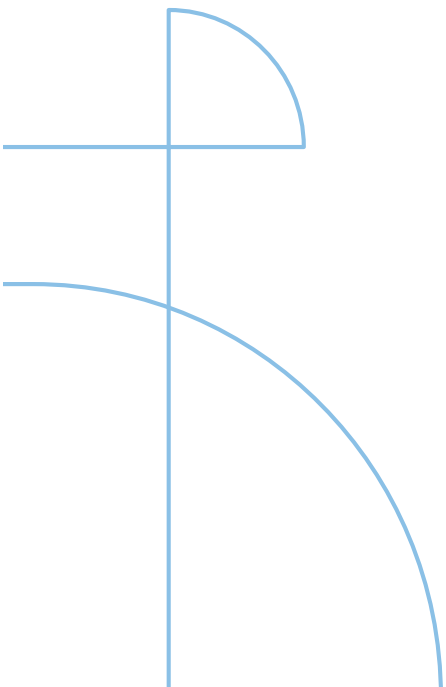
Doctoral Thesis in Computer Science

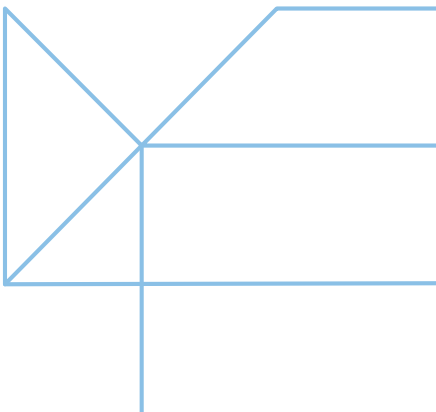
Interactive Representation Learning

Symmetries, Metric Spaces and Uncertainty

ALFREDO REICHLIN

KTH ROYAL INSTITUTE OF TECHNOLOGY





Interactive Representation Learning

Symmetries, Metric Spaces and Uncertainty

ALFREDO REICHLIN

Academic Dissertation which, with due permission of the KTH Royal Institute of Technology, is submitted for public defence for the Degree of Doctor of Philosophy ontodo in todo.

Doctoral Thesis in Computer Science
KTH Royal Institute of Technology
Stockholm, Sweden 2025

TRITA-EECS-AVL-2025:XX
ISBN xxx-xx-xxxx-xxx-x

Typeset in LaTeX2e (2025-06-01).
Printed by Universitetsservice US-AB, Sweden 2025

Abstract

This thesis investigates how *interaction* can be used as self-supervision to learn structured state representations that simplify downstream tasks. We formalize two inductive biases naturally present in the trajectories generated by agents that interact with their environment: geometry and temporal consistency of the underlying state space. We show that injecting these biases into representation learning yields additional, task-relevant properties. First, we focus on geometric bias: we learn translationally equivariant latent spaces from images in which agent actions correspond to vector additions. We show how these representations can be used to estimate a recovery policy that mitigates the compounding of error in data-driven sequential decision-making policies. We further extend equivariant representations to scenes with external objects. Under an interaction-by-contact model, we prove that aligning the object's and the agent's latent embeddings yields an isometric, disentangled representation of both. Second, we relax the geometry assumption and explore the milder temporal consistency bias. This allows us to construct representations where the temporal order between states is preserved, a property we refer to as *distance monotonicity*. In the reinforcement learning setting, we show that, under suitable conditions, this property is enough to recover an approximation of the value function and provably estimate an optimal policy. In a multiple-sensor framework, these representations can be used to construct a Bayesian filtering state estimate robust under unknown noise. Lastly, we extend the concept of interactions from physical systems to the parametric space of a learner. We show how distance monotonic representations of the parameters of a model can be used to approximate the posterior distribution of a Bayesian neural network. Finally, in a meta-learning setting, we explore implicit representations of the learner to reduce the variance of a fast-adaptation model. Collectively, these results demonstrate that interaction-driven biases produce structured representations that simplify or enhance the learning process.

Sammanfattning

Denna avhandling undersöker hur *interaktion* kan användas som självövervakning för att lära strukturerade tillståndsrepresentationer som förenklar nedströmsuppgifter. Vi formaliserar två induktiva bias som naturligt uppstår i trajektorier genererade av agenter som interagerar med sin omgivning: geometri samt temporal konsistens i det underliggande tillståndsrummet. Vi visar att införandet av dessa bias i representationsinlärning ger ytterligare, uppgiftsrelevanta egenskaper. Först fokuserar vi på geometrisk bias: vi lär translationsekvivarianta latenta rum från bilder där agentens handlingar motsvarar vektoradditioner. Vi visar hur sådana representationer kan användas för att estimera en återhämtningsstrategi som dämpar felackumulering i datadrivna, sekventiella beslutspolicys. Vi utvidgar därefter ekvivarianta representationer till scener med externa objekt. Under en kontaktbaserad interaktionsmodell bevisar vi att en inriktning (alignment) av objektets och agentens latenta inbäddningar ger en isometrisk och separerad (disentangled) representation av båda. Därefter lättar vi på geometriantagandet och studerar den mildare biasen temporal konsistens. Detta möjliggör konstruktion av representationer där den temporala ordningen mellan tillstånd bevaras—en egenskap vi benämner *distansmonotonicitet*. I en förstärkningsinlärningsmiljö visar vi att denna egenskap, under lämpliga villkor, räcker för att återvinna en approximation av värdefunktionen och bevisligen skatta en optimal policy. I ett flersensorramverk kan dessa representationer dessutom användas för att konstruera en Bayesiansk filtreringsbaserad tillståndsskattning som är robust mot okänt brus. Slutligen utvidgar vi interaktionsbegreppet från fysikaliska system till en lärares parametriska rum. Vi visar hur distansmonotona representationer av modellparametrar kan utnyttjas för att approximera posteriordistributionen i en Bayesiansk neuronnätmodell. I en meta-inlärningssättning undersöker vi även implicita representationer av läraren för att minska variansen hos en modell för snabb anpassning. Sammantaget demonstrerar resultaten att interaktionsdrivna bias leder till strukturerade representationer som förenklar eller förbättrar inlärningsprocessen.

Acknowledgment

TODO.

Alfredo Reichlin
Stockholm, November 29, 2025

Contents

Acronyms

BC Behavioral Cloning 15–17

BNN Bayesian Neural Network 32–34

DM Distance Monotonic 25–30, 34, 35, 38, 42

GBML Gradient-Based Meta-Learning 36–39, 42

i.i.d. Independent and Identically Distributed 4

IL Imitation Learning 15

MAP Maximum a Posteriori 36

MCMC Markov Chain Monte Carlo 33

MDN Mixture Density Network 16

MSE Mean-Squared Error 36–38

OOD Out-of-Distribution 16, 17

POMDP Partially Observable Markov Decision Process 5, 8, 21, 24, 26–28, 42

RL Reinforcement Learning 21–23, 26–28, 30

UVFA Universal Value Function Approximators 27

VI Variational Inference 33

Introduction

1 Introduction

This thesis studies structures and properties that can be recovered when working with datasets of interactions. In many real-world settings, learning does not happen in a vacuum. Instead, it is often the result of knowledge accumulation from data collected within a particular environment. In practice, these data are generated by some agent interacting with and observing the environment, producing trajectories with rich temporal and causal structure. Treating such data as independent discards inductive biases that can make learning of downstream tasks substantially easier, e.g., state estimation, prediction, control. This work develops on *interactive representations*: latent spaces whose geometry is constrained to mirror the geometry induced by actions and dynamics. The central premise is: if actions transform the world in a structured, often low-dimensional way, then useful representations should preserve (or deliberately relax) that structure.

The view that learning is shaped by *interaction*, not passive exposure, is well established across biology, cognitive science, causality, machine learning, and robotics. Classic sensorimotor deprivation studies show that coupling between self-generated movement and sensation is necessary for normal perceptual development: kittens prevented from moving do not acquire visually guided behavior [1]. More broadly, action is fundamental to perception, shaping, and in some cases constituting, basic perceptual categories (e.g., roundness) [2, 3]. In causal inference, interaction, often termed *intervention*[1], is required to uncover mechanism: Pearl's calculus distinguishes observation from intervention to identify causal relations [4]. This perspective informs representation learning, where disentanglement typically needs auxiliary signals or interventions and is provably unidentifiable in the purely unsupervised setting without inductive bias [5]. In machine learning, interaction is operationalized directly: reinforcement learning exploits self-play and environment feedback [6], DAgger queries experts on the learner's own state distribution to avoid error compounding [7], and removing the abil-

[1] In the causal literature, interactions that set or perturb variables are formalized as interventions.

ity to interact (offline reinforcement learning) exposes severe distribution-shift and extrapolation issues [8]. Human feedback provides another interactive channel, turning preferences into learning signals for large language models [9, 10]. In robotics, *active* and *interactive* perception make action the central component of sensing—moving the sensor or probing the environment yields more informative measurements and reveals latent object properties [11, 12, 13]. Together, these lines of work converge on a single principle: the trajectory of learning depends critically on how the learner probes and perturbs its world.

Continuing on this same direction, we frame the thesis in the following idea: data generated by acting in the world are not Independent and Identically Distributed (i.i.d.) but carry *interaction-induced* regularities. Here, we claim that learning should expose and exploit these regularities by building *interaction-aware representations* in which downstream prediction and control become simpler, more data-efficient, and more robust. In particular, we reduce the scope and focus on two specific regularities stemming from interactions.

Geometric consistency. Physical laws constrain how states can vary in the real world: many transformations are structured (e.g., translations, rotations, scalings). Symmetries and invariants induce a *geometry* in the state space. When a sensor map converts states into observations, these geometric regularities are imprinted—possibly distorted—onto the *observation manifold*. In practice, these constraints are often available a priori and can be used to inject suitable biases in the learning system.

Temporal consistency. A milder form of supervision arises from the temporal ordering of the collected data. When the agent interacts with its environment, information is collected in the form of trajectories. This ordering defines implicit structures that can be used to guide the learning process with a weaker form of prior knowledge.

These different regularities require different learning objectives to effectively exploit their information. Moreover, the structures that can be built differ substantially. In the thesis, we additionally explore a number of properties that arise from the different structures induced by these regularities. Specifically, we investigate the following research questions:

- **RQ1 (Process).** How should the learning objective change in a machine learning framework when data is not i.i.d. but rather the result of some generative process, i.e., interaction with the learning environment?
- **RQ2 (Properties).** What kind of properties can be recovered when injecting different forms of structure in the learning system?

1.0.1 A Motivating Example

Robotics, in particular, offers us a very interesting testing ground as it allows us to embody the learning agent in a physical environment. Consider a general robotic arm with n-degrees of freedom able to move within its environment. To perceive the external environment, assume an RGB camera captures the robot and its surroundings. Having access to the morphology and kinematics of the robot allows us to convert desired end-effector movements into the required joint commands. Suppose the robot is the only source of variation (static background, fixed lighting/camera, no external disturbances). Even a simple Cartesian translation of the end-effector typically induces a highly non-linear transformation in image space (perspective effects, depth discontinuities, self-occlusions), often without a tractable closed-form pixel mapping, Figure 1.0.1. This observation summarizes the core idea of this thesis. Often, even though the change in the physical system can be relatively simple and low-dimensional (translation of the end-effector), the data collected to describe such a system can be unnecessarily complex (images of the scene). However, because the only factors of variation are fully determined by the low-dimensional movement of the end-effector, the intrinsic dimensionality of a dataset of images collected in this setup has to be equivalent to the one of the end-effector.

Figure 1.0.1: Small Cartesian translations of the end-effector can induce complex, non-linear image-space deformations.

1.1 Framework

The concept of interactions can be framed as the evolution of a general dynamical system. We can assume the existence of a system described by its state, $s \in \mathcal{S}$, and perturbed by actions $a \in \mathcal{A}$ that evolves according to a governing equation $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. Whenever specified otherwise, we'll focus on the particular case of a deterministic, discrete-time, autonomous and controllable system. Moreover, we'll assume access to sensory information, $o \in \mathcal{O}$, on the state of the system generated by an emission function depending on the current state, $\omega : \mathcal{S} \to \mathcal{O}$. When the system is Markovian (the state is fully determined by s), we can refer to this model as a Partially Observable Markov Decision Process (POMDP) [14]. Describing with s' the future state, we can summarize the system as follows:

$$\begin{cases} s' = f(s, a), \\ o = \omega(s). \end{cases} \tag{1.1.1}$$

In this context, interactions are described by the influence of the actions in the evolution of the system. We can further complement this by introducing an additional map generating the actions which correspond to the policy of the agent, i.e., $\pi : \mathcal{O} \to \mathcal{A}$. When collecting a dataset of observations, we are implicitly assuming the execution of a set of actions within the system and the recording of the resulting observations. Accordingly, an interaction dataset is a finite collection of one or more trajectories, K, of length H_k:

$$\mathcal{D} = \{ (o_i, a_i, o_i') \}_{i=1}^{N}, \quad N = \sum_{k=1}^{K} H_k. \tag{1.1.2}$$

As stated in the previous section, the actions performed by the system are, in practice, available and allow us to treat the data non-independently. Possible goals include:

- State estimation from sensory observations when the state is not directly observable.
- Forward dynamics prediction.
- Policy estimation for achieving goal states.

Two of the main issues often encountered in practical systems are:

- Unknown and non-linear f, ω and π.
- High-dimensional and unstructured sensory information o.

Data-driven approaches can offer a solution to these problems, e.g., learning non-linear dynamics [15], learning representations of high-dimensional data [16]. In this case, the subject of study changes. Instead of formulating explicit solutions, the goal becomes designing optimization procedures over parametric models. Data in this context is, however, generally high-dimensional and unstructured, making the learning procedure complex and expensive. In the example of the robotic arm, the transformation in the pixel space is considerably more complex than the actual movement of the end-effector. A dataset collected in this scenario would necessarily reflect the degrees of freedom of the end-effector movement (in this case, three-dimensional) [17]. This observation–dynamics mismatch motivates learning structured, low-dimensional representations that expose the underlying geometry and simplify downstream control.

1.2 Representation Learning

While learning might refer to the general problem of parametric function approximation, here we turn our attention to the specific technique of representation learning, [16]. It refers to a class of methods to extract semantic information from unstructured data and organize it to ease downstream tasks. More precisely, a representation is a map $\varphi : \mathcal{O} \to \mathcal{Z}$ from observations to a latent space $\mathcal{Z} \subseteq \mathbb{R}^n$ in which subsequent tasks are solved by (typically simpler) maps acting on $\mathcal{Z}$, i.e., $\psi : \mathcal{Z} \to \mathcal{Y}$ (assuming $\mathcal{Y}$ is the output space of the downstream task). In deep learning, φ is commonly a neural network trained by optimizing a task or proxy objective on data [16]. When learning a downstream task, the parametric function can be constructed as the composition of a

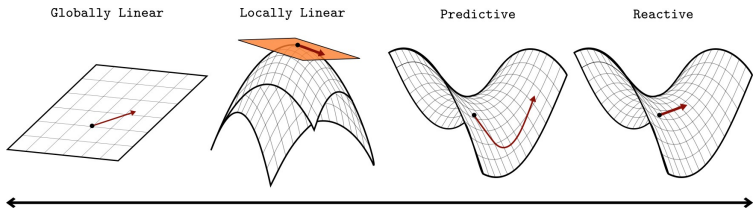


Figure 1.2.2: Representation - control tradeoff. The geometry of the representation space can simplify the control problem and vice versa.

representation and a downstream learner, i.e., $\psi \circ \varphi$. Modern feedforward architectures used in deep learning can already be seen as the composition of multiple hidden layers. When these are learned solely by optimizing the downstream loss function, we can refer to the intermediate layers as implicit representations. On the other hand, when auxiliary objectives are optimized along with the main one (downstream task) on a specific set of layers, we can refer to this map as a structured representation.

Functionally, this is equivalent to learning directly the task at hand (implicit representations), given enough expressivity in the two maps (universal approximation of neural networks [18, 19]). Structured representations have been shown to provide additional inductive properties that extend beyond task performance. Respecting the symmetries of the environment can improve generalization in reinforcement learning systems, [20, 21]. In transfer learning, large pretrained models demonstrate structured embeddings to generalize across domains, [22, 23, 24]. Beyond transfer, explicit latent structure can allow for specific operations in the learned embedding space, e.g., sampling from approximated variational distributions [25], algebraic operations on the semantics of the embeddings [26], planning and control [27, 28].

1.2.1 Representation vs. Control Trade-off

The robotic example presented before, Section 1.0.1, can be used to illustrate the potential benefits of structured representations. Suppose we are aiming to learn a policy to move the robot's end-effector to a desired position using the images as sensory feedback. This policy necessarily requires an estimate of the current state of the robot when modeled closed loop. This can either be implicit in the case of end-to-end control or explicit. In the simplified case of a linearly observable state and linear dynamics, the optimal controller could be decomposed into a state estimator (Kalman filter) and a state-feedback controller (LQR)[2]. This is also referred to as the separation principle or Kalman duality, [29, 30]. In our setting, raw images rarely satisfy these structural assumptions, but a learned representation can bring us "closer" to them. A representation that estimates a latent space linear to the true state (and thus with globally linear dynamics) could allow for this. On the very opposite of the spectrum, with no explicit structure, the controller increases in complexity, e.g., end-to-end visuomotor policies [31]. Figure 1.2.2 depicts four main categories of structure. On the very left side of the tradeoff, we can place a representation that fully recovers the global linearity of

[2]Assuming quadratic cost.

the underlying dynamics. This can be relaxed into a locally linear alternative. Alternatively, the representation space can be kept arbitrarily non-linear but paired with a latent transition model to allow for planning algorithms (e.g., MPC [32]) or rely fully on an end-to-end reactive policy (far right on Figure 1.2.2). This is the essence of the representation–control trade-off: we can "pay" complexity in the encoder φ to simplify the controller ψ, or conversely, rely on a powerful controller to compensate for an unstructured φ, e.g., the identity map. In practice, investing structure in φ reduces sample complexity, improves robustness to perturbations, and makes downstream control more interpretable. The work presented in this thesis lies on the more structured side of this trade-off.

1.3 Contributions of the Thesis

In the rest of the thesis, we will present the work done on the topic. In Chapter 2 we will consider the first regularity from interactions, i.e., geometric consistency. We will assume full knowledge of the geometry of the interactions. This allows for learning isometric representations of the true state of the system, i.e., points in the learned latent space preserve the true distances. In **Paper A** we explore isometric representations for a robotic task. We additionally present a method to avoid the compounding of the error of an imitation learning policy through the use of this structured representation. In **Paper B** we extend this representation to include an external passive object that the agent can interact with. We show how this representation is agnostic on the nature of the observations and can be used for a downstream controller. Assuming full knowledge of the geometry of the interactions might often be unfeasible. In Chapter 3 we consider the milder bias of temporal consistency. **Paper C** presents a relaxation of isometries that we refer to as *distance monotonicity*, i.e., preserving the relative order of states but not necessarily their distance. Distance monotonicity allows us to compare triplets of states in terms of which one is closer. Here, the representation is a metric space embedding with an appropriate notion of distance. We additionally show that this representation can be used to approximate the optimal value function of a family of POMDPs and can be used in the context of offline reinforcement learning. In **Paper D**, we used distance monotonic representations to geometrically estimate uncertainties in noisy multimodal POMDPs. Finally, in Chapter 4, we generalize the concept of interactions to non-real systems. We study interactions as synthetic operations (that still carry a structure), and the goal is not the representation of a dynamical system but rather the parameters of a learner. In **Paper E**, we use temporal consistency to structure the posterior probabilistic distribution of a Bayesian neural network to improve sampling efficiency. An even milder form of structure is a binary relationship between datasets. **Paper F** explores the few-shot adaptation setting used in meta-learning. We proposed a structured latent space of the parameters of a learner to improve the adaptation estimator. We conclude the first part of the thesis with possible future works in Chapter 5.

1.3.1 List of Papers

List of the papers included in the thesis:

A: **A. Reichlin**, G. L. Marchetti, H. Yin, A. Ghadirzadeh, D. Kragic. *Back to the Man-*

ifold: Recovering from Out-of-Distribution States. In International Conference on Intelligent Robots and Systems (IROS), 2022, [33].

B: **A. Reichlin***, G. L. Marchetti*, H. Yin, A. Varava, D. Kragic. *Learning Geometric Representations of Objects via Interaction*. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML), 2023, [34].

C: **A. Reichlin**, M. Vasco, H. Yin, D. Kragic. *Goal-Conditioned Reinforcement Learning from Sub-Optimal Data on Metric Spaces*. Under Submission, 2025.

D: **A. Reichlin**, A. Pacciarelli, M. Vasco, D. Kragic. *Geometry of Uncertainty: Learning Metric Spaces for Multimodal State Estimation in RL*. Under Submission, 2025.

E: **A. Reichlin**, M. Vasco, D. Kragic. *Walking on the Fiber: A Simple Geometric Approximation for Bayesian Neural Networks*. In Transactions on Machine Learning Research (TMLR), 2025, [35].

F: **A. Reichlin***, G. Tegnér*, M. Vasco, H. Yin, M. Björkman, D. Kragic. *Reducing Variance in Meta-Learning via Laplace Approximation for Regression Tasks*. In Transactions on Machine Learning Research (TMLR), 2024, [36].

Additionally, a list of papers contributed by the author but not included in the thesis:

X-1: A. Longhini, M. Moletta, **A. Reichlin**, M. C. Welle, A. Kravberg, Y. Wang, D. Held, Z. Erickson, D. Kragic. *Elastic Context: Encoding Elasticity for Data-driven Models of Textiles*. In International Conference on Robotics and Automation (ICRA), 2023, [37].

X-2: A. Longhini*, M. Moletta*, **A. Reichlin**, M. C. Welle, D. Held, Z. E., D. Kragic. *Edo-net: Learning elastic properties of deformable objects from graph dynamics*. In International Conference on Robotics and Automation (ICRA), 2023, [38].

X-3: G. Tegnér*, **A. Reichlin***, H. Yin, M. Björkman, D. Kragic. *On the Subspace Structure of Gradient-Based Meta-Learning*. In the First Workshop of Pre-training: Perspectives, Pitfalls, and Paths Forward (ICML) 2022, [39].

X-4: R. Gieselmann, A. Longhini, **A. Reichlin**, D. Kragic, F. T. Pokorny. *DLO@Scale - A Large-Scale Meta Dataset for Learning Non-Rigid Object Pushing Dynamics*. In the Workshop of Physical Reasoning and Inductive Biases for the Real World (NeurIPS) 2021.

X-5: N. Rajabi, C. Chernik, **A. Reichlin**, F. Taleb, M. Vasco, A. Ghadirzadeh, M. Björkman, D. Kragic. *Mental face image retrieval based on a closed-loop brain-computer interface*. In International Conference on Human-Computer Interaction (HCII) 2023, [40].

X-6: F. Renard, C. Courtot, **A. Reichlin**, O. Bent. *Model-based reinforcement learning for protein backbone design*. In the Workshop on Generative and Experimental Perspectives for Biomolecular Design (ICLR) 2024, [41].

2 Preserving Distances

When the geometry of the agent-environment interaction is known, we can inject this knowledge as a geometric inductive bias and *recover isometric representations* of the underlying physical state, given some assumptions. We realize this by enforcing *equivariance* to the known group of transformations (e.g., translations of a robot end-effector). The chapter develops the mathematical background, formalizes the robotic setting, and presents two contributions: *Back to the Manifold* (**Paper A**), which uses translationally equivariant latent spaces to define a recovery policy that brings a behavior-cloned controller back in-distribution; and *Learning Geometric Representations of Objects via Interaction* (**Paper B**), which extends the equivariant formulation to include an external passive object with unknown dynamics and proves that an ideal learner recovers an *isometric*, disentangled agent-object representation purely from interaction data.

