FMs to downstream tasks is a form of transfer learning, where the knowledge acquired during pretraining is transferred to new tasks. Before addressing ways of adapting FMs to MBRL data, we first review examples of FMs in both text and non-text domains.

2.3.1 Text data: LLMs

The most prominant example of FMs is within the field of NLP where LLMs have emerged as powerful tools for understanding and generating human-like text. An LLM is typically defined as a neural network model, often based on the transformer architecture [Vas+17], that is trained on a vast corpus of sequences, $U = \{U_1, U_2, \ldots, U_i, \ldots, U_N\}$, where each sequence $U_i = (u_1, u_2, \ldots, u_j, \ldots, u_{n_i})$ consists of tokens u_j from a vocabulary $\mathcal{V}$. Decoder-only LLMs [Rad+19; DL24] typically encode an autoregressive distribution, where the probability of each token is conditioned only on the previous tokens in the sequence, expressed as $p_\theta(U_i) = \prod_{j=1}^{n_i} p_\theta(u_j|u_{0:j-1})$. The parameters θ are learned by maximizing the probability of the entire dataset, $p_\theta(U) = \prod_{i=1}^{N} p_\theta(U_i)$. When trained on a massive scale of data and compute, and plugged-in standalone software systems that manage user interactions (from prompt to response), LLMs have constituted a technological revolution, as evidenced by the widespread adoption of ChatGPT and the like.

Tokenization. Every LLM has an associated tokenizer, whose goal is to break down an input string into a sequence of tokens, each belonging to $\mathcal{V}$. The most naive way to break down text into tokens is character-level tokenization, where each character is treated as a separate token. However, this approach can lead to long sequences and may not capture the semantic meaning of words effectively. On the other hand, word-level tokenization treats each word as a separate token, which can lead to an explosion in vocabulary size. Modern LLMs use subword tokenization, which strikes a balance by

breaking rare words into common components (e.g., *"unbelievable"* $\rightarrow$ [*"un"*, *"believ"*, *"able"*]). Popular subword tokenization algorithms include Byte Pair Encoding (BPE) [Gag94], WordPiece [Dev+19a], and Unigram [KR18]. To ensure that all possible inputs can be tokenized, modern tokenizers use Byte-level BPE [Rad+19] which treats text as a sequence of UTF-8[22] bytes.

Connection with the thesis

Beyond specific text-generation tasks such as instruction following and question answering, LLMs have shown remarkable generalization capabilities across a wide range of other tasks. These tasks include settings where the input data is not textual, yet the model is still able to process it effectively. For instance, LLMs have been successfully used for time series forecasting, where sequences of numerical values replace sequences of text tokens at the input [Gru+23b]. In Chapter 4, we will bridge the gap between LLMs and MBRL, and explore ways to use them for dynamics modeling, providing insights into multidimensional numerical input tokenization.

2.3.2 Non-text data: Time series

The FMs revolution extended beyond text data, reaching various other modalities such as images [Rad+; Car+26; Pod+23], audio [Déf+24], video [Wie+25], and structured data [Qu+; Gri+26; Ans+25]. Structured data refers to data that is organized in a predefined manner, mainly rows and columns as in tabular data, and temporal sequences as in time series data. In the context of MBRL, we are more interested in structured data, since in continuous state and action spaces, the problem of dynamics modeling is akin to a tabular regression problem with temporal dependencies. Therefore, we focus on time series data as a non-text modality where FMs have shown promising abilities that can be transferred to MBRL [Gru+23b; Ans+24; Ans+25].

Specifically, we consider a multivariate long-term time series forecasting task, represented by: a data matrix $\mathbf{X} \in \mathbb{R}^{L \times D}$ where L is the context window size and D is the multivariate time series dimensionality, and a target matrix $\mathbf{Y} \in \mathbb{R}^{H \times D}$, where H is the forecasting horizon. We denote by $\mathbf{x}_d \in \mathbb{R}^{L \times 1}$ (respectively $\mathbf{y}_d \in \mathbb{R}^{H \times 1}$) the d-th component of the input (respectively target) multivariate time series. We distinguish between two forecasting paradigms: *direct* forecasting where a model $f_\theta : \mathbb{R}^{L \times D} \rightarrow \mathbb{R}^{H \times D}$ predicts the entire target horizon $\mathbf{Y}$ in one shot, and *autoregressive* forecasting where the model $f_\theta : \mathbb{R}^{L \times D} \rightarrow \mathbb{R}^{1 \times D}$ predicts one step at a time. Similarly, we distinguish between *univariate* forecasting where each dimension d is forecasted independently, and *multivariate*

[22] Unicode Transformation Format (UTF) 8-bit is a character encoding standard used for electronic communication.

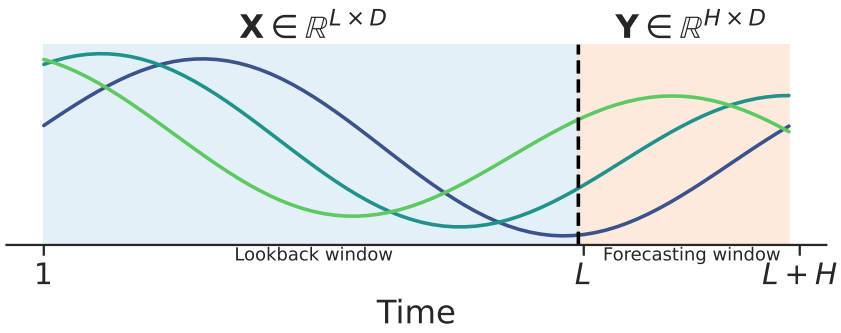


Figure 2.8: Multivariate time series.

forecasting where all dimensions are forecasted jointly. Figure 2.8 illustrates a multivariate time series with $D = 3$ dimensions, highlighting windows corresponding to the context $\mathbf{X}$ and the target $\mathbf{Y}$.

TSFMs. In the steps of the NLP field, TSFMs have emerged as a powerful tool for time series forecasting [Gos+24; Ans+25; Liu+24d; Ras+24; Das+24]. Analoguous to the diversity found in large text corpora akin to those present in internet scale data, time series data can also be diverse, spanning various domains such as finance, healthcare, weather, and energy. This diversity makes TSFMs particularly well-suited for learning generalizable representations that can be adapted to specific forecasting tasks. Many time series-specific FMs have been proposed in the literature, leveraging architectures such as transformers (decoder-only [Das+24], encoder-only [Gos+24], encoder-decoder [Liu+24d]), recurrent neural networks [Aue+25], and fully Multilayer Perceptron (MLP) [Eka+24]. In terms of tokenization, TSFMs often employ techniques such as *patching* [Feo+25] and *quantization* [Jin+24a]. Patching involves segmenting the time series into smaller, fixed-size patches that are treated as individual tokens. They are particularly useful to capture local information present in the time series signal. Quantization, on the other hand, involves mapping continuous time series values to discrete tokens, enabling the repurposing of LLM-like models for time series data [Jin+24a]. One particular challenge faced by TSFMs is the handling of multivariate time series. Indeed, training the same neural network on inputs of different dimensionality is not straightforward as it involves designing architectures that adapt to inputs with varying size.

Connection with the thesis

Akin to the datasets found in MBRL, the goal of this thesis is to model multivariate data with varying number of dimensions. In Chapter 5, we will explore ways to adapt TSFMs to effectively handle multivariate time series with varying dimensionality, drawing parallels with dynamics modeling in MBRL. In addition to the multivariate aspect, we will also investigate adapting deterministic TSFMs to probabilistic forecasting, which is crucial for uncertainty estimation.

2.3.3 Adapting Foundation Models

There exist multiple way to use a pretrained FM for downstream tasks. We refer to this process as *adaptation*, which encompasses a variety of techniques ranging from zero-shot transfer to fine-tuning. In this section, we review the main adaptation methods used in practice, highlighting their advantages and limitations.

Zero-shot transfer. The most immediate way to adapt a pretrained foundation model is via zero-shot transfer, where the model is applied to a downstream task without any parameter updates or task-specific training [LEB08; Cha+08a; Rad+19]. Formally, given a model f_θ pretrained on a source dataset $\mathcal{D}$, we evaluate its performance on a target task $\mathcal{D}_{\text{target}} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ (in the case of a supervised learning task). The model generates a prediction $\hat{\mathbf{y}}_i = f_\theta(\textit{prompt}(\mathbf{x}_i))$, where the *prompt* function maps the input data into a format compatible with the pretraining objective. For LLMs, this typically corresponds to prompt formatting, where the task is described in natural language (e.g., *"Translate the following sentence to French:* [*Input*]*"*), and the model output is interpreted accordingly [Bro+20]. In the time series setting, zero-shot transfer may consist in rescaling a target time series to match the normalization used during pretraining, or reshaping a multivariate input $\mathbf{X} \in \mathbb{R}^{L \times D}$ into a sequence of patches compatible with the pretrained tokenizer. As an example, a TSFM pretrained on logs monitoring data may be directly applied to electricity consumption time series without observing any labeled data from this domain during training [Coh+24]. The effectiveness of zero-shot transfer relies on the alignment between the pretraining distribution and the downstream task, and serves as a strong indicator of the generality of the learned representations [Pha+23].

In-context Learning (ICL). ICL is a specialized form of adaptation where the model's behavior is conditioned on a few demonstrations provided within the context window, again without modifying the weights θ [Aky+23; Li+23; Xie+22]. Given a target input $\mathbf{x}_{\text{target}}$ and a set of N examples $\mathcal{S} = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\}$, the model predicts an output conditioned on the target input and the demonstrations: $\hat{\mathbf{y}}_{\text{target}} = f_\theta(\textit{prompt}(\mathcal{S}, \mathbf{x}_{\text{target}}))$.

In LLMs, ICL manifests as *few-shot* prompting, where examples such as question-answer pairs are embedded directly into the prompt [McC+18; Rad+19; Bro+20]. In time series forecasting, ICL can be implemented by concatenating multiple realizations $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$ along the temporal dimension, and asking the model to predict the next timestep [LSY25]. For instance, in a MBRL setting, providing a sequence of previous state-action transitions $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$ in the context allows the model to capture the environment's dynamics and predict future states more accurately. The capacity for ICL is often considered an emergent property of scale [Bro+20], particularly in transformer-based architectures, where the self-attention mechanism can dynamically attend to the provided examples.

Fine-tuning. Unlike the previous gradient-free methods, fine-tuning involves adapting the model's parameters to a specific downstream dataset $\mathcal{D}_{\text{target}} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ through empirical risk minimization [Dev+19b]. Starting with the pretrained weights θ_{pre}, we optimize a task-specific loss $\mathcal{L}$ to obtain $\theta_{ft} = \arg\min_\theta \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_{target}}[\mathcal{L}(f_\theta(\mathbf{x}), \mathbf{y})]$. Depending on the amount of available data and computational constraints, fine-tuning can range from full-model updates to parameter-efficient variants [Hou+19] where only a subset of parameters is trained (e.g., LoRA [Hu+21]). For example, a TSFM pretrained on generic hourly data can be fine-tuned on a small set of high-frequency physiological sensor data to capture specific medical anomalies. This step bridges the gap between the general-purpose representations learned during pretraining and the requirements of a niche domain, ensuring the model's outputs are aligned with the target distribution. Fine-tuning generally offers the best downstream performance, at the cost of additional training and reduced task generality compared to zero-shot and in-context approaches.

Connection with the thesis

In this thesis, we explore various adaptation methods for leveraging FMs in the context of MBRL. In Chapter 4, we investigate zero-shot transfer of LLMs and ICL for dynamics modeling, assessing their effectiveness in capturing environment dynamics without task-specific training. In Chapter 5, we focus on a hybrid adaptation approach that combines fine-tuning with ICL to enhance the performance of TSFMs on multivariate time series forecasting. Finally, the reward modelling approach developped in Chapter 6 is applied in the context of fine-tuning LLMs for reasoning tasks.

Chapter 3

A Multi-step Loss Function for Robust Learning of the Dynamics in Model-based Reinforcement Learning

Contents

In model-based reinforcement learning, most algorithms rely on simulating trajectories from one-step models of the dynamics learned on data. A critical challenge of this approach is the compounding of one-step prediction errors as the length of the trajectory grows. In this chapter we tackle this issue by using a multi-step objective to train one-step models. Our objective is a weighted sum of the MSE loss at various future horizons. We find that this new loss is particularly useful when the data is noisy

(additive Gaussian noise in the observations), which is often the case in real-life environments. To support the multi-step loss, first we study its properties in two tractable cases: i) uni-dimensional linear system, and ii) two-parameter non-linear system. Second, we show in a variety of tasks (environments & datasets) that the models learned with this loss achieve a significant improvement in terms of the averaged R2-score on future prediction horizons. Finally, in the pure batch RL setting, we demonstrate that one-step models serve as strong baselines when dynamics are deterministic, while multi-step models would be more advantageous in the presence of noise, highlighting the potential of our approach in real-world applications.

3.1 Introduction

In RL we learn a control agent (or policy) by interacting with a dynamic system (or environment), receiving feedback in the form of rewards. This approach has proven successful in addressing some of the most challenging problems, as evidenced by [Sil+17; Sil+18; Mni+15; Vin+19]. However, RL remains largely confined to simulated environments and does not extend to real-world engineering systems. This limitation is, in large part, due to the scarcity of data resulting from operational constraints (physical systems with fixed time steps or safety concerns). MBRL, can potentially narrow the gap between RL and applications thanks to better sample efficiency.

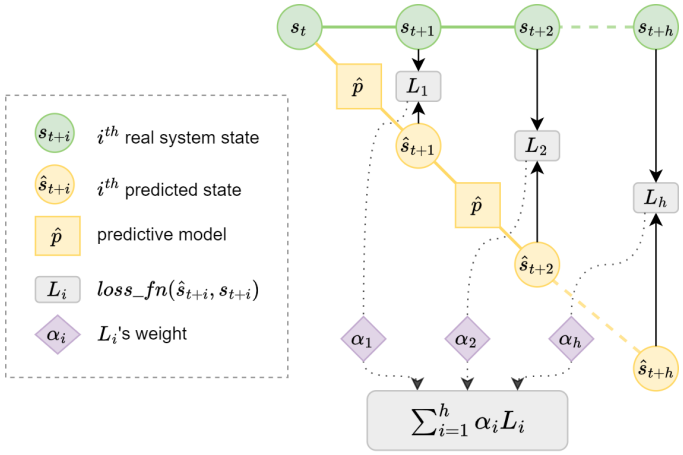


Figure 3.1: Schematic representation of the multi-step prediction framework using a one-step predictive model $\hat{\mathrm{P}}$. The diagram illustrates the iterative prediction of future states $\hat{\mathrm{s}}_{t+i}$, the computation of per-horizon losses L_i against real system states s_{t+1}, and the weighting of these losses α_i to optimize the predictive model over a horizon of h steps.

MBRL algorithms alternate between two steps: i) model learning, a supervised learning

problem to learn the dynamics of the environment, and ii) policy optimization, where a policy and/or a value function is updated by sampling from the learned dynamics. MBRL is recognized for its sample efficiency, as policy/value learning is conducted (either wholly or partially) from imaginary model rollouts (also referred to as background planning), which are more cost-effective and readily available than rollouts in the true dynamics [Jan+19]. Moreover, a predictive model that performs well out-of-distribution facilitates easy transfer to new tasks or areas not included in the model training dataset [Yu+20b].

While MBRL algorithms have achieved significant success, they are prone to *compounding errors* when planning over extended horizons [LPC22]. This issue arises due to the propagation of one-step errors, leading to highly inaccurate predictions at longer horizons. This can be problematic in real-world applications, as it can result in out-of-distribution states that may violate the physical constraints of the environment, and misguide planning. The root of compounding errors is in the nature of the models used in MBRL. Typically, these are one-step models that predict the next state based on the current state and executed action. Long rollouts are then generated by iteratively applying these models, leading to compounding errors.

To tackle this problem, our approach involves adjusting the training objective of these models to focus on optimizing for long-horizon error (Figure 3.1). This strategy is especially beneficial in presence of additive noise in observations, a context that mirrors real-world scenarios. Indeed, in situations where state perturbations occur due to unavoidable measurement errors or adversarial attacks [Zha+21; Sun+23], aiming to minimize the long-horizon error is advantageous. It enhances the signal-to-noise ratio by incorporating future information, thereby improving the practical effectiveness of the models.

The key contributions are the following.

- We propose a novel training objective for one-step predictive models consisting in a weighted sum of MSE losses at several future horizons.
- We use two tractable cases to demonstrate the advantages of our weighted multi-step loss. In a linear system, this loss allows for the identification of minimizers achieving a bias-variance trade-off. In the case of a two-parameter neural network, we show how important the weights are to achieve strong performance across multiple future horizons.
- Finally, we analyze the optimal weight configurations of the multi-step loss and show that models trained with this loss improve over the one-step baseline across diverse environments, datasets, and levels of noise.

3.2 Preliminaries (reminder)

The conventional framework used in RL is the finite-horizon MDP, defined by the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathrm{P}, \mathrm{r}, \rho_0, \gamma \rangle$. In this notation, $\mathcal{S}$ is the state space and $\mathcal{A}$ is the action space. The transition dynamics, which could be stochastic, are represented by $\mathrm{P} : \mathcal{S} \times \mathcal{A} \rightsquigarrow \mathcal{S}$[1]. The reward function is denoted by $\mathrm{r} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The initial state distribution is given by ρ_0, and the discount factor is represented by $\gamma \in (0, 1]$. The objective of RL is to identify a policy $\pi : \mathcal{S} \rightsquigarrow \mathcal{A}$. This policy aims at maximizing the expected sum of discounted rewards, denoted as $\eta(\pi, \mathcal{M}) := \mathbb{E}_{s_0 \sim \rho_0, \mathbf{a}_t \sim \pi, \mathbf{s}_{t>0} \sim p}\left[\sum_{t=0}^{H} \gamma^t \mathrm{r}(\mathbf{s}_t, \mathbf{a}_t)\right]$, where H represents the horizon of the MDP.

Model-based RL (MBRL) algorithms learn a model of the dynamics of the environment $\hat{\mathrm{P}}$ (and sometimes of the reward function $\hat{\mathrm{r}}$ as well) in a supervised way from data collected when interacting with the real system P. The loss function is typically the NLL $\mathcal{L}(\mathcal{D}; \hat{\mathrm{P}}) = -\frac{1}{N} \sum_{i=1}^{N} \log \hat{\mathrm{P}}(\mathbf{s}_{t+1}^i | \mathbf{s}_t^i, \mathbf{a}_t^i)$ for stochastic models or MSE in the case of deterministic ones. The learned model can subsequently be employed for policy search under the MDP $\widehat{\mathcal{M}} = \langle \mathcal{S}, \mathcal{A}, \hat{\mathrm{P}}, \mathrm{r}, \rho_0, \gamma \rangle$. This MDP shares the state and action spaces $\mathcal{S}, \mathcal{A}$, and the reward function r, with the true MDP $\mathcal{M}$, but uses the dynamics $\hat{\mathrm{P}}$ learned from the dataset $\mathcal{D}$. The policy $\hat{\pi} = \arg\max_\pi \eta(\pi, \widehat{\mathcal{M}})$ learned on $\widehat{\mathcal{M}}$ is not guaranteed to be optimal under $\mathcal{M}$ due to distribution shift and model error.

In this work, we also consider state-perturbed MDPs that fall under the formalism of **partially-observed Markov decision process (POMDP)**. Indeed, a POMDP is defined by the underlying MDP $\langle \mathcal{S}, \mathcal{A}, \mathrm{P}, \mathrm{r}, \rho_0, \gamma \rangle$, in addition to a set of possible observations Ω, and an observation function $f_o : \mathcal{S} \times \mathcal{A} \rightsquigarrow \Omega$ (where $o_t \sim f_o(\cdot | \mathbf{a}_{t-1}, \mathbf{s}_t)$). In practice, we consider systems with additive Gaussian noise in observations $\mathbf{o}_t = \mathbf{s}_t + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \Sigma)$, which is a special case of a POMDP. This formalism is closely related to *state-noisy* MDPs [Sun+23] and *state-adversarial* MDPs [Zha+21] where an attacker engineers the perturbation to intentionally decrease the performance of solvers.

The formal definition of the multi-step loss is given in Section 3.3. We then present the related work in Section 3.4. We study the properties of the multi-step loss in Section 3.5. This is done through two tractable cases: a uni-dimensional linear system in and a two-parameter non-linear system. Finally, the experimental setup and the performance evaluation of the models is discussed in Section 3.6.

[1] In this context, *to* denotes both probabilistic and deterministic mappings.

3.3 The multi-step loss

In MBRL, it is common to use a parametric model $\hat{\mathrm{P}}_\theta$ that predicts the state[2] one-step ahead $\hat{\mathbf{s}}_{t+1} \to \hat{\mathrm{P}}_\theta(\mathbf{s}_t, \mathbf{a}_t)$. We train this model to optimize the one-step predictive error $\mathcal{L}\Big(\mathbf{s}_{t+1}, \hat{\mathrm{P}}_\theta(\mathbf{s}_t, \mathbf{a}_t)\Big)$, where $\mathcal{L}$ is MSE or NLL in a supervised learning setting. To learn a policy, we use these models for predicting h steps ahead by applying a procedure called *rollout*.

Definition 3.3.1 (Rollout)**.** *We generate $\hat{\mathbf{s}}_{t+j} \to \hat{\mathrm{P}}_\theta(\hat{\mathbf{s}}_{t+j-1}, \mathbf{a}_{t+j-1})$ recursively for $j = 1, \dots, h$, to collect a trajectory $\tau = (\mathbf{s}_0, \mathbf{a}_0, \hat{\mathbf{s}}_1, \mathbf{a}_1, ..., \hat{\mathbf{s}}_j, \mathbf{a}_j, ...)$, where $(\mathbf{a}_t, \dots, \mathbf{a}_{t+h-1}) = \mathbf{a}_{t:t+h-1} = \mathbf{a}_\tau$ is either a fixed action sequence generated by planning or sampled from a policy $\mathbf{a}_{t+j} \to \pi(\mathbf{s}_{t+j})$ for $j = 1, \dots, h$, on the fly.*

Formally, let

$$
\begin{aligned}
\hat{\mathbf{s}}_t &= \mathbf{s}_t \\
\hat{\mathbf{s}}_{t+j-1} &\to \hat{\mathrm{P}}_\theta(\hat{\mathbf{s}}_{t+j-2}, \mathbf{a}_{t+j-2}) \\
\hat{\mathrm{P}}^j_\theta(\mathbf{s}_t, \mathbf{a}_{t:t+j-1}) &= \hat{\mathrm{P}}_\theta\Big(\hat{\mathbf{s}}_{t+j-1}, \mathbf{a}_{t+j-1}\Big) \quad \text{for } j = 1, \dots, h
\end{aligned}
$$

Using $\hat{\mathrm{P}}^h_\theta(\mathbf{s}_t, \mathbf{a}_{t:t+h-1})$ to estimate $\mathbf{s}_{t+h}$ is problematic for two reasons:

- A distribution mismatch occurs between the inputs that the model was trained on $(\mathbf{s}_{t+1} \to p(\mathbf{s}_t, \mathbf{a}_t))$ and the inputs the model is being evaluated on $(\hat{\mathbf{s}}_{t+1} \to \hat{\mathrm{P}}_\theta(\mathbf{s}_t, \mathbf{a}_t)$[Tal14; Tal17].
- The predictive error (and the modeled uncertainty in the case of stochastic models) will propagate through the successive model calls, leading to compounding errors [LPC22; Tal14; VHB15b].

To mitigate these issues, we study models that, given the full action sequence $\mathbf{a}_{t:t+h-1}$, learn to predict the state $\mathbf{s}_{t+h}$ by recursively predicting the intermediate states $\mathbf{s}_{t+j}$, for $j = 1, \dots, h$. We address this problem through the use of a weighted multi-step loss that accounts for the predictive error at different future horizons.

[2] In this section, we do not make the distinction between states $\mathbf{s}$ and observations $\mathbf{o}$ as the definition of the multi-step loss is independent of the underlying MDP.

Definition 3.3.2 (Weighted multi-step loss)**.** *Given horizon-dependent weights* $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_h)$ *with* $\sum_{i=1}^{h} \alpha_i = 1$*, a one-step loss function* $\mathcal{L}$*, an initial state* $\mathbf{s}_t$*, an action sequence* $\mathbf{a}_\tau = \mathbf{a}_{t:t+h-1}$*, and the real (groundtruth) visited states* $\mathbf{s}_\tau = \mathbf{s}_{t+1:t+h}$*, we define the weighted multi-step loss as*

$$\mathcal{L}_{\boldsymbol{\alpha}}^{h}\Big(\mathbf{s}_\tau, \hat{\mathrm{P}}_\theta(\mathbf{s}_t, \mathbf{a}_\tau)\Big) = \sum_{j=1}^{h} \alpha_j \mathcal{L}\Big(\mathbf{s}_{t+j}, \hat{\mathrm{P}}_\theta^{j}(\mathbf{s}_t, \mathbf{a}_{t:t+j-1})\Big).$$

The dependency of $\mathcal{L}_{\boldsymbol{\alpha}}^{h}$ on h is omitted in the rest of the chapter when h is clear from the context. Furthermore, the loss $\mathcal{L}$ used in the multi-step loss $\mathcal{L}_{\boldsymbol{\alpha}}$ will be considered to be the MSE.

Algorithm 14 shows the training procedure of multi-step models. We emphasize that unlike popular methods in the existing literature (teacher forcing [WZ89], hallucinated replay [Tal14] and data-as-demonstrator [VHB15b]), which consists in augmenting the training data with predicted states, our method consists in back-propagating the gradient of the loss through the successive compositions of the model $\hat{\mathrm{P}}_\theta$ figuring in $\mathcal{L}_{\boldsymbol{\alpha}}$ (closer in concept to Back-Propagation Through Time BPTT).

Algorithm 14 Training a model using the multi-step loss

Input: model $\hat{\mathrm{P}}_\theta$, trajectory $\tau = (\mathbf{s}_{t+i}, \mathbf{a}_{t+i})_{i=0}^{h}$, loss function $\mathcal{L}$, horizon h, normalized weights $\{\alpha_i\}_{i=1}^{h}$
Initialize the loss $l = 0$
$\mathbf{s} = \mathbf{s}_t$
for $i = 0$ **to** $h - 1$ **do**
  Sample from the model $\hat{\mathbf{s}}_{t+i+1} \to \hat{\mathrm{P}}_\theta(\mathbf{s}, \mathbf{a}_{t+i})$
  Update the loss $l \mathrel{+}= \alpha_i \cdot \mathcal{L}(\hat{\mathbf{s}}_{t+i+1}, \mathbf{s}_{t+i+1})$
  $\mathbf{s} = \hat{\mathbf{s}}_{t+i+1}$
end for
Update θ using a gradient step minimizing l

3.4 Related Work

The premises of multi-step dynamics modeling can be tracked back to early work about temporal abstraction [SPS99; PSS98] and mixture of timescale models in tabular MDPs [PS97; Sin92; SP85; Sut95]. These works study fixed-horizon models that learn an abstract dynamics mapping from initial states to the states j steps ahead. A different approach consists in optimizing the multi-step prediction error of single-step models that

are used recursively, which we study here. This approach has been studied for recurrent neural networks (RNNs) and is referred to as *teacher forcing* [Lam+16; Ben+15a; Hus15; Pin88; WZ89]. The idea consists in augmenting the training data with predicted states. More recent works have built on this idea [AGN05; Tal14; Tal17; VHB15b]. These methods, albeit optimizing for future prediction errors, assume the independence of the intermediate predictions on the model parameters, making them more of a data augmentation technique than a proper optimization of multi-step errors.

The closest works to ours which also consider the intermediate predictions to be dependent on the model parameters are [Lut+21], [Byr+21] and [Xu+18]. These works all use an equally-weighted multi-step loss whereas we emphasize on the need of having a weighted multi-step loss. [Nag+18] only use an equally-weighted multi-step loss for validation, which is also common in the time series literature [TON95; FR98; McN02; BB12; Ben+12; CGG21]. [Lut+21] and [Xu+18] find that only small horizons ($h = 2, 3, 5$) yield an improvement over the baseline, which we suggest is due to using equal weights in the multi-step loss. [Byr+21] successfully use $h = 10$ with equal weights in the context of model-predictive control (MPC). However, they consider it as a fixed design choice and might have tailored their approach accordingly. To the best of our knowledge we are the first work highlighting the importance of the weighting mechanism in balancing the multi-step errors. Futhermore, we particularly stress the importance of having a weighted multi-step loss for state-noisy MDPs and study the impact of the noise level. We believe that such a setting is still under-studied even though previous model-free methods have been proposed [Hes+23; Sun+23; Zha+21; Pat+17] and a more recent work [YX24b] presents a model-based approach using diffusion models.

3.5 Understanding the multi-step loss: two case studies

3.5.1 Uni-dimensional linear system

The first case consists in studying the solutions of the multi-step loss for $h = 2$ in the case of an uncontrolled linear (discrete) dynamical system with additive Gaussian noise, and a linear model. With such a simple formulation, we benefit from the fact that the optimization is tractable and closed-form solutions can be obtained for each $\boldsymbol{\alpha} \in [0, 1]$. We start by defining the system and the model.

Definition 3.5.1. (*Uni-dimensional linear system with additive Gaussian noise*). *For an initial state $s_0 \in \mathbb{R}$ and an unknown parameter $\theta_{true} \in (-1, 1)$ (for stability) we define the*

transition function and observations as

$$\mathbf{s}_{t+1} = \theta_{true} \cdot \mathbf{s}_t$$
$$o_{t+1} = \mathbf{s}_{t+1} + \epsilon_{t+1} \text{ with } \epsilon_{t+1} \sim \mathcal{N}(0, \sigma^2) \text{ and } \sigma \in \mathbb{R}$$

Definition 3.5.2. (*Linear model*). *For an initial state $s_0 \in \mathbb{R}$ and a parameter $\theta \in \mathbb{R}$ that we learn by minimizing the multi-step loss for $h = 2$, we define the linear model as $\hat{\mathbf{s}}_{t+1} = \hat{\mathrm{P}}_\theta(\mathbf{s}_t) = \theta \cdot \mathbf{s}_t$*

In this setup, the multi-step loss for $h = 2$ boils down to a polynomial in the model's parameter θ:

$$L_\alpha\Big(\boldsymbol{o}_\tau, \hat{\mathrm{P}}_\theta(\mathbf{s}_t)\Big) = \alpha(\theta\mathbf{s}_t - o_{t+1})^2 + (1-\alpha)(\theta^2\mathbf{s}_t - o_{t+2})^2$$

where $\boldsymbol{o}_\tau = (o_{t+1}, o_{t+2})$. The aim of our study is to analyze the statistical properties of $\hat{\theta}(\alpha) \in \arg\min_\theta L_\alpha\Big(\boldsymbol{o}_\tau, \hat{\mathrm{P}}_\theta(\mathbf{s}_t)\Big)$ for different values of α and different values of the observational noise scale[3] σ.

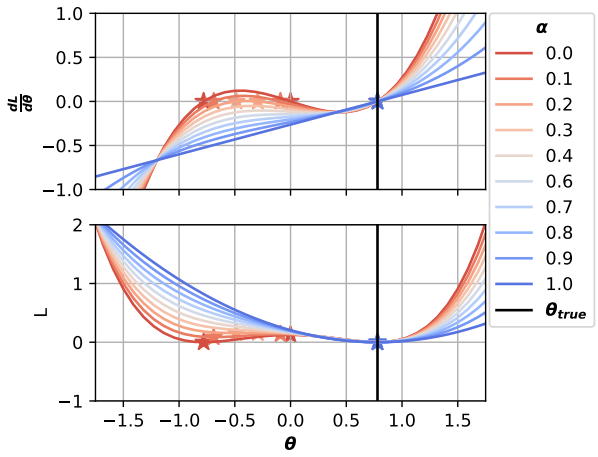


Figure 3.2: The loss function and its derivative for different values of θ and α, in absence of noise ($\sigma = 0$). In this figure, θ_{true} is fixed to a randomly selected value, $\theta_{true} = 0.78$. The roots of the derivative are highlighted with stars.

Figure 3.2 shows the loss function curve and its critical points for different values of α. The minimizers $\hat{\theta}(\alpha)$ can be obtained by solving the polynomial equation $dL_\alpha/d\theta = 0$. When $\alpha \in (0, 1)$, we compute the roots of the cubic polynomial equation using Cardano's

[3] *noise* and *noise scale* are used interchangeably. In practice, σ is computed as a percentage (e.g. 2%) of the state space width.

formulas. These latter include at least one real root ($\alpha \geq 0.3$ in Figure 3.2) and two (potentially real) conjugate complex roots ($\alpha < 0.3$ in Figure 3.2).

In the rest of the experiments, we fix a dataset of initial states $\mathcal{S}_0$ and sample K times a two-step transition, yielding a dataset $\mathcal{D} = (\mathcal{S}_0, \mathcal{O}_1^j, \mathcal{O}_2^j)_{j=1,\dots,K}$. This dataset showcases different realizations of the observational noise, which is sampled i.i.d. from a Gaussian distribution $\mathcal{N}(0, \sigma^2)$.

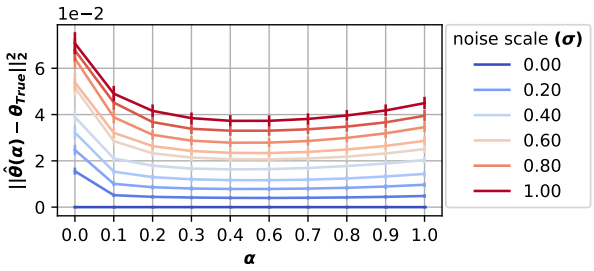


Figure 3.3: The distance between the true parameters and the optimal parameters for different values of α and noise scales.

We then compare the distance between the minimizers of the loss function with different values of α and the true parameter θ_{true}. When there is more than one root, we assume access to the sign of θ_{true} so that we can choose the correct estimator $\hat{\theta}(\alpha)$. Figure 3.3 shows that as the noise increases, the vanilla MSE loss estimator ($\alpha = 1$) is not the best estimator with respect to the distance to the true parameter θ_{true}. Interestingly, the best solution is obtained for $\alpha \in (0, 1)$.

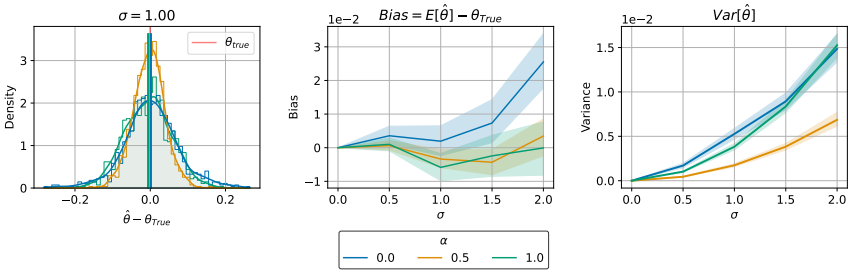


Figure 3.4: The left panel shows the density distribution of $\hat{\theta} - \theta_{true}$ for a fixed σ of 1.0. The middle panel delineates the bias of the estimator, defined as $E[\hat{\theta}] - \theta_{true}$, across varying levels of σ, and weights $\alpha \in \{0, 0.5, 1\}$ indicated by color. The right panel presents the variance of the estimator, $Var[\hat{\theta}]$, as a function of σ for the same set of α values. The shaded regions represent the 95% bootstrap confidence intervals across ten θ_{true} values and 100 Monte Carlo simulations.

To understand the observed results, we compute the closed-form solutions of the multi-step loss in the case of $\alpha = 0$ and $\alpha = 1$:

Proposition 3.5.1. ($\alpha = 1$). *Given a transition* $(\mathbf{s}_t \neq 0, o_{t+1})$ *from the linear system and a linear model with parameter* θ*, the minimizer of the* $\alpha = 1$ *multi-step loss can be computed as:*

$$\hat{\theta}_1 = \frac{o_{t+1}}{\mathbf{s}_t} = \theta_{true} + \frac{\epsilon_{t+1}}{\mathbf{s}_t}$$

Proposition 3.5.2. ($\alpha = 0$). *Given a transition* $(\mathbf{s}_t \neq 0, o_{t+1}, o_{t+2})$ *from the linear system, a linear model with parameter* θ*, the sign of the true parameter (for instance* $\theta_{true} > 0$*), and assuming its existence* ($\frac{o_{t+2}}{\mathbf{s}_t} > 0$)*, the minimizer of the* $\alpha = 0$ *multi-step MSE loss can be computed as:*

$$\hat{\theta}_0 = \sqrt{\frac{o_{t+2}}{\mathbf{s}_t}} = \sqrt{\theta^2_{true} + \frac{\epsilon_{t+2}}{\mathbf{s}_t}}$$

Remark 3.5.1. *For ease of notation in proposition 3.5.1 and proposition 3.5.2, we compute the solutions given only one transition* $(\mathbf{s}_t, o_{t+1}, o_{t+2})$*. In practice, one minimizes the empirical risk based on a training dataset of size* N*:* $\mathcal{D} = \{(\mathbf{s}_{i,t}, o_{i,t+1}, o_{i,t+2})\}_{i=1:N}$*, in which case the closed-form solutions become:*

$$\begin{cases} \hat{\theta}_1 = \theta_{true} + \frac{\sum_{i=1}^N \epsilon_{i,t+1}}{\sum_{i=1}^N \mathbf{s}_{i,t}} \\ \hat{\theta}_0 = \sqrt{\theta^2_{true} + \frac{\sum_{i=1}^N \epsilon_{i,t+2}\mathbf{s}_{i,t}}{\sum_{i=1}^N \mathbf{s}^2_{i,t}}} \end{cases}$$

On the one hand, we observe that while $\hat{\theta}_1$ is an unbiased estimator of θ_{true} ($E_{\epsilon_{t+1}\sim\mathcal{N}(0,\sigma^2)}[\hat{\theta}_1] = \theta_{true}$), its variance grows linearly with the noise scale: $\mathrm{Var}_{\epsilon_{t+1}\sim\mathcal{N}(0,\sigma^2)}[\hat{\theta}_1] = \frac{\sigma^2}{\mathbf{s}^2_t}$. On the other hand, $\hat{\theta}_0$ is a potentially biased estimator, but with a smaller variance if $\theta_{true} \gg 1$ ($\frac{\epsilon_{t+2}}{4\theta^2_{true}\mathbf{s}^2_t} \approx 0$). In this case we can use a first-order Taylor expansion to approximate $\mathrm{Var}[\hat{\theta}_0] = \mathrm{Var}_{\epsilon_{t+2}\sim\mathcal{N}(0,\sigma^2)}[\hat{\theta}_0]$:

$$\begin{aligned} \mathrm{Var}[\hat{\theta}_0] &= \mathrm{Var}\Big[\sqrt{\theta^2_{true} + \frac{\epsilon_{t+2}}{\mathbf{s}_t}}\Big] \\ &\approx \theta^2_{true}\mathrm{Var}\Big[1 + \frac{\epsilon_{t+2}}{2\theta^2_{true}\mathbf{s}_t}\Big] = \frac{\sigma^2}{4\theta^2_{true}\mathbf{s}^2_t} \\ &\leq \mathrm{Var}[\hat{\theta}_1] \end{aligned}$$

For intermediate models ($\alpha \in (0, 1)$), and in general when the conditions of the last result do not necessarily hold, we use Monte Carlo simulations to compare the variance of $\hat{\theta}_{\alpha\in\{0,0.5,1\}}$. Figure 3.4 shows the variance reduction brought by the multi-step loss when $\alpha = 0.5$. It is also noticeable that, up to the noise scales considered in this experiment, $\alpha = 0$ generates a large bias (which matches the theoretical insights), while no significant bias is observed for the estimator with $\alpha = 0.5$. We conclude that in the case of a

noisy linear system, the multi-step MSE loss minimizer with $\alpha = 0.5$ is a statistical estimator that (empirically) has a smaller variance and a comparable bias to the one-step loss minimizer. It is worth noticing that when the noise is non-zero the best solution for the one-step MSE is obtained for $\alpha \in (0, 1)$ and not $\alpha = 1$ (which corresponds to optimizing exactly the one-step MSE).

In this linear case, we had access to closed-form solutions. However, Figure 3.2 shows that choosing the multi-step MSE loss as an optimization objective introduces additional critical points where a gradient-based optimization algorithm might get stuck. We now empirically study the optimization process in the case of a two-parameter neural network.

3.5.2 Two-parameter non-linear system

As an attempt to get closer to a realistic MBRL setup where neural networks are used for dynamics learning, we study a non-linear dynamical system using a two-parameters neural network model:

Definition 3.5.3. (*Two-parameter non-linear system with additive Gaussian noise*). *For an initial state $s_0 \in \mathbb{R}$ and unknown parameters $\boldsymbol{\theta}^{true} = (\theta_1^{true}, \theta_2^{true}) \in \mathbb{R}^2$ we define the transition function and observations as*

$$\mathbf{s}_{t+1} = \theta_1^{true} \cdot sigmoid(\theta_2^{true} \cdot \mathbf{s}_t)$$
$$o_{t+1} = \mathbf{s}_{t+1} + \epsilon_{t+1} \text{ with } \epsilon_{t+1} \to \mathcal{N}(0, \sigma^2) \text{ and } \sigma \in \mathbb{R}$$

Definition 3.5.4. (*Two-parameters neural network model*). *A single-neuron two-layer (without bias) neural network. We denote its parameters $\boldsymbol{\theta} = (\theta_1, \theta_2) \in \mathbb{R}^2$:*

$$\hat{\mathbf{s}}_{t+1} = \hat{\mathrm{P}}(\mathbf{s}_t) = \theta_1 \cdot sigmoid(\theta_2 \cdot \mathbf{s}_t)$$

To get a broader idea about the optimization challenges in this setup, we show the loss landscape for different values of α and noise scales σ in Section B.4.2. Regarding the gradient-based optimization, we compare the solution reached after training in terms of the one-step and two-step *validation* MSE losses (the details of the experiment are given in Section B.4.2).

Figure 3.5 shows that even in the presence of noise, compared to the linear case, the best value of α for the one-step MSE is $\alpha = 1$. This can be explained by the use of lower levels of noise in the simulations than the linear case. The intermediate models obtained for $\alpha \in \{0.25, 0.5, 0.75\}$ represent a trade-off between the one-step and two-step MSEs. The average of the two MSEs, which we use in the experiment section to assess the overall

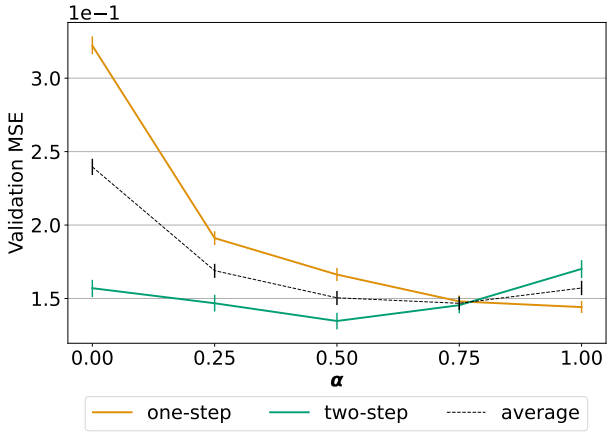


Figure 3.5: The validation one-step MSE L_1 (in yellow), the validation two-step MSE L_0 (in green) and the average of these two MSEs (dashed black line) for different values of α. The error bars represent the 95% bootstrap confidence intervals across 2 optimizers, 3 initialization distributions, 10 initial points, 3 noise levels, and 10 Monte Carlo simulations.

quality of our models over a range of horizons, achieves its minimum for $\alpha = 0.75$. Notice that the validation losses are only used for evaluation, and do not match the training loss which depends on α.

3.6 Experiments & results

In this section, we evaluate the performance of models trained with the multi-step loss $L_{\boldsymbol{\alpha}}$ for different values of the horizon h and different noise levels of the dynamics. We first describe the multi-step loss hyperparameters: the horizon h and the weights $\boldsymbol{\alpha}$. For the evaluation of the models we consider both a static and a dynamic setup. The *static evaluation* denotes the evaluation of dynamics models in terms of predictive error in held-out test datasets. For the *dynamic evaluation* we will consider the offline MBRL setting [Lev+20b] where the goal is to learn a policy from a given dataset without interacting further with the environment. The evaluation is done on three classical RL environments (*Cartpole swing-up*, *Swimmer* and *Halfcheetah*) and various datasets (eight in total) collected with different behavior policies (*random*, *medium*, *full_replay* and *mixed_replay*) on these environments. The details of these tasks are provided in Section B.1. For all these tasks, we use the same neural network model for $\hat{p}_\theta$. Implementation details for the model are given in Section B.2.

3.6.1 Hyperparameters of the multi-step loss

The multi-step loss depends on the following hyperparameters: the loss horizon h and the corresponding weights $\{\alpha_i\}_{i=1}^{h}$.

The horizon h

The horizon of the loss is the maximal prediction horizon considered in the loss. Typically, in this chapter we consider $h \in \{2, 3, 4, 10\}$ for the multi-step loss and $h = 1$ for the single-step baseline.

The weights α

For a horizon h, the weights $\boldsymbol{\alpha}$ live in the $(h-1)$-dimensional probability simplex, which means that the search space is $(0, 1)^{h-1}$. As the MSE has been shown to grow exponentially with the horizon when applying one-step models recursively (Theorem 1 of [VHB15b]) we choose an exponentially parameterized profile for the weights. The objective is to give each MSE term the same importance when we optimize the MSE loss. Given a weighted multi-step loss $L_{\boldsymbol{\alpha}}$ with horizon h, weights $\boldsymbol{\alpha}$, and $\beta > 0$, the exponentially parameterized weights are defined as:

$$\alpha_i = \underbrace{\left(\frac{1-\beta}{1-\beta^{h+1}}\right)}_{\text{normalization constant}} \cdot \beta^i \quad \text{for } i \in \{1, \dots, h\}.$$

Applying this exponential parametrization of the weights, we further reduce the search space dimension. In our experiments, we still consider values greater than 1 in the grid, values corresponding to an increase of the weights over the horizon. This approach is used to analyze whether certain settings would benefit from focusing on large horizons.

3.6.2 Static evaluation with the R2 metric

The commonly used metrics for the static evaluation are the standard mean squared error (MSE) or the explained variance (R2) which we prefer over the MSE because it is normalized and can be aggregated over multiple dimensions. In our attempt to reduce compounding errors in MBRL, we are especially interested in the long-term predictive error of models. For each horizon h, the error is computed by considering all the sub-trajectories of size h from the test dataset. The predictions are computed by calling the model recursively h times (using the ground truth actions of the sub-trajectories) and

the average R2 score at horizon h, R2(h), averaged over the sub-trajectories is reported: given a dataset $\mathcal{D} = \{(\mathbf{s}_{i,t}, a_{i,t}, \mathbf{s}_{i,t+1}, \dots)\}_{i=1,\dots,N}$, a predictive model $\hat{\mathrm{P}}_\theta$, for $h = 1, \dots, H$:

$$\mathrm{R2}(h) = \frac{1}{d_s} \sum_{j=1}^{d_s} 1 - \frac{\frac{1}{N} \sum_{i=1}^{N} \left(\mathbf{s}^j_{i,t+h} - \hat{\mathrm{P}}^h_\theta(\mathbf{s}_{i,t}, \mathbf{a}_{i,t:t+h})^j \right)^2}{\frac{1}{N} \sum_{i=1}^{N} \left(\mathbf{s}^j_{i,t+h} - \bar{s}^j_{t+h} \right)^2}.$$

We also report the average R2 score, $\overline{R2}(H)$, over all prediction horizons from 1 to H.

$$\overline{R2}(H) = \frac{1}{H} \sum_{h=1}^{H} \mathrm{R2}(h).$$

For the weights of the loss we perform a grid search over β values, selecting the value giving the best $\overline{R2}(H)$ averaged over 3 cross-validation folds.

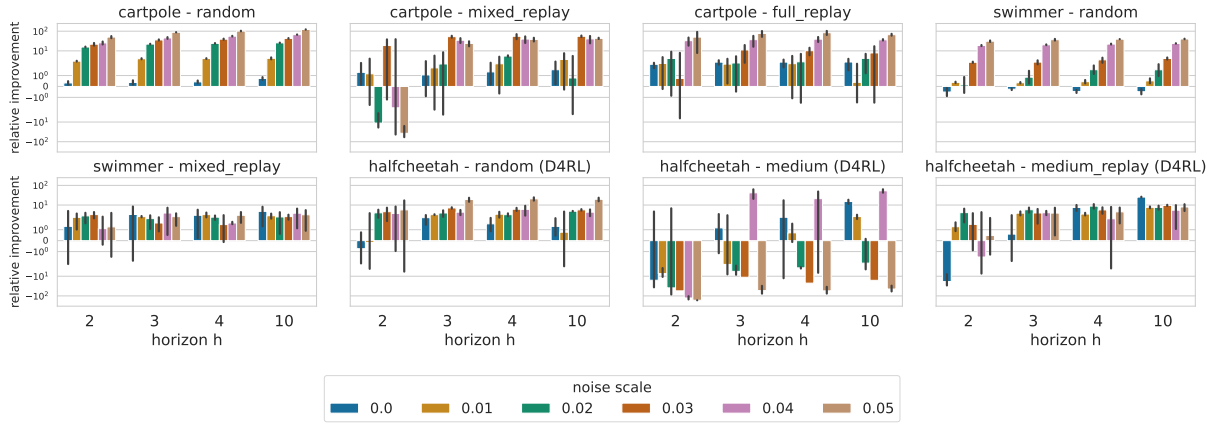


Figure 3.6: The series of bar plots display the relative improvement with respect to the $h = 1$ baseline, in the test $\overline{R2}(50)$ metric for various environments, and datasets. Performance is evaluated over loss horizons h with the relative improvement measured on a logarithmic scale. The relative noise scale, ranging from 0.0 to 0.05, is color-coded and represents a ratio of the range of the state space, for each dataset. The error bars indicate the 95% confidence intervals (mean $\pm$ 1.96 $\cdot$ standard error) obtained with the three cross-validation folds.

Figure 3.6 shows the relative improvement in percentage over the one-step model of the $\overline{R2}(50)$ metric for the different datasets (the absolute values of the metric and the full $R2(h)$ curves are shown in Section B.4.3). For most of the datasets (all the *Cartpole* and *Swimmer* datasets, and *Halfcheetah random* and *medium_replay*) the benefit of using the multi-step loss when there is noise is clear and for most of them (all the *Cartpole* datasets, *Swimmer random* and *Halfcheetah random*) the larger the noise the higher the benefit. The impact of the horizon h of the loss is less clear although for some datasets increasing the horizon h as the noise increases also helps. This result is more mitigated on *Halfcheetah*

medium which we suggest is due to the optimization process converging to sub-optimal solutions. We also note that even if the multi-step loss is using $h = 10$, because of the weights, the multi-step loss could finally be close to the multi-step losses obtained with $h = 2$, 3 or 4, which would mean that a smaller horizon is sufficient. This is an advantage of our flexible multi-step loss that can adapt its horizon thanks to the selection of the best weights

We now study the best weights obtained for each multi-step loss ($h = 2$, 3, 4, and 10) and each noise scale. First, we observe from Table 3.1 that compared to what is usually done in the literature, the best weights are not uniform. Second, we also notice an increase of the parameter β with the noise scale. This finding supports the idea that multi-step models are increasingly needed when incorporating information from the future is crucial to achieve better performance. We also discuss the effective horizon obtained with the weights in Section B.4.3.

Table 3.1: Best β values found with a grid search for each horizon and each noise scale. The values are averaged over the eight datasets.

	horizon h			
	2	3	4	10
σ (%)	β			
0	0.81	0.45	0.41	0.46
1	0.56	0.76	0.51	0.47
2	0.86	0.72	0.78	0.54
3	0.67	1.21	0.81	0.50
4	0.74	1.01	0.74	0.55
5	1.33	0.83	0.97	0.51

3.6.3 Dynamic evaluation: offline MBRL

We consider the offline setting where given a set of N trajectories $\mathcal{D} = \{(\mathbf{s}_t^i, \mathbf{a}_t^i, \mathbf{s}_{t+1}^i, \ldots)\}_{i=1}^N$, the goal is to learn a policy maximizing the return in a single shot, without further interacting with the environment.

We consider a *Dyna*-style agent that learns a parametric model of the policy based on data generated from the learned dynamics model $\hat{\mathrm{P}}_\theta$. Specifically, we train a Soft Actor-Critic (SAC) [Haa+18b] with short rollouts on the model a la MBPO [Jan+19], a popular MBRL algorithm. We then rely on model predictive control (MPC) at decision-time where the action search is guided by the SAC policy. The details of this agent are given in Section B.1.3.

The experiments are run on the *Cartpole mixed_replay* dataset without noise (noise scale of 1%). The reason is that the range of episode returns spanned by this dataset makes us hope that it is possible to learn a model that is good enough to learn a successful policy (for which episode returns are higher than 800). The distribution of the returns on the random and full replay datasets makes it more challenging to learn a successful policy. The goal here is not to study a new offline MBRL algorithm, avoiding unknown regions of the state-action space, but studying the improvement that can be obtained with the multi-step loss when varying the horizon h. Finally, in order to isolate the effect of dynamics learning, we assume that the reward function is a known deterministic function of the observations.

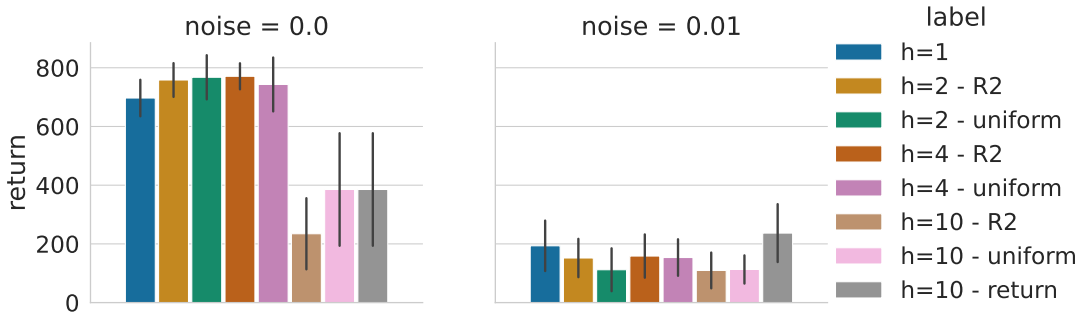


Figure 3.7: Returns of the agents trained on multi-step models in the *Cartpole mixed_replay* task. The error bars indicate the 95% confidence intervals (mean $\pm$ 1.96 $\cdot$ standard error).

As it is acknowledged that static evaluation metrics may not always align with the final return of agents (see for instance [Lam+21a]) we also include the performance of the models trained on the 10-step loss with weights tuned to maximize the return, using the same grid search as for the R2.

Without noise. Figure 3.7 shows the return obtained by the different models on the *Cartpole mixed_replay* task. Against a strong baseline ($h = 1$) with a near 700 return, it is observed that the multi-step models exhibit marginally superior performance, especially for h values of 2 and 4. However, this trend is not maintained for $h = 10$, indicating that larger horizons may not be beneficial in this context. Additionally, the study compares these outcomes with those obtained using uniform weights, a common approach in most related research. As depicted in Figure 3.7, the returns for uniform weight variants at h values of 2 and 4 are comparable to those of the models selected based on the $\overline{R2}(50)$ metric. While uniform weights help the $h = 10$ model reach a better return, it remains sub-optimal with respect to the baseline. These findings confirm a common result in

the literature [Lut+21; Xu+18], stating that the multi-step loss is only useful for small values of h.

With noise. In the presence of noise, we observe that the performance of all models, including the baseline, significantly decreases. Notably, the optimal multi-step models do not demonstrate any improvement in this setting, a finding that also holds true for the variant with uniform weights. We found that $\beta = 0.3$ maximizes the return for the 10-step loss and leads to a marginal improvement over the baseline model. This finding challenges the common practice of using uniform weights in related research, suggesting that a more tailored approach is necessary for each specific task. We believe that a better hyperparameter search for the multi-step loss weights could lead to better performance of the corresponding multi-step models.

3.7 Conclusion

In this chapter, we introduce a weighted multi-step loss that leads to models exhibiting significant improvements in the average R2 score over future horizons in various datasets derived from popular RL environments with noisy dynamics. These outcomes align with our analysis of two tractable scenarios: models optimized for the multi-step loss outperform those focused on minimizing the one-step loss in the presence of noise. The insights from dynamic evaluation present a more complex picture. Specifically, we found that in absence of noise, the multi-step loss slightly improved the return of an already strong one-step baseline. However, a noticeable improvement in the noisy scenario would require an extensive search for optimal hyperparameters, including the weights and, implicitly, the prediction horizon.

Chapter 4

Zero-shot Model-based Reinforcement Learning using Large Language Models

Contents

The emerging zero-shot capabilities of Large Language Models (LLMs) have led to their applications in areas extending well beyond natural language processing tasks. In RL, while LLMs have been extensively used in text-based environments, their integration with continuous input spaces remains understudied. In this chapter, we investigate how pre-trained LLMs can be leveraged to predict in-context the dynamics of continuous MDPs. We identify handling multivariate data and incorporating the control signal as key challenges that limit the potential of LLMs deployment in this setup

and propose Disentangled In-Context Learning (DICL) to address them. We present proof-of-concept applications in two RL settings: model-based policy evaluation and data-augmented off-policy RL, supported by theoretical analysis of the proposed methods. Our experiments further demonstrate that our approach produces well-calibrated uncertainty estimates.

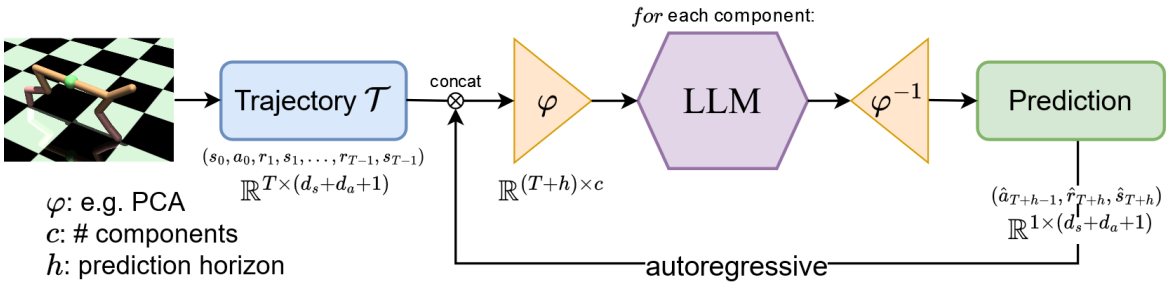


Figure 4.1: **The DICL Framework.** DICL projects trajectories into a disentangled feature space before performing zero-shot forecasting using a pre-trained LLM and in-context learning.