2.1 Mathematical Preliminaries

In this chapter, we formalize *interactions* as *group actions* and use them to induce structure in learned representations. The key idea of this chapter is that we assume to know how actions compose algebraically. By enforcing the same algebra in the learned latent space, we can recover an equivariant representation, that is, we can preserve how interactions compose in the latent space. Additionally, by assuming these interactions to be distance-preserving in the original space (e.g., translations, rotations), the learned latent representation will be isometric to the real world, under some mild assumptions.

2.1.1 Groups, Actions, and Equivariance

We can formalize the geometry of interactions via the concept of groups.

Groups. A *group* $(G, \circ)$ is a set G with a binary operation $\circ$ such that:

1. **Closure:** $g_1 \circ g_2 \in G$ for all $g_1, g_2 \in G$.
2. **Associativity:** $(g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$.
3. **Identity:** There exists $e \in G$ such that $e \circ g = g \circ e = g$.
4. **Invertibility:** For all $g \in G$ there exists $g^{-1} \in G$ with $g \circ g^{-1} = g^{-1} \circ g = e$.

In particular, here we are interested in continuous and differentiable groups, i.e., Lie groups. Examples of this are $(\mathbb{R}^n, +)$ (translations), $SO(n)$ (rotations), and $SE(3)$ (rigid-body motions).

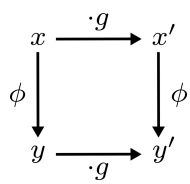
Group actions. An *action* of G on a set $\mathcal{X}$ is a map $\cdot : G \times \mathcal{X} \to \mathcal{X}$ such that

$$e \cdot x = x, \qquad (g_1 g_2) \cdot x = g_1 \cdot (g_2 \cdot x).$$

The *orbit* of $x \in \mathcal{X}$ is $G \cdot x = \{g \cdot x : g \in G\}$; Actions formalize transformations of $\mathcal{X}$ with the exact algebra of their composition. A geometry can be described by the subgroup of transformations that preserve it. For example, the rigid-motion group $E(n)$ acts by distance-preserving transformations; lengths, angles, and distances are invariants of that action in Euclidean geometry.

Equivariance. Let $\mathcal{X}$ and $\mathcal{Y}$ carry actions of G. A map $\phi : \mathcal{X} \to \mathcal{Y}$ is *equivariant* if

$$\phi(g \cdot x) = g \cdot \phi(x), \qquad \forall g \in G,\ x \in \mathcal{X}.$$



Equivariance intertwines the two actions, ensuring that transforming the input and then encoding equals encoding and then transforming.

2.1.2 Metric Spaces, Embeddings, and Isometries

Equivariance ensures the algebraic structure of the interactions is preserved between representations. The properties that can be preserved in these structures vary depending on the nature of the original space and the geometry of the interactions. In this chapter, we will focus on the specific family of metric spaces.

Metric spaces. A *metric space* $(\mathcal{X}, d_{\mathcal{X}})$ consists of a set $\mathcal{X}$ equipped with metric $d_{\mathcal{X}} : \mathcal{X} \times \mathcal{X} \to [0, \infty)$ such that for all x, y, z in $\mathcal{X}$:

1. $d_{\mathcal{X}}(x, y) = 0 \iff x = y$ (identity of indiscernibles),
2. $d_{\mathcal{X}}(x, y) = d_{\mathcal{X}}(y, x)$ (symmetry),
3. $d_{\mathcal{X}}(x, z) \leq d_{\mathcal{X}}(x, y) + d_{\mathcal{X}}(y, z)$ (triangle inequality).

Euclidean spaces with ℓ_2-norm, Riemannian manifolds with geodesic distance, and graphs with shortest-path distance are common examples.

Metric Space Embeddings. We can define an *embedding*, $\phi : (\mathcal{X}, d_{\mathcal{X}}) \to (\mathcal{Z}, d_{\mathcal{Z}})$, as an injective map between metric spaces. This embedding is deemed an *isometry* if:

$$d_{\mathcal{Z}}\left(\phi(x), \phi(x')\right) = d_{\mathcal{X}}(x, x') \quad \forall x, x' \in \mathcal{X}. \tag{2.1.1}$$

That is, the distance between any two points in $\mathcal{X}$ is the same in the latent space $\mathcal{Z}$.

Equivariance vs. Isometry on Euclidean spaces. Equivariance preserves the *algebraic* structure (how transformations compose); isometry preserves the *metric* structure (how distances measure change). When the group action preserves distances in the original space, i.e., $d_{\mathcal{X}}(g \cdot x, g \cdot y) = d_{\mathcal{X}}(x, y)\ \forall g \in G,\ \forall x, y \in \mathcal{X}$, equivariance can be related to isometries under mild assumptions. In particular, consider a Euclidean metric space $\mathcal{X} = (\mathbb{R}^n, \|.\|_2)$. We can define a second latent Euclidean metric space $\mathcal{Z} = (\mathbb{R}^n, \|.\|_2)$ and an appropriate continuous and injective, on the orbit, embedding $\phi : \mathcal{X} \to \mathcal{Z}$. Let the acting group on $\mathcal{X}$ and $\mathcal{Z}$ be the group of translations[1], i.e., $a \cdot x = x + a$. Enforcing ϕ to be equivariant results in the embedding being isometric as well on each connected reachable region [42, 43].

2.2 Equivariant Representation Learning

When we have access to the underlying algebraic structure of the interactions, we can enforce equivariance on a learned representation. Interestingly, this representation does not need to be an embedding of the true state of the system but can be defined on any observables (as long as they are injective to the state). For example, considering again the robotic arm scenario in Section 1.0.1, we can define a representation from the image space to a latent space because we know that the true state (in this case inaccessible) evolves according to translations. Using the notation of Chapter 1, we define a parametric representation $\varphi_\theta : \mathcal{O} \to \mathcal{Z}$. We can learn the parameters θ to enforce translational equivariance by minimizing the following objective on a pre-collected, task-agnostic interaction data $\mathcal{D}$:

$$\min_{\theta} \sum_{(o,a,o') \in \mathcal{D}} \|\varphi_\theta(o') - \varphi_\theta(o) - a\|_2^2. \tag{2.2.2}$$

This objective "straightens" pixel-space trajectories into *Euclidean translations* in latent space, thus preserving physical distances (scale is one-to-one with action magnitude). Intuitively, the loss in Equation (2.2.2) measures the distance between the

[1] This can be generalized to rigid motions, i.e., $SE(n)$.

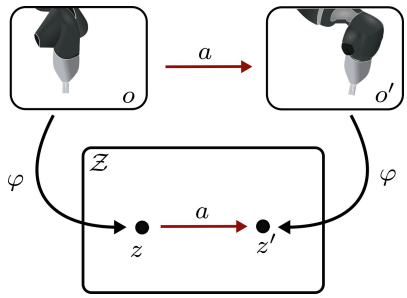


Figure 2.2.1: Encoding the image and translating the embedding should result in the embedding of the next image.

encoding of the observation of the next state ($\varphi(\omega(f(s,a)))$) and the transformation in the learned latent space induced by the same action from the current state ($\varphi(\omega(s))+a$) as these should be equivalent, Figure 2.2.1.

2.3 Robotic Setting

Consider the robotic manipulator operating in free space described in Section 1.0.1. It provides a practical scenario where the necessary assumptions presented so far are met. This setting will serve as the experimental ground for both **Paper A** and **Paper B**. Specifically, consider three simplifying assumptions:

1. The end-effector moves by *translations only* in $\mathbb{R}^3$.
2. The manipulator operates away from singularities.
3. There are no external interactions; the system evolves only according to the robot's actions.

This is equivalent to saying the robot moves freely according to a low-level position or velocity controller aware of its forward kinematics. Although these assumptions can be restrictive in practice for any real-world manipulation system, they allow us to fully determine the geometry of the actions of the system. On the other hand, these conditions allow us to move the complexity to the observation space as already hinted in Chapter 1. In fact, consider the case where observations are high-dimensional camera observations. The emission function, ω, is the unknown function mapping the position of the robot joints and the background of the scene into the image captured by the camera. Although the state transition is a simple translation, the corresponding transformation in the pixel space is complex and unknown. Nevertheless, the executed action a fully determines the transition of the images. The general system of Equation (1.1.1) can be described as follows:

- The **state space** is the 3D position of the end-effector, $s \in \mathbb{R}^3$.
- The **action space** is $a \in \mathbb{R}^3$, corresponding to translations.
- The **dynamics** are $s' = f(s,a) = s + a$.
- The **observations** are high-dimensional images $o \in \mathbb{R}^{p_H \times p_W \times 3}$.

Throughout the rest of the Chapter, we will assume access to a dataset of interactions in this setting in the form of Equation (1.1.2). Moreover, we can assume access to a representation module trained to minimize the objective described in Equation (2.2.2).

2.4 Imitation Learning

Estimating the necessary movements of the robot to perform a manipulation task can be a complex process. This is especially true when the feedback received is high-dimensional camera observations of the scene. In this case, we can define the downstream task as estimating a parametric policy $\pi_\theta : \mathcal{O} \to \mathcal{A}$ that can guide the robot accordingly, e.g., move an external object from one position to another. Data-driven approaches represent a possible solution and have been recently gaining more attention [44, 45, 46]. While requiring expensive and potentially dangerous training and lacking stability and interpretability guarantees, they can offer unmatched flexibility. As long as data is accessible, a policy can be estimated independently of the complexity of the problem and of the complexity of the sensory feedback.

One particular family of these methods is Imitation Learning (IL) [47]. IL frames the learning problem as a supervised one. It assumes access to a dataset of expert demonstrations in the form of trajectories, i.e., $\mathcal{D}_{\text{exp}} = \{(o, a^*)_{\times H}\}_{\times K}$. Behavioral Cloning (BC) in particular, aims at learning a policy to replicate the behavior of the expert demonstrations via explicit regression. For continuous control, the objective, $\mathcal{L}$, can be defined as the minimization of the squared error between the parametric policy approximation and the actual expert's actions as follows:

$$\mathcal{L} = \sum_{(o,a^*)\in\mathcal{D}_{\text{exp}}} \|\pi_\theta(o) - a^*\|_2^2. \tag{2.4.3}$$

Interestingly, while being a relatively straightforward learning process, BC provides a solution for state estimation and control estimation simultaneously. This flexibility, however, can induce a large function-approximation burden: the mapping from observations to actions is highly nonlinear and context-dependent (appearance, lighting, distractors). Small pixel perturbations can correspond to large changes in the latent state. As a result, high-capacity models (e.g., deep neural networks) can exhibit significant approximation errors.

Compounding error. A policy learned over a dataset of demonstrations (as in Equation (2.4.3)) can be expected to approximate relatively well the true expert's actions when the input is "close" to the training data distribution. As per any statistical model, however, the approximation error is never exactly zero. In sequential decision-making problems, this error can accumulate over every time step. In fact, when a learned policy is deployed, every state encountered will be the result of the action estimated in the time step before. As a result, every time the policy estimates an action, this approximation error will drift the state of the system away from the training distribution. As the current state moves further away, the approximation error generally increases, thus producing an even stronger drift. This is of particular relevance when the time horizon of the policy increases substantially. This phenomenon is often called the compounding of error [7] or distribution shift [8].

2.5 Stable Imitation Learning

In **Paper A**, we explore the first property that can be derived from equivariant representations φ, as described in Section 2.2. We propose a method to increase the stability of a BC policy when the environment adheres to the assumptions made so far, e.g., the system described in Section 2.3. The goal is to modify the data-driven policy such that the resulting trajectory does not deviate "too much" from the data it has been trained on. Within this region, in fact, we can ensure the approximation error to be bounded and the trajectory not divergent. Let ρ be a notion of data density, that is, large on observations that are in-distribution and small Out-of-Distribution (OOD), blue shape in Figure 2.5.2. Stabilizing a BC policy amounts to keeping the trajectory in high-density regions (*invariance*) and, when it drifts out, pushing it back (*recovery*). Direct implementation of this can be difficult when working on high-dimensional and non-linear observations, i.e., in pixel space. Distances and angles do not necessarily correspond to any physical quantity or meaningful motion. Equivariant representations, on the other hand, can offer a solution. Distances between latent points, in fact, do correspond to the required action to transition from one to the other.

2.5.1 Density Estimation

To compute how in-distribution the current state of the system is, we propose the use of a density estimator. Although many techniques exist for density estimation, ranging from parametric models and kernel methods [48, 49] to autoregressive models [50] and normalizing flows [51], we adopt a Mixture Density Network (MDN) [52]. An MDN parametrizes the conditional density of the data as a mixture of K Gaussian distributions whose mixing weights w, means μ, and variances Σ are predicted by a neural network from the current context (e.g., an image of the scene):

$$\rho(z) = \sum_{i=1}^{K} w_i \mathcal{N}(z; \mu_i, \Sigma_i). \tag{2.5.4}$$

The possibility of conditioning the density on the initial state of the system is relevant. Assume, for example, that the robot needs to pick an object. The position of the object (visible in the conditioning image) would determine the density of the trajectories where the object is in the same position. MDN can be trained on the expert trajectories data by minimizing the negative log-likelihood. Moreover, the MDN is differentiable. Crucially, we compute the density of the dataset, not in the image space, but on the equivariant latent space output by φ. The estimated density can then be normalized using a sigmoid function, i.e., $\bar{\rho}(z) = 1/(1 + e^{-\frac{\rho(z)+\epsilon}{\tau}})$ where ϵ and τ are appropriate temperature and offset parameters.

2.5.2 Recovery Policy

The quantity $\bar{\rho}$ can be interpreted as the probability of the current state being in-distribution or not. Stabilizing the BC policy effectively translates to keeping the current state within regions where $\bar{\rho}$ is close to 1 and eventually recovering when it gets closer to 0. In practice, in **Paper A**, we propose to augment the policy by pairing it with an

additional one, here referred to as the *recovery policy*, π_R. We implement this via a linear combination of the two, scaled by $\bar{\rho}$:

$$\tilde{\pi}(o) = \bar{\rho}(\varphi(o))\pi(o) + (1 - \bar{\rho}(\varphi(o)))\pi_R(\varphi(o)). \tag{2.5.5}$$

Here $\tilde{\pi}(o)$ represents the new policy, $\bar{\rho}(\varphi(o))$ is the normalized density estimate in the equivariant space, $\pi(o)$ is the original BC policy and $\pi_R(\varphi(o))$ is the proposed recovery policy. Intuitively, Equation (2.5.5) states that the agent should follow mostly the BC policy when the current state is in-distribution and correct it with the recovery policy whenever the state drifts OOD.

The objective of the recovery policy is to output actions that would bring the agent towards regions of increased data density. That is, to ascend ρ. When the density is computed in the equivariant space, the gradient of the density estimate with respect to the latent state z corresponds to the action required to move toward the higher density state. As such, we can define the recovery policy as:

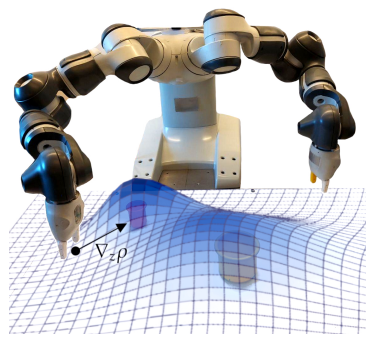


Figure 2.5.2: The recovery policy can be computed as the gradient over the estimated density of the training dataset in the equivariant space $\mathcal{Z}$.

$$\pi_R = \eta \nabla_z \rho(z), \tag{2.5.6}$$

where $z = \varphi(o)$ and η is a scalar parameter to modulate the strength of the recovery, Figure 2.5.2.

2.6 Extension to an External Object

So far, the main assumption is the knowledge of the agent's group action. This can be limiting in practice. In **Paper B**, we study an extension of the system described in Section 2.3 to include an external object and more complex forms of interaction. For example, when the agent collides with the object, the motion of the latter is generally difficult to describe in closed form. Nevertheless, we study how to recover structured representations when the active interactions are known, i.e., the agent's motion. In particular, we make a number of assumptions on the new extended system. We consider the state as the product of the state of the agent, $\mathcal{S}_{\text{int}} \subseteq \mathbb{R}^n$, and the state of the external object, $\mathcal{S}_{\text{ext}} \subseteq \mathbb{R}^n$. As before, we consider translations of the agent as the actions of the system and assume the external object to be passive, i.e., it evolves in time if and only if the agent interacts with it. In this new setting, we still have a strong prior knowledge of the geometry of the actions on the agent. Crucially, we relax the prior knowledge on the dynamics of the object. We consider the object to be abstracted to a single point[2]. If the agent, along its motion, intersects the position of the object (i.e., a collision), we

[2]We later explore objects with volumes.

allow the dynamics to be of arbitrary complexity and possibly even stochastic. Referring to $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}_{\text{ext}}$ as the transition of the object in case of collision, we can represent this with:

$$s'_{\text{ext}} = \begin{cases} s_{\text{ext}} & \text{if } s_{\text{ext}} \notin \lfloor s_{\text{int}},\ s_{\text{int}} + a \rfloor, \\ T(s, a) & \text{otherwise.} \end{cases} \tag{2.6.7}$$

In particular, we extend the previous system as follows:

- The **state space** is now the product of the state space of an agent, $\mathcal{S}_{\text{int}} \subseteq \mathbb{R}^n$, and an external object, $\mathcal{S}_{\text{ext}} \subseteq \mathbb{R}^n$, i.e., $\mathcal{S} = \mathcal{S}_{\text{int}} \times \mathcal{S}_{\text{ext}}$.
- The **action space** is again in $\mathbb{R}^n$. , which corresponds only to agent translations.
- The **dynamics** can be decomposed into the dynamics of the agent (i.e., $s'_{\text{int}} = s_{\text{int}} + a$) and of the object (i.e., Equation (2.6.7)).
- The **observations** are, again, high-dimensional images.

In **Paper B**, we additionally consider two more assumptions. The first is that the agent can reach any state via a sequence of actions. The second is that for every state of the object, an interaction is possible.

2.6.1 Object Localization

Under these assumptions, we ask whether one can learn a structured representation of an external object from interactions. We seek a joint representation of the agent and the object that preserves their relative distances, an isometric embedding of the physical world. We define $\varphi : \mathcal{O} \to \mathcal{Z} = \mathcal{Z}_{\text{int}} \times \mathcal{Z}_{\text{ext}}$ where $\mathcal{Z}_{\text{int}} = \mathcal{Z}_{\text{ext}}$ is a three-dimensional vector space and the distance is Euclidean. We aim at $\|z_{\text{int}} - z_{\text{ext}}\|_2 = \|s_{\text{int}} - s_{\text{ext}}\|_2$ for every state. The key idea is twofold: equivariance allows us to isolate the agent's state, and interaction events then localize the complementary component corresponding to the external object. In **Paper B** we show that this is possible and can be formalized by the following theorem:

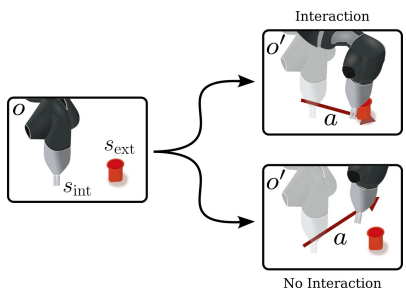


Figure 2.6.3: Actions of the agent can result in either interactions with the external object (top image on the left) or no interaction (bottom image on the left).

Theorem 1. *Suppose that the representation $\varphi : \mathcal{O} \to \mathcal{Z}$ satisfies:*

1. *φ is translational equivariant on $\mathcal{Z}_{int}$ (i.e., $z'_{int} = a + z_{int}$),*
2. *φ is injective,*
3. *for all $o \in \mathcal{O}$ and $a \in \mathcal{A}$ it holds that either $z'_{ext} = z_{ext}$ or $z_{ext} \in \lfloor z_{int}, z_{int} + a \rfloor$ where $(z_{int}, z_{ext}) = \varphi(o)$ and $(z'_{int}, z'_{ext}) = \varphi(a \cdot o)$.*

Then $\varphi \circ \omega$ is a translation i.e., there is a constant vector $h \in \mathbb{R}^n$ such that for all $s \in \mathcal{S}$ it holds that $\varphi(\omega(s)) = s + h$. In particular, $\varphi \circ \omega$ is an isometry w.r.t. the Euclidean metric on both $\mathcal{S}$ and $\mathcal{Z}$.

In **Paper B**, we additionally present a practical implementation of a representation to recover isometries. We propose decomposing the representation into two distinct components: $\varphi_{\text{int}} : \mathcal{O} \to \mathcal{Z}_{\text{int}}$ and $\varphi_{\text{ext}} : \mathcal{O} \to \mathcal{Z}_{\text{ext}}$. Having access to a dataset of interactions in the form of Equation (1.1.2), we can learn φ_{int} as before using the objective of Equation (2.2.2). Learning φ_{ext} requires a more challenging optimization design as we only have partial knowledge on the dynamics of the object. To do this, we rely on the conditions of interactions. Being the object passive, it should stay at rest if no collision with the agent happens. On the other hand, if the object's state does change, it means that in the time step before, its state was along the path of the agent. The loss objective can thus be split into these two cases:

$$\begin{cases} \mathcal{L}_- = d(z_{\text{ext}}, z'_{\text{ext}}) & \text{if no interaction,} \\ \mathcal{L}_+ = d(z_{\text{ext}}, \lfloor z_{\text{int}}, z_{\text{int}} + a \rfloor) & \text{if interaction.} \end{cases} \tag{2.6.8}$$

Here we refer to the distance in $\mathcal{L}_+$ as the distance between the object (i.e., a point) and the segment representing the translation on the agent (i.e., a set). This can be implemented in practice by considering the infimum of the distance between the object and the agent's trajectory.

Unfortunately, Equation (2.6.8) cannot be optimized directly as we do not have access to information on whether a contact has occurred or not. To this end, we propose a secondary representation, $\varphi_{\text{cont}} : \mathcal{O} \to \mathcal{W}$, to estimate it unsupervised. In fact, we can learn the parameters of φ_{cont} by forcing the embedding of subsequent observations, w, w', to lie closer than the embedding of random observations, w''. This can be achieved with a contrastive learning loss[3]:

$$\mathcal{L}_{\text{cont}}(o, o') = d_{\mathcal{W}}(w, w') + \log \mathbb{E}_{o''} \left[e^{-d_{\mathcal{W}}(w', w'') - d(z'_{\text{int}}, z''_{\text{int}})} \right]. \tag{2.6.9}$$

Intuitively, when an interaction does not occur, subsequent images will have encodings that lie closer on average in $\mathcal{W}$. Given a dataset of interactions, we can cluster the representation vectors of each tuple of images based on their relative distances in $\mathcal{W}$. We can then consider in the interaction class, $\mathcal{C}_+$, those tuples with greater distances and as a non-interaction class those with relatively small distances, $\mathcal{C}_-$. We implement this, in practice, via a two-means clustering technique (Otsu's algorithm [54]). Equation (2.6.8) can thus be rewritten as:

$$\begin{cases} \mathcal{L}_- = d(z_{\text{ext}}, z'_{\text{ext}}) & \text{if } (o, o') \in \mathcal{C}_-, \\ \mathcal{L}_+ = d(z_{\text{ext}}, \lfloor z_{\text{int}}, z_{\text{int}} + a \rfloor) & \text{if } (o, o') \in \mathcal{C}_+. \end{cases} \tag{2.6.10}$$

These three models, φ_{int}, φ_{ext}, φ_{cont}, can then be optimized altogether by means of a linear combination of their objectives.

[3] In the literature, this is often referred to as InfoNCE [53].

We additionally present an extension of the representation of the external object to include volume. This is possible by changing the output of φ_{ext} to be the parameters of a Gaussian distribution. It can be optimized as before by changing the notion of distance in Equation (2.6.8) to a Kullback-Leibler divergence. The learned covariance matrix can represent the volume of the external object.

2.7 Discussion

These two contributions illustrate how strong priors on the geometry of the state space can be exploited to recover *isometric latent representations* directly from high-dimensional observations. In the simple setting of **Paper A**, the equivariance imposed by the robot's actions suffices to recover the true geometry of the end-effector. The extension in **Paper B** shows that this idea can be generalized to represent an external object jointly with the robot by leveraging the structure of agent–object interactions.

Although the assumptions of known action geometry and contact-induced object motion are restrictive, they allow us to recover highly structured representations without supervision or auxiliary losses. In later chapters, we will progressively relax these assumptions, moving from strict isometries to more general metric-preserving representations applicable to complex environments.

3 Preserving Orders

Knowledge of the geometry can be restrictive for many real systems. In this chapter, we consider the bias of temporal consistency as an alternative to recover structured representations. In practice, this is a much milder bias, it allows us to model more complex forms of interactions without knowing a priori the symmetries of the system. This, however, comes at the cost of less structured representations. Here, we present a relaxation of isometric representations when the group of interactions is not known. The chapter introduces the necessary background on reinforcement learning and contrastive learning, describes a new objective for learning representations using the temporal consistency bias, and presents two contributions: *Goal-Conditioned Reinforcement Learning from Sub-Optimal Data on Metric Spaces* (**Paper C**), which uses this objective to approximate the value function of goal-conditioned offline Reinforcement Learning (RL) problems; and *Geometry of Uncertainty: Learning Metric Spaces for Multimodal State Estimation in RL* (**Paper D**), which uses these representations for state estimation under noisy multimodal settings.

3.1 Reinforcement Learning Background

Both contributions presented in this chapter are based on the RL formalism. As such, in this section, we provide an overview of some of the concepts used later on. For a comprehensive treatment, see [6].

Consider again the dynamical system introduced in Section 1.1. As already hinted at in the previous section, we can extend the framework to a POMDP defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, f, \omega, r, \gamma)$. As before, $\mathcal{S}$ represents the space of all states of the system, $\mathcal{A}$ the space of the actions, $\mathcal{O}$ the space of sensory observations, $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ the transition function[1], $\omega : \mathcal{S} \to \mathcal{O}$ the emission function. In addition, we are defining two more quantities needed to formalize the POMDP. With $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ we refer to a reward function and $\gamma \in [0, 1]$ is a scalar value representing the discount factor. We can actively

[1]This can be stochastic.

control the evolution of the system by interacting with it. The function representing the interaction strategy is referred to as the policy and is described by $\pi : \mathcal{S} \rightarrow \mathcal{A}$ or $\pi : \mathcal{O} \rightarrow \mathcal{A}$ depending on the availability of the actual state of the system. Unless stated otherwise, we assume state access and use the corresponding notation. Moreover, for the rest of the section, we will use the shorthand notation $\mathbb{E}_{\pi,f}[\cdot] \equiv \mathbb{E}_{\substack{a\sim\pi(\cdot|s) \\ s'\sim f(\cdot|s,a)}}[\cdot]$ for the expectation and $r_t \equiv r(s_t, a_t)$ for the reward function.

The aim is to estimate a policy that maximizes the expectation of the cumulative discounted reward $J(\pi)$. We refer to the policy that maximizes this quantity as the optimal policy:

$$\pi^* = \arg\max_{\pi} J(\pi) = \arg\max_{\pi} \mathbb{E}_{\pi,f}\left[\sum_{t=1}^{H} \gamma^t r_t\right]. \tag{3.1.1}$$

Following a policy induces trajectories over different states of the environment. We can measure the quality of a visited state under a policy, according to the reward function, with the value function $V^\pi : \mathcal{S} \rightarrow \mathbb{R}$. This can be computed as the expected cumulative discounted reward of starting in that state and performing actions based on the policy, i.e., $V^\pi(s) = \mathbb{E}_{\pi,f}\left[\sum_{t=1}^{H} \gamma^t r_t \mid s_1 = s\right]$. This quantity can be computed recursively via dynamic programming using the Bellman equation, [55]:

$$V^\pi(s) = \mathbb{E}_{\pi,f}[r_t + \gamma V^\pi(s')]. \tag{3.1.2}$$

The value function associated with the optimal policy is referred to as the optimal value function, $V^* = V^{\pi^*}$, and is proven to be always greater than or equal to the value function of any other policy [56], i.e., $V^*(s) \geq V^\pi(s) \quad \forall s, \pi$. Conversely, we can define a policy as greedy on a value function when $\pi_g^V = \arg\max_a V(f(s,a))$, that is, a policy that selects actions leading to successor states of maximal value. A greedy policy on the optimal value function is an optimal policy [56]. Dynamic programming offers a solution to the overall aim of RL. Algorithms such us Value Iteration [55], SARSA [6] and Q-learning [57] exploit the Bellman equation to iteratively estimate the optimal value function and induce optimal behavior.

The policy gradient family of methods offers an alternative approach to the problem. In this case, the optimal policy is computed directly without the need for a value function (and thus the Bellman equation). This can be done by estimating the policy parametrically directly on the main objective. That is, finding the parameters of the policy that maximizes the expected return (Equation (3.1.1)). However, the expected return is not directly optimizable as it depends on the policy 'only' in expectation. This dependency can be made explicit via the REINFORCE algorithm [58] by rewriting the gradient of the objective using the log trick as follows:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\pi_\theta,f}\left[\sum_t \underbrace{\left(\sum_{\tau=t}^{H} \gamma^{\tau-t} r_\tau\right)}_{R_t} \nabla_\theta \log \pi_\theta(a \mid s)\right]. \tag{3.1.3}$$

Modern RL often combines both perspectives in *actor–critic* methods, where a value (or advantage [59]) estimator serves as a baseline to reduce variance while the actor updates the policy. The most common algorithms follow this general pattern, e.g., PPO [60], DDPG [61], SAC [62].

Offline RL Of particular interest is the problem of Offline RL. This refers to the condition where an optimal policy must be derived from a fixed dataset of interactions, with no further environment access. This inability to actively interact with the environment can severely limit the learning capabilities of the agent. This setting sidesteps the exploration problem of online RL, the dataset is assumed to contain all information required for learning, but it introduces its own central difficulty: *distribution shift*. The data are generated by some (unknown) behavior policy, whereas the learned policy induces a different visitation distribution. As a result, estimates of $J(\pi)$ (in Equation (3.1.1)) built from the dataset are unbiased for the behavior policy but not necessarily for the current policy, leading to biased optimization and potential extrapolation error. See [8] for an in-depth discussion. This mismatch has several consequences already hinted to in Chapter 1; for instance, it can exacerbate error compounding in sequential decision-making (Section 2.4).

3.2 Contrastive Learning

During interactions, data are generally collected sequentially as a stream. Here, we refer to temporal consistency as knowing which observation results from an interaction carried out in the state represented by the current observation. As such, we are assuming access to a dataset in the form of Equation (1.1.2), i.e., a collection of triplets (o, a, o'). In this scenario, we no longer have access to the absolute supervision of the group action and have to rely on the relative supervision of temporal adjacency. Two subsequent observations should share more information than a third observation sampled at random in the dataset. This is 'only' a comparative relationship. Here, the equivariant formulation in Equation (2.2.2) does not apply anymore, as we do not explicitly know the relation between these two successive states.

This comparative relationship between data points is in line with the contrastive learning formulation. In its classical form, this assumes access to a dataset where data is clustered according to some similarity measure. The aim is to estimate a parametric representation φ_θ where similar data points lie closer than dissimilar ones. Similarity can be explicitly defined in the dataset (e.g., semantic classes, Euclidean distances [63]) or generated synthetically through data augmentation techniques (e.g., image transformations [64]). In particular, given an observation o, we can denote with $\mathcal{D}_+$ the dataset of observations similar to o (also referred to as positives) and with $\mathcal{D}_-$ the dataset of dissimilar ones (also referred to as negatives). As such, the objective of contrastive learning can be formulated as minimizing some notion of distance d between the latent embedding of o and any $o_+ \in \mathcal{D}_+$ while maximizing the distance with the negatives $o_- \in \mathcal{D}_-$:

$$\min_\theta \sum_{o \in \mathcal{D}} \left(\sum_{o_+ \in \mathcal{D}_+(o)} d(\varphi_\theta(o), \varphi_\theta(o_+)) - \sum_{o_- \in \mathcal{D}_-(o)} d(\varphi_\theta(o), \varphi_\theta(o_-)) \right). \tag{3.2.4}$$

This optimization mirrors the equivariant formulation of Equation (2.2.2) with two key differences. First, as we do not know exactly the relation between positive pairs, their latent distance is minimized (differently from Equation (2.2.2) where their distance is governed by the actual action taken). Second, an additional term maximizing the distance between negative pairs is needed (Equation (2.2.2) does not require this negative term). In this context, the negative term is necessary to avoid trivial solutions[2]. In fact, by optimizing the positive term only, the objective could be minimized by any map collapsing to a constant representation. In practice, most of the works rely on bounded spaces to avoid the negative term from growing uncontrollably, e.g., unit-norm embeddings on the hypersphere with cosine similarity as distance [66]. In our temporally supervised case, we treat (o, o') from the same transition as positives and use other batch elements as negatives.

3.3 Distance Monotonicity

Temporal consistency provides a measure of similarity. Here, we consider a way of exploiting this bias to guide the learning of representations. Consider a general POMDP described by the states in $\mathcal{S}$. When performing an action a in s, the system evolves to a new state s'. Because we do not have prior knowledge of the system symmetries, we do not know the algebraic transformation from s to s'. On the other hand, we do know that s' is one action away from s. Given all the states of the system, we can define a measure of distance between any two states as the minimum number of actions required to go from one state to the other, i.e., the shortest-path (geodesic) distance. Note that this distance does not necessarily reflect the Euclidean distance in the state space, e.g., when the agent needs to come around an obstacle, Figure 3.3.1. We can generalize this with the metric space $(\mathcal{S}, d_{\mathcal{S}})$, where distances here are defined as the minimum number of actions needed to move from one state to another. As in Chapter 2, we consider the problem of learning isometric representations. Let $\varphi : \mathcal{S} \to \mathcal{Z} \subseteq \mathbb{R}^n$ be a representation, and endow $\mathcal{Z}$ with a metric $d_{\mathcal{Z}}$. Notice that, differently from before, we have now generalized the notion of distance in the two spaces such that they do not need to be equivalent. The intuition is that, while $d_{\mathcal{S}}$ might be quite a complex measure, we aim at learning a representation where distances can be computed more easily, e.g., we can define $d_{\mathcal{Z}}$ as a Euclidean norm.

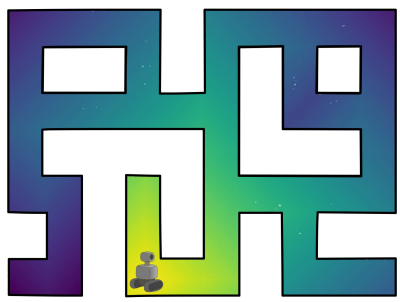


Figure 3.3.1: Distances in a maze do not necessarily correspond to Euclidean ones. In yellow states that are closer to the robot, in blue the more distant ones.

Depending on the definition of the latent metric space, isometric representations do not always exist [67]. This is particularly true when the defined latent metric space

[2]There have been works studying contrastive objectives that do not require the negative term explicitly [65].

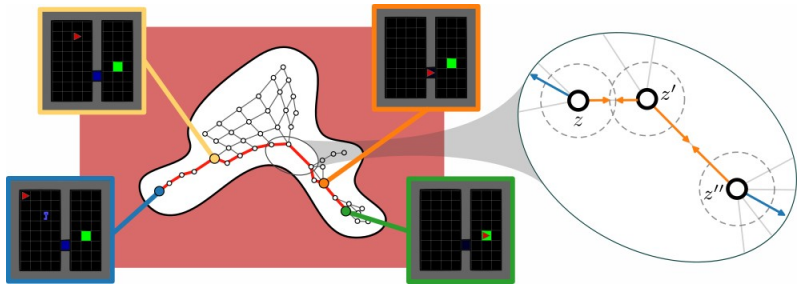


Figure 3.3.2: The latent embedding of a representation learned via Equation (3.3.5) on a navigation environment. The learned latent space gets deformed to increase the ratio of DM triplets. This is achieved by bringing adjacent states at a constant distance (orange arrows on the left) while maximizing the distance between non-adjacent ones (blue arrows on the left).

is relatively simple, e.g., Euclidean space. However, casting the original problem into a structured Euclidean representation can greatly simplify many downstream tasks. Here, we consider a relaxation of isometries as a form of structure for learned representations and study some properties that can be recovered. Given two metric spaces $(\mathcal{S}, d_{\mathcal{S}})$, $(\mathcal{Z}, d_{\mathcal{Z}})$ and a map φ between them, we propose the following property on the representation:

Definition 1. *We say φ is* Distance Monotonic (DM) *if for all $s_1, s_2, s_3 \in \mathcal{S}$, the following holds:*

$$d_{\mathcal{S}}(s_1, s_3) < d_{\mathcal{S}}(s_2, s_3) \implies d_{\mathcal{Z}}(\varphi(s_1), \varphi(s_3)) < d_{\mathcal{Z}}(\varphi(s_2), \varphi(s_3)).$$

A DM map does not preserve distances between states. Instead, it preserves orders between them. That is, if a state is closer to another with respect to a third one, then their embeddings will also be closer than the embedding of the third one. This is a relaxation of isometries as the distances on the learned space can now be deformed as long as the orders are preserved. The property of DM has a similar connotation to the objective of contrastive learning. The relation between embeddings is relative. In this case, similarity is given by adjacency in terms of actions. That is, states connected by an action should lie closer to each other than a third state at random. This, however, should be true for any triplet of states, Figure 3.3.2. Different from classic contrastive learning, we are not aiming for invariance between similar data points. Two subsequent states should not be considered equivalent but rather close in terms of $d_{\mathcal{Z}}$. That is, their latent embeddings should be at a small constant distance. On the other hand, non-adjacent states should be at a greater distance. We can formulate the objective of

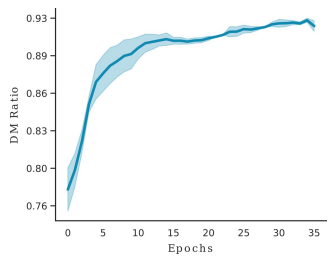


Figure 3.3.3: Optimizing Equation (3.3.5) increases the number of DM triplets in a maze navigation environment.

the representation as follows:

$$\mathcal{L} = \mathbb{E}_{\substack{(s,s')\sim\mathcal{S},\\ s''\sim\mathcal{S}}} \left[\left(\|\varphi(s') - \varphi(s)\|_2 - 1 \right)^2 - \|\varphi(s'') - \varphi(s)\|_2 \right], \tag{3.3.5}$$

where s and s' are states connected by a single action while s'' is another state sampled at random. Maximizing the distance between every two states (second term of Equation (3.3.5)) allows us to recover the DM property. Figure 3.3.3 shows how the number of triplets complying with Definition 1 increases as Equation (3.3.5) gets optimized for a maze navigation environment. Intuitively, this term encourages the embedding of every state to spread apart as much as possible. However, the first term of the loss chains together every subsequent state. Because of the triangle inequality, the second term of the loss is bounded automatically, Figure 3.4.5.

3.3.1 Extension of the Robotic Example

We now revisit the robotic setting of Section 1.0.1 without symmetry assumptions. Suppose the end-effector must route around an obstacle, Figure 3.3.4. Then $d_{\mathcal{S}}$ between two configurations reflects the minimal number of feasible motion primitives—not their Euclidean separation in configuration or image space. The DM objective (3.3.5) leverages temporal consistency to shape a latent space where (i) successive, feasible moves sit at roughly one "unit" of distance and (ii) configurations requiring more moves lie correspondingly farther away. As a result, the learned geometry respects task-relevant accessibility (geodesics around the obstacle) rather than raw pixel or pose differences, yielding a structured embedding that remains useful for downstream planning and control.

Figure 3.3.4: Example of an obstacle avoidance manipulation problem.

3.4 A Value Function Approximation

The property of order preservation can have interesting applications. In **Paper C**, we study these representations in the context of offline RL. Consider a POMDP as described in Section 3.1. In particular, consider the case of sparse-reward goal-conditioned RL problems. That is, POMDPs where the reward function can be defined as:

$$r = \begin{cases} r_g & \text{if } f(s,a) = s_g, \\ 0 & \text{else,} \end{cases} \tag{3.4.6}$$

where s_g denotes the goal state and r_g a positive scalar (often set to 1). Due to the sparsity of the reward signal, these problems can be challenging to address with classic

Figure 3.4.5: The latent embedding of the states of a maze stretches open as Objective (3.3.5) gets optimized. Color indicates the estimated distance with respect to one point in the maze.

RL objectives. Estimating the value function with dynamic programming approaches can be computationally inefficient. In fact, in the Bellman update of Equation (3.1.2), the reward being almost always zero means the value function rarely updates. On the other hand, these problems offer an interesting perspective. Goal-conditioned RL is equivalent to the problem of geodesic estimation. A policy is optimal, in fact, when given a starting state, it generates a trajectory that reaches the goal in the minimum number of steps (i.e., the discount factor is applied the minimum number of times). Assuming deterministic transitions, if we denote $d_{\mathcal{S}}$ as the geodesics in the state space, the optimal value function can be rewritten as:

$$V^*(s) = \gamma^{d_{\mathcal{S}}(s,s_g)} r_g. \tag{3.4.7}$$

Estimating $d_{\mathcal{S}}(s, s_g)$ can be complex in general. In **Paper C**, we explore DM representations as a way to approximate Equation (3.4.7). Learning a representation φ, by optimizing Equation (3.3.5), allows to approximate the optimal value function[3] as:

$$\tilde{V}(s) = \gamma^{d_{\mathcal{Z}}(\varphi(s),\varphi(s_g))} r_g. \tag{3.4.8}$$

Equations (3.4.7) and (3.4.8) coincide only in the case of φ being isometric. Under the weaker DM property, they don't necessarily match in value. However, this approximation still allows us to recover the optimal policy. In **Paper C**, we show that a greedy policy built on this approximation is optimal.

Theorem 2. *If the POMDP is deterministic, sparse, and goal-conditioned, then*

$$\pi_g^{\tilde{V}}(s) \subseteq \pi_g^{V^*}(s), \quad \forall s \in \mathcal{S}$$

holds if φ is distance monotonic.

This means that this approximation of the value function can still be used to estimate an optimal policy. Moreover, this approximation has two very attractive properties with respect to the original Bellman formulation. First, Equation (3.3.5) is generally more efficient than dynamic programming in the context of highly sparse reward signals. Second, avoiding the explicit use of the max operator in the Bellman equation

[3]Note, this formulation allows to consider any state as goal which is more akin to the Universal Value Function Approximators (UVFA) formulation [68].

mitigates the overestimation of the value function in offline RL conditions. In the case of a continuous action space, an optimal policy can be estimated via classic actor-critic techniques by replacing the critic with Equation (3.4.8).

3.5 A Geometric Approximation of Uncertainty

A second property that can be extracted from DM representations is presented in **Paper D**. When the state of the POMDP is not directly observable, we need to rely on sensory observations to guide the decisions of the agent. In the presence of noise, extracting this information through the representations described so far might not be enough. A more robust approach is to use a Bayesian filter technique [69]. That is, to maintain a belief of the possible states in the form of a probability distribution and to recursively update it based on a prediction over the dynamics of the system and an update based on the evidence provided by the (possibly) noisy observations. Assuming access to N observations for each time step and making use of the Markov assumption, this can be written as:

$$p(s' \mid s, a, o'_{1:N}) = \frac{p(o'_{1:N} \mid s')p(s' \mid s, a)}{p(o'_{1:N})}. \tag{3.5.9}$$

In general, this formulation is intractable and requires assumptions on the functional form of these probabilities. Particularly when the nature of the noise in the observations is unknown.

Having access to a perfect transition model, f, would be sufficient to correctly predict the state of the system at any point in time. As already pointed out in Chapter 2, when approximating it parametrically, this will inevitably lead to error compounding. This error on the approximated transition, however, is generally local to its prediction. That is, the uncertainty induced by the approximation error is high around states close in action terms to the prediction. Let us consider an ideal DM representation and the latent metric space induced by it: $\varphi : \mathcal{S} \to \mathcal{Z}$ where $\mathcal{Z} \subseteq \mathbb{R}^m$ and assume $d_{\mathcal{Z}} = \|\cdot\|_2$. We can approximate the dynamics of the system via a parametric function in this latent space: $\varphi_T : \mathcal{Z} \times \mathcal{A} \to \mathcal{Z}$. We make the following assumption:

Assumption 1. *The transition error at time t is constrained such that the predicted latent state $\varphi_T(z, a)$ lies within a small ball of states that are close in terms of action-based distance. That is, there exists an ϵ-radius region in the latent space such that:*

$$z' \in \mathcal{B}(\varphi_T(z, a), \epsilon), \tag{3.5.10}$$

where $\mathcal{B}(z, \epsilon) = \{z' \in \mathcal{Z} \mid d_{\mathcal{Z}}(z, z') \leq \epsilon\}$.

Assumption 1 allows us to consider the transition error to be *local* to the prediction in the space induced by the dynamics of the environment. The compounding of this error, however, requires periodic adjustments to keep this uncertainty bounded. Updating the estimate with sensory observations at each step is therefore crucial to maintaining a reliable belief about the state of the environment. In **Paper D**, we propose to learn a DM representation, φ_i, for each of the N modalities together with a latent transition

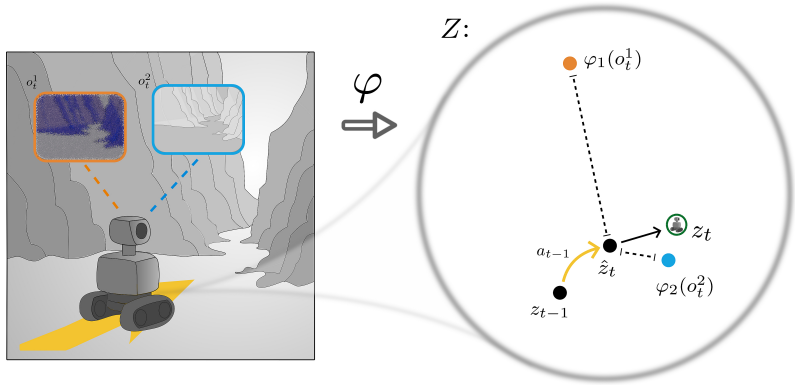


Figure 3.5.6: In a DM space, the uncertainty induced by noise on sensors can be approximated via Euclidean distances. This allows for state estimation through a geometric Bayesian filter technique.

model, such that these representations are aligned in a shared latent space. This can be done by defining the following objectives:

$$\mathcal{L}_+ = \mathbb{E}_{\mathcal{D}}\left[\left(\|\bar{z}' - \bar{z}\|_2 - 1\right)^2\right], \tag{3.5.11}$$

$$\mathcal{L}_- = \mathbb{E}_{\mathcal{D}}\left[-\log(\|\bar{z}'' - \bar{z}\|_2)\right], \tag{3.5.12}$$

$$\mathcal{L}_T = \mathbb{E}_{\mathcal{D}}\left[\|\varphi_T(\bar{z}, a) - \bar{z}'\|_2\right], \tag{3.5.13}$$

$$\mathcal{L}_{inv} = \mathbb{E}_{\mathcal{D}}\left[\|\varphi_i(o_i) - \varphi_j(o_j)\|_2\right] \quad \forall i, j \in [1, N], \tag{3.5.14}$$

with total loss $\mathcal{L} = \mathcal{L}_T + \mathcal{L}_+ + \mathcal{L}_- + \mathcal{L}_{inv}$. Here, $\bar{z}$ refers to the mean embedding across modalities. Specifically, Equations (3.5.11) and (3.5.12) force the learned space to be DM with the true state space, Equation (3.5.13) recovers the latent transition approximating the dynamics of the environment, while Equation (3.5.14) aligns the representation of every modality in the same space.