4.1 Introduction

The rise of large language models (LLMs) has significantly impacted the field of NLP. LLMs [Bro+20; HL23; DL24], which are based on the transformer architecture [Vas+17], have redefined tasks such as machine translation [Bro+20], sentiment analysis [Zha+23d], and question answering [RRS20; PS24] by enabling machines to understand and generate human-like text with remarkable fluency. One of the most intriguing aspects of LLMs is their emerging capabilities, particularly in-context learning (ICL) [Osw+23]. Through ICL, an LLM can learn to perform a new task simply by being provided examples of the task within its input context, without any gradient-based optimization. This phenomenon has been observed not only in text generation but also in tasks such as image classification [AAG24; Zhe+24] and even solving logic puzzles [Gia+24], which is unexpected in the context of the standard statistical learning theory. To our knowledge, ICL capabilities of pre-trained LLMs have been only scarcely explored in RL [Wan+23] despite the demonstrated success of the former in understanding the behavior of deterministic and chaotic dynamical systems [Liu+24b].

In this chapter, we show how ICL with pre-trained LLMs can improve the sample efficiency of RL, with two proof-of-concepts in policy evaluation and data-augmented off-policy RL. Following the dynamical system perspective on ICL introduced in [Li+23] and experimentally studied in [Liu+24b], we use the observed trajectories of a given agent to predict its future state and reward in commonly used RL environments. To

achieve this, we solve two crucial challenges related to considering continuous state-space MDPs: 1) incorporating the action information into the LLM's context and 2) handling the interdependence between the state-actions dimensions, as prior approaches were known to treat multivariate data's covariates independently. Our framework, DICL (Disentangled In-Context Learning), is summarized in Figure 4.1. The core idea of DICL is to apply a feature space transformation, denoted as φ, which captures the interdependencies between state and action features in order to disentangle each dimension. Subsequently, a LLM is employed to forecast each component independently in a zero-shot manner through in-context learning. Finally, the predictions are transformed back to the original trajectory space using the inverse transformation φ^{-1}.

Our approach leads to several novel insights and contributions, which we summarize as follows:

1. *Methodological.* We develop a novel approach to integrate state dimension interdependence and action information into in-context trajectories. This approach, termed Disentangled In-Context Learning (DICL), leads to a new methodology for applying ICL in RL environments with continuous state spaces. We validate our proposed approach on tasks involving proprioceptive control.
2. *Theoretical.* We theoretically analyze the policy evaluation algorithm resulting from multi-branch rollouts with the LLM-based dynamics model, leading to a novel return bound.
3. *Experimental.* We show how the LLM's MDP modeling ability can benefit two RL applications: policy evaluation and data-augmented offline RL. Furthermore, we show that the LLM is a calibrated uncertainty estimator, a desirable property for MBRL algorithms.

Organization of the chapter. The chapter is structured as follows: Section 4.2 introduces the main concepts from the literature used in our work (while a more detailed related work is deferred to Section C.2). We then start our analysis in Section 4.3.1, by analyzing LLM's attention matrices. DICL is presented in Section 4.3.3, while Section 4.4 contains different applications of the proposed method in RL, along with the corresponding theoretical analysis. Finally, Section 4.5 provides a short discussion and future research directions triggered by our approach.

4.2 Background knowledge (reminder)

Reinforcement Learning (RL). The standard framework of RL is the infinite-horizon **Markov decision process (MDP)** $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathrm{P}, \mathrm{r}, \mu_0, \gamma \rangle$ where $\mathcal{S}$ represents the state

space, $\mathcal{A}$ the action space, $\mathrm{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ the (possibly stochastic) transition dynamics, $\mathrm{r} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ the reward function, μ_0 the initial state distribution, and $\gamma \in [0, 1]$ the discount factor. The goal of RL is to find, for each state $\mathbf{s} \in \mathcal{S}$, a distribution $\pi(\mathbf{s})$ over the action space $\mathcal{A}$, called the *policy*, that maximizes the expected sum of discounted rewards $\eta[\pi] := \mathbb{E}_{\mathbf{s}_0 \sim \mu_0, \mathbf{a}_t \sim \pi, \mathbf{s}_{t>0} \sim \mathrm{P}^t}[\sum_{t=0}^{\infty} \gamma^t \mathrm{r}(\mathbf{s}_t, \mathbf{a}_t)]$. Under a policy π, we define the state value function at $\mathbf{s} \in \mathcal{S}$ as the expected sum of discounted rewards, starting from the state $\mathbf{s}$, and following the policy π afterwards until termination: $V^{\pi}(\mathbf{s}) = \mathbb{E}_{\mathbf{a}_t \sim \pi, \mathbf{s}_{t>0} \sim \mathrm{P}^t}[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \mid \mathbf{s}_0 = \mathbf{s}]$.

Model-based RL (MBRL). MBRL algorithms address the supervised learning problem of estimating the dynamics of the environment $\hat{\mathrm{P}}$ (and sometimes also the reward function $\hat{\mathrm{r}}$) from data collected when interacting with the real system. The model's loss function is typically the log-likelihood $\mathcal{L}(\mathcal{D}; \hat{\mathrm{P}}) = \frac{1}{N} \sum_{i=1}^{N} \log \hat{\mathrm{P}}(\mathbf{s}_{t+1}^i | \mathbf{s}_t^i, \mathbf{a}_t^i)$ or MSE for deterministic models. The learned model can subsequently be used for policy search under the MDP $\widehat{\mathcal{M}} = \langle \mathcal{S}, \mathcal{A}, \hat{\mathrm{P}}, \mathrm{r}, \mu_0, \gamma \rangle$. This MDP shares the state and action spaces $\mathcal{S}, \mathcal{A}$, reward function r, with the true environment $\mathcal{M}$, but learns the transition probability $\hat{\mathrm{P}}$ from the dataset $\mathcal{D}$.

Large Language Models (LLMs). Within the field of NLP, Large Language Models (LLMs) have emerged as a powerful tool for understanding and generating human-like text. An LLM is typically defined as a neural network model, often based on the transformer architecture [Vas+17], that is trained on a vast corpus of sequences, $U = \{U_1, U_2, \ldots, U_i, \ldots, U_N\}$, where each sequence $U_i = (u_1, u_2, \ldots, u_j, \ldots, u_{n_i})$ consists of tokens u_j from a vocabulary $\mathcal{V}$. Decoder-only LLMs [Rad+19; DL24] typically encode an autoregressive distribution, where the probability of each token is conditioned only on the previous tokens in the sequence, expressed as $p_\theta(U_i) = \prod_{j=1}^{n_i} p_\theta(u_j | u_{0:j-1})$. The parameters θ are learned by maximizing the probability of the entire dataset, $p_\theta(U) = \prod_{i=1}^{N} p_\theta(U_i)$. Every LLM has an associated tokenizer, which breaks an input string into a sequence of tokens, each belonging to $\mathcal{V}$.

Algorithm 15 ICL$_\theta$ [Liu+24c; Gru+23b]

Input: Time series $(x_i)_{i \le t}$, LLM p_θ, sub-vocabulary $\mathcal{V}_{\text{num}}$, precision k
1. Tokenize time series $\hat{x}_t = \text{“}x_1^1 x_1^2 \ldots x_1^k, \ldots\text{”}$
2. logits $\leftarrow p_\theta(\hat{x}_t)$ ▷ Forward pass
3. $\{P(X_{i+1} | x_i, \ldots, x_0)\}_{i \le t} \leftarrow \text{softmax}(\text{logits}(\mathcal{V}_{\text{num}}))$ ▷ using Masked Causal Attention
Return: $\{P(X_{i+1} | x_i, \ldots, x_0)\}_{i \le t}$

In-Context Learning (ICL). In order to use trajectories as inputs in ICL, we use the tokenization of time series proposed in [Gru+23b] and [Jin+24a]. This approach uses a subset of the LLM vocabulary $\mathcal{V}_{\text{num}}$ representing digits to tokenize the time series (Algorithm 15). Specifically, given a univarite time series, we rescale it into a specific

range [Liu+24c; ZBR; Req+24], encode it with k digits precision, and concatenate each value to build the LLM prompt:

$$\underbrace{[0.2513, 5.2387, 9.7889]}_{\text{time series}} \rightarrow \underbrace{[1.5, 5.16, 8.5]}_{\text{rescaled}} \rightarrow \underbrace{\text{“}150, 516, 850\text{”}}_{\text{prompt}}$$

After the LLM's forward pass, the logits corresponding to tokens in $\mathcal{V}_{\text{num}}$ can be used to fit a *Softmax* categorical distribution over the next value as demonstrated in [Liu+24b], thereby enabling uncertainty estimation.

4.3 Zero-shot dynamics learning using LLMs

4.3.1 Motivation

Before considering the multivariate trajectories of agents collected in RL environments, we first want to verify whether a pre-trained LLM model is sensitive to the primitive univariate signals akin to those encountered in them. For this, we investigate the attention mechanism of the `LLaMA 3-8B` model [DL24] when we feed it with different signals, including the periodic *fthigh* dimension from the HalfCheetah system [Bro+16]. By averaging the attention matrices over the 32 heads for each of the 32 layers of the multi-head attention in Llama3, we observed distinct patterns that provide insight into the model's focus and behavior (Figure 4.2 shows selected attention layers for each signal). The attention matrices exhibit a diagonal pattern, indicative of strong self-correlation among timestamps, and a subtriangular structure due to the causal masked attention in decoder-only transformers.

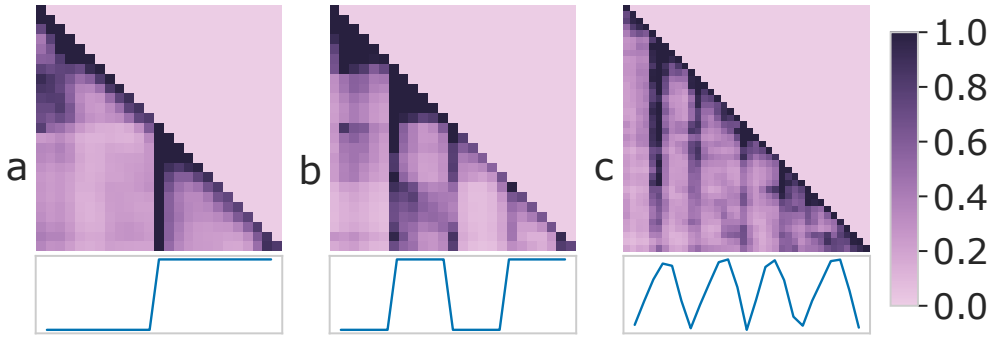


Figure 4.2: **LLMs can perceive time patterns.** The LLM (`LLaMA 3-8B`) is fed with 3 time series presenting distinct patterns. (**a**) Rectangular pulse. (**b**) Rectangular signal with constant sub-parts. (**c**) The *fthigh* dimension of HalfCheetah under an expert policy. Tokens belonging to constant slots (or peaks) attend to all the similar ones that precede them, focusing particularly on their respective first occurrence.

Further examination of the attention matrices reveals a more intricate finding. Tokens within repeating patterns (e.g., signal peaks, constant parts) not only attend to past tokens within the same cycle but also to those from previous occurrences of the same pattern, demonstrating a form of in-context learning. The ability to detect and exploit repeating patterns within such signals is especially valuable in RL, where state transitions and action outcomes often exhibit cyclical or recurring dynamics, particularly in continuous control tasks. However, applying this insight to RL presents two critical challenges related to 1) the integration of actions into the forecasting process, and 2) handling of the multivariate nature of RL problems. We now address these challenges by building on the insights from the analysis presented above.

4.3.2 Problem setup

Given an initial trajectory $\mathcal{T} = (\mathbf{s}_0, \mathbf{a}_0, \mathrm{r}_1, \mathbf{s}_1, \mathbf{a}_1, \mathrm{r}_2, \mathbf{s}_2, \ldots, \mathrm{r}_{T-1}, \mathbf{s}_{T-1})$ of length T, with $\mathbf{s}_t \in \mathcal{S}$, $\mathbf{a}_t = \pi(\mathbf{s}_t) \in \mathcal{A}$[1], where the policy π is fixed for the whole trajectory, and $\mathrm{r}_t \in \mathbb{R}$, we want to predict future transitions: given $(\mathbf{s}_{T-1}, \mathbf{a}_{T-1})$ predict the next state and reward $(\mathbf{s}_T, \mathrm{r}_T)$ and subsequent transitions autoregressively. For simplicity we first omit the actions and the reward, focusing instead on the multivariate sequence $\tau^\pi = (\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_T)$ where we assume that the state dimensions are independent. Later, we show how to relax the assumptions of omitting actions and rewards, as well as state independence, which is crucial for applications in RL. The joint probability density function of τ^π can be written as:

$$\begin{cases} \mathbf{P}(\tau^\pi) = \mu_0(\mathbf{s}_0) \prod_{t=1}^{T} \mathrm{P}^\pi(\mathbf{s}_t|\mathbf{s}_{t-1}) \\ \text{where } \mathrm{P}^\pi(\mathbf{s}_t|\mathbf{s}_{t-1}) = \int_{a\in\mathcal{A}} \pi(\mathbf{a}|\mathbf{s}_{t-1}) \mathrm{P}(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a})\, d\mathbf{a} \ . \end{cases} \tag{4.1}$$

Using the decoder-only nature of the in-context learner defined in Section 4.2, we can apply Algorithm 15 to each dimension of the state vector to infer the transition rule of each visited state in τ^π conditioned on its relative history: for all $j \in \{1, \ldots, d_s\}$,

$$\{\hat{\mathrm{P}}_\theta^{\pi,j}(s_t^j | s_{t-1}^j, \ldots, s_1^j, s_0^j)\}_{t\leq T} = \mathrm{ICL}_\theta(\tau^{\pi,j}) \tag{4.2}$$

where θ are the fixed parameters of the LLM used as an in-context learner, and T its context length. Assuming complete observability of the MDP state, the Markovian property unveils an equivalence between the learned transition rules and the corresponding Markovian ones: $\hat{\mathrm{P}}_\theta(\mathbf{s}_t|\mathbf{s}_{t-1}, \ldots, \mathbf{s}_1, \mathbf{s}_0) = \hat{\mathrm{P}}_\theta(\mathbf{s}_t|\mathbf{s}_{t-1})$.

This approach, that we name vICL (for vanilla ICL), thus applies Algorithm 15 on each dimension of the state individually, assuming their independence. Furthermore, the

[1] In practice, states and actions are real valued vectors spanning a space of dimensions respectively d_s and d_a: $\mathcal{S} = \mathbb{R}^{d_s}$, $\mathcal{A} = \mathbb{R}^{d_a}$

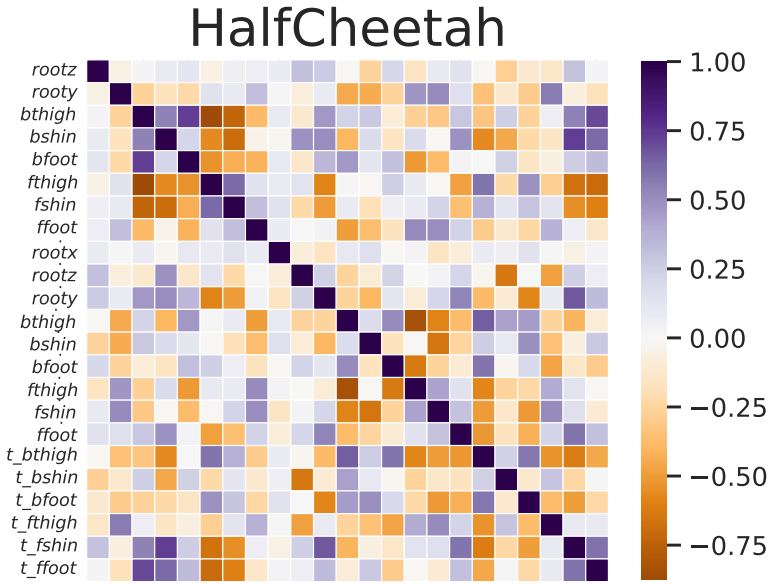


Figure 4.3: The covariance matrix from an expert dataset in the Halfcheetah environment indicates linear correlations between state-action features.

action information is integrated-out (as depicted in Equation (4.1)), which in theory, limits the application scope of this method to quantities that only depend on a policy through the expectation over actions (e.g., the value function $V^{\pi}(\mathbf{s})$). We address these limitations in the next section.

On the zero-shot nature of DICL. Our use of the term "zero-shot" aligns with the literature on LLMs and time series [Gru+23a], indicating that we do not perform any gradient updates or fine-tuning of the pretrained LLM's weights. Specifically, we adopt the dynamical systems formulation of ICL as studied in [Li+23], where the query consists of the trajectory "$s_0^j, s_1^j, \ldots, s_{t-1}^j$" and the label is the subsequent value s_t^j.

4.3.3 State and action dimension interdependence

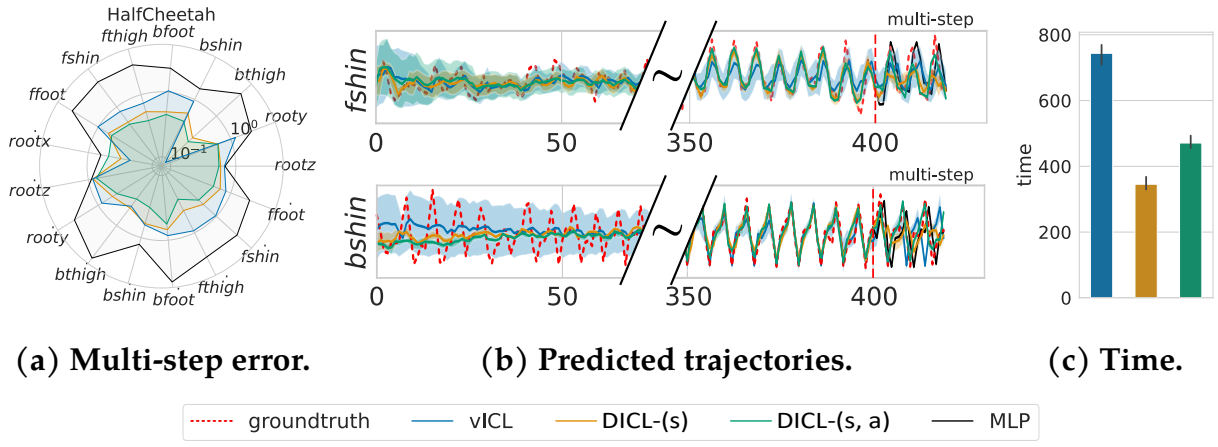
In this section we address the two limitations of vICL discussed in Section 4.3.2 by introducing Disentangled In-Context Learning (DICL), a method that relaxes the assumption of state feature independence and reintroduces the action by employing strategies that aim to map the state-action vector to a latent space where the features are independent. We can then apply vICL, which operates under the assumption of feature independence, to the latent representation. An added benefit of using such a latent space is that it can potentially reduce the dimensionality, leading to a speed-up of the overall approach.

While sophisticated approaches[2] like disentangled autoencoders could be considered for DICL, in this work we employ Principal Component Analysis (PCA). In fact, the absence of pre-trained models for this type of representation learning requires training from scratch on a potentially large dataset. This goes against our goal of leveraging the pre-trained knowledge of LLMs and ICL. Instead, we find that PCA, which generates new linearly uncorrelated features and can reduce dimensionality, strikes a good balance between simplicity, tractability, and performance (Figure 4.3 and Figure 4.4). Nonetheless, DICL is agnostic to this aspect and any transformation φ that can disentangle features can be used in place of PCA.

In the rest of the chapter we present two variants of DICL:

- DICL-$(\mathbf{s}, \mathbf{a})$, which applies the rotation matrix of PCA to the feature space of states and actions and then runs Algorithm 15 in the projection space of principal components;
- DICL-$(\mathbf{s})$, which applies the same transformation solely to the trajectory of states. This is useful in settings in which integrating the actions is not necessary, as when we only want to estimate the value function $V^{\pi}(\mathbf{s})$.

4.3.4 An illustrative example



(a) Multi-step error. **(b) Predicted trajectories.** **(c) Time.**

Figure 4.4: **PCA-based DICL achieves smaller multi-step error in less computational time.** We compare DICL-$(\mathbf{s})$ and DICL-$(\mathbf{s}, \mathbf{a})$ using a number of components equal to half the number of features, with the vanilla approach vICL and an MLP baseline. (*Llama 3-8B*).

In this section, we aim to challenge our approach against the HalfCheetah system from the MuJoCo Gym environment suite [Bro+16; TET12]. All our experiments are con-

[2] A more detailed discussion of alternative approaches to PCA is provided in Section C.3 and Chapter 5.

ducted using the Llama 3 series of models [DL24]. Figure 4.4 panel (**a**) shows the average MSE over a prediction horizon of $h \in \{1, \dots, 20\}$ steps for each state dimension. Figure 4.4 panel (**b**) shows predicted trajectories for selected state dimensions of the HalfCheetah system (the details of the experiment, the metrics and the remaining state dimensions are deferred to Section C.6).

We first observe that the LLM-based dynamics forecasters exhibit a burn-in phase (≈ 70 steps in Figure 4.4 panel (**b**)) that is necessary for the LLM to gather enough context. For multi-step prediction, Figure 4.4 panel (**a**), showing the average MSE over prediction horizons and trajectories, demonstrates that both versions of DICL improve over the vanilla approach and the MLP baseline trained on the context data, in almost all state dimensions. Indeed, we hypothesize that this improvement is especially brought by the projection in a linearly uncorrelated space that PCA enables. Furthermore, we also leveraged the dimensionality reduction feature by selecting a number of components c equal to half the number of the original features $d_s + d_a$ (or d_s in DICL-(**s**)). This results in a significant decrease in the computational time of the method without loss of performance, as showcased by Figure 4.4 panel (**c**).

LLMs comparison. In Table 4.1 we compare the performance obtained by the baselines and DICL when using different LLMs. Similarly to Figure 4.4 panel (**a**), the scores are calculated as the average over a given prediction horizon h across all dimensions (refer to Section C.6 for details on the MSE, and Section C.7 for details on the KS statistic). Note that similarly to Figure 4.4, we use PCA-based dimensionality reduction for both DICL-(**s**, **a**) and DICL-(**s**) in this experiment, *reducing the original number of features by half*. Overall, we can see that DICL, especially the DICL-(**s**, **a**) version, demonstrates improved calibration compared to both vICL and the MLP baselines, thanks to the disentangling effect of PCA. Moreover, DICL-(**s**) with the 3.1-70B model achieves the lowest Mean Squared Error (MSE) of 3.59. Nonetheless, DICL-(**s**, **a**) exhibits the highest MSE across all models. This is likely due to the additional error introduced by predicting action information, thereby modeling both the dynamics and the data-generating policy. This aspect differs from the MLP baseline, which is provided with real actions at test time (acting as an oracle), and from DICL-(**s**) and vICL, which operate solely on states. We show the detailed results of this ablation study in Section C.8. Notice that we exclusively used LLMs based on the LLaMA series of models [DL24]. This was a strategic choice due to the LLaMA tokenizer, which facilitates our framework by assigning a separate token to each number between 0 and 999. For other LLMs, algorithms have been suggested in the literature to extract transition rules from their output logits. For example, the Hierarchical Softmax algorithm [Liu+24c] could be employed for this purpose.

LLaMA	Metrics	
	MSE/$10^{-2}\downarrow$	**KS**/$10^{-2}\downarrow$
vICL		
3.2-1B	384 ± 31	52 ± 7
3.2-3B	399 ± 40	54 ± 8
3.1-8B	380 ± 32	53 ± 7
3-8B	375 ± 30	53 ± 7
3.1-70B	392 ± 35	55 ± 7
DICL-(s)		
3.2-1B	389 ± 38	50 ± 7
3.2-3B	404 ± 41	51 ± 7
3.1-8B	372 ± 44	50 ± 7
3-8B	370 ± 36	50 ± 7
3.1-70B	$\mathbf{359 \pm 33}$	54 ± 7
DICL-(s, a)		
3.2-1B	449 ± 37	46 ± 5
3.2-3B	450 ± 47	48 ± 6
3.1-8B	412 ± 39	$\mathbf{45 \pm 6}$
3-8B	418 ± 46	46 ± 5
3.1-70B	428 ± 47	47 ± 5
baseline		
MLP	406 ± 59	55 ± 3

Table 4.1: Comparison of different LLMs. Results are average over 5 episodes from each one of 7 D4RL [Fu+21a] tasks. $\downarrow$ means lower the better. The best average score is shown in **bold**. We show the average score $\pm$ the 95% Gaussian confidence interval.

4.4 Use-cases in Reinforcement Learning

As explored in the preceding sections, LLMs can be used as accurate dynamics learners for proprioceptive control through in-context learning. We now state our main contributions in terms of the integration of DICL into MBRL. First, we generalize the return bound of MBPO [Jan+19] to the more general case of multiple branches and use it to analyze our method. Next, we leverage the LLM to augment the replay buffer of an off-policy RL algorithm, leading to a more sample-efficient algorithm. In a second application, we apply our method to predict the reward signal, resulting in a hybrid model-based policy evaluation technique. Finally, we show that the LLM provides calibrated uncertainty estimates and conclude with a discussion of our results.

4.4.1 Theoretical analysis: Return bound under multi-branch rollouts

When using a dynamics model in MBRL, one ideally seeks monotonic improvement guarantees, ensuring that the optimal policy under the model is also optimal under the true dynamics, up to some bound. Such guarantees generally depend on system parameters (e.g., the discount factor γ), the prediction horizon k, and the model generalization error ε_{m}. As established in [Jan+19] and [Fra+24], the framework for deriving these theoretical guarantees is the one of branched model-based rollouts.

A branched rollout return $\eta^{\text{branch}}[\pi]$ of a policy π is defined in [Jan+19] as the return of a rollout which begins under the true dynamics P and at some point in time switches to rolling out under learned dynamics $\hat{\mathrm{P}}$ for k steps.

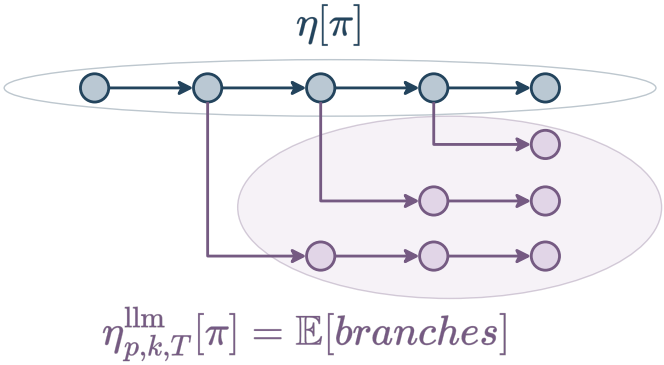


Figure 4.5: **Multi-branch return.** The rollout following the true dynamics P is shown in blue. The branched rollouts following LLM-based dynamics $\hat{\mathrm{P}}_{\text{llm}}$ are in purple. Branched rollouts can overlap, with the expectation over the overlapping branches as the return.

For our LLM-based dynamics learner, we are interested in studying a more general branching scheme that will be later used to analyze the results of our data-augmented off-policy algorithm. We begin by defining the multi-branch rollout return.

Definition 4.4.1 (Multi-branch rollout return)**.** *The multi-branch rollout return $\eta^{llm}_{p,k,T}[\pi]$ of a policy π is defined as the expected return over rollouts with the following dynamics:*

1. *for $t < T$, where T is the minimal context length, the rollout follows the true dynamics P.*
2. *for $t \geq T$, with probability p, the rollout switches to the LLM-based dynamics $\hat{\mathrm{P}}_{llm}$ for k steps, otherwise the rollout continues with the true dynamics P.*

See Section C.1 for a more formal definition.

These different rollout realizations, referred to as branches, can overlap, meaning that multiple LLM-based dynamics can run in parallel if multiple branchings from the true dynamics occur within the k-step window (see Figure 4.5). With this definition, we now state our main theoretical result, consisting of a return bound between the true return and the multi-branch rollout return.

Theorem 4.4.1 (Multi-branch return bound)**.** *Let T be the minimal length of the in-context trajectories, $p \in [0, 1]$ the probability that a given state is a branching point. We assume that the expected total variation between the LLM-based model and the true dynamics under a policy π is bounded at each timestep by*

$$\max_{t \geq T} \mathbb{E}_{\mathbf{s} \sim \mathrm{P}^t, \mathbf{a} \sim \pi} \Big[d_{\mathrm{TV}} \Big(\mathrm{P}(.|\mathbf{s}, \mathbf{a}) || \hat{\mathrm{P}}_{llm}(.|\mathbf{s}, \mathbf{a}) \Big) \Big] \leq \varepsilon_{llm}(T).$$

Then under a multi-branched rollout scheme with a branch length of k, the return is bounded as follows:

$$|\eta(\pi) - \eta^{llm}_{p,k,T}(\pi)| \leq 2 \frac{\gamma^T}{1-\gamma} r_{\max} k^2 \, p \, \varepsilon_{llm}(T) \ , \tag{4.3}$$

where $r_{\max} = \max_{s \in \mathcal{S}, a \in \mathcal{A}} r(s, a)$.

theorem 4.4.1 generalizes the single-branch return presented in [Jan+19], incorporating an additional factor of the prediction horizon k due to the presence of multiple branches, and directly accounting for the impact of the amount of LLM training data through the branching factor p. Additionally, the bound is inversely proportional to the minimal context length T, both through the power in the discount factor γ^T and the error term $\varepsilon_{\mathrm{llm}}(T)$. Indeed, the term $\varepsilon_{\mathrm{llm}}(T)$ corresponds to the generalization error of in-context learning. Several works in the literature studied it and showed that it typically decreases in $\mathcal{O}(T^{-1/2})$ with T the length of the context trajectories [Zek+24; Zha+23e; Li+23].

4.4.2 Data-augmented off-policy Reinforcement Learning

In this section, we show how DICL can be used for data augmentation in off-policy model-free RL algorithms such as Soft Actor-Critic (SAC) [Haa+18b]. The idea is to augment the replay buffer of the off-policy algorithm with transitions generated by DICL, using trajectories already collected by previous policies. The goal is to improve sample-efficiency and accelerate the learning curve, particularly in the early stages of learning as the LLM can generate accurate transitions from a small trajectory. We name this application of our approach DICL-SAC.

As defined in [CH23], data-augmented off-policy RL involves perturbing previously observed transitions to generate new transitions, without further interaction with the environment. The generated transitions should ideally be diverse and feasible under the MDP dynamics to enhance sample efficiency while ensuring that the optimal policy remains learnable.

Algorithm 16 DICL-SAC

Inputs: LLM-based dynamics learner (e.g. DICL-(s)), batch size b, LLM data proportion α, minimal context length T, and maximal context length T_{max}
Initialize policy π_ϕ, critic Q_ψ, replay buffer $\mathcal{R}$, and LLM replay buffer $\mathcal{R}_{\text{llm}}$, and context size T_{max}
for $t = 1, \dots, N\_interactions$ **do**
    New transition $(\mathbf{s}_t, \mathbf{a}_t, \mathrm{r}_t, \mathbf{s}_{t+1})$ from π_θ
    Add $(\mathbf{s}_t, \mathbf{a}_t, \mathrm{r}_t, \mathbf{s}_{t+1})$ to $\mathcal{R}$
    Store auxiliary action $\tilde{\mathbf{a}}_t \sim \pi_\theta(.|\mathbf{s}_t)$
    if Generate LLM data **then**
        Sample trajectory $\mathcal{T} = (\mathbf{s}_0, \dots, \mathbf{s}_{T_{\text{max}}})$ from $\mathcal{R}$
        $\{\hat{\mathbf{s}}_{i+1}\}_{0 \leq i \leq T_{\text{max}}} \sim$ DICL-(s) $(\mathcal{T})$
        Add $\{(\mathbf{s}_i, \tilde{\mathbf{a}}_i, r_i, \hat{\mathbf{s}}_{i+1})\}_{T \leq i \leq T_{\text{max}}}$ to $\mathcal{R}_{\text{llm}}$
    end if
    if update SAC **then**
        Sample batch $\mathcal{B}$ of size b from $\mathcal{R}$
        Sample batch $\mathcal{B}_{\text{llm}}$ of size $\alpha \cdot b$ from $\mathcal{R}_{\text{llm}}$
        Update ϕ and ψ on $\mathcal{B} \cup \mathcal{B}_{\text{llm}}$
    end if
end for

Algorithm 16 (DICL-SAC) integrates multiple components to demonstrate a novel proof-of-concept for improving the sample efficiency of SAC using DICL for data augmentation. Let $\mathcal{T} = (\mathbf{s}_0, \mathbf{a}_0, \mathrm{r}_0, \dots, \mathbf{s}_{T_{\text{max}}}, \mathbf{a}_{T_{\text{max}}}, \mathrm{r}_{T_{\text{max}}})$ be a real trajectory collected with a fixed policy π_ϕ, sampled from the real transitions being stored in a replay buffer $\mathcal{R}$. We gener-

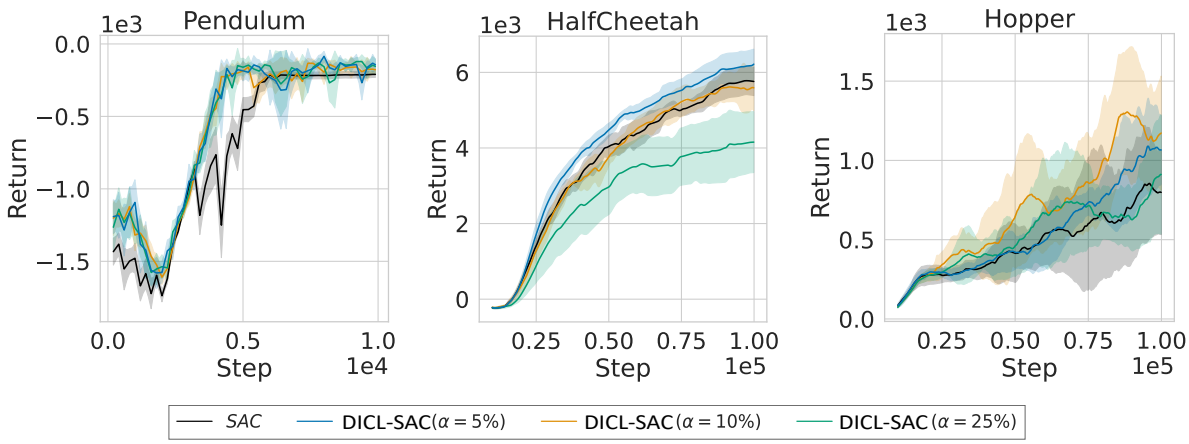


Figure 4.6: **Data-augmented off-policy RL.** In the early stages of training DICL-SAC improves the sample efficiency of SAC on three Gym control environments. Due to the intensive use of the LLM within DICL-SAC, we conducted this experiment using the *Llama 3.2-1B* model.

ate synthetic transitions $(\mathbf{s}_t, \tilde{\mathbf{a}}_t, \mathrm{r}_t, \hat{\mathbf{s}}_{t+1})_{T \leq t \leq T_{\max}}$; where $\hat{\mathbf{s}}_{t+1}$ is the next state generated by the LLM model applied on the trajectory of the states only, $\tilde{\mathbf{a}}_t$ is an action sampled from the data collection policy $\pi_\phi(.|\mathbf{s}_t)$, and T is the minimal context length. These transitions are then stored in a separate replay buffer $\mathcal{R}_{\text{llm}}$. At a given update frequency, DICL-SAC performs G gradient updates using data sampled from $\mathcal{R}$ and $\alpha\% \cdot G$ gradient updates using data sampled from $\mathcal{R}_{\text{llm}}$. Other hyperparameters of our method include the LLM-based method (vICL, DICL-$(\mathbf{s})$ or DICL-$(\mathbf{s}, \mathbf{a})$), how often we generate new LLM data and the maximal context length $T_{\max}$ (see Section C.4 for the full list of hyperparameters).

Figure 4.6 compares the return curves obtained by DICL-SAC against SAC in three control environments from the Gym library [Bro+16]. As anticipated with our data augmentation approach, we observe that our algorithm improves the sample efficiency of SAC at the beginning of training. This improvement is moderate but significant in the Pendulum and HalfCheetah environments, while the return curves tend to be noisier in the Hopper environment. Furthermore, as the proportion of LLM data α increases, the performance of the algorithm decreases (particularly in HalfCheetah), as predicted by theorem 4.4.1. Indeed, a larger proportion of LLM data correlates with a higher probability of branching p, as more branching points will be sampled throughout the training. Regarding the other parameters of our bound in theorem 4.4.1, we set $T = 1$, meaning all LLM-generated transitions are added to $\mathcal{R}_{\text{llm}}$, and $k = 1$ to minimize LLM inference time.

4.4.3 Policy Evaluation

System engineers are often presented with several policies to test on their systems. On the one hand, off-policy evaluation (e.g., [USK22]) involves using historical data collected from a different policy to estimate the performance of a target policy without disrupting the system. However, this approach is prone to issues such as distributional shift and high variance. On the other hand, online evaluation provides a direct and unbiased comparison under real conditions. System engineers often prefer online evaluation for a set of pre-selected policies because it offers real-time feedback and ensures that deployment decisions are based on live data, closely reflecting the system's true performance in production. However, online evaluations can be time-consuming and may temporarily impact system performance. To address this, we propose a hybrid approach using LLM dynamics predictions obtained through ICL to reduce the time required for online evaluation: the initial phase of policy evaluation is conducted as a standard online test, while the remainder is completed offline using the dynamics predictions enabled by the LLM's ICL capabilities.

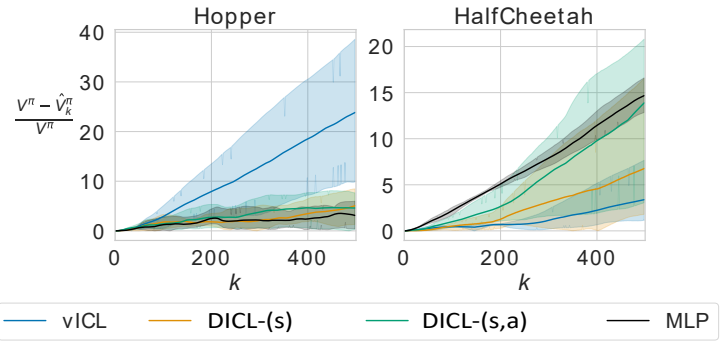


Figure 4.7: **Policy evaluation with DICL.** Relative error on the predicted value over $k = 500$ steps, with context length of $T = 500$. This experiment is conducted using the *Llama 3-8B* model.

Figure 4.7 illustrates the relative error in value obtained by predicting the trajectory of rewards for k steps, given a context length of $T = 500$. When $k \leq 500$, we complete the remaining steps of the 1000-step episode using the actual rewards. For the two versions of DICL, the reward vector is concatenated to the feature space prior to applying PCA. In the Hopper environment, it is evident that predicting the reward trajectory alone is a challenging task for the vanilla method vICL. On the contrary, both DICL-$(\mathbf{s})$ and DICL-$(\mathbf{s}, \mathbf{a})$ effectively capture some of the dependencies of the reward signal on the states and actions, providing a more robust method for policy evaluation, and matching the MLP baseline that has been trained on a dataset of transitions sampled from the same policy. However, in HalfCheetah we observe that the vanilla method largely improves upon both the baseline and DICL. We suspect that this is due to the fact that the reward

signal is strongly correlated with the $\dot{rootx}$ dimension in HalfCheetah, which proved to be harder to predict by our approach, as can be seen in Figure 4.4 panel (**a**).

Note that the experimental setup that we follow here is closely related to the concept of Model-based Value Expansion [Fei+18; Buc+18], where we use the dynamics model to improve the value estimates through an n-step expansion in an Actor Critic algorithm.

4.4.4 Calibration of the LLM uncertainty estimates

An intriguing property observed in Figure 4.4 panel (**b**) is the confidence interval around the predictions. As detailed in Algorithm 15, one can extract a full probability distribution for the next prediction given the context, enabling uncertainty estimation in the LLM's predictions. Notably, this uncertainty is pronounced at the beginning when context is limited, around peaks, and in regions where the average prediction exhibits large errors. We explore this phenomenon further in the next section by evaluating the calibration of the LLM's uncertainty estimates.

Calibration is known to be an important property of a dynamics model when used in RL [Mal+19]. In this section, we aim to investigate whether the uncertainty estimates derived from the LLM's logits are well-calibrated. We achieve this by evaluating the quantile calibration [KFE18b] of the probability distributions obtained for each LLM-based method.

Quantile calibration. For a regression problem with variable $y \in \mathcal{Y} = \mathbb{R}$, and a model that outputs a cumulative distribution function (CDF) F_i over y_i (where i indexes data points), quantile calibration implies that y_i (groundtruth) should fall within a $p\%$-confidence interval $p\%$ of the time:

$$\frac{\sum_{i=1}^{N} \mathbb{I}\{y_i \leq F_i^{-1}(p)\}}{N} \to p \quad \text{for all} \quad p \in [0,1] \quad \text{as} \quad N \to \infty \tag{4.4}$$

where $F_i^{-1} : [0,1] \to \mathcal{Y}$ denotes the quantile function $F_i^{-1}(p) = \inf\{y : p \leq F_i(y)\}$ for all $p \in [0,1]$, and N the number of samples.

LLMs are well-calibrated forecasters. Figure 4.8 shows the reliability diagram for the $bfoot$ dimension of the HalfCheetah system. The overall conclusion is that, regardless of the LLM-based sub-routine used to predict the next state, the uncertainty estimates derived from the LLM's logits are well-calibrated in terms of quantile calibration. Ideally, forecasters should align with the diagonal in Figure 4.8, which the LLM approach nearly achieves. Furthermore, when comparing with a naive baseline (the details are deferred to Section C.7), the LLM-forecaster matches the baseline when it's already

calibrated, and improves over it when it's not. To quantify a forecaster's calibration with a point statistic, we compute the Kolmogorov-Smirnov goodness-of-fit test Equation (C.6), shown in the legend of Figure 4.8.

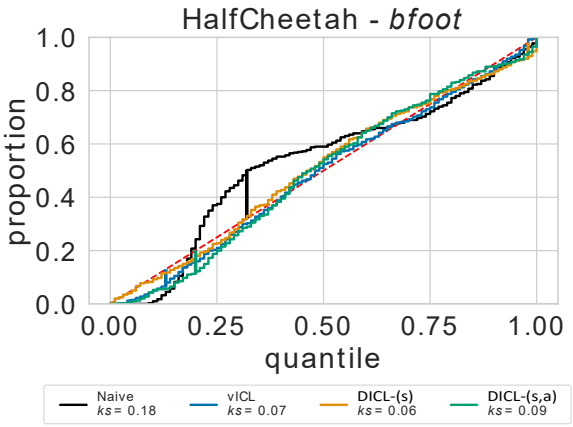


Figure 4.8: **Quantile calibration reliability diagram.** The LLM (*Llama 3 8B*) uncertainty estimates are well-calibrated. Vertical lines show the Kolmogorov-Smirnov statistic for each fit.

4.5 Discussion

By introducing the DICL framework, our goal is to bridge the gap between MBRL and LLMs. Our study raises multiple open questions and future research directions. Notably, the choice of the feature transformation is crucial for improving performance in specific applications. We plan to explore transformations that capture not only linear but also non-linear dependencies, such as AutoEncoders, as discussed in Section C.3. Another possible direction is the integration of textual context information into the LLM prompt. This approach has been shown to enhance the overall pipeline for time series forecasting [Jin+24a; XS23] and policy learning [Wan+23].

Besides this, our algorithm DICL-SAC performs data augmentation by applying the LLM to generate next states as in Equation (4.2). This operation requires a total of d_s calls to the LLM (or c after the φ transformation) to generate $T_{\max} - T$ transitions, as the time steps can be batched. This approach assumes a fixed policy in the context, allowing the LLM to implicitly learn P^{π_ϕ} using only the states. Looking ahead, a future research direction is to explore how to apply DICL to MBRL by replacing the dynamics model with an LLM. Naively applying DICL-$(\mathbf{s}, \mathbf{a})$ would require $(T_{\max} - T) \cdot d_s$ calls to the LLM, as transitions need to be predicted sequentially when actions change. This results in an extremely computationally expensive method, making it infeasible for many ap-

plications. Therefore, further research is needed to make this approach computationally efficient.

Conclusion

In this chapter, we ask how we can leverage the emerging capabilities of Large Language Models to benefit model-based RL. We build on previous work that successfully conceptualized in-context learning for univariate time series prediction, and provide a systematic methodology to apply ICL to an MDP's dynamics learning problem. Our methodology, based on a projection of the data in a linearly uncorrelated representation space, proved to be efficient in capturing the dynamics of typical proprioceptive control environments, in addition to being more computationally efficient through dimensionality reduction.

To derive practical applications of our findings, we tackled two RL use-cases: data-augmented off-policy RL, where our algorithm DICL-SAC improves the sample efficiency of SAC, and benefits from a theoretical guarantee under the framework of model-based multi-branch rollouts. Our second application, consisted in predicting the trajectory of rewards in order to perform hybrid online and model-based policy evaluation. Finally, we showed that the LLM-based dynamics model also provides well-calibrated uncertainty estimates.

Chapter 5

AdaPTS: Adapting Univariate Foundation Models to Probabilistic Multivariate Time Series Forecasting

Contents

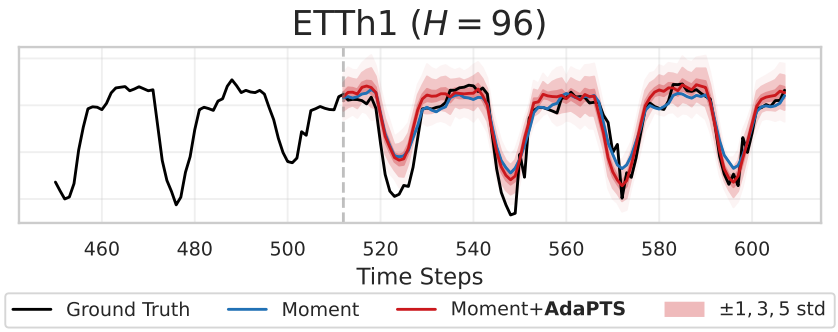
Pretrained foundation models (FMs) have shown exceptional performance in univariate time series forecasting tasks. However, several practical challenges persist, including managing intricate dependencies among features and quantifying uncertainty in predictions. This study aims to tackle these critical limitations by introducing **adapters**, which are feature-space transformations that facilitate the effective use of pre-trained univariate time series FMs for multivariate tasks. Adapters operate by projecting multivariate inputs into a suitable latent space and applying the FM independently to each dimension. Inspired by the literature on representation learning and partially stochastic Bayesian neural networks, we present a range of adapters and optimization/inference strategies. Experiments conducted on both synthetic and real-world datasets confirm the efficacy of adapters, demonstrating substantial enhancements in forecasting accuracy and uncertainty quantification compared to baseline methods. Our framework, `AdaPTS`, positions adapters as a modular, scalable, and effective solution for leveraging time series FMs in multivariate contexts, thereby promoting their wider adoption in real-world applications.

5.1 Introduction

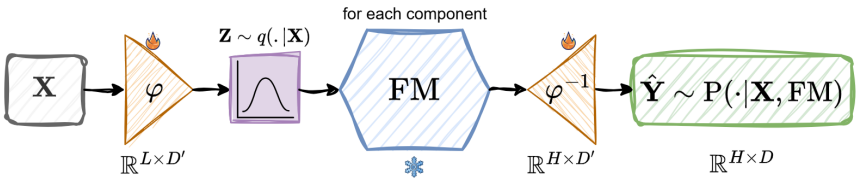
Time series forecasting is a well-established machine learning task that involves analyzing sequential data to predict future trends based on historical patterns. Two key challenges frequently arise in this context: (a) time series are often multivariate, incorporating multiple descriptive features [Wei19], and (b) estimating the uncertainty of a forecast is equally important, requiring probabilistic model outputs [GK14]. These challenges are particularly relevant in real-world applications where risk assessment depends on reliable forecasts, such as healthcare [JS12], finance [GPR13], energy management [ZWW14; NW18], and weather prediction [Pal12; Bi+23].

Existing FMs for time series forecasting, such as `Chronos` [Ans+24], are typically trained for univariate forecasting tasks due to tractability constraints, as a wide range of real-world time series problems often involve different numbers of features. Nevertheless, handling multivariate time series directly within these models has been attempted [Liu+24d], yet remains computationally challenging due to the high-dimensional dependencies among features. This limitation raises a fundamental question: how can we leverage existing pre-trained univariate FMs to enable probabilistic forecasting for multivariate time series?

To address this, we introduce `AdaPTS`, a novel framework designed to augment FMs with probabilistic adapters. As illustrated in Figure 5.1, `AdaPTS` applies a stochastic feature transformation that maps the input time series into a latent space, where predictions are made using the frozen FM. Our framework sets itself apart from existing literature



(a)



(b)

Figure 5.1: (a) Augmenting `Moment` time series foundation model with the **`AdaPTS`** framework provides *probabilistic* and *more accurate* predictions. (b) **The `AdaPTS` framework:** The input time series is transformed through a feature space transformation φ that maps into a stochastic latent space, where the FM now operates. The fire symbol indicate trainable weights while the snowflake implicates that the parameters of the FM are kept frozen.

by fitting an inverse transformation as part of the adapter, allowing predictions to be transformed back into the original feature space. Beyond enhancing forecasting accuracy, the integration of stochasticity into the adapter's latent representation ensures that the model captures uncertainty, thereby improving both calibration and robustness.

Our approach leads to several novel insights and contributions, which we summarize as follows:

1. **Multivariate FM adaptation.** We introduce a principled methodology for adapting existing pre-trained univariate FMs to multivariate probabilistic forecasting, resulting in the **`AdaPTS`** framework.
2. **Theoretical foundations of adapters.** We provide a theoretical analysis to support the necessity of adapters, starting with the analytically tractable case of linear adapters and linear FMs. We then build on the literature on partially stochastic Bayesian neural networks to introduce probabilistic adapters.
3. **Empirical validation.** We conduct experiments on multivariate time series fore-

casting benchmarks, demonstrating that our approach improves forecasting accuracy over baseline methods. We then analyze the interpretability of the learned latent representation and show that adapters enable cost-efficient adaptation by reducing the dimensionality of the feature space.

The rest of this chapter is organized as follows: Section 5.2 states the problem setup and related work, Section 5.3 details the theoretical analysis on linear adapters. Section 5.4 extends our framework to probabilistic adapters, while Section 5.5 showcases experimental results. Finally, we conclude with future directions in Section 5.6.

5.2 Background knowledge

5.2.1 Problem setup

Consider a multivariate long-term time series forecasting task, represented by: a data matrix $\mathbf{X} \in \mathbb{R}^{L \times D}$ where L is the context window size and D is the multivariate time series dimensionality, and a target matrix $\mathbf{Y} \in \mathbb{R}^{H \times D}$, where H is the forecasting horizon. We denote by $\mathbf{x}_d \in \mathbb{R}^{L \times 1}$ (respectively $\mathbf{y}_d \in \mathbb{R}^{H \times 1}$) the d-th component of the input (respectively target) multivariate time series.

Our goal is to leverage a frozen pre-trained univariate time series foundation model denoted by $f_{\mathrm{FM}} : \mathbb{R}^{L \times 1} \to \mathbb{R}^{H \times 1}$ (Section 5.1). In the case of a multivariate input $\mathbf{X}$, for the sake of simplicity, we denote by $f_{\mathrm{FM}}(\mathbf{X})$ the application of f_{FM} to each channel independently. We aim to achieve best forecasting quality by minimizing the mean squared error (MSE) loss defined as follows:

$$\mathcal{L} = \|\mathbf{Y} - f_{\mathrm{FM}}(\mathbf{X})\|_{\mathrm{F}}^2 = \frac{1}{D} \sum_{d=1}^{D} \|\mathbf{y}_d - f_{\mathrm{FM}}(\mathbf{x}_d)\|_2^2. \tag{5.1}$$

We now formally define an adapter, a tool by means of which we aim to best use the foundation model f_{FM} for multivariate forecasting:

Definition 5.2.1 (adapter)**.** *An adapter is a feature-space transformation $\varphi : \mathbb{R}^D \to \mathbb{R}^{D'}$ that is applied to the data prior to the foundation model*[a]*. The forecast is then obtained by transforming the predictions back to the original feature space:*

$$\hat{\mathbf{Y}}(\mathbf{X}; \varphi) = \varphi^{-1}\Big(f_{\mathrm{FM}}(\varphi(\mathbf{X}))\Big).$$

[a] In practice, φ is applied on matrices $\mathbf{X}$ in $\mathbb{R}^{L \times D}$. This denotes the application of φ on each row of $\mathbf{X}$.

According to this definition, an adapter is valid only if the inverse transformation $\varphi^{-1} : \mathbb{R}^{D'} \to \mathbb{R}^{D}$, such that $\forall \mathbf{x} \in \mathbb{R}^{D}$, $\varphi^{-1} \circ \varphi(\mathbf{x}) = \mathbf{x}$, is well-defined on $\mathbb{R}^{D'}$. In the rest of the chapter, we relax this condition by implementing the direct transformation as *encoder* ($\varphi \triangleq \text{enc}$), and respectively, the inverse transformation as *decoder* ($\varphi^{-1} \triangleq \text{dec}$). In this case, the predictions obtained after the application of the adapter become:

$$\hat{\mathbf{Y}}(\mathbf{X}; \text{enc}, \text{dec}) = \text{dec}\Big(f_{\text{FM}}(\text{enc}(\mathbf{X}))\Big).$$

We note that in the literature, there exist alternatives to adapt to the multivariate setting [ZY23; Tia+23], but we have chosen this family of adapters due to their high flexibility as: (a) any foundation model can be plugged-in, (b) no requirement of fine-tuning due to feature-level transformations [Feo+24], (c) adaptation to the computation budget by defining the number of encoded channels.

Optimality of an adapter. In order for an adapter to be useful, it has to achieve a forecasting error lower compared to an adapter that represents the identity matrix $\mathbf{I}$ of size D and does not perform any transformation of the feature space. Therefore, we define the optimality of the adapter based on improving the forecasting error of the identity baseline:

$$\mathcal{L} \geq \mathcal{L}(\varphi) = \|\mathbf{Y} - \varphi^{-1}\Big(f_{\text{FM}}(\varphi(\mathbf{X}))\Big)\|_{\text{F}}^{2},$$

where $\mathcal{L}$ is defined by Equation (5.1).

5.2.2 Related work

Time Series Foundational Models. Over the past two years, a plethora of foundation models have been proposed with a particular focus on time series forecasting. Some of these models like `GPT4TS` [Tia+23] and `Time-LLM` [Jin+24b] "reprogram" a Large Language Model to the forecasting setting by freezing most of its layers and fine-tuning additional time series-specific modules to a new downstream task. The majority of time series FMs including `Lag-Llama` [Ras+24], `Chronos` [Ans+24], `Moirai` [Liu+24d], `TimesFM` [Das+24] and `Moment` [Gos+24] are trained from scratch on a large volume of time series data. Beyond the transformer architecture, [Eka+24] present `TTM`, an all MLP foundation model.

Adapters. The multivariate setting presents a significant challenge for time series FMs, as different tasks involve varying numbers of channels[1]. To the best of our knowledge, the only model that naturally accommodates any number of channels is `Moirai` [Liu+24d], which, however, suffers from high computational demand due to processing all channels flattened in the transformer, leading to a quadratic memory complexity w.r.t. to

[1]Throughout the chapter, the words *features*, *channels*, and *components* are used interchangeably to refer to the number of variates in a multivariate time series, represented as D in Section 5.2.1.

the number of channels. Most foundation models, instead, treat each one of these independently, which, as noted by [Feo+24; Feo+25], remains computationally expensive when full fine-tuning is required. For classification tasks with hundreds or thousands of features, they demonstrated that simple adapters like the rotation matrix obtained through Principal Components Analysis (`PCA`) mitigate this issue. At the same time, [Ben+25] showed that `PCA` preserves channel interactions by learning new disentangled components. However, in both cases, `PCA` provided little improvement over independent processing, leaving room for further enhancements. In the context of tabular regression, foundation models such as `TabDPT` [Ma+24a] also use `PCA` to adapt to a variable number of features.
Less related to our work, [LM16] use a Gaussian process adapter in the context of irregular time series classification. In other domains, adapters have been used for multimodal (text-time series) representation learning [Zha+24c] and computer vision [Li+25; Yin+23; Pan+22].

5.3 Theoretical analysis

The purpose of this section is to study the optimization problem that the adapter φ is aiming to solve:

$$\varphi^* = \arg\min_{\varphi} \|\mathbf{Y} - \varphi^{-1}\Big(f_{\mathrm{FM}}(\varphi(\mathbf{X}))\Big)\|_{\mathrm{F}}^2. \tag{5.2}$$

Under mild assumptions on the class of adapter functions and the backbone foundation model f_{FM}, we aim to characterize the optimal solution φ^* and prove that it fulfills the optimality condition: $\mathcal{L}(\varphi^*) \leq \mathcal{L}$.

We first consider the linear case where we constrain the adapter φ to the class of linear transformations, parametrized by a matrix $\mathbf{W}_\varphi \in \mathbb{R}^{D\times D}$: $\varphi(\mathbf{X}) = \mathbf{X}\mathbf{W}_\varphi$.