When the sensory information is affected by noise, the embedding generated by the learned representations can be unreliable. DM representations allow us to recast the problem of uncertainty estimation over noisy observations to a geometric problem. Based on Assumption 1 and the definition of DM space, we can make the following considerations:

- If an observation's latent encoding $\varphi_i(o_i')$ is *close* to the latent transition prediction $\varphi_T(z, a)$, then it is more likely to be an accurate state estimate.
- Conversely, if an observation encoding is *far* from the prediction, it has likely been corrupted by noise or uninformative.

From this perspective, distances provide a measure of uncertainty, Figure 3.5.6. As such, we propose to model the update step as an average over the observations' embed-

dings weighted by their distance to the prediction of the transition model:

$$z' = \left(\sum_i \frac{1}{\|z'_i - \hat{z}'\|_2} \right)^{-1} \sum_i \frac{z'_i}{\|z'_i - \hat{z}'\|_2}, \tag{3.5.15}$$

where $\hat{z}' = \varphi_T(z, a))$ and $z'_i = \varphi_i(o'_i)$. This formulation is particularly convenient when the nature of uncertainty is unknown a priori and thus impossible to model. In the paper, we show how this representation can be used as a feature extractor for an RL agent trained end-to-end. We additionally provide empirical evidence of how this form of filtering results in robust performances in the presence of multiple and unknown sensory disturbances.

3.6 Discussion

Temporal consistency offers a weak yet highly scalable form of self-supervision. Compared to full geometric consistency, it imposes fewer invariances and requires no explicit knowledge of group actions, making it attractive for large and complex problems. In this chapter, we exploited this prior through DM representations, which encode progression over time so that "being closer in the latent space" correlates with "being closer to achieving the task". Under this structure, shortest-path objectives can be recast into simple Euclidean norms in the learned latent space, enabling lightweight planning and control.

We explored two consequences of this idea. In **Paper C**, DM representations were used to approximate the optimal value function in goal-conditioned RL: a greedy policy that descends latent distance toward the goal serves as a practical surrogate for dynamic programming. In **Paper D**, we showed that DM structure is also useful for multimodal state estimation under noise, where temporal alignment acts as a glue to fuse modalities and attenuate sensor-specific corruption without requiring an explicit generative model.

The supervision signal here is comparative rather than absolute: it relies on reliable timestamps and action alignment. DM assumes that local temporal progress is informative about global task progress. Heavy stochasticity and complex reward functions can reduce the efficacy of these representations. Moreover, using Euclidean norms in the latent space implicitly favors locally flat geometry; tasks with pronounced non-Euclidean structure may require learned (quasi-)metrics or geodesic approximations. Overall, these contributions illustrate an example of weaker structures in the representation that can still yield useful properties for downstream tasks.

4 Synthetic Interactions

In the previous chapters, we explored interactions as actions of an agent within a physical system. Here, we generalize this concept. We define interactions as synthetic operations on the parameter space of a learner. In particular, we explore structured representations (and their properties) of the parameters of a neural network trained on downstream objectives. We focus on two specific tasks: meta-learning and uncertainty estimation. The chapter presents a brief background on Bayesian neural networks for uncertainty estimation and gradient-based meta-learning. Moreover, it describes two contributions: *Walking on the Fiber: A Simple Geometric Approximation for Bayesian Neural Networks* (**Paper E**), which uses distance monotonic representations to model the posterior probabilistic distribution of a Bayesian neural network; *Reducing Variance in Meta-Learning via Laplace Approximation for Regression Tasks* (**Paper F**), which extends the gradient-based meta-learning adaptation method to reduce the variance of the estimator.

4.1 Interacting in Parameter Space

In earlier chapters, interactions were *physical*: an agent applied actions $a \in \mathcal{A}$ to a system with state $s \in \mathcal{S}$ and observed transitions $s' = f(s, a)$. Here, we generalize the notion of interaction to the *parameter space* of a learner. We assume a supervised task specified by an unknown map $f : \mathcal{X} \to \mathcal{Y}$ that we approximate with a parametric learner $f_\theta : \mathcal{X} \to \mathcal{Y}$ by minimizing a loss $\mathcal{L}(\theta; \mathcal{D})$ over a labeled dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$. When $\mathcal{Y}$ is continuous, we refer to the task as *regression*; when $\mathcal{Y}$ is a finite, discrete set (e.g., $\mathcal{Y} = \{1, \dots, K\}$), we refer to it as *classification*. In particular, we take f_θ to be a neural network with a high-dimensional parameter vector $\theta \in \Theta \subseteq \mathbb{R}^d$. Here, we consider the case of over-parameterized networks, that is, $d \gg n$. Modern over-parameterized networks often contain broad, connected regions in parameter space where the loss stays nearly the same: different parameter settings that behave almost equivalently [70, 71]. Locally, their shape is influenced by curvature (how steep or flat the landscape feels nearby), while globally their connectivity reflects symmetries and redundancies

in the model (such as neuron permutations or scale reparameterizations). Exploiting this geometry can improve the learning process.

Synthetic interactions. We define a *synthetic interaction* as any structured operation that moves parameters to a new point in parameter space,

$$\theta' = T(\theta, u), \tag{4.1.1}$$

where u encodes the rule of interaction (e.g., a step along a direction, a symmetry transformation, a learned update, or a projection), and $T : \Theta \times \mathcal{U} \to \Theta$ is the interaction operator. Unlike physical interactions, synthetic interactions manipulate the *model itself*. Thus, the learning algorithm induces *dynamics on* Θ (e.g., by $\mathcal{L}$). The vector of parameters θ implicitly *encodes the task* via the hypothesis f_θ fitted to data $\mathcal{D}$. From this perspective, an interaction in Θ is an interaction with the induced hypothesis itself. Synthetic interactions allow us to *move along* these low-loss regions in the space of parameters, sliding in directions that keep performance stable, without explicitly interacting with the real world.

In this chapter, rather than representing sensory data in a latent space, we aim at learning representations of the parameters themselves such that the *geometry of* Θ is easier to traverse. In particular, we explore two structures:

- **Explicit deformation (Paper E).** Directly learning a distance monotonic representation $z = \varphi(\theta)$ in which regions of low loss are embedded as approximately straight subsets. This allows for easy sampling of low-loss parameters.
- **Implicit straightening (Paper F).** Learning a set of parameters θ_0 such that the region surrounding it is locally straight and flat with respect to the loss. This allows a few gradient steps on a specialized dataset to quickly adapt the learn model to specific tasks. Here, the representation is implicit in the coordinates of θ_0.

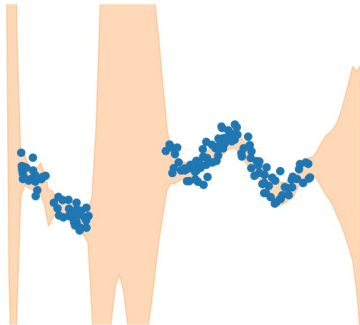
Under both views, interactions are operators that can be used to explore the space of parameters by *moving along low-loss regions*.

4.2 Uncertainty on the Parameters

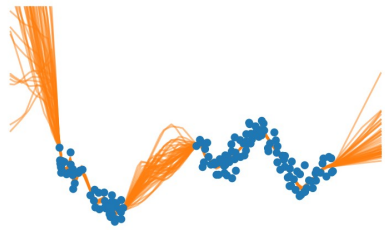
4.2.1 Bayesian Neural Networks

Approximating an unknown map with a parametric model inevitably induces prediction errors. Explicitly modeling predictive *uncertainty* helps quantify the associated risk and informs downstream decisions. Consider, for example, a simple one-dimensional regression problem as in Figure 4.2.1a where the task is to regress a best-fitting curve for the blue points. Regions of low data density should induce a higher uncertainty on the approximation (in orange). Bayesian Neural Networks (BNNs) provide a framework to quantify this uncertainty in parametric models [72]. They achieve this by treating the network weights θ as random variables and inferring a posterior distribution $p(\theta \mid \mathcal{D})$ from the learning data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$. More precisely, given a set of new (unseen) points (x', y'), we are interested in estimating the following:

$$p(y' \mid x', \mathcal{D}) = \int p(y' \mid x', \theta)\, p(\theta \mid \mathcal{D})\, d\theta, \tag{4.2.2}$$



(a) Simple 1D regression task. Regions of low density should increase uncertainty (in orange).



(b) Possible low-loss parametric approximations for a set of points in a regression task.

which marginalizes uncertainty in θ rather than committing to the best performing single point estimate. The integral in (4.2.2) is, in general, intractable. Using the Bayes rule [73], we can rewrite the posterior of the parameters as $p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}$. Here, $p(\mathcal{D} \mid \theta)$ is the likelihood of the data, $p(\theta)$ is a prior on the parameters, and $p(\mathcal{D})$ is the intractable marginal likelihood. This formulation is particularly challenging when the function approximation used is a high-dimensional, nonlinear deep neural network. Exact posteriors are intractable for modern networks, BNNs relies on approximate inference:

- **Variational Inference (VI).** Choose a tractable family $q_\phi(\theta)$ and fit it by minimizing $\mathrm{KL}(q_\phi(\theta) \,\|\, p(\theta \mid \mathcal{D}))$. Monte Carlo samples $\theta^{(s)} \sim q_\phi$ provide plug-in estimates of (4.2.2), [74].
- **Laplace Approximation.** Around a trained mode θ, approximate the negative log posterior with a quadratic; equivalently, $q(\theta) = \mathcal{N}(\theta, H^{-1})$, with H the Hessian (or Fisher/Gauss–Newton) at θ, [75].
- **Markov Chain Monte Carlo (MCMC)/sampling.** Simulate posterior draws via stochastic gradient Langevin dynamics [76] or Hamiltonian Monte Carlo [77]. Sampling is flexible and better reflects multi-modal posteriors, but wall-clock and memory costs can be large in deep nets.
- **Ensembles.** Train an ensemble of independent models and approximate the posterior via their variance, [78].

Of these, both the Laplace approximation and sampling methods offer convenient solutions. In fact, they can be applied post-hoc to an already trained model to recover a measure of uncertainty. When considering over-parameterized neural networks, however, these approximations have several limitations, [79]. As previously stated, these networks generally admit broad and connected regions of (near) equal loss. The posterior mass concentrates along curved sets, and local quadratic fits around one mode tend to underestimate the true posterior. Moreover, due to the high-dimensionality of the parameter space, scalability becomes a sensible issue. These two observations motivate the first method in this chapter.

4.2.2 A Geometric Approximation for BNNs

Estimating a good posterior is equivalent to identifying the distribution of these low-loss parameters, Figure 4.2.1b. **Paper E** explores synthetic interactions in the parameter space as a possible solution. Interactions, in fact, can provide a simple and scalable sampling procedure that perturbs and refines parameters to explore the low-loss set near the solution of an already trained network. Learning a DM latent deformation of the parameter space allows us to straighten those solution trajectories (interactions), enabling fast posterior sampling without iterative methods.

Given an already trained model, with parameters θ^*, we can explore the landscape of the loss around these parameters by means of synthetic interactions. We instantiate N particles at θ^* and repeatedly apply T times a two-step interaction that alternates a small *drift* in a random direction (Equation (4.2.3)) with M *refinements* by gradient descent on $\mathcal{L}$ (Equation (4.2.4)):

$$\hat{\theta}^0_{i,t+1} = \theta_{i,t} + \alpha\, d_i, \qquad \|d_i\| = 1, \tag{4.2.3}$$

$$\hat{\theta}^{m+1}_{i,t+1} = \hat{\theta}^m_{i,t+1} - \eta\, \nabla_\theta \mathcal{L}(\hat{\theta}^m_{i,t+1}), \qquad m = [1, M]\,, \tag{4.2.4}$$

$$\theta_{i,t+1} = \hat{\theta}^M_{i,t+1}, \qquad i = [1, N]\,,\ t = [1, T]\,. \tag{4.2.5}$$

This induced dynamics on Θ results in a number of trajectories where each element, $\theta_{i,t+1}$, is a set of parameters with a low loss, Figure 4.2.2. In fact, applying the refinement step (Equation (4.2.4)) for enough iterations (M) ensures a good solution. In general, this sampling strategy does not scale well for a sufficiently high-dimensional space of parameters. The number of particles needed to accurately characterize the low-loss region around θ^* would have to increase accordingly. Over-parameterized modern neural networks offer an exception to this. Empirically, the dimensionality of this low-loss region has been observed to be significantly lower-dimensional and connected for these models [70, 80]. This allows, in practice, for reasonable computational complexity.

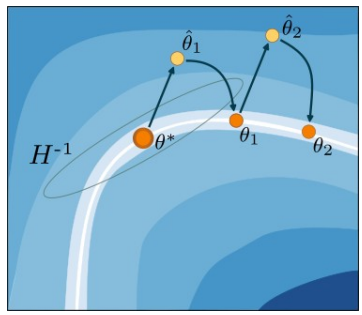


Figure 4.2.2: Proposed sampling scheme for a trajectory of parameters along a region of low loss (in white) in the parameter space.

The set of solutions found via this sampling procedure already represents an approximation to the posterior inference problem. The predictive mean and uncertainty can be estimated by Monte Carlo: $\hat{y}' = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}\left[y' \mid x', \theta_i\right]$. However, this algorithm does not allow for sampling the posterior over the parameters, $p(\theta \mid \mathcal{D})$, directly. The path of these trajectories, in fact, can be arbitrarily non-linear. In this regard, we can train a representation of the parameters such that sampling becomes easier. Define an autoencoder as: $\varphi_{\psi_1} : \Theta \to \mathcal{Z}, \varphi^{-1}_{\psi_2} : \mathcal{Z} \to \Theta$. This can be trained on the set of parameters collected during the sampling phase (i.e., $D_\theta = \{\theta_{i,t}\}$ for every time step and trajectory) to minimize the following objective for both ψ_1 and ψ_2:

$$\mathcal{L} = \lambda_+ \mathcal{L}_+ + \lambda_- \mathcal{L}_- + \mathcal{L}_d, \tag{4.2.6}$$

with:

$$\mathcal{L}_+ = \mathbb{E}_{\mathcal{D}_\theta} \left[\left(\|\varphi_{\psi_1}(\theta_t) - \varphi_{\psi_1}(\theta_{t+1})\|_2 - \frac{1}{T} \right)^2 \right], \tag{4.2.7}$$

$$\mathcal{L}_- = \mathbb{E}_{\mathcal{D}_\theta} \left[-\log \left(\|\varphi_{\psi_1}(\theta_t) - \varphi_{\psi_1}(\theta_{t+2})\|_2 \right) \right], \tag{4.2.8}$$

$$\mathcal{L}_d = \mathbb{E}_{\mathcal{D}_\theta} \left[\|\varphi_{\psi_2}^{-1}(\varphi_{\psi_1}(\theta_t)) - \theta_t\|_2 \right], \tag{4.2.9}$$

This is similar to the optimization used in Chapter 3 to recover DM representations. The representation of two subsequent (in terms of synthetic interactions) sets of parameters should lie at a constant distance from each other (Equation (4.2.7)) while every other embedding should spread apart as much as possible (Equation (4.2.8)). Additionally, we require a third term in the loss for learning the decoder. This guides the overall autoencoder to be as close to the identity as possible (Equation (4.2.9)). Interestingly, in order to maximize distances, the representation unrolls the embedding of each trajectory into a straight line radially exiting from the representation of θ^*. This deformation of the space of parameters allows for an easier sampling technique. We can, in fact, uniformly sample one of the collected trajectories and then linearly interpolate between the representation of θ^* and the representation of the last sample of that trajectory, i.e., $\theta_{i,T}$. Decoding this interpolated value results in a valid set of parameters:

$$\theta = \varphi_{\psi_2}^{-1} \left(\varphi_{\psi_1}(\theta^*) + \epsilon \cdot (\varphi_{\psi_1}(\theta_{i,T}) - \varphi_{\psi_1}(\theta^*)) \right), \quad \epsilon \sim \text{Uniform}(0, 1). \tag{4.2.10}$$

This allows for sampling new weights from the posterior distribution without the expensive sampling procedure described by Equations (4.2.3), (4.2.4).

4.3 Fast Adaptation

Thus far, we have focused on learning a parametric approximation for a single task implicitly represented by a dataset, i.e., $\mathcal{D} = \{(x_i, y_i = f(x_i))\}_{i=1}^n$. This assumes the task is fixed in time. Meta-learning [81, 82] addresses this limitation by endowing the learning system with the ability to adapt quickly to changing tasks. This is typically achieved in two stages: first, estimating a base learner; second, refining it on a small set of new points that characterize the changed task.

Consider again the dynamical system described in Chapter 1, Equation (1.1.1). We can generalize it by introducing parameters that encode constants of the system in either the forward model or the emission function, i.e., $s' = f(s, a, \beta)$ and $o = \omega(s, \beta)$. Examples include friction coefficients, object dimensions and shapes, camera parameters, and so on. Even small changes in these parameters β can severely affect the performance of a learned agent [83]. Meta-learning enables the transfer of acquired knowledge to a new environment without retraining from scratch[1].

[1]Closely related is the problem of system identification [84], where the aim is to estimate β directly, often assuming the functional form of the system.

We formalize meta-learning as learning to adapt across a distribution of tasks. Let $\mathcal{T}$ denote the space of tasks and $p(\mathcal{T})$ a task distribution that captures shared structure (e.g., common dynamics with task-specific parameters). A task $\tau \sim p(\mathcal{T})$ corresponds to an unknown map $f_\tau : \mathcal{X} \to \mathcal{Y}$ described by a dataset $\mathcal{D}$ (i.e., one particular choice of parameters β). In practice, we absorb β into the function and write f_τ to emphasize the correspondence to task τ. Each task dataset is split episodically into a *support* set $\mathcal{D}_\tau^S$ (for adaptation) and a *query* set $\mathcal{D}_\tau^Q$ (for meta-training/evaluation), with $|\mathcal{D}_\tau^Q| \gg |\mathcal{D}_\tau^S|$ to simulate fast adaptation. We focus on supervised regression, i.e., continuous f_τ. Our goal is to learn an adaptation rule that maps the support data of each task to an approximation of the task's true function, i.e., f_τ.

4.3.1 Gradient-Based Meta-Learning

This adaptation algorithm has been formulated in different ways. Gradient-Based Meta-Learning (GBML) implements it as a gradient-descent step on the learner's parameters computed from the support set [85]. The objective of GBML is to estimate a base parameter vector θ_0 such that a single gradient step suffices to approximate any task drawn from $p(\mathcal{T})$. Using Mean-Squared Error (MSE) for regression, we obtain the following bi-level optimization:

$$\min_{\theta_0} \quad \mathbb{E}_{(\mathcal{D}_\tau^S, \mathcal{D}_\tau^Q)\sim p(\mathcal{T})} \left[\sum_{(x_\tau^i, y_\tau^i)\in \mathcal{D}_\tau^Q} \| f_{\theta_\tau}(x_\tau^i) - y_\tau^i \|_2^2 \right] \tag{4.3.11}$$

$$\text{s.t. } \theta_\tau = \theta_0 - \alpha \nabla_\theta \left[\sum_{(x_\tau^j, y_\tau^j)\in \mathcal{D}_\tau^S} \| f_{\theta_0}(x_\tau^j) - y_\tau^j \|_2^2 \right], \tag{4.3.12}$$

where α is the learning rate. The only learned parameter is θ_0. Effectively, GBML aims at finding a set of parameters whose surrounding region in parameter space is well conditioned for a single gradient step independent of the complexity of the parametric function, e.g., a high-dimensional deep neural network. That is, it *implicitly learns a structured representation of parameters* by shaping a neighborhood around θ_0 where gradient descent (i.e., a synthetic interaction) is effective [86]. From a Bayesian perspective: θ_0 acts as a learned prior; the inner update produces a (Maximum a Posteriori (MAP)-like) estimator of $p(\theta \mid \theta_0, \mathcal{D}_\tau^S)$; the outer optimization maximizes $p(\mathcal{D}_\tau^Q \mid \theta_\tau)$ across tasks [87]. In other words, GBML optimizes θ_0 so that the induced geometry of Θ near θ_0 renders one-step descent a reliable transport from the prior to the task-adapted parameters. Note that, in this case, the learned representation of the parameters is implicit. Unlike before, the representation is the parameters themselves or, equivalently, φ is just the identity map and the structure is the geometry of the space surrounding θ_0.

4.3.2 Reducing Variance in the Adaptation Estimation

GBML offers an appealing solution to the problem of fast adaptation. One limitation can arise when data points in the support set can belong to multiple tasks simultaneously, Figure 4.3.3 on the left.

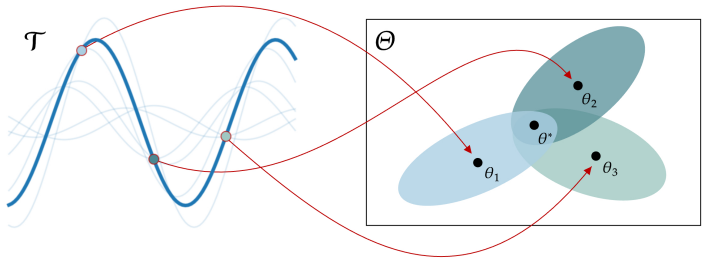


Figure 4.3.3: For a family of sine waves with varying phase and amplitude, a single (x, y) support tuple can correspond to multiple curves (left). In **Paper F**, we approximate this distribution with a Gaussian via the Laplace approximation (right).

Definition 2 (Task Overlap)**.** *We define task overlap as the condition for which* $\forall x \in \mathcal{X}$ *the map* $f(\cdot, x) : \mathcal{T} \to \mathcal{Y}$ *is non-injective.*

Under task overlap, in fact, the estimation of the adapted parameters in GBML can suffer from high variance. By linearity of the gradient operator, the estimator's variance is directly proportional to the variance of the per-point gradients in the support set:

$$\text{Var}\left[p(\theta_\tau \mid \mathcal{D}_\tau^S)\right] = \frac{1}{N^2} \sum_{i=1}^{N} \text{Var}\left[p(\theta_\tau \mid x_\tau^j, y_\tau^j)\right]. \tag{4.3.13}$$

With task overlap, distinct tasks can induce similar support data points (variance is non-zero). The per-task parameter estimate from a small support set, $|\mathcal{D}_\tau^S|$, thus has high variance as well. Uniform averaging of per-point updates (as in standard GBML) degrades query performance considerably in this setting. In **Paper F**, we propose a modification of the adaptation method to reduce the variance of the estimated parameters. We reduce this variance by treating each support example as defining a *local* posterior over task parameters and then aggregating these posteriors in a precision-weighted manner. Specifically, we propose to model the distribution of parameters that minimize the MSE on each support point with a Gaussian distribution. That is, each support point $(x_i, y_i) \in \mathcal{D}_\tau^S$ induces a local Gaussian posterior on the parameters, Figure 4.3.3 on the right. This distribution can be estimated via the Laplace approximation by using the original GBML formulation as the mean and the inverse of the Hessian on the MSE as the covari-

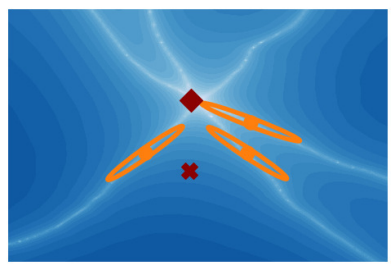


Figure 4.3.4: Space of parameters, from white to blue the loss on the support. In orange are the induced Laplace approximations per data point in the support, the red cross and red diamond indicate the prior and posterior parameters.

ance:

$$p(\theta_\tau \mid x_\tau^j, y_\tau^j) \approx \mathcal{N}\left(\theta_\tau^j, \left[H^j\right]^{-1}\right) \tag{4.3.14}$$

$$\text{with } \theta_\tau^j = \theta_0 - \alpha \nabla_\theta \mathcal{L}(\theta_0; x_j, y_j) \tag{4.3.15}$$

$$H^j = \nabla^2_{\theta_\tau^j} \mathcal{L}(\theta_\tau^j; x_i, y_i), \tag{4.3.16}$$

where we denote the MSE as $\mathcal{L}(\theta; x, y) = \|f_\theta(x) - y\|_2^2$. As such, the overall adaptation algorithm can be rewritten as the aggregate of the product of these Gaussians:

$$\theta_\tau = \left(\sum_{i=1}^{N} H^j\right)^{-1} \sum_{j=1}^{N} H^j \theta_\tau^j. \tag{4.3.17}$$

Equation (4.3.17) is the minimum-variance linear unbiased estimator under the Laplace model, strictly reducing estimator variance compared to the uniform GBML average whenever the H^j differ. Empirically, this mitigates the effect of task overlap.