Assumption 5.3.1. *$\mathbf{W}_\varphi$ has full rank: rank$(\mathbf{W}_\varphi) = D$, ensuring its invertibility.*

Assumption 5.3.2. *For ease of derivation, we consider a similar linear parameterization for the foundation model: $f_{\mathrm{FM}}(\mathbf{X}) = \mathbf{W}_{FM}^\top\mathbf{X} + \mathbf{b}_{FM}\mathbf{1}^\top$ where $\mathbf{W}_{FM} \in \mathbb{R}^{L\times H}$, $\mathbf{b}_{FM} \in \mathbb{R}^H$, and $\mathbf{1}$ a vector of ones of dimension D.*

Proposition 5.3.1 (Optimal linear adapter)**.** *Under assumption 5.3.1 and assumption 5.3.2, the closed-form solution of the problem*

$$\mathcal{L}(\mathbf{W}_\varphi) = \|\mathbf{Y} - \Big(\mathbf{W}_{FM}^\top\mathbf{X}\mathbf{W}_\varphi + \mathbf{b}_{FM}\mathbf{1}^\top\Big)\mathbf{W}_\varphi^{-1}\|_F^2 \tag{5.3}$$

writes as:

$$\mathbf{W}_{\varphi}^{*} = (\mathbf{B}^{\top}\mathbf{A})^{+}\mathbf{B}^{\top}\mathbf{B}, \tag{5.4}$$

where $\mathbf{W}_{\varphi}^{*} = \arg\min_{\mathbf{W}_{\varphi}} \mathcal{L}(\mathbf{W}_{\varphi})$, $\mathbf{A} = \mathbf{Y} - \mathbf{W}_{FM}^{\top}\mathbf{X}$, $\mathbf{B} = \mathbf{b}_{FM}\mathbf{1}^{\top}$, *and* $(\mathbf{B}^{\top}\mathbf{A})^{+}$ *denoting the pseudo-inverse operator.*

Proof. The result follows by differentiating $\mathcal{L}(\mathbf{W}_{\varphi})$ with respect to $\mathbf{W}_{\varphi}$, and solving the Euler equation: $\nabla_{\mathbf{W}_{\varphi}}\mathcal{L} = 0$. The detailed proof is deferred to Section D.1. □

Remark 5.3.1. *In this case, the fact that the matrix* $\mathbf{B} = \mathbf{b}_{FM}\mathbf{1}^{\top}$ *have identical columns renders the matrix* $\mathbf{B}^{\top}\mathbf{A}$ *degenerate (with* $rank(\mathbf{B}^{\top}\mathbf{A}) = 1$*). In practice, we add a positive constant to the diagonal in order to numerically stabilize the matrix inversion:* $\mathbf{W}_{\varphi}^{*} = (\mathbf{B}^{\top}\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{B}^{\top}\mathbf{B}$*, with* $\lambda > 0$*. In Section 5.3.1, we show that we are able to reach an optimal solution regardless of this added regularization.*

proposition 5.3.1 establishes the existence of a solution to the optimal adapter problem defined in Equation (5.2) and demonstrates that this solution does not necessarily correspond to the identity matrix. This analysis in the linear case provides the foundational motivation for our gradient descent based optimization approach to determine the optimal adapter configuration.

5.3.1 Illustrative example

Synthetic data. Our synthetic dataset comprises two multivariate time series, one with several independent components and the other with linearly correlated ones (Figure 5.2), designed to evaluate a linear feature-space transformation. The data generation process creates five *uncorrelated* base signals (sinusoids with distinct frequencies, amplitudes, and iid noise) and derives eight additional channels through linear combinations of these bases with additive Gaussian noise of different magnitude $\sigma \in (0.1, 0.2, 0.5)$.

This construction provides a controlled environment where the ground truth relationship between channels is known: the underlying data manifold is effectively five dimensional, but in the correlated case the observed eight-dimensional multivariate time series includes varying levels of noise and linear mixing.

Randomly generated linear FMs. The experimental setup in Figure 5.2 consists in randomly sampling the linear parameters of a toy foundation model: $\mathbf{W}_{\text{FM}}$ and b_{FM}. To simulate a realistic scenario, we use Glorot-uniform initialization distribution as it would be the case in neural network-based architectures. We then compute the closed-form solution $\mathbf{W}_{\varphi}^{*}$ Equation (5.4) on raw data $\mathbf{X}$, and compare the resulting loss value with the baseline (using the identity matrix $\mathbf{I}$ as adapter) and the `PCA`-only adapter.

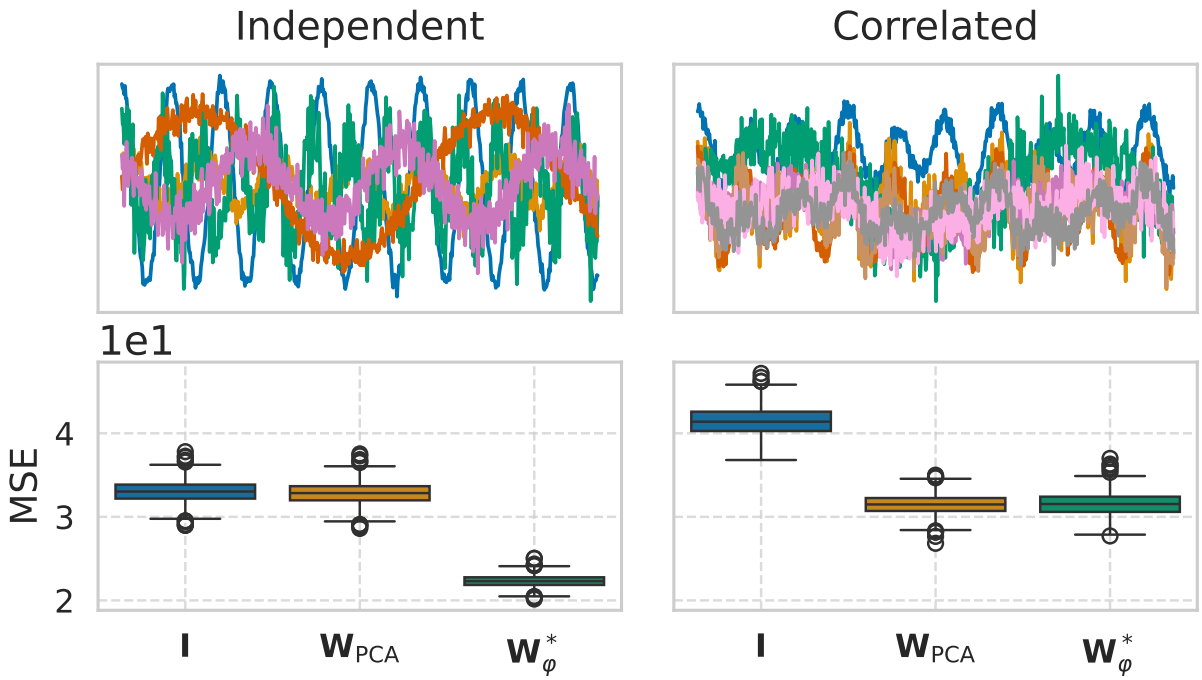


Figure 5.2: **Optimality of $\mathbf{W}_{\varphi}^{*}$.** Comparing the MSE obtained with $\mathbf{W}_{\varphi}^{*}$ against the baseline, for 1000 randomly generated linear FM.

Figure 5.2 shows that in the case of uncorrelated data (*left* column) `PCA` is equivalent to the identity matrix, while the solution $\mathbf{W}_{\varphi}^{*}$ to the problem $\mathcal{L}(\mathbf{W}_{\varphi})$ reaches an order of magnitude better forecasting loss. In the correlated case, we observe that `PCA` has a similar performance to the optimal solution. This example motivates the adapter idea through the existence of better linear transformations than the identity matrix in the case of linear foundation models.

5.4 `AdaPTS`: Adapters for Probabilistic Multivariate Time Series Forecasting

In this section, we introduce families of adapters that verify the conditions stated in definition 5.2.1. Furthermore, we extend this definition to include probabilistic variants of adapters, useful for uncertainty quantification.

5.4.1 Families of adapters

Our framework accommodates various families of transformations that can serve as adapters. Initially, we define linear AutoEncoders and subsequently extend them to their deep non-linear counterparts. Ultimately, we introduce the probabilistic adapter framework, encompassing Variational AutoEncoders (`VAE`) and `Dropout`-based families of adapters.

Linear AutoEncoders. In addition to the linear setup introduced in Section 5.3, we extend Linear AutoEncoders to provide a simple yet effective method for dimensionality reduction while preserving the temporal relationships within time series data. In this more general case, the encoder compresses the multivariate time series $\mathbf{X}$ into a potentially lower-dimensional representation $\mathbf{Z} = \mathbf{X}\mathbf{W}_{\text{enc}}$, where $\mathbf{W}_{\text{enc}} \in \mathbb{R}^{D \times D'}$ is the linear transformation matrix, and $D' \leq D$. The decoder reconstructs the forecast to the original feature space as $\hat{\mathbf{Y}} = f_{\text{FM}}(\mathbf{Z})\mathbf{W}_{\text{dec}}$. Finally, the parameters of the encoder $\mathbf{W}_{\text{enc}}$ and the decoder $\mathbf{W}_{\text{dec}}$ are jointly optimized to minimize the objective in Equation (5.2).

Deep non-linear AutoEncoders. Deep non-linear AutoEncoders extend their linear counterparts by employing multiple layers of non-linear transformations. The encoder maps the input $\mathbf{X}$ to a latent space $\mathbf{Z} = \text{enc}(\mathbf{X}; \theta_{\text{enc}})$, where enc is parameterized by a deep neural network. Similarly, the decoder reconstructs the predictions of the foundation model in the latent space: $\hat{\mathbf{Y}} = \text{dec}(f_{\text{FM}}(\mathbf{Z}); \theta_{\text{dec}})$.

Besides AutoEncoders, Normalizing Flows [KPB21] such as `RealNVP` [DSB17] are a valid choice in the context of adapters, thanks to their inherently invertible nature. However, their training proved challenging due to the transformation being tied to its inverse leading to poor performance in practice. We defer a discussion on Normalizing Flows as adapters to Section D.2.

5.4.2 Probabilistic Adapters

We now discuss an alternative to the optimization of adapters, which is based on a Bayesian treatment of their parameters. There are many options on how to carry out inference over these parameters, and we can draw from the literature on Bayesian inference for neural networks [Pap+24].

Considering a FM which yields point predictions, the appeal of Bayesian adapters is that they enable probabilistic predictions, which can be used for uncertainty quantification. Note that this is the case for models such as `Chronos` and `Moirai`, which output a distribution over the time series continuous values[2]. For deterministic FMs such as

[2]In the case of `Chronos`, this distribution is obtained through a categorical distribution (with *softmax* probabilities) over a tokenized space of the time series values.

`Moment` [Gos+24], a Bayesian treatment of adapters yields an ensemble of such predictions, which is key for accounting for the predictive uncertainty.

Inference. Recalling that θ represents the set of parameters of encoder (enc_θ) and decoder (dec_θ), we can attempt to obtain the posterior distribution over these parameters through Bayes theorem [Gel+13]:

$$p(\theta|\mathbf{Y}, \mathbf{X}) \propto p(\mathbf{Y}|\mathbf{X}, \theta)p(\theta),$$

where $p(\theta)$ is the prior distribution over the parameters and $p(\mathbf{Y}|\mathbf{X}, \theta)$ the likelihood, with $\mathbf{Y}, \mathbf{X}$ representing a training dataset in this context. Alternatively, we can treat the latent representation $\mathbf{Z}$ as stochastic, where the interest is now to characterize the following posterior:

$$p(\mathbf{Z}|\mathbf{Y}, \mathbf{X}) \propto p(\mathbf{Y}|\mathbf{X}, \mathbf{Z})p(\mathbf{Z}).$$

In these two formulations, the posterior distribution over the parameters is instrumental in obtaining predictive distributions useful for uncertainty quantification. For instance, in the case of inference over θ, for new test data $\mathbf{Y}^*, \mathbf{X}^*$ we obtain:

$$p(\mathbf{Y}^*|\mathbf{X}^*, \mathbf{Y}, \mathbf{X}) = \int p(\mathbf{Y}^*|\mathbf{X}^*, \theta)p(\theta|\mathbf{Y}, \mathbf{X})d\theta.$$

Characterizing the posterior analytically, however, is intractable and we need to resort to approximations. The literature on Bayesian inference offers various strategies, which can be adapted to neural networks [Pap+24], including variational inference [Gra11], Laplace approximations [Yan+24], and Markov Chain Monte Carlo (MCMC) [CFG14; Tra+22].

Within the **`AdaPTS`** framework, we focus in particular on variational inference (VI) for `VAE` adapters and on Monte Carlo dropout [GG16a] as an approximate form of VI for carrying out inference over θ.

Variational AutoEncoders. Following the Bayesian perspective on adapters, `VAE` assume a prior distribution over the latent representation $\mathbf{Z}$, typically $\mathcal{N}(0, \mathbf{I})$. The encoder then outputs parameters of the posterior distribution $q_\phi(\mathbf{Z}|\mathbf{X})$, and the decoder generates reconstructions of predictions $\hat{\mathbf{Y}} \sim p_\theta(\mathbf{Y}|\mathbf{X}, f_{\text{FM}}(\mathbf{Z}))$ where θ parametrize a likelihood model p. In the context of adapters, we define the training objective of the `VAE`, which brings together the forecasting loss and a regularization term, similarly to the *evidence lower bound* (ELBO) [KW13] objective, in proposition 5.4.1. The derivation of this lower bound and a discussion on the implications of each term of the loss are deferred to Section D.1.2.

Proposition 5.4.1 (VAE adapter training objective)**.** *The training objective for the VAE adapter is the maximization of an* ELBO*-like lower bound on the marginal likelihood of the target* $\mathbf{Y}$*:*

$$\log p_\theta(\mathbf{Y}|\mathbf{X}, f_{\text{FM}}) \geq \mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{X})}\left[\log p_\theta(\mathbf{Y}|\mathbf{X}, f_{\text{FM}}(\mathbf{Z}))\right] - d_{\text{KL}}(q_\phi(\mathbf{Z}|\mathbf{X})||p(\mathbf{Z})),$$

where d_{KL} *denotes the Kullback-Leibler divergence.*

Remark 5.4.1. *In practice, we use the Gaussian likelihood as our likelihood model:*

$$p_\theta(\mathbf{Y}|\mathbf{X}, f_{\text{FM}}(\mathbf{Z})) = \mathcal{N}(\mathbf{Y}; \hat{\mathbf{Y}}, \sigma^2\mathbf{I}), \quad \hat{\mathbf{Y}} = \text{dec}_\theta(f_{\text{FM}}(\mathbf{Z})).$$

In this case, the forecasting loss term boils down to the MSE objective in Equation (5.2) (up to multiplicative and additive noise-related constants):

$$\log \mathcal{N}(\mathbf{Y}; \hat{\mathbf{Y}}, \sigma^2\mathbf{I}) = -\frac{1}{2\sigma^2}\|\mathbf{Y} - \hat{\mathbf{Y}}\|_{\text{F}}^2 - \frac{HD}{2}\log(2\pi\sigma^2).$$

Notice that one can also learn a model of the noise where

$$\text{dec}_\theta(f_{\text{FM}}(\mathbf{Z})) = [\mu_\theta(\mathbf{Y}|\mathbf{X}, f_{\text{FM}}(\mathbf{Z})), \sigma_\theta(\mathbf{Y}|\mathbf{X}, f_{\text{FM}}(\mathbf{Z}))].$$

Remark 5.4.2. *The* KL *divergence regularization term can be multiplied by a scaling factor* β *to control the disentanglement – independence of the latent representation components. This results in* β*-VAE* [Hig+17]*, that we refer to as the VAE adapter throughout the chapter.*

Dropout as approximate VI. Dropout [Sri+14a] can be interpreted as a form of variational inference, where a variational distribution is imposed over the weights of a neural network [GG16a]. Specifically, applying dropout during training corresponds to approximating a posterior over the weights using a Bernoulli distribution. This perspective allows the deterministic models to be transformed into probabilistic models by introducing stochasticity through dropout.

Adapters and partially stochastic Bayesian neural networks. Treating adapters in a Bayesian manner while keeping the FM fixed aligns with the concept of partially stochastic Bayesian neural networks, which provides theoretical guarantees on universal conditional density estimation [Sha+23]. This framework ensures that the model can approximate any conditional density, provided that stochasticity is introduced early enough in the architecture and that the number of stochastic units matches or exceeds the output dimension. Using probabilistic adapters, we comply with these conditions by making the encoder stochastic, allowing the learned latent space to capture uncertainty while leveraging the FM's fixed parameters.

Dataset	H	No adapter	with adapter				
		Moment	PCA	LinearAE	dropoutLAE	LinearVAE	VAE
ETTh1	96	$0.411_{\pm 0.012}$	$0.433_{\pm 0.001}$	$0.402_{\pm 0.002}$	$\mathbf{0.395_{\pm 0.003}}$	$0.400_{\pm 0.001}$	$0.404_{\pm 0.001}$
	192	$\mathbf{0.431_{\pm 0.001}}$	$0.440_{\pm 0.000}$	$0.452_{\pm 0.002}$	$0.446_{\pm 0.001}$	$0.448_{\pm 0.002}$	$\mathbf{0.431_{\pm 0.001}}$
Illness	24	$2.902_{\pm 0.023}$	$2.98_{\pm 0.001}$	$2.624_{\pm 0.035}$	$2.76_{\pm 0.061}$	$2.542_{\pm 0.036}$	$\mathbf{2.461_{\pm 0.008}}$
	60	$3.000_{\pm 0.004}$	$3.079_{\pm 0.000}$	$3.110_{\pm 0.127}$	$2.794_{\pm 0.015}$	$\mathbf{2.752_{\pm 0.040}}$	$2.960_{\pm 0.092}$
Weather	96	$0.177_{\pm 0.010}$	$0.176_{\pm 0.000}$	$0.169_{\pm 0.000}$	$\mathbf{0.156_{\pm 0.001}}$	$0.161_{\pm 0.001}$	$0.187_{\pm 0.001}$
	192	$0.202_{\pm 0.000}$	$0.208_{\pm 0.001}$	$\mathbf{0.198_{\pm 0.001}}$	$0.200_{\pm 0.001}$	$0.204_{\pm 0.000}$	$0.226_{\pm 0.000}$
ExchangeRate	96	$\mathbf{0.130_{\pm 0.011}}$	$0.147_{\pm 0.000}$	$0.167_{\pm 0.013}$	$\mathbf{0.130_{\pm 0.011}}$	$0.243_{\pm 0.039}$	$0.455_{\pm 0.010}$
	192	$\mathbf{0.210_{\pm 0.002}}$	$0.222_{\pm 0.000}$	$0.304_{\pm 0.005}$	$0.305_{\pm 0.013}$	$0.457_{\pm 0.020}$	$0.607_{\pm 0.021}$

Table 5.1: Performance comparison between the baseline `Moment` model without adapters against different adapter architectures (`PCA`, `LinearAE`, `dropoutLinearAE`, `LinearVAE`, and `VAE`), for multivariate long-term forecasting with different horizons H. We display the average test MSE $\pm$ standard error obtained on 3 runs with different seeds. **Best** results are in bold, with lower values indicating better performance.

5.5 Experiments & Results

In this section, we empirically demonstrate quantitative and qualitative superiority of **`AdaPTS`** for multivariate long-term time series forecasting on common benchmarks. We show that the proposed adapter approach yields an improvement to `Moment`, a commonly used time series forecasting foundation model, over different datasets and prediction horizons. We then extend the analysis to other foundation models such as `TTM` and `TimesFM`.

5.5.1 Time series forecasting

Datasets. Our experiments are conducted on four publicly available real-world multivariate time series datasets, commonly used for long-term forecasting [Ilb+24; Wu+21; Che+23; Nie+23; Zen+23]. These datasets include the Electricity Transformer Temperature dataset (ETTh1) [Zho+21], ExchangeRate [Lai+18], Weather [Max21], and Influenza-like Illness [US 24]. All time series are segmented with an input length of $L = 512$, prediction horizons $H \in [96, 192]$ and $H \in [24, 60]$ for the Illness dataset, and a stride of 1, meaning each subsequent window is shifted by one step. These datasets (detailed in Section D.3.1) originate from various application domains, enabling a comprehensive evaluation of our framework across diverse real-world scenarios.

Baseline. We compare our method against the vanilla application of $\texttt{Moment}_s$ (where s stands for *small*) from the `Moment` family of models [Gos+24]. This means that for each

dataset, we apply $\texttt{Moment}_s$ independently to each feature. Additionally, we compare our learning-based adapters against PCA, an adapter that has been used in the literature for model-based reinforcement learning [Ben+25] and time series classification [Feo+24; Feo+25].

AdaPTS improves the performance of Moment. We present the forecasting error measured by the Mean Squared Error (MSE) in Table 5.1 and the Mean Absolute Error (MAE) in Section D.5. On the ETTh1 dataset with a prediction horizon $H = 96$, all adapter-based variants outperform the baseline Moment model, with dropoutLinearAE achieving the best performance, showing an 8% improvement. Similar results are observed for the Illness dataset, where all adapters improve over the baseline. Notably, the VAE achieves a significant 15% improvement, reducing the MSE from 2.902 to 2.461 at $H = 24$. In the Weather dataset, the dropoutLinearAE adapter shows the best improvement across all adapter architectures for $H = 96$, while its deterministic counterpart, LinearAE, takes the lead at $H = 192$. The results on the ExchangeRate dataset are mixed, with some adapters matching the baseline performance (dropoutLinearAE at $H = 96$) while others show degraded performance, particularly at a longer prediction horizon ($H = 192$), which is also observed for the ETTh1 dataset. Overall, **AdaPTS** improves the forecasting accuracy of Moment in 5 out of the 8 considered tasks, matches its performance in 2, and degrades performance in 1 task.

Beyond Moment. To validate our approach at a broader scope, we tested **AdaPTS** against other TSFMs with distinct architectural characteristics: (1) TTM, an all-MLP architecture that contrasts with the transformer-based designs prevalent in other foundation models, (2) TimesFM, a decoder-only transformer model (in contrast to Moment's encoder-only architecture). Table D.3 summarizes the characteristics of these models, all of which have been integrated into our package (see Section D.4). We evaluated these foundation models on the Illness($H = 24$) benchmark, measuring the relative percentage improvement achieved through optimal adapter selection and corresponding hyperparameters.

Table 5.2: MSE relative improvement on the Illness($H = 24$) task ($\pm$ the standard error).

FM	Optimal adapter	MSE Imp (%)
TTM	VAE $(\beta = 2, \sigma = 1)$	$12.35_{\pm 2.51}$
TimesFM	dropoutLAE $(p = 0.1)$	$3.64_{\pm 2.81}$

Table 5.2 reveals varying performance improvements across different foundation models, with TTM exhibiting substantial gains while TimesFM showing more modest enhancements. However, these findings should be interpreted as preliminary as evaluation across additional datasets and benchmarks is necessary to fully characterize the performance of different foundation models, a direction for ongoing research. This observation

might also suggest that the effectiveness of the adapter-based approach depends on the underlying foundation model.

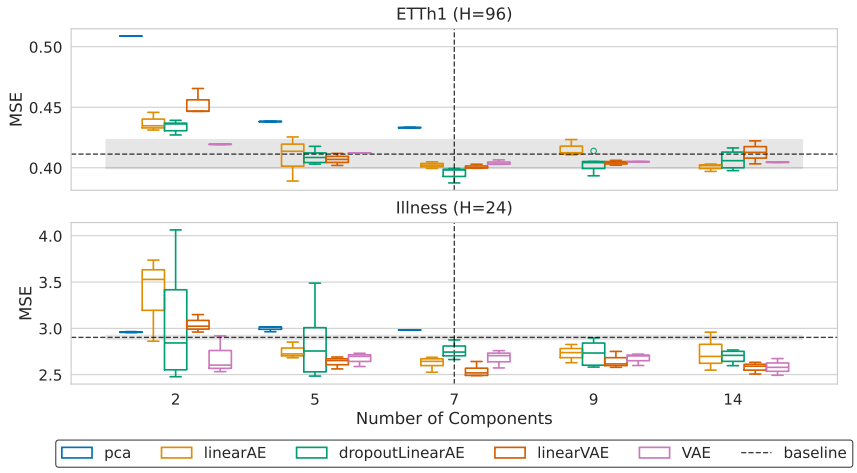


Figure 5.3: Impact of the number of components on the model performance. The horizontal dashed line indicates the performance of `Moment` without adapters. The shaded area represents its standard deviation, and the vertical line indicates the number of original features.

5.5.2 Dimensionality Reduction

Figure 5.3 illustrates the impact of varying the adapter's latent space dimension D' on the forecasting performance across different adapters. For the ETTh1 dataset with a 96-step horizon, all adapter architectures achieve optimal performance at 7 components (matching the original feature count), with MSE values consistently lower than the baseline. Notably, at just 5 components, all adapters (except the `PCA` baseline) match the baseline score, demonstrating the suitability of our framework for low-resource setups through dimensionality reduction. The Illness dataset ($H = 24$) presents more compelling results, as the `VAE` adapter achieves significantly optimal performance with only 2 components, underscoring the potential of our approach for cost-effective adaptation of time series foundation models. Ultimately, we find that expanding dimensionality beyond the original feature count does not yield further improvements, as no adapter shows notable enhancements past this point.

5.5.3 Interpretability of the latent representations

Figure 5.4 compares the representation learning capabilities of different adapters on the Illness ($H = 24$) dataset, focusing on their ability to distinguish between training

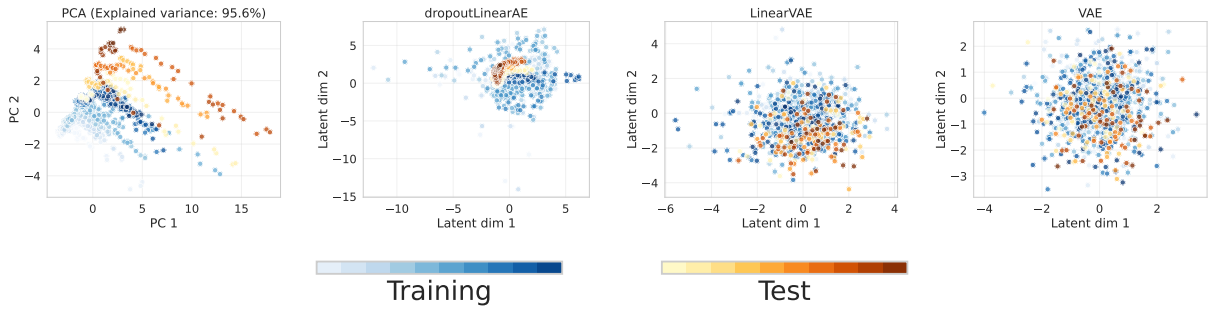


Figure 5.4: Visualization of the latent representation obtained by different adapters (with number of components equal to 2) on Illness($H = 24$). Shaded colors indicate the time dimension, with lighter colors representing earlier timesteps.

and test data. To visualize the raw dataset, we employ PCA for dimensionality reduction, retaining only two principal components, which is justified by the 95.6% explained variance. When representing the training and test datasets in the space of the first two principal components, we observe a clear distribution shift, potentially complicating the forecasting task for the baseline foundational model. In contrast, using **`AdaPTS`** results in well-overlapping Gaussian distributions for the training and test data in the latent space. This demonstrates our framework's ability to enforce a structured, isotropic representation that mitigates distribution shift. This effect is particularly pronounced with the `VAE` adapter and, to a lesser extent, with `LinearVAE` and `dropoutLinearAE`.

The findings emphasize the advantages of `VAE` in managing distribution shift, a critical challenge in time series representation learning. By modeling uncertainty and enforcing a continuous latent space, `VAE` enhance generalization, making them especially valuable for real-world applications where test distributions differ from training data. This aligns with the objective of utilizing adapters in foundational models to optimize zero-shot performance, ensuring robustness across various tasks without extensive fine-tuning.

5.5.4 On the calibration of the probabilistic adapters

To evaluate the calibration of our adapter-based probabilistic forecasters, we use quantile calibration [GBR] as depicted in the reliability diagram in Figure 5.5. In an ideal scenario, a well-calibrated probabilistic forecast should align with the red dashed diagonal, indicating that the empirical proportion of observations falls within the predicted quantiles at the expected rate. The overall conclusion is that we observe a gradual deviation from ideal calibration as the prediction horizon increases (darker shades). While early prediction horizons display reasonably well-calibrated predictions, longer-horizon forecasts systematically underestimate uncertainty, as shown by the curve falling below the diagonal. This indicates that observed values exceed predicted quantiles more fre-

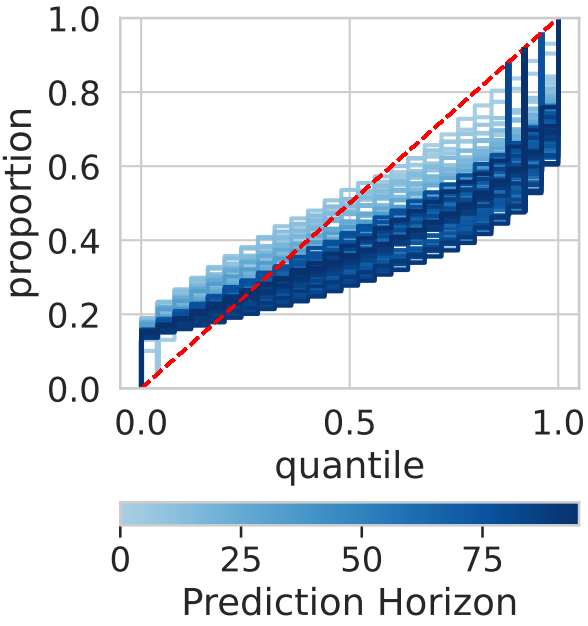


Figure 5.5: Reliability diagram for the first feature of the ETTh1 ($H = 96$) dataset using `LinearVAE`.

quently than expected, suggesting that the predictive distribution becomes too narrow, resulting in overconfident forecasts.

5.5.5 Ablation studies

Influence of σ and β in the `VAE` Adapter. Figure 5.6 illustrates an ablation study examining the β parameter in β-`VAE` and the noise scale σ of the likelihood model applied to the prediction $\hat{\mathbf{Y}}$, assessing their effects on MSE and Expected Calibration Error (ECE) [Guo+17b]. The MSE heatmap (left) demonstrates that increasing β generally diminishes MSE, with the lowest values observed at $\beta = 2.0$ and $\beta = 4.0$, particularly for higher $\log \sigma^2$. This indicates that stronger regularization through β can enhance forecasting accuracy, possibly due to the disentangling effect of regularization towards a prior distribution with statistically independent components. Conversely, the ECE heatmap (right) shows that higher β and $\log \sigma^2$ values result in lower calibration error, with optimal results at $\beta = 4.0$ and $\log \sigma^2 = 3.0$. This outcome is anticipated, as larger values of β and σ mitigate overfitting, where the model tends to exhibit overconfidence in its predictions. Additionally, it is observed that maintaining a fixed σ during training generally outperforms including it in the optimization loop, a configuration denoted as *auto* in Figure 5.6.

`LinearAE` components. The ablation study presented in Figure 5.7 examines the per-

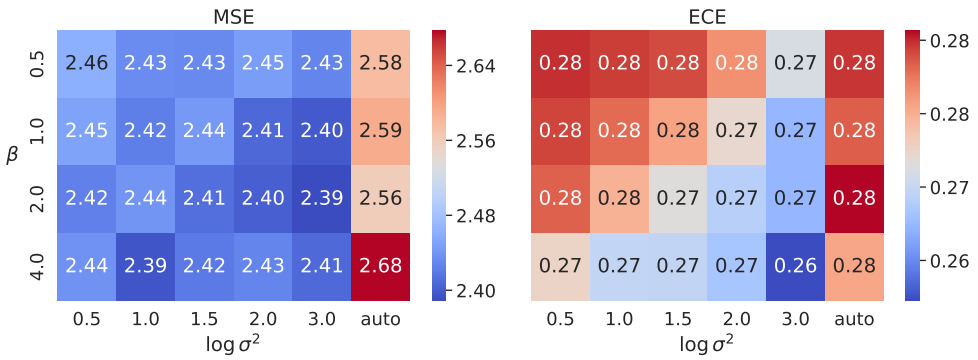


Figure 5.6: β and $\log \sigma^2$ `VAE` hyperparameters ablation on the Illness($H = 24$) dataset. For reference, the `Moment` baseline score on this task is $2.902_{\pm 0.023}$.

formance of different components of the linear autoencoder adapter (`LinearAE`) across three datasets: ETTh1, Weather, and ExchangeRate. The figure compares the full linear autoencoder with its encoder-only (`LinearEncoder`) and decoder-only (`LinearDecoder`) variants. Overall, the results reveal that the decoder component of the linear autoencoder plays the most important role. The encoder-only variant's contribution varies, being more impactful in the Weather dataset compared to ETTh1 and ExchangeRate. These findings highlight the significance of the decoder in the `LinearAE` adapter and suggest that, in the deterministic case, a decoder might be sufficient to capture feature dependencies.

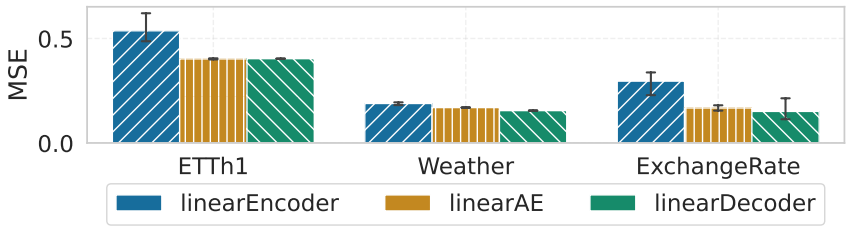


Figure 5.7: LinearAE components ablation.

Nevertheless, as shown in our previous experiments, particularly Table 5.1, probabilistic adapters generally outperformed the deterministic ones. This underscores the importance of the encoder as well, which is responsible for approximating the posterior distribution in the latent space—a mechanism inherent to our probabilistic framework.

5.6 Conclusion

In this chapter, we investigate how pre-trained univariate time series foundation models can be adapted for probabilistic multivariate forecasting. Our method, **`AdaPTS`**, offers a

novel approach to training feature space transformations that facilitate uncertainty quantification and enhance the performance of baseline foundation models. Through a series of experiments, we demonstrate that our framework improves forecasting accuracy, provides reasonably well-calibrated uncertainty estimates, reduces inference cost through dimensionality reduction, and offers interpretable feature space latent representations.

Limitations & Future directions. While our primary analysis focuses on `Moment`, our approach generalizes to other foundation models, several of which have been integrated into our publicly available package. Extensive experimentation across different foundation models and benchmark datasets remains necessary to fully characterize our approach's potential. Furthermore, while we use variational inference for computational efficiency, exploring alternative methods such as MCMC could enhance uncertainty estimation and calibration, albeit with increased computational overhead.

Chapter 6

Bridging SFT and RL: Learning implicit rewards from supervised data

Contents

RL has emerged as a powerful paradigm for adapting LLMs, offering advantages over Supervised Fine-tuning (SFT) including reduced catastrophic forgetting and improved generalization. However, these benefits require explicit reward signals, often obtained from human preferences or verifiable outcomes, which are unavailable in many cases. We address this gap by introducing a framework that derives

reward functions directly from supervised data, enabling RL-based training without additional annotation. Our approach formulates reward learning as a bilevel optimization problem: the outer level optimizes the reward parameters to maximize likelihood on the training data, while the inner level trains a policy via entropy-regularized RL. Theoretical analysis in a tractable Gaussian setting reveals that the optimal reward takes the form of a negative Mahalanobis distance, scaled by the inverse covariance of the target distribution. We extend this insight to language models by defining rewards as distances in a pretrained embedding space. Our experiments on LLM fine-tuning demonstrate that our learned rewards match the performance of oracle RL that have access to ground-truth rewards. Additional results on synthetic data, tabular classification and dynamics modeling further validate the generality of our framework.

6.1 Introduction

Generative models have become central to modern machine learning research, driving advances in text [Bro+20; Dee+25], image [Rom+21; Ram+], and multimodality [Zha+24a; Bai+25; Fu+25; Łaj+24; Yin+24] under the umbrella of "Generative AI" (*GenAI*). Their ability to synthesize realistic content has made them foundational in applications ranging from decision making [Shi+25; Kim+24; Int+25b] to scientific discovery [Man+23; Lu+24].

Traditionally, such models are trained via MLE, where the parameters of the generative model are optimized to maximize the probability of observed data. This approach provides a principled framework for fitting models to large datasets and remains the backbone of many learning pipelines for generative models. Notably, this approach is omnipresent in today's *Large Language Models* (LLMs) through the *next token prediction* paradigm [Vas+23; Bro+20; Dee+25].

However, recent breakthroughs in LLMs research, demonstrate the limitations of MLE alone. Techniques based on RL [Bel58], such as *Reinforcement Learning from Human Feedback* (RLHF) [Chr+17; Sti+20] and more recently *Reinforcement Learning from Verifiable Rewards* [Sha+24; Dee+25], have proven more effective than SFT at aligning models with human preferences and improving generation quality [SPA25; Lai+25; Swa+25]. These methods leverage explicit or implicit reward signals to guide training beyond likelihood objectives.

In many real-world scenarios, explicit reward functions for the tasks we aim to solve are not readily available. Instead, we often have access to high-quality datasets that we wish to use for aligning our models. Depending on the structure of these datasets, several techniques have been proposed to derive reward functions, such as from preference data [Raf+23] or from demonstrations [Fin+16; FLA16] when framed within a Markov

Decision Process (MDP) formalism. Despite these advances, the fundamental question surrounding this problem remains unresolved:

*How can we learn **implicit reward functions** from **supervised data**?*

In this chapter, we propose the following contributions toward addressing this question:

- **Bilevel optimization perspective on MLE:** We adopt a reward parametrization in the form of a negative scaled distance between generative samples and targets. We then reinterpret the MLE objective as a Bilevel Optimization (Bi-O) problem where the outer-level optimizes over the reward parameters, and the inner-level is defined by an RL objective in the model parameters.
- **Theoretical analysis:** We study this formulation for a Gaussian linear model, deriving insights into the theoretically optimal parameters of the reward function.
- **Experimental results:** Guided by the theoretical analysis and leveraging implicit differentiation solvers, we propose practical algorithms for addressing the bilevel optimization problem. We evaluate these algorithms on synthetic data, LLM fine-tuning, and tabular regression and classification tasks.

The remainder of the chapter is organized as follows: Section 6.2 positions our work within the relevant literature, while Section 6.3 introduces the problem setup and motivates our approach. In Section 6.4, we address the theoretical analysis of the bilevel optimization problem in the Gaussian case. We showcase our experimental results in Section 6.5, and conclude with a discussion in Section 6.6.

6.2 Related Work

RL vs SFT. The tension between maximum likelihood estimation and reinforcement learning-based training has a long history in sequence modeling. Early work identified fundamental limitations of MLE in autoregressive models, including compounding errors and exposure bias [Ben+15b; VHB15a; Ran+16; Bah+17; Tan+19]. These observations motivated policy gradient alternatives such as REINFORCE-based training [Wil92a], Reward-Augmented Maximum Likelihood [Nor+16], and scheduled transitions from MLE to RL [Ran+16]. More recently, this debate has intensified in the context of large language model fine-tuning. [SPA25], [Lai+25], and [Swa+25] provide theoretical and empirical evidence that on-policy RL fine-tuning mitigates catastrophic forgetting more effectively than supervised fine-tuning, as RL updates converge to solutions closest in KL divergence to the original policy. Complementing these findings, [Zhu+25] demonstrate that RL with verifiable rewards provably explores reasoning paths that supervised fine-tuning cannot reach. Our work addresses a key bottleneck in leveraging these RL advantages: the need for explicit reward functions.

Reward modeling. Existing approaches to reward learning typically require structured supervision beyond standard labeled data. Learning from preferences [BT52; Ouy+22; Raf+23] transforms pairwise comparisons into reward models and has become standard for LLM post-training [Tou+23; Dai+24; Sha+24; Dee+25]. However, this requires costly preference annotations. Inverse RL methods [AN04; Zie+08; Fin+16; FLA16] learn rewards under which expert demonstrations are optimal, assuming access to state-action trajectories within an MDP formalism. Goal-conditioned approaches define rewards via spatial [Nac+18; Maz+24], temporal [Har+20; Wan+25], or semantic [Son+23; Fan+22] distances to target states. Our framework complements these paradigms by deriving rewards from supervised data without requiring preferences or an MDP structure.

Connection to Maximum Entropy Inverse RL. Our bilevel formulation shares conceptual ties with the MaxEnt IRL framework [Zie+08], which recovers reward functions that make observed behavior optimal under an entropy-regularized policy. The closest prior work is [Zen+22], who combine MLE with MaxEnt IRL for imitation learning in control tasks. However, their approach is tailored to episodic RL setting, while our framework targets general supervised learning problems. For LLM finetuning, the closest work to ours is [Li+24] who consider a RLHF-typical LLM-based reward model in a similar bilevel formulation. We distinguish ourselves by instead considering a more general reward parametrization based on distances between embeddings.

6.3 Preliminaries

In Section 6.3.1, we begin by motivating the idea of learning reward functions from data, outlining scenarios in which RL methods may be preferred over MLE. We then formally define the problem setup in Section 6.3.2 and the proposed reward modeling framework in Section 6.3.3.

6.3.1 Motivation

In RL, Policy Gradient (PG) methods are traditionally viewed as producing unbiased yet high-variance gradient estimates, especially in long-horizon or high-dimensional tasks [GBB01]. In contrast, MLE has historically served as the dominant paradigm in supervised learning and probabilistic modeling [Aka73]. However, in the current era of large pretrained models and advanced RL algorithms, these limitations have become less restrictive, giving rise to many cases where PG methods are more advantageous than MLE.

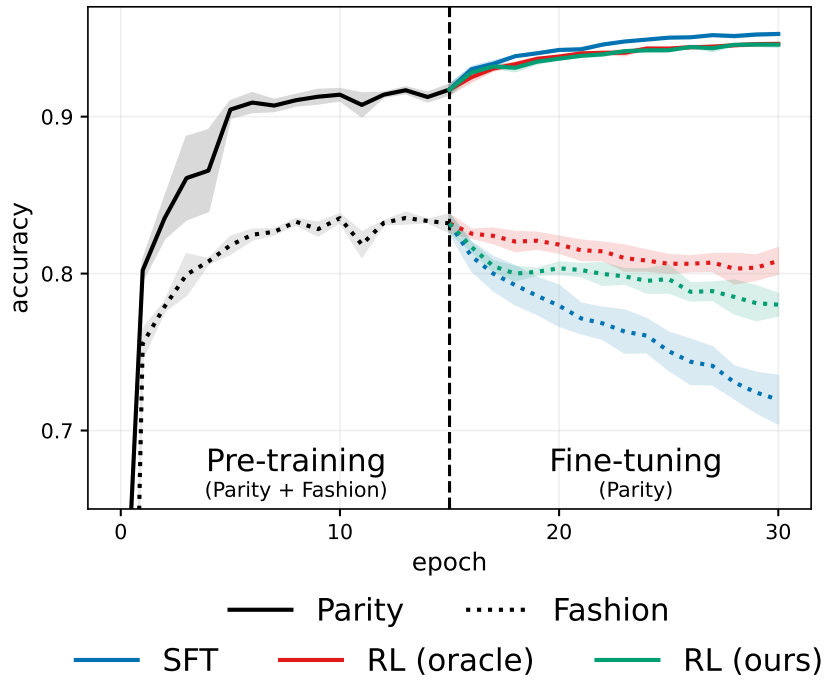


Figure 6.1: **RL using rewards learned from SFT data.** Our framework enables the learning of reward functions solely from the data presented to SFT, achieving a level of catastrophic forgetting mitigation comparable to RL with access to the oracle reward function.

Catastrophic forgetting is an area where these advantages are the most apparent. When adapting LLMs to downstream tasks through post-training, it is often desirable to preserve prior knowledge while specializing to new distributions. Recent studies [SPA25; Lai+25; Swa+25; Zhu+25] suggest that on-policy RL fine-tuning achieves this balance more effectively than SFT, since its updates converge to solutions closest in KL divergence to the original policy.

To elucidate the motivation behind our approach, we conduct an experiment to demonstrate how our method can mitigate the issue of catastrophic forgetting. Specifically, we replicate the experimental setup from section 4 of [SPA25], where the *ParityMNIST* task is introduced. This task involves predicting the parity of the number in the input image, where the predicted parity corresponds to the parity of the predicted class $(0 \to 9)$. The experiment begins with a pre-training phase, during which an MLP model is trained on data from both *ParityMNIST* and *FashionMNIST*, where the latter consists in classifying 10 different cloth types. Subsequently, we run a fine-tuning phase focused solely on the *ParityMNIST* task. During this phase, we compare SFT (cross-entropy minimization) and RL (based on the REINFORCE algorithm) with either of the oracle parity reward or the reward function derived from our framework. Given that the model outputs logits for 10 classes, multiple optimal policies exist for *ParityMNIST*, e.g. mapping even $\to 0$

and odd $\to 1$, or even $\to 0, 2, 4, 6, 8$ and odd $\to 1, 3, 5, 7, 9$). As shown in Figure 6.1, by using the same data as SFT, and assuming that the true reward function is unknown, we are able to learn a reward function through our framework, achieving a similar reduction in catastrophic forgetting as the RL oracle variant, which has access to the optimal parity reward. These observations motivate our approach: we propose a general framework that interprets data signals as reward functions, thereby also enabling RL methods.

6.3.2 Problem setup

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, and let $X : \Omega \to \mathcal{X}$ and $Y : \Omega \to \mathcal{Y}$ be two random variables, with $\mathcal{X} \subseteq \mathbb{R}^m$ and $\mathcal{Y} \subseteq \mathbb{R}^n$, where $(n, m) \in \mathbb{N}_\star^2$. Consider a maximum likelihood estimation problem where we observe N iid realizations $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=0}^N$ from a fixed unknown distribution over $\mathcal{X} \times \mathcal{Y}$. The goal is to model the conditional distribution $Y|X \sim q$ using a parametric model $\widehat{Y}|X \sim \hat{p}_\theta$ where $\theta \in \Theta := \mathbb{R}^{d_\theta}$ are parameters spanning a finite dimensional space with dimension d_θ. In the MLE formalism, we optimize the parameters θ by maximizing the log-likelihood:

$$\theta^\star = \arg\max_{\theta \in \Theta} \mathbb{E}_X \mathbb{E}_{Y|X \sim q}[\log \hat{p}_\theta(Y|X)]. \tag{MLE}$$

A parallel approach, based on reinforcement learning, consists in maximizing a reward function $r : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ that evaluates the quality of generated $\hat{\mathbf{y}}$ against the true observations $\mathbf{y}$, resulting in the Policy Gradient (PG) objective. Here we state the entropy-regularized PG objective, a variant that is commonly considered in RL algorithms [Haa+17; Haa+18a; Wen+24]:

$$\theta^\star = \arg\max_{\theta \in \Theta} \mathbb{E}_X \mathbb{E}_{Y|X \sim q}\Big[\mathbb{E}_{\widehat{Y}|X \sim \hat{p}_\theta}\Big[r(\widehat{Y}, Y)\Big] + \lambda \mathrm{H}(\hat{p}_\theta)\Big], \tag{PG}$$

where $\lambda > 0$ is a parameter controlling the strength of the regularization, and H denotes the entropy.

In this work, we ask whether the reward function itself can be seen as an optimization variable r over a Hilbert space $\mathcal{H}$. The *optimal reward function* is then determined based on the MLE objective, which now represents the outer-level of the following bilevel optimization problem:

Definition 6.3.1 (Bilevel Optimization problem)**.**

$$\max_{r\in\mathcal{H}} \mathbb{E}_X \mathbb{E}_{Y|X\sim q}\big[\log \hat{p}_{\theta_r^\star}(Y|X)\big] \quad s.t. \tag{Bi-O}$$
$$\theta_r^\star = \arg\max_{\theta\in\Theta} \mathbb{E}_X \mathbb{E}_{Y|X\sim q}\Big[\mathbb{E}_{\widehat{Y}|X\sim\hat{p}_\theta}\big[r(Y',Y)\big] + \lambda \mathrm{H}(\hat{p}_\theta)\Big]$$

6.3.3 Reward Modeling Framework

Given the definition of a reward function as evaluating the quality of generative samples $\hat{\mathbf{y}}$ against true labels $\mathbf{y}$, we assume that a valid reward has to be maximal in $\mathbf{y}$: $\mathbf{y} \in \max_{\hat{\mathbf{y}}} r(\hat{\mathbf{y}}, \mathbf{y})$. Therefore, we naturally define the reward function as a negative quadratic form of the difference between generative samples and true labels:

Definition 6.3.2 (Reward model)**.** *Let $\mathrm{U} \in S_n^{++}(\mathbb{R})$, we define the reward model as the following quadratic form: $\forall(\hat{\mathbf{y}}, \mathbf{y}) \in \mathbb{R}^n \times \mathbb{R}^n, \quad r_{\mathrm{U}}(\hat{\mathbf{y}}, \mathbf{y}) = -(\hat{\mathbf{y}} - \mathbf{y})^T \mathrm{U} (\hat{\mathbf{y}} - \mathbf{y})$.*

The matrix U represents the reward parameters we will attempt to optimize over when solving Equation (Bi-O). In the case of complex data modalities such as images or text, we define a modified version where the reward measures distances in a precomputed embedding space:

Definition 6.3.3 (Embedding reward model)**.** *Let $\mathrm{U} \in S_d^{++}(\mathbb{R})$ and an embedding model $\mathrm{e} : \mathcal{Y} \to \mathbb{R}^d$, we define the embedding reward model as the following quadratic form in the $\mathbb{R}^d$ space: $\forall(\hat{\mathbf{y}}, \mathbf{y}) \in \mathbb{R}^n \times \mathbb{R}^n$,*
$r_{\mathrm{U}}(\hat{\mathbf{y}}, \mathbf{y}) = -(\mathrm{e}(\hat{\mathbf{y}}) - \mathrm{e}(\mathbf{y}))^T \mathrm{U} (\mathrm{e}(\hat{\mathbf{y}}) - \mathrm{e}(\mathbf{y}))$.

In the next section We will theoretically analyze the closed form optimal parametrization $\mathrm{U}^\star$ in the Gaussian case.

6.4 Theoretical analysis

6.4.1 The Gaussian case

We start our analysis by stating the following assumption on the data generating distribution:

Assumption 6.4.1 (Gaussian linear model)**.** *We assume that both the true conditional density q and the model density $\hat{p}_\theta$ are Gaussian distributions with linear mean functions and fixed covariance matrices:*

$$Y \mid X \sim q := \mathcal{N}(\Lambda X, \Sigma), \quad \widehat{Y} \mid X \sim \hat{p}_\theta := \mathcal{N}(\mathrm{A}X, \mathrm{B}),$$

where $\Lambda \in \mathbb{R}^{n\times n}$, $\Sigma \in \mathrm{S}_n^{++}(\mathbb{R})$[1], *and* $\theta := (\mathrm{A}, \mathrm{B}) \in \Theta := \mathbb{R}^{n\times n} \times \mathrm{S}_n^{++}(\mathbb{R})$.

We now state the main results, showcasing closed-form solutions of the Bi-O problem, along with a follow-up corollary showing the isotropic case.

Proposition 6.4.1 (Closed-form solution)**.** *Under assumption 6.4.1 and definition 6.3.2, the Bi-O problem*

$$\mathrm{U}^\star = \underset{\mathrm{U}\in S_n^{++}(\mathbb{R})}{\arg\max}\ \mathbb{E}_X \mathbb{E}_{Y|X\sim q}\Big[\log \hat{p}_{\theta^\star_{\mathrm{U}}}(Y|X)\Big] \qquad s.t.$$
$$\theta^\star_{\mathrm{U}} = \underset{\theta\in\Theta}{\arg\max}\, \mathbb{E}_X \mathbb{E}_{Y|X\sim q}\Big[\mathbb{E}_{\widehat{Y}|X\sim\hat{p}_\theta}\Big[r_{\mathrm{U}}(\widehat{Y}, Y)\Big] + \lambda \mathrm{H}(\hat{p}_\theta)\Big]$$

has exactly one solution that writes:

$$\mathrm{U}^\star = \tfrac{\lambda}{2}\Sigma^{-1}.$$

The proof of proposition 6.4.1 is deferred to Section E.4.1.

Corollary 6.4.1 (Isotropic case)**.** *For* $\mathrm{B} = \sigma^2 \mathrm{I}_n$*, the set of solutions of Bi-O is:*

$$\mathrm{U}^\star \in \mathrm{F}_{\lambda,\Sigma} := \left\{ \mathrm{U} \in S_n^{++}(\mathbb{R}) \,\middle|\, \mathrm{Tr}(\mathrm{U}) = \frac{\lambda n^2}{2\,\mathrm{Tr}(\Sigma)} \right\}.$$

Note that, for any given $\lambda > 0, \Sigma \in \mathrm{S}_n^{++}(\mathbb{R})$, $\mathrm{F}_{\lambda,\Sigma} \neq \varnothing$ since $\frac{\lambda n}{2\,\mathrm{Tr}(\Sigma)}\mathrm{I}_n \in \mathrm{F}_{\lambda,\Sigma}$ which corresponds to reward functions we consider for the empirical experiments on synthetic data in Section 6.5.1.

Interpretation as Mahalanobis distance. We observe that the optimal reward function is characterized by a matrix $\mathrm{U}^\star \in \mathrm{S}_n^{++}(\mathbb{R})$ that is inversely proportional to the covariance matrix of the data-generating process. With this choice, the reward admits a clear interpretation as the negative squared Mahalanobis distance [Mah36], which can be seen as the Euclidean distance between vectors after applying the linear transformation $\mathrm{U}^{\star 1/2}$. This perspective suggests that, in practice, the noisier the data, the less strongly the model should be penalized for deviations from samples. Finally, the scaling by λ serves to balance the two competing optimization objectives, reward maximization and entropy regularization.

Interpretation as reverse KL minimization. An interesting observation arises when substituting the optimal reward parametrization $\mathrm{U}^\star$ into the inner-level objective: the PG formulation becomes equivalent to minimizing the reverse Kullback-Leibler (KL)

[1] We denote by $\mathrm{S}_n^{++}(\mathbb{R})$ the set of real symmetric positive definite $n \times n$ matrices.

divergence between the model distribution $\hat{p}_\theta$ and the data-generating distribution q. This connection (also established in [Swa+25]), provides an explanation for the empirical results presented in the next section. We therefore state it as a corollary, with the proof deferred to Section E.4.3:

Corollary 6.4.2. *Under the assumptions of proposition 6.4.1, the optimal parameters $\theta^\star_{\mathrm{U}^\star}$ obtained from the lower-level problem with $\mathrm{U}^\star = \frac{\lambda}{2}\Sigma^{-1}$ minimize the reverse KL divergence between $\hat{p}_\theta$ and q:*

$$\theta^\star_{\mathrm{U}^\star} = \arg\min_{\theta\in\Theta} \mathbb{E}_X \left[d_{KL} \left(\hat{p}_\theta(\cdot|X) \,\|\, q(\cdot|X) \right) \right].$$

In contrast to the previous analysis, where we assumed access to the data-generating distribution and provided a closed-form solution to problem Bi-O, real-world applications typically do not satisfy such assumptions. Consequently, solving the bilevel optimization problem Bi-O by directly optimizing the outer objective offers a more general approach applicable to a broader class of problems.

Bilevel optimization solvers can generally be divided into three categories. *Explicit gradient* methods treat the gradient update as a differentiable mapping and backpropagate through the unrolled optimization path of the inner-level problem [Fra+17]. *Gradient-free methods* instead rely on evolutionary strategies, optimizing the outer objective while considering the inner problem as a black-box function [Son+20; Fen+21]. Finally, implicit differentiation methods leverage the implicit function theorem to reformulate gradient estimation as the solution to a linear system [Dag+24; PMA24].

In this work, we focus on implicit differentiation, as explicit gradient methods often encounter memory issues from storing long computational graphs, while gradient-free approaches are generally limited by the curse of dimensionality. We provide a detailed review of bilevel optimization solvers in Section E.1.

6.4.2 Implicit differentiation

Consider this time a general reward parametrization r_ϕ with $\phi \in \Phi := \mathbb{R}^{d_\phi}$, where d_ϕ denotes the dimension of the reward parameter space. The optimization of the outer-level problem can thus be restricted to the Hilbert space of reward functions spanned by parameters $\phi \in \Phi$. Within this setup, implicit differentiation treats the solution of the inner problem, $\theta^\star$, as an implicit function of ϕ and allows one to compute the best-response derivatives $\nabla_\phi \theta^\star(\phi)$ analytically via the implicit function theorem.

To proceed, we define an operator $\mathrm{f} : \Phi \times \Theta \to \Theta := \mathbb{R}^{d_\theta}$ whose roots characterize the inner-level optimal solution $\theta^\star(\phi)$. That is, for all $\phi \in \Phi$, we have $\mathrm{f}(\phi, \theta^\star(\phi)) = 0$. Leveraging this property, the derivative of interest $\nabla_\phi \theta^\star(\phi)$ can be determined by solving

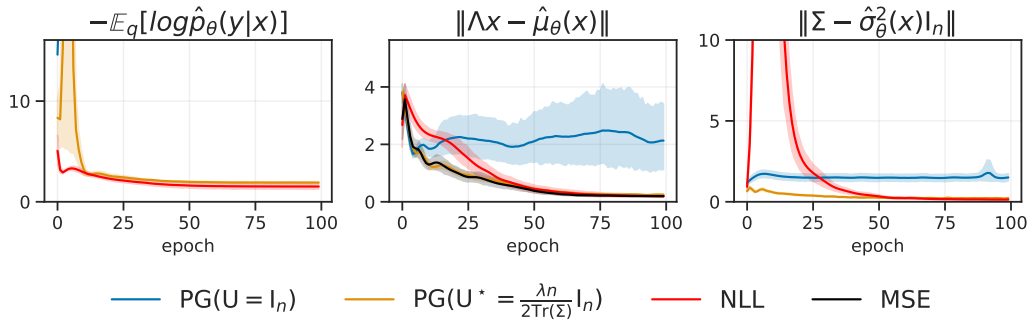


Figure 6.2: **Synthetic data experiment.** The PG loss when paired with the optimal reward function matches the NLL-trained baseline in terms of NLL (left panel), all while having faster convergence in terms of moment matching (center and right panels for the mean and variance, respectively).

for $\nabla_\phi \mathrm{f}(\phi, \theta^\star(\phi)) = 0$:

$$\forall \phi \in \Phi, \quad \nabla_\theta f(\phi, \theta^\star(\phi))\, \nabla_\phi \theta^\star(\phi) + \nabla_\phi f(\phi, \theta^\star(\phi)) = 0, \tag{6.1}$$

where $\nabla_\phi \theta^\star(\phi)$ is obtained by solving the linear system in Eq. (6.1), enabling gradient descent on the outer problem via the chain rule.

In our bilevel optimization formulation, the operator f arises naturally from the fixed-point characterization of the gradient update: $\mathrm{f}(\phi, \theta) = \alpha \nabla_\theta \mathcal{L}_{\text{in}}(\phi, \theta)$ where $\mathcal{L}_{\text{in}}$ is the inner-level objective and α is a learning rate. Under this definition, the first-order optimality condition holds whenever the inner-level optimization converges to a local minimum $\theta^\star(\phi)$ where the gradient vanishes. We assume that this is a plausible hypothesis given any modern stochastic optimizer (e.g. Adam [KB17]).

6.5 Experimental results

6.5.1 Synthetic data

In this section, we empirically evaluate the theoretical results from Section 6.4.1. To this end, we generate synthetic data that satisfy assumption 6.4.1: $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=0}^N$, where $\mathbf{x}_i \sim \mathcal{U}([-5, 5]^n)$ ($\mathcal{U}$ denotes the uniform distribution), and $\mathbf{y}_i \sim q(.|\mathbf{x}_i) := \mathcal{N}(\Lambda \mathbf{x}_i, \Sigma)$ with diagonal covariance matrix $\Sigma = \beta^2 \mathrm{I}_n$ and $\beta > 0$. For the model $\hat{p}_\theta$, we *relax* the linearity and homoscedasticity assumptions by considering a neural network that parametrizes a Gaussian distribution, in which both the mean function and the diagonal covariance matrix depend on the input: $\hat{p}_\theta(.|\mathbf{x}_i) := \mathcal{N}(\mu_\theta(\mathbf{x}_i), \sigma^2_\theta(\mathbf{x}_i)\mathrm{I}_n)$. We compare baselines trained with the *negative log-likelihood* (NLL) and *mean squared error* (MSE)

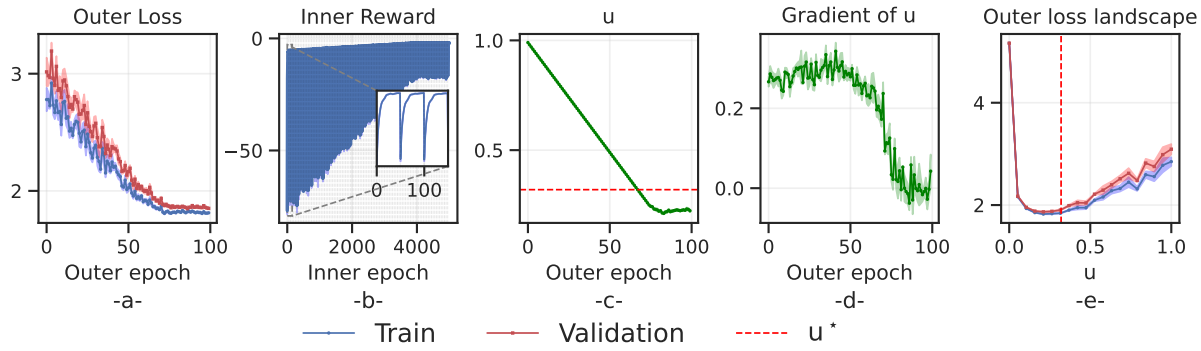


Figure 6.3: **Implicit differentiation solver on synthetic data experiment.** From left to right: **-a-** outer loss (NLL), **-b-** inner reward optimization loop, **-c-** trajectory of the reward parameter u, **-d-** gradient of the outer loss with respect to u, **-e-** the outer loss landscape.

losses against PG variants (specifically based on the REINFORCE algorithm), using either a negative squared distance reward $\mathrm{U} = \mathrm{I}_n$ or the optimal reward function derived in corollary 6.4.1 with $\mathrm{U}^\star = \frac{\lambda n}{2\,\mathrm{Tr}(\Sigma)}\mathrm{I}_n$.

The closed form solution. Figure 6.2 shows how the validation NLL and moment-matching errors (mean and covariance) change over training. Consistent with theory, we observe that adjusting the reward function with the optimal matrix $\mathrm{U}^\star$ yields a learning curve nearly identical to the NLL baseline (yellow and red curves in the left panel of Figure 6.2). Moreover, the PG variant with the optimal matrix converges faster than the NLL baseline in matching the moments of the data-generating distribution (center and right panel). Finally, we note that the vanilla PG method (with $\mathrm{U} = \mathrm{I}_n$) suffers from a diverging NLL due to the variance shrinking to zero for some values of λ which leads to numerical instabilities.

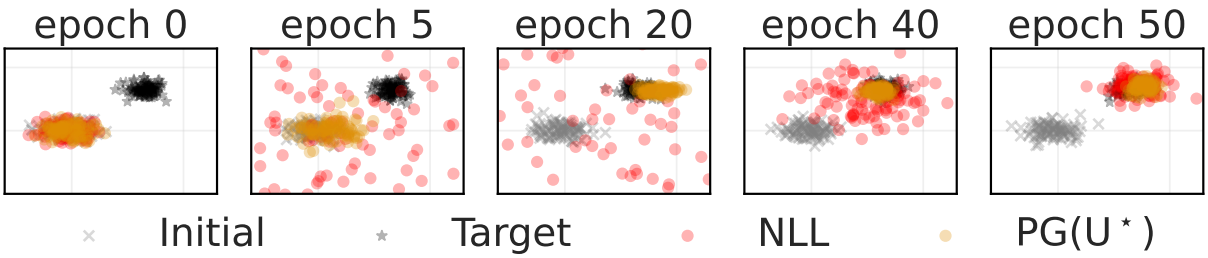


Figure 6.4: **The learned distribution dynamics for a single training example.** While NLL initially expands variance to cover the target (*mass-covering*), PG maintains a compact shape and shift directly toward the target mode (*mode-seeking*).

To illustrate the training dynamics, Figure 6.4 tracks the evolution of the learned distribution for a single data point across training epochs. The PG variant with the optimal reward shows a direct and stable convergence toward the target mode. In contrast, the NLL baseline initially over-disperses before gradually contracting to match the target distribution. This difference is consistent with corollary 6.4.2: minimizing $d_{KL}\left(\hat{p}_\theta \,\|\, q\right)$ induces mode-seeking behavior, encouraging the learned distribution to concentrate on the dominant mode of q.

Implicit differentiation solvers. To evaluate implicit differentiation-based bilevel solvers using the same synthetic data, we use TorchOpt [Ren+23], a python package that enables differentiable optimization solvers that can be integrated with Pytorch [Pas+19] neural network implementations. More specifically, we run the implicit differentiation-based solvers using the Conjugate Gradient algorithm for the linear system resolution, as in [Raj+19].

Figure 6.3 presents the results of running an implicit differentiation solver for 100 outer iterations, each with 50 inner iterations, and a learning rate of 10^{-2} for both optimization loops. The outer optimization variable is a single parameter (panel **-c-**) initialized at 1, which defines the diagonal Mahalanobis matrix for the reward: $\mathrm{u} > 0$ s.t. $\mathrm{U} = \mathrm{u} \cdot \mathrm{I}_n$. Panel **-a-** illustrates the evolution of the outer loss (NLL evaluated on the optimal policy from the inner PG loop), showing clear improvement relative to the initialization at 1 (which corresponds to the Euclidean distance). Additionally, the optimization parameter u converges to a value close to the theoretical optimum $\mathrm{u}^\star = \frac{\lambda n}{2\,\mathrm{Tr}(\Sigma)}$, as derived in corollary 6.4.1. This convergence is further supported by panel **-e-**, which plots the outer loss landscape as a function of the reward parameter u, revealing a roughly convex landscape with a global minimum near the theoretical optimum. These results validate our intuition from the tractable case discussed in Section 6.4, even in the more general setting of stochastic optimization solvers within the implicit differentiation framework.

Beyond synthetic data. For the remainder of the chapter, our goal is to leverage the intuition gained from the previous analysis to derive practical algorithms that we can validate on LLM finetuning and other MLE tasks. For this, we build on the theoretical analysis in Section 6.4.1 to suggest a realistic way to estimate the optimal reward parametrization $\mathrm{U}^\star$. The main challenge with this approach lies in estimating the covariance matrix-dependent term. As stated in Algorithm 17, we propose an approximate approach that estimates an empirical covariance matrix $\hat{\Sigma}$ from the training data. In the following, we use Algorithm 17 to benchmark our framework against vanilla PG and MLE in three real-world applications: LLM finetuning, tabular classification, and model-based reinforcement learning. Note that, in the experiments, we are effectively solving the inner-level problem of the Bi-O formulation, while substituting the reward function

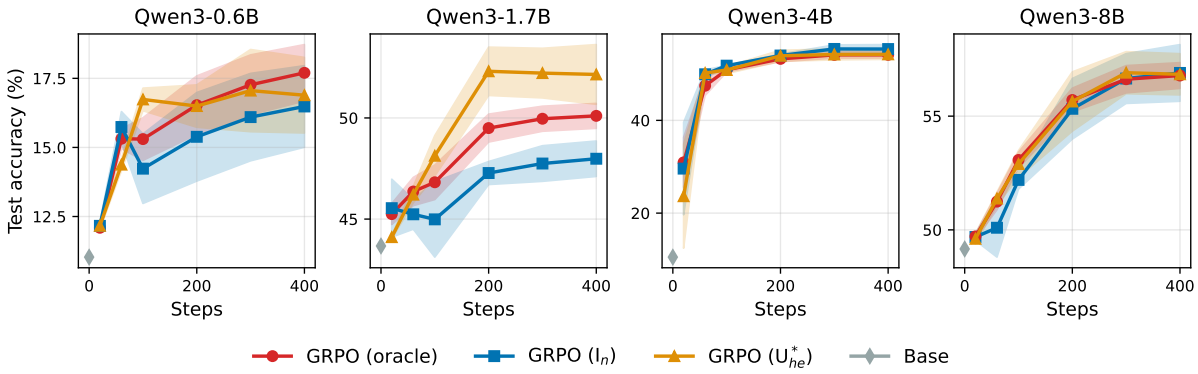


Figure 6.5: **Qwen3 fine-tuning using GRPO on the Countdown task.** Using our embedding reward models with either $\mathrm{U}_{\mathrm{he}}^{\star}$ or I_n as parameters, and `EmbeddingGemma` as embedding model, we are able to achieve the same levels of accuracy as the oracle method that uses the true reward.

either with the optimal matrix $\mathrm{U}_{\mathrm{he}}^{\star}$ or with the identity I_n for the squared-distance baseline.

Algorithm 17 PG/GRPO$(\mathrm{U}_{\mathrm{he}}^{\star})$ - heuristic

Input: Data $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=0}^{N}$, model $\hat{p}_\theta$, λ, $\mathrm{e} : \mathcal{Y} \to \mathbb{R}^d$
1. Estimate cov matrix $\hat{\Sigma} = \mathrm{cov}(\{\mathrm{e}(\mathbf{y}_i)\}_{i=0}^{N})$
2. loss $\leftarrow$ PG/GRPO$(\mathrm{U}_{\mathrm{he}}^{\star}(\lambda, \hat{\Sigma}))$
3. `train_policy`$(\hat{p}_\theta, \mathcal{D}, \mathrm{loss}, \mathrm{e})$
Return: learned model $\hat{p}_{\theta^\star}$

6.5.2 LLM fine-tuning

As discussed in the motivation section, fine-tuning large models is the setting in which our method is expected to have the greatest impact. In the case of LLMs, we substitute the standard policy gradient approach with the Group-Relative Policy Optimization (GRPO) algorithm [Sha+24], which is specifically designed for LLMs and has demonstrated strong performance across a range of environments. The reward model is constructed using the embedding reward formulation (definition 6.3.3), where the embedding function $\mathrm{e} : \mathcal{Y} \to \mathbb{R}^d$ can be instantiated with any text embedding model, such as `RoBERTa` [Liu+20]. All the implementation details are included in Section E.2.

We focus on the Countdown task, in which an LLM must identify the correct algebraic operations applied to a given set of numbers (e.g., $nums = [93, 78, 46]$, $target = 61$, with the expected solution being $93 - (78 - 46) = 61$). For this purpose, we rely on the `HuggingFaceTB/Countdown-Task-GOLD` dataset, which comprises 30k samples of an-

notated questions and answers, along with their corresponding reasoning traces: e.g. "<think> To solve this, I need to use the numbers 78, 46, and 93 to create an equation that equals 61. I'll start by [...] </think><answer>93 - (78 - 46) = 61</answer>". The oracle reward function for the Countdown task is defined as a binary signal, assigning a reward of 1 to correct equations that use all the required numbers, and 0 otherwise.

Figure 6.5 reports the performance of several checkpoints extracted from training runs of 400 steps, evaluated on a held-out test set of 10k questions. We find that experiments using GRPO with our embedding-based reward function achieve performance comparable to that of the GRPO oracle baseline, which has access to the true reward. These results suggest that the proposed reward parametrization provides sufficient learning signal to fine-tune LLMs solely from annotated data, thereby extending the applicability of RL fine-tuning beyond strictly verifiable tasks.

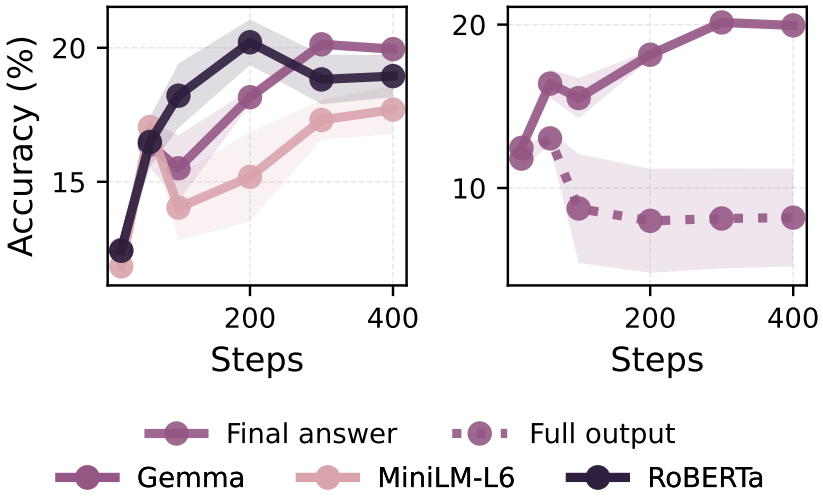


Figure 6.6: **Ablation.** For `Qwen3-0.6B`, the optimal performance is observed using the `EmbeddingGemma` embedding model, and only the final answer when computing rewards.

Ablation studies. First, we note from Figure 6.5 that the performance of our approach remains stable across different model sizes, indicating robustness with respect to the initial capability of the base model. In addition, we perform an ablation study examining the impact of the embedding model, as illustrated in the left panel of Figure 6.6. We observe that the choice of the embedding model is indeed important as `EmbeddingGemma` and `RoBERTa` show the best performance compared to the less capable `MiniLM-v6` model. Finally, we investigate whether the embedding reward function should be computed using the `Full output` or only the `Final answer` in the right panel of Figure 6.6. We conclude that our reward modeling framework strongly benefits from computing the reward only on the final answer, highlighting the importance of a structured output when

computing distances in the embedding space.

6.5.3 Tabular data

To further validate our findings, we will attempt the more ambitious quest of pretraining models using RL, focusing specifically on tabular data applications.

Classification. We evaluate our framework on two tabular classification datasets from the UCI repository [Wan23]. Specifically, we train a multiclass logistic regression model with PG (specifically Reinforce), where the reward is defined as an embedding reward model (definition 6.3.3) where $\mathrm{e} : \mathcal{Y} \to \mathbb{R}^d$ is the one-hot encoding of the predicted and true label. Table 6.1 summarizes the accuracy and AUC results across the considered datasets. On the binary imbalanced *Credit default* dataset, the accuracy is high across all methods but AUC reveals that $\mathbf{PG}(\mathrm{U}^\star_{\mathrm{he}})$ better separates classes. The *Poker* dataset remains challenging for all methods, yet $\mathbf{PG}(\mathrm{U}^\star_{\mathrm{he}})$ still provides the best performance.

Dataset	Method	Accuracy$_{/10^{-2}\uparrow}$ (AUC$_{/10^{-2}\uparrow}$)
Credit default	**NLL**	$79.8 \pm .012$ (70.5 ± .005)
	$\mathbf{PG}(\mathrm{I}_n)$	$75.5 \pm .023$ (57.7 ± .632)
	$\mathbf{PG}(\mathrm{U}^\star_{\mathrm{he}})$	$\mathbf{82.0} \pm .001$ (**71.3** ± .001)
Poker	**NLL**	$48.6 \pm .001$ (NA)
	$\mathbf{PG}(\mathrm{I}_n)$	$38.2 \pm .039$ (NA)
	$\mathbf{PG}(\mathrm{U}^\star_{\mathrm{he}})$	$\mathbf{52.4} \pm .001$ (NA)

Table 6.1: Tabular classification results.

Regression. We study a dynamics modeling problem within a model-based reinforcement learning setting. The objective is to predict the next state $s_{t+1} \in \mathbb{R}^{d_s}$ from the current state–action pair $(s_t, a_t) \in \mathbb{R}^{d_s+d_a}$, where t denotes the interaction timestep. Given a dataset $\mathcal{D} = (s_t^i, a_t^i, s_{t+1}^i)_{i=0}^N$, the dynamics model is commonly formulated as a Gaussian probabilistic predictor trained via MLE [Chu+18; Jan+19]. This standard formulation naturally fits our framework by directly applying the no-embedding reward model (definition 6.3.2) in the state space. Table 6.2 summarizes the average MSE and NLL over three HalfCheetah tasks (see Section E.3 for implementation details). As anticipated, models optimized for NLL achieve the best NLL scores. Nonetheless, in line with the synthetic experiments in Section 6.5.1, the PG objective with the optimal reward heuristic $\mathbf{PG}(\mathrm{U}^\star_{\mathrm{he}})$ yields substantial NLL gains over PG with the negative squared-distance reward $\mathbf{PG}(\mathrm{I}_n)$. Furthermore, the reward learned via implicit differentiation, $\mathbf{PG}(\mathrm{U}^\star_{\mathrm{im}})$,

attains the second-best NLL while also achieving the best MSE, a desirable trade-off in MBRL settings, particularly when the learned dynamics are used for deterministic planning.

Task	Method	MSE$_{/10^{-2}\downarrow}$	NLL$_{/10^{-2}\downarrow}$
HalfCheetah	**NLL**	474 ± 3	$\mathbf{56 \pm 1}$
	PG(I_n)	$\underline{205 \pm 3}$	581 ± 37
	PG$(\mathrm{U}^{\star}_{\mathrm{he}})$	234 ± 1	281 ± 1
	PG$(\mathrm{U}^{\star}_{\mathrm{im}})$	$\mathbf{199 \pm 1}$	$\underline{216 \pm 2}$

Table 6.2: **MBRL experiment.** The PG loss with optimal reward comes <u>second</u> to the NLL baseline in terms of NLL, and ranks **first** in terms of MSE.

6.6 Conclusion

This chapter studied how to learn reward functions that, when plugged in policy gradient algorithms, induce models that are optimal in the maximum likelihood sense with respect to observed data. We introduced a bilevel optimization framework and derived closed-form solutions under specific assumptions on the reward parametrization and the data-generating process. We then demonstrated the practical relevance of the approach across several settings, including LLM fine-tuning, tabular classification, and dynamics modeling. In particular, for LLM fine-tuning, we showed that GRPO combined with rewards learned through our framework achieves test performance comparable to GRPO using the true task reward.

Limitations & Future Work. The reward parametrization considered in this study is intentionally simple, which may limit expressiveness in more complex tasks. An interesting direction for future work is to extend the framework to richer classes of parametric reward functions while preserving theoretical guarantees. In addition, although we evaluated the method on both synthetic data and real-world problems, larger-scale experiments would help further characterize its behavior. Notably, our LLM experiments focus on a verifiable task where an oracle reward is available for comparison. A natural extension is to investigate non-verifiable tasks, where such rewards are unavailable, and to compare SFT with RL driven by rewards learned within our framework. We view these directions as promising avenues for future exploration.

Chapter 7

Conclusions and Perspectives

This dissertation has investigated Model-based Reinforcement Learning, with a particular focus on its interaction with modern Foundation Models. By considering the supervised task of learning dynamics models that mimic Reinforcement Learning environments, we examined this problem from multiple complementary perspectives. Throughout the thesis, we introduced methodological contributions that enhance dynamics models and facilitate their seamless integration with modern Foundation Models, including Large Language Models and Time Series Foundation Models. Moreover, we complemented our empirical studies with theoretical analyses in simplified settings, demonstrating that insights derived from theory translated into practical improvements at scale. Finally, we released open-source implementations accompanying each chapter to ensure reproducibility and to foster adoption and subsequent research.

Summary of Contributions

Chapter 3. Setting the floor for the MBRL and FMs integration, we started by examining the problem of long-horizon prediction in learned dynamics models in Chapter 3. We revisit the objective mismatch paradox induced by standard one-step losses in MBRL, and establish that directly optimizing multi-step losses provides a principled way to improve long-horizon prediction quality. In a controlled linear setting, closed-form solutions are derived to characterize the bias–variance trade-offs induced by different weighting schemes, offering theoretical insight into when multi-step objectives are beneficial. Empirical results confirm that emphasizing multi-step accuracy substantially reduces long-term prediction error, while clarifying the nuanced relationship between predictive error and final return.

Chapter 4. This chapter reframes dynamics modeling as an in-context learning problem, thereby enabling the zero-shot integration of pre-trained LLMs into the MBRL pipeline. A latent projection mechanism is introduced to encode multidimensional state and action trajectories into linearly uncorrelated representations, resolving challenges raised by structured MBRL data and text-based sequence models. The resulting framework (DICL) demonstrates that large pre-trained sequence models can serve as general-purpose world models without task-specific training. We further provide a theoretical analysis characterizing the return gap induced by such framework. Empirically, we show improved sample efficiency in a data-augmented off-policy RL setup, together with calibrated uncertainty estimates using the LLM-based dynamics model.

Chapter 5. This chapter addresses the structural limitation of univariate TSFMs when applied to multivariate probabilistic forecasting, a setup closely related to MBRL data. It proposes a family of learnable encoder and decoder adapters that extend pre-trained univariate models to capture cross-dimensional dependencies while preserving their pre-trained representations. By incorporating stochasticity into the encoder, the framework further enables probabilistic forecasting for otherwise deterministic TSFMs. A theoretical analysis of linear adapters clarifies their expressiveness, showing that a linear adapter that improves the prediction error always exists. Finally, our experimental results demonstrated consistent gains in multivariate forecasting accuracy and uncertainty modeling compared to independent per-dimension application.

Chapter 6. This chapter broadens the scope of RL by introducing a framework for learning implicit reward functions from supervised data through bilevel optimization. Maximum likelihood estimation is reformulated as a bilevel problem in which reward parameters are optimized in the outer loop while the inner loop corresponds to a standard RL objective. A class of reward functions based on a negative Mahalanobis distance in an embedding space is analyzed theoretically in the Gaussian linear case, yielding closed-form solutions and geometric interpretations. Furthermore, we addressed more general settings through implicit differentiation-based solvers. Empirical validation shows that the learned rewards can substitute explicit verifiers in an LLM reasoning task, and can also support tabular regression and classification objectives, illustrating how reward learning can bridge supervised learning and RL within a unified framework.

Future work

Properties of the multi-step loss. Although we have demonstrated that the multi-step loss enhances long-horizon prediction, a more thorough analysis of its properties and

of optimal weighting strategies remains an open problem. In particular, the smoothness of the induced dynamics model and its consequences for planning algorithms constitute a compelling direction for future research. Specifically, a variant of the simulation lemma established by [AML18] shows that the return gap is upper bounded by a form of multi-step loss that can also be expressed in terms of the Lipschitz constant of the dynamics model. This observation indicates that multi-step losses may implicitly promote smoother dynamics models, potentially resulting in improved planning performance. Moreover, it has been shown that a probabilistic ensemble is unnecessary when a Lipschitz regularization is imposed on the learned value function [Zhe+23]. Together, these results further emphasize that regularity conditions of the learned dynamics model are central to the performance of MBRL algorithms, and that multi-step losses may provide a promising mechanism for enforcing such regularity.

Applying Time Series Foundation Models to MBRL data. A promising direction for future work consists in positioning dynamics modeling in MBRL as a systematic testbed for TSFMs. In this setting, state trajectories collected from benchmark environments (e.g., HalfCheetah) can be framed as forecasting problem, where past states serve as context and control signals (actions) are incorporated as covariates. In particular, both past and future actions could be provided as exogenous inputs, enabling a principled comparison between purely autoregressive forecasting and action-conditioned prediction. Such a study would evaluate zero-shot performance of recent TSFMs (e.g., Chronos2 [Ans+25]) on multi-step mean prediction error, while also assessing probabilistic metrics such as calibration for probabilistic TSFMs. Beyond benchmarking, this line of research would shed light on the specific challenges posed by MBRL data, namely multivariate structure, non-stationarity induced by evolving policies, and the presence of control covariates.

Extending reward models developed in Chapter 6 to non-verifiable tasks. The framework proposed in Chapter 6 makes it possible to learn reward functions from supervised data, thereby enabling the use of RL in scenarios where explicit reward design is impractical. Within an LLM fine-tuning setting, it was validated on a reasoning task for which a ground-truth reward function is available. This configuration was selected to allow a direct comparison between the learned reward and the explicit verifier. However, applying the framework to tasks lacking an explicit reward signal, such as machine translation or creative writing, would constitute a natural and important extension. In addition, the mitigation of catastrophic forgetting observed in the Mnist experiment (Figure 6.1) remains to be validated in an LLM fine-tuning context. In such a setting, catastrophic forgetting would be assessed by evaluating the fine-tuned models on general LLMs benchmarks and comparing their performance to that of the original pre-trained models. Demonstrating that the learned reward mitigates catastrophic forgetting in this

scenario would further highlight the potential of our contribution as a unifying framework bridging SFT and RL.

Concluding remarks

In summary, this dissertation has advanced Model-based Reinforcement Learning by systematically bridging it with modern Foundation Models, unifying perspectives from supervised learning, sequence modeling, forecasting, and bilevel optimization. Across chapters, we have shown that carefully revisiting objectives, representations, and optimization principles yields both theoretical insights and practical algorithms. By grounding empirical improvements in analytical insights, we aimed to demonstrate that principled modeling choices remain central to progress in ML research.

More broadly, this work positions dynamics and reward modeling as a fertile interface between Reinforcement Learning and Foundation Models, where ideas can flow bidirectionally: RL offers structured decision-making problems that put general-purpose models to the test, while Foundation Models provide expressive priors and scalable architectures that reshape classical RL pipelines. We hope that the methodological frameworks, analyses, and open-source implementations presented here contribute to a deeper integration of these fields and inspire further research toward more general, data-efficient, and reliable decision-making systems.

Bibliography

This bibliography contains 462 references.

[AAG24] Abdelrahman Abdelhamed, Mahmoud Afifi, and Alec Go. "What Do You See? Enhancing Zero-Shot Image Classification with Multimodal Large Language Models". In: *arXiv preprint arXiv:2405.15668* (2024).

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. *Wasserstein GAN*. 2017. arXiv: 1701.07875 [stat.ML].

[AD21] Arthur Argenson and Gabriel Dulac-Arnold. "Model-Based Offline Planning". In: *International Conference on Learning Representations*. 2021.

[AGF21] Romina Abachi, Mohammad Ghavamzadeh, and Amir-massoud Farahmand. *Policy-Aware Model Learning for Policy Gradient Methods*. 2021. eprint: 2003.00030.

[AGN05] Pieter Abbeel, Varun Ganapathi, and Andrew Ng. "Learning vehicular dynamics, with application to modeling helicopters". en. In: *Advances in Neural Information Processing Systems* 18 (2005).

[AJO08] Peter Auer, Thomas Jaksch, and Ronald Ortner. "Near-optimal Regret Bounds for Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. Ed. by D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou. Vol. 21. Curran Associates, Inc., 2008.

[Aka73] H Akaike. "Information Theory and an Extension of the Maximum Likelihood Principle". In: *2nd International Symposium on Information Theory, 1973*. Publishing House of the Hungarian Academy of Sciences. 1973, pp. 268–281.

[Aky+23] Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. *What learning algorithm is in-context learning? Investigations with linear models*. May 2023. DOI: 10.48550/arXiv.2211.15661.

[AM22] Michael Arbel and Julien Mairal. "Amortized Implicit Differentiation for Stochastic Bilevel Optimization". In: *International Conference on Learning Representations*. 2022.

[AML18] Kavosh Asadi, Dipendra Misra, and Michael Littman. "Lipschitz Continuity in Model-based Reinforcement Learning". In: *Proceedings of the 35th*

International Conference on Machine Learning. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 264–273.

[AN04] Pieter Abbeel and Andrew Y. Ng. "Apprenticeship learning via inverse reinforcement learning". In: *Proceedings of the Twenty-First International Conference on Machine Learning*. ICML '04. Banff, Alberta, Canada: Association for Computing Machinery, 2004, p. 1. ISBN: 1581138385. DOI: 10.1145/1015330.1015430.

[Ans+24] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. "Chronos: Learning the language of time series". In: *arXiv preprint arXiv:2403.07815* (2024).

[Ans+25] Abdul Fatir Ansari et al. *Chronos-2: From Univariate to Universal Forecasting*. 2025. eprint: 2510.15821.

[AOM17] Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. "Minimax Regret Bounds for Reinforcement Learning". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 263–272.

[Aue+25] Andreas Auer, Patrick Podest, Daniel Klotz, Sebastian Böck, Günter Klambauer, and Sepp Hochreiter. "TiRex: Zero-Shot Forecasting Across Long and Short Horizons with Enhanced In-Context Learning". In: *The Thirty-ninth Annual Conference on Neural Information Processing Systems*. 2025.

[aws25] aws. *What is a Foundation Model?* 2025.

[Bac24] Francis Bach. *Learning Theory from First Principles*. MIT Press, 2024.

[Bah+17] Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. *An Actor-Critic Algorithm for Sequence Prediction*. 2017. arXiv: 1607.07086 [cs.LG].

[Bai+25] Shuai Bai et al. *Qwen2.5-VL Technical Report*. 2025. arXiv: 2502.13923 [cs.CV].

[Ban22] Stefan Banach. "Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales". French. In: 3.1 (1922), pp. 133–181.

[Bao+20] Fan Bao, Chongxuan Li, Kun Xu, Hang Su, Jun Zhu, and Bo Zhang. "Bi-level score matching for learning energy-based latent variable models". In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS '20. Vancouver, BC, Canada: Curran Associates Inc., 2020. ISBN: 9781713829546.

[Bar+18] Gabriel Barth-Maron, Matthew W. Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva TB, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. "Distributional Policy Gradients". In: *International Conference on Learning Representations*. 2018.

[Bay63] T. Bayes. "An essay towards solving a problem in the doctrine of chances". In: 53 (1763), pp. 370–418.

[BB12] Souhaib Ben Taieb and Gianluca Bontempi. *Recursive Multi-step Time Series Forecasting by Perturbing Data*. en-US. Jan. 2012.

[BDM17] Marc G. Bellemare, Will Dabney, and Rémi Munos. "A Distributional Perspective on Reinforcement Learning". In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. PMLR, 2017, pp. 449–458.

[Bel+13] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. "The Arcade Learning Environment: An Evaluation Platform for General Agents". In: vol. 47. June 2013, pp. 253–279.

[Bel57a] Richard Bellman. "A Markovian decision process". In: 6.5 (1957), pp. 679–684.

[Bel57b] Richard Bellman. *Dynamic Programming*. 1st ed. Princeton University Press, 1957.

[Bel58] Richard Bellman. "Dynamic programming and stochastic control processes". In: *Information and Control* 1.3 (1958), pp. 228–239. ISSN: 0019-9958. DOI: 10.1016/S0019-9958(58)80003-0.

[Ben+06] K.P. Bennett, Jing Hu, Xiaoyun Ji, G. Kunapuli, and Jong-Shi Pang. "Model Selection via Bilevel Optimization". In: *The 2006 IEEE International Joint Conference on Neural Network Proceedings*. 2006, pp. 1922–1929. DOI: 10.1109/IJCNN.2006.246935.

[Ben+12] Souhaib Ben Taieb, Gianluca Bontempi, Amir F. Atiya, and Antti Sorjamaa. "A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition". In: *Expert Systems with Applications* 39.8 (June 2012), pp. 7067–7083. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2012.01.039.

[Ben+15a] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. *Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks*. Sept. 2015. DOI: 10.48550/arXiv.1506.03099.

[Ben+15b] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. *Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks*. 2015. arXiv: 1506.03099 [cs.LG].

[Ben+24] Abdelhakim Benechehab, Albert Thomas, Giuseppe Paolo, Maurizio Filippone, and Balázs Kégl. *A Multi-step Loss Function for Robust Learning of the Dynamics in Model-based Reinforcement Learning*. 2024. arXiv: 2402.03146 [cs.LG].

[Ben+25] Abdelhakim Benechehab, Youssef Attia El Hili, Ambroise Odonnat, Oussama Zekri, Albert Thomas, Giuseppe Paolo, Maurizio Filippone, Ievgen Redko, and Balázs Kégl. "Zero-shot Model-based Reinforcement Learning

using Large Language Models". In: *The Thirteenth International Conference on Learning Representations*. 2025.

[Ben00] Yoshua Bengio. "Gradient-Based Optimization of Hyperparameters". In: *Neural Computation* 12.8 (2000), pp. 1889–1900. DOI: 10.1162/089976600300015187.

[Bi+23] Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. "Accurate medium-range global weather forecasting with 3D neural networks". In: *Nature* 619.7970 (2023), pp. 533–538.

[Bis07] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. 1st ed. Springer, 2007. ISBN: 0387310738.

[Bla+24] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. "Training Diffusion Models with Reinforcement Learning". In: *The Twelfth International Conference on Learning Representations*. 2024.

[Bro+16] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. *OpenAI Gym*. 2016. eprint: arXiv:1606.01540.

[Bro+20] Tom B. Brown et al. *Language Models are Few-Shot Learners*. en. July 2020.

[BT02] Ronen I. Brafman and Moshe Tennenholtz. "R-MAX - A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning". In: 3.10 (2002), pp. 213–231. ISSN: ISSN 1533-7928.

[BT25] Maxime Burchi and Radu Timofte. "Learning Transformer-based World Models with Contrastive Predictive Coding". In: *The Thirteenth International Conference on Learning Representations*. 2025.

[BT52] Ralph Allan Bradley and Milton E. Terry. "Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons". In: *Biometrika* 39.3/4 (1952), pp. 324–345. ISSN: 00063444, 14643510.

[BTK24] Abdelhakim Benechehab, Albert Thomas, and Balázs Kégl. "Deep autoregressive density nets vs neural ensembles for model-based offline reinforcement learning". In: *arXiv preprint arXiv:2402.02858* (2024).

[Buc+18] Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. "Sample-efficient reinforcement learning with stochastic ensemble value expansion". In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS'18. Montréal, Canada: Curran Associates Inc., 2018, pp. 8234–8244.

[Byr+21] Arunkumar Byravan et al. "Evaluating model-based planning and planner amortization for continuous control". In: *CoRR* abs/2110.03363 (2021). arXiv: 2110.03363.

[Cao+24] Yuji Cao, Huan Zhao, Yuheng Cheng, Ting Shu, Yue Chen, Guolong Liu, Gaoqi Liang, Junhua Zhao, Jinyue Yan, and Yun Li. *Survey on Large Language Model-Enhanced Reinforcement Learning: Concept, Taxonomy, and Methods*. 2024. arXiv: 2404.00282 [cs.LG].

[Car+23] Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. *Grounding Large Language Models in Interactive Environments with Online Reinforcement Learning*. 2023. arXiv: 2302.02662 [cs.LG].

[Car+26] Nicolas Carion et al. "SAM 3: Segment Anything with Concepts". In: *The Fourteenth International Conference on Learning Representations*. 2026.

[CB99] Eduardo F. Camacho and Carlos Bordons, eds. *Model predictive control*. Springer-Verlag, 1999.

[CFG14] Tianqi Chen, Emily Fox, and Carlos Guestrin. "Stochastic Gradient Hamiltonian Monte Carlo". In: *Proceedings of the 31st International Conference on Machine Learning*. Ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Research 2. Bejing, China: PMLR, Sum. 2014, pp. 1683–1691.

[CGG21] Rohitash Chandra, Shaurya Goyal, and Rishabh Gupta. "Evaluation of deep learning models for multi-step ahead time series prediction". In: *IEEE Access* 9 (2021), pp. 83105–83123. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3085085.

[CH23] Nicholas E Corrado and Josiah P Hanna. "Understanding when Dynamics-Invariant Data Augmentations Benefit Model-Free Reinforcement Learning Updates". In: *arXiv preprint arXiv:2310.17786* (2023).

[Cha+08a] Ming-Wei Chang, Lev Ratinov, Dan Roth, and Vivek Srikumar. "Importance of semantic representation: dataless classification". In: *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*. AAAI'08. Chicago, Illinois: AAAI Press, 2008, pp. 830–835. ISBN: 9781577353683.

[Cha+08b] Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck. "Monte-carlo tree search: a new framework for game AI". In: *Proceedings of the Fourth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. AIIDE'08. Stanford, California: AAAI Press, 2008, pp. 216–217.

[Cha+24] Souradip Chakraborty, Amrit Bedi, Alec Koppel, Huazheng Wang, Dinesh Manocha, Mengdi Wang, and Furong Huang. "PARL: A Unified Framework for Policy Alignment in Reinforcement Learning from Human Feedback". In: *The Twelfth International Conference on Learning Representations*. 2024.

[Che+21] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. *Decision Transformer: Reinforcement Learning via Sequence Modeling*. 2021. arXiv: 2106.01345 [cs.LG].

[Che+22] Chang Chen, Yi-Fu Wu, Jaesik Yoon, and Sungjin Ahn. *TransDreamer: Reinforcement Learning with Transformer World Models*. 2022. arXiv: 2202.09481 [cs.LG].

[Che+23] Si-An Chen, Chun-Liang Li, Sercan O Arik, Nathanael Christian Yoder, and Tomas Pfister. "TSMixer: An All-MLP Architecture for Time Series Forecast-ing". In: *Transactions on Machine Learning Research* (2023). ISSN: 2835-8856.

[Che+24] Chang Chen, Yi-Fu Wu, Jaesik Yoon, and Sungjin Ahn. *TransDreamer: Reinforcement Learning with Transformer World Models*. 2024. eprint: 2202.09481.

[Chr+17] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. "Deep Reinforcement Learning from Human Preferences". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017.

[Chu+15] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron Courville, and Yoshua Bengio. "A recurrent latent variable model for sequential data". In: *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'15. Montreal, Canada: MIT Press, 2015, pp. 2980–2988.

[Chu+18] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. "Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models". In: *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc., 2018, pp. 4754–4765.

[CL18] David Corne and Michael A. Lones. "Evolutionary Algorithms". In: *Handbook of Heuristics*. Springer International Publishing, 2018, pp. 1–22. ISBN: 9783319071534. DOI: 10.1007/978-3-319-07153-4_27-1.

[Cod+23] Julian Coda-Forno, Marcel Binz, Zeynep Akata, Matthew Botvinick, Jane X. Wang, and Eric Schulz. *Meta-in-context learning in large language models*. May 2023. DOI: 10.48550/arXiv.2305.12907.

[Coh+24] Ben Cohen, Emaad Khwaja, Kan Wang, Charles Masson, Elise Ramé, Youssef Doubli, and Othmane Abou-Amal. *Toto: Time Series Optimized Transformer for Observability*. 2024. eprint: 2407.07874.

[Cow+22] Alexander Cowen-Rivers et al. "HEBO: Pushing The Limits of Sample-Efficient Hyperparameter Optimisation". In: *Journal of Artificial Intelligence Research* 74 (July 2022).

[CSY21] Tianyi Chen, Yuejiao Sun, and Wotao Yin. "Closing the Gap: Tighter Analysis of Alternating Stochastic Gradient Methods for Bilevel Problems". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., 2021, pp. 25294–25307.

[CZY24] Shijie Chen, Yu Zhang, and Qiang Yang. "Multi-Task Learning in Natural Language Processing: An Overview". In: 56.12 (July 2024). ISSN: 0360-0300. DOI: 10.1145/3663363.

[Dag+24] Mathieu Dagréou, Pierre Ablin, Samuel Vaiter, and Thomas Moreau. *A framework for bilevel optimization that enables stochastic and global variance reduction algorithms*. 2024. arXiv: 2201.13409 [stat.ML].

[Dai+24] Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. "Safe RLHF: Safe Reinforcement Learning from Human Feedback". In: *The Twelfth International Conference on Learning Representations*. 2024.

[Dao23] Tri Dao. *FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning*. 2023. eprint: 2307.08691.

[Das+24] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. "A decoder-only foundation model for time-series forecasting". In: *Forty-first International Conference on Machine Learning*. 2024.

[de +05] Pieter-Tjerk {de Boer}, {Dirk P.} Kroese, Shie Mannor, and {Reuven Y.} Rubinstein. "A Tutorial on the Cross-Entropy Method". English. In: 134.1 (Jan. 2005), pp. 19–67. ISSN: 0254-5330. DOI: 10.1007/s10479-005-5724-z.

[Dee+25] DeepSeek-AI et al. *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. 2025. arXiv: 2501.12948 [cs.CL].

[Déf+24] Alexandre Défossez, Laurent Mazaré, Manu Orsini, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. *Moshi: a speech-text foundation model for real-time dialogue*. 2024. eprint: 2410.00037.

[Dep+17] Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. "Learning and Policy Search in Stochastic Dynamical Systems with Bayesian Neural Networks". In: *International Conference on Learning Representations*. 2017.

[Dep+18] Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. "Decomposition of Uncertainty in Bayesian Deep Learning for Efficient and Risk-sensitive Learning". In: *ICML*. 2018.

[DER95] Andreas Draeger, Sebastian Engell, and Horst Ranke. "Model Predictive Control Using Neural Networks". In: *IEEE Control Systems* 15 (5 1995), pp. 61–66. ISSN: 1066033X. DOI: 10.1109/37.466261.

[Dev+19a] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *North American Chapter of the Association for Computational Linguistics*. 2019.

[Dev+19b] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. eprint: 1810.04805.

[Die00] Thomas G. Dietterich. "Ensemble Methods in Machine Learning". In: *Proceedings of the First International Workshop on Multiple Classifier Systems*. MCS '00. Springer-Verlag, 2000, pp. 1–15. ISBN: 3540677046.

[DJA] Fei Deng, Ingook Jang, and Sungjin Ahn. "DreamerPro: Reconstruction-Free Model-Based Reinforcement Learning with Prototypical Representations". In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato. Vol. 162. Proceedings of Machine Learning Research. PMLR, pp. 4956–4975.

[DL24] Abhimanyu Dubey and the Llama 3 team. *The Llama 3 Herd of Models*. 2024. arXiv: 2407.21783 [cs.AI].

[Dom12] Justin Domke. "Generic Methods for Optimization-Based Modeling". In: *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Neil D. Lawrence and Mark Girolami. Vol. 22. Proceedings of Machine Learning Research. La Palma, Canary Islands: PMLR, Spr. 2012, pp. 318–326.

[DPA23] Fei Deng, Junyeong Park, and Sungjin Ahn. "Facing Off World Model Backbones: RNNs, Transformers, and S4". In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine. Vol. 36. Curran Associates, Inc., 2023, pp. 72904–72930.

[DR11] Marc Peter Deisenroth and Carl Edward Rasmussen. "PILCO: A Model-Based and Data-Efficient Approach to Policy Search". In: *Proceedings of the International Conference on Machine Learning*. 2011.

[DS17] Nan Ding and Radu Soricut. "Cold-Start Reinforcement Learning with Softmax Policy Gradient". In: *Neural Information Processing Systems*. 2017.

[DSB17] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. *Density estimation using Real NVP*. 2017. arXiv: 1605.08803 [cs.LG].

[DWS12] Thomas Degris, Martha White, and Richard S. Sutton. "Off-policy actor-critic". In: *Proceedings of the 29th International Coference on International Conference on Machine Learning*. ICML'12. Edinburgh, Scotland: Omnipress, 2012, pp. 179–186. ISBN: 9781450312851.

[Eka+24] Vijay Ekambaram, Arindam Jati, Pankaj Dayama, Sumanta Mukherjee, Nam H. Nguyen, Wesley M. Gifford, Chandra Reddy, and Jayant Kalagnanam. *Tiny Time Mixers (TTMs): Fast Pre-trained Models for Enhanced Zero/Few-Shot Forecasting of Multivariate Time Series*. 2024. arXiv: 2401.03955 [cs.LG].

[Fan+22] Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. "MineDojo: Building Open-Ended Embodied Agents with Internet-Scale Knowledge". In: *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. 2022.

[FBN17] Amir-Massoud Farahmand, Andre Barreto, and Daniel Nikovski. "Value-Aware Loss Function for Model-based Reinforcement Learning". In: *Pro-*

ceedings of the 20th International Conference on Artificial Intelligence and Statistics. Ed. by Aarti Singh and Jerry Zhu. Vol. 54. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1486–1494.

[Fei+18] Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I. Jordan, Joseph E. Gonzalez, and Sergey Levine. *Model-Based Value Estimation for Efficient Model-Free Reinforcement Learning*. 2018. arXiv: 1803.00101 [cs.LG].

[Fen+21] Xidong Feng, Oliver Slumbers, Ziyu Wan, Bo Liu, Stephen McAleer, Ying Wen, Jun Wang, and Yaodong Yang. *Neural Auto-Curricula*. 2021. arXiv: 2106.02745 [cs.AI].

[Fen+24] Xueyang Feng, Zhi-Yuan Chen, Yujia Qin, Yankai Lin, Xu Chen, Zhiyuan Liu, and Ji-Rong Wen. *Large Language Model-based Human-Agent Collaboration for Complex Task Solving*. Feb. 2024.

[Feo+24] Vasilii Feofanov, Romain Ilbert, Malik Tiomoko, Themis Palpanas, and Ievgen Redko. "User-friendly Foundation Model Adapters for Multivariate Time Series Classification". In: *arXiv preprint arXiv:2409.12264* (2024).

[Feo+25] Vasilii Feofanov, Songkang Wen, Marius Alonso, Romain Ilbert, Hongbo Guo, Malik Tiomoko, Lujia Pan, Jianfeng Zhang, and Ievgen Redko. "Mantis: Lightweight calibrated foundation model for user-friendly time series classification". In: *arXiv preprint arXiv:2502.15637* (2025).

[FHM18] Scott Fujimoto, Herke van Hoof, and David Meger. "Addressing Function Approximation Error in Actor-Critic Methods". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 1587–1596.

[Fin+16] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. *A Connection between Generative Adversarial Networks, Inverse Reinforcement Learning, and Energy-Based Models*. 2016. arXiv: 1611.03852 [cs.LG].

[FLA16] Chelsea Finn, Sergey Levine, and Pieter Abbeel. "Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization". In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 2016, pp. 49–58.

[FMP19] Scott Fujimoto, David Meger, and Doina Precup. "Off-Policy Deep Reinforcement Learning without Exploration". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 2052–2062.

[FR98] Klaus Fraedrich and Bernd Rückert. "Metric adaption for analog forecasting". In: *Physica A: Statistical Mechanics and its Applications* 253.1 (1998), pp. 379–393. ISSN: 0378-4371. DOI: 10.1016/S0378-4371(97)00668-7.

[Fra+17] Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil. "Forward and Reverse Gradient-Based Hyperparameter Optimization". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 1165–1173.

[Fra+18] Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. "Bilevel Programming for Hyperparameter Optimization and Meta-Learning". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 1568–1577.

[Fra+24] Bernd Frauenknecht, Artur Eisele, Devdutt Subhasish, Friedrich Solowjow, and Sebastian Trimpe. *Trust the Model Where It Trusts Itself – Model-Based Actor-Critic with Uncertainty-Aware Rollout Adaption*. 2024. arXiv: 2405.19014 [cs.LG].

[Fu+21a] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. *D4RL: Datasets for Deep Data-Driven Reinforcement Learning*. 2021.

[Fu+21b] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. *D4RL: Datasets for Deep Data-Driven Reinforcement Learning*. 2021. arXiv: 2004.07219 [cs.LG].

[Fu+25] Chaoyou Fu et al. *VITA-1.5: Towards GPT-4o Level Real-Time Vision and Speech Interaction*. 2025. arXiv: 2501.01957 [cs.CV].

[Gag94] Philip Gage. "A new algorithm for data compression". In: 12 (1994), pp. 23–38.

[Gar+23] Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. *What Can Transformers Learn In-Context? A Case Study of Simple Function Classes*. Aug. 2023. DOI: 10.48550/arXiv.2208.01066.

[Gar+24] Quentin Garrido, Mahmoud Assran, Nicolas Ballas, Adrien Bardes, Laurent Najman, and Yann LeCun. "Learning and Leveraging World Models in Visual Representation Learning". In: abs/2403.00504 (2024).

[Gau+25] Mudit Gaur, Utsav Singh, Amrit Singh Bedi, Raghu Pasupathu, and Vaneet Aggarwal. *On The Sample Complexity Bounds In Bilevel Reinforcement Learning*. 2025. arXiv: 2503.17644 [cs.LG].

[GB13] Hector Geffner and Blai Bonet. *A Concise Introduction to Models and Methods for Automated Planning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2013. ISBN: 9781608459698. DOI: 10.2200/S00513ED1V01Y201306AIM022.

[GBB01] Evan Greensmith, Peter Bartlett, and Jonathan Baxter. "Variance Reduction Techniques for Gradient Estimates in Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. Ed. by T. Dietterich, S. Becker, and Z. Ghahramani. Vol. 14. MIT Press, 2001.

[GBB04] Evan Greensmith, Peter L. Bartlett, and Jonathan Baxter. "Variance Reduction Techniques for Gradient Estimates in Reinforcement Learning". In: 5 (2004), pp. 1471–1530.

[GBR] Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E Raftery. "Probabilistic forecasts, calibration and sharpness". en. In: ().

[Gel+13] Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. *Bayesian Data Analysis*. English. 3rd ed. United States: Chapman and Hall/CRC, 2013. ISBN: 9781439840955.

[GG16a] Yarin Gal and Zoubin Ghahramani. "Dropout As a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. ICML'16. New York, NY, USA: JMLR.org, 2016, pp. 1050–1059.

[GG16b] Yarin Gal and Zoubin Ghahramani. "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. PMLR, 2016, pp. 1050–1059.

[Gia+24] Panagiotis Giadikiaroglou, Maria Lymperaiou, Giorgos Filandrianos, and Giorgos Stamou. "Puzzle Solving using Reasoning of Large Language Models: A Survey". In: *arXiv preprint arXiv:2402.11291* (2024).

[GK14] Tilmann Gneiting and Matthias Katzfuss. "Probabilistic forecasting". In: *Annual Review of Statistics and Its Application* 1.1 (2014), pp. 125–151.

[GMR16a] Yarin Gal, Rowan McAllister, and Carl Edward Rasmussen. "Improving PILCO with Bayesian neural network dynamics models". In: *Data-Efficient Machine Learning workshop, International Conference on Machine Learning*. 2016.

[GMR16b] Yarin Gal, Rowan McAllister, and Carl Edward Rasmussen. "Improving PILCO with Bayesian neural network dynamics models". In: *Data-Efficient Machine Learning workshop, International Conference on Machine Learning*. 2016.

[Goo+14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[Gos+24] Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. "MOMENT: A Family of Open Time-series Foundation Models". In: *International Conference on Machine Learning*. 2024.

[GPR13] Jan JJ Groen, Richard Paap, and Francesco Ravazzolo. "Real-time inflation forecasting in a changing world". In: *Journal of Business & Economic Statistics* 31.1 (2013), pp. 29–44.

[Gra11] Alex Graves. "Practical Variational Inference for Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by J. Shawe-Taylor, R.

Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger. Vol. 24. Curran Associates, Inc., 2011.

[Gri+26] Léo Grinsztajn et al. *TabPFN-2.5: Advancing the State of the Art in Tabular Foundation Models*. 2026. eprint: 2511.08667.

[Gru+23a] Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Gordon Wilson. *Large Language Models Are Zero-Shot Time Series Forecasters*. Oct. 2023.

[Gru+23b] Nate Gruver, Marc Anton Finzi, Shikai Qiu, and Andrew Gordon Wilson. "Large Language Models Are Zero-Shot Time Series Forecasters". In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023.

[Gui+24] Jie Gui, Tuo Chen, Jing Zhang, Qiong Cao, Zhenan Sun, Hao Luo, and Dacheng Tao. *A Survey on Self-supervised Learning: Algorithms, Applications, and Future Trends*. 2024. eprint: 2301.05712.

[Guo+17a] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. "On Calibration of Modern Neural Networks". en. In: *Proceedings of the 34th International Conference on Machine Learning*. PMLR, July 2017, pp. 1321–1330.

[Guo+17b] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. "On calibration of modern neural networks". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 1321–1330.

[Haa+17] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. "Reinforcement Learning with Deep Energy-Based Policies". In: (2017).

[Haa+18a] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. *Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor*. 2018. arXiv: 1801.01290 [cs.LG].

[Haa+18b] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 1861–1870.

[Haf+19a] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. "Learning Latent Dynamics for Planning from Pixels". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 2555–2565.

[Haf+19b] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. "Learning Latent Dynamics for Planning from Pixels". In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. Proceedings of Machine Learning Research. 2019, pp. 2555–2565.

[Haf+20] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. "Dream to Control: Learning Behaviors by Latent Imagination". In: *International Conference on Learning Representations*. 2020.

[Haf+21a] Danijar Hafner, Timothy P Lillicrap, Mohammad Norouzi, and Jimmy Ba. "Mastering Atari with Discrete World Models". In: *International Conference on Learning Representations*. 2021.

[Haf+21b] Danijar Hafner, Timothy P Lillicrap, Mohammad Norouzi, and Jimmy Ba. "Mastering Atari with Discrete World Models". In: *International Conference on Learning Representations*. 2021.

[Haf+25] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. "Mastering diverse control tasks through world models". In: (2025), pp. 1–7.

[Har+20] Kristian Hartikainen, Xinyang Geng, Tuomas Haarnoja, and Sergey Levine. "Dynamical Distance Learning for Semi-Supervised and Unsupervised Skill Discovery". In: *International Conference on Learning Representations*. 2020.

[Has70] W. K. Hastings. "Monte Carlo sampling methods using Markov chains and their applications". In: 57.1 (1970), pp. 97–109. DOI: 10.1093/biomet/57.1.97. eprint: http://biomet.oxfordjournals.org/cgi/reprint/57/1/97.pdf.

[Hee+15] Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. "Learning Continuous Control Policies by Stochastic Value Gradients". In: *Advances in Neural Information Processing Systems*. Vol. 28. 2015, pp. 2944–2952.

[Heg+23] Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. *TabLLM: Few-shot Classification of Tabular Data with Large Language Models*. Mar. 2023. DOI: 10.48550/arXiv.2210.10723.

[Hen+18] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. "Deep reinforcement learning that matters". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*. New Orleans, Louisiana, USA: AAAI Press, 2018.

[Hes+18] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. "Rainbow: combining improvements in deep reinforcement learning". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*. New Orleans, Louisiana, USA: AAAI Press, 2018.

[Hes+23] Florian Hess, Zahra Monfared, Manuel Brenner, and Daniel Durstewitz. *Generalized Teacher Forcing for Learning Chaotic Dynamics*. Oct. 2023.

[HGS16] Hado van Hasselt, Arthur Guez, and David Silver. "Deep Reinforcement Learning with Double Q-Learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 30. 1. 2016.

[Hig+17] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In: *International Conference on Learning Representations*. 2017.

[HL23] Louis Martin Hugo Touvron and the Llama 2 team. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 2023. arXiv: 2307.09288 [cs.CL].

[Hon+22] Mingyi Hong, Hoi-To Wai, Zhaoran Wang, and Zhuoran Yang. *A Two-Timescale Framework for Bilevel Optimization: Complexity Analysis and Application to Actor-Critic*. 2022. arXiv: 2007.05170 [math.OC].

[HOP18] Yi Hao, Alon Orlitsky, and Venkatadheeraj Pichapati. "On learning Markov chains". In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS'18. Curran Associates Inc., 2018, pp. 646–655.

[Hou+19] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. "Parameter-Efficient Transfer Learning for NLP". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 2790–2799.

[How60] R. A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, 1960.

[HS18a] David Ha and Jürgen Schmidhuber. "Recurrent World Models Facilitate Policy Evolution". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 2450–2462.

[HS18b] David Ha and Jürgen Schmidhuber. "Recurrent World Models Facilitate Policy Evolution". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018.

[HSW24] Nicklas Hansen, Hao Su, and Xiaolong Wang. "TD-MPC2: Scalable, Robust World Models for Continuous Control". In: *International Conference on Learning Representations (ICLR)*. 2024.

[Hu+21] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. *LoRA: Low-Rank Adaptation of Large Language Models*. 2021. eprint: 2106.09685.

[Hu+22] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. "LoRA: Low-Rank Adaptation of Large Language Models". In: *International Conference on Learning Representations*. 2022.

[Hua+22a] Shengyi Huang, Rousslan Fernand Julien Dossa, Antonin Raffin, Anssi Kanervisto, and Weixun Wang. "The 37 Implementation Details of Proximal Policy Optimization". In: *ICLR Blog Track*. 2022.

[Hua+22b] Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. "CleanRL: High-quality Single-file Implementations of Deep Reinforcement Learning Algorithms". In: *Journal of Machine Learning Research* 23.274 (2022), pp. 1–18.

[Hua+24] Weidong Huang, Jiaming Ji, Chunhe Xia, Borong Zhang, and Yaodong Yang. "SafeDreamer: Safe Reinforcement Learning with World Models". In: *The Twelfth International Conference on Learning Representations*. 2024.

[Hus15] Ferenc Huszár. *How (not) to Train your Generative Model: Scheduled Sampling, Likelihood, Adversary?* Nov. 2015. DOI: 10.48550/arXiv.1511.05101.

[HVD15] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. "Distilling the Knowledge in a Neural Network". In: abs/1503.02531 (2015). arXiv: 1503.02531.

[HWS22] Nicklas Hansen, Xiaolong Wang, and Hao Su. "Temporal Difference Learning for Model Predictive Control". In: *International Conference on Machine Learning (ICML)*. 2022.

[HYL25] Danijar Hafner, Wilson Yan, and Timothy Lillicrap. *Training Agents Inside of Scalable World Models*. 2025. eprint: 2509.24527.

[Ilb+24] Romain Ilbert, Ambroise Odonnat, Vasilii Feofanov, Aladin Virmaux, Giuseppe Paolo, Themis Palpanas, and Ievgen Redko. "SAMformer: Unlocking the Potential of Transformers in Time Series Forecasting with Sharpness-Aware Minimization and Channel-Wise Attention". In: *Proceedings of the 41st International Conference on Machine Learning*. Vol. 235. PMLR, 2024.

[Int+25a] Physical Intelligence et al. $\pi_{0.5}$*: a Vision-Language-Action Model with Open-World Generalization*. 2025. eprint: 2504.16054.

[Int+25b] Physical Intelligence et al. $\pi_{0.5}$*: a Vision-Language-Action Model with Open-World Generalization*. 2025. arXiv: 2504.16054 [cs.LG].

[IS15] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France: JMLR.org, 2015, pp. 448–456.