Crucially, we propose to substitute Equation (4.3.12) with Equation (4.3.17) and optimize it together with Equation (4.3.11) in a bi-level fashion. This incentivizes the learner to find a region of parameters where the Laplace approximation coincides with the actual distribution of the parameters, Figure 4.3.4. We effectively impose an additional structure on the space of parameters that allows us to formulate a minimum variance estimator for fast adaptation in the task overlap regime.

4.4 Discussion

Interactions as a form of self-supervision can guide the learning of representations for a multitude of contexts. In this chapter, we have abstracted this from physical dynamical systems to general parametric learning. We refer to these operations in the space of parameters as *synthetic interactions*. In particular, the main operation studied was gradient descent on the loss function. This, in fact, allows for the exploration of equivalent solutions local to the parameters of a trained model. Over-parametrized deep neural networks, in particular, offer an interesting testing ground in this regard. Empirically, it has been observed that these low-loss regions are considerably lower-dimensional than the ambient space of the parameters.

Specifically, we have presented two properties that can be extracted by injecting structure via synthetic interactions. In **Paper E** we used a simple sampling technique to find loss-wise equivalent solutions to a downstream task. These, effectively, form trajectories in parameter space. Learning a DM representation of these allows for easy sampling from the posterior distribution of these parameters to get a measure of uncertainty of a learned model. In **Paper F**, we consider implicit representations in the form of the parameters them-self. Structure, in this context, can be imposed by finding a set of parameters whose surrounding region has some regularity. This is applied to the problem of fast adaptation within the meta-learning framework. Imposing a quadratic structure on the probability of the adapted parameters per adaptation point (via the Laplace approximation) effectively linearizes the space of solutions around the

prior parameters. This, allows for a low variance estimation of the adapted parameters in the GBML meta-learning context.

Nevertheless, working in the space of parameters can be computationally challenging. Modern networks have reached considerable dimensions, making synthetic interactions less effective. In practice, this can be mitigated with structured curvature approximations (e.g., Gauss–Newton/Fisher, block-diagonal/K-FAC) and with dimensionality reduction in parameter space (e.g., learning a low-dimensional parameter latent in which to operate). Subspace methods (low-rank adapters) and early stopping help keep trajectories within the low-loss region, while stochastic variants of synthetic interactions (e.g., Langevin-style refinements) can improve exploration without leaving the basin.

5 Conclusions and Future Work

In this thesis, we have studied how structure (algebraic and metric) can be injected into learned representations to improve perception, control, and adaptation in machine learning and robotics. The central idea was to formalize interaction as a source of supervision and then endow the latent space with symmetries or orderings that mirror those of the underlying system. In particular, *algebraic structure* (equivariance) lets us preserve the known symmetries of the system in the learned representation. This, in turn, can be used to estimate the precise relation between the states of the system. *Order structure* (temporal consistency) relaxes the need for this prior knowledge while still allowing for a relative relation between states. We have then explored different properties that can be extracted from these structures.

This includes:

Stabilizing visual imitation. We introduced an equivariant encoder that maps images to a Euclidean latent space where translations correspond to robot end–effector motions. In this space, we estimated a task–conditioned state-density with an MDN and defined a *recovery policy* that ascends the density gradient, blending it with a BC policy. The result is a closed-loop controller that returns to (and stays within) high-data-density regions, mitigating the compounding of error in deployment on a real robot.

Localizing external objects from interaction alone. We extended the equivariant formulation to scenes with an external rigid object of unknown dynamics. Under mild assumptions—contact-triggered motion, reachability, and injective observations, we proved that an ideal learner recovers an *isometric, disentangled* agent–object representation from actions and observations only.

Optimal value function approximation. We introduced *distance monotonicity* as a relaxation of isometry between metric spaces and an objective to estimate such representations. We have proved that approximating the value function in this learned

metric space yields an optimal policy under deterministic, sparse, goal-conditioned POMDPs.

Uncertainty estimation in the state space. We explored how DM representations can be used to estimate how noisy observations are in a multimodal state estimation setting. When distances are correlated with the system's dynamics, in fact, they can be used as a measure of uncertainty in the observations to build a Bayesian filter model.

Uncertainty estimation in the parameter space. We defined synthetic interactions as operations in the parameter space of a learner. We showed how these can be used to recover regions of low loss of a trained model to measure its uncertainty in a Bayesian neural network fashion. Moreover, DM representations can be used to approximate the posterior on the parameters to speed up the sampling of new low-loss parameters.

Variance-reduced fast adaptation in GBML. We analyzed *task overlap* in meta-regression and proposed a novel method to reduce the variance in the adaptation. Each support point is used to induce a Laplace-approximated local posterior in parameter space; these are then aggregated with Hessian-based weights, yielding a minimum-variance estimator for the adapted parameters.

Limitations and Future Directions

This work is subject to several limitations, which may open interesting extensions in the future. **Paper A** assumes translation equivariance. Extensions to other groups are possible. Generalizing the equivariant latent to $SE(3)$ with mixed translational/rotational actions is possible. This would require a modification of the objective described in (2.2.2) with a rotation component and of the gradient for the recovery policy (2.5.6) with an appropriate Riemannian formulation. **Paper B** as well could benefit from the same group extension just described. Moreover, the two main limitations of the work are the need for the quasi-static assumption on the object and being able to handle only one object at a time. The latter could be addressed by considering multiple datasets with interactions for each object or alternatively, pairing the representation with an agent trained to interact with each object in an interactive perception fashion.

On the metric side, there are a number of limitations as well. In **Paper C**, we assume the underlying POMDP to be possible to be described by a metric space. Often, in control settings, this might not be possible as there are irreversible actions (e.g., falling from a cliff). A more appropriate description would require a notion of quasi-metric, i.e., removing the symmetry axiom. **Paper D** could benefit from a quasi-metric representation as well. Additionally, the work explicitly considers the uncertainty of the observations only as the transition is used only for the comparative measure. A potential extension could estimate the volume of the transition error as well for a more informed estimation.

On the parametric side, **Paper E** explores a relatively simple form of interaction. A more complex sampling scheme could be used, involving adaptive sampling or something more similar to the Markov Chain Monte Carlo family of methods. Moreover, it would be interesting to apply this method to low-rank adaptations of considerably large

models. Furthermore, **Paper F** specifically focuses on the problem of fast adaptation on regression problems. Moving beyond the Laplace approximation could allow us to tackle non-continuous or topologically complex problems like classification. Turning the adaptation into a sequential Bayesian update would allow for continual learning solutions as well.

6 Summary of Included Papers

This chapter contains abstracts of the included papers and contributions by the author of the thesis. The symbol * denotes equal contribution.

Paper A

Back to the Manifold: Recovering from Out-of-Distribution States

A. Reichlin, G. L. Marchetti, H. Yin, A. Ghadirzadeh and D. Kragic
In *In International Conference on Intelligent Robots and Systems (IROS), 2022*

Abstract: Learning from previously collected datasets of expert data offers the promise of acquiring robotic policies without unsafe and costly online explorations. However, a major challenge is a distributional shift between the states in the training dataset and the ones visited by the learned policy at the test time. While prior works mainly studied the distribution shift caused by the policy during the offline training, the problem of recovering from out-of-distribution states at the deployment time is not very well studied yet. We alleviate the distributional shift at the deployment time by introducing a recovery policy that brings the agent back to the training manifold whenever it steps out of the in-distribution states, e.g., due to an external perturbation. The recovery policy relies on an approximation of the training data density and a learned equivariant mapping that maps visual observations into a latent space in which translations correspond to the robot actions. We demonstrate the effectiveness of the proposed method through several manipulation experiments on a real robotic platform. Our results show that the recovery policy enables the agent to complete tasks while the behavioral cloning alone fails because of the distributional shift problem.

Contributions by the author: co-designed the method and its implementation, designed and implemented the experiments and wrote parts of the paper.

Paper B

Learning Geometric Representations of Objects via Interaction

A. Reichlin*, G. L. Marchetti*, H. Yin, A. Varava and D. Kragic
In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML), 2023*

Abstract: We address the problem of learning representations from observations of a scene involving an agent and an external object the agent interacts with. To this end, we propose a representation learning framework extracting the location in physical space of both the agent and the object from unstructured observations of arbitrary nature. Our framework relies on the actions performed by the agent as the only source of supervision, while assuming that the object is displaced by the agent via unknown dynamics. We provide a theoretical foundation and formally prove that an ideal learner is guaranteed to infer an isometric representation, disentangling the agent from the object and correctly extracting their locations. We evaluate empirically our framework on a variety of scenarios, showing that it outperforms vision-based approaches such as a state-of-the-art keypoint extractor. We moreover demonstrate how the extracted representations enable the agent to solve downstream tasks via reinforcement learning in an efficient manner.

Contributions by the author: co-designed the method, implemented the model, designed and implemented the experiments and wrote parts of the paper.

Paper C

Goal-Conditioned Reinforcement Learning from Sub-Optimal Data on Metric Spaces

A. Reichlin, M. Vasco, H. Yin and D. Kragic
In *Preprint, 2025*

Abstract: We study the problem of learning optimal behavior from sub-optimal datasets for goal-conditioned offline reinforcement learning under sparse rewards, invertible actions and deterministic transitions. To mitigate the effects of *distribution shift*, we propose MetricRL, a method that combines metric learning for value function approximation with weighted imitation learning for policy estimation. MetricRL avoids conservative or behavior-cloning constraints, enabling effective learning even in severely sub-optimal regimes. We introduce distance monotonicity as a key property linking metric representations to optimality and design an objective that explicitly promotes it. Empirically, MetricRL consistently outperforms prior state-of-the-art goal-conditioned RL methods in recovering near-optimal behavior from sub-optimal offline data.

Contributions by the author: designed the method, implemented the model, designed and implemented the experiments and wrote most of the paper.

Paper D

Geometry of Uncertainty: Learning Metric Spaces for Multimodal State Estimation in RL

A. Reichlin, A. Pacciarelli, M. Vasco and D. Kragic
In *Preprint, 2025*

Abstract: Estimating the state of an environment from high-dimensional, noisy observations is a fundamental challenge in reinforcement learning (RL). Traditional approaches rely on probabilistic models to account for the uncertainty, but often require explicit noise assumptions, in turn limiting generalization. In this work, we propose a novel method to learn a structured latent representation, in which distances between states directly correlate with the minimum number of actions required to transition between them. The proposed metric space formulation provides a geometric interpretation of uncertainty without the need for explicit probabilistic modeling. To achieve this, we introduce a multimodal latent transition model and a sensor fusion mechanism based on inverse distance weighting, allowing for the adaptive integration of multiple sensor modalities without prior knowledge of noise distributions. We empirically validate the approach on a range of RL tasks, demonstrating improved robustness to sensor noise and superior state estimation compared to baseline methods. Our experiments show enhanced performance of an RL agent via the learned representation, eliminating the need of explicit noise augmentation. The presented results suggest that leveraging transition-aware metric spaces provides a principled and scalable solution for robust state estimation in sequential decision-making.

Contributions by the author: designed the method, co-implemented the model, designed and implemented the experiments and wrote the paper.

Paper E

Walking on the Fiber: A Simple Geometric Approximation for Bayesian Neural Networks

A. Reichlin, M. Vasco and D. Kragic
In *Transactions on Machine Learning Research (TMLR), 2025*

Abstract: Bayesian Neural Networks provide a principled framework for uncertainty quantification by modeling the posterior distribution of network parameters. However, exact posterior inference is computationally intractable, and widely used approximations like the Laplace method struggle with scalability and posterior accuracy in modern deep networks. In this work, we revisit sampling techniques for posterior exploration, proposing a simple variation tailored to efficiently sample from the posterior in over-parameterized networks by leveraging the low-dimensional structure of loss minima. Building on this, we introduce a model that learns a deformation of the parameter space, enabling rapid posterior sampling without requiring iterative methods. Empirical results demonstrate that our approach achieves competitive posterior approximations with improved scalability compared to recent refinement techniques. These contributions provide a practical alternative for Bayesian inference in deep learning.

Contributions by the author: designed the method, implemented the model, designed and implemented the experiments and wrote the paper.

Paper F

Reducing Variance in Meta-Learning via Laplace Approximation for Regression Tasks

A. Reichlin*, G. Tegnér*, M. Vasco, H. Yin, M. Björkman and D. Kragic
In *Transactions on Machine Learning Research (TMLR), 2024*

Abstract: Given a finite set of sample points, meta-learning algorithms aim to learn an optimal adaptation strategy for new, unseen tasks. Often, this data can be ambiguous as it might belong to different tasks concurrently. This is particularly the case in meta-regression tasks. In such cases, the estimated adaptation strategy is subject to high variance due to the limited amount of support data for each task, which often leads to sub-optimal generalization performance. In this work, we address the problem of variance reduction in gradient-based meta-learning and formalize the class of problems prone to this, a condition we refer to as *task overlap*. Specifically, we propose a novel approach that reduces the variance of the gradient estimate by weighing each support point individually by the variance of its posterior over the parameters. To estimate the posterior, we utilize the Laplace approximation, which allows us to express the variance in terms of the curvature of the loss landscape of our meta-learner. Experimental results demonstrate the effectiveness of the proposed method and highlight the importance of variance reduction in meta-learning.

Contributions by the author: co-designed the method and its implementation, co-designed and implemented the experiments and wrote parts of the paper.

Bibliography

[1] Richard Held and Alan Hein. “Movement-produced stimulation in the development of visually guided behavior”. In: *Journal of Comparative and Physiological Psychology* 56.5 (1963), pp. 872–876.

[2] James J. Gibson. *The Senses Considered as Perceptual Systems*. Boston, MA: Houghton Mifflin, 1966.

[3] Alva Noë. *Action in Perception*. Cambridge, MA: MIT Press, 2004.

[4] Judea Pearl. *Causality: Models, Reasoning and Inference*. 2nd ed. Cambridge: Cambridge University Press, 2009.

[5] Francesco Locatello, Stefan Bauer, Mario Lucic, et al. “Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations”. In: *Proceedings of the 36th International Conference on Machine Learning*. ICML. PMLR, 2019, pp. 4114–4124.

[6] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. 2nd ed. Cambridge, MA: MIT Press, 2018.

[7] Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. “A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning”. In: *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*. Vol. 15. AISTATS. PMLR, 2011, pp. 627–635.

[8] Sergey Levine, Aviral Kumar, George Tucker, et al. “Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems”. In: *arXiv preprint arXiv:2005.01643* (2020).

[9] Paul F. Christiano, Jan Leike, Tom B. Brown, et al. “Deep Reinforcement Learning from Human Preferences”. In: *Advances in Neural Information Processing Systems*. Vol. 30. NeurIPS. 2017.

[10] Long Ouyang, Jeff Wu, Xu Jiang, et al. “Training Language Models to Follow Instructions with Human Feedback”. In: *Advances in Neural Information Processing Systems*. NeurIPS. 2022.

[11] Ruzena Bajcsy. “Active Perception”. In: *Proceedings of the IEEE* 76.8 (1988), pp. 966–1005.

[12] Yiannis Aloimonos, Isaac Weiss, and Amnon Bandyopadhyay. “Active Vision”. In: *International Journal of Computer Vision* 1.4 (1988), pp. 333–352.

[13] Jeannette Bohg, Karol Hausman, Bharath Sankaran, et al. “Interactive Perception: Leveraging Action in Perception and Perception in Action”. In: *IEEE Transactions on Robotics* 33.6 (2017), pp. 1273–1291.

[14] Karl Johan Åström. “Optimal control of Markov processes with incomplete state information I”. In: *Journal of mathematical analysis and applications* 10 (1965), pp. 174–205.

[15] Steven L Brunton and J Nathan Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022.

[16] Yoshua Bengio, Aaron Courville, and Pascal Vincent. “Representation learning: A review and new perspectives”. In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1798–1828.

[17] Joshua B Tenenbaum, Vin de Silva, and John C Langford. “A global geometric framework for nonlinear dimensionality reduction”. In: *science* 290.5500 (2000), pp. 2319–2323.

[18] G. Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[19] Kurt Hornik. “Approximation capabilities of multilayer feedforward networks”. In: *Neural networks* 4.2 (1991), pp. 251–257.

[20] Elise Van der Pol, Daniel Worrall, Herke van Hoof, et al. “Mdp homomorphic networks: Group symmetries in reinforcement learning”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 4199–4210.

[21] Amy Zhang, Rowan McAllister, Roberto Calandra, et al. “Learning invariant representations for reinforcement learning without reconstruction”. In: *arXiv preprint arXiv:2006.10742* (2020).

[22] Alec Radford, Jeffrey Wu, Rewon Child, et al. “Language models are unsupervised multitask learners”. In: *OpenAI blog* 1.8 (2019), p. 9.

[23] Mathilde Caron, Hugo Touvron, Ishan Misra, et al. “Emerging properties in self-supervised vision transformers”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 9650–9660.

[24] Alec Radford, Jong Wook Kim, Chris Hallacy, et al. “Learning transferable visual models from natural language supervision”. In: *International conference on machine learning*. PmLR. 2021, pp. 8748–8763.

[25] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).

[26] Tomas Mikolov, Kai Chen, Greg Corrado, et al. “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781* (2013).

[27] Manuel Watter, Jost Springenberg, Joschka Boedecker, et al. “Embed to control: A locally linear latent dynamics model for control from raw images”. In: *Advances in neural information processing systems* 28 (2015).

[28] Danijar Hafner, Timothy Lillicrap, Ian Fischer, et al. “Learning latent dynamics for planning from pixels”. In: *International conference on machine learning*. PMLR. 2019, pp. 2555–2565.

[29] RE Kalman. “A new approach to linear filtering and prediction problems”. In: *Trans. ASME, D* 82 (1960), pp. 35–44.

[30] Karl Johan Åström and Richard Murray. *Feedback systems: an introduction for scientists and engineers*. Princeton university press, 2021.

[31] Sergey Levine, Chelsea Finn, Trevor Darrell, et al. "End-to-end training of deep visuomotor policies". In: *Journal of Machine Learning Research* 17.39 (2016), pp. 1–40.

[32] Carlos E Garcia, David M Prett, and Manfred Morari. "Model predictive control: Theory and practice—A survey". In: *Automatica* 25.3 (1989), pp. 335–348.

[33] Alfredo Reichlin, Giovanni Luca Marchetti, Hang Yin, et al. "Back to the manifold: Recovering from out-of-distribution states". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, pp. 8660–8666.

[34] Alfredo Reichlin, Giovanni Luca Marchetti, Hang Yin, et al. "Learning Geometric Representations of Objects via Interaction". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2023, pp. 629–644.

[35] Alfredo Reichlin, Miguel Vasco, and Danica Kragic. "Walking on the Fiber: A Simple Geometric Approximation for Bayesian Neural Networks". In: *Transactions on Machine Learning Research* (2025).

[36] Alfredo Reichlin, Gustaf Tegnér, Miguel Vasco, et al. "Reducing Variance in Meta-Learning via Laplace Approximation for Regression Tasks". In: *Transactions on Machine Learning Research* (2024).

[37] Alberta Longhini, Marco Moletta, Alfredo Reichlin, et al. "Elastic Context: Encoding Elasticity for Data-driven Models of Textiles Elastic Context: Encoding Elasticity for Data-driven Models of Textiles". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 1764–1770.

[38] Alberta Longhini, Marco Moletta, Alfredo Reichlin, et al. "EDO-Net: Learning Elastic Properties of Deformable Objects from Graph Dynamics". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 3875–3881.

[39] Gustaf Tegnér, Alfredo Reichlin, Hang Yin, et al. "On the Subspace Structure of Gradient-Based Meta-Learning". In: *arXiv preprint arXiv:2207.03804* (2022).

[40] Nona Rajabi, Charles Chernik, Alfredo Reichlin, et al. "Mental face image retrieval based on a closed-loop brain-computer interface". In: *International Conference on Human-Computer Interaction*. Springer. 2023, pp. 26–45.

[41] Frederic Renard, Cyprien Courtot, Alfredo Reichlin, et al. "Model-based reinforcement learning for protein backbone design". In: *arXiv preprint arXiv:2405.01983* (2024).

[42] Taco Cohen and Max Welling. "Group equivariant convolutional networks". In: *International conference on machine learning*. PMLR. 2016, pp. 2990–2999.

[43] Giovanni Luca Marchetti, Gustaf Tegnér, Anastasiia Varava, et al. "Equivariant representation learning via class-pose decomposition". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2023, pp. 4745–4756.

[44] Sergey Levine, Peter Pastor, Alex Krizhevsky, et al. "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection". In: *The International journal of robotics research* 37.4-5 (2018), pp. 421–436.

[45] Chelsea Finn, Xin Yu Tan, Yan Duan, et al. “Deep spatial autoencoders for visuomotor learning”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 512–519.

[46] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, et al. “An algorithmic perspective on imitation learning”. In: *Foundations and Trends® in Robotics* 7.1-2 (2018), pp. 1–179.

[47] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, et al. “Imitation learning: A survey of learning methods”. In: *ACM Computing Surveys (CSUR)* 50.2 (2017), pp. 1–35.

[48] Murray Rosenblatt. “Remarks on Some Nonparametric Estimates of a Density Function”. In: *The Annals of Mathematical Statistics* 27.3 (1956), pp. 832–837.

[49] Emanuel Parzen. “On estimation of a probability density function and mode”. In: *The annals of mathematical statistics* 33.3 (1962), pp. 1065–1076.

[50] Hugo Larochelle and Iain Murray. “The neural autoregressive distribution estimator”. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 29–37.

[51] Danilo Rezende and Shakir Mohamed. “Variational inference with normalizing flows”. In: *International conference on machine learning*. PMLR. 2015, pp. 1530–1538.

[52] Christopher M Bishop. “Mixture density networks”. In: (1994).

[53] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. “Representation learning with contrastive predictive coding”. In: *arXiv preprint arXiv:1807.03748* (2018).

[54] Nobuyuki Otsu et al. “A threshold selection method from gray-level histograms”. In: *Automatica* 11.285-296 (1975), pp. 23–27.

[55] Richard Bellman. “The theory of dynamic programming”. In: *Bulletin of the American Mathematical Society* 60.6 (1954), pp. 503–515.

[56] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[57] Christopher JCH Watkins and Peter Dayan. “Q-learning”. In: *Machine learning* 8.3 (1992), pp. 279–292.

[58] Ronald J Williams. “Simple statistical gradient-following algorithms for connectionist reinforcement learning”. In: *Machine learning* 8.3 (1992), pp. 229–256.

[59] John Schulman, Philipp Moritz, Sergey Levine, et al. “High-dimensional continuous control using generalized advantage estimation”. In: *arXiv preprint arXiv:1506.02438* (2015).

[60] John Schulman, Filip Wolski, Prafulla Dhariwal, et al. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[61] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, et al. “Continuous control with deep reinforcement learning”. In: *arXiv preprint arXiv:1509.02971* (2015).

[62] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, et al. “Soft actor-critic algorithms and applications”. In: *arXiv preprint arXiv:1812.05905* (2018).

[63] Raia Hadsell, Sumit Chopra, and Yann LeCun. “Dimensionality reduction by learning an invariant mapping”. In: *2006 IEEE computer society conference*

on computer vision and pattern recognition (CVPR'06). Vol. 2. IEEE. 2006, pp. 1735–1742.

[64] Ting Chen, Simon Kornblith, Mohammad Norouzi, et al. "A simple framework for contrastive learning of visual representations". In: *International conference on machine learning*. PmLR. 2020, pp. 1597–1607.