[Jan+19] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. "When to Trust Your Model: Model-Based Policy Optimization". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelz-

imer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019.

[Jin+24a] Ming Jin et al. *Time-LLM: Time Series Forecasting by Reprogramming Large Language Models*. 2024. arXiv: 2310.01728 [cs.LG].

[Jin+24b] Ming Jin et al. "Time-LLM: Time series forecasting by reprogramming large language models". In: *International Conference on Learning Representations (ICLR)*. 2024.

[JLL21] Michael Janner, Qiyang Li, and Sergey Levine. *Offline Reinforcement Learning as One Big Sequence Modeling Problem*. Nov. 2021. DOI: 10.48550/arXiv.2106.02039.

[JS12] Hayley E Jones and David J Spiegelhalter. "Improved probabilistic prediction of healthcare performance indicators using bidirectional smoothing models". In: *Journal of the Royal Statistical Society Series A: Statistics in Society* 175.3 (2012), pp. 729–747.

[JS96] Mark Jerrum and Alistair Sinclair. "The Markov chain Monte Carlo method: an approach to approximate counting and integration". In: *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Co., 1996, pp. 482–520. ISBN: 0534949681.

[JYL21] Kaiyi Ji, Junjie Yang, and Yingbin Liang. *Bilevel Optimization: Convergence Analysis and Enhanced Design*. 2021. arXiv: 2010.07962 [cs.LG].

[Kan+22] Ge Kan, Jinhu Lü, Tian Wang, Baochang Zhang, Aichun Zhu, Lei Huang, Guodong Guo, and Hichem Snoussi. *Bi-level Doubly Variational Learning for Energy-based Latent Variable Models*. 2022. arXiv: 2203.14702 [cs.CV].

[KB15] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015.

[KB17] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[KCG12] Emilie Kaufmann, Olivier Cappe, and Aurelien Garivier. "On Bayesian Upper Confidence Bounds for Bandit Problems". In: *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Neil D. Lawrence and Mark Girolami. Vol. 22. Proceedings of Machine Learning Research. PMLR, Spr. 2012, pp. 592–600.

[Kég+18] Balázs Kégl, Alexandre Boucaud, Mehdi Cherti, Akin Kazakci, Alexandre Gramfort, Guillaume Lemaitre, Joris Van den Bossche, Djalel Benbouzid, and Camille Marini. "The RAMP framework: from reproducibility to transparency in the design and optimization of scientific workflows". In: *ICML workshop on Reproducibility in Machine Learning*. 2018.

[KFE18a] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. "Accurate Uncertainties for Deep Learning Using Calibrated Regression". en. In: *Proceedings of the 35th International Conference on Machine Learning*. PMLR, July 2018, pp. 2796–2804.

[KFE18b] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. "Accurate Uncertainties for Deep Learning Using Calibrated Regression". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 2796–2804.

[KFF17] Meelis Kull, Telmo Silva Filho, and Peter Flach. "Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers". en. In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. PMLR, Apr. 2017, pp. 623–631.

[KG17] Alex Kendall and Yarin Gal. "What uncertainties do we need in Bayesian deep learning for computer vision?" In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Long Beach, California, USA, 2017, pp. 5580–5590.

[KHT21] Balázs Kégl, Gabriel Hurtado, and Albert Thomas. "Model-based micro-data reinforcement learning: what are the crucial model properties and which model to choose?" In: *International Conference on Learning Representations*. 2021.

[Kid+20] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. "MOReL: Model-Based Offline Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 21810–21823.

[Kim+22] Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. "Reversible Instance Normalization for Accurate Time-Series Forecasting against Distribution Shift". In: *International Conference on Learning Representations*. 2022.

[Kim+23] Changyeon Kim, Jongjin Park, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. "Preference Transformer: Modeling Human Preferences using Transformers for RL". In: *The Eleventh International Conference on Learning Representations*. 2023.

[Kim+24] Moo Jin Kim et al. *OpenVLA: An Open-Source Vision-Language-Action Model*. 2024. arXiv: 2406.09246 [cs.RO].

[KKL03] Sham Kakade, Michael Kearns, and John Langford. "Exploration in Metric State Spaces". en. In: (2003).

[KL02] Sham Kakade and John Langford. "Approximately Optimal Approximate Reinforcement Learning". In: *Proceedings of the Nineteenth International Con-*

ference on Machine Learning. ICML '02. Morgan Kaufmann Publishers Inc., 2002, pp. 267–274. ISBN: 1558608737.

[KLC95] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. "Partially Observable Markov Decision Processes for Artificial Intelligence". In: *Proceedings of the International Workshop on Reasoning with Uncertainty in Robotics*. RUR '95. Springer-Verlag, 1995, pp. 146–163. ISBN: 3540613765.

[KM19] Hyunjik Kim and Andriy Mnih. *Disentangling by Factorising*. 2019. arXiv: 1802.05983 [stat.ML].

[KM53] Herman Kahn and Andy W. Marshall. "Methods of reducing sample size in Monte Carlo computations". In: 1.5 (1953), pp. 263–278.

[KPB21] Ivan Kobyzev, Simon J.D. Prince, and Marcus A. Brubaker. "Normalizing Flows: An Introduction and Review of Current Methods". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.11 (Nov. 2021), pp. 3964–3979. ISSN: 1939-3539. DOI: 10.1109/tpami.2020.2992934.

[KR18] Taku Kudo and John Richardson. "SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing". In: *Conference on Empirical Methods in Natural Language Processing*. 2018.

[KS02] Michael Kearns and Satinder Singh. "Near-Optimal Reinforcement Learning in Polynomial Time". en. In: 49.2 (2002), pp. 209–232. ISSN: 1573-0565. DOI: 10.1023/A:1017984413808.

[Kur+18] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. "Model-Ensemble Trust-Region Policy Optimization". In: *International Conference on Learning Representations*. 2018.

[KVM24] Shyam Sundar Kannan, Vishnunandan L. N. Venkatesh, and Byung-Cheol Min. *SMART-LLM: Smart Multi-Agent Robot Task Planning using Large Language Models*. Mar. 2024. DOI: 10.48550/arXiv.2309.10062.

[KW13] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *CoRR* abs/1312.6114 (2013).

[KW14] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. 2014.

[Kwo+23] Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. *Reward Design with Language Models*. 2023. arXiv: 2303.00001 [cs.LG].

[Lai+18] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. *Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks*. 2018. arXiv: 1703.07015 [cs.LG].

[Lai+20] Hang Lai, Jian Shen, Weinan Zhang, and Yong Yu. "Bidirectional model-based policy optimization". In: *Proceedings of the 37th International Conference on Machine Learning*. ICML'20. JMLR.org, 2020.

[Lai+25] Song Lai et al. *Reinforcement Fine-Tuning Naturally Mitigates Forgetting in Continual Post-Training*. 2025. arXiv: 2507.05386 [cs.LG].

[Łaj+24] Mateusz Łajszczak et al. *BASE TTS: Lessons from building a billion-parameter Text-to-Speech model on 100K hours of data*. 2024. arXiv: 2402.08093 [cs.LG].

[Lam+16] Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. "Professor Forcing: A New Algorithm for Training Recurrent Networks". In: *Advances in Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc., 2016.

[Lam+21a] Nathan Lambert, Brandon Amos, Omry Yadan, and Roberto Calandra. *Objective Mismatch in Model-based Reinforcement Learning*. 2021. arXiv: 2002.04523 [cs.LG].

[Lam+21b] Nathan Lambert, Albert Wilcox, Howard Zhang, Kristofer S. J. Pister, and Roberto Calandra. "Learning Accurate Long-term Dynamics for Model-based Reinforcement Learning". In: *2021 60th IEEE Conference on Decision and Control (CDC)*. Dec. 2021, pp. 2880–2887. DOI: 10.1109/CDC45484.2021.9683134.

[LaV98] Steven M. LaValle. "Rapidly-exploring random trees : a new tool for path planning". In: (1998).

[LCZ21] Ruizhen Liu, Zhicong Chen, and Dazhi Zhong. "DROMO: Distributionally Robust Offline Model-based Policy Optimization". In: (2021).

[LEB08] Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. "Zero-data learning of new tasks". In: *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*. AAAI'08. Chicago, Illinois: AAAI Press, 2008, pp. 646–651. ISBN: 9781577353683.

[Lee+20] Kimin Lee, Younggyo Seo, Seunghyun Lee, Honglak Lee, and Jinwoo Shin. "Context-aware dynamics model for generalization in model-based reinforcement learning". In: *Proceedings of the 37th International Conference on Machine Learning*. ICML'20. JMLR.org, 2020.

[Lev+16] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. "End-to-End Training of Deep Visuomotor Policies". In: 17.39 (2016), pp. 1–40.

[Lev+20a] Dan Levi, Liran Gispan, Niv Giladi, and Ethan Fetaya. *Evaluating and Calibrating Uncertainty Prediction in Regression Tasks*. Feb. 2020. DOI: 10.48550/arXiv.1905.11659.

[Lev+20b] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. "Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems". In: *arXiv preprint arXiv:2005.01643* (2020). arXiv: 2005.01643 [cs.LG].

[Li+23] Yingcong Li, Muhammed Emrullah Ildiz, Dimitris Papailiopoulos, and Samet Oymak. "Transformers as Algorithms: Generalization and Stability in In-context Learning". In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause, Emma Brunskill, Kyunghyun

Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett. Vol. 202. Proceedings of Machine Learning Research. PMLR, Aut. 2023, pp. 19565–19594.

[Li+24] Jiaxiang Li, Siliang Zeng, Hoi To Wai, Chenliang Li, Alfredo Garcia, and Mingyi Hong. "Getting More Juice Out of the SFT Data: Reward Learning from Human Demonstration Improves SFT for LLM Alignment". In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. 2024.

[Li+25] Youwei Li, Junyong Ye, Xubin Wen, Guangyi Xu, Jingjing Wang, and Xinyuan Liu. "PAdapter: Adapter combined with prompt for image and video classification". In: *Image and Vision Computing* 154 (2025), p. 105395. ISSN: 0262-8856. DOI: 10.1016/j.imavis.2024.105395.

[Lia+18] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. "Tune: A Research Platform for Distributed Model Selection and Training". In: *arXiv preprint arXiv:1807.05118* (2018).

[Lia+23] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. *Code as Policies: Language Model Programs for Embodied Control*. May 2023.

[Lil+16] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. "Continuous control with deep reinforcement learning". In: *ICLR (Poster)*. 2016.

[Lin+24] Jessy Lin, Yuqing Du, Olivia Watkins, Danijar Hafner, Pieter Abbeel, Dan Klein, and Anca Dragan. *Learning to Model the World with Language*. May 2024. DOI: 10.48550/arXiv.2308.01399.

[Lin76] Seppo Linnainmaa. "Taylor expansion of the accumulated rounding error". In: *BIT* 16.2 (June 1976), pp. 146–160. ISSN: 0006-3835. DOI: 10.1007/BF01931367.

[Liu+20] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. *Ro{BERT}a: A Robustly Optimized {BERT} Pretraining Approach*. 2020.

[Liu+23] Zuxin Liu, Jesse Zhang, Kavosh Asadi, Yao Liu, Ding Zhao, Shoham Sabach, and Rasool Fakoor. *TAIL: Task-specific Adapters for Imitation Learning with Large Pretrained Models*. Oct. 2023. DOI: 10.48550/arXiv.2310.05905.

[Liu+24a] Shaoteng Liu, Haoqi Yuan, Minda Hu, Yanwei Li, Yukang Chen, Shu Liu, Zongqing Lu, and Jiaya Jia. *RL-GPT: Integrating Reinforcement Learning and Code-as-policy*. Feb. 2024.

[Liu+24b] Toni J. B. Liu, Nicolas Boullé, Raphaël Sarfati, and Christopher J. Earls. *LLMs learn governing principles of dynamical systems, revealing an in-context neural scaling law*. Feb. 2024.

[Liu+24c] Toni J. B. Liu, Nicolas Boullé, Raphaël Sarfati, and Christopher J. Earls. *LLMs learn governing principles of dynamical systems, revealing an in-context neural scaling law*. 2024. arXiv: 2402.00795 [cs.LG].

[Liu+24d] Xu Liu, Juncheng Liu, Gerald Woo, Taha Aksu, Yuxuan Liang, Roger Zimmermann, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. "Moirai-MoE: Empowering Time Series Foundation Models with Sparse Mixture of Experts". In: *arXiv preprint arXiv:2410.10469* (2024).

[Liu+25] Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. "Understanding R1-Zero-Like Training: A Critical Perspective". In: *Second Conference on Language Modeling*. 2025.

[LK] Sergey Levine and Vladlen Koltun. "Guided Policy Search". In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, pp. 1–9.

[LLK21] Byung-Jun Lee, Jongmin Lee, and Kee-Eung Kim. "Representation Balancing Offline Model-based Reinforcement Learning". In: *International Conference on Learning Representations*. 2021.

[LM16] Steven Cheng-Xian Li and Benjamin M Marlin. "A scalable end-to-end Gaussian process adapter for irregularly sampled time series classification". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016.

[LPC22] Nathan Lambert, Kristofer Pister, and Roberto Calandra. *Investigating Compounding Prediction Errors in Learned Dynamics Models*. Mar. 2022.

[LR85] T.L Lai and Herbert Robbins. "Asymptotically efficient adaptive allocation rules". In: 6.1 (1985), pp. 4–22. ISSN: 0196-8858. DOI: 10.1016/0196-8858(85)90002-8.

[LSY25] Jiecheng Lu, Yan Sun, and Shihao Yang. "In-context Time Series Predictor". In: *The Thirteenth International Conference on Learning Representations*. 2025.

[Lu+22] Cong Lu, Philip Ball, Jack Parker-Holder, Michael Osborne, and Stephen J. Roberts. "Revisiting Design Choices in Offline Model Based Reinforcement Learning". In: *International Conference on Learning Representations*. 2022.

[Lu+24] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. "The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery". In: *arXiv preprint arXiv:2408.06292* (2024).

[Luo+19] Yuping Luo, Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. "Algorithmic Framework for Model-based Deep Reinforcement Learning with Theoretical Guarantees". In: *International Conference on Learning Representations*. 2019.

[Luo+22] Fan-Ming Luo, Tian Xu, Hang Lai, Xiong-Hui Chen, Weinan Zhang, and Yang Yu. *A Survey on Model-based Reinforcement Learning*. 2022. eprint: 2206.09328.

[Luo+24] Trung Quoc Luong, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. *ReFT: Reasoning with Reinforced Fine-Tuning*. 2024. arXiv: 2401.08967 [cs.CL].

[Lut+21] Michael Lutter, Leonard Hasenclever, Arunkumar Byravan, Gabriel Dulac-Arnold, Piotr Trochim, Nicolas Heess, Josh Merel, and Yuval Tassa. *Learning Dynamics Models for Model Predictive Agents*. Sept. 2021. DOI: 10.48550/arXiv.2109.14311.

[LZZ22] Minghuan Liu, Menghui Zhu, and Weinan Zhang. *Goal-Conditioned Reinforcement Learning: Problems and Solutions*. 2022. arXiv: 2201.08299 [cs.AI].

[Ma+24a] Junwei Ma, Valentin Thomas, Rasa Hosseinzadeh, Hamidreza Kamkari, Alex Labach, Jesse C. Cresswell, Keyvan Golestan, Guangwei Yu, Maksims Volkovs, and Anthony L. Caterini. *TabDPT: Scaling Tabular Foundation Models*. 2024. arXiv: 2410.18164 [cs.LG].

[Ma+24b] Runyu Ma, Jelle Luijkx, Zlatan Ajanovic, and Jens Kober. *ExploRLLM: Guiding Exploration in Reinforcement Learning with Large Language Models*. Mar. 2024. DOI: 10.48550/arXiv.2403.09583.

[MA93] Andrew W. Moore and Christopher G. Atkeson. "Prioritized Sweeping: Reinforcement Learning with Less Data and Less Time". In: 13.1 (Oct. 1993), pp. 103–130. ISSN: 0885-6125. DOI: 10.1023/A:1022635613229.

[Mac+19] Matthew Mackay, Paul Vicol, Jonathan Lorraine, David Duvenaud, and Roger Grosse. "Self-Tuning Networks: Bilevel Optimization of Hyperparameters using Structured Best-Response Functions". In: *International Conference on Learning Representations*. 2019.

[Mac95] David J.C MacKay. "Bayesian neural networks and density networks". In: 354.1 (1995), pp. 73–80. ISSN: 0168-9002. DOI: 10.1016/0168-9002(94)00931-7.

[MAF22] Vincent Micheli, Eloi Alonso, and François Fleuret. "Transformers are Sample-Efficient World Models". en. In: Sept. 2022.

[MAF23] Vincent Micheli, Eloi Alonso, and François Fleuret. "Transformers are Sample-Efficient World Models". In: *The Eleventh International Conference on Learning Representations*. 2023.

[Mah36] P.C. Mahalanobis. "On the Generalised Distance in Statistics". In: *Proceedings of the National Institute of Sciences of India* 2 (1936), pp. 49–55.

[Mal+19] Ali Malik, Volodymyr Kuleshov, Jiaming Song, Danny Nemer, Harlan Seymour, and Stefano Ermon. *Calibrated Model-Based Deep Reinforcement Learning*. June 2019.

[Man+23] Matteo Manica et al. “Accelerating material design with the generative toolkit for scientific discovery”. In: *npj Computational Materials* 9.1 (2023). ISSN: 2057-3960. DOI: 10.1038/s41524-023-01028-1.

[Mat+21] Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. “Deployment-Efficient Reinforcement Learning via Model-Based Offline Optimization”. In: *International Conference on Learning Representations*. 2021.

[Max21] Max Planck Institute. *Weather dataset*. 2021.

[Maz+24] Pietro Mazzaglia, Tim Verbelen, Bart Dhoedt, Aaron Courville, and Sai Rajeswar. *GenRL: Multimodal-foundation world models for generalization in embodied agents*. 2024. arXiv: 2406.18043 [cs.AI].

[McC+18] Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. *The Natural Language Decathlon: Multitask Learning as Question Answering*. 2018. eprint: 1806.08730.

[McN02] James McNames. “Local averaging optimization for chaotic time series prediction”. In: *Neurocomputing* 48.1 (2002), pp. 279–297. ISSN: 0925-2312. DOI: 10.1016/S0925-2312(01)00647-6.

[Mer25] Rick Merritt. *What Are Foundation Models?* 2025.

[Min+21] Matthias Minderer, Josip Djolonga, Rob Romijnders, Frances Ann Hubis, Xiaohua Zhai, Neil Houlsby, Dustin Tran, and Mario Lucic. “Revisiting the Calibration of Modern Neural Networks”. en. In: Nov. 2021.

[Mni+15] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. In: *Nature* 518.7540 (2015), pp. 529–533.

[Mni+16] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, and Koray Kavukcuoglu. “Asynchronous methods for deep reinforcement learning”. In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. ICML’16. New York, NY, USA: JMLR.org, 2016, pp. 1928–1937.

[Moe+23] Thomas M. Moerland, Joost Broekens, Aske Plaat, and Catholijn M. Jonker. “Model-based Reinforcement Learning: A Survey”. In: 16.1 (2023), pp. 1–118. ISSN: 1935-8237. DOI: 10.1561/2200000086.

[Mon+24] Arnab Kumar Mondal, Siba Smarak Panigrahi, Sai Rajeswar, Kaleem Siddiqi, and Siamak Ravanbakhsh. “Efficient Dynamics Modeling in Interactive Environments with Koopman Theory”. In: *The Twelfth International Conference on Learning Representations*. 2024.

[MU49] Nicholas Metropolis and Stanislaw Ulam. “The Monte Carlo Method”. In: 44.247 (1949), pp. 335–341. DOI: 10.1080/01621459.1949.10483310.

[Mun+16] Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare. "Safe and Efficient Off-Policy Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. Vol. 29. 2016, pp. 1054–1062.

[Nac+18] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. *Data-Efficient Hierarchical Reinforcement Learning*. 2018. arXiv: 1805.08296 [cs.LG].

[Nag+18] Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey Levine. "Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning". In: *2018 IEEE International Conference on Robotics and Automation, ICRA 2018*. IEEE, 2018, pp. 7559–7566.

[NC05] Alexandru Niculescu-Mizil and Rich Caruana. "Predicting good probabilities with supervised learning". In: *Proceedings of the 22nd International Conference on Machine Learning*. ICML '05. Bonn, Germany: Association for Computing Machinery, 2005, pp. 625–632. ISBN: 1595931805. DOI: 10.1145/1102351.1102430.

[NDD] Yael Niv, {Michael O.} Duff, and {Peter D.} Dayan. "Dopamine, uncertainty and TD learning". In: (). ISSN: 1744-9081. DOI: 10.1186/1744-9081-1-6.

[Nea96] Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, 1996. ISBN: 0387947248.

[NG10] Jason Ng and Jeffrey J. Goldberger. "Signal Averaging for Noise Reduction". In: *Practical Signal and Image Processing in Clinical Cardiology*. Ed. by Jeffrey J Goldberger and Jason Ng. London: Springer London, 2010, pp. 69–77. ISBN: 978-1-84882-515-4. DOI: 10.1007/978-1-84882-515-4_7.

[Ngu+] Tung D Nguyen, Rui Shu, Tuan Pham, Hung Bui, and Stefano Ermon. "Temporal Predictive Coding For Model-Based Planning In Latent Space". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 8130–8139.

[Nie+23] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. "A Time Series is Worth 64 Words: Long-term Forecasting with Transformers". In: *The Eleventh International Conference on Learning Representations*. 2023.

[Nik+21] Evgenii Nikishin, Romina Abachi, Rishabh Agarwal, and Pierre-Luc Bacon. *Control-Oriented Model-Based Reinforcement Learning with Implicit Differentiation*. 2021. arXiv: 2106.03273 [cs.LG].

[Niv09] Yael Niv. "Reinforcement learning in the brain". In: 53.3 (2009), pp. 139–154. ISSN: 0022-2496. DOI: 10.1016/j.jmp.2008.12.005.

[Nix+19] Jeremy Nixon, Mike Dusenberry, Ghassen Jerfel, Timothy Nguyen, Jeremiah Liu, Linchuan Zhang, and Dustin Tran. *Measuring Calibration in Deep Learning*. en. Apr. 2019.

[Nor+16] Mohammad Norouzi, Samy Bengio, zhifeng Chen, Navdeep Jaitly, Mike Schuster, Yonghui Wu, and Dale Schuurmans. "Reward Augmented Maximum Likelihood for Neural Structured Prediction". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016.

[NW18] Jakub Nowotarski and Rafał Weron. "Recent advances in electricity price forecasting: A review of probabilistic forecasting". In: *Renewable and Sustainable Energy Reviews* 81 (2018), pp. 1548–1568.

[Osw+23] Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. *Transformers learn in-context by gradient descent*. May 2023. DOI: 10.48550/arXiv.2212.07677.

[OT22] Masashi Okada and Tadahiro Taniguchi. *DreamingV2: Reinforcement Learning with Discrete World Models without Reconstruction*. 2022. eprint: 2203.00494.

[Ouy+22] Long Ouyang et al. *Training language models to follow instructions with human feedback*. 2022. arXiv: 2203.02155 [cs.CL].

[Pal12] TN Palmer. "Towards the probabilistic Earth-system simulator: A vision for the future of climate and weather prediction". In: *Quarterly Journal of the Royal Meteorological Society* 138.665 (2012), pp. 841–861.

[Pan+22] Junting Pan, Ziyi Lin, Xiatian Zhu, Jing Shao, and Hongsheng Li. "ST-Adapter: Parameter-Efficient Image-to-Video Transfer Learning". In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. Vol. 35. Curran Associates, Inc., 2022, pp. 26462–26477.

[Pao+24] Giuseppe Paolo, Jonas Gonzalez-Billandon, Albert Thomas, and Balázs Kégl. *Guided Safe Shooting: model based reinforcement learning with safety constraints*. 2024. eprint: 2206.09743.

[Pap+24] Theodore Papamarkou et al. "Position: Bayesian deep learning is needed in the age of large-scale AI". In: *Proceedings of the 41st International Conference on Machine Learning*. ICML'24. Vienna, Austria: JMLR.org, 2024.

[Pas+19] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[Pat+17] Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. *Robust Deep Reinforcement Learning with Adversarial Attacks*. Dec. 2017. DOI: 10.48550/arXiv.1712.03632.

[Ped16] Fabian Pedregosa. "Hyperparameter optimization with approximate gradient". In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Pro-

ceedings of Machine Learning Research. New York, New York, USA: PMLR, 2016, pp. 737–746.

[PFR] Eduardo Pignatelli, Johan Ferret, and Tim Rocktaschel. "Assessing the Zero-Shot Capabilities of LLMs for Action Evaluation in RL". en. In: ().

[Pha+23] Hieu Pham et al. *Combined Scaling for Zero-shot Transfer Learning*. 2023. eprint: 2111.10050.

[Pin+20] Cristina Pinneri, Shambhuraj Sawant, Sebastian Blaes, Jan Achterhold, Joerg Stueckler, Michal Rolinek, and Georg Martius. "Sample-efficient Cross-Entropy Method for Real-time Planning". In: *Conference on Robot Learning 2020*. 2020.

[Pin88] Fernando J Pineda. "Dynamics and architecture for neural computation". In: *Journal of Complexity* 4.3 (Sept. 1988), pp. 216–245. ISSN: 0885-064X. DOI: 10.1016/0885-064X(88)90021-0.

[Pla99] John C. Platt. "Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods". In: *Advances in Large Margin Classifiers*. MIT Press, 1999, pp. 61–74.

[PLB17] Biswajit Paria, Avisek Lahiri, and Prabir Kumar Biswas. "PolicyGAN: Training generative adversarial networks using policy gradient". In: *2017 Ninth International Conference on Advances in Pattern Recognition (ICAPR)*. 2017, pp. 1–6. DOI: 10.1109/ICAPR.2017.8593063.

[PMA24] Ieva Petrulionyte, Julien Mairal, and Michael Arbel. "Functional Bilevel Optimization for Machine Learning". In: *NeurIPS (Spotlight Poster)*. 2024.

[PMB21] Keiran Paster, Sheila A. McIlraith, and Jimmy Ba. "Planning from Pixels using Inverse Dynamics Models". In: *International Conference on Learning Representations*. 2021.

[Pod+23] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. *SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis*. 2023. eprint: 2307.01952.

[Pou+23] Rudra P. K. Poudel, Harit Pandya, Chao Zhang, and Roberto Cipolla. *LanGWM: Language Grounded World Model*. en. Nov. 2023.

[PS24] Nooshin Pourkamali and Shler Ebrahim Sharifi. "Machine Translation with Large Language Models: Prompt Engineering for Persian, English, and Russian Directions". In: *arXiv preprint arXiv:2401.08429* (2024).

[PS97] Doina Precup and Richard S Sutton. "Multi-time Models for Temporally Abstract Planning". In: *Advances in Neural Information Processing Systems*. Vol. 10. MIT Press, 1997.

[PSS00] Doina Precup, Richard S. Sutton, and Satinder Singh. "Eligibility Traces for Off-Policy Policy Evaluation". In: *Proceedings of the 17th International*

Conference on Machine Learning (ICML). Morgan Kaufmann, 2000, pp. 759–766.

[PSS98] Doina Precup, Richard S. Sutton, and Satinder Singh. "Theoretical results on reinforcement learning with temporally abstract options". en. In: *Machine Learning: ECML-98*. Ed. by Claire Nédellec and Céline Rouveirol. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1998, pp. 382–393. ISBN: 978-3-540-69781-7. DOI: 10.1007/BFb0026709.

[Put94] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. 1st. John Wiley & Sons, Inc., 1994. ISBN: 0471619779.

[PY10] Sinno Jialin Pan and Qiang Yang. "A Survey on Transfer Learning". In: 22.10 (2010), pp. 1345–1359.

[Qu+] Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. "TabICL: A Tabular Foundation Model for In-Context Learning on Large Data". In: *Proceedings of the 42nd International Conference on Machine Learning*. Ed. by Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu. Vol. 267. Proceedings of Machine Learning Research. PMLR, pp. 50817–50847.

[Rad+] Alec Radford et al. "Learning Transferable Visual Models From Natural Language Supervision". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 8748–8763.

[Rad+19] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. "Language models are unsupervised multitask learners". In: *OpenAI blog* 1.8 (2019), p. 9.

[Raf+21] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. "Stable-Baselines3: Reliable Reinforcement Learning Implementations". In: *Journal of Machine Learning Research* 22.268 (2021), pp. 1–8.

[Raf+23] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. "Direct Preference Optimization: Your Language Model is Secretly a Reward Model". In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023.

[Raj+19] Aravind Rajeswaran, Chelsea Finn, Sham Kakade, and Sergey Levine. *Meta-Learning with Implicit Gradients*. 2019. arXiv: 1909.04630 [cs.LG].

[Ram+] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. "Zero-Shot Text-to-Image Generation". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 8821–8831.

[Ran+16] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. *Sequence Level Training with Recurrent Neural Networks*. 2016. arXiv: 1511.06732 [cs.LG].

[Ras+24] Kashif Rasul et al. *Lag-Llama: Towards Foundation Models for Probabilistic Time Series Forecasting*. 2024. arXiv: 2310.08278 [cs.LG].

[Ren+23] Jie Ren, Xidong Feng, Bo Liu, Xuehai Pan, Yao Fu, Luo Mai, and Yaodong Yang. "TorchOpt: An Efficient Library for Differentiable Optimization". In: *Journal of Machine Learning Research* 24.367 (2023), pp. 1–14.

[Req+24] James Requeima, John Bronskill, Dami Choi, Richard E. Turner, and David Duvenaud. *LLM Processes: Numerical Predictive Distributions Conditioned on Natural Language*. May 2024. DOI: 10.48550/arXiv.2405.12856.

[RN94] Gavin A. Rummery and Mahesan Niranjan. "On-Line Q-Learning Using Connectionist Systems". In: CUED/F-INFENG/TR 166 (1994).

[Rom+21] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2021. arXiv: 2112.10752 [cs.CV].

[RRS20] Adam Roberts, Colin Raffel, and Noam Shazeer. "How Much Knowledge Can You Pack Into the Parameters of a Language Model?" In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu. Online: Association for Computational Linguistics, Nov. 2020, pp. 5418–5426. DOI: 10.18653/v1/2020.emnlp-main.437.

[RW06] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[RYG22] Machel Reid, Yutaro Yamada, and Shixiang Shane Gu. *Can Wikipedia Help Offline Reinforcement Learning?* July 2022. DOI: 10.48550/arXiv.2201.12122.

[Sam+24] Mohammad Reza Samsami, Artem Zholus, Janarthanan Rajendran, and Sarath Chandar. "Mastering Memory Tasks with World Models". In: *The Twelfth International Conference on Learning Representations*. 2024.

[SB18] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Second. The MIT Press, 2018.

[Sch+15] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. "Trust Region Policy Optimization". In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. PMLR, July 2015, pp. 1889–1897.

[Sch+16a] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. "Prioritized Experience Replay". In: *International Conference on Learning Representations*. 2016.

[Sch+16b] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. "High-Dimensional Continuous Control Using Generalized Advantage Estimation". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2016.

[Sch+17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG].

[Sch+20] Julian Schrittwieser et al. "Mastering Atari, Go, chess and shogi by planning with a learned model". In: 588.7839 (Dec. 2020), pp. 604–609. ISSN: 1476-4687. DOI: 10.1038/s41586-020-03051-4.

[Sei17] Daniel Seita. "Going Deeper Into Reinforcement Learning: Fundamentals of Policy Gradients". In: Mar. 2017.

[Sha+23] Mrinank Sharma, Sebastian Farquhar, Eric Nalisnick, and Tom Rainforth. "Do Bayesian Neural Networks Need To Be Fully Stochastic?" In: *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*. Ed. by Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent. Vol. 206. Proceedings of Machine Learning Research. PMLR, Spr. NH 2023, pp. 7694–7722.

[Sha+24] Zhihong Shao et al. *DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models*. 2024. arXiv: 2402.03300 [cs.CL].

[Shi+23] Ruizhe Shi, Yuyao Liu, Yanjie Ze, Simon S. Du, and Huazhe Xu. *Unleashing the Power of Pre-trained Language Models for Offline Reinforcement Learning*. Nov. 2023. DOI: 10.48550/arXiv.2310.20587.

[Shi+25] Lucy Xiaoyang Shi et al. *Hi Robot: Open-Ended Instruction Following with Hierarchical Vision-Language-Action Models*. 2025. arXiv: 2502.19417 [cs.RO].

[Sil+14] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. "Deterministic Policy Gradient Algorithms". In: *Proceedings of the 31st International Conference on Machine Learning*. Ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Research 1. PMLR, Sum. 2014, pp. 387–395.

[Sil+16] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. "Mastering the game of Go with deep neural networks and tree search". In: 529.7587 (2016), pp. 484–489. DOI: 10.1038/nature16961.

[Sil+17] David Silver et al. *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm*. 2017. arXiv: 1712.01815 [cs.AI].

[Sil+18] David Silver et al. "A general reinforcement learning algorithm that masters Chess, Shogi, and Go through self-play". In: *Science* 362.6419 (2018), pp. 1140–1144. ISSN: 0036-8075. DOI: 10.1126/science.aar6404.

[Sin92] Satinder P. Singh. "Scaling Reinforcement Learning Algorithms by Learning Variable Temporal Resolution Models". en. In: *Machine Learning Proceedings 1992*. Ed. by Derek Sleeman and Peter Edwards. San Francisco (CA): Morgan Kaufmann, Jan. 1992, pp. 406–415. ISBN: 978-1-55860-247-2. DOI: 10.1016/B978-1-55860-247-2.50058-9.

[Soh+15] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*. 2015. arXiv: 1503.03585 [cs.LG].

[Son+19] Hao Song, Tom Diethe, Meelis Kull, and Peter Flach. *Distribution Calibration for Regression*. May 2019.

[Son+20] Xingyou Song, Wenbo Gao, Yuxiang Yang, Krzysztof Choromanski, Aldo Pacchiano, and Yunhao Tang. "ES-MAML: Simple Hessian-Free Meta Learning". In: *International Conference on Learning Representations*. 2020.

[Son+23] Sumedh Anand Sontakke, Jesse Zhang, Séb Arnold, Karl Pertsch, Erdem Biyik, Dorsa Sadigh, Chelsea Finn, and Laurent Itti. "RoboCLIP: One Demonstration is Enough to Learn Robot Policies". In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023.

[Son+24] Feifan Song, Bowen Yu, Minghao Li, Haiyang Yu, Fei Huang, Yongbin Li, and Houfeng Wang. *Preference Ranking Optimization for Human Alignment*. 2024. arXiv: 2306.17492 [cs.CL].

[SP85] Richard S Sutton and Brian Pinette. *The learning of world models by connectionist networks*. 1985.

[SPA25] Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. *RL's Razor: Why Online Reinforcement Learning Forgets Less*. 2025. arXiv: 2509.04259 [cs.LG].

[SPS99] Richard S. Sutton, Doina Precup, and Satinder Singh. "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning". en. In: *Artificial Intelligence* 112.1 (Aug. 1999), pp. 181–211. ISSN: 0004-3702. DOI: 10.1016/S0004-3702(99)00052-1.

[Sri+14a] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958.

[Sri+14b] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958.

[Sta34] Heinrich von Stackelberg. *Marktform und Gleichgewicht*. Berlin: Springer-Verlag, 1934.

[Sti+20] Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. "Learn-

ing to summarize from human feedback". In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS '20. Vancouver, BC, Canada: Curran Associates Inc., 2020. ISBN: 9781713829546.

[Sun+23] Ke Sun, Yingnan Zhao, Shangling Jui, and Linglong Kong. *Exploring the Training Robustness of Distributional Reinforcement Learning against Noisy State Observations*. June 2023. DOI: 10.48550/arXiv.2109.08776.

[Sun+24] Minjun Sung, Sambhu Harimanas Karumanchi, Aditya Gahlawat, and Naira Hovakimyan. "Robust Model Based Reinforcement Learning Using $\mathcal{L}_1$ Adaptive Control". In: *The Twelfth International Conference on Learning Representations*. 2024.

[Sut+00] Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. "Policy Gradient Methods for Reinforcement Learning with Function Approximation". In: *Advances in Neural Information Processing Systems*. Vol. 12. MIT Press, 2000, pp. 1057–1063.

[Sut+92] Richard S Sutton, Csaba Szepesvári, Alborz Geramifard, and Michael Bowling. "Dyna-Style Planning with Linear Function Approximation and Prioritized Sweeping". In: *Moore and Atkeson* (1992).

[Sut88] Richard S. Sutton. "Learning to predict by the methods of temporal differences". In: 3.1 (1988), pp. 9–44. DOI: 10.1007/BF00115009.

[Sut91] Richard S. Sutton. "Dyna, an integrated architecture for learning, planning, and reacting". In: *ACM SIGART Bulletin* 2 (4 July 1991), pp. 160–163. ISSN: 0163-5719. DOI: 10.1145/122344.122377.

[Sut95] Richard S. Sutton. "TD Models: Modeling the World at a Mixture of Time Scales". en. In: *Machine Learning Proceedings 1995*. Ed. by Armand Prieditis and Stuart Russell. San Francisco (CA): Morgan Kaufmann, Jan. 1995, pp. 531–539. ISBN: 978-1-55860-377-6. DOI: 10.1016/B978-1-55860-377-6.50072-4.

[Swa+25] Gokul Swamy, Sanjiban Choudhury, Wen Sun, Zhiwei Steven Wu, and J. Andrew Bagnell. *All Roads Lead to Likelihood: The Value of Reinforcement Learning in Fine-Tuning*. 2025. arXiv: 2503.01067 [cs.LG].

[SYC24] Han Shen, Zhuoran Yang, and Tianyi Chen. *Principled Penalty-based Methods for Bilevel Reinforcement Learning and RLHF*. 2024. arXiv: 2402.06886 [cs.LG].

[Tal14] Erik Talvitie. "Model Regularization for Stable Sample Rollouts". en. In: (2014).

[Tal17] Erik Talvitie. "Self-Correcting Models for Model-Based Reinforcement Learning". en. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 31.1 (Feb. 2017). ISSN: 2374-3468, 2159-5399. DOI: 10.1609/aaai.v31i1.10850.

[Tan+19] Bowen Tan, Zhiting Hu, Zichao Yang, Ruslan Salakhutdinov, and Eric Xing. *Connecting the Dots Between MLE and RL for Sequence Prediction*. 2019. arXiv: 1811.09740 [cs.LG].

[Tas+18] Yuval Tassa et al. "DeepMind Control Suite". In: *CoRR* abs/1801.00690 (2018). arXiv: 1801.00690.

[TET12] Emanuel Todorov, Tom Erez, and Yuval Tassa. "MuJoCo: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 5026–5033. DOI: 10.1109/IROS.2012.6386109.

[Tho+24] Albert Thomas, Abdelhakim Benechehab, Giuseppe Paolo, and Balázs Kégl. "Fair Model-Based Reinforcement Learning Comparisons with Explicit and Consistent Update Frequency". In: *The Third Blogpost Track at ICLR 2024*. 2024.

[Tia+23] Zhou Tian, Niu Peisong, Wang Xue, Sun Liang, and Jin Rong. "One Fits All: Power General Time Series Analysis by Pretrained LM". In: *NeurIPS*. 2023.

[TON95] Naoki Tanaka, Hiroshi Okamoto, and Masayoshi Naito Masayoshi Naito. "An Optimal Metric for Predicting Chaotic Time Series". In: *Japanese Journal of Applied Physics* 34.1R (Jan. 1995), p. 388. DOI: 10.1143/JJAP.34.388.

[Tou+23] Hugo Touvron et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 2023. arXiv: 2307.09288 [cs.CL].

[Tra+22] Ba-Hien Tran, Simone Rossi, Dimitrios Milios, and Maurizio Filippone. "All you need is a good functional prior for Bayesian deep learning". In: 23.1 (2022). ISSN: 1532-4435.

[Ueh+24] Masatoshi Uehara, Yulai Zhao, Ehsan Hajiramezanali, Gabriele Scalia, Gökcen Eraslan, Avantika Lal, Sergey Levine, and Tommaso Biancalani. *Bridging Model-Based Optimization and Generative Modeling via Conservative Fine-Tuning of Diffusion Models*. 2024. arXiv: 2405.19673 [cs.LG].

[UR20] Saiteja Utpala and Piyush Rai. *Quantile Regularization: Towards Implicit Calibration of Regression Models*. Feb. 2020.

[US 24] U.S. Centers for Disease Control and Prevention. *Fluview: Flu activity & surveillance*. 2024.

[USK22] Masatoshi Uehara, Chengchun Shi, and Nathan Kallus. *A Review of Off-Policy Evaluation in Reinforcement Learning*. 2022. arXiv: 2212.06355 [stat.ML].

[Vac+24] Robert Vacareanu, Vlad-Andrei Negru, Vasile Suciu, and Mihai Surdeanu. *From Words to Numbers: Your Large Language Model Is Secretly A Capable Regressor When Given In-Context Examples*. Sept. 2024. DOI: 10.48550/arXiv.2404.07544.

[van07] Wim van Drongelen. "4 - Signal Averaging". In: *Signal Processing for Neuroscientists*. Ed. by Wim van Drongelen. Burlington: Academic Press, 2007,

pp. 55–70. ISBN: 978-0-12-370867-0. DOI: 10.1016/B978-012370867-0/50004-8.

[Vas+] Apostol Vassilev, Alina Oprea, Alie Fordyce, Hyrum Anderson, Xander Davies, and Maia Hamin. *Adversarial Machine Learning: A Taxonomy and Terminology of Attacks and Mitigations*. en. DOI: 10.6028/NIST.AI.100-2e2025.

[Vas+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010. ISBN: 9781510860964.

[Vas+23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL].

[VHB15a] Arun Venkatraman, Martial Hebert, and J. Andrew Bagnell. "Improving multi-step prediction of learned time series models". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI'15. Austin, Texas: AAAI Press, 2015, pp. 3024–3030. ISBN: 0262511290.

[VHB15b] Arun Venkatraman, Martial Hebert, and J.. Bagnell. "Improving Multi-Step Prediction of Learned Time Series Models". en. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 29.1 (Feb. 2015). ISSN: 2374-3468, 2159-5399. DOI: 10.1609/aaai.v29i1.9590.

[Vin+19] Oriol Vinyals et al. "Grandmaster level in StarCraft II using multi-agent reinforcement learning". In: *Nature* (2019), pp. 1–5.

[VLZ11] Maksims N. Volkovs, Hugo Larochelle, and Richard S. Zemel. *Loss-sensitive Training of Probabilistic Conditional Random Fields*. 2011. arXiv: 1107.1805 [stat.ML].

[Voe+22] Claas A Voelcker, Victor Liao, Animesh Garg, and Amir-massoud Farahmand. "Value Gradient weighted Model-Based Reinforcement Learning". In: *International Conference on Learning Representations (ICLR)*. 2022.

[Wan+16] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. "Dueling Network Architectures for Deep Reinforcement Learning". In: *Proceedings of the 33rd International Conference on Machine Learning*. Vol. 48. PMLR, 2016, pp. 1995–2003.

[Wan+17] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. "Sample Efficient Actor-Critic with Experience Replay". In: *International Conference on Learning Representations*. 2017.

[Wan+19] Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy

Ba. "Benchmarking Model-Based Reinforcement Learning". In: *arXiv preprint arXiv:1907.02057* (2019). arXiv: 1907.02057 [cs.LG].

[Wan+23] Yen-Jen Wang, Bike Zhang, Jianyu Chen, and Koushil Sreenath. *Prompt a Robot to Walk with Large Language Models*. Nov. 2023. DOI: 10.48550/arXiv.2309.09969.

[Wan+24] Xiyao Wang, Ruijie Zheng, Yanchao Sun, Ruonan Jia, Wichayaporn Wongkamjan, Huazhe Xu, and Furong Huang. "COPlanner: Plan to Roll Out Conservatively but to Explore Optimistically for Model-Based RL". In: *The Twelfth International Conference on Learning Representations*. 2024.

[Wan+25] Yucen Wang, Rui Yu, Shenghua Wan, Le Gan, and De-Chuan Zhan. "FOUNDER: Grounding Foundation Models in World Models for Open-Ended Embodied Decision Making". In: *Forty-second International Conference on Machine Learning*. 2025.

[Wan23] Jingcong Wang. *UCI datasets*. 2023. DOI: 10.21227/g4y0-sw34.

[WAT17] Grady Williams, Andrew Aldrich, and Evangelos A. Theodorou. "Model Predictive Path Integral Control: From Theory to Parallel Computation". In: 40.2 (2017), pp. 344–357. DOI: 10.2514/1.G001921. eprint: https://doi.org/10.2514/1.G001921.

[Wat89] Christopher J. C. H. Watkins. "Learning from Delayed Rewards". In: (1989).

[WD92] Christopher J. C. H. Watkins and Peter Dayan. "Q-learning". In: 8.3-4 (1992), pp. 279–292. DOI: 10.1007/BF00992698.

[Wei19] William WS Wei. *Multivariate time series analysis and applications*. John Wiley & Sons, 2019.

[Wen+24] Muning Wen, Junwei Liao, Cheng Deng, Jun Wang, Weinan Zhang, and Ying Wen. *Entropy-Regularized Token-Level Policy Optimization for Language Agent Reinforcement*. 2024. arXiv: 2402.06700 [cs.LG].

[Wen64] R. E. Wengert. "A simple automatic derivative evaluation program". In: *Commun. ACM* 7.8 (Aug. 1964), pp. 463–464. ISSN: 0001-0782. DOI: 10.1145/355586.364791.

[Wer+20] Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. *TRL: Transformers Reinforcement Learning*. 2020.

[Wie+25] Thaddäus Wiedemer, Yuxuan Li, Paul Vicol, Shixiang Shane Gu, Nick Matarese, Kevin Swersky, Been Kim, Priyank Jaini, and Robert Geirhos. "Video models are zero-shot learners and reasoners". In: (2025).

[Wil92a] R. J. Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine Learning* 8 (1992), pp. 229–256.

[Wil92b] Ronald J. Williams. "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning". In: *Mach. Learn.* 8.3–4 (May 1992), pp. 229–256. ISSN: 0885-6125. DOI: 10.1007/BF00992696.

[Wol+20] Thomas Wolf et al. "Transformers: State-of-the-Art Natural Language Processing". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45.

[WT01] Lex Weaver and Nigel Tao. "The Optimal Reward Baseline for Gradient-Based Reinforcement Learning". In: *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence (UAI)*. Morgan Kaufmann, 2001, pp. 538–545.

[Wu+21] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. "Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting". In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan. 2021.

[Wu+24] Yue Wu, Yewen Fan, Paul Pu Liang, Amos Azaria, Yuanzhi Li, and Tom M. Mitchell. *Read and Reap the Rewards: Learning to Play Atari with the Help of Instruction Manuals*. 2024. arXiv: 2302.04449 [cs.LG].

[WZ89] Ronald J. Williams and David Zipser. "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks". In: *Neural Computation* 1.2 (June 1989), pp. 270–280. ISSN: 0899-7667. DOI: 10.1162/neco.1989.1.2.270.

[Xia+25] Quan Xiao, Hui Yuan, A F M Saif, Gaowen Liu, Ramana Rao Kompella, Mengdi Wang, and Tianyi Chen. "A First-order Generative Bilevel Optimization Framework for Diffusion Models". In: *Forty-second International Conference on Machine Learning*. 2025.

[Xie+22] Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. *An Explanation of In-context Learning as Implicit Bayesian Inference*. July 2022. DOI: 10.48550/arXiv.2111.02080.

[Xin+25] Huajian Xin et al. "DeepSeek-Prover-V1.5: Harnessing Proof Assistant Feedback for Reinforcement Learning and Monte-Carlo Tree Search". In: *The Thirteenth International Conference on Learning Representations*. 2025.

[XLY20a] Tian Xu, Ziniu Li, and Yang Yu. "Error Bounds of Imitating Policies and Environments". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 15737–15749.

[XLY20b] Tian Xu, Ziniu Li, and Yang Yu. "Error bounds of imitating policies and environments". In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS '20. Vancouver, BC, Canada: Curran Associates Inc., 2020. ISBN: 9781713829546.

[XS23] Hao Xue and Flora D. Salim. *PromptCast: A New Prompt-based Learning Paradigm for Time Series Forecasting*. Dec. 2023. DOI: 10.48550/arXiv.2210.08964.

[Xu+18] Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. "Algorithmic Framework for Model-based Reinforcement Learning with Theoretical Guarantees". In: *CoRR* abs/1807.03858 (2018). arXiv: 1807.03858.

[Yan+23] Sherry Yang, Ofir Nachum, Yilun Du, Jason Wei, Pieter Abbeel, and Dale Schuurmans. *Foundation Models for Decision Making: Problems, Methods, and Opportunities*. Mar. 2023.

[Yan+24] Adam X. Yang, Maxime Robeyns, Xi Wang, and Laurence Aitchison. "Bayesian Low-rank Adaptation for Large Language Models". In: *The Twelfth International Conference on Learning Representations*. 2024.

[Yan+25] An Yang et al. *Qwen3 Technical Report*. 2025. eprint: 2505.09388.

[YGY25] Yan Yang, Bin Gao, and Ya-xiang Yuan. *Bilevel reinforcement learning via the development of hyper-gradient without lower-level convexity*. 2025. arXiv: 2405.19697 [math.OC].

[Yin+23] Dongshuo Yin, Leiyi Hu, Bin Li, and Youqun Zhang. *Adapter is All You Need for Tuning Visual Tasks*. 2023. arXiv: 2311.15010 [cs.CV].

[Yin+24] Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. "A survey on multimodal large language models". In: *National Science Review* 11.12 (Nov. 2024). ISSN: 2053-714X. DOI: 10.1093/nsr/nwae403.

[You+24] Omar G. Younis, Rodrigo Perez-Vicente, John U. Balis, Will Dudley, Alex Davey, and Jordan K Terry. *Minari*. Version 0.5.0. Sept. 2024. DOI: 10.5281/zenodo.13767625.

[Yu+17] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. *SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient*. 2017. arXiv: 1609.05473 [cs.LG].

[Yu+20a] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. "MOPO: Model-based Offline Policy Optimization". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 14129–14142.

[Yu+20b] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. "MOPO: Model-based Offline Policy Optimization". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 14129–14142.

[Yu+21a] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. "COMBO: Conservative Offline Model-Based

Policy Optimization". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., 2021, pp. 28954–28967.

[Yu+21b] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. "COMBO: Conservative Offline Model-Based Policy Optimization". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., 2021, pp. 28954–28967.

[Yu+25] Qiying Yu et al. "DAPO: An Open-Source LLM Reinforcement Learning System at Scale". In: *The Thirty-ninth Annual Conference on Neural Information Processing Systems*. 2025.

[YX24a] Yu Yang and Pan Xu. *Pre-trained Language Models Improve the Few-shot Prompt Ability of Decision Transformer*. en. Aug. 2024.

[YX24b] Zhihe YANG and Yunjian Xu. "DMBP: Diffusion model-based predictor for robust offline reinforcement learning against state observation perturbations". In: *The Twelfth International Conference on Learning Representations*. 2024.

[ZB19] Andrea Zanette and Emma Brunskill. "Tighter Problem-Dependent Regret Bounds in Reinforcement Learning without Domain Knowledge using Value Function Bounds". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 7304–7312.

[ZB25] Oussama Zekri and Nicolas Boullé. *Fine-Tuning Discrete Diffusion Models with Policy Gradient Methods*. 2025. arXiv: 2502.01384 [stat.ML].

[ZBR] Oussama Zekri, Abdelhakim Benechehab, and Ievgen Redko. "Can LLMs predict the convergence of Stochastic Gradient Descent?" In: *ICML 2024 Workshop on In-Context Learning*.

[ZE01] Bianca Zadrozny and Charles Elkan. "Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers". In: *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML '01. Morgan Kaufmann Publishers Inc., 2001, pp. 609–616. ISBN: 1558607781.

[ZE02] Bianca Zadrozny and Charles Elkan. "Transforming classifier scores into accurate multiclass probability estimates". In: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '02. Edmonton, Alberta, Canada: Association for Computing Machinery, 2002, pp. 694–699. ISBN: 158113567X. DOI: 10.1145/775047.775151.

[Zek+24] Oussama Zekri, Ambroise Odonnat, Abdelhakim Benechehab, Linus Bleistein, Nicolas Boullé, and Ievgen Redko. "Large language models as markov chains". In: *arXiv preprint arXiv:2410.02724* (2024).

[Zen+22] Siliang Zeng, Chenliang Li, Alfredo Garcia, and Mingyi Hong. "Maximum-likelihood inverse reinforcement learning with finite-time guarantees". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 10122–10135.

[Zen+23] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. "Are Transformers Effective for Time Series Forecasting?" In: 2023.

[Zha+21] Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Mingyan Liu, Duane Boning, and Cho-Jui Hsieh. *Robust Deep Reinforcement Learning against Adversarial Perturbations on State Observations*. July 2021. DOI: 10.48550/arXiv.2003.08938.

[Zha+23a] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. "Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning". In: *The Eleventh International Conference on Learning Representations*. 2023.

[Zha+23b] Weipu Zhang, Gang Wang, Jian Sun, Yetian Yuan, and Gao Huang. *STORM: Efficient Stochastic Transformer based World Models for Reinforcement Learning*. en. Oct. 2023.

[Zha+23c] Weipu Zhang, Gang Wang, Jian Sun, Yetian Yuan, and Gao Huang. "STORM: Efficient Stochastic Transformer based World Models for Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine. Vol. 36. Curran Associates, Inc., 2023, pp. 27147–27166.

[Zha+23d] Wenxuan Zhang, Yue Deng, Bing Liu, Sinno Jialin Pan, and Lidong Bing. *Sentiment Analysis in the Era of Large Language Models: A Reality Check*. 2023. arXiv: 2305.15005 [cs.CL].

[Zha+23e] Yufeng Zhang, Fengzhuo Zhang, Zhuoran Yang, and Zhaoran Wang. *What and How does In-Context Learning Learn? Bayesian Model Averaging, Parameterization, and Generalization*. 2023. arXiv: 2305.19420 [stat.ML].

[Zha+24a] Jingyi Zhang, Jiaxing Huang, Sheng Jin, and Shijian Lu. *Vision-Language Models for Vision Tasks: A Survey*. 2024. arXiv: 2304.00685 [cs.CV].

[Zha+24b] Shenao Zhang, Sirui Zheng, Shuqi Ke, Zhihan Liu, Wanxin Jin, Jianbo Yuan, Yingxiang Yang, Hongxia Yang, and Zhaoran Wang. *How Can LLM Guide RL? A Value-Based Approach*. Feb. 2024.

[Zha+24c] Weiqi Zhang, Jiexia Ye, Ziyue Li, Jia Li, and Fugee Tsung. *DualTime: A Dual-Adapter Multimodal Language Model for Time Series Representation*. 2024. arXiv: 2406.06620 [cs.LG].

[Zhe+23] Ruijie Zheng, Xiyao Wang, Huazhe Xu, and Furong Huang. "Is Model Ensemble Necessary? Model-based RL via a Single Model with Lipschitz Regularized Value Function". In: *The Eleventh International Conference on Learning Representations*. 2023.

[Zhe+24] Zhaoheng Zheng, Jingmin Wei, Xuefeng Hu, Haidong Zhu, and Ram Nevatia. "Large Language Models are Good Prompt Learners for Low-Shot Image Classification". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 28453–28462.

[Zho+21] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. "Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting". In: *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference*. Vol. 35. 12. AAAI Press, 2021, pp. 11106–11115.

[Zhu+25] Hanqing Zhu et al. *The Path Not Taken: RLVR Provably Learns Off the Principals*. 2025. eprint: 2511.08567.

[Zie+08] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. "Maximum entropy inverse reinforcement learning". In: *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*. AAAI'08. Chicago, Illinois: AAAI Press, 2008, pp. 1433–1438. ISBN: 9781577353683.

[Zit+23] Brianna Zitkovich et al. "RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control". In: *7th Annual Conference on Robot Learning*. 2023.

[Zou+19] Haosheng Zou, Tongzheng Ren, Dong Yan, Hang Su, and Jun Zhu. "Reward shaping via meta-learning". In: *arXiv preprint arXiv:1901.09330* (2019).

[ZWW14] Yao Zhang, Jianxue Wang, and Xifan Wang. "Review on probabilistic forecasting of wind power generation". In: *Renewable and Sustainable Energy Reviews* 32 (2014), pp. 255–270.

[ZY21] Yu Zhang and Qiang Yang. *A Survey on Multi-Task Learning*. 2021. eprint: 1707.08114.

[ZY23] Yunhao Zhang and Junchi Yan. "Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting". In: *The eleventh international conference on learning representations*. 2023.

[ZZX21] Xianyuan Zhan, Xiangyu Zhu, and Haoran Xu. "Model-Based Offline Planning with Trajectory Pruning". In: (2021).

Appendix A

Résumé détaillé en Français

Apprentissage par renforcement basé sur des modèles à l'ère des modèles de fondation

Abdelhakim Benechehab

L'apprentissage par renforcement (RL) fournit un cadre général pour la prise de décision séquentielle, permettant à des agents d'apprendre par interaction avec un environnement. Malgré des succès dans des domaines tels que les jeux et la robotique, son déploiement dans des contextes réels demeure difficile, notamment en raison d'une efficacité limitée en termes d'échantillons.

L'apprentissage par renforcement basé sur des modèles (MBRL) répond en partie à ce problème en apprenant un modèle de la dynamique de l'environnement afin de planifier, réduisant ainsi les interactions coûteuses. Cependant, les modèles appris introduisent de nouveaux défis, notamment l'accumulation d'erreurs et le décalage entre les objectifs du modèle et de la politique.

Parallèlement, l'apprentissage automatique a évolué vers les modèles de fondation (FM), des modèles pré-entraînés à grande échelle qui apprennent des représentations transférables à partir de jeux de données massifs. Initialement développés pour le traitement automatique du langage naturel, ils émergent désormais dans des domaines tels que la modélisation de séries temporelles, soulevant des questions quant à leur rôle en apprentissage par renforcement.

Cette thèse étudie comment l'intégration de modèles de fondation peut renforcer le MBRL.

A.1 Résumé du chapitre 3

A.1.1 Introduction

Ce chapitre, basé sur l'article intitulé *A Multi-step Loss Function for Robust Learning of the Dynamics in Model-based Reinforcement Learning*, s'inscrit dans le cadre de l'apprentissage par renforcement basé sur un modèle (*Model-Based Reinforcement Learning*, MBRL). Dans ce paradigme, l'agent apprend explicitement un modèle des dynamiques de l'environnement, c'est-à-dire une approximation de la transition $s_{t+1} \sim p(\cdot \mid s_t, a_t)$, afin de planifier ou d'améliorer sa politique.