[65] Jean-Bastien Grill, Florian Strub, Florent Altché, et al. "Bootstrap your own latent-a new approach to self-supervised learning". In: *Advances in neural information processing systems* 33 (2020), pp. 21271–21284.

[66] Tongzhou Wang and Phillip Isola. "Understanding contrastive representation learning through alignment and uniformity on the hypersphere". In: *International conference on machine learning*. PMLR. 2020, pp. 9929–9939.

[67] Jiří Matoušek. "Embedding finite metric spaces into normed spaces". In: *Lectures on Discrete Geometry*. Springer, 2002, pp. 355–400.

[68] Tom Schaul, Daniel Horgan, Karol Gregor, et al. "Universal value function approximators". In: *International conference on machine learning*. PMLR. 2015, pp. 1312–1320.

[69] Sebastian Thrun. "Probabilistic robotics". In: *Communications of the ACM* 45.3 (2002), pp. 52–57.

[70] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, et al. "Loss surfaces, mode connectivity, and fast ensembling of dnns". In: *Advances in neural information processing systems* 31 (2018).

[71] Felix Draxler, Kambis Veschgini, Manfred Salmhofer, et al. "Essentially no barriers in neural network energy landscape". In: *International conference on machine learning*. PMLR. 2018, pp. 1309–1318.

[72] Jouko Lampinen and Aki Vehtari. "Bayesian approach for neural networks—review and case studies". In: *Neural networks* 14.3 (2001), pp. 257–274.

[73] Thomas Bayes. "An essay towards solving a problem in the doctrine of chances". In: ().

[74] Alex Graves. "Practical variational inference for neural networks". In: *Advances in neural information processing systems* 24 (2011).

[75] David JC MacKay. "Choice of basis for Laplace approximation". In: *Machine learning* 33.1 (1998), pp. 77–86.

[76] Max Welling and Yee W Teh. "Bayesian learning via stochastic gradient Langevin dynamics". In: *Proceedings of the 28th international conference on machine learning (ICML-11)*. 2011, pp. 681–688.

[77] Radford M Neal et al. "MCMC using Hamiltonian dynamics". In: *Handbook of markov chain monte carlo* 2.11 (2011), p. 2.

[78] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. "Simple and scalable predictive uncertainty estimation using deep ensembles". In: *Advances in neural information processing systems* 30 (2017).

[79] Erik Daxberger, Agustinus Kristiadi, Alexander Immer, et al. "Laplace redux-effortless bayesian deep learning". In: *Advances in neural information processing systems* 34 (2021), pp. 20089–20103.

[80] Berfin Simsek, François Ged, Arthur Jacot, et al. "Geometry of the loss landscape in overparameterized neural networks: Symmetries and invariances". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 9722–9732.

[81] Marcin Andrychowicz, Misha Denil, Sergio Gomez, et al. “Learning to learn by gradient descent by gradient descent”. In: *Advances in neural information processing systems* 29 (2016).

[82] Sachin Ravi and Hugo Larochelle. “Optimization as a model for few-shot learning”. In: *International conference on learning representations*. 2017.

[83] Nick Jakobi, Phil Husbands, and Inman Harvey. “Noise and the reality gap: The use of simulation in evolutionary robotics”. In: *European conference on artificial life*. Springer. 1995, pp. 704–720.

[84] Karl Johan Åström and Peter Eykhoff. “System identification—a survey”. In: *Automatica* 7.2 (1971), pp. 123–162.

[85] Chelsea Finn, Pieter Abbeel, and Sergey Levine. “Model-agnostic meta-learning for fast adaptation of deep networks”. In: *International conference on machine learning*. PMLR. 2017, pp. 1126–1135.

[86] Eunbyung Park and Junier B Oliva. “Meta-curvature”. In: *Advances in neural information processing systems* 32 (2019).

[87] Erin Grant, Chelsea Finn, Sergey Levine, et al. “Recasting gradient-based meta-learning as hierarchical bayes”. In: *arXiv preprint arXiv:1801.08930* (2018).

Appended papers

Paper A

Paper A

Back to the Manifold: Recovering from Out-of-Distribution States

ALFREDO REICHLIN, GIOVANNI LUCA MARCHETTI, HANG YIN, ALI GHADIRZADEH AND DANICA KRAGIC

Published in
International Conference on Intelligent Robots and Systems (IROS), 2022

Abstract:
Learning from previously collected datasets of expert data offers the promise of acquiring robotic policies without unsafe and costly online explorations. However, a major challenge is a distributional shift between the states in the training dataset and the ones visited by the learned policy at the test time. While prior works mainly studied the distribution shift caused by the policy during the offline training, the problem of recovering from out-of-distribution states at the deployment time is not very well studied yet. We alleviate the distributional shift at the deployment time by introducing a recovery policy that brings the agent back to the training manifold whenever it steps out of the in-distribution states, e.g., due to an external perturbation. The recovery policy relies on an approximation of the training data density and a learned equivariant mapping that maps visual observations into a latent space in which translations correspond to the robot actions. We demonstrate the effectiveness of the proposed method through several manipulation experiments on a real robotic platform. Our results show that the recovery policy enables the agent to complete tasks while the behavioral cloning alone fails because of the distributional shift problem.

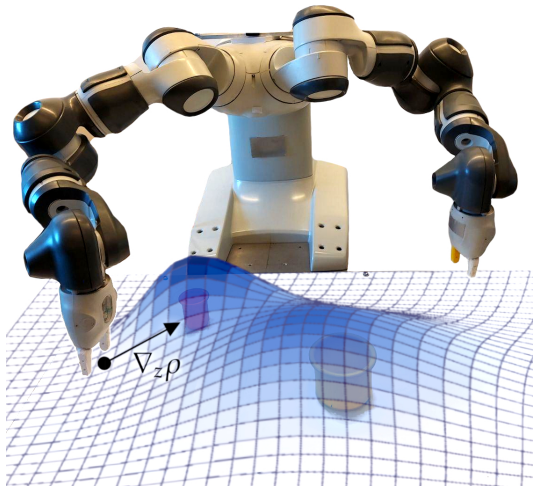


Figure A.1.1: The proposed recovery policy performs gradient ascent on the estimated density ρ of the demonstrations to get the agent back in-distribution.

A.1 Introduction

Data-driven methods in robotics, including reinforcement learning (RL), are often challenged by expensive, slow and unsafe data collection on real systems [1]. Offline solutions, such as *Offline RL* [2] and *Behavior Cloning* (BC) [3], learn a control policy from a pre-collected dataset hence avoiding the problems of interacting with a physical robot. However, learning from a fixed offline dataset may compromise the capacity of the learner on dealing with novel situations not contained in the training dataset at the test time. Querying the trained policy on such out-of-distribution (OOD) inputs can exacerbate the compounding of the errors when the policy is subsequently applied to states evolved according to the previous state and action [4]. Offline RL avoids this problem by constraining the learned policy to deviate minimally from the policy that collected the data [5, 6, 7]. However, there is no mechanism for offline RL to recover from OOD conditions, for example, when starting at a random unseen initial state or being exposed to external perturbations during execution.

One way to improve the performance of the agent in the deployment phase is to optimize an extra objective, e.g., by minimizing the uncertainty-level of the agent in predicting the next state [8, 9] which implicitly helps the agent to stay in-distribution. However, designing such objectives is challenging and they may require learning probabilistic visual forward dynamic models that output a measure of uncertainty.

In this work, we propose a method to augment a policy trained by offline behavior cloning with a *recovery policy* whose actions are computed by a gradient ascent on

the estimated density of the training distribution. This is illustrated in Figure A.1.1, in which the robot is guided by the recovery policy providing action directions to stay in-distribution while approaching the task object. We achieve this by training a model that encodes input visual observations into a Euclidean latent space, where translations in this space correspond to robot actions, such as Cartesian displacements of the robot end-effector. The encoding has a property known as translational *equivariance* that allows for the conversion of the aforementioned gradient of the estimated training data density into an action. Therefore, the recovery design benefits from a latent representation that (1) is low-dimensional, thus amenable to density estimation, and (2) is task-agnostic, i.e., it can be shared among other tasks.

We empirically demonstrate the feasibility of the proposed method on real-robotic visuomotor policy training tasks. Compared to a behavioral cloning policy, we show that the augmented policy improves the success rate on manipulation tasks. Additionally, when the robot is externally pushed OOD, it allows to recover and successfully complete the task. We also demonstrate how the trained latent equivariant representation can be shared among several tasks, making it task-agnostic. Our main contributions are:

- A method to augment policies trained with offline data to recover from OOD conditions through the gradient of a conditional density estimator.
- An empirical evaluation of the performances of the proposed method on real-robotic manipulation tasks.

A.2 Related Work

Offline policy learning mainly fall into behavior cloning [10] and offline RL [2]. The classic formulation of BC has a number of limitations when learning on real-world data. The most prominent of which, is called the compounding error [11] and occurs due to the sequential nature of the learning framework. This problem is even more profound when learning from small-sized datasets. There have been a number of works targeting this, however, they generally break the fully offline formulation [4, 12, 13].

Offline RL, on the other hand, formulates the problem of learning a policy using offline data from an RL perspective. A naive application of RL on previously collected data results in either an high variance of the learning process [14] or wrong estimates of the expected return [15]. Possible solutions to this involve either constraining the learned policy to minimize the deviation from the one that collected the data (behavioral policy) [16, 5, 17] or incentivizing the policy to avoid actions on the boundary of the training distribution by changing the reward function [6, 18, 7]. Moreover, in case the agent happens to step OOD, or it is forcefully brought there, there is no explicit way to recover. Which is what we address in this work.

Safety measures in the context of a learned controller have been addressed in different forms [19]. One common thread is the identification of *safe* regions where the agent can operate and the use of a recovery policy. In [20], unknown regions are defined by the uncertainty estimate of a perception module. When the agent steps there,

a model-based reset policy is triggered. Differently from our method, they require a model of the objects and they assume the transition function is known in a subset of the state space. Other works instead assume to directly have access to a *constraint function* from the environment or approximate it from data, which quantifies the safeness of states. A policy can then be learned to actively avoid such states [21, 22] or plan a trajectory that remains in the safe region of the state space [23, 24, 25]. Having access to the constraint function or data-points in unsafe states can, however, be problematic for robotic applications. A similar work to ours proposes to learn an approximation of the tangent space of the task manifold at any point [26]. This can, in turn, be used to plan the overall trajectory. If some kind of perturbation occurs, the agent can project its current position in the manifold and plan a path to get back in. The projection is learned explicitly using a dataset of perturbed points. On the contrary, the way we learn the encoder allows us to automatically get the projection direction even from high-dimensional states like images.

State Representation for control has been widely studied in prior works [27, 28, 29]. Dividing the optimization of the representation from the policy has the advantage of easing the learning process and enables to constrain the policy formulation [30, 31]. Moreover, the learned representation can be re-used for different tasks assuming the underlying dynamics remain the same. In [32] and [33] an encoder is trained to compress the state representation while retaining all of the information. A transition model is then inferred on top of the representation to predict the dynamics of the environment. This formulation, however, produces a generic representation with no particular properties. To this end, more rigid constraints have been proposed. In [34] the representation is forced to evolve linearly in time, while in [30] the model outputs spatial features representing the observations. Moreover, by imposing a linear dynamic on the representation, the learned controller can be simplified and, under some assumptions, learned optimally [35, 36]. Unlike our method, this dynamic cannot be directly converted into an action. [37] train a variational autoencoder with the additional constraint of making the latent representation evolve according to Newtonian physics. This allows for classical controllers, like a PID, to be applied directly. In [38], they propose to learn an encoder and a transition model at the same time. Both models are learned such that the latent representation is equivariant to the transition model. Differently from all of these methods, we require our representation to be globally translational-equivariant in order to convert the gradient of the density estimator into a viable action.

A.3 Background

We study the problem of learning a policy using a Markov decision process (MDP). We assume the MDP to be fully observable and specified by the tuple (S, A, T). Here S is the set of observations representing the state of the environment, A is the set of actions the agent can take, and $T : S \times A \to S$ is the transition function governing the change in observations when the agent performs an action. We can then formulate the problem of policy training as learning a map $\pi : S \to A$ by minimizing a cost function. Throughout this paper we define the set of states S to be images and the set of actions

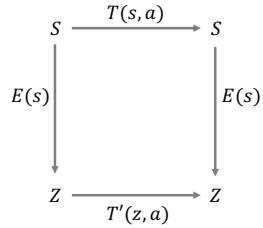


Figure A.3.1: Commutative diagram illustrating equivariance as a property of the mapping E.

$A \subseteq \mathbb{R}^n$ to be continuous translations of the agent's end-effector in the Euclidean space. A dataset $\mathcal{D}$ of experts' demonstrations is a collection of trajectories representing the robot completing a task in different conditions. This dataset can be considered as a collection of tuples $\mathcal{D} = \{(s, a, s')\}$ with $s' = T(s, a)$.

A.3.1 Behavioral Cloning

Behavioral Cloning is one of the most widely used imitation learning (IL) approaches due to its ease of implementation. It defines the cost function of the policy π as a supervised learning loss. As such, the policy's parameters θ can be inferred by minimizing the Mean Squared Error (MSE) between its estimated actions and the ones in the dataset $\mathcal{D}$:

$$\pi = \pi_{\theta^*}, \quad \theta^* = \text{argmin}_\theta \sum_{(s,a)\in\mathcal{D}} \|\pi_\theta(s) - a\|^2. \tag{A.3.1}$$

This simple IL strategy offers a number of advantages. First, it is easy to implement and stable to train. Second, it can be learned completely from offline data requiring neither access to the environment nor any knowledge of the transition model or a reward function.

A.3.2 Equivariant Mapping

In this section, we explain *equivariance* as a property of a mapping E that in our case maps an input observation s into a latent representation $z \in Z$, i.e. $z = E(s)$. Here, we consider two transition functions, $T : S \times A \to S$ and $T' : Z \times A \to Z$. The map $E : S \to Z$ is defined to be equivariant with respect to the transitions T, T' if [39, 40]:

$$E(T(s, a)) = T'(E(s), a). \tag{A.3.2}$$

Figure A.3.1 visually illustrates the equivariance property for the mapping E and the transitions T and T'. Intuitively, mapping the observation into the latent space followed by the transition T' must result in the same latent value z as applying the transition T followed by mapping the resulting observation into the latent space.

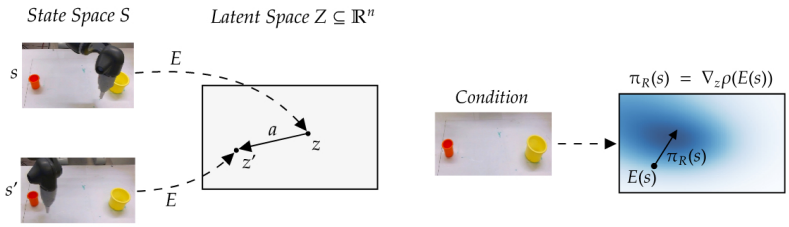


Figure A.3.2: Overview of the proposed method. **Left**: the equivariant model E maps the observation space (images) to the Euclidean latent space Z contained in the action space A. Actions of the agent correspond to translations in Z. **Right**: the latent density ρ is estimated conditioned on task-specific information. The recovery policy π_R follows the gradient of ρ and thus redirects the agent back to the training manifold.

A.4 Method

In this section, we introduce our method that augments a policy trained with BC with a recovery strategy which can bring the agent back *in-distribution*, where the BC policy performs optimally. We achieve this by introducing a *recovery policy* $\pi_R : S \to A$ whose actions move the agent to the training manifold, i.e., within the support of the observations in the training dataset. The intuition is that the agent should follow the recovery policy to recover from OOD states, and the BC policy when in-distribution to complete the task. Therefore, we propose to compute the actions given by each policy, and find the weighted sum of the actions to obtain the final policy output. The weights are computed according to a normalized estimation of the density of the training states. However, as we describe in section A.4.2, we estimate the density by first mapping the observation into a latent space $z = E(s)$, and then computing the density given the latent variable z as the input $\rho(z)$. To ensure values in $(0, 1)$, the estimated density is normalized using a sigmoid function $\bar{\rho}(z) = 1/(1 + \exp(-(\rho(z) + \epsilon)/\tau))$ with an appropriate offset ϵ and temperature τ parameters, and then used as the following to compute the output of the augmented policy $\tilde{\pi}$:

$$\tilde{\pi}(s) = \bar{\rho}(z)\pi(s) + (1 - \bar{\rho}(z))\,\pi_R(s). \tag{A.4.1}$$

In the following sections, we first introduce our proposed equivariant mapping which maps raw visual observations into a latent space in which translation corresponds to the robot actions. Then, we introduce our method to estimate the density of the training observations in the latent space. Finally, we describe how the recovery policy is constructed based on the learned equivariant mapping.

A.4.1 Learning an Equivariant Mapping

We propose to learn a low-dimensional representation of the input visual observations by explicitly learning an equivariant mapping $E : S \to Z$. As we describe in section A.4.3, we exploit the equivariant property of the mapping to construct the recovery

policy. Besides, learning a low-dimensional representation of visual inputs can also help in estimating the density of the training data.

Given the transition of the MDP $T(s, a)$, we consider the following transition in the latent space: $T'(z, a) = z + a$. This is because the robot actions *translate* the end-effector in the Euclidean space. We refer to Section A.6 for a discussion on how this can be generalized to actions beyond translations of an end-effector of a manipulator. We want to learn an equivariant mapping with respect to T, T' i.e., E has to satisfy the following version of Equation A.3.2:

$$E(T(s, a)) = E(s) + a. \tag{A.4.2}$$

As shown in Figure A.3.2 (left), E implements translational equivariance since it converts transitions into translations. Note that Equation A.4.2 assumes that A and Z share the same ambient space $\mathbb{R}^n$. The equivariant model E is trained on states and actions well-distributed within the environment i.e., a dataset $\mathcal{D}'$ is pre-collected independently from the specific task. As long as the scene composed of the extrinsic objects in the robot's environment remains constant, E can be deployed in different tasks. In order to ensure a correct representation, $\mathcal{D}'$ needs to be distributed as uniformly as possible. The mapping E is parameterized by a neural network $E = E_\varphi$ with output space $\mathbb{R}^n$ and optimized by minimizing the following objective function on the dataset $\mathcal{D}'$:

$$\varphi^* = \text{argmin}_\varphi \sum_{(s,a,s') \in \mathcal{D}'} \|E_\varphi(s') - E_\varphi(s) - a\|^2. \tag{A.4.3}$$

A.4.2 Estimating the Density of the Training States

We estimate the probability density of the agent being within the support of the training data by learning a parametric *density estimator*. The density estimator needs to be conditioned on the position of the manipulation objects for the task. It is thus conditioned on the image observation for the initial configuration of the manipulation task (Figure A.3.2, right). We use *Mixture Density Networks* (MDN) [41], which estimate a conditional Gaussian mixture density. The MDN outputs the density in the form of means μ_i, (diagonal) co-variances σ_i and weights w_i of a mixture of Gaussians $\mathcal{N}(z;\ \mu_i, \sigma_i)$. The density in the point z can then be computed as follows:

$$\rho(z) = \sum_{i=1}^{N} w_i \mathcal{N}(z;\ \mu_i, \sigma_i). \tag{A.4.4}$$

The MDN is trained by minimizing the average negative log-likelihood of ρ over the observations in the training dataset $\mathcal{D}$.

A.4.3 Recovery Policy

The recovery policy is responsible to output actions that bring the agent closer to states within the support of the training data. This is done by finding an action that brings

the agent into a state with higher estimated density. This is equivalent to performing gradient ascent on the estimated density function in the latent space. Because of the translational-equivariant property of our mapping, i.e., $z' = E(s') = E(s) + a = z + a$, the gradient $\nabla_z \rho(z)$ is equal to a robot action that moves the agent to higher density states. Therefore, we can simply define the output of the recovery policy for an observation s as:

$$\pi_R(s) = \eta \nabla_z \, \rho(E(s)) \tag{A.4.5}$$

where η is a scale parameter. Once the density of states has been estimated in the latent space, the recovery policy can be implemented accordingly without any further training phase.

A.5 Experiments

In order to assess the effectiveness of the recovery of the proposed model, we present the results of a number of experiments. First, the experimental setup is described, then details on each of the experiments are presented.

- The first experiment compares the performances of a BC agent with and without recovery on a robotic manipulation task.
- The second experiment tests the ability of a BC agent learned from noisy data, with and without recovery, in performing the same task.
- The third experiment compares the ability of BC, with and without recovery, in resuming a task if brought forcefully OOD.
- The fourth experiment involves solving a different task. The goal of this experiment is to show that the representation is agnostic to the task.

A.5.1 Experimental Setup

All the experiments are performed in the real world using a YuMi-IRB 14000 collaborative robot by ABB. We record all the data through teleoperation of the robot by a human. To implement the teleoperation system we used a virtual reality (VR) system connected to the robot's controllers. Teleoperation through VR has, in fact, proved to be a viable option for robotics applications thanks mainly to its ease in use and speed of data collection [42, 43, 44, 45]. In particular, for our experiments, we interface an Oculus Quest 2 device to the robot's operating system.

To record the data, the human operator stands in front of the robot and operates the VR hand controller to command desired velocities to the end-effector of the robotic arm. Commands are thus mirrored with respect to the human perspective. Velocity commands are sent with a frequency of 10Hz to the robot that then translates them to the equivalent joint velocities. In all of the experiments, the velocities are just translations in space as no angular velocities are considered, meaning that $A \subseteq \mathbb{R}^3$. Images, representing the state of the system by a static camera placed in front of the robot in coordination with the VR commands.

The setup is the same across all experiments and its shown in Figure A.5.1. The robot is placed in front of a table where two objects are placed, a small plastic orange cylinder, the *manipulated object*, and a bigger plastic yellow cylinder, the *target object*. Throughout all the experiments the target object is never moved while the robot needs to interact with the manipulated object.

A.5.2 Networks' Architectures

The equivariant encoder E is parameterized by 5 convolutional layers with 64 channels except the last one with 8 followed by a hidden fully-connected layer with 64 units, every hidden layer is followed by a ReLU activation function. The network is trained with a learning rate of 10^{-3} using the Adam optimizer. Input images are cropped, resized and normalized before being fed to the network.

The policy is parameterized by a neural network with a ResNet18 backbone pre-trained on ImageNet and a randomly initialized fully connected head with 2 hidden layers of 64 units each. The MDN density estimator is parameterized by a CNN model of 4 layers with 64 channels and one with 8 channels followed by 3 output layers for the mean, diagonal variance and weights of the mixture of Gaussians. The MDN is trained, as stated in Section A.4.2, to minimize the negative log-likelihood of the latent representation of the equivariant encoder. The MDN is conditioned on the initial image of each trajectory and the gripper state. The reason being that the model needs to be aware of the position of the object to be picked up to output a conditional density but it should not have access to the position of the gripper. There is, in fact, a functional dependency between images where the gripper is shown and the density estimate itself. By conditioning the MDN on the current image during the roll-out, the density would collapse. Conditioning the model on the gripper state is also needed as the position of the gripper should be considered in-distribution or not based on the object being grasped or not, respectively. Training an MDN is notoriously unstable [46] and we found that adding a deconvolution decoder that maps a middle representation of the model into a reconstruction of the original image can stabilize the training. Both the policy and the MDN are trained using the Adam optimizer with a learning rate of 10^{-4}. The other hyper-parameters used for this experiment are the following: $\eta = 0.05$, $\epsilon = 2.0$ and $\tau = 0.5$.

A.5.3 Equivariant Encoder

The equivariant representation used for the density estimator is agnostic to the task and can be learned a priori.

For the dataset $\mathcal{D}'$, we collect 6 trajectories of the robotic arm moving uniformly in the space and interacting with the manipulated object for a total of 1741 steps. Interaction with the objects is needed in order to make the learned encoder invariant to its position. Equivariance is defined with respect to the robot's actions only and the learned representation should not be sensitive to the object's position.