Un problème central du MBRL réside dans l'erreur de modélisation. Lorsque le modèle appris est utilisé de manière itérative pour prédire plusieurs pas de temps dans le futur, les erreurs à un pas se propagent et peuvent croître de manière significative. Cette accumulation d'erreurs dégrade fortement la performance en planification, même si l'erreur de prédiction à un pas est faible.

On propose donc une nouvelle fonction de perte multi-pas visant à améliorer la robustesse de l'apprentissage des dynamiques, en prenant explicitement en compte l'effet cumulatif des erreurs de prédiction. L'idée principale est de remplacer (ou compléter) la perte classique à un pas par une perte intégrant des prédictions sur plusieurs horizons temporels.

A.1.2 Contexte et motivation

Dans le cadre standard, on apprend un modèle paramétrique f_θ tel que $\hat{s}_{t+1} = f_\theta(s_t, a_t)$, en minimisant une perte de type erreur quadratique moyenne :

$$\mathcal{L}_{\text{1-step}}(\theta) = \mathbb{E}_{(s_t, a_t, s_{t+1})} \left[\left| f_\theta(s_t, a_t) - s_{t+1} \right|^2 \right].$$

Cette approche optimise uniquement la qualité de la prédiction à un pas. Or, en planification, le modèle est utilisé récursivement :

$$\hat{s}_{t+k} = f_\theta(\hat{s}_{t+k-1}, a_{t+k-1}),$$

ce qui introduit une dépendance forte aux erreurs précédentes.

Des travaux antérieurs ont étudié la propagation d'erreur et la stabilité des modèles dynamiques [AGN05; Tal14; Tal17; VHB15b]. Cependant, beaucoup de méthodes continuent d'optimiser uniquement une perte locale à un pas.

A.1.3 Fonction de perte multi-pas proposée

On introduit une perte multi-step de la forme :

$$\mathcal{L}_{K\text{-step}}(\theta) = \mathbb{E}\left[\sum_{k=1}^{K} w_k \cdot |\hat{s}_{t+k} - s_{t+k}|^2\right],$$

où :

- $\hat{s}_{t+k}$ est obtenu par déroulement récursif du modèle,
- s_{t+k} est l'état réel observé,
- w_k sont des poids éventuellement décroissants,
- K est l'horizon multi-pas considéré.

De part cette définition, la perte proposée peut être vue comme une forme de régularisation structurelle qui intègre implicitement une contrainte de cohérence temporelle. Ainsi, au lieu d'optimiser uniquement la prédiction immédiate, on contraint le modèle à produire des trajectoires cohérentes sur un horizon étendu. La procédure de calcul de la perte multi-pas est différentiable de bout en bout, ce qui permet l'utilisation de la rétropropagation à travers le déroulement temporel (similaire à *Backpropagation Through Time*). Le coût computationnel augmente linéairement avec K, mais reste compatible avec les pratiques standards en deep learning.

A.1.4 Résultats expérimentaux

Les expériences sont conduites sur plusieurs environnements continus classiques en apprentissage par renforcement [Bro+16; Tas+18; TET12].

Les principaux résultats empiriques sont les suivants :

- La perte multi-pas améliore significativement la qualité des trajectoires simulées à horizon long.
- Les modèles entraînés avec la perte proposée présentent une erreur cumulative plus faible surtout en cas de bruit Gaussian dans les données d'entraînement.
- En planification (via MPC ou autre méthode), les performances finales en termes de récompense sont mitigées, et legèrement supérieures à celles obtenues avec une perte à un pas.
- Les résultats montrent également une meilleure robustesse face aux perturbations et aux erreurs de distribution entre données d'entraînement et états visités lors de la planification.

A.1.5 Conclusion

Le chapitre propose une contribution simple mais conceptuellement forte : remplacer la perte standard à un pas par une perte multi-pas afin d'améliorer la robustesse des modèles dynamiques en MBRL. Cette modification aligne l'objectif d'apprentissage avec l'usage réel du modèle en planification, réduisant ainsi l'écart entre performance supervisée et performance en contrôle. Empiriquement, elle améliore de manière consistante les performances sur plusieurs environnements à espace d'états continu. En résumé, cette approche constitue une avancée méthodologique pertinente pour le MBRL, en mettant en évidence l'importance du choix de la fonction de perte dans l'apprentissage des dynamiques.

Limitations potentielles. Cependant, certains compromis apparaissent : 1) Coût computationnel plus élevé pour de grands horizons K. 2) Sensibilité possible au choix des poids w_k. 3) Risque d'optimisation plus difficile en raison de gradients à travers de longues chaînes de prédiction. De plus, le choix de l'horizon K introduit un hyperparamètre supplémentaire, dont l'impact dépend probablement de la nature de l'environnement.

A.2 Résumé du chapitre 4

A.2.1 Introduction et Problématique

L'émergence des capacités "zero-shot" des modèles de langage à large échelle (LLM) a révolutionné le traitement du langage naturel, mais leur application à d'autres domaines, notamment le contrôle continu en apprentissage par renforcement (RL), reste un défi majeur. Ce chapitre explore la capacité des LLM pré-entraînés à prédire, en contexte, la dynamique de processus de décision Markoviens (MDP) continus. L'idée centrale repose sur l'apprentissage en contexte (*In-Context Learning* ou ICL), où le modèle apprend à réaliser une tâche à partir d'exemples fournis dans son prompt, sans mise à jour de ses poids par gradient. Cependant, l'application directe des LLM aux trajectoires de RL se heurte à deux obstacles : la gestion de données multivariées interdépendantes et l'intégration efficace des signaux de contrôle (actions). Pour y répondre, nous introduisons le framework **DICL (Disentangled In-Context Learning)**.

A.2.2 Cadre Méthodologique

Motivation. Une analyse préliminaire des mécanismes d'attention (notamment sur le modèle Llama 3-8B) révèle que les LLM sont naturellement sensibles aux motifs tem-

porels, comme les signaux périodiques ou les constantes par morceaux. Dans des environnements de RL comme HalfCheetah, les tokens appartenant à des motifs répétitifs (pics de signaux) s'attendent mutuellement à travers les cycles, ce qui suggère une capacité intrinsèque à modéliser des dynamiques récurrentes.

Formalism. Le problème est défini par une trajectoire initiale $\mathcal{T} = (s_0, a_0, r_1, s_1, ..., r_{T-1}, s_{T-1})$ de longueur T. L'objectif est de prédire les transitions futures (s_{T+h}, r_{T+h}) de manière autorégressive. Le cadre DICL repose sur trois étapes clés :

- **Transformation de l'espace des caractéristiques (φ) :** On utilise l'Analyse en Composantes Principales (PCA) pour projeter les vecteurs d'état et d'action dans un espace latent où les dimensions sont linéairement décorrélées (désentrelacées).
- **Prédiction par LLM :** Chaque composante désentrelacée est traitée indépendamment comme une série temporelle univariée. Le LLM effectue une prédiction zero-shot en utilisant une tokenisation numérique spécifique.
- **Transformation inverse (φ^{-1}) :** Les prédictions sont projetées à nouveau dans l'espace original de la trajectoire.

Deux variantes sont proposées : **DICL-(s)**, qui transforme uniquement les états, et **DICL-(s, a)**, qui intègre les actions dans la transformation. L'utilisation de la PCA permet non seulement de respecter l'hypothèse d'indépendance des caractéristiques nécessaire aux prédicteurs univariés, mais aussi de réduire la dimensionnalité, accélérant ainsi le temps de calcul sans perte significative de performance.

A.2.3 Analyse Théorique

Pour garantir l'efficacité de l'utilisation d'un LLM comme modèle de dynamique, nous analysons le rendement sous un schéma de *déroulements à branches multiples* (**multi-branch rollouts**). Soit T la longueur minimale du contexte. Pour $t \geq T$, le système bascule avec une probabilité p vers la dynamique apprise par le LLM ($\hat{P}_{llm}$) pour k étapes, sinon il suit la dynamique réelle P. En supposant que l'erreur de généralisation de l'apprentissage en contexte est bornée par $\epsilon_{llm}(T)$, l'écart entre le rendement réel $\eta(\pi)$ et le rendement du modèle $\eta^{llm}_{p,k,T}(\pi)$ est donné par :

$$|\eta(\pi) - \eta^{llm}_{p,k,T}(\pi)| \leq 2\frac{\gamma^T}{1-\gamma} r_{max} k^2 p \epsilon_{llm}(T)$$

où γ est le facteur d'escompte et r_{max} la récompense maximale. Cette borne montre que l'erreur diminue avec la longueur du contexte T (typiquement en $\mathcal{O}(T^{-1/2})$) et augmente avec l'horizon de prédiction k et la fréquence de branchement p.

A.2.4 Applications et Résultats Expérimentaux

Les performances de DICL ont été évaluées sur plusieurs environnements de contrôle continu (MuJoCo, Gym). Nous intégrons DICL dans l'algorithme Soft Actor-Critic (SAC) pour améliorer l'efficacité échantillonnale. En générant des transitions synthétiques précises à partir de trajectoires courtes, DICL-SAC accélère significativement la courbe d'apprentissage lors des premières étapes de l'entraînement, notamment dans les environnements Pendulum et HalfCheetah. Cependant, une trop grande proportion de données synthétiques peut dégrader les performances finales, conformément à notre analyse théorique.

Une propriété remarquable des LLM est leur capacité à fournir des estimations d'incertitude bien calibrées. En extrayant la distribution de probabilité complète à partir des logits du modèle, nous observons que l'incertitude est plus élevée lorsque le contexte est limité ou près des zones de transition brusque. Les tests de Kolmogorov-Smirnov confirment que les prédictions suivent fidèlement les intervalles de confiance théoriques, une propriété cruciale pour la robustesse des algorithmes de MBRL.

A.2.5 Conclusion

Ce travail démontre que les LLM pré-entraînés peuvent servir de modèles de dynamique efficaces et "zero-shot" pour le RL continu, à condition de traiter correctement l'interdépendance des données via le désentrelacement. Bien que la PCA offre un compromis idéal entre simplicité et performance, l'exploration de transformations non linéaires (comme les auto-encodeurs) constitue une perspective prometteuse. De plus, l'intégration de contextes textuels pourrait encore améliorer la précision des prédictions. Néanmoins, le coût computationnel des appels répétés au LLM reste un verrou technologique à lever pour une application à grande échelle.

A.3 Résumé du chapitre 5

Ce chapitre est dédié à l'extension des modèles de fondation (FMs) de séries temporelles, initialement conçus pour des contextes univariés, vers des applications multivariées et probabilistes. Les travaux présentés ici introduisent le framework **AdaPTS** (du papier: Adapting Univariate Foundation Models to Probabilistic Multivariate Time Series Forecasting).

A.3.1 Introduction et Motivation

La prévision des séries temporelles est un pilier de l'apprentissage automatique, essentiel pour anticiper les tendances futures à partir de motifs historiques. Cependant, deux défis majeurs persistent dans les applications réelles : la nature souvent multivariée des données, impliquant des dépendances complexes entre plusieurs caractéristiques, et la nécessité d'estimer l'incertitude des prévisions pour une gestion des risques efficace. Bien que des modèles de fondation pré-entraînés tels que Chronos [Ans+24; Ans+25], Moirai [Liu+24d] ou Moment [Gos+24] aient démontré des performances exceptionnelles, ils sont majoritairement optimisés pour des tâches univariées pour des raisons de tractabilité computationnelle. Le traitement direct des séries multivariées reste coûteux en raison de la complexité quadratique liée au nombre de variables. AdaPTS propose de combler cette lacune en introduisant des adapteurs : des transformations de l'espace des caractéristiques qui projettent les entrées multivariées dans un espace latent où le modèle de fondation peut être appliqué indépendamment sur chaque dimension.

A.3.2 Cadre Théorique et Formulation

Définition de l'adapteur. Considérons une tâche de prévision multivariée à long terme définie par une matrice de données $X \in \mathbb{R}^{L \times D}$ (où L est la fenêtre contextuelle et D la dimensionnalité) et une matrice cible $Y \in \mathbb{R}^{H \times D}$ (H étant l'horizon de prévision). L'objectif est d'exploiter un modèle de fondation univarié gelé $f_{FM} : \mathbb{R}^{L \times 1} \to \mathbb{R}^{H \times 1}$. Un adapteur est défini comme une transformation $\varphi : \mathbb{R}^{D} \to \mathbb{R}^{D'}$ appliquée avant le modèle. La prévision finale est obtenue par la transformation inverse:

$$\tilde{Y}(X; \varphi) = \varphi^{-1}(f_{FM}(\varphi(X)))$$

Pour plus de flexibilité, nous relâchons la condition d'inversibilité stricte en utilisant un encodeur (enc) et un décodeur (dec), tels que $\hat{Y} = dec(f_{FM}(enc(X)))$.

Afin de démontrer l'optimalité de cette approche, une analyse théorique a été menée sur le cas des adapteurs et modèles de fondation linéaires. Sous certaines hypothèses de rang et de paramétrisation, la solution fermée pour l'adapteur linéaire optimal W_{φ}^{*} est donnée par:

$$W_{\varphi}^{*} = (B^{\top} A)^{+} B^{\top} B$$

où $A = Y - W_{FM}^{\top} X$ et $B = b_{FM} 1^{\top}$. Les expérimentations sur données synthétiques confirment que cet adapteur optimal surpasse significativement l'identité (traitement indépendant des canaux) et les méthodes basées sur l'Analyse en Composantes Principales (PCA) seule, particulièrement lorsque les données présentent de fortes corrélations.

A.3.3 Familles d'Adapteurs et Approche Probabiliste

AdaPTS intègre plusieurs architectures d'adapteurs, allant des structures déterministes aux modèles bayésiens.

Auto-encodeurs Déterministes:

- **Auto-encodeurs Linéaires (LinearAE) :** Ils compressent la série multivariée dans une représentation de dimension potentiellement réduite $D' \leq D$.
- **Auto-encodeurs Profonds non-linéaires :** Ils utilisent des réseaux de neurones profonds pour capturer des relations spatiales complexes avant l'application du FM.

Adapteurs Probabilistes:

L'innovation majeure de ce chapitre réside dans le traitement bayésien des paramètres des adapteurs pour permettre une quantification de l'incertitude, même avec des modèles de fondation déterministes comme Moment. L'utilisation d' **Auto-encodeurs Variationnels (VAE)** permet de considérer la représentation latente Z comme stochastique. L'objectif d'entraînement repose sur une borne inférieure de l'évidence (ELBO):

$$\log p_\theta(Y|X, f_{FM}) \geq \mathbb{E}_{q_\phi(Z|X)}[\log p_\theta(Y|X, f_{FM}(Z))] - KL(q_\phi(Z|X)||p(Z))$$

où le premier terme correspond à la perte de prévision (généralement l'erreur quadratique moyenne) et le second est un terme de régularisation via la divergence de Kullback-Leibler.

Une alternative plus légère consiste à utiliser le *Dropout de Monte Carlo*, où la stochasticité est introduite par l'application du dropout durant l'inférence, transformant ainsi des modèles déterministes en estimateurs probabilistes.

A.3.4 Résultats Expérimentaux et Analyse

Les performances d'AdaPTS ont été évaluées sur quatre jeux de données réels : ETTh1, ExchangeRate, Weather et Illness. Les résultats démontrent qu'AdaPTS améliore la précision de prévision du modèle Moment dans la majorité des configurations testées. Par exemple :

- Sur le jeu de données ETTh1 ($H = 96$), l'adapteur *dropout LinearAE* réduit l'erreur quadratique moyenne (MSE) de 8% par rapport au modèle de base.
- Sur le jeu de données Illness ($H = 24$), l'adapteur VAE permet une amélioration substantielle de 15%, abaissant la MSE de 2,902 à 2,461.

Réduction de Dimensionnalité. AdaPTS s'avère particulièrement efficace pour les configurations à ressources limitées. L'analyse de l'impact du nombre de composantes latentes montre que le cadre peut maintenir, voire surpasser, les performances de base avec un nombre de variables réduit. Sur le jeu de données Illness, l'adapteur VAE atteint des performances optimales avec seulement 2 composantes latentes, contre 7 à 14 pour les autres méthodes.

Robustesse face au Décalage de Distribution. L'étude de l'espace latent révèle que les adapteurs probabilistes, en particulier le VAE, forcent une représentation isotrope et structurée. Cela permet de mitiger les décalages de distribution entre les données d'entraînement et de test, améliorant ainsi la généralisation du modèle de fondation sans nécessiter de réglage fin de ses paramètres gelés.

Calibration de l'Incertitude. L'évaluation de la calibration via des diagrammes de fiabilité montre que les prévisions sont raisonnablement bien calibrées pour les horizons courts. Cependant, on observe une tendance à la sous-estimation de l'incertitude (surconfiance) à mesure que l'horizon de prédiction augmente.

A.3.5 Conclusion

En résumé, ce travail établit AdaPTS comme une solution modulaire et scalable pour adapter les modèles de fondation univariés aux contextes multivariés. En préservant les paramètres du FM gelés et en n'entraînant que des transformations légères de l'espace des caractéristiques, nous parvenons à améliorer la précision, quantifier l'incertitude et réduire les coûts d'inférence. Les perspectives futures incluent l'application de ce cadre à d'autres types de modèles déterministes et l'exploration de méthodes d'inférence plus complexes, comme les chaînes de Markov Monte Carlo, pour affiner davantage l'estimation de l'incertitude.

A.4 Résumé du chapitre 6

Ce chapitre explore une problématique centrale dans l'adaptation des grands modèles de langage (LLM) : la réconciliation de l'apprentissage supervisé (SFT) et de l'apprentissage par renforcement (RL). Bien que le RL offre des avantages significatifs en termes de généralisation et de réduction de l'oubli catastrophique, son application est souvent limitée par la nécessité de signaux de récompense explicites, souvent coûteux ou indisponibles. Nous présentons ici un cadre méthodologique permettant de dériver des

fonctions de récompense directement à partir de données supervisées, rendant ainsi le RL applicable sans annotation supplémentaire.

A.4.1 Motivation et Problématique

La méthode traditionnelle pour entraîner des modèles génératifs repose sur l'estimation de maximum de vraisemblance (MLE). Dans le contexte des LLM, cela se traduit par le paradigme de la prédiction du prochain jeton (next-token prediction). Cependant, le MLE présente des limites intrinsèques, notamment l'exposition au biais (exposure bias) et une tendance à l'oubli catastrophique lors de l'adaptation à de nouvelles tâches. L'apprentissage par renforcement, particulièrement avec des algorithmes comme le Reinforcement Learning from Human Feedback (RLHF), s'est avéré plus efficace pour aligner les modèles avec les préférences humaines. L'avantage du RL réside dans sa capacité à explorer des chemins de raisonnement que le SFT ne peut atteindre et à converger vers des solutions plus proches de la politique originale en termes de divergence KL. Le défi majeur reste l'absence de fonctions de récompense pour la majorité des jeux de données supervisés.

A.4.2 Cadre Théorique : L'Optimisation Bilevel

L'originalité de notre approche consiste à traiter la fonction de récompense r comme une variable d'optimisation au sein d'un problème d'optimisation bilevel (Bi-O).

Formulation Mathématique. Le problème est structuré comme suit:

- **Niveau extérieur (Outer level) :** Optimiser les paramètres de la récompense pour maximiser la log-vraisemblance sur les données d'entraînement.
- **Niveau intérieur (Inner level) :** Entraîner une politique θ via un objectif de RL régularisé par l'entropie:

$$\theta^* = \arg\max_{\theta \in \Theta} \mathbb{E}_X \mathbb{E}_{Y|X\sim q}[\mathbb{E}_{\hat{Y}|X\sim\hat{p}_\theta}[r(\hat{Y}, Y)] + \lambda H(\hat{p}_\theta)]$$

Paramétrisation de la Récompense. Nous définissons la récompense comme une forme quadratique négative de la différence entre les échantillons générés $\hat{y}$ et les cibles réelles y: $r_U(\hat{y}, y) = -(\hat{y}-y)^T U(\hat{y}-y)$. Pour les données complexes comme le texte, cette mesure est transposée dans un espace d'encastrement (*embedding space*) pré-entraîné: $r_U(\hat{y}, y) = -(e(\hat{y}) - e(y))^T U(e(\hat{y}) - e(y))$.

A.4.3 Analyse du Cas Gaussien

L'analyse théorique dans un cadre Gaussien révèle que la solution optimale pour la matrice U est inversement proportionnelle à la matrice de covariance Σ du processus de génération de données:

$$U^* = \frac{\lambda}{2}\Sigma^{-1}$$

Cette découverte permet d'interpréter la récompense optimale comme une distance de Mahalanobis négative. Pratiquement, cela signifie que plus les données sont bruitées, moins le modèle doit être pénalisé pour ses déviations par rapport aux échantillons. De plus, nous démontrons qu'en utilisant cette récompense optimale, l'objectif de RL devient équivalent à la minimisation de la divergence de Kullback-Leibler (KL) inversée entre la distribution du modèle et celle des données. Cela induit un comportement de recherche de mode (mode-seeking), contrairement au MLE qui tend à couvrir toute la masse de la distribution.

A.4.4 Résultats Expérimentaux

Les performances de notre cadre ont été évaluées sur diverses tâches, allant des données synthétiques à l'ajustement fin de LLM.

Atténuation de l'Oubli Catastrophique. Sur la tâche ParityMNIST (combinant reconnaissance de chiffres et types de vêtements), nos résultats montrent que le RL utilisant des récompenses apprises réduit l'oubli catastrophique de manière comparable à un RL utilisant une récompense oracle (parité réelle). Alors que le SFT voit ses performances chuter sur les tâches non ciblées par le fine-tuning, notre approche préserve mieux les connaissances antérieures.

Fine-tuning de LLM (Tâche Countdown) Nous avons appliqué l'algorithme Group Relative Policy Optimization (GRPO) sur des modèles Qwen3 (de 0.6B à 8B paramètres) pour résoudre des problèmes arithmétiques.

- **Performance :** Les récompenses apprises via notre formulation d'encastrement atteignent des niveaux de précision identiques à ceux de la récompense oracle (qui vérifie mathématiquement le résultat).
- **Robustesse :** Ces performances sont stables quelle que soit la taille du modèle.

- **Importance des Encastrements :** L'étude d'ablation montre que le choix du modèle d'encastrement (ex: RoBERTa ou Gemma) est crucial, et que le calcul de la récompense sur la réponse finale seule est plus efficace que sur l'intégralité de la sortie incluant la trace de raisonnement.

Données Tabulaires. Classification: Sur les jeux de données Credit Default et Poker, notre méthode $PG(U^*_{he})$ surpasse le MLE et le PG standard en termes d'AUC et de précision. Apprentissage par Renforcement Basé sur Modèle (MBRL) : Dans les tâches HalfCheetah, notre approche atteint le meilleur score en termes d'erreur quadratique moyenne (MSE) tout en restant très compétitive sur la log-vraisemblance négative (NLL), offrant un compromis idéal pour la planification déterministe.

A.4.5 Conclusion et Perspectives

Ce travail démontre qu'il est possible de transformer des signaux de données supervisées en fonctions de récompense structurées pour le RL. En interprétant la récompense comme une distance de Mahalanobis apprise via une optimisation bilevel, nous offrons une alternative robuste au SFT traditionnel.

Limitations et perspectives : La paramétrisation actuelle reste relativement simple ; l'extension à des fonctions de récompense plus complexes et expressives est envisagée. L'application à des tâches non vérifiables (où aucune récompense oracle n'existe) constitue la prochaine étape naturelle pour valider la généralisation de ce cadre dans des scénarios de production réels.

A.5 Conclusion globale

Cette thèse explore l'apprentissage par renforcement basé sur un modèle (MBRL) en l'intégrant aux modèles de fondation modernes, tels que les LLM et les modèles pour séries temporelles. À travers une approche multidisciplinaire mêlant apprentissage supervisé et optimisation, l'étude propose des contributions méthodologiques qui améliorent la modélisation de la dynamique des environnements. En s'appuyant sur des analyses théoriques rigoureuses validées par des résultats empiriques à grande échelle, ce travail démontre que des choix de modélisation fondés sur des principes solides sont essentiels pour faire progresser la recherche en intelligence artificielle.

Plus largement, ces travaux présentent la modélisation de la dynamique comme une interface clé où l'apprentissage par renforcement et les modèles de fondation s'enrichissent

mutuellement : l'un offre des cadres de décision complexes, tandis que l'autre apporte des architectures extensibles et performantes. En mettant à disposition des implémentations open-source pour chaque chapitre, cette thèse vise à garantir la reproductibilité des résultats et à stimuler le développement de systèmes de prise de décision plus robustes, généralisables et économes en données.

Appendix B

Appendix of Chapter 3

B.1 The evaluation setup

B.1.1 Environments

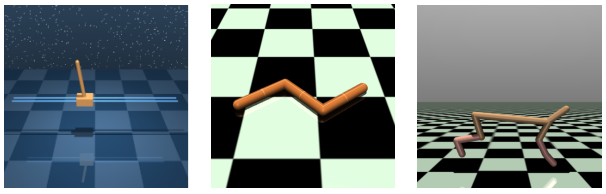


Figure B.1: The environments: Cartpole swing-up, Swimmer and Halfcheetah.

In the present study, we examine three distinct environments within the scope of continuous control reinforcement learning, as delineated in Figure B.1, each exhibiting varying degrees of complexity. The complexity of a given environment is primarily determined by the dimension of the state space d_s, and the dimension of the action space d_a. Notably, *Cartpole swing-up* is a classic problem in the field where the task is to swing up a pole starting downwards, and balance it upright. The other considered tasks are *Swimmer* and *Halfcheetah*, which are two locomotion tasks, aiming at maximizing the velocity of a virtual robot along a given axis. *Swimmer* incorporates fluid dynamics with the goal of learning an agent that controls a multi-jointed snake moving through water. On the other hand, *Halfcheetah* simulates a two-leg Cheetah with the goal of making it run as fast as possible. For the latter environments, we use the implementation of OpenAI Gym [Bro+16], and the implementation of Deepmind Control [Tas+18] for *Cartpole*. Both libraries are based on the Mujoco physics simulator [TET12]. A detailed description of these environments is provided in Table B.1.

Table B.1: The environments characteristics. d_s: the dimension of the state space, d_a: the dimension of the action space, x_t: position along the x-axis, $\dot{x}_t$: velocity along the x-axis, $\|a_t\|_2^2$: the action a_t magnitude, θ: the angle of the pole (only for Cartpole).

environment	d_s	d_a	task horizon	reward function
Cartpole swing-up	5	1	1000	$\frac{1+\cos\theta_t}{2} \times \frac{1+e^{-0.25\log(10)x_t^2}}{2} \times \left(1-\frac{a_t^2}{5}\right) \times \frac{1+e^{-0.04\log(10)\dot{x}_t{}^2}}{2}$
Swimmer	8	2	1000	$\dot{x}_t - 0.0001 \times \|a_t\|_2^2$
Halfcheetah	17	6	1000	$\dot{x}_t - 0.1 \times \|a_t\|_2^2$

B.1.2 Datasets

In this section, we introduce the different datasets that are used to evaluate the multi-step models. These datasets are collected using some *behavior policies* that are unknown to the models.

Table B.2 illustrates the features of datasets across the three environments: Cartpole Swing-up, Swimmer, and Halfcheetah. These environments vary in dataset size and behavioral policies. In the Cartpole Swing-up setting, each of the three datasets (*random*, *mixed_replay*, and *full_replay*) includes 50 episodes, which are split into training, validation, and testing subsets. The and *full_replay* depict complete learning trajectories of an unstable model and a state-of-the-art (sota) model-based Soft Actor-Critic (SAC) respectively [Jan+19], and integrated with shooting-based planning Section B.1.3. For the Swimmer environment, both the random and *full_replay* datasets consist of 50 episodes each. The random dataset is derived from a random policy, whereas the *full_replay* dataset is generated using a model-based SAC with planning. For both the Cartpole Swing-up and Swimmer environments, the datasets were self-collected due to the absence of a unified benchmark that includes datasets from both tasks.

Table B.2: The datasets characteristics. *mf*: model-free, *mb*: model-based, random $\rightarrow \pi$: all episodes collected to learn the policy π. The datasets size is given in episodes (of 1000 steps each).

environment	dataset	size (train/valid/test)	behavior policy
Cartpole swing-up	random	50 (36/4/10)	random policy
	mixed_replay	50 (36/4/10)	random $\rightarrow$ unstable *mb* SAC + planning
	full_replay	50 (36/4/10)	random $\rightarrow$ *mb* SAC + planning
Swimmer	random	50 (36/4/10)	random policy
	mixed_replay	50 (36/4/10)	random $\rightarrow$ unstable *mb* SAC + planning
Halfcheetah	random (*D4RL*)	100 (76/4/20)	random policy
	medium (*D4RL*)	100 (76/4/20)	*mf* sac at half convergence
	medium_replay (*D4RL*)	200 (156/4/40)	random $\rightarrow$ *mf* sac at half convergence

To enhance our understanding of the differences among these datasets, we present the distribution of returns for each dataset in Figure B.2. It is important to note that the variance in returns within a dataset serves as an indicator of the extent of the state space

covered by that dataset. Specifically, datasets collected using a fixed policy exhibit a notably narrow distribution, predominantly concentrated around their mean values, as exemplified by the Halfcheetah *random* and *medium* datasets. This characteristic of the datasets significantly influences the out-of-distribution generalization error in offline MBRL, which represents a major challenge in this context.

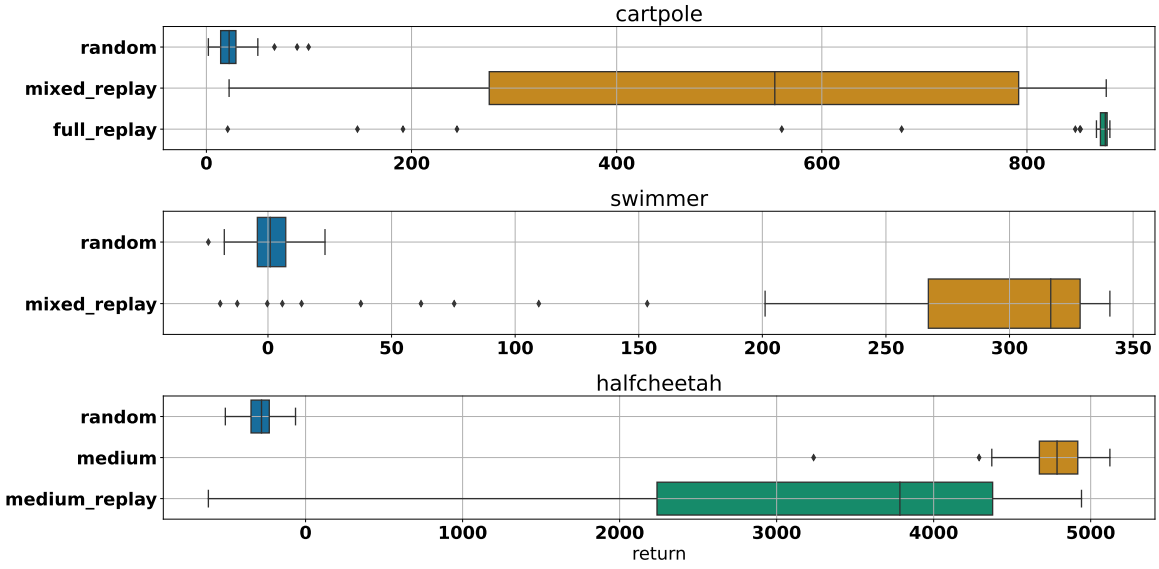


Figure B.2: A comparison of the distribution of returns across the considered datasets.

B.1.3 Agent: SAC + planning

Soft Actor-Critic (SAC) [Haa+18b] is an off-policy algorithm that incorporates the maximum entropy framework, which encourages exploration by seeking to maximize the entropy of the policy in addition to the expected return. SAC uses a deep neural network to approximate the policy (actor) and the value functions (critics), employing two Q-value functions to mitigate positive bias in the policy improvement step typical of off-policy algorithms. This approach helps in learning more stable and effective policies for complex environments, making SAC particularly suitable for tasks with high-dimensional, continuous action spaces.

In addition to *Dyna*-style training of the SAC agent on the learned model with short rollouts a la MBPO [Jan+19], we use Model Predictive Control (MPC). MPC is the process of using the model recursively to plan and select the action sequence that maximizes the expected cumulative reward over a planning horizon H. The set of K (population size) action sequences $\{(a^k_{t:t+H})_{k\in\{1,\dots,K\}}\}$ is usually generated by an evolutionary algorithm, e.g. Cross Entropy Method (CEM) [Chu+18]. In this study, the pre-trained SAC guides

the MPC process by generating candidate action sequences from the learned stochastic policy.

B.2 Implementation details

For all the models, we use a neural network composed of a common number of hidden layers and two output heads (with *Tanh* activation functions) for the mean and standard deviation of the learned probabilistic dynamics (The standard deviation is fixed when we want to use the MSE loss). We use batch normalization [IS15], Dropout layers [Sri+14b] ($p = 10\%$), and set the learning rate of the Adam optimizer [KB15] to 0.001, the batch size to 64, the number of common layers to 2, and the number of hidden units to 256 based on a hyperparameter search executed using the RAMP framework [Kég+18]. The evaluation metric of the hyperparameter optimi zation is the aggregated one-step validation R2 score across all the offline datasets. The neural networks are trained to predict the difference between the next state and the current state $\Delta_{t+1} = s_{t+1} - s_t$. More precisely, the baseline consists in the single-step model trained to predict the difference Δ_{t+1} using the one-step MSE. The other multi-step models, take their own predictions as input to predict the difference $\Delta_{t+h} = s_{t+h} - \hat{s}_{t+h-1}$ at horizon h

For the offline RL experiments, we use SAC agents from the StableBaselines3 open-source library [Raf+21] while keeping its default hyperparameters. In the offline setting, we train the SAC agents for $500,000$ steps on a fixed model by generating short rollouts of length 100 from states of the the dataset selected uniformly at random. At evaluation time, the MPC planning is done by sampling 500 action sequences from the SAC policy, and rolling out short rollouts of horizon 20 for return computation. This return is then bootstrapped with the value function learned by SAC.

B.3 Probabilistic interpretation

So far we have only considered deterministic models that directly learn to predict the next observation. However, it is very common in the literature to use *probabilistic* models that learn to predict the parameters of a *dist ribution* over the next observations [Chu+18; Jan+19; Nag+18; KHT21], rather than a point estimate. Probabilistic models are useful because they enable uncertainty estimation, which can represent finite-sample fitting errors (epistemic uncertainty) and/or the intrinsic uncertainty in the environment dynamics (aleatory uncertainty).

In the probabilistic case, we represent $\hat{p}_\theta$ as a Gaussian over the next state

$$\hat{p}_\theta\Big(s_{t+1}|s_t, a_t\Big) = \mathcal{N}\Big(s_{t+1}|\hat{\mu}_\theta(s_t, a_t), \hat{\sigma}_\theta(s_t, a_t)\Big),$$

where the parameters $(\hat{\mu}_\theta, \hat{\sigma}_\theta)$ are the output of the neural network. The loss function to train the probabilistic model on a transition[1] (s_t, a_t, s_{t+1}) is the negative log-likelihood $L\Big(s_{t+1}, \hat{p}_\theta(\cdot|s_t, a_t)\Big) = -\log \hat{p}_\theta(s_{t+1}|s_t, a_t)$.

Using this formulation, the jth-horizon negative log-likelihood loss is

$$\begin{aligned} L\Big(s_{t+j}, \hat{p}^j_\theta(s_t, \boldsymbol{a}_{t:t+j})\Big) &= -\log \hat{p}^j_\theta(s_{t+j}|s_t, \boldsymbol{a}_{t:t+j}) \\ &= -\log \mathcal{N}\Big(s_{t+j}|\hat{\mu}^j_\theta(s_t, \boldsymbol{a}_{t:t+j}), \hat{\sigma}^j_\theta(s_t, \boldsymbol{a}_{t:t+j})\Big). \end{aligned}$$

Instead of the direct next prediction, the parameters of the distribution $\hat{\mu}^j_\theta(s_t, \boldsymbol{a}_{t:t+j})$ and $\hat{\sigma}^j_\theta(s_t, \boldsymbol{a}_{t:t+j})$ represent the output of the model after j recursive calls using the ground truth actions $\boldsymbol{a}_{t:t+j}$. From this definition, we can obtain the weighted multi-step MSE loss by solving a Maximum (joint) Likelihood Estimation (MLE) problem:

Proposition B.3.1. (*Multi-step MSE loss as MLE*). *Under the assumption of the conditional independence on $(s_t, \boldsymbol{a}_{t:t+h})$, the weighted multi-step MSE loss can be recovered from the negative log-likelihood of the joint distribution of $(s_{t+1}, \dots, s_{t+h}) = \boldsymbol{s}_{t+1:t+h}$:*

$$\begin{aligned} L\Big(\boldsymbol{s}_{t+1:t+h}, \hat{p}^{1:h}_\theta(s_t, \boldsymbol{a}_{t:t+h})\Big) &= -\log \hat{p}^{1:h}_\theta(\boldsymbol{s}_{t+1:t+h}|s_t, \boldsymbol{a}_{t:t+h}) \\ &= -\log \Pi_{j=1}^h \hat{p}^j_\theta(s_{t+j}|s_t, \boldsymbol{a}_{t:t+j}) \\ &= \underbrace{\sum_{j=1}^h \log \hat{\sigma}^j_\theta}_{\text{regularization}} + \underbrace{\sum_{j=1}^h \underbrace{\frac{1}{2(\hat{\sigma}^j_\theta)^2}}_{\alpha_j}(s_{t+j} - \hat{\mu}^j_\theta)^2}_{L_{\boldsymbol{\alpha}}} + \underbrace{C}_{constant} \end{aligned}$$

with $\hat{\mu}^j_\theta = \hat{\mu}^j_\theta(s_t, \boldsymbol{a}_{t:t+j})$ and $\hat{\sigma}^j_\theta = \hat{\sigma}^j_\theta(s_t, \boldsymbol{a}_{t:t+j})$.

Remark B.3.1. *proposition B.3.1 states the result in the case of uni-dimensional state spaces. While this is considered for simplicity, we can straight-forwardly generalize to the multi-variate case with the inverse of the covariance matrix as weight $\alpha_j = -\frac{1}{2}\hat{\Sigma}^{-1}_\theta$.*

Can we automatically learn the weights using the formulation in proposition B.3.1? To answer this question, we train a probabilistic model $\hat{p}_\theta$ to minimize the Negative Log-Likelihood (NLL) loss shown at proposition B.3.1. When computing the multi-step loss at training time, the predicted states are obtained by sampling from the learned distribution using the reparametrization trick $\hat{s}_{t+j} = \hat{\mu}^j_\theta(s_t, \boldsymbol{a}_{t:t+j}) + \hat{\sigma}^j_\theta(s_t, \boldsymbol{a}_{t:t+j}) \cdot \xi$ where $\xi \sim \mathcal{N}(0, \boldsymbol{I})$[2]. We refer to this as *stoch*-astic sampling, as opposed to *det*-erministic sampling that refers to MSE-based models where $\hat{s}_{t+j} = \hat{\mu}^j_\theta(s_t, \boldsymbol{a}_{t:t+j})$.

[1] In this section, we don't make the distinction between states s and observations o as the probabilistic

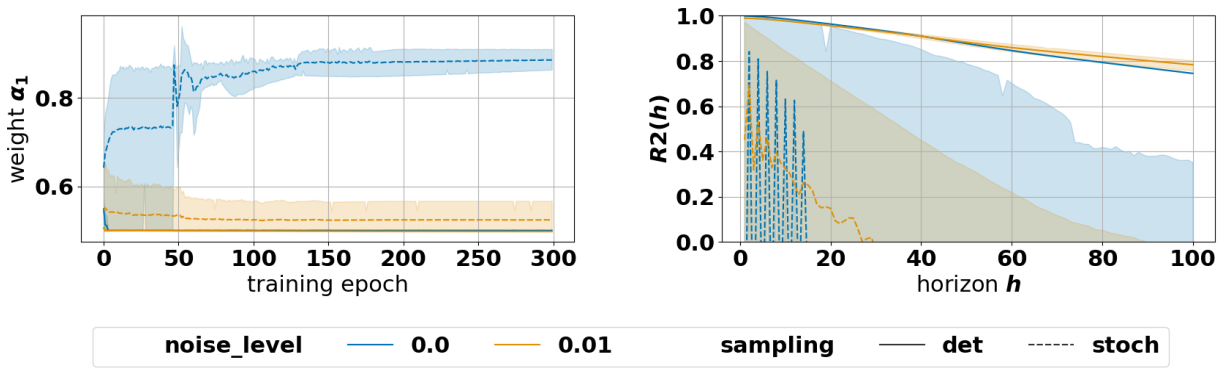


Figure B.3: The left panel shows the normalized learned weights of a stochastic model trained using the $h = 2$ multi-step loss. Basically, $\alpha_1 = \frac{\frac{1}{|\mathcal{D}_{train}|}\sum_{\mathcal{D}_{train}}\frac{1}{2(\hat{\sigma}^1_\theta)^2}}{\frac{1}{|\mathcal{D}_{train}|}\sum_{\mathcal{D}_{train}}\frac{1}{2(\hat{\sigma}^1_\theta)^2}+\frac{1}{|\mathcal{D}_{train}|}\sum_{\mathcal{D}_{train}}\frac{1}{2(\hat{\sigma}^2_\theta)^2}}$ and $\alpha_2 = 1 - \alpha_1$. The right panel shows the $R2(h)$ curves on a held-out test dataset from the *Cartpole swing-up random* task.

Interestingly, we find that the *stoch* model automatically learns to put more weight on $h = 1$ in absence of noise, while learning to balance the weights at 1% noise (Figure B.3). In the same dataset (*Cartpole-Random* with respectively 0% and 1% noise scale), we have found for the $h = 2$ MSE-based model that the optimal weights were obtained by respectively $\beta = 0.1$ ($\alpha_1 = 0.9$ and $\alpha_2 = 0.1$) and $\beta = 2.0$ ($\alpha_1 = 0.33$ and $\alpha_2 = 0.66$). This result shows the potential of the MLE formulation to learn the correct weight profile depending on the intrinsic noise on the training data, which is captured through the learned variance.

In practice, the *stoch* model fails to predict accurately at future horizons (Figure B.3), which is the reason why we don't adopt this formulation in the main paper. We suspect that the model is underestimating the variance, leading to inaccurate predictions during inference and resulting in divergent trajectories. This issue stems from inadequate uncertainty propagation during the generation of rollouts from the stochastic model. In fact, accurate uncertainty quantification over future horizons necessitates the generation of multiple trajectories and the subsequent measurement of variance at the j-th horizon. We leave the exploration of this promising direction to future work.

interpretation is independent of the underlying MDP.

[2]Notice that in this context, s is not necessarily uni-dimensional. In fact, $s \in \mathbb{R}^d$ where d is the dimension of the state space. $\boldsymbol{I}$ is thus, the corresponding identity matrix.

B.4 Additional experiments & results

B.4.1 Uni-dimensional linear system

Comparison with the data augmentation variant

To further understand the multi-step loss, we propose to reconduct the bias-variance analysis in Section 3.5.1 with a data augmentation variant that consists in training the $\alpha = 1.0$ model (baseline) on an augmented dataset: $\mathcal{D}_{aug} = \{\mathcal{S}_0, \mathcal{O}_1\}$, with the next states being $y = \{\mathcal{O}_1, \mathcal{O}_2\}$.

Indeed, one may argue that the variance reduction gained by using the $\alpha = 0.5$ multi-step loss is due to sampling the noise twice (both through $\mathcal{O}_1$, and $\mathcal{O}_2$), while the vanilla model only sees one realization of the noise through $\mathcal{O}_1$.

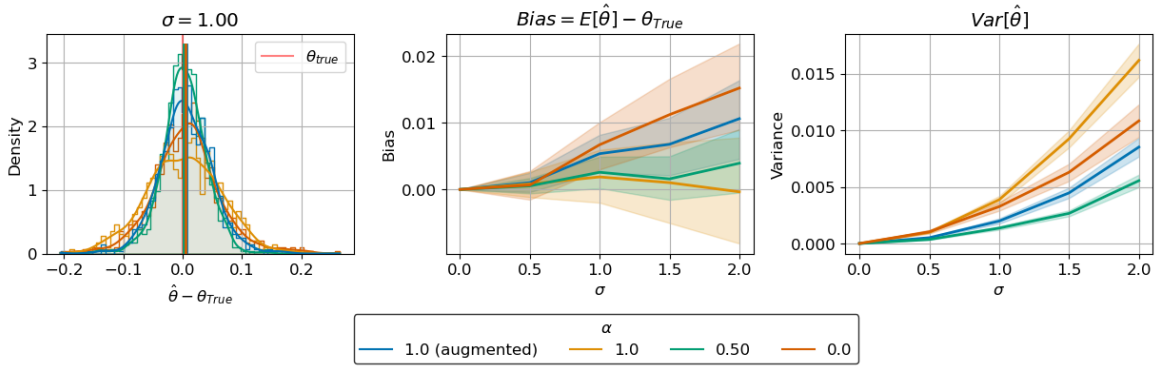


Figure B.4: The left panel shows the density distribution of $\hat{\theta} - \theta_{true}$ for a fixed σ of 1.0. The middle panel delineates the bias of the estimator, defined as $E[\hat{\theta}] - \theta_{true}$, across varying levels of σ, and weights $\alpha \in \{0, 0.5, 1\}$, in addition to the augmented variant of the $\alpha = 1.0$ model. The right panel presents the variance of the estimator, $Var[\hat{\theta}]$, as a function of σ for the same set of α values. The shaded regions represent the 95% bootstrap confidence intervals across 10 θ_{true} values, and 100 Monte-Carlo simulations.

As suggested by the previous reasoning, Figure B.4 shows that indeed augmenting the training data of the baseline model reduces its variance, yet the multi-step model with $\alpha = 0.5$ remains the optimal in terms of variance. Interestingly, we notice that the bias increases for the data-augmented model, this is due to the noise appearing in the inputs now, which changes the closed-form solutions derived in Section 3.5.1 and the estimator is no longer unbiased. We conclude on the optimality of the multi-step even in the data augmented case.

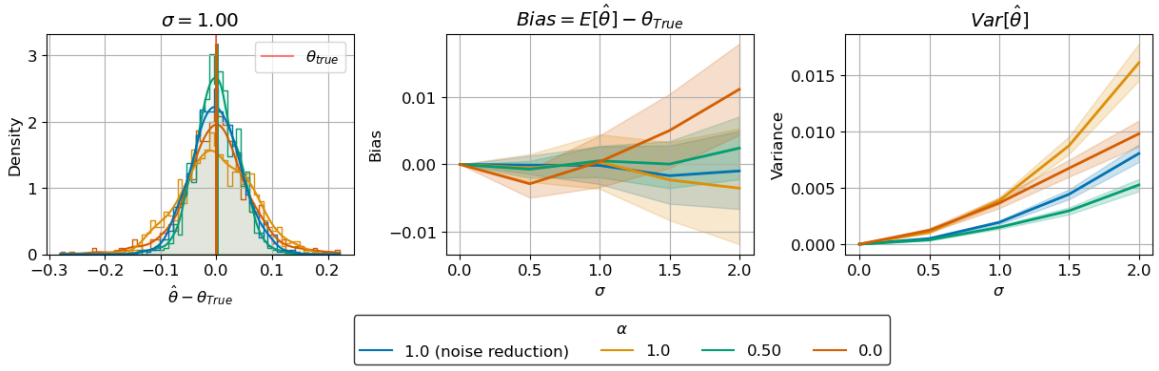


Figure B.5: Similar to Figure B.4, in addition to the noise reduction variant of the $\alpha = 1.0$ model.

Analogy to noise reduction by averaging

One way to increase the signal-to-noise ratio (SNR) is *averaging* [NG10; van07]. It consists in sampling the same noisy event k times, and observing a noise reduction in the average by a factor of $\sqrt{k}$. Although we have no guarantees about the factor of noise reduction observed when we use the multi-step loss, we hypothesize that the observed improvement is similar to that of noise reduction by averaging.

To test this hypothesis, we reconduct the bias-variance analysis in Section 3.5.1 and Section B.4.1 with a noise reduction variant that consists in training the $\alpha = 1.0$ model (baseline) on a modified dataset: $\mathcal{D}_{reduc} = \{\mathcal{S}_0\}$, with the next states being $y = \{\frac{\mathcal{O}_1 + \mathcal{O}'_1}{2}\}$, where $\mathcal{O}_1$ and $\mathcal{O}'_1$ are two realizations of the noisy next transitions. As illustrated in Figure B.5, the noise reduction approach applied to the $h = 1$ model demonstrated a significant reduction in variance while maintaining unbiasedness, a result that is supported by theoretical justification. Nonetheless, it is important to note that the multi-step model with $\alpha = 0.5$ continues to display a lower variance, albeit with a marginally increased bias.

B.4.2 Two-parameter Non-linear system

The loss landscape and approximate global minima

In the non-linear case, the problem of computing the minimizers of the loss function is intractable. To build an understanding about the optimization challenges in this setup, we tried to replicate the same analysis we did in the linear case, visualizing the loss landscape and its approximate global minima Figure B.6.

The global minima in Figure B.6 were determined by evaluating the loss function across

a finely meshed grid surrounding the true parameters θ^{true}. This process was carried out for ten distinct sets of future observations at varying noise levels. The resulting data include the average loss surface, individual instances of global minima (indicated by yellow dots) and their mean (shown as an orange star). In absence of noise, all global minima are aligned with θ^{true}, serving as a preliminary validation of the multi-step loss's consistency. However, as the noise level increases, the challenge of optimizing the two-step MSE loss becomes more and more apparent, evidenced by an increasing number of minimum points diverging from the true parameters. In contrast, the model with an intermediate value of $\alpha = 0.5$ either matched or outperformed the baseline model ($\alpha = 1$) in terms of the proximity of the global minima to the true parameters. Consequently, in this basic non-linear scenario, it is evident that the multi-step loss does not introduce significant optimization difficulties at lower α values.

Details of the experiment in Section 3.5.2

The goal of the experiment in Section 3.5.2 is to assess the solutions of the multi-step MSE loss after training. Indeed, the intractability of the optimization problem forces to only analyze the solution after a gradient-based optimization procedure, all while integrating counfounding factors of this latter. The confounding factors that influence the outcome of the optimization incude: the optimizer, the initial points distribution, the learning rate, among others.

In this study we generate 10 Monte-Carlo simulations for each of: 10 randomly sampled initial points, 3 initialization distributions (*default*, *uniform*, *Xavier_uniform*), 2 optimizers (*Adam*, *SGD*), 3 noise levels (0%, 20%, and 40%), 5 different values of the multi-step loss parameter $\alpha \in \{0, 0.25, 0.5, 0.75, 1\}$. For each of these experiments, we train the neural network to minimize the multi-step MSE loss L_α for 60 epochs. We report and analyze the validation losses and proximity to the true parameters in Figure 3.5, Section 3.5.2.

Figure B.7 shows the per-epoch training and validation loss curves, in addition to the MSE between the model parameters and the true parameters.

B.4.3 Static evaluation

$\overline{R2}(100)$ **table**

The static evaluation has been conducted by training models on the multi-step loss for different values of $h \in \{2, 3, 4, 10\}$ and $\beta \in \{0.1, 0.3, 0.5, 0.75, 1.0, 1.5, 2.0, 3.0, 5.0, 20.0\}$. Table B.3 shows the corresponding test $\overline{R2}(100)$ scores and their standard deviations after the symbol $\pm$. For each horizon h, we show the best value of the β parameter, as it's different across environments, datasets, and noise scales.

Environment	Dataset	Noise scale	One-step	Multi-step			
				h=2 [β]	h=3 [β]	h=4 [β]	h=10 [β]
Cartpole swing-up	random	0	972 +- 4	975 +- 1 [0.1]	977 +- 2 [0.3]	980 +- 1 [0.1]	**986 +- 1 [0.75]**
		0.01	864 +- 5	930 +- 1 [2.0]	**950 +- 2 [2.0]**	946 +- 3 [1.0]	**954 +- 4 [0.75]**
		0.02	508 +- 16	690 +- 15 [2.0]	769 +- 7 [2.0]	812 +- 7 [1.5]	**836 +- 7 [0.75]**
	mixed_replay	0	812 +- 106	915 +- 22 [0.75]	921 +- 32 [2.0]	925 +- 24 [0.5]	931 +- 20 [0.75]
		0.01	541 +- 51	574 +- 26 [0.1]	659 +- 44 [3.0]	653 +- 51 [1.5]	679 +- 38 [0.75]
		0.02	428 +- 9	335 +- 4 [2.0]	459 +- 71 [0.75]	481 +- 12 [1.0]	457 +- 38 [0.75]
	full_replay	0	705 +- 52	828 +- 6 [2.0]	843 +- 42 [2.0]	858 +- 32 [0.75]	**880 +- 21 [0.5]**
		0.01	714 +- 7	709 +- 10 [0.1]	718 +- 6 [1.5]	722 +- 33 [0.75]	683 +- 32 [0.1]
		0.02	530 +- 14	551 +- 34 [0.1]	527 +- 21 [1.5]	530 +- 46 [0.1]	541 +- 24 [0.1]
Swimmer	random	0	**983 +- 1**	974 +- 3 [0.1]	978 +- 0 [0.1]	975 +- 1 [0.1]	974 +- 2 [0.1]
		0.01	934 +- 6	941 +- 0 [0.1]	**941 +- 2 [0.1]**	**942 +- 1 [0.5]**	**943 +- 2 [0.5]**
		0.02	865 +- 14	869 +- 17 [0.1]	880 +- 8 [0.5]	**892 +- 4 [0.5]**	**891 +- 4 [0.5]**
	mixed_replay	0	609 +- 46	680 +- 184 [20.0]	734 +- 18 [20.0]	**875 +- 25 [3.0]**	735 +- 120 [0.75]
		0.01	874 +- 14	904 +- 16 [0.5]	901 +- 16 [3.0]	**936 +- 5 [0.1]**	920 +- 8 [0.75]
		0.02	850 +- 13	893 +- 8 [1.0]	878 +- 6 [0.3]	886 +- 4 [1.0]	883 +- 8 [0.75]
Halfcheetah	random	0	755 +- 3	746 +- 7 [0.1]	**775 +- 3 [0.1]**	765 +- 9 [0.5]	766 +- 5 [0.3]
		0.01	737 +- 4	732 +- 27 [0.1]	756 +- 9 [0.1]	**763 +- 6 [0.1]**	725 +- 5 [0.5]
		0.02	704 +- 3	711 +- 17 [0.1]	709 +- 23 [0.1]	701 +- 3 [0.5]	**731 +- 0** [0.3]
	medium	0	516 +- 19	268 +- 51 [0.1]	547 +- 10 [0.3]	562 +- 63 [0.3]	**640 +- 13 [0.3]**
		0.01	691 +- 11	634 +- 20 [0.1]	674 +- 28 [0.1]	695 +- 3 [0.1]	**710 +- 4 [0.1]**
		0.02	**718 +- 4**	422 +- 242 [0.1]	675 +- 15 [0.1]	690 +- 1 [0.1]	702 +- 13 [0.1]
	medium_replay	0	541 +- 54	446 +- 70 [3.0]	589 +- 15 [0.75]	632 +- 7 [0.75]	**771 +- 12 [0.5]**
		0.01	709 +- 12	725 +- 4 [0.1]	737 +- 7 [0.1]	737 +- 16 [0.1]	**773 +- 9 [0.3]**
		0.02	687 +- 26	723 +- 16 [0.1]	731 +- 6 [0.1]	765 +- 18 [0.3]	**750 +- 7 [0.3]**

Table B.3: $\overline{R2}(100)$ for different environments, datasets, and noise scales. We highlight entries that have significantly larger score. In addition, to the mean ^{p}m standard deviation of the reported metric, the table also shows the best β selected for each loss horizon h.

The weight profile and the effective horizon

We can define the effective horizon h_e that reflects the prediction horizon at which we effectively optimize the prediction error: given a weighted multi-step loss $L_{\boldsymbol{\alpha}}$ with nominal horizon h and weights $\boldsymbol{\alpha}$, the effective horizon h_e is defined as $h_e = \sum_{i=1}^{h} \alpha_i \cdot i$.

For a given model trained using the multi-step loss, the optimal effective horizon is an indication of the time scale needed for optimal performance. However, models that have a different nominal horizon h and the same effective horizon h_e do not necessarily have the same performance. Precisely, the loss with the larger nominal horizon is setting small weights on the furthest horizons, which has a direct impact on the loss landscape and consequently on the optimization process.

Table B.4 shows the optimal effective horizon h_e, and the exponentially parametrized weight profiles (characterized by the decay parameter β), for different values of the horizon h and the noise scale σ.

The main insight of Table B.4 is that regardless of the nominal horizon h, the effective horizon h_e (and equivalently the decay parameter β) increases with the noise scale. This finding supports the idea that multi-step models are increasingly needed when incorpo-

Table B.4: Best $h_e(\beta)$ values found with a grid search for each horizon and each noise scale. The values are averaged over the eight datasets.

	horizon h			
	2	3	4	10
σ (%)	h_e (β)			
0.0	1.30 (0.81)	1.48 (0.45)	1.58 (0.41)	2.09 (0.46)
0.01	1.26 (0.56)	1.65 (0.76)	1.74 (0.51)	2.23 (0.47)
0.02	1.36 (0.86)	1.65 (0.72)	2.08 (0.78)	2.72 (0.54)
0.03	1.33 (0.67)	1.95 (1.21)	2.13 (0.81)	2.33 (0.50)
0.04	1.40 (0.74)	1.88 (1.01)	2.02 (0.74)	2.49 (0.55)
0.05	1.49 (1.33)	1.76 (0.83)	2.28 (0.97)	2.22 (0.51)

rating information from the future is crucial to achieve noise reduction. As discussed in the previous experiment, the results are highly dependent on the task (environment/-dataset), while in Table B.4 we aggregate the results across tasks, and still observe the increasing trend.

Another important result highlighted in Table B.4 is the upper bound on the effective horizon (h_e does not go beyond 2.52), even when the nominal horizon is large (e.g 10). This suggests that while putting weight on future horizons error does help the model, it is not beneficial to fully optimize for these horizons. Indeed, the additional components of the multi-step MSE loss act as a regularizer to the one-step loss, rather than a completely different training objective.

$R2(h)$ curves

In this section, we show the full $R2(h)$ curves for all environments / datasets.