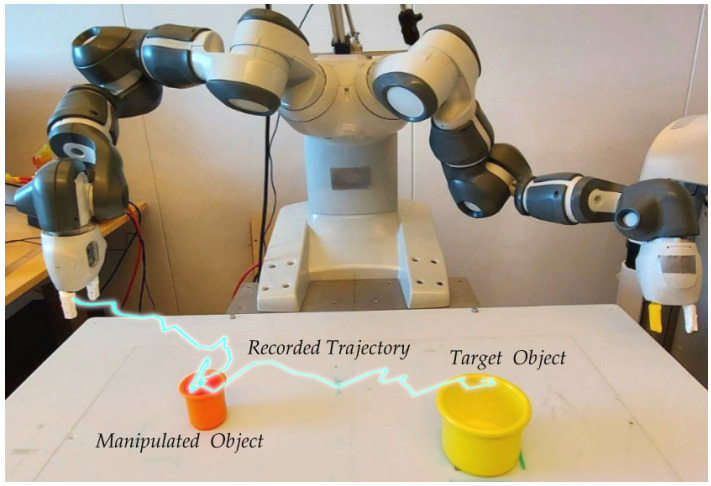


Figure A.5.1: Experimental setup for the pick-and-drop task. The small bin (*manipulated object*) has to be dropped inside the larger one (*target object*). A trajectory recorded by the expert via teleoperation is also displayed.

Because the actions are translations in the Euclidean 3D space the equivariant representation will correspond to points in 3D where the gripper of the robot is located (the central point of the actual movement). The representation preserves distances so the scale is one to one with the actions' magnitude. This can be seen in Figure A.5.1 where the expert's trajectories have been mapped into the latent space of the encoder and projected in 2D. In fact, the shape of the table and the relative position of the objects are maintained. For interpretability, we force the initial configuration of the robot to be the origin of the representation by including a second term in the loss of Equation A.4.3.

A.5.4 Pick-and-Drop Experiment

Experiment description

In the first experiment, the goal is for the robot to pick the manipulated object, move it on top of the target object and drop it inside. Here the actions are the combination of the gripper's velocity and a binary value representing the state of the gripper (either open or close). The initial position of the manipulated object is initialized randomly in the first half of the table while the target object is kept fixed on the other half of the table. A dataset of 120 trajectories of a human demonstrating the task is used to train the model. The dataset accounts for 5526 steps in total and is used to train both the behavioral cloning policy and the density estimator. The demonstrations are collected using the VR teleoperation system described above. However, these demonstrations cannot be assumed optimal due to the noise in the teleoperation system and the non-Markovianity of the environment. In fact, there are two elements here that break the

Markovian assumption. The first is the current velocity and acceleration the robot has before giving it a command. The next state will vary depending on these properties that are not inferable from one single image. The second element is the inverse kinematic module of the robot. The resulting actual displacement of the end effector will also depend on the current state of the joints that is not fully observable from the images.

Results

In the first set of experiments, we test the proposed method against the imitation learning policy without the recovery term. The models are evaluated on 20 trials with the manipulated object in different positions. Performances are based on their ability to successfully grasp the manipulated object and their ability to then drop it inside the target bin. Table A.5.1 shows the results of this experiment. The standard BC model manages to complete the task only one-fourth of the time. On the other hand, coupling the same policy with the proposed recovery lets the agent adjust its position every time it makes a mistake that would bring it OOD. This results in a much higher success rate.

Further, to assess the robustness of the recovery policy, we test it in noisy conditions. We train a second BC policy on the same dataset but with all the actions shifted by one step with respect to the images. We effectively simulate a data gathering scenario where there is a delay between the camera sensor and the actual movement of the robot. Two subsequent states are not connected by the saved action. However, because of the uniformity of the task, the real action does not differ considerably and the overall motion of the agent keeps a similar behavior. Nonetheless, a policy trained on this sub-optimal dataset does not manage to solve the task even once. On the other hand, by coupling the same policy with the proposed recovery module the performances are quite unchanged with respect to the correct dataset case, see Table A.5.1.

Lastly, we test the BC policy with and without recovery by applying a random displacement to the robot. At the beginning of the task, we move the gripper in a random direction and let it continue the task from there. The new position could be outside of the training distribution, making the imitation learning policy behave randomly. The recovery policy instead can climb back to the training manifold and continue with the task normally. The object is initialized in a position where the imitation learning agent is able to complete the task. As shown in Table A.5.1, the learned policy suffers considerably from this kind of perturbations. On the other hand, when coupled with the proposed recovery it can always get back in and most of the time complete the task.

A.5.5 Pushing Experiment

The second set of experiments involves the same environment with unchanged dynamics. The goal is for the agent to insert the gripper's tip into the manipulated object and push it all the way towards the target object. In this experiment, the gripper is always closed. Because the robotic arm moves in the same way and only the manipulated

MODEL	GRASP	DROP
PICK-AND-DROP		
BC	25%	25%
BC with Recovery	70%	55%
SHIFTED ACTIONS PICK-AND-DROP		
BC	0%	0%
BC with Recovery	70%	50%
PERTURBED PICK-AND-DROP		
BC	30%	10%
BC with Recovery	100%	70%

Table A.5.1: Results of the pick-and-drop task for a standard behavioral cloning policy with and without the proposed recovery. The models are compared in their ability of picking the manipulated object correctly and dropping it inside the target object. The table shows results on models learned on correct demonstrations as well as demonstrations with actions that are shifted by one time step with respect to the corresponding images. Models are also tested on their ability to overcome perturbations while performing the task.

PUSH	
MODEL	COMPLETE
BC	20%
BC with Recovery	25%

Table A.5.2: Results of the pushing task for standard behavioral cloning policy with and without the proposed recovery. The models are compared in their ability of inserting the tip of the gripper within the manipulated object correctly and pushing it towards the target.

object is moved throughout the roll-outs, the encoder can be used without retraining. Both the imitation learning policy and the density estimator have to be retrained on the new demonstrations. For this experiment, a new dataset of 60 demonstrations is collected for a total of 3895 steps. Results in Table A.5.2 show that the encoder can be used without retraining and that the recovery policy increases the performances of the learned policy.

A.6 Conclusions and Future Work

We proposed to couple an agent learned on experts' demonstrations with a recovery policy to keep it within the training data. This is achieved by explicitly modeling the training distribution with a density estimator and bypassing the agent's action whenever the current state is detected to be OOD. By training an encoder to be equivariant to

the agent's actions, the recovery policy can be formulated as a form of gradient ascent on the density estimate.

We applied the proposed methodology to a robotic manipulator whose actions correspond to Euclidean translations. As a possible extension, more complex actions could be considered such as rotations of joins and end-effectors. This would involve *Lie groups* beyond the Euclidean space such as the group of rotations $\mathrm{SO}(n)$, $n = 2, 3$. A further extension of the framework lies in designing more complex recovery strategies than pure gradient ascent in order to smooth the resulting movement.

Acknowledgements

This work has been supported by the Swedish Research Council, Knut and Alice Wallenberg Foundation, European Research Council (BIRD-884807) and H2020 CANOPIES.

References

[1] Sergey Levine, Peter Pastor, Alex Krizhevsky, et al. “Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection”. In: *The International journal of robotics research* 37.4-5 (2018), pp. 421–436.

[2] Sergey Levine, Aviral Kumar, George Tucker, et al. “Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems”. In: *arXiv preprint arXiv:2005.01643* (2020).

[3] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, et al. “Imitation learning: A survey of learning methods”. In: *ACM Computing Surveys (CSUR)* 50.2 (2017), pp. 1–35.

[4] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. “A reduction of imitation learning and structured prediction to no-regret online learning”. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 627–635.

[5] Wenxuan Zhou, Sujay Bajracharya, and David Held. “Plas: Latent action space for offline reinforcement learning”. In: *arXiv preprint arXiv:2011.07213* (2020).

[6] Aviral Kumar, Aurick Zhou, George Tucker, et al. “Conservative q-learning for offline reinforcement learning”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 1179–1191.

[7] Tianhe Yu, Aviral Kumar, Rafael Rafailov, et al. “Combo: Conservative offline model-based policy optimization”. In: *arXiv preprint arXiv:2102.08363* (2021).

[8] Ali Ghadirzadeh, Judith Bütepage, Atsuto Maki, et al. “A sensorimotor reinforcement learning framework for physical human-robot interaction”. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 2682–2688.

[9] Gregory Kahn, Adam Villaflor, Vitchyr Pong, et al. “Uncertainty-aware reinforcement learning for collision avoidance”. In: *arXiv preprint arXiv:1702.01182* (2017).

[10] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, et al. “An algorithmic perspective on imitation learning”. In: *arXiv preprint arXiv:1811.06711* (2018).

[11] Stephane Ross and Drew Bagnell. “Efficient Reductions for Imitation Learning”. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 2010, pp. 661–668.

[12] Pieter Abbeel and Andrew Y Ng. *Inverse Reinforcement Learning*. 2010.

[13] Jonathan Ho and Stefano Ermon. “Generative adversarial imitation learning”. In: *Advances in neural information processing systems* 29 (2016).

[14] Doina Precup. “Eligibility traces for off-policy policy evaluation”. In: *Computer Science Department Faculty Publication Series* (2000), p. 80.

[15] Aviral Kumar, Justin Fu, Matthew Soh, et al. “Stabilizing off-policy q-learning via bootstrapping error reduction”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[16] Scott Fujimoto, David Meger, and Doina Precup. “Off-policy deep reinforcement learning without exploration”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2052–2062.

[17] Xi Chen, Ali Ghadirzadeh, Tianhe Yu, et al. “Latent-Variable Advantage-Weighted Policy Optimization for Offline RL”. In: *arXiv preprint arXiv: 2203.08949* (2022).

[18] Tianhe Yu, Garrett Thomas, Lantao Yu, et al. “Mopo: Model-based offline policy optimization”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 14129–14142.

[19] Lukas Brunke, Melissa Greeff, Adam W Hall, et al. “Safe learning in robotics: From learning-based control to safe reinforcement learning”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 5 (2021).

[20] Michelle A Lee, Carlos Florensa, Jonathan Tremblay, et al. “Guided uncertainty-aware policy optimization: Combining learning and model-based strategies for sample-efficient policy learning”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 7505–7512.

[21] Brijen Thananjeyan, Ashwin Balakrishna, Suraj Nair, et al. “Recovery rl: Safe reinforcement learning with learned recovery zones”. In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 4915–4922.

[22] Krishnan Srinivasan, Benjamin Eysenbach, Sehoon Ha, et al. “Learning to be safe: Deep rl with a safety critic”. In: *arXiv preprint arXiv:2010.14603* (2020).

[23] Brijen Thananjeyan, Ashwin Balakrishna, Ugo Rosolia, et al. “Safety augmented value estimation from demonstrations (saved): Safe deep model-based rl for sparse cost robotic tasks”. In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 3612–3619.

[24] Albert Wilcox, Ashwin Balakrishna, Brijen Thananjeyan, et al. “LS3: Latent Space Safe Sets for Long-Horizon Visuomotor Control of Sparse Reward Iterative Tasks”. In: *Conference on Robot Learning*. PMLR. 2022, pp. 959–969.

[25] Ioanna Mitsioni, Pouria Tajvar, Danica Kragic, et al. “Safe Data-Driven Contact-Rich Manipulation”. In: *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*. IEEE. 2021, pp. 120–127.

[26] Miao Li, Kenji Tahara, and Aude Billard. “Learning task manifolds for constrained object manipulation”. In: *Autonomous Robots* 42.1 (2018), pp. 159–174.

[27] Timothée Lesort, Natalia Díaz-Rodríguez, Jean-Franois Goudou, et al. “State representation learning for control: An overview”. In: *Neural Networks* 108 (2018), pp. 379–392.

[28] Xi Chen, Ali Ghadirzadeh, Mårten Björkman, et al. “Adversarial feature training for generalizable robotic visuomotor control”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 1142–1148.

[29] Aleksi Hämäläinen, Karol Arndt, Ali Ghadirzadeh, et al. “Affordance learning for end-to-end visuomotor robot control”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 1781–1788.

[30] Chelsea Finn, Xin Yu Tan, Yan Duan, et al. “Deep spatial autoencoders for visuomotor learning”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 512–519.

[31] Ali Ghadirzadeh, Atsuto Maki, Danica Kragic, et al. “Deep predictive policy training using reinforcement learning”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 2351–2358.

[32] John-Alexander M Assael, Niklas Wahlström, Thomas B Schön, et al. "Data-efficient learning of feedback policies from image pixels using deep dynamical models". In: *arXiv preprint arXiv:1510.02173* (2015).

[33] David Ha and Jürgen Schmidhuber. "World models". In: *arXiv preprint arXiv:1803.10122* (2018).

[34] Ross Goroshin, Michael F Mathieu, and Yann LeCun. "Learning to linearize under uncertainty". In: *Advances in neural information processing systems* 28 (2015).

[35] Manuel Watter, Jost Springenberg, Joschka Boedecker, et al. "Embed to control: A locally linear latent dynamics model for control from raw images". In: *Advances in neural information processing systems* 28 (2015).

[36] Marvin Zhang, Sharad Vikram, Laura Smith, et al. "Solar: Deep structured representations for model-based reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 7444–7453.

[37] Miguel Jaques, Michael Burke, and Timothy M Hospedales. "NewtonianVAE: Proportional Control and Goal Identification from Pixels via Physical Latent Spaces". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 4454–4463.

[38] Thomas Kipf, Elise van der Pol, and Max Welling. "Contrastive learning of structured world models". In: *arXiv preprint arXiv:1911.12247* (2019).

[39] Taco Cohen and Max Welling. "Group equivariant convolutional networks". In: *International conference on machine learning*. PMLR. 2016, pp. 2990–2999.

[40] Giovanni Luca Marchetti, Gustaf Tegnér, Anastasiia Varava, et al. "Equivariant Representation Learning via Class-Pose Decomposition". In: *arXiv preprint arXiv:2207.03116* (2022).

[41] Christopher M Bishop. "Mixture density networks". In: (1994).

[42] Nishanth Koganti, Abdul Rahman HAG, Yusuke Iwasawa, et al. "Virtual reality as a user-friendly interface for learning from demonstrations". In: *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*. 2018, pp. 1–4.

[43] Ajay Mandlekar, Danfei Xu, Josiah Wong, et al. "What matters in learning from offline human demonstrations for robot manipulation". In: *arXiv preprint arXiv:2108.03298* (2021).

[44] Tianhao Zhang, Zoe McCarthy, Owen Jow, et al. "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 5628–5635.

[45] David Whitney, Eric Rosen, Daniel Ullman, et al. "Ros reality: A virtual reality framework using consumer-grade hardware for ros-enabled robots". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 1–9.

[46] Osama Makansi, Eddy Ilg, Ozgun Cicek, et al. "Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 7144–7153.

Paper B

Paper B

Learning Geometric Representations of Objects via Interaction

ALFREDO REICHLIN*, GIOVANNI LUCA MARCHETTI*,
HANG YIN ANASTASIIA VARAVA AND DANICA KRAGIC

This chapter is based on the author's accepted manuscript of: A. Reichlin et al., "Learning Geometric Representations of Objects via Interaction," in Proceedings of ECML PKDD 2023. The Version of Record was first published in Part IV, LNCS 14172 by Springer Nature.

Abstract:
We address the problem of learning representations from observations of a scene involving an agent and an external object the agent interacts with. To this end, we propose a representation learning framework extracting the location in physical space of both the agent and the object from unstructured observations of arbitrary nature. Our framework relies on the actions performed by the agent as the only source of supervision, while assuming that the object is displaced by the agent via unknown dynamics. We provide a theoretical foundation and formally prove that an ideal learner is guaranteed to infer an isometric representation, disentangling the agent from the object and correctly extracting their locations. We evaluate empirically our framework on a variety of scenarios, showing that it outperforms vision-based approaches such as a state-of-the-art keypoint extractor. We moreover demonstrate how the extracted representations enable the agent to solve downstream tasks via reinforcement learning in an efficient manner.

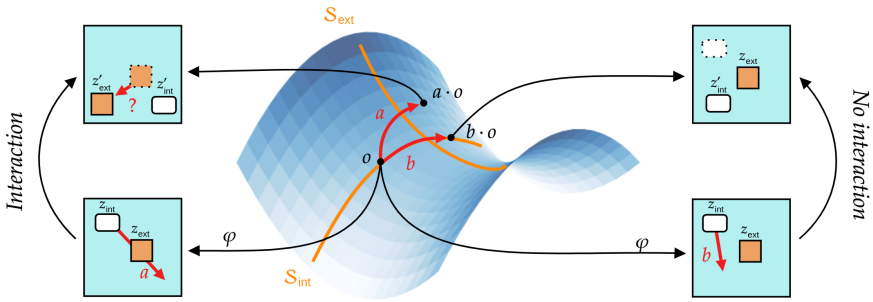


Figure B.1.1: Our framework enables to learn a representation φ recovering the geometric and disentangled state of both an agent (z_{int}, white) and an interactable object (z_{ext}, brown) from unstructured observations o (e.g., images). The only form of supervision comes from actions a, b performed by the agent, while the transition of the object (question mark) in case of interaction is unknown. In case of no interaction, the object stays invariant.

B.1 Introduction

A fundamental aspect of intelligent behavior by part of an agent is building rich and structured *representations* of the surrounding world [1]. Through structure, in fact, a representation potentially leads to semantic understanding, efficient reasoning and generalization [2]. However, in a realistic scenario an agent perceives observations of the world that are high-dimensional and unstructured e.g., images. Therefore, the ultimate goal of inferring a representation consists of extracting structure from the observed data [3]. This is challenging and in some instances requires supervision or biases. For example, it is known that *disentangling* factors of variation in data is mathematically impossible in a completely unsupervised way [4]. In order to extract structure, it is therefore necessary to design methods and paradigms relying on additional information and specific assumptions.

In the context of an agent interacting with the world, a fruitful source of information is provided by the *actions* performed and collected together with the observations. Based on this, several recent works have explored the role of actions in representation learning and proposed methods to extract structure from interaction [5, 6, 7]. The common principle underlying this line of research is encouraging the representation to replicate the effect of actions in a structured space – a property referred to as *equivariance* [1]. In particular, it has been shown in [9] that equivariance enables to extract the location of the agent in physical space, resulting in a lossless and geometric representation. The question of how to represent features of the world which are extrinsic to the agent (e.g., objects) has been left open. Such features are dynamic since they change as a consequence of interaction. They are thus challenging to capture in the representation but are essential for understanding and reasoning by part of the agent.

[1]Alternative terminologies from the literature are *World Model* [5] and *Markov Decision Process Homomorphism* [8].

In this work we consider the problem of learning representations of a scene involving an agent and an external rigid object the agent interacts with (see Figure B.1.1). We aim for a representation disentangling the agent from the object and extracting the locations of both of them in physical space. In order words, we aim for representations that are isometric w.r.t. to the geometry of the world. To this end, we focus on a scenario where the object displaces only when it comes in contact with the agent, which is realistic and practical. We make no additional assumption on the complexity of the interaction: the object is allowed to displace arbitrarily and its dynamics is unknown. Our assumption around the interaction enables to separate the problem of representing the agent – whose actions are known and available as a supervisory signal – from the problem of representing the object – whose displacement is unknown. Following this principle, we design an optimization objective relying on actions as the only form of supervision. This makes the framework general and in principle applicable to observations of arbitrary nature. We moreover provide a formalization of the problem and theoretical grounding for the method. Our core theoretical result guarantees that the representation inferred by an ideal learner recovers isometric representations as desired. We complement the theoretical analysis with an empirical investigation. Results show that our proposed representations outperform in quality of structure a state-of-the-art keypoint extractor and can be leveraged by the agent in order to solve control tasks efficiently by reinforcement learning. In summary, our contributions include:

- A representation learning framework extracting representations from observations of a scene involving an agent interacting with an object.
- A theoretical result guaranteeing that the above learning framework, when implemented by an ideal learner, infers an isometric representation for data of arbitrary nature.
- An empirical investigation of the framework on a variety of environments with comparisons to computer vision approaches (i.e., keypoint extraction) and applications to a control task.

We provide Python code implementing our framework together with all the experiments at the following public repository: `https://github.com/reichlin/GeomRepObj`. The repository additionally includes the Appendix of the present work.

B.2 Related Work

Equivariant Representation Learning. Several recent works have explored the idea of incorporating interactions into representation learning. The common principle is to infer a representation which is equivariant i.e., such that transitions in observations are replicated as transitions in the latent space. One option is to learn the latent transition end-to-end together with the representation [5, 8, 10]. This approach is however non-interpretable and the resulting representations are not guaranteed to extract any structure. Alternatively, the latent transition can be designed a priori. Linear and

affine latent transitions have been considered in [11], [6] and [7] while transitions defined by (the multiplication of) a Lie group have been discussed in [9], [12]. As shown in [9], for static scenarios (i.e., with no interactive external objects) the resulting representations are structured and completely recover the geometry of the underlying state of the agent. Our framework adheres to this line of research by modelling the latent transitions via the additive Lie group $\mathbb{R}^n$. We however further extend the representation to include external objects. Our framework thus applies to more general scenarios and dynamics while still benefiting from the geometrical guarantees.

Keypoint Extraction. When observations are images, computer vision offers a spectrum of classical approaches to extract geometric structure. In particular, extracting keypoints enables to identify any object appearing in the observed images. Popular keypoint extractors include classical non-parametric methods [13], [14] as well as modern self-supervised learning approaches [15], [16]. However, keypoints from an image provide a representation based on the geometry of the field of view or, equivalently, of the pixel plane. This means that the intrinsic three-dimensional geometry of states of objects is not preserved since the representation differs from it by an unknown projective transformation. In specific situations such transformation can still be recovered by processing the extracted keypoints. This is the case when images are in first person view w.r.t. the observer: the keypoints can then be converted into three-dimensional landmarks via methods such as bundle adjustment [17], [18]. Differently from computer vision approaches, our framework is data-agnostic and does not rely on specific priors tied to the nature of observations. It instead extracts representations based on the actions performed by the agent, which is possible due to the dynamical assumptions described in Section B.3.

Interactive Perception. The role of interaction in perception has been extensively studied in cognitive sciences and neuroscience [19, 20, 21]. Inspired by those, the field of interactive perception from robotics aims to enhance the understanding of the world by part of an artificial system via interactions [22]. Applications include active control of cameras [23] and manipulators [24] in order to improve the perception of objects [25, 26, 27]. Our work fits into the program of interactive perception since we crucially rely on performed actions as a self-supervisory signal to learn the representation. We show that the location of objects can be extracted from actions alone, albeit in a particular dynamical setting. Without interaction, this would require strong assumptions and knowledge around the data and the environment as discussed in Section B.2.

B.3 Formalism and Assumptions

In this section we introduce the relevant mathematical formalism together with the assumptions necessary for our framework. We consider the following scenario: an agent navigates in a Euclidean space and interacts in an unknown way with an external object. This means that the space of states $\mathcal{S}$ is decomposed as

$$\mathcal{S} = \mathcal{S}_{\text{int}} \times \mathcal{S}_{\text{ext}} \tag{B.3.1}$$

where $\mathcal{S}_{\text{int}}$ is the space of states of the agent (*internal* states) and $\mathcal{S}_{\text{ext}}$ is the space of states of the object (*external* states). We identify both the agent and the object with their location in the ambient space, meaning that $\mathcal{S}_{\text{int}} \subseteq \mathbb{R}^n \supseteq \mathcal{S}_{\text{ext}}$, where n is the ambient dimension. The actions that the agent performs are displacements of its state i.e., the space of actions consists of translations $\mathcal{A} = \mathbb{R}^n$. In our formalism we thus abstract objects as material points for simplicity of the theoretical analysis. The practical extension to volumetric objects together with their orientation is discussed in Section B.4.3 while the extension of agent's actions to arbitrary Lie groups is briefly discussed in Section B.6.