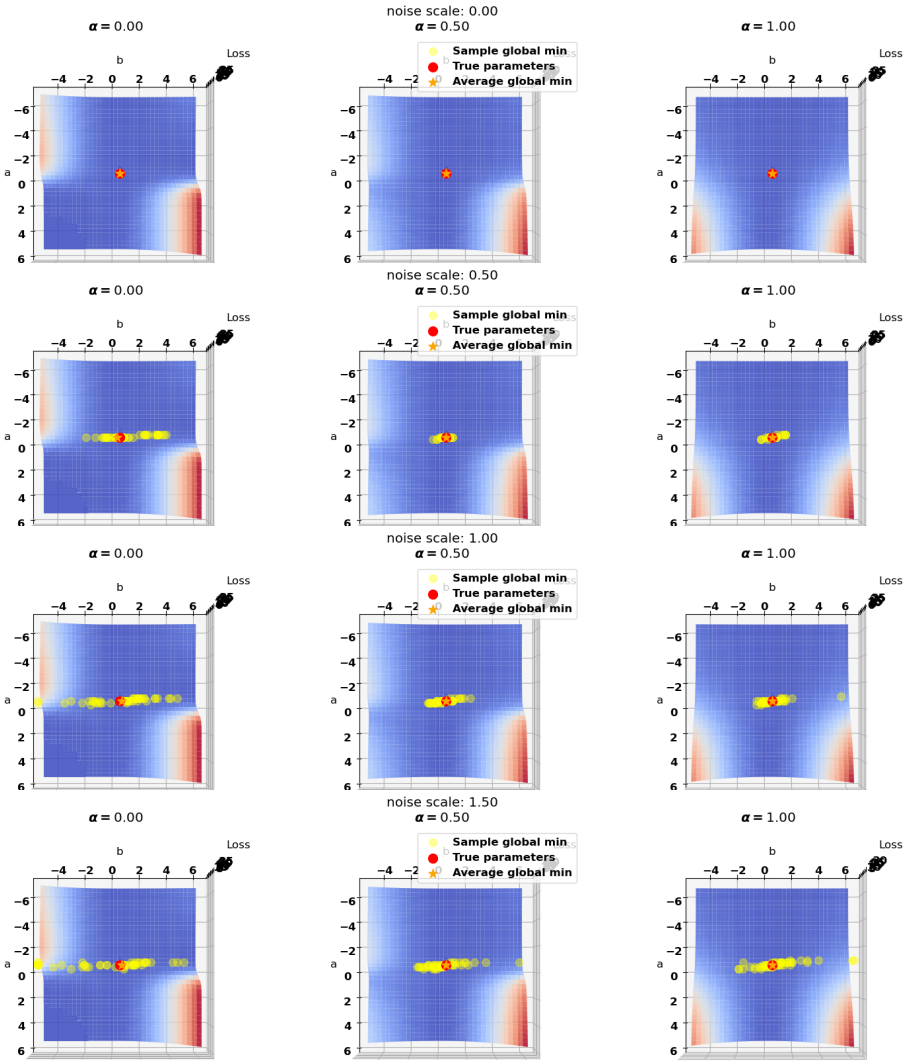


Figure B.6: This figure illustrates the impact of noise on the optimization of a multi-step MSE loss function by displaying loss landscapes at varying noise levels and alpha (α) values. The true parameters (θ^{true}) are marked with red dots. Each row represents a different level of noise, with yellow dots indicating individual instances of global minima and orange stars denoting their average.

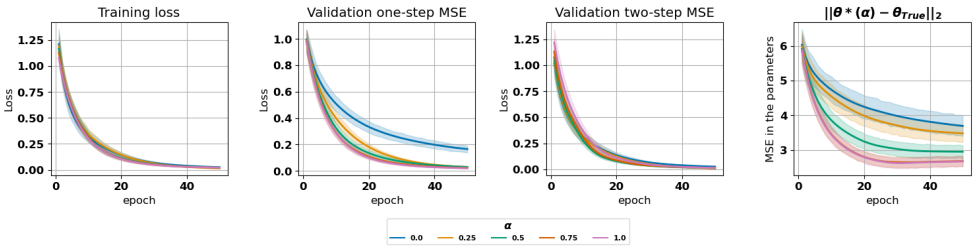


Figure B.7: The three left panels show the training, the one-step, and the two-step validation losses across 50 training epochs. The right panel show the evolution of the distance of the NN parameters to the system parameters during training.

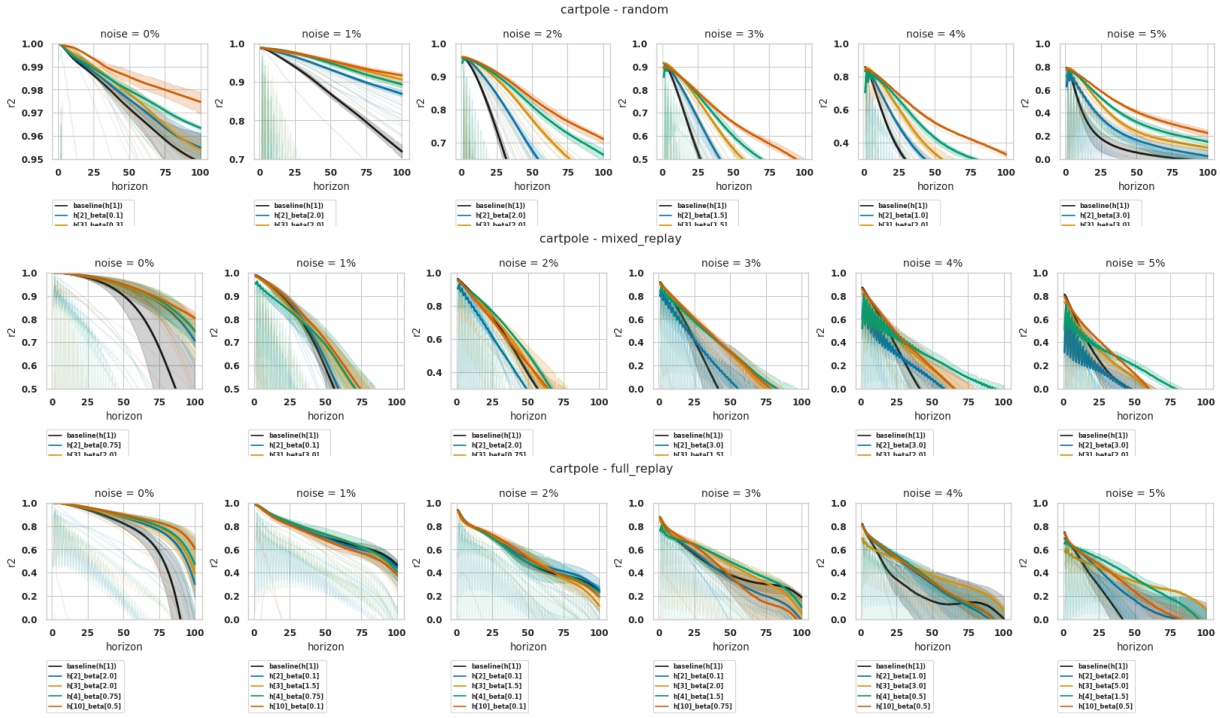


Figure B.8: Cartpole.

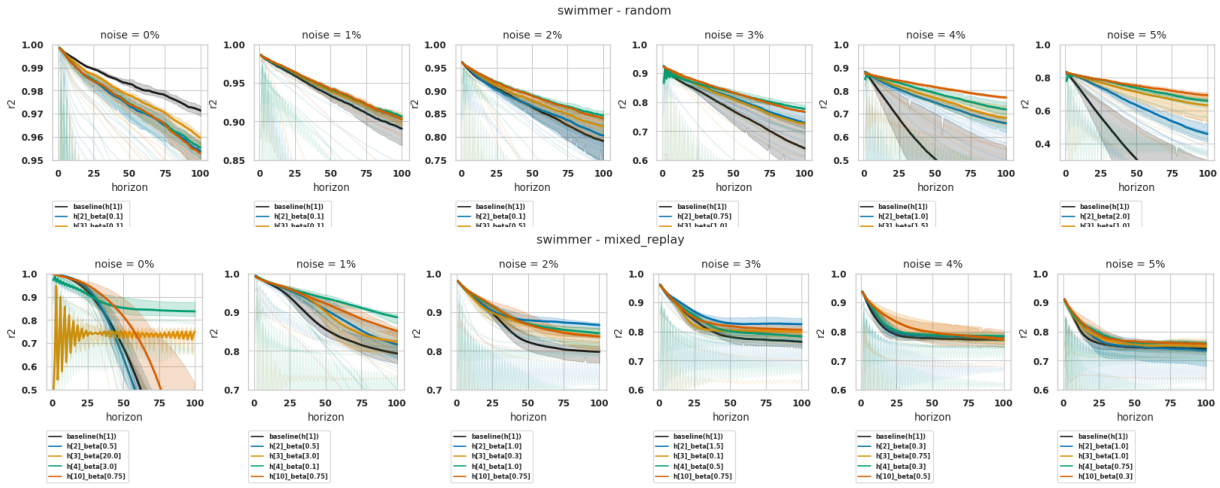


Figure B.9: Swimmer.

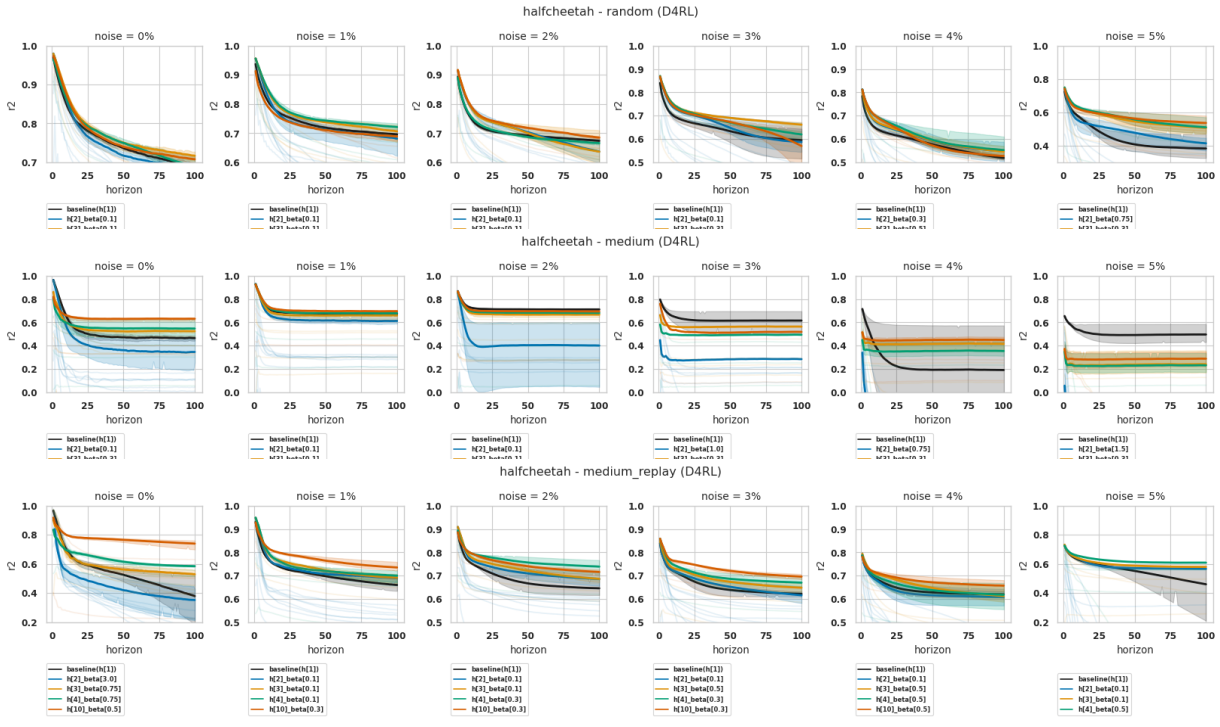


Figure B.10: Halfcheetah.

Appendix C

Appendix of Chapter 4

C.1 Theoretical analysis

C.1.1 Proof of theorem 4.4.1

We start by formally defining the LLM multi-branch return $\eta^{\text{llm}}_{p,k,T}$. To do so, we first denote A_t the random event of starting a k-step LLM branch at timestep t and we denote X_t the associated indicator random variable $X_t = \mathbb{1}[A_t]$. We assume that the $(X_t)_{t \geq T}$ are independent. We then define the random event A^k_t that at least one of the k preceding timesteps has been branched, meaning that the given timestep t belongs to at least one LLM branch among the k possible branches: $A^k_t = \bigcup_{i=0}^{k-1} A_{t-i}$. The LLM multi-branch return can then be written as follows:

$$\begin{aligned}
\eta^{\text{llm}}_{p,k,T}(\pi) = & \underbrace{\sum_{t=0}^{T-1} \gamma^t \mathbb{E}_{s_t \sim P^t, a_t \sim \pi}\Big[r(s_t, a_t)\Big]}_{\text{Burn-in phase to gather minimal context size } T} \\
& + \sum_{t=T}^{\infty} \gamma^t \mathbb{E}_{X_{t-i} \sim b(p), 1 \leq i \leq k} \Bigg[\mathbb{1}[A^k_t] \underbrace{\frac{1}{\sum_{i=1}^k X_{t-i}} \sum_{i=1}^{k} X_{t-i} \mathbb{E}_{s_t \sim \hat{P}^i_{t,\text{llm}}, a_t \sim \pi}\Big[r(s_t, a_t)\Big]}_{\text{average reward among the branches spanning timestep } t} \\
& + \underbrace{\mathbb{1}[\bar{A}^k_t] \mathbb{E}_{s_t \sim P^t, a_t \sim \pi}\Big[r(s_t, a_t)\Big]}_{\text{When no branch is spanning timestep } t} \Bigg],
\end{aligned} \tag{C.1}$$

where $P^t = P(.|P^{t-1})$ with $P^0 = \mu_0$ the initial state distribution and $\hat{P}^i_{t,\text{llm}} = \hat{P}^i_{\text{llm}}(.|P^{t-i})$. Before continuing, we first need to establish the following lemma.

Lemma C.1.1. (*Multi-step Error Bound, Lemma B.2 in [Fra+24] and [Jan+19].*) *Let P and $\tilde{P}$ be two transition functions. Define the multi-step error at time step t, starting from any initial state distribution μ_0, as:*

$$\varepsilon_t := D_{TV}(P^t(\cdot|\mu_0)\|\tilde{P}^t(\cdot|\mu_0))$$

with $P^0 = \tilde{P}^0 = \mu_0$.
Let the one-step error at time step $t \geq 1$ be defined as:

$$\xi_t := \mathbb{E}_{s\sim P^{t-1}(\cdot|\mu_0)}\left[D_{TV}(P(\cdot|s)\|\tilde{P}(\cdot|s))\right],$$

and $\xi_0 = \varepsilon_0 = 0$.

Then, the multi-step error satisfies the following bound:

$$\varepsilon_t \leq \sum_{i=0}^{t} \xi_i.$$

Proof. Let $t > 0$. We start with the definition of the total variation distance:

$$\begin{aligned}
\varepsilon_t &= D_{\text{TV}}(P^t(\cdot|\mu_0)\|\tilde{P}^t(\cdot|\mu_0)) \\
&= \frac{1}{2}\int_{s'\in\mathcal{S}} \left|P^t(s'|\mu_0) - \tilde{P}^t(s'|\mu_0)\right| ds' \\
&= \frac{1}{2}\int_{s'\in\mathcal{S}} \left|\int_{s\in\mathcal{S}} P(s'|s)P^{t-1}(s|\mu_0) - \tilde{P}(s'|s)\tilde{P}^{t-1}(s|\mu_0)\, ds\right| ds' \\
&\leq \frac{1}{2}\int_{s'\in\mathcal{S}}\int_{s\in\mathcal{S}} \left|P(s'|s)P^{t-1}(s|\mu_0) - \tilde{P}(s'|s)\tilde{P}^{t-1}(s|\mu_0)\right| ds\, ds' \\
&= \frac{1}{2}\int_{s'\in\mathcal{S}}\int_{s\in\mathcal{S}} \left|P(s'|s)P^{t-1}(s|\mu_0) - \tilde{P}(s'|s)\tilde{P}^{t-1}(s|\mu_0)\right| ds\, ds' \\
&= \frac{1}{2}\int_{s'\in\mathcal{S}}\int_{s\in\mathcal{S}} \Big|P(s'|s)P^{t-1}(s|\mu_0) - \tilde{P}(s'|s)P^{t-1}(s|\mu_0) \\
&\qquad + \tilde{P}(s'|s)P^{t-1}(s|\mu_0) - \tilde{P}(s'|s)\tilde{P}^{t-1}(s|\mu_0)\Big| ds\, ds'
\end{aligned}$$

Using the triangular inequality, we can split the above expression into two terms:

$$
\begin{aligned}
\varepsilon_t &\leq \frac{1}{2}\int_{s'\in\mathcal{S}}\int_{s\in\mathcal{S}} P^{t-1}(s|\mu_0)\left|P(s'|s)-\tilde{P}(s'|s)\right| ds\, ds' \\
&\quad + \frac{1}{2}\int_{s'\in\mathcal{S}}\int_{s\in\mathcal{S}} \tilde{P}(s'|s)\left|P^{t-1}(s|\mu_0)-\tilde{P}^{t-1}(s|\mu_0)\right| ds\, ds' \\
&= \int_{s\in\mathcal{S}}\left[\frac{1}{2}\int_{s'\in\mathcal{S}}\left|P(s'|s)-\tilde{P}(s'|s)\right| ds'\right] P^{t-1}(s|\mu_0)\, ds \\
&\quad + \frac{1}{2}\int_{s\in\mathcal{S}}\left(\int_{s'\in\mathcal{S}}\tilde{P}(s'|s)\, ds'\right)\left|P^{t-1}(s|\mu_0)-\tilde{P}^{t-1}(s|\mu_0)\right| ds \\
&= \mathbb{E}_{s\sim P^{t-1}(\cdot|\mu_0)}\left[D_{\mathrm{TV}}(P(\cdot|\mu_0)\|\tilde{P}(\cdot|s))\right] + D_{\mathrm{TV}}(P^{t-1}(\cdot|\mu_0)\|\tilde{P}^{t-1}(\cdot|\mu_0)) \\
&= \xi_t + \varepsilon_{t-1}
\end{aligned}
$$

Given that $\xi_0 = \varepsilon_0 = 0$, by induction we have:

$$
\varepsilon_t \leq \sum_{i=0}^{t} \xi_i.
$$

□

We now restate and prove theorem 4.4.1:

Theorem C.1.1 (Multi-branch return bound)**.** *Let T be the minimal length of the in-context trajectories, $p \in [0,1]$ the probability that a given state is a branching point. We assume that the expected total variation between the LLM-based model and the true dynamics under a policy π is bounded at each timestep by*

$$
\max_{t\geq T} \mathbb{E}_{\mathbf{s}\sim \mathrm{P}^t, \mathbf{a}\sim\pi}\Big[d_{\mathrm{TV}}\Big(\mathrm{P}(.|\mathbf{s},\mathbf{a})||\hat{\mathrm{P}}_{\mathit{llm}}(.|\mathbf{s},\mathbf{a})\Big)\Big] \leq \varepsilon_{\mathit{llm}}(T).
$$

Then under a multi-branched rollout scheme with a branch length of k, the return is bounded as follows:

$$
|\eta(\pi) - \eta^{\mathit{llm}}_{p,k,T}(\pi)| \leq 2\frac{\gamma^T}{1-\gamma} r_{\max} k^2\, p\, \varepsilon_{\mathit{llm}}(T) \quad , \tag{C.2}
$$

where $r_{\max} = \max_{s\in\mathcal{S}, a\in\mathcal{A}} r(s,a)$.

Proof. **Step 1: Expressing the bound in terms of horizon-dependent errors.**

$$
\begin{aligned}
|\eta(\pi) - \eta^{\text{llm}}_{p,k,T}(\pi)| = \Bigg| & \sum_{t=T}^{\infty} \gamma^t \mathbb{E}_{s_t \sim P^t, a_t \sim \pi}\Big[r(s_t, a_t)\Big] \\
& - \mathbb{E}_{X_{t-i} \sim b(p), 1 \le i \le k}\Bigg[\mathbb{1}[A_t^k] \frac{1}{\sum_{i=1}^k X_{t-i}} \sum_{i=1}^k X_{t-i} \mathbb{E}_{s_t \sim \hat{P}^i_{t,\text{llm}}, a_t \sim \pi}\Big[r(s_t, a_t)\Big] \\
& - \mathbb{1}[\bar{A}_t^k] \mathbb{E}_{s_t \sim P^t, a_t \sim \pi}\Big[r(s_t, a_t)\Big]\Bigg]\Bigg| \\
& \le \sum_{t=T}^{\infty} \gamma^t \Bigg| \mathbb{E}_{X_{t-i} \sim b(p), 1 \le i \le k}\Bigg[\mathbb{1}[A_t^k] \mathbb{E}_{s_t \sim P^t, a_t \sim \pi}\Big[r(s_t, a_t)\Big] + \mathbb{1}[\bar{A}_t^k] \mathbb{E}_{s_t \sim P^t, a_t \sim \pi}\Big[r(s_t, a_t)\Big]\Bigg] \\
& - \mathbb{E}_{X_{t-i} \sim b(p), 1 \le i \le k}\Bigg[\mathbb{1}[A_t^k] \frac{1}{\sum_{i=1}^k X_{t-i}} \sum_{i=1}^k X_{t-i} \mathbb{E}_{s_t \sim \hat{P}^i_{t,\text{llm}}, a_t \sim \pi}\Big[r(s_t, a_t)\Big] \\
& - \mathbb{1}[\bar{A}_t^k] \mathbb{E}_{s_t \sim P^t, a_t \sim \pi}\Big[r(s_t, a_t)\Big]\Bigg]\Bigg| \\
& \le \sum_{t=T}^{\infty} \gamma^t \Bigg| \mathbb{E}_{X_{t-i} \sim b(p), 1 \le i \le k}\Bigg[\mathbb{1}[A_t^k] \Bigg(\\
& \mathbb{E}_{s_t \sim P^t, a_t \sim \pi}\Big[r(s_t, a_t)\Big] - \frac{1}{\sum_{i=1}^k X_{t-i}} \sum_{i=1}^k X_{t-i} \mathbb{E}_{s_t \sim \hat{P}^i_{t,\text{llm}}, a_t \sim \pi}\Big[r(s_t, a_t)\Big]\Bigg)\Bigg]\Bigg| \\
& \le \sum_{t=T}^{\infty} \gamma^t \Bigg| \mathbb{E}_{X_{t-i} \sim b(p), 1 \le i \le k}\Bigg[\mathbb{1}[A_t^k] \frac{1}{\sum_{i=1}^k X_{t-i}} \sum_{i=1}^k X_{t-i} \Bigg(\\
& \mathbb{E}_{s_t \sim P^t, a_t \sim \pi}\Big[r(s_t, a_t)\Big] - \mathbb{E}_{s_t \sim \hat{P}^i_{t,\text{llm}}, a_t \sim \pi}\Big[r(s_t, a_t)\Big]\Bigg)\Bigg]\Bigg|
\end{aligned}
$$

We then expand the integrals in the terms $\mathbb{E}_{s_t \sim P^t, a_t \sim \pi}\Big[r(s_t, a_t)\Big] - \mathbb{E}_{s_t \sim \hat{P}^i_{t,\text{llm}}, a_t \sim \pi}\Big[r(s_t, a_t)\Big]$ and express it in terms of horizon-dependent multi-step model errors:

$$
\begin{aligned}
& \mathbb{E}_{s_t \sim P^t, a_t \sim \pi}\Big[r(s_t, a_t)\Big] - \mathbb{E}_{s_t \sim \hat{P}^i_{t,\text{llm}}, a_t \sim \pi}\Big[r(s_t, a_t)\Big] \\
& \qquad = \int_{s \in \mathcal{S}} \int_{a \in \mathcal{A}} r(s, a) \Big(P^t(s, a) - \hat{P}^i_{t,\text{llm}}(s, a)\Big)\, da\, ds \\
& \qquad \le r_{\max} \int_{s \in \mathcal{S}} \int_{a \in \mathcal{A}} \Big(P^t(s, a) - \hat{P}^i_{t,\text{llm}}(s, a)\Big)\, da\, ds \\
& \qquad \le r_{\max} \int_{s \in \mathcal{S}} \int_{a \in \mathcal{A}} \Big(P^t(s) - \hat{P}^i_{t,\text{llm}}(s)\Big) \pi(a|s)\, da\, ds \\
& \qquad \le r_{\max} \int_{s \in \mathcal{S}} \Big(P^t(s) - \hat{P}^i_{t,\text{llm}}(s)\Big)\, ds \\
& \qquad \le 2 r_{\max} D_{\text{TV}}(P^t || \hat{P}^i_{t,\text{llm}})
\end{aligned}
\tag{C.3}
$$

Step 2: Simplifying the bound.
By applying lemma C.1.1 we can bound the multi-step errors using the bound on one-step errors:

$$D_{\text{TV}}(P^t || \hat{P}^i_{t,\text{llm}}) \leq i\, \varepsilon_{\text{llm}}(T) \leq k\, \varepsilon_{\text{llm}}(T) \tag{C.4}$$

Therefore, the bound becomes:

$$\begin{aligned}
|\eta(\pi) - \eta^{\text{llm}}_{p,k,T}(\pi)| &\leq 2r_{\max}\, k\, \varepsilon_{\text{llm}}(T) \sum_{t=T}^{\infty} \gamma^t \left| \mathbb{E}_{X_{t-i}\sim b(p), 1\leq i\leq k} \left[\mathbb{1}[A^k_t] \frac{1}{\sum_{i=1}^k X_{t-i}} \sum_{i=1}^k X_{t-i} \right] \right| \\
&= 2r_{\max}\, k\, \varepsilon_{\text{llm}}(T) \sum_{t=T}^{\infty} \gamma^t \left| \mathbb{E}_{X_{t-i}\sim b(p), 1\leq i\leq k} \left[\mathbb{1}[A^k_t] \right] \right| \\
&\leq 2r_{\max}\, k\, \varepsilon_{\text{llm}}(T) \sum_{t=T}^{\infty} \gamma^t kp \\
&= 2\frac{\gamma^T}{1-\gamma} r_{\max} k^2\, p\, \varepsilon_{\text{llm}}(T)
\end{aligned} \tag{C.5}$$

□

C.2 Related Work

Model-based reinforcement learning (MBRL). MBRL has been effectively used in iterated batch RL by alternating between model learning and planning [DR11; Haf+21b; GMR16b; LK; Chu+18; Jan+19; KHT21], and in the offline (pure batch) RL where we do one step of model learning followed by policy learning [Yu+20b; Kid+20; LLK21; AD21; ZZX21; Yu+21b; LCZ21; BTK24]. Planning is used either at decision time via model-predictive control (MPC) [DER95; Chu+18; Haf+19b; Pin+20; KHT21], or in the background where a model-free agent is learned on imagined model rollouts (Dyna; [Jan+19; Sut91; Sut+92; HS18a]), or both. For example, model-based policy optimization (MBPO) [Jan+19] trains an ensemble of feed-forward models and generates imaginary rollouts to train a soft actor-critic agent.

LLMs in RL. LLMs have been integrated into reinforcement learning (RL) [Cao+24; Yan+23], playing key roles in enhancing decision-making [KVM24; PFR; Zha+24b; Fen+24], reward design [Kwo+23; Wu+24; Car+23; Liu+23], and information processing [Pou+23; Lin+24]. The use of LLMs as world models is particularly relevant to our work. More generally, the Transformer architecture [Vas+17] has been used in offline RL (Decision Transformer [Che+21]; Trajectory Transformer [JLL21]). Pre-trained LLMs have been used to initialize decision transformers and fine-tune them for offline RL tasks [Shi+23; RYG22; YX24a]. As world models, Dreamer-like architectures based on Transformers

have been proposed [MAF22; Zha+23b; Che+22], demonstrating efficiency for long-memory tasks such as Atari games. In text-based environments, LLMs have found multiple applications [Lin+24; Fen+24; Zha+24b; Ma+24b], including using code-generating LLMs to generate policies in a zero-shot fashion [Lia+23; Liu+24a].

The closest work to ours is [Wan+23], where a system prompt consisting of multiple pieces of information about the control environment (e.g., description of the state and action spaces, nature of the controller, historical observations, and actions) is fed to the LLM. Unlike our approach, which focuses on predicting the dynamics of RL environments, [Wan+23] aim to directly learn a low-level control policy from the LLM, incorporating extra information in the prompt. Furthermore, [Wan+23] found that only GPT-4 was usable within their framework, while we provide a proof-of-concept using smaller open LLMs such as Llama 3.2 1B.

ICL on Numerical Data. In-context learning for regression tasks has been theoretically analyzed in several works, providing insights based on the Transformer architecture [Li+23; Osw+23; Aky+23; Gar+23; Xie+22]. Regarding time series forecasting, LLMTime [Gru+23a] successfully leverages ICL for zero-shot extrapolation of one-dimensional time series data. Similarly, [**das_decoder-only_2024**] introduce a foundational model for one-dimensional zero-shot time series forecasting, while [XS23] combine numerical data and text in a question-answer format. ICL can also be used to approximate a continuous density from the LLM logits. For example, [Liu+24b] develop a Hierarchical *softmax* algorithm to infer the transition rules of uni-dimensional Markovian dynamical systems. Building on this work, [ZBR] provide an application that predicts the parameter value trajectories in the Stochastic Gradient Descent algorithm. More relevant to our work, [Req+24] presented *LLMProcesses*, a method aimed at extracting multi-dimensional distributions from LLMs. Other practical applications of ICL on numerical data include few-shot classification on tabular data [Heg+23], regression [Vac+24], and meta ICL [Cod+23].

C.3 State and action dimensions interdependence - additional materials

C.3.1 Principal Component Analysis (PCA)

Principal Component Analysis. PCA is a dimensionality reduction technique that transforms the original variables into a new set of variables, the principal components, which are linearly uncorrelated. The principal components can be ordered such that the first few retain most of the variation present in all of the original variables. Formally, given

a data matrix $\mathbf{X}$ with n observations and p variables, PCA diagonalizes the covariance matrix $\mathbf{C} = \frac{1}{n-1}\mathbf{X}^T\mathbf{X}$ to find the eigenvectors, which represent the directions of the principal components: PCA: $\mathbf{X} \to \mathbf{Z} = \mathbf{XW}$, where $\mathbf{W}$ are the eigenvectors of $\mathbf{C}$. In our case, the data represents a dataset of states and actions given a data collecting policy π_D, while the p variables represent the state (eventually also the action) dimensions.

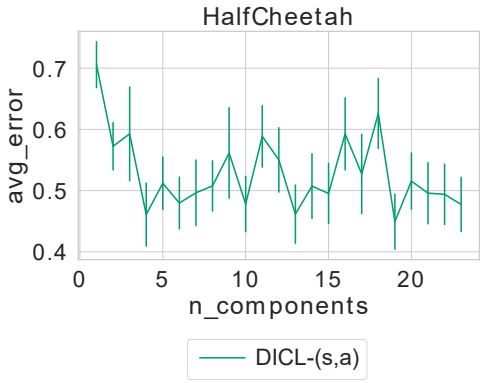


Figure C.1: Ablation study on the number of principal components in the DICL-$(\mathbf{s}, \mathbf{a})$ method.

Ablation on the number of components. Figure C.1 shows an ablation study on the number of components used in the DICL-$(\mathbf{s}, \mathbf{a})$ method. Surprisingly, we observe a sharp decline in the average multi-step error (see Section C.6 for a detailed definition) given only 4 components among 23 in the HalfCheetah system. The error then slightly increases for an intermediate number of components, before going down again when the full variance is recovered. This finding strengthens the position of PCA as our Disentangling algorithm of choice in DICL.

C.3.2 Independent Component Analysis (ICA)

ICA is a statistical and computational technique used to separate a multivariate signal into additive, statistically independent components. Unlike PCA, which decorrelates the data, ICA aims to find a linear transformation that makes the components as independent as possible. Given a data matrix $\mathbf{X}$, ICA assumes that the data is generated as linear mixtures of independent components: $\mathbf{X} = \mathbf{AS}$, where $\mathbf{A}$ is an unknown mixing matrix and $\mathbf{S}$ is the matrix of independent components with independent rows. The goal of ICA is to estimate an unmixing matrix $\mathbf{W}$ such that $\mathbf{Y} = \mathbf{WX}$ is a good approximation of the independent components $\mathbf{S}$. The implications of ICA on independence are profound: while PCA only guarantees uncorrelated components, ICA goes a step further by op-

timizing for statistical independence, often measured by non-Gaussianity (kurtosis or negentropy).

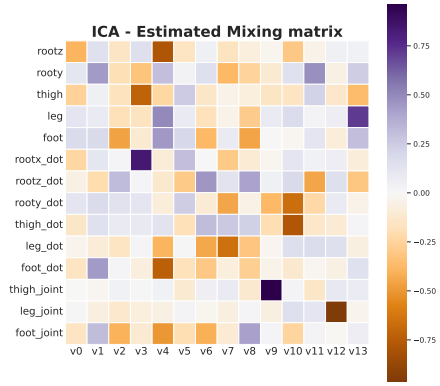


Figure C.2: ICA estimated mixing matrix.

Figure C.2 shows the estimated mixing matrix $\mathbf{A}$ when running ICA on the D4RL-*expert* dataset on the Hopper environment. Under the assumptions of ICA, notably the statistical independence of the source signals, their linear mixing and the invertibility of the original (unknown) mixing matrix, the original sources are successfully recovered if each line of the estimated mixing matrix is mostly dominated by a single value, meaning that it's close to an identity matrix up to a permutation with scaling. In the case of our states and actions data, it's not clear that this is the case from Figure C.2. Similarly to PCA, we can transform the in-context multi-dimensional signal using ICA, and apply the ICL procedure to the recovered independent sources. We plan on exploring this method in future follow-up work.

C.3.3 AutoEncoder-based approach

Variational Autoencoders (VAEs) [**kingma2022autoencodingvariationalbayes**] offer a powerful framework for learning representations. A disentangled representation is one where each dimension of the latent space captures a distinct and interpretable factor of variation in the data. By combining an encoder network that maps inputs to a probabilistic latent space with a decoder network that reconstructs the data, VAEs employ the reparameterization trick to enable backpropagation through the sampling process. The key to disentanglement lies in the KL-divergence term of the VAE loss function, which regularizes the latent distribution to be close to a standard normal distribution. Variants such as β-VAE [Hig+17] further emphasize this regularization by scaling the

KL-divergence term, thereby encouraging the model to learn a more disentangled representation at the potential cost of reconstruction quality. Beyond simple VAEs, there exist previous work in the literature that specifically aim at learning a factorized posterior distribution in the latent space [KM19]. Although this direction looks promising, it strikes different concerns about the learnability of these models in the low data regime considered in our paper.

C.3.4 Sensitivity Analysis

The preceding analysis examines state dimensions as features within a representation space, disregarding their temporal nature and our ultimate objective of predicting the next state. In practice, our interest lies in capturing the dependencies that most significantly influence the next state through the dynamics function of the MDP. To achieve this, we use Sensitivity Analysis (SA) to investigate how variations in the input of the dynamics function impact its output.

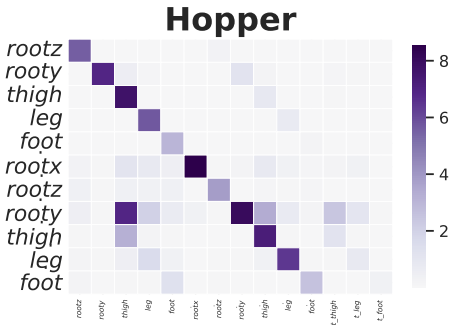


Figure C.3: Sensitivity matrix.

Sensitivity Analysis. Sensitivity analysis is a systematic approach to evaluate how the uncertainty in the output of a model can be attributed to different sources of uncertainty in the model's inputs. The One-at-a-Time (OAT) method is a technique used to understand the impact of individual input variables on the output of a model. In the context of a transition function of a MDP, the OAT method involves systematically varying one current state or action dimension at a time, while keeping all others fixed, and observing the resulting changes in the output dimensions: $\frac{\partial(\mathbf{s}_{t+1})_k}{\partial(\mathbf{s}_t)_i}$ and $\frac{\partial(\mathbf{s}_{t+1})_k}{\partial(\mathbf{a}_t)_j}$, where $(\mathbf{s}_t)_i$, $(\mathbf{a}_t)_j$ and $(\mathbf{s}_{t+1})_k$ denote the i-th dimension of the state, the j-th dimension of the action, and the k-th dimension of the next state, respectively.

In practice, we measure the sensitivity by applying a perturbation (of scale 10%) to each input dimension separately, reporting the absolute change that occurs in each dimension of the output. Precisely, for a deterministic transition function f, input state dimension i,

and output dimension k, we measure $|f(s+\epsilon, a)_k - f(s, a)_k|$ where $\epsilon_i = 0.1 \times scale(i)$ and 0 elsewhere. The sensitivity matrix in Figure C.3 demonstrates that most of the next state dimensions are mostly affected by their respective previous values (the diagonal shape in the state dimensions square). In addition to that, actions only directly affect some state dimensions, specifically velocities, which is expected from the nature of the physics simulation underlying those systems. This finding suggests that the vICL method might give good results in practice for the considered RL environments, and makes us hope that the DICL-(s) approach is enough to capture the state dimensions dependencies, especially for single-step prediction.

Remark C.3.1. *This sensitivity analysis is specific to the single-step transition function. In practice, such conclusions might change when looking at a larger time scale of the simulation.*

C.4 Algorithms

C.4.1 Soft-Actor Critic

Soft Actor-Critic (SAC) [Haa+18b] is an off-policy algorithm that incorporates the maximum entropy framework, which encourages exploration by seeking to maximize the entropy of the policy in addition to the expected return. SAC uses a deep neural network to approximate the policy (actor) and the value functions (critics), employing two Q-value functions to mitigate positive bias in the policy improvement step typical of off-policy algorithms. This approach helps in learning more stable and effective policies for complex environments, making SAC particularly suitable for tasks with high-dimensional, continuous action spaces.

We use the implementation provided in CleanRL [Hua+22b] for SAC. In all environments, we keep the default hyperparameters provided with the library, except for the update frequency. We specify in Table C.1 the complete list of hyperparameters used for every considered environment.

Table C.1: SAC hyperparameters.

Environment	HalfCheetah	Hopper	Pendulum
Update frequency	1000	1000	200
Learning starts	5000	5000	1000
Batch size	128	128	64
Total timesteps	$1e6$	$1e6$	$1e4$
Gamma γ	0.99	0.99	0.99
policy learning rate	$3e-4$	$3e-4$	$3e-4$

C.4.2 DICL-SAC

For our algorithm, we integrate an LLM inference interface (typically the Transformers library from Huggingface [Wol+20]) with CleanRL [Hua+22b]. Table C.2 shows all DICL-SAC hyperparameter choices for the considered environments.

Table C.2: DICL-SAC hyperparameters.

Environment	HalfCheetah	Hopper	Pendulum
Update frequency	1000	1000	200
Learning starts	5000	5000	1000
LLM Learning starts	10000	10000	2000
LLM Learning frequency	256	256	16
Batch size	128	128	64
LLM Batch size ($\alpha\%$)	$7(5\%), 13(10\%), 32(25\%)$	$7(5\%), 13(10\%), 32(25\%)$	$4(5\%), 7(10\%), 16(25\%)$
Total timesteps	$1e6$	$1e6$	$1e4$
Gamma γ	0.99	0.99	0.99
Max context length	500	500	198
Min context length	1	1	1
LLM sampling method	*mode*	*mode*	*mode*
LLM dynamics learner	vICL	vICL	vICL

Balancing gradient updates. To ensure that DICL-SAC performs equally important gradient updates on the LLM generated data, we used a gradient updates balancing mechanism. Indeed, since the default reduction method of loss functions is averaging, the batch $\mathcal{B}$ with the smallest batch size gets assigned a higher weight when doing gradient descent: $\frac{1}{|\mathcal{B}|}$. To address this, we multiply the loss corresponding to the LLM generated batch $\mathcal{B}_{\text{llm}}$ with a correcting coefficient $\frac{|\mathcal{B}_{\text{llm}}|}{|\mathcal{B}|}$ ensuring equal weighting across all samples.

We now show the full training curves on the HalfCheetah and Hopper environments (Figure C.4). The return curves show smoothed average training curves $\pm$ 95% Gaussian confidence intervals for 5 seeds in HalfCheetah and Hopper, and 10 seeds for Pendulum.

The update frequency. The default update frequency of SAC is 1 step, meaning that the policy that interacts with the environment gets updated after every interaction. In our LLM-based framework, this introduces an additional layer of complexity at this implies that the state visitation distribution of the in-context trajectories will be moving from one timestamp to another. We therefore assume an update frequency equal to the maximal number of steps of an episode of a given environment.

It is important to mention that the choice of setting the update frequency for all algorithms to the number of steps equivalent to a full episode has dual implications: it can stabilize the data collection policy, which is beneficial, but it may also lead to overtraining on data gathered by early, low-quality policies, which is detrimental. This trade-off has been previously studied in the RL literature [Mat+21; Tho+24]. Notably, [Tho+24]

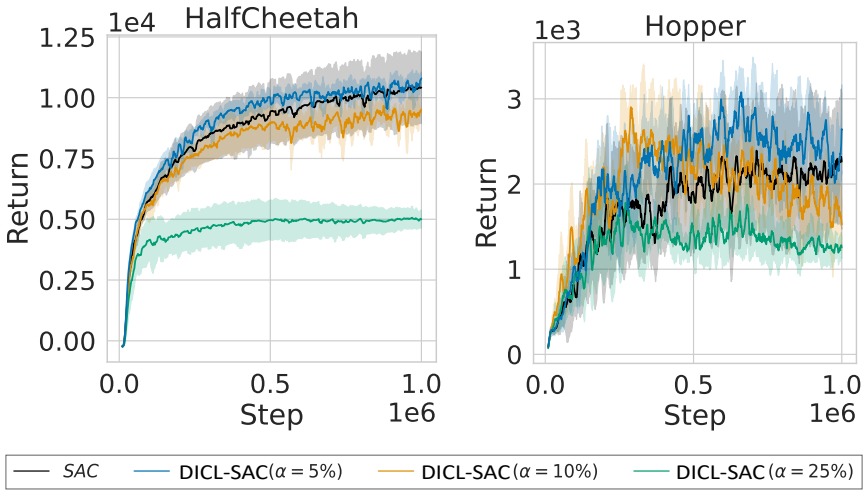


Figure C.4: **Data-augmented off-policy RL.** Full training curves.

argues that the update frequency is more of a system constraint than a design choice or hyperparameter. For instance, controlling a physically grounded system, such as a helicopter, inherently imposes a minimal update frequency. Therefore, we deem it a fair comparison as this constraint is uniformly applied to all algorithms.

For the sake of completeness and comparison, we also evaluated the SAC baseline using its default update frequency of one step. Figure C.5 shows the comparison of our algorithm DICL-SAC, the baseline SAC with update frequency 1000, and the default SAC with update frequency 1. We see that on Halfcheetah the default SAC ($uf = 1$) performs similarly to SAC with an update frequency of 1000. On Pendulum and Hopper it performs slightly better with DICL remaining competitive while having the constraint of an update frequency of 1000.

C.5 What is the impact of the policy on the prediction error?

In this experiment, We investigate how a policy impacts the accuracy and calibration of our LLM-based dynamics models. To do so, we train three model-free algorithms (PPO [Sch+17], SAC [Haa+18b], and TD3 [FMP19]) on the HalfCheetah environment, selecting different checkpoints throughout training to capture diverse policies. We then analyze the correlation between policy characteristics, specifically state coverage (defined as the maximum distance between any two states encountered by the policy) and entropy, with the Mean Squared Error and Kolmogorov-Smirnov (KS) statistic. Our findings indicate that the state coverage correlates with both MSE and KS, possibly be-

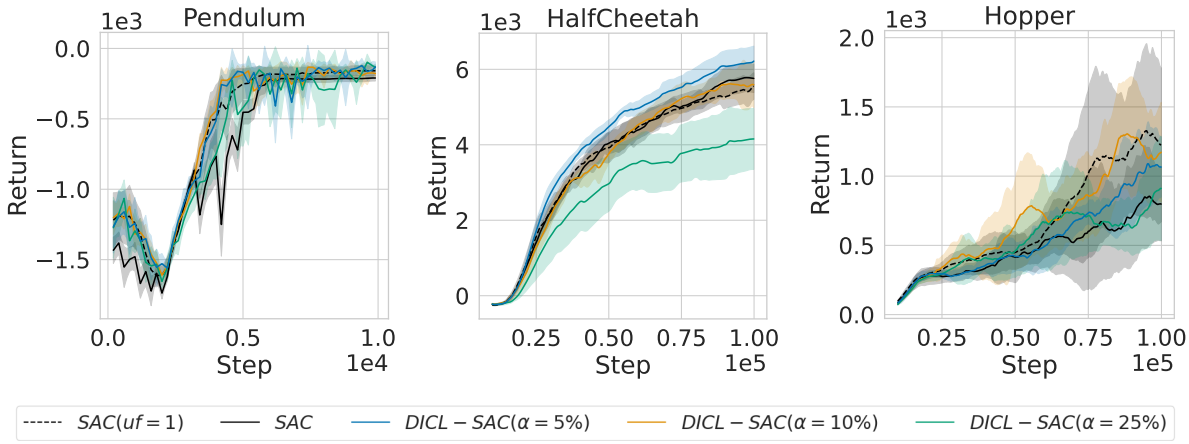


Figure C.5: **Data-augmented off-policy RL.** Comparison with SAC in the default update frequency regime. We conducted this experiment using the *Llama 3.2-1B* model.

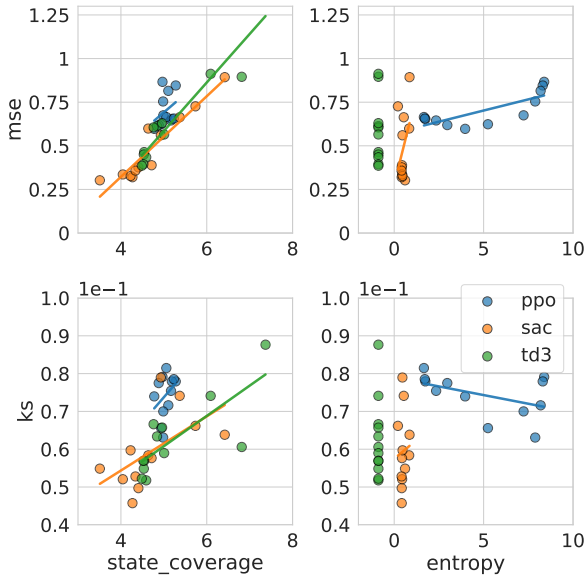


Figure C.6: Correlation plots between state coverage and entropy of policies with MSE and KS metrics under the vICL dynamics learner.

cause policies that explore a wide range of states generate trajectories that are more difficult to learn. Regarding the entropy, we can see that it also correlates with MSE, but interestingly, it does not appear to impact the calibration.

C.6 Multi-step prediction errors

The average multi-step error. In Figure 4.4, we compute the average Mean Squared Error over prediction horizons for $h = 1, \dots, 20$, and 5 trajectories sampled uniformly from the D4RL expert dataset. For visualization purposes, we first rescale all the dimensions (using a pipeline composed of a *MinMaxScaler* and a *StandardScaler*) so that the respective MSEs are on the same scale. The MSE metric in Table 4.1 is also computed in a similar fashion, with the exception that it's average over 7 different tasks (HalfCheetah: random, medium, expert; Hopper: medium, expert; Walker2d: medium, expert).

The MLP baseline. For the MLP baseline, we instantiate an MLP with: 4 layers, 128 neurons each, and $ReLU$ activations. We then format the in-context trajectory as a dataset of $\{(s_t, a_t, s_{t+1})\}$ on which we train the MLP for 150 epochs using early stopping and the Adam optimizer [KB15].

We now extend Figure 4.4 to show the multi-step generated trajectories for all the dimensions of the HalfCheetah system in Figure C.7.

C.7 Calibration

The naive baseline. In the calibration plots Figures C.8 and 4.8, we compare the LLM-based dynamics models with a (naive) baseline that estimates a Gaussian distribution using the in-context moments (mean and variance).

Kolmogorov-Smirnov Statistic (KS): This metric is computed using the quantiles (under the model distribution) of the ground truth values. Hypothetically, these quantiles are uniform if the error in predicting the ground truth is a random variable distributed according to a Gaussian with the predicted standard deviation, a property we characterize as *calibration*. To assess this, we compute the Kolmogorov-Smirnov (KS) statistics. Formally, starting from the model cumulative distribution function (CDF) $F_\theta(s_{t+1}|s_t, a_t)$, we define the empirical CDF of the quantiles of ground truth values by $\mathcal{F}_{\theta,j}(x) = \frac{\left|\left\{(s_t,a_t,s_{t+1})\in\mathcal{D} | F^j_\theta(s_{t+1}|s_t,a_t)\leq x\right\}\right|}{N}$ for $x \in [0, 1]$. We denote by $U(x)$ the CDF of the uniform distribution over the interval $[0, 1]$, and we define the KS statistics as the largest absolute difference between the two CDFs across the dataset $\mathcal{D}$:

$$\text{KS}(\mathcal{D}; \theta; j \in \{1, \dots, d_s\}) = \max_{i\in\{1,\dots,N\}} \left|\mathcal{F}_{\theta,j}(F^j_\theta(s_{i,t+1}|s_{i,t}, a_{i,t})) - U(F^j_\theta(s_{i,t+1}|s_{i,t}, a_{i,t}))\right| \tag{C.6}$$

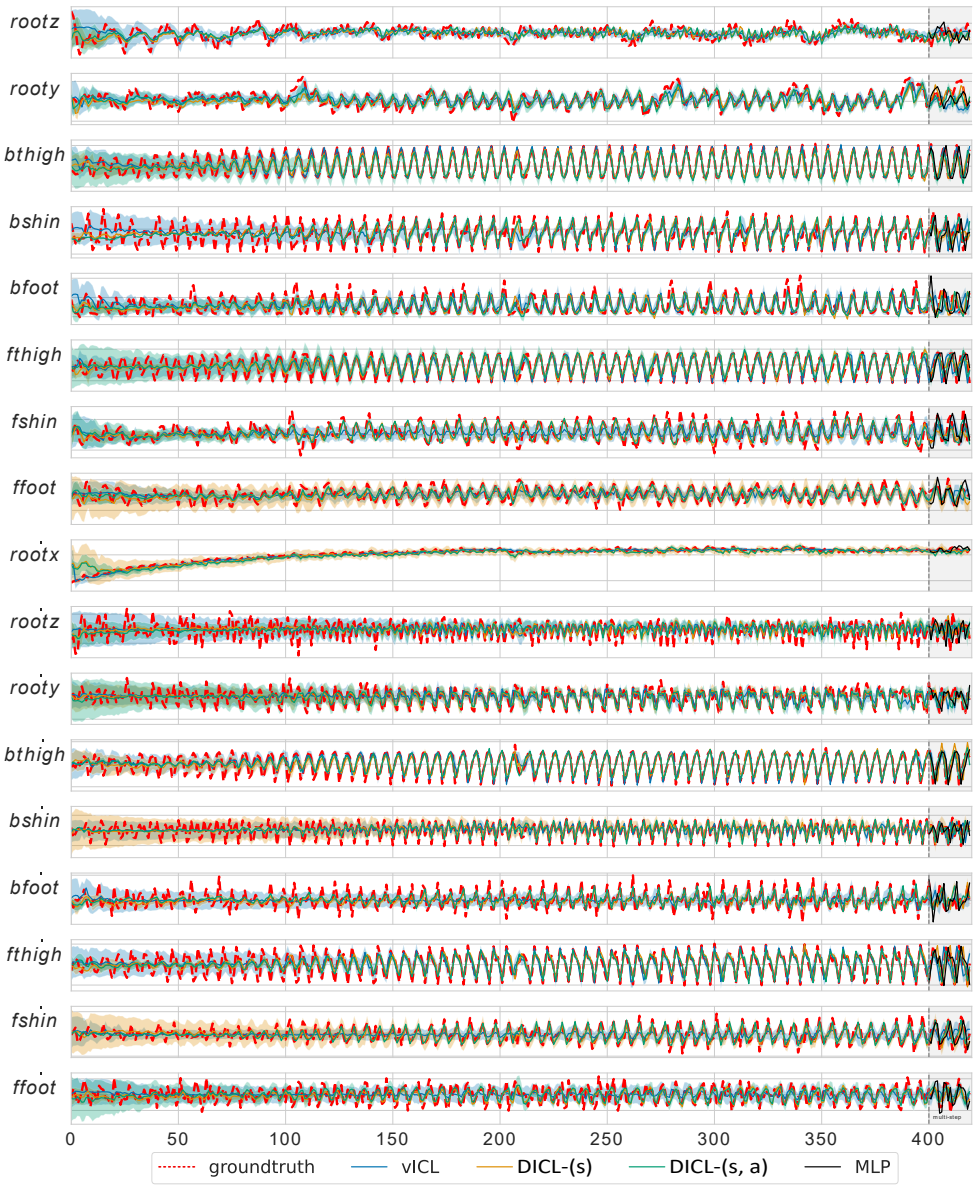


Figure C.7: Halfcheetah

The KS score ranges between zero and one, with lower values indicating better calibration.

C.8 On the choice of the LLM

In this ablation study, we investigate the impact of LLM size on prediction performance and calibration on D4RL tasks. The LLMs analyzed are all from the LLaMA 3 family of models [DL24], with size range from 1B to 70B parameters, including intermediate

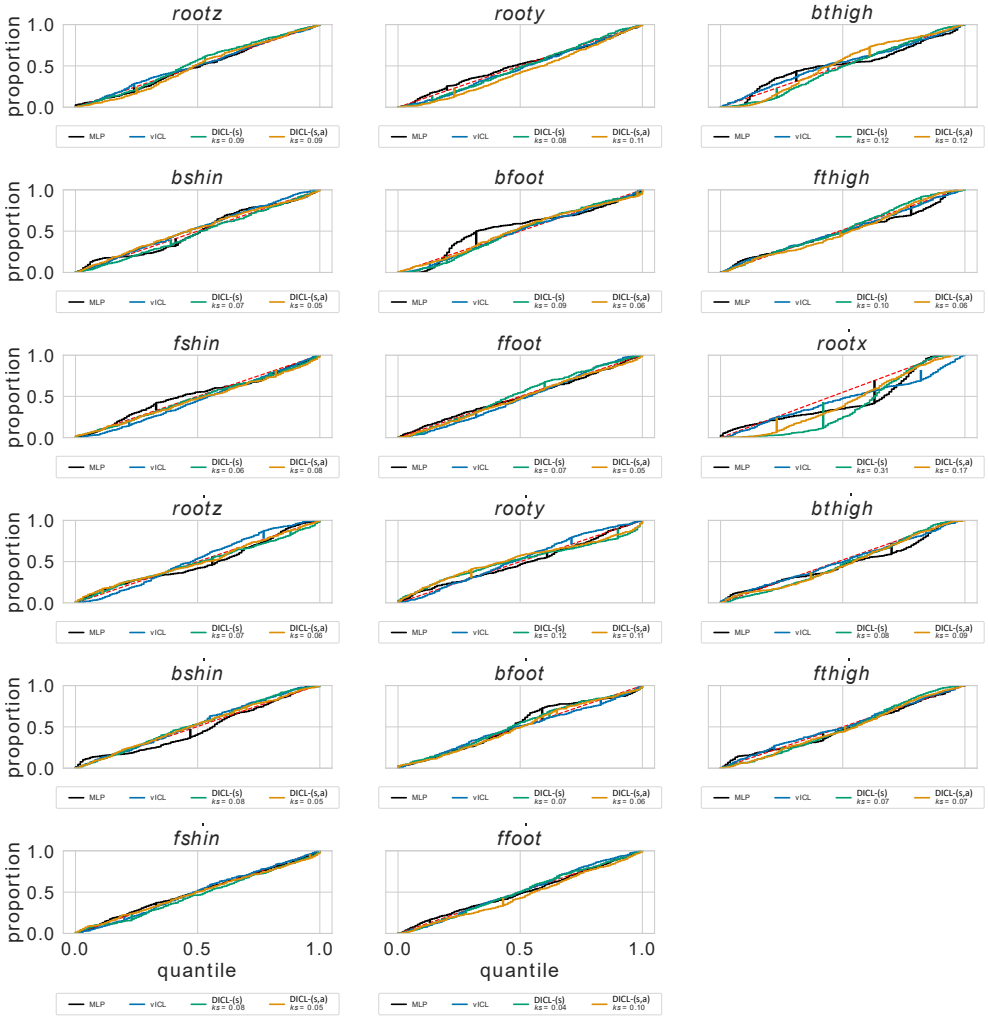


Figure C.8: Halfcheetah

sizes of 3B and 8B. Each model is fed with 5 randomly sampled trajectories of length $T = 300$ from the D4RL datasets: expert, medium, and random. This latter task is only evaluated on HalfCheetah, since the Hopper and Walker2d environments random policies episodes do not have enough context yet to apply DICL. For the medium and expert datasets, we evaluate them on all the environments HalfCheetah (Figure C.9), Hopper (Figure C.10), and Walker2d (Figure C.11). The metrics used to evaluate the models are:

- Mean Squared Error (MSE): Applied after rescaling the data similarly to Section C.6 to measure the prediction error.
- Kolmogorov-Smirnov (KS) statistic: To evaluate calibration, indicating how well the predicted probabilities match the observed outcomes. This metric is formally described in Section C.7.

All results are averaged over prediction horizons $h \in \{1, \ldots, 20\}$. In the HalfCheetah environment, we observe that DICL-(s) consistently outperforms the other variants across all tasks and with almost all LLMs in terms of prediction error. DICL-(s, a) is outperformed by vICL in the *random* and *medium* datasets, while its performance improves in the *expert* dataset. This is likely because the policy has converged to a stable expert policy, making it easier for DICL-(s, a) to predict actions as well. Regarding calibration, the three methods generally perform similarly, with a slight advantage for DICL-(s, a), especially with smaller LLMs. In the Hopper environment, the MSE improvement of DICL over vICL is less pronounced with the smallest LLMs but becomes more evident with the LLaMA 3.1 70B model. However, DICL-(s, a) consistently and significantly outperforms both vICL and DICL-(s) in terms of the KS statistic (calibration). In the Walker2d environment, vICL proves to be a strong baseline in the expert task, while DICL-(s) shows improvements over it in the medium dataset. For calibration in Walker2d, DICL-(s, a) continues to outperform the other variants across all tasks and LLM sizes.

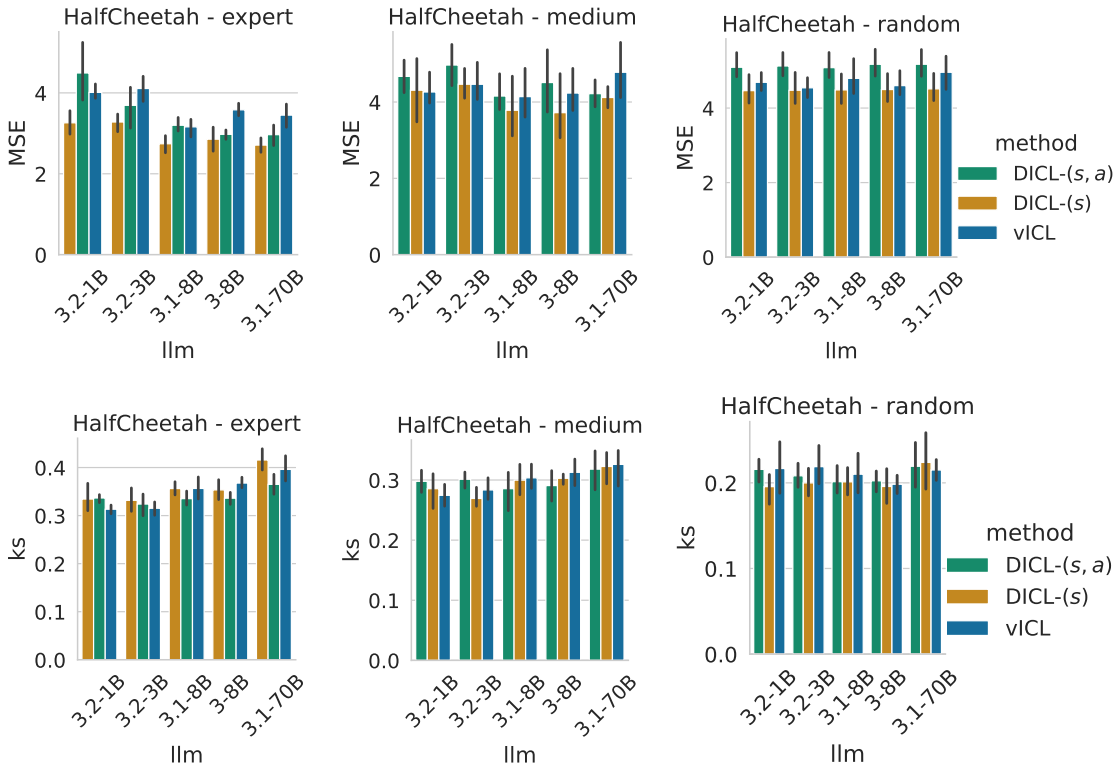


Figure C.9: HalfCheetah.

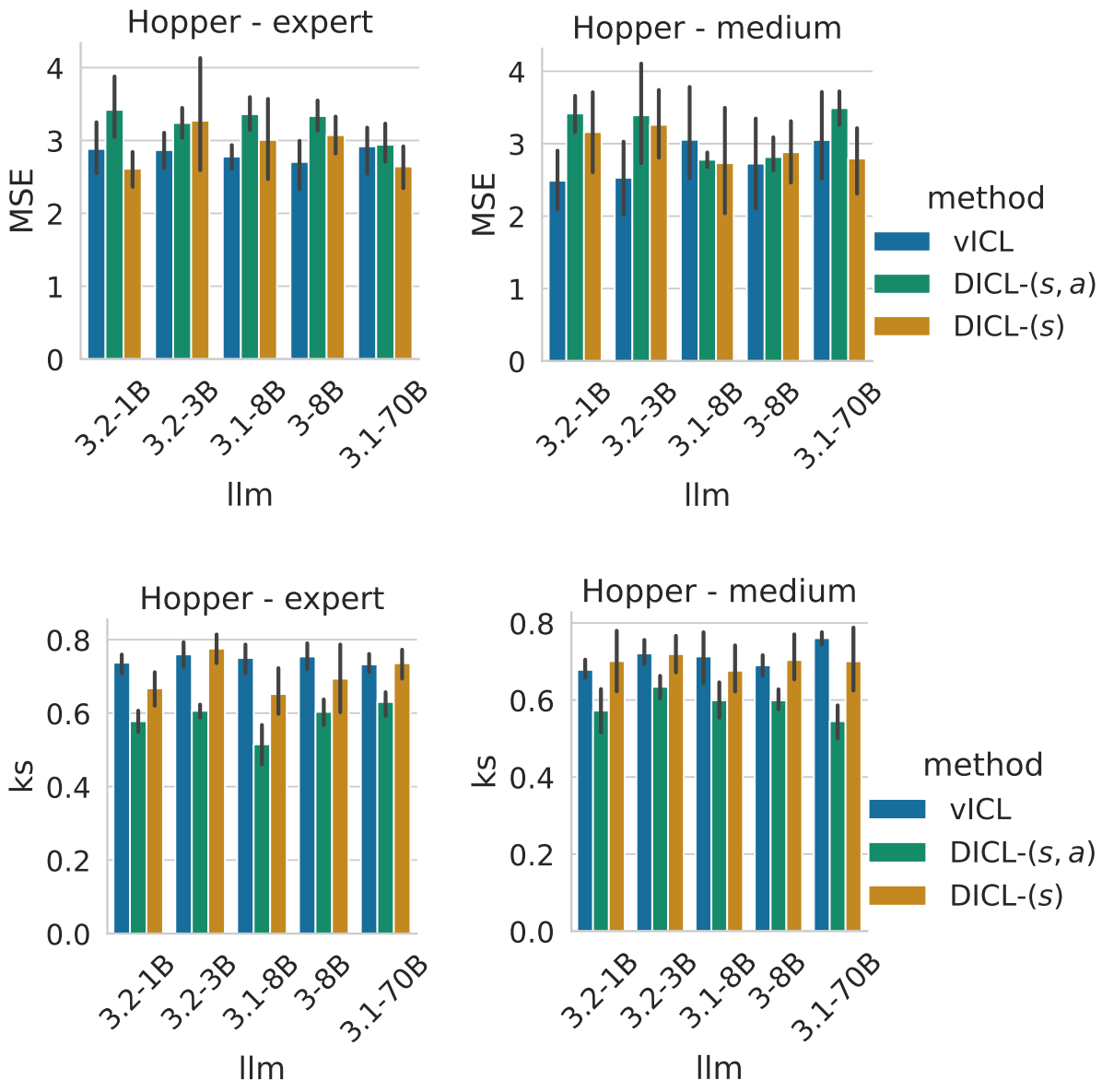


Figure C.10: Hopper.

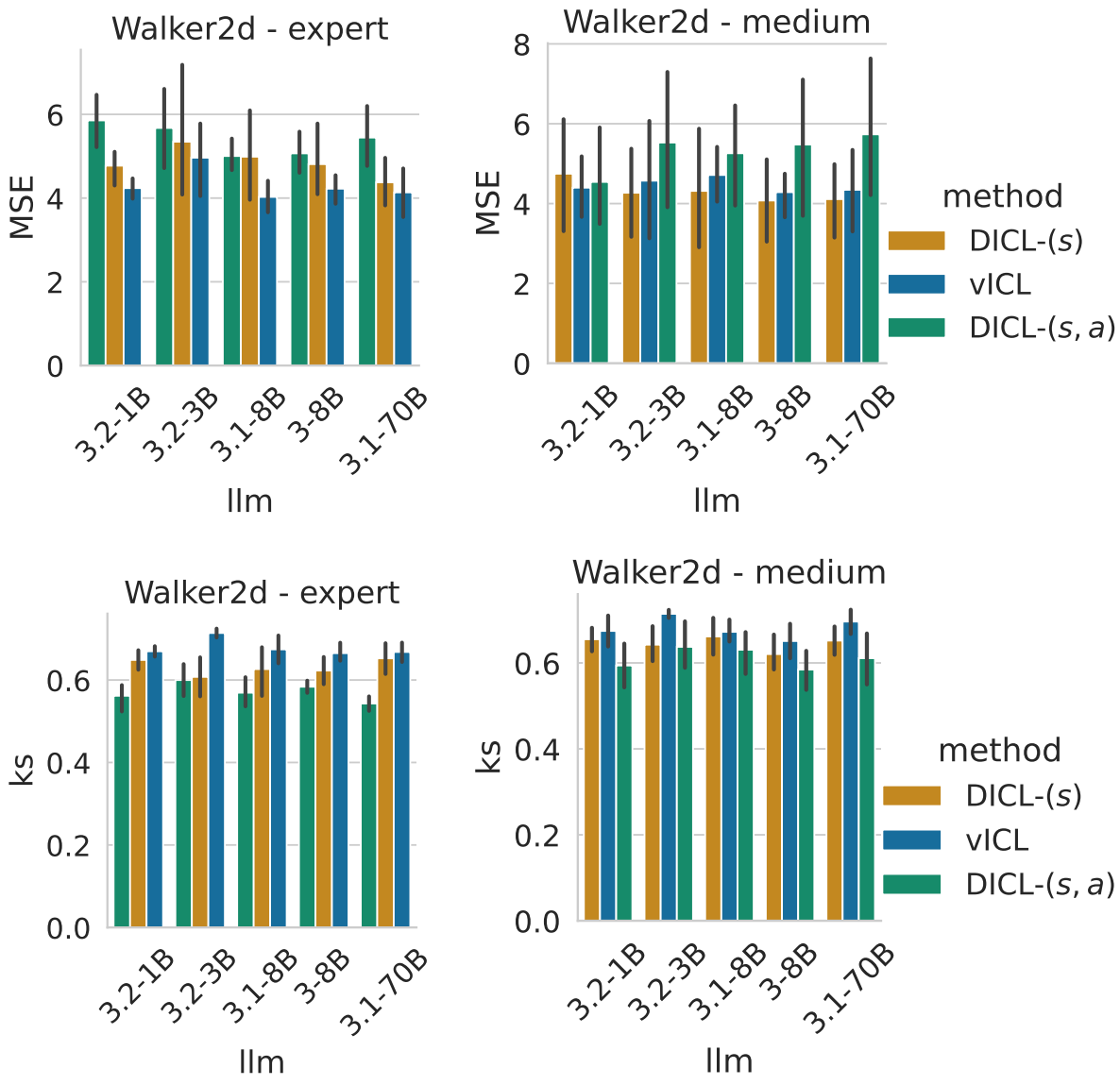


Figure C.11: Walker2d.

Appendix D

Appendix of Chapter 5

D.1 Theoretical analysis

D.1.1 Proof of proposition 5.3.1

We begin by restating the proposition, and its underlying assumptions:

Assumption D.1.1. *$\mathbf{W}_\varphi$ has full rank: rank$(\mathbf{W}_\varphi) = D$, insuring its invertibility.*

Assumption D.1.2. *For ease of derivation, we consider a similar linear parametrization for the foundation model: $f_{\text{FM}}(\mathbf{X}) = \mathbf{W}_{FM}^\top \mathbf{X} + \mathbf{b}_{FM}\mathbf{1}^\top$ where $\mathbf{W}_{FM} \in \mathbb{R}^{L\times H}$, $\mathbf{b}_{FM} \in \mathbb{R}^H$, and $\mathbf{1}$ a vector of ones of dimension D.*

Proposition D.1.1 (Optimal linear adapter)**.** *Under assumption 5.3.1 and assumption 5.3.2, the* closed-form *solution of the problem:*

$$\mathcal{L}(\mathbf{W}_\varphi) = \|\mathbf{Y} - \left(\mathbf{W}_{FM}^\top \mathbf{X}\mathbf{W}_\varphi + \mathbf{b}_{FM}\mathbf{1}^\top\right)\mathbf{W}_\varphi^{-1}\|_F^2 \tag{D.1}$$

writes as:

$$\mathbf{W}_\varphi^* = (\mathbf{B}^\top \mathbf{A})^+ \mathbf{B}^\top \mathbf{B} \tag{D.2}$$

where $\mathbf{W}_\varphi^* = \arg\min_{\mathbf{W}_\varphi \in \mathcal{GL}_D(\mathbb{R})} \mathcal{L}(\mathbf{W}_\varphi)$, $\mathbf{A} = \mathbf{Y} - \mathbf{W}_{FM}^\top \mathbf{X}$, $\mathbf{B} = \mathbf{b}_{FM}\mathbf{1}^\top$, and $(\mathbf{B}^\top\mathbf{A})^+$ denoting the pseudo-inverse *operator.*

Proof. We begin by expanding the loss function:

$$\begin{aligned}\mathcal{L}(\mathbf{W}_\varphi) &= \|\mathbf{Y} - (\mathbf{W}_{\text{FM}}^\top \mathbf{X}\mathbf{W}_\varphi + \mathbf{b}_{\text{FM}}\mathbf{1}^\top)\mathbf{W}_\varphi^{-1}\|_F^2 \\ &= \|\mathbf{A} - \mathbf{B}\mathbf{W}_\varphi^{-1}\|_F^2\end{aligned}$$

where $\mathbf{A} = \mathbf{Y} - \mathbf{W}_{FM}^\top \mathbf{X}$ and $\mathbf{B} = \mathbf{b}_{FM}\mathbf{1}^\top$. Expanding the Frobenius norm:

$$\mathcal{L}(\mathbf{W}_\varphi) = \text{Tr}\left((\mathbf{A} - \mathbf{B}\mathbf{W}_\varphi^{-1})^\top(\mathbf{A} - \mathbf{B}\mathbf{W}_\varphi^{-1})\right)$$

Taking the gradient with respect to $\mathbf{W}_\varphi^{-1}$ yields:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_\varphi^{-1}} = -2\mathbf{B}^\top\mathbf{A} + 2\mathbf{B}^\top\mathbf{B}\mathbf{W}_\varphi^{-1}$$

Knowing that $\mathbf{W}_\varphi$ is invertible, We have that: $\frac{\partial \mathcal{L}}{\partial \mathbf{W}_\varphi} = -\mathbf{W}_\varphi^{-\top}\frac{\partial \mathcal{L}}{\partial \mathbf{W}_\varphi^{-1}}\mathbf{W}_\varphi^{-\top}$

hence

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_\varphi} = -2\mathbf{W}_\varphi^{-T}\left(\mathbf{B}^\top\mathbf{A} - \mathbf{B}^\top\mathbf{B}\mathbf{W}_\varphi^{-1}\right)\mathbf{W}_\varphi^{-T}.$$

Setting $\frac{\partial \mathcal{L}}{\partial \mathbf{W}_\varphi} = 0$ and multiplying both sides by $\mathbf{W}_\varphi^\top$, we obtain:

$$\mathbf{B}^\top\mathbf{A} = \mathbf{B}^\top\mathbf{B}\mathbf{W}_\varphi^{-1}.$$

Multiplying both sides by $\mathbf{W}_\varphi$:

$$\mathbf{B}^\top\mathbf{A}\mathbf{W}_\varphi = \mathbf{B}^\top\mathbf{B}.$$

Finally applying the pseudo-inverse to solve for $\mathbf{W}_\varphi$ gives our final result:

$$\mathbf{W}_\varphi^* = (\mathbf{B}^\top\mathbf{A})^+\mathbf{B}^\top\mathbf{B}.$$

Given the convexity of $\mathcal{L}(\mathbf{W}_\varphi)$ (which follows from the convexity of the Frobenius norm $\|\cdot\|_{\text{F}}^2$, the inverse operation, and an affine transformation), we conclude that $\mathbf{W}_\varphi^*$ is a global solution for Equation (5.3).

Remark D.1.1. *We make use of the pseudo-inverse due to the current construction of the matrix* $\mathbf{B}$ (*with identical rows*) *which implies that the product* $\mathbf{B}^\top\mathbf{A}$ *is degenerate. To bypass this limitation and further ensure the invertibility of* $\mathbf{W}_\varphi^*$*, we can revisit the definition of the foundation model in assumption 5.3.2 to include channel dependent biases and ensure a full rank matrix* $\mathbf{B}$*.*

□

D.1.2 Proof of proposition 5.4.1

To derive the evidence lower bound (ELBO) used in the training objective of the VAE adapter, we start from the marginal likelihood of the observed data $\mathbf{Y}$ given the inputs $\mathbf{X}$ and foundation model f_{FM}. The marginal likelihood is expressed as:

$$\log p_\theta(\mathbf{Y}|\mathbf{X}, f_{\text{FM}}) = \log \int p_\theta(\mathbf{Y}|\mathbf{X}, f_{\text{FM}}(\mathbf{Z}))p(\mathbf{Z})\, d\mathbf{Z}, \tag{D.3}$$

where $\mathbf{Z}$ is the latent variable, $p_\theta(\mathbf{Y}|\mathbf{X}, f_{\text{FM}}(\mathbf{Z}))$ is the likelihood model parameterized by θ, and $p(\mathbf{Z})$ is the prior distribution over the latent variable $\mathbf{Z}$.

Direct optimization of this marginal likelihood is generally intractable due to the integration over $\mathbf{Z}$. To make this optimization feasible, we introduce a variational distribution $q_\phi(\mathbf{Z}|\mathbf{X})$, parameterized by ϕ, as an approximation to the true posterior $p_\theta(\mathbf{Z}|\mathbf{X}, \mathbf{Y}, f_{\text{FM}})$. Using $q_\phi(\mathbf{Z}|\mathbf{X})$, we can reformulate the log-marginal likelihood as follows:

$$\log p_\theta(\mathbf{Y}|\mathbf{X}, f_{\text{FM}}) = \log \int q_\phi(\mathbf{Z}|\mathbf{X}) \frac{p_\theta(\mathbf{Y}|\mathbf{X}, f_{\text{FM}}(\mathbf{Z}))p(\mathbf{Z})}{q_\phi(\mathbf{Z}|\mathbf{X})}\, d\mathbf{Z} \tag{D.4}$$

$$= \log \mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{X})} \left[\frac{p_\theta(\mathbf{Y}|\mathbf{X}, f_{\text{FM}}(\mathbf{Z}))p(\mathbf{Z})}{q_\phi(\mathbf{Z}|\mathbf{X})} \right]. \tag{D.5}$$

Using Jensen's inequality, we can derive a lower bound on this log-marginal likelihood:

$$\log p_\theta(\mathbf{Y}|\mathbf{X}, f_{\text{FM}}) \geq \mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{X})} \left[\log \frac{p_\theta(\mathbf{Y}|\mathbf{X}, f_{\text{FM}}(\mathbf{Z}))p(\mathbf{Z})}{q_\phi(\mathbf{Z}|\mathbf{X})} \right] \tag{D.6}$$

$$= \mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{X})} \left[\log p_\theta(\mathbf{Y}|\mathbf{X}, f_{\text{FM}}(\mathbf{Z})) \right] - \mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{X})} \left[\log \frac{q_\phi(\mathbf{Z}|\mathbf{X})}{p(\mathbf{Z})} \right]. \tag{D.7}$$

The second term can be rewritten as the Kullback-Leibler (KL) divergence between the variational posterior $q_\phi(\mathbf{Z}|\mathbf{X})$ and the prior $p(\mathbf{Z})$:

$$\text{KL}\left(q_\phi(\mathbf{Z}|\mathbf{X}) \,\|\, p(\mathbf{Z})\right) = \mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{X})} \left[\log \frac{q_\phi(\mathbf{Z}|\mathbf{X})}{p(\mathbf{Z})} \right]. \tag{D.8}$$

Substituting this into the inequality, we obtain the evidence lower bound (ELBO):

$$\log p_\theta(\mathbf{Y}|\mathbf{X}, f_{\text{FM}}) \geq \mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{X})} \left[\log p_\theta(\mathbf{Y}|\mathbf{X}, f_{\text{FM}}(\mathbf{Z})) \right] - \text{KL}\left(q_\phi(\mathbf{Z}|\mathbf{X}) \,\|\, p(\mathbf{Z})\right). \tag{D.9}$$

The ELBO consists of two terms:

- The *forecasting* term, $\mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{X})} \left[\log p_\theta(\mathbf{Y}|\mathbf{X}, f_{\text{FM}}(\mathbf{Z})) \right]$, which measures how well the model can reconstruct $\mathbf{Y}$ given the latent variable $\mathbf{Z}$.
- The *regularization* term, $\text{KL}\left(q_\phi(\mathbf{Z}|\mathbf{X}) \,\|\, p(\mathbf{Z})\right)$, which encourages the variational posterior to stay close to the prior distribution $p(\mathbf{Z})$.

Thus, maximizing the ELBO provides a tractable way to train the parameters θ and ϕ by optimizing the balance between forecasting accuracy and latent space regularization.

□

D.2 Normalizing Flows

Normalizing Flows make use of invertible transformations to map a simple base distribution (e.g. Gaussian) to a complex data distribution. Each transformation T is designed to maintain invertibility and efficient Jacobian computation. The transformation is applied iteratively: $\mathbf{Z} = T_k \circ T_{k-1} \circ \cdots \circ T_1(\mathbf{X})$. Current Normalizing Flow instantiations (e.g. `RealNVP`) make use of generic invertible transformations such as *coupling flows*; the latters can be parametrized using a neural network leading to powerful non-linear generative models that are trained to maximize the data log-likelihood:

$$\log p(\mathbf{X}) = \log p(\mathbf{Z}) + \sum_{i=1}^{k} \log \left| \det \frac{\partial T_i(\cdot;\theta)}{\partial \mathbf{Z}_{i-1}} \right|$$

where θ denote the parameters of the non-linear parametrization of the invertible transformations T_i, and $\mathbf{Z}_{i-1}$ is the output of the transformation T_{i-1}. In the context of time series adapters, we directly optimize the parameters of the transformations based on their direct and inverse application on the time series forecasting problem:

$$\mathcal{L}_{\text{flow}} = \|\mathbf{Y} - T_1^{-1} \circ T_2^{-1} \circ \cdots \circ T_k^{-1}(\\ f_{\text{FM}}\Big(T_k \circ T_{k-1} \circ \cdots \circ T_1(\mathbf{X};\theta)\Big);\theta)\|_F^2$$

where the encoder is represented by the series of direct transformations: $\text{enc}(\cdot) = T_k \circ T_{k-1} \circ \cdots \circ T_1(\cdot;\theta)$, and respectively the decoder by the series of inverse transformations $\text{dec}(\cdot) = T_1^{-1} \circ T_2^{-1} \circ \cdots \circ T_k^{-1}(\cdot;\theta)$.

As defined here, Normalizing Flows suffer from the constraint of keeping the same dimension in both original and learned representation space. For this purpose, we investigate coupling a normalizing flow with a linear encoder-decoder type of architecture to enable dimensionality reduction prior to applying the transformation T_i. The parameters of the additional encoder and decoder are then jointly trained to optimize the learning objective $\mathcal{L}_{\text{flow}}$.

Given that the parameters of the encoder and the decoder are shared in Normalizing Flows, the gradient-based optimization within our framework receives conflicting directions due to gradient flow from both the direct and inverse transformations simultaneously. We discovered that this adapter construction was challenging to optimize in practice, and we defer the exploration of this direction to future research endeavors.

Dataset	ETTh1	Illness	ExchangeRate	Weather
# features	7	7	8	21
# time steps	13603	169	6791	51899
Granularity	1 hour	1 week	1 day	10 minutes
(Train, Val, Test)	(8033, 2785, 2785)	(69, 2, 98)	(4704, 665, 1422)	(36280, 5175, 10444)

Table D.1: Characteristics of the multivariate time series datasets used in our experiments with various sizes and dimensions.

D.3 Experimental setup

D.3.1 Datasets

D.3.2 Implementation details

In this section, we describe the full `AdaPTS` framework, starting from the data preprocessing, the training algorithm, and the hyperparameters optimization.

Preprocessing. Given that the adapter as defined in definition 5.2.1 is a feature space transformation, we start by rescaling (*StandardScaler* and *MinMaxScaler*) the data where all the timesteps are regarded as data points. To account for the temporal specificities in each batch, we use Reversible Instance Normalization (RevIn) [Kim+22] that has been proven to mitigate time-related distribution shifts in time series problems. Finally, and following the observation that `PCA` when composed with a linear adapter showed the best result in the case of correlated data (Figure 5.2), we include the possibility of applying *full-component* `PCA` as part of our data pre-processing pipeline.

Training parameters. After the pre-processing phase, we proceed to split the data into a *train-validation-test* sets, where the validation set serves as a tool to select the best hyperparameters for the adapter. The resulting adapter that is instantiated with the optimal hyperparameters is then tested against the unseen test dataset. For all of our experiments, we first train the linear forecasting head of `Moment` (referred to as *Linear Probing* in [Gos+24]) with the Adam optimizer [KB17], a batch size of 32, a one cycle scheduler starting with 0.001 as learning rate. Once the forecasting linear head is trained, we freeze its parameters and proceed to training the adapter. This is done using the Adam optimizer, a batch size of 32, a reduce on plateau scheduler starting with 0.001 as learning rate.

Hyperparameter optimization. In order to select the best hyperparameters for the adapter architecture we use *Ray tune* [Lia+18] with the Heteroscedastic and Evolutionary Bayesian Optimisation solver (HEBO) [Cow+22] engine, reporting the average mean squred error (MSE) from *k-fold* cross validation. Table D.2 shows the default hyperparameters for each considered adapter.

adapter	`LinearAE`	`DropoutLinearAE`	`LinearVAE`	`VAE`
p dropout	–	0.1	–	–
Number of layers	–	–	–	2
Hidden dimension	–	–	–	128
β	–	–	0.5	0.5
σ	–	–	1.0	1.0

Table D.2: Adapters hyperparameters.

D.4 The `AdaPTS` Python package

To facilitate reproducibility and promote widespread adoption of our framework, we have developed `AdaPTS` as a Python package that provides a unified interface for adapting time series foundation models to multivariate probabilistic forecasting tasks. The package is designed with modularity, extensibility, and ease of use in mind, enabling researchers and practitioners to seamlessly integrate our adapter-based approach into their workflows.

D.4.1 Unified Foundation Model Interface

A key feature of the `AdaPTS` package is the development of a consistent interface across different time series foundation models. The package currently supports the foundation model families listed in Table D.3.

The package abstracts the complexities inherent in working with different foundation models through the *ICLTrainer* class hierarchy. Each foundation model family is implemented as a specialized child class that handles model-specific initialization, preprocessing, and inference procedures while maintaining a unified API. This architectural design enables users to effortlessly switch between different foundation models and implement custom *ICLTrainer* classes for newly-released or proprietary foundation models.

Model	MV	P	Link (HF)
`Moment`	✗	✗	autonlab/moment
`TTM`	✗	✗	ibm-granite/ttm
`TimesFM`	✗	✗	google/timesfm
`Chronos`	✗	✓	amazon/chronos
`Moirai`	!	✓	salesforce/moirai

! `Moirai` uses flatten + attention bias approach to MV.

Table D.3: Time Series Foundation Models supported in the `adapts` Python package. MV: Multivariate support; P: Probabilistic forecasting.

D.4.2 Training modes

Our framework supports multiple adapter training paradigms to handle different experimental scenarios:

- **Supervised Training.** This constitutes the primary training methodology analyzed throughout the chapter and showcased in Section 5.1. The adapter components (encoder and decoder) are optimized to minimize the forecasting MSE loss (Equation (5.2)) while the frozen foundation model generates predictions within the learned latent space.
- **Unsupervised Training.** Adapters are trained using a reconstruction loss, where the encoder-decoder pair learns to reconstruct the input time series. This approach is useful for representation learning and can serve as a pretraining step before supervised fine-tuning.
- **Fine-tuning support.** This training mode permits to fine-tune the foundation model parameters alongside adapter training. This is particularly relevant for models like `Moment`, where the linear forecasting head (denoted as `LP` for linear probing in [Gos+24]) is not pre-trained. Additionally, this can be implemented through a **bi-level** approach, where the foundation model undergoes secondary fine-tuning after completing adapter training.

Figure D.1 illustrates the command-line interface for running the **`AdaPTS`** approach, where users can configure their experiments by specifying the foundation model, dataset, adapter type, and training method through simple script parameters.

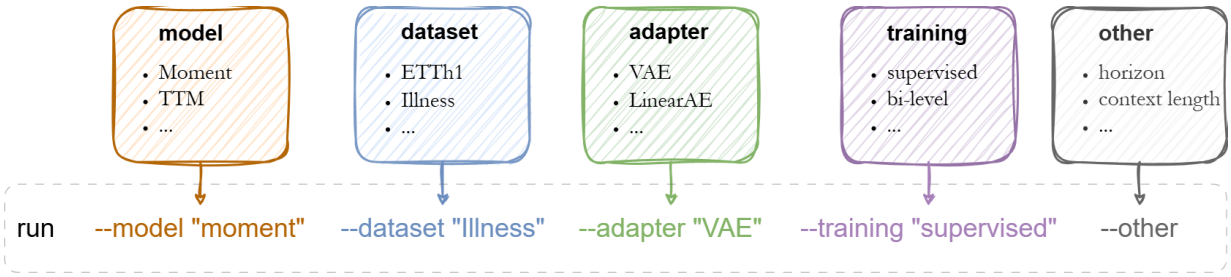


Figure D.1: Command-line interface for configuring and running experiments within `AdaPTS`.

D.5 Additional results

D.5.1 `Moment` applied to synthetic data.

To validate the adapter *optimality* condition with large non-linear foundation models, we use `Moment` [Gos+24]. The optimal linear adapter in this case minimizes the following intractable objective:

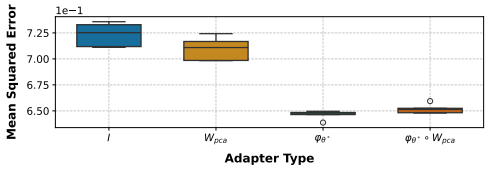
$$\mathcal{L}(\mathbf{W}_\varphi) = \|\mathbf{Y} - f_{\texttt{Moment}}\left(\mathbf{X}\mathbf{W}_\varphi\right)\mathbf{W}_\varphi^{-1}\|_{\mathrm{F}}^2 \tag{D.10}$$

To approximately solve this optimization problem, we instantiate $\mathbf{W}_\varphi$ as a single-linear-layer encoder denoted enc_θ, and respectively the inverse transformation $\mathbf{W}_\varphi^{-1}$ as a single-linear-layer decoder denoted dec_θ. We then use gradient-based optimization of the parameters θ using the Adam optimizer, aiming at solving the following optimization problem:

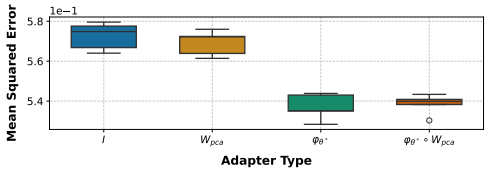
$$\theta^* = \arg\min_\theta \|\mathbf{Y} - \mathrm{dec}_\theta\Big(f_{\texttt{Moment}}(\mathrm{enc}_\theta(\mathbf{X}))\Big)\|_{\mathrm{F}}^2 \tag{D.11}$$

Figure D.2 shows the performance gain obtained by optimizing a linear adapter on *`Moment`-small* foundation model. Unlike the tractable case, we observe that in both data modalities (independent and correlated data), `PCA` has little to no improvement over the identity baseline, while φ_{θ^*} reaches an order of magnitude better solution. This confirms our intuition about the existence of a better solution than the identity matrix, even in the case of real-world complex foundation models.

D.5.2 Mean Absolute Error



Independent



Correlated

Figure D.2: `Moment` on simulated independent data.

Dataset	H	No adapter	with adapter				
		$\text{Moment}_{\text{small}}$	pca	linear	dropout	linear VAE	VAE
ETTh1	96	$0.422_{\pm 0.006}$	$0.440_{\pm 0.000}$	$0.423_{\pm 0.003}$	$\mathbf{0.415_{\pm 0.002}}$	$0.420_{\pm 0.001}$	$0.426_{\pm 0.001}$
	192	$\mathbf{0.436_{\pm 0.000}}$	$0.445_{\pm 0.000}$	$0.449_{\pm 0.003}$	$0.450_{\pm 0.001}$	$0.451_{\pm 0.001}$	$0.444_{\pm 0.001}$
Illness	24	$1.143_{\pm 0.007}$	$1.163_{\pm 0.001}$	$2.624_{\pm 0.035}$	$1.156_{\pm 0.016}$	$1.074_{\pm 0.011}$	$\mathbf{1.057_{\pm 0.012}}$
	60	$1.149_{\pm 0.001}$	$1.161_{\pm 0.001}$	$1.227_{\pm 0.030}$	$1.173_{\pm 0.015}$	$1.112_{\pm 0.021}$	$\mathbf{1.105_{\pm 0.021}}$
Weather	96	$0.232_{\pm 0.010}$	$0.235_{\pm 0.000}$	$0.226_{\pm 0.000}$	$0.212_{\pm 0.001}$	$\mathbf{0.218_{\pm 0.001}}$	$0.243_{\pm 0.001}$
	192	$\mathbf{0.251_{\pm 0.001}}$	$0.260_{\pm 0.001}$	$\mathbf{0.251_{\pm 0.001}}$	$\mathbf{0.251_{\pm 0.000}}$	$0.255_{\pm 0.000}$	$0.274_{\pm 0.000}$
ExchangeRate	96	$\mathbf{0.252_{\pm 0.010}}$	$0.264_{\pm 0.000}$	$0.308_{\pm 0.010}$	$0.269_{\pm 0.012}$	$0.376_{\pm 0.031}$	$0.488_{\pm 0.003}$
	192	$\mathbf{0.329_{\pm 0.001}}$	$0.335_{\pm 0.000}$	$0.415_{\pm 0.002}$	$0.419_{\pm 0.010}$	$0.513_{\pm 0.010}$	$0.585_{\pm 0.008}$

Table D.4: Performance comparison between the baseline `Moment` model without adapters against different adapter architectures (`PCA`, `LinearAE`, `dropoutLAE`, `LinearVAE`, and `VAE`), for multivariate long-term forecasting with different horizons H. We display the average test MAE $\pm$ standard error obtained on 3 runs with different seeds. **Best** results are in bold, with lower values indicating better performance.

Appendix E

Appendix of Chapter 6

E.1 Extended related work

PG vs MLE for Generative models. Generative models aim to capture the underlying distribution of observed data, with the goal of synthesizing realistic samples afterwards, e.g. text generation [Bro+20] and image generation [Rom+21; Ram+]. Many of the existing generative modeling approaches such as Autoregressive models [Rad+19; Vas+23], Variational AutoEncoders [KW13; Hig+17], Generative Adversarial Networks [Goo+14; ACB17], Diffusion Models [Soh+15; Rom+21], can be framed through the lens of MLE or its approximations. However, and especially in the context of sequence generation, MLE in autoregressive models has been proven to suffer from compounding errors and exposure bias, among other problems [Tan+19; Bah+17; Ran+16; Ben+15b; VHB15a; Ben+24]. As an alternative approach, PG methods have emerged as a more effective way to sample the output space when a reward function is available [Bah+17]. Beyond vanilla PG, more sophisticated methods have been developed, such as Reward-Augmented Maximum Likelihood [Nor+16; VLZ11], where a reward-based stationary sampling distribution is defined, Softmax Policy Gradient [DS17], an intermediate approach between sampling the model and sampling a reward-based distribution, and MIXER [Ran+16], a scheduling approach that gradually transitions from MLE to PG using the REINFORCE algorithm [Wil92a]. Besides autoregressive models, policy gradient methods have also been used to train (or finetune) Diffusion models [Bla+24; Ueh+24; ZB25], and GANs [PLB17; Yu+17].

Reward models. Policy Gradient methods constitute one class of algorithms for solving MDPs [Bel58], the central formalism underpinning the RL field. Training generative models with PG methods builds on the formulation of the task as an MDP. In this setting, the reward function plays a pivotal role. The most direct way of learning a reward model

is via supervised learning from past interactions, as done in *Model-based Reinforcement Learning* [Chu+18; Jan+19; Yu+20b; Haf+21b; KHT21; Ben+25]. Beyond the supervised approach, several other paradigms for reward learning have been developed. *Learning from Demonstrations* includes Inverse RL methods [AN04; Zie+08; Fin+16; FLA16] that learn a reward model R_θ under which demonstrations of the form (s, a, s_{next}) are optimal. Another paradigm, *Learning from Goals*, defines the reward function with respect to reaching a goal g in the state space $\mathcal{S}$ [LZZ22]. In this setting, goal attainment has been modeled in terms of spatial distances [Nac+18; Maz+24], temporal distances [Har+20; Wan+25], and semantic similarity [Son+23; Fan+22]. The *Learning from Preferences* approach relies on transforming preference data of the form $(\tau_0 \succ \tau_1)$, where τ_i is a trajectory $(s_1, a_1, \ldots, s_{|\tau_i|}, a_{|\tau_i|})$ and $\succ$ is a preference relationship, into a reward model using the *Bradley-Terry* model [BT52]. Reward models learned from preference data have enabled significant progress in *post-training* generative models [Kim+23; Tou+23; Raf+23; Son+24]. Starting with *InstructGPT* [Ouy+22], this approach has become a standard for improving targeted aspects of LLMs, e.g. safety [Dai+24], as well as for applications such as mathematical reasoning [Xin+25; Sha+24; Luo+24] and code generation [Dee+25].

Bilevel optimization. Bi-O was originally introduced in economics and game theory by [Sta34] to model hierarchical decision-making problems between a leader and a follower. More broadly, Bi-O offers a framework for addressing problems with hierarchical structures, where the task is to optimize two interdependent objective functions: an *inner-level* objective and an *outer-level* objective. In machine learning, Bi-O was first applied to feature selection [Ben+06] and was later extended to a wide spectrum of applications, including hyperparameter optimization [Mac+19; Fra+17; Ped16], reinforcement learning [Hon+22; Nik+21], and meta-learning [Fra+18]. Various Bi-O solvers have been proposed to address different regularity conditions on the inner- and outer-level objectives. Among these, *automatic differentiation*-based approaches compute gradients of the outer-level objective by differentiating through the iterative steps of the inner-level optimization algorithm [Wen64; Lin76; Dom12; Fra+17]. In parallel, *implicit differentiation* methods [Ben00] leverage the implicit differentiation theorem to approximately estimate the gradient of the outer loss by solving a linear system [Ped16; CSY21; JYL21; AM22]. Beyond alternating methods, [Dag+24] introduce a framework where inner- and outer-level variables evolve jointly within a single training loop. Bi-O has also been generalized to functional settings [PMA24], where the inner-level optimization is carried out over functions in infinite-dimensional spaces. In the context of generative models, some approaches enhance the training efficiency of energy-based latent variable models through bilevel formulations [Bao+20; Kan+22], while [Xia+25] propose a bilevel framework for tuning hyperparameters and noise schedules in diffusion models.

Bilevel Reinforcement Learning. Bilevel RL optimizes an outer-level objective in the reward parameters, often a policy alignment signal, while an inner loop learns a policy under that reward [Gau+25; SYC24; YGY25]. This framework has been applied in areas such as reward shaping [Zou+19] and RLHF [Cha+24]. The closest work to ours is [Zen+22], which combines MLE with the *Maximum-Entropy* inverse RL framework. However, they focus on control tasks in the episodic RL setting while we aim at providing a general framework for any data modality and MLE task.

E.2 Implementation Details

E.2.1 Training Framework

We implement our reinforcement learning framework using the Transformer Reinforcement Learning (TRL) library [Wer+20], which provides efficient implementations of modern RLHF algorithms. Specifically, we use the Group Relative Policy Optimization (GRPO) algorithm [Sha+24] through the `GRPOTrainer` class, which enables stable policy optimization through group-based advantage normalization.

GRPO is a policy gradient algorithm designed to improve the stability and sample efficiency of RL finetuning of LLMs. Unlike traditional policy optimization methods that compute advantages against a global baseline, GRPO employs a group-based normalization scheme that compares multiple generations from the same prompt.

For each training prompt x, GRPO generates K independent completions $\{y_1, y_2, \ldots, y_K\}$ using the current policy π_θ. Each completion is evaluated using a reward function $R(x, y_i)$, and the advantages are computed relative to the group mean:

$$A_i = \frac{R(x, y_i) - \mu_{\text{group}}}{\sigma_{\text{group}} + \epsilon}$$

where

$$\mu_{\text{group}} = \frac{1}{K}\sum_{j=1}^{K} R(x, y_j), \quad \sigma_{\text{group}} = \sqrt{\frac{1}{K}\sum_{j=1}^{K}(R(x, y_j) - \mu_{\text{group}})^2}$$

This group-based normalization provides several benefits: (1) it reduces variance by comparing samples from the same distribution, (2) it automatically adapts to the difficulty of each prompt, and (3) it mitigates reward hacking by preventing the policy from exploiting absolute reward values.

The policy is updated by maximizing the following objective function (omitting technical details such as likelihood ratios and clipping):

$$\mathcal{L}(\theta) = \mathbb{E}_{x\sim\mathcal{D},y\sim\pi_\theta(\cdot|x)}\left[A(x,y)\log\pi_\theta(y|x) - \beta\cdot d_{\mathrm{KL}}(\pi_\theta\|\pi_{\mathrm{ref}})\right]$$

where π_{ref} is the reference policy (typically the supervised fine-tuned model before RL training), β is the KL penalty coefficient that controls the deviation from the reference policy, and d_{KL} is the Kullback-Leibler divergence. The KL penalty serves as a regularization term that prevents catastrophic forgetting of the pretrained knowledge and ensures stable training dynamics.

E.2.2 Model Architecture and Fine-tuning

We evaluate multiple model sizes from the Qwen3 family [Yan+25], ranging from 0.6B to 8B parameters, to analyze the impact of model scale on reinforcement learning performance. For parameter-efficient fine-tuning, we employ Low-Rank Adaptation (LoRA) [Hu+22] with the configuration detailed in Table E.1.

Table E.1: LoRA Configuration for Parameter-Efficient Fine-tuning

Parameter	**Value**
LoRA Rank (r)	8
LoRA Alpha (α)	16
LoRA Dropout	0.05
Target Modules	q_proj, v_proj
Trainable Parameters	$\sim$0.1% of base model

The LoRA modules are applied exclusively to the query and value projection matrices in the attention mechanism, following recent findings that selective module targeting can achieve comparable performance to full fine-tuning while significantly reducing memory overhead [Zha+23a].

E.2.3 Dataset

We conduct our experiments on the `HuggingFaceTB/Countdown-Task-GOLD` dataset, a mathematical reasoning benchmark that requires models to construct arithmetic equations to reach target values using a given set of numbers. We specifically use the subset verified by `Qwen2.5-7B-Instruct`, which ensures groundtruth annotations. The dataset is split into 90% training and 10% validation sets.

Each training example follows a structured chat format with a system prompt that establishes the reasoning paradigm, followed by a user query specifying the task. An annotated example prompt is shown below:

```
[
  {
    "content": "You are a helpful assistant. You first think
               about the reasoning process in the mind and
               then provide the user with the answer.",
    "role": "system"
  },
  {
    "content": "Using the numbers [78, 46, 93], create an
               equation that equals 61. You can use basic
               arithmetic operations (+, -, *, /) and each
               number can only be used once. Show your work
               in <think> </think> tags. And return the
               final equation and answer in <answer> </answer>
               tags, for example <answer> (1 + 2) / 3 = 1
               </answer>.",
    "role": "user"
  },
  {
    "content": "<think>
               To solve this, I need to use the numbers 78,
               46, and 93 to create an equation that equals
               61. I'll start by considering the basic
               arithmetic operations and how they can be used
               to get close to 61.

               First, I'll try subtraction, as it's the most
               straightforward operation to get close to 61:
               - 78 - 46 = 32 (too low)
               - 93 - 46 = 47 (closer but still too low)
               - 93 - 78 = 15 (too low)

               Next, I'll try addition and subtraction in
               combination:
               - 78 - 93 = -15 (too low, and I need to add
                 something positive)
               - 46 + 93 = 139 (too high, and I need to
```

```
                  subtract something)

                Since 93 is the highest number, I'll try to
                use it in a subtraction:
                - 93 - 46 = 47 (still too low, I need to
                  add 14)
                - 78 - 46 = 32 (still too low, I need to
                  add 29)

                I can use 78 and 46 to get 32, and then add
                29 to get 61. 29 can be derived from 93 - 64,
                but since I can only use each number once, I
                need to find a way to use 93 directly.

                The correct combination is:
                93 - (78 - 46) = 93 - 32 = 61

                This uses each number exactly once and the
                result is 61.
                </think>
                <answer>
                93 - (78 - 46) = 61
                </answer>",
    "role": "assistant"
  }
]
```

The structured output format with `<think>` and `<answer>` tags enables clear separation between the reasoning trace and the final solution, facilitating both automated evaluation and reward computation during reinforcement learning. This format allows our reward functions to selectively evaluate either the complete response or only the final answer, depending on the experimental configuration (see Figure 6.6 for the comparison).

E.2.4 Training Hyperparameters

Table E.2 presents the core training hyperparameters used across all experiments. We use a cosine learning rate schedule with warmup to ensure stable optimization dynamics.

All experiments are conducted using mixed-precision training (bfloat16) with Flash At-

Table E.2: Core Training Hyperparameters

Parameter	Value
Learning Rate	5×10^{-5}
LR Schedule	Cosine
Warmup Ratio	0.05
Training Steps	400
Per-device Train Batch Size	4
Gradient Accumulation Steps	2
Effective Batch Size	8 per device
Mixed Precision	bfloat16

tention 2 [Dao23] for computational efficiency.

E.2.5 GRPO-Specific Configuration

The GRPO algorithm requires several algorithm-specific hyperparameters that control the policy optimization process. These are detailed in Table E.3.

Table E.3: GRPO Algorithm Hyperparameters

Parameter	Value
Number of Generations	8
KL Penalty Coefficient (β)	0.001
Max Prompt Length	256 tokens
Max Completion Length	1024 tokens
Temperature	1.0 (default)
Top-p Sampling	1.0 (default)

During training, the model generates 8 completions per prompt for group-based advantage estimation. The KL penalty coefficient β controls the divergence between the learned policy and the reference policy, preventing catastrophic forgetting of the pre-trained knowledge.

Task	Metrics	
	MSE$_{/10^{-2}\downarrow}$	**NLL**$_{/10^{-2}\downarrow}$
NLL		
simple	425 ± 3	$\mathbf{47 \pm 1}$
medium	459 ± 3	$\mathbf{73 \pm 1}$
expert	539 ± 3	$\mathbf{49 \pm 1}$
PG(I_n)		
simple	199 ± 1	528 ± 18
medium	241 ± 4	796 ± 70
expert	$\mathbf{174 \pm 3}$	420 ± 24
PG$(\mathrm{U}^{\star}_{\mathrm{he}})$		
simple	230 ± 1	267 ± 1
medium	274 ± 1	286 ± 1
expert	$\underline{198 \pm 1}$	290 ± 1
PG$(\mathrm{U}^{\star}_{\mathrm{im}})$		
simple	$\mathbf{190 \pm 1}$	$\underline{208 \pm 1}$
medium	$\mathbf{231 \pm 1}$	$\underline{232 \pm 2}$
expert	$\mathbf{176 \pm 1}$	$\underline{208 \pm 2}$

Table E.4: **MBRL experiment.** The PG loss with optimal reward comes <u>second</u> to the NLL baseline in terms of NLL, and ranks **first** in terms of MSE.

E.3 Additional experiments

E.3.1 Tabular regression (MBRL)

Model-Based Reinforcement Learning (MBRL) addresses the supervised learning problem of estimating the (possibly stochastic) transition function of a MDP. Typically, we assume access to data of the form $\mathcal{D} = \{(s_t^i, a_t^i, s_{t+1}^i)\}_{i=0}^N$, consisting of trajectories of states s and actions a collected by an unknown policy. The goal is to approximate the next-state distribution $S_{t+1} \mid S_t, A_t \sim q$. In practice, the dynamics model is often a Gaussian probabilistic model trained via log-likelihood [Chu+18; Jan+19], which makes it directly applicable to our experimental setup. We consider three D4RL [Fu+21b] *HalfCheetah* tasks, each from a different data-collecting policy: *simple*, *medium*, and *expert*, accessible through the *Minari* project [You+24]. All models train for 400 epochs with Adam optimizer (learning rate = 10^{-3}) and $\lambda = 1$.

Table E.4 shows the detailed results per task (data-generating policy) on the HalfCheetah system.

E.3.2 Distribution comparison

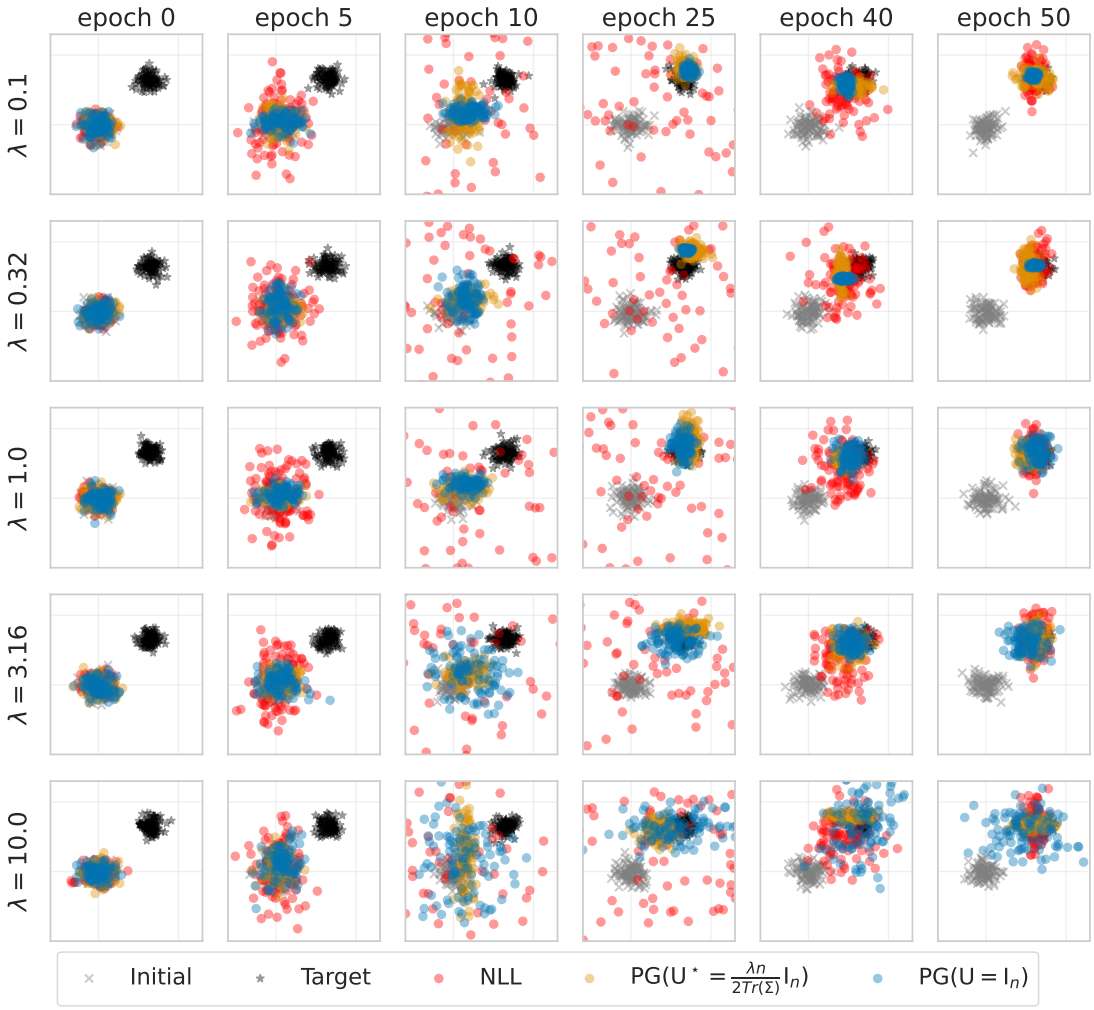


Figure E.1: Distribution comparison, different value of λ

E.4 Theoretical analysis

E.4.1 Proof of proposition 6.4.1

We prove proposition 6.4.1 in two steps: first solving the inner-level problem in closed form, then optimizing the outer-level objective.

Step 1: Solution of the inner-level problem

Proposition E.4.1. *Under assumption 6.4.1 and definition 6.3.2, the inner-level optimization problem*

$$\theta_U^\star = \arg\max_{\theta\in\Theta} \mathbb{E}_{X,Y\sim q}\Big[\mathbb{E}_{\widehat{Y}|X\sim\hat{p}_\theta}\Big[-(\widehat{Y}-Y)^\top U(\widehat{Y}-Y)\Big] + \lambda\mathcal{H}(\hat{p}_\theta)\Big]$$

admits exactly one solution: $\theta^\star(U) = \left(\Lambda,\ \frac{\lambda}{2}U^{-1}\right)$.

Proof. Since $\widehat{Y} \mid X \sim \mathcal{N}(AX, B)$, its differential entropy is $H(\widehat{Y} \mid X) = \frac{1}{2}\log(2\pi e \det(B))$. Using the identity $\mathbb{E}[W^\top MW] = \mu^\top M\mu + \operatorname{Tr}(MK)$ for $W \sim \mathcal{N}(\mu, K)$ and expanding the quadratic form, a direct calculation yields:

$$J(A,B) = -\operatorname{Tr}\Big((A-\Lambda)^\top U(A-\Lambda)\Sigma_X\Big) - \operatorname{Tr}(U(B+\Sigma)) + \frac{\lambda}{2}\log(2\pi e\det(B)), \tag{E.1}$$

where $\Sigma_X = \mathbb{E}[XX^\top]$.

Strict concavity. We verify that J is strictly concave in (A, B). Consider a perturbation $(A_1 + tA_2,\ B_1 + tB_2)$ for t in the feasible set. The quadratic term in A contributes a second derivative $-2\operatorname{Tr}(UA_2\Sigma_X A_2^\top) \le 0$, since $U, \Sigma_X \succeq 0$ implies $\operatorname{Tr}(UC\Sigma_X C^\top) \ge 0$ for any C. The log-determinant term contributes a second derivative proportional to $-\operatorname{Tr}((B_1+tB_2)^{-1}B_2(B_1+tB_2)^{-1}B_2) < 0$ by strict positive definiteness. Hence J is strictly concave and admits a unique maximizer.

First-order conditions. Setting $\nabla_A J = -2U(A-\Lambda)\Sigma_X = 0$ gives $A^\star = \Lambda$. Setting $\nabla_B J = -U + \frac{\lambda}{2}B^{-1} = 0$ gives $B^\star = \frac{\lambda}{2}U^{-1}$. □

Step 2: Solution of the outer-level problem

Proof of proposition 6.4.1. From proposition E.4.1, substituting $\theta^\star(U) = (\Lambda, \frac{\lambda}{2}U^{-1})$ into the outer objective, both distributions share the mean ΛX, so the KL divergence between $q(\cdot|X) = \mathcal{N}(\Lambda X, \Sigma)$ and $\hat{p}_{\theta_U^\star}(\cdot|X) = \mathcal{N}(\Lambda X, \frac{\lambda}{2}U^{-1})$ reduces to:

$$\mathrm{D}_{\mathrm{KL}}\Big(q \,\|\, \hat{p}_{\theta_U^\star}\Big) = \frac{1}{\lambda}\operatorname{Tr}(U\Sigma) - \frac{1}{2}\ln\det(U) + C,$$

where C collects terms independent of U. Denote this objective $\varphi(U)$.

Strict convexity. To show φ is strictly convex on $S_n^{++}(\mathbb{R})$, note that along any line $U+tV$ in $S_n^{++}(\mathbb{R})$, the second derivative of φ is $\frac{1}{2}\operatorname{Tr}((U+tV)^{-1}V(U+tV)^{-1}V)$. Writing $B = (U+tV)^{-1/2}V(U+tV)^{-1/2}$, this equals $\frac{1}{2}\operatorname{Tr}(B^2) = \frac{1}{2}\sum_i \lambda_i^2 > 0$, confirming strict convexity and hence a unique minimizer.

Optimality condition. Setting $\nabla_U \varphi = \frac{1}{\lambda}\Sigma - \frac{1}{2}U^{-1} = 0$ yields $U^\star = \frac{\lambda}{2}\Sigma^{-1}$. □

E.4.2 Proof of corollary 6.4.1

Proof. When $B = \sigma^2 I_n$, the inner-level objective (E.1) is optimized over (A, σ^2) instead of (A, B). The same concavity arguments apply. The first-order condition $\nabla_{\sigma^2} J = 0$ gives $\sigma^{2\star} = \frac{\lambda n}{2\,\mathrm{Tr}(U)}$, so $\theta^\star(U) = (\Lambda, \frac{\lambda n}{2\,\mathrm{Tr}(U)})$. Substituting into the outer-level KL divergence and optimizing over U yields the condition $\mathrm{Tr}(U) = \frac{\lambda n^2}{2\,\mathrm{Tr}(\Sigma)}$, which defines the solution set $F_{\lambda,\Sigma}$. □

E.4.3 Proof of corollary 6.4.2

Proof. Substituting $U^\star = \frac{\lambda}{2}\Sigma^{-1}$ into the inner-level objective, we show the reward term reduces to a log-likelihood under q. Writing $Y = \Lambda X + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, \Sigma)$ and expanding:

$$\mathbb{E}_{Y|X}\Big[-\tfrac{\lambda}{2}(\widehat{Y} - Y)^\top \Sigma^{-1}(\widehat{Y} - Y)\Big] = -\tfrac{\lambda}{2}\Big[(\widehat{Y} - \Lambda X)^\top \Sigma^{-1}(\widehat{Y} - \Lambda X) + n\Big],$$

where the cross-term vanishes since $\mathbb{E}[\varepsilon] = 0$ and $\mathbb{E}[\varepsilon^\top \Sigma^{-1}\varepsilon] = n$. Recognizing that $(\widehat{Y} - \Lambda X)^\top \Sigma^{-1}(\widehat{Y} - \Lambda X) = -2\log q(\widehat{Y}|X) - \log((2\pi)^n|\Sigma|)$, we obtain up to a constant c_n independent of θ:

$$J(\theta) = \lambda\,\mathbb{E}_X\Big[\mathbb{E}_{\widehat{Y}\sim\hat{p}_\theta}\Big[\log q(\widehat{Y}|X) - \log \hat{p}_\theta(\widehat{Y}|X)\Big]\Big] + c_n = -\lambda\,\mathbb{E}_X[\mathrm{D_{KL}}(\hat{p}_\theta(\cdot|X)\|q(\cdot|X))] + c_n.$$

Hence maximizing $J(\theta)$ is equivalent to minimizing the reverse KL divergence. □