Our first assumption is that the agent can reach any position from any other via a sequence of actions. This translates in the following connectivity condition:
Assumption 2. (Connectedness) *The space $\mathcal{S}_{\text{int}}$ is connected and open.*

When the agent performs an action $a \in \mathcal{A}$ the state $s = (s_{\text{int}}, s_{\text{ext}})$ transitions into a novel one denoted by $a \cdot s = (s'_{\text{int}}, s'_{\text{ext}})$. Since the actions displace the agent, the internal state gets translated as $s'_{\text{int}} = s_{\text{int}} + a$.[2] However, the law governing the transition of the object $s'_{\text{ext}} = T(s, a)$ is assumed to be unknown and can be arbitrarily complex and stochastic. We stick to deterministic transitions for simplicity of explanation. Crucially, the agent does not have access to the ground-truth state s. Instead it perceives unstructured and potentially high-dimensional observations $o \in \mathcal{O}$ (e.g., images) via an unknown emission map $\omega : \mathcal{S} \to \mathcal{O}$. We assume that ω is injective so that actions induce deterministic transitions of observations, which we denote as $o' = a \cdot o$. This assumption is equivalent to total observability of the scenario and again simplifies the forthcoming discussions by avoiding the need to model stochasticity in $\mathcal{O}$.

The fundamental assumption of this work is that the dynamics of the external object revolves around *contact* i.e., the object does not displace unless it is touched by the agent. This is natural and often satisfied in practice. In order to formalize it, note that when the agent in state s_{int} performs an action $a \in \mathcal{A}$ we can imagine it moving along the open segment $\rfloor s_{\text{int}},\ s_{\text{int}} + a \lfloor = \{s_{\text{int}} + ta\}_{0<t<1}$. Our assumption then translates into (see Figure B.1.1 for a graphical depiction):
Assumption 3. (Interaction Occurs at Contact) *For all agent states $s_{\text{int}} \in S$ and actions $a \in \mathcal{A}$ it holds that $s'_{\text{ext}} = s_{\text{ext}}$ if and only if $s_{\text{ext}} \notin \rfloor s_{\text{int}},\ s_{\text{int}} + a \lfloor$.*

As such, the dynamics of the external object can be summarized as follows:

$$s'_{\text{ext}} = \begin{cases} s_{\text{ext}} & \text{if } s_{\text{ext}} \notin \rfloor s_{\text{int}},\ s_{\text{int}} + a \lfloor, \\ T(s, a) & \text{otherwise.} \end{cases} \tag{B.3.2}$$

Finally, we need to assume that interaction is possible for every state of the object i.e., the latter has to be always reachable by the agent. This is formalized via the following inclusion:
Assumption 4. (Reachability) *It holds that $\mathcal{S}_{\text{ext}} \subseteq \mathcal{S}_{\text{int}}$.*

[2] Whenever we write $a \cdot s$ we implicitly assume that the action is valid i.e., that $s_{\text{int}} + a \in \mathcal{S}_{\text{int}}$.

B.4 Method

B.4.1 Representations and Equivariance

We now outline the inference problem addressed in the present work. Given the setting introduced in Section B.3, the overall goal is to infer a *representation* of observations $\varphi : \mathcal{O} \to \mathcal{Z} = \mathcal{Z}_{\text{int}} \times \mathcal{Z}_{\text{ext}}$, where $\mathcal{Z}_{\text{int}} = \mathcal{Z}_{\text{ext}} = \mathbb{R}^n$. Ideally φ recovers the underlying inaccessible state in $\mathcal{S} \subseteq \mathcal{Z}$ and disentangles $\mathcal{S}_{\text{int}}$ from $\mathcal{S}_{\text{ext}}$. In order to achieve this, our central idea is to split the problem of representing the agent and the object. Since the actions of the agent are available, $z_{\text{int}} \in \mathcal{Z}_{\text{int}}$ can be inferred geometrically by existing representation learning methods. The representation of the object $z_{\text{ext}} \in \mathcal{Z}_{\text{ext}}$ can then be inferred based on the one of the agent by exploiting the relation between the dynamics of the two (Equation B.3.2). In order to represent the agent, we consider the fundamental concept of (translational) *equivariance*:

Definition 3. *The representation φ is said to be* equivariant *(on internal states) if for all $a \in \mathcal{A}$ and $o \in \mathcal{O}$ it holds that $z'_{\text{int}} = z_{\text{int}} + a$ where $(z_{\text{int}}, z_{\text{ext}}) = \varphi(o)$ and $(z'_{\text{int}}, z'_{\text{ext}}) = \varphi(a \cdot o)$.*

We remark that Definition 3 refers to internal states only, making our terminology around equivariance unconventional. As observed in previous work [9], equivariance guarantees a faithful representation of internal states. Indeed if φ is equivariant then z_{int} differs from s_{int} by a constant vector. This means that the representation of internal states is a translation of ground-truth ones and as such is lossless (i.e., bijective) and isometrically recovers the geometry of $\mathcal{S}_{\text{int}}$.

The above principle can be leveraged in order to learn a representation of external states with the same benefits as the representation of internal ones. Since the external object displaces only when it comes in contact with the agent (Assumption 3), the intuition is that z_{ext} can be inferred by aligning it with z_{int}. The following theoretical result formalizes the possibility of learning such representations and traces the foundation of our learning framework.

Theorem 3. *Suppose that the representation $\varphi : \mathcal{O} \to \mathcal{Z}$ satisfies:*

1. *φ is equivariant (Definition 3),*
2. *φ is injective,*
3. *for all $o \in \mathcal{O}$ and $a \in \mathcal{A}$ it holds that either $z'_{\text{ext}} = z_{\text{ext}}$ or $z_{\text{ext}} \in \lfloor z_{\text{int}}, z_{\text{int}} + a \rfloor$ where $(z_{\text{int}}, z_{\text{ext}}) = \varphi(o)$ and $(z'_{\text{int}}, z'_{\text{ext}}) = \varphi(a \cdot o)$.*

Then $\varphi \circ \omega$ is a translation i.e., there is a constant vector $h \in \mathbb{R}^n$ such that for all $s \in \mathcal{S}$ it holds that $\varphi(\omega(s)) = s + h$. In particular, $\varphi \circ \omega$ is an isometry w.r.t. the Euclidean metric on both $\mathcal{S}$ and $\mathcal{Z}$.

We refer to the Appendix for a proof. Theorem 3 states that if the conditions $1. - 3.$ are satisfied (together with the assumptions stated in Section B.3) then the representation recovers the inaccessible state up to a translation and thus isometrically preserves the geometry of the environment. All the conditions from Theorem 3 refer to properties of φ depending on observations and the effect of actions on them, which are accessible in

practice. The goal of the forthcoming section is to describe how these conditions can be enforced on φ by optimizing a system of losses.

B.4.2 Learning the Representation

In this section we describe a viable implementation of a representation learning framework adhering to the conditions of Theorem 3. We model the representation learner $\varphi = (\varphi_{\text{int}}, \varphi_{\text{ext}})$ as two parameterized functions $\varphi_{\text{int}} : \mathcal{O} \to \mathcal{Z}_{\text{int}}$, $\varphi_{\text{ext}} : \mathcal{O} \to \mathcal{Z}_{\text{ext}}$ e.g., two deep neural network models. In order to train the models, we assume that the dataset $\mathcal{D}$ consists of transitions observed by the agent in the form of $\mathcal{D} = \{(o, a, o' = a \cdot o)\} \subseteq \mathcal{O} \times \mathcal{A} \times \mathcal{O}$. Such data can be collected by the agent autonomously exploring its environment and randomly interacting with the external object. This implies that the only form of supervision required consists of the actions performed by the agent together with their effect on the observations.

First, we propose to enforce equivariance, condition 1 from Theorem 3, by minimizing the loss:

$$\mathcal{L}_{\text{int}}(o, a, o') = d(z'_{\text{int}}, z_{\text{int}} + a) \tag{B.4.1}$$

where d is a measure of similarity on $\mathcal{Z}_{\text{int}} = \mathbb{R}^n$ and the notation is in accordance with Definition 3. Typically d is chosen as the squared Euclidean distance as described in previous work [6, 5].

Next, we focus on the representation of the external object. As stated before, the dataset consists of transitions either with or without interaction. When an interaction occurs, z_{ext} should belong to the segment $\lfloor z_{\text{int}}, z_{\text{int}} + a \rfloor$. When it doesn't, the representation should be invariant i.e., $z_{\text{ext}} = z'_{\text{ext}}$. These two cases are outlined in condition 2 of Theorem 3 and can be enforced via the following losses:

$$\mathcal{L}_{-}(o, a, o') = d(z_{\text{ext}}, z'_{\text{ext}}) \qquad \mathcal{L}_{+}(o, a, o') = d(z_{\text{ext}}, \lfloor z_{\text{int}}, z_{\text{int}} + a \rfloor). \tag{B.4.2}$$

The distance involved in $\mathcal{L}_{+}$ represents a point-to-set metric and is typically set as $d(z, E) = \inf_{x \in E} d(z, x)$. The latter has a simple explicit expression in the case E is a segment.

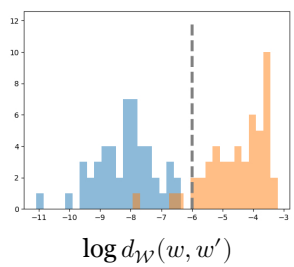


Figure B.4.1: Histograms of the log-distances in $\mathcal{W}$. Colors indicate whether interaction occurs (orange) or not (blue). The dotted line represents the threshold from Otsu's algorithm.

However, the data contains no information on whether interaction occurs or not. It is, therefore, necessary to design a procedure determining when to optimize $\mathcal{L}_+$ and $\mathcal{L}_-$. To this end, we propose to train a parallel model $\varphi_{\text{cont}} : \mathcal{O} \to \mathcal{W}$ with latent *contrastive representation* $\mathcal{W}$ (potentially different from $\mathcal{Z}$). This is trained to attract $w = \varphi_{\text{cont}}(o)$ to $w' = \varphi_{\text{cont}}(o')$ while forcing injectivity of φ (condition 2 from Theorem 3). To this end, we stick to the popular *InfoNCE* loss from contrastive learning literature [28]:

$$\mathcal{L}_{\text{cont}}(o, o') = d_{\mathcal{W}}(w, w') + \log \mathbb{E}_{o''} \left[e^{-d_{\mathcal{W}}(w', w'') - d(z'_{\text{int}}, z''_{\text{int}})} \right] \tag{B.4.3}$$

where o'' is marginalized from $\mathcal{D}$. The second summand of Equation B.4.3 encourages the joint encodings (z_{int}, w) to spread apart and thus encourages φ to be injective. Since subsequent observations where interaction does not occur share the same external state, these will lie closer in $\mathcal{W}$ than the ones where interaction occurs. This enables to exploit distances in $\mathcal{W}$ in order to choose whether to optimize $\mathcal{L}_-$ or $\mathcal{L}_+$. We propose to partition (the given batch of) the dataset in two disjoint classes $\mathcal{D} = C_- \sqcup C_+$ by applying a natural thresholding algorithm to the quantities $d_{\mathcal{W}}(w, w')$. This can be achieved via one-dimensional 2-means clustering, which is equivalent to Otsu's algorithm [29] (see Figure B.4.1 for an illustration). We then optimize:

$$\mathcal{L}_{\text{ext}}(o, a, o') = \begin{cases} \mathcal{L}_-(o, a, o') & \text{if } (o, a, o') \in C_-, \\ \mathcal{L}_+(o, a, o') & \text{if } (o, a, o') \in C_+. \end{cases} \tag{B.4.4}$$

In summary, the total loss minimized by the models $(\varphi_{\text{int}}, \varphi_{\text{ext}}, \varphi_{\text{cont}})$ w.r.t. the respective parameters is (see the pseudocode included in the Appendix):

$$\mathcal{L} = \mathbb{E}_{(o,a,o') \sim \mathcal{D}} [\mathcal{L}_{\text{int}}(o, a, o') + \mathcal{L}_{\text{ext}}(o, a, o') + \mathcal{L}_{\text{cont}}(o, o')]. \tag{B.4.5}$$

B.4.3 Incorporating Volumes of Objects

So far we have abstracted the external object as a point in Euclidean space. However, the object typically manifests with a body and thus occupies a volume. Interaction and consequent displacement (Assumption 4) occur when the agent comes in contact with the boundary of the object's body. The representation thus needs to take volumetric features into account in order to faithfully extract the geometry of states.

In order to incorporate volumetric objects into our framework we propose to rely on *stochastic* outputs i.e., to design z_{ext} as a probability density over $\mathcal{Z}_{\text{ext}}$ representing (a fuzzy approximation of) the body of the object. More concretely, the output of φ_{ext} consists of (parameters of) a Gaussian distribution whose covariance matrix represents the inertia ellipsoid of the object i.e., the ellipsoidal approximation of its shape. By diagonalizing the covariance matrix via an orthonormal frame, the orientation of the object can be extracted in the form of a rotation matrix in $\mathrm{SO}(n)$. The losses of our model are naturally adapted to the stochastic setting as follows. The distance d appearing in

Equation B.4.2 is replaced with Kullback-Leibler divergence. The latter has an explicit simple expression for Gaussian densities which allows to compute $\mathcal{L}_{-}$ directly. In order to compute $\mathcal{L}_{+}$ we rely on a Monte Carlo approximation, meaning that we sample a point uniformly from the interval and set $\mathcal{L}^{+}$ as the negative log-likelihood of the point w.r.t. the density defining z_{ext}.

B.5 Experiments

We empirically investigate the performance of our framework in correctly identifying the position of an agent and of an interactive object. The overall goal of the experimental evaluation is to show that our representation is capable of extracting the geometry of states without relying on any prior knowledge of observations e.g., depth information. All the scenarios are normalized so that states lie in the unit cube. Observations are RGB images of resolution 100×100 in all the cases considered. We implement each of φ_{int}, φ_{ext} and φ_{cont} as a ResNet-18 [30] and train them for 100 epochs via the Adam optimizer with learning rate 0.001 and batch-size 128. We compare our framework with two baselines:

- *Transporter Network* [15]: a vision-based state-of-the-art unsupervised keypoint extractor. The approach heavily relies on image manipulation in order to infer regions of the pixel plane that are persistent between pairs of images. We train the model in order to extract two (normalized) keypoints representing z_{int} and z_{ext} respectively.
- *Variational AutoEncoder* (VAE) [31, 32]: a popular representation learner with a standard Gaussian prior on its latent space. We impose the prior on $\mathcal{Z}_{\text{ext}}$ only, while φ_{int} is still trained via the equivariance loss (Equation B.4.1). The decoder takes the joint latent space $\mathcal{Z}$ in input. We set $\dim(\mathcal{Z}_{\text{ext}}) = 32$. This makes the representations disentangled, so that z_{int} and z_{ext} are well-defined. The resulting representation of the object is generic and is not designed to extract any specific structure from observations.

In order to evaluate the preservation of geometry we rely on the following evaluation metric $\mathcal{L}_{\text{test}}$. Given a trained representation $\varphi : \mathcal{O} \to \mathcal{Z}$ and a test set $\mathcal{D}_{\text{test}}$ of observations with known ground-truth states, we define:

$$\mathcal{L}_{\text{test}} = \mathbb{E}_{o \sim \mathcal{D}_{\text{test}}} \left[\, d(z_{\text{int}} - z_{\text{ext}}, s_{\text{int}} - s_{\text{ext}}) \, \right] \tag{B.5.1}$$

where d is the squared Euclidean distance. Since both our framework and (the encoder of) VAE have stochastic outputs (see Section B.4.3), we set z_{ext} as the mean of the corresponding Gaussian distribution. Equation B.5.1 measures the quality of preservation of the relative position between the agent and the object by part of the representation. When $\mathcal{L}_{\text{test}} = 0$, φ is an isometry (w.r.t. the Euclidean metric) and thus recovers the geometry of states. The translational invariance of $\mathcal{L}_{\text{test}}$ makes the comparison agnostic to any reference frame eventually inferred by the given learner.

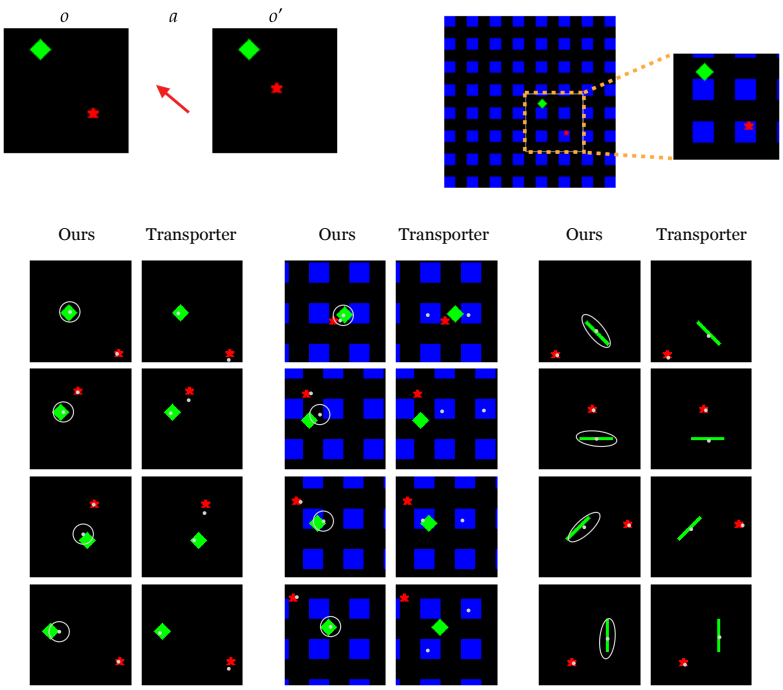


Figure B.5.1: Top: Visualization of the dataset from the Sprites experiment. On the left, an example of a datapoint $(o, a, o') \in \mathcal{D}$. On the right, an example of an observation from the second version of the dataset where a dynamic background is added as a visual distractor. **Bottom:** Comparison of z_{int}, z_{ext} (gray dots, with the ellipse representing the learned std) extracted via our model and the Transporter network on the three versions of the Sprites dataset: vanilla version (left), with dynamic background (middle) and with anisotropic object (right).

B.5.1 Sprites

For the first experiment we procedurally generate images of two sprites (the agent and the object) moving on a black background (see Figure B.5.1, top-left). Between images, the agent (red figure) moves according to a known action. If the agent comes in contact with the object (green diamond) during the execution of the action (see Assumption 3) the object is randomly displaced on the next image. In other words, the object's transition function $T(s, a)$ is stochastic with a uniform distribution. Such a completely stochastic dynamics highlights the independence of the displacement of the agent w.r.t. the one of the object. We generate the following two additional versions of the dataset:

- A version with *dynamic background*. Images are now overlaid on top of a nine-times larger second image (blue squares in Figure B.5.1, top-right). The field of

view and thus the background moves together with the agent. The background behaves as a visual distractor and makes it challenging to extract structure (e.g., keypoints) via computer vision.

- A version with *anisotropic object*. The latter is now a rectangle with one significantly longer side. Besides translating, the object rotates as well when interaction occurs. The goal here is showcasing the ability of our model in inferring the orientation of the object as described in Section B.4.3.

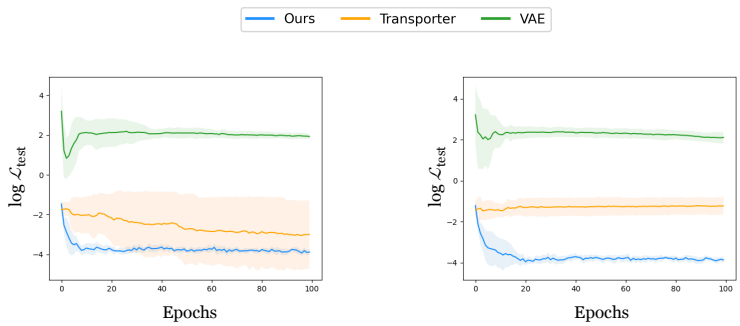


Figure B.5.2: Log-scale plots of the evaluation metric (Equation B.5.1) as the training progresses for the Sprite experiment. The curves display mean and std (for 10 experimental runs). **Left**: vanilla version of the dataset. **Right:** version with a dynamic background.

Figure B.5.2 displays the analytic comparison of the performances between our model and the baselines in terms of the evaluation metric (Equation B.5.1). The plot is in log-scale for visualization purposes. Moreover, Figure B.5.1 (bottom) reports a qualitative comparison between our model and the Transporter network. As can be seen, for the simpler version of the experiment (plot on the left) both our model and the Transporter network successfully achieve low error and recover the geometry of both the agent and the object. Note that the Transporter network converges slowly and with high variance (Figure B.5.2, left). This is probably due to the presence of a decoder in its architecture. Our framework instead involves losses designed directly in the latent space, avoiding an additional model to decode observations. As expected, VAE achieves significantly worse performances because of the lack of structure in its representation. As can be seen from Figure B.5.1 (bottom-right), when the object is anisotropic our model correctly infers its orientation by encoding it into the covariance of the learned Gaussian distribution. The Transporter network instead places a keypoint on the barycenter of the object and is therefore unable to recover the orientation.

For the more challenging version of the experiment with dynamic background, the transporter is not able to extract the expected keypoints. As can be seen from Figure B.5.1 (bottom-middle), the distracting background causes the model to focus on regions of the image not corresponding to the agent and the object. This is reflected by a significantly higher error (and variance) w.r.t. our framework (Figure B.5.2, right).

The latter still infers the correct representation and preserves geometry. This empirically confirms that our model is robust to visual distractors since it does not rely on any data-specific feature or structure.

B.5.2 Soccer

For the second experiment we test our framework on an environment consisting of an agent on a soccer field colliding with a ball (see Figure B.5.3, left). The scene is generated and rendered via the Unity engine. The physics of the ball is simulated realistically: in case of contact, rolling takes gravity and friction into account. Note that even though the scene is generated via three-dimensional rendering, the (inaccessible) state space is still two-dimensional since the agent navigates on the field. We generate two datasets of 10000 triples $(o, a, o' = a \cdot o)$ with observations of different nature. The first one consists of views in third-person perspective from a fixed external camera. In the second one, observations are four views in first-person perspective from four cameras attached on top of the agent and pointing in the 4 cardinal directions. We refer to Figure B.5.3 (left) for a visualization of the two types of observations. In Figure B.5.3 (right), we report visualizations of the learned representations. The extracted representation of our proposed method depends solely on the geometry of the problem at hand rather than the nature of the observation. The learned representation is thus identical when learned from the third-person dataset or the first-person one, as shown in B.5.3 (right).

Figure B.5.4 (left) displays the comparison of the performances between our model and the baselines in terms of the evaluation metric (Equation B.5.1). The Transporter network is trained on observations in third person and as can be seen, correctly extracts the keypoints on the *pixel plane*. As discussed in Section B.2, such a plane differs from $\mathcal{S}_{\text{int}}$ by an unknown projective (and thus non-isometric) transformation. This means that despite the successful keypoint extraction, the geometry of the state space is not preserved, which is reflected by the high error on the plot. This is a general limitation of vision-based approaches: they are unable to recover the intrinsic geometry due to perspective in the case of a three-dimensional scene. Differently from that, our framework extracts an isometric representation and achieves low error independently from the type of observations.

B.5.3 Control Task

In our last experiment we showcase the benefits of our representations in solving downstream control tasks. The motivation is that a geometric and low-dimensional representation improves efficiency and generalization compared to solving the task directly from observations. To this end we design a control task for the Soccer environment consisting in kicking the ball *into the goal*. The reward is given by the negative distance between the (barycenter of the) ball and the (barycenter of the) goal. Observations are views in third person perspective. In each episode the agent and the ball are initially placed in a random location while the ball is placed in the center. The maximum episode length is 20 steps.

We train a number of models via the popular reinforcement learning method *Proximal Policy Optimization* (PPO; [33]). One model (*End-to-End*) receives raw observations as inputs. The others operate on pre-trained representations z given by the Transporter network, the VAE and our method respectively. All the models implement a comparable architecture for a fair comparison.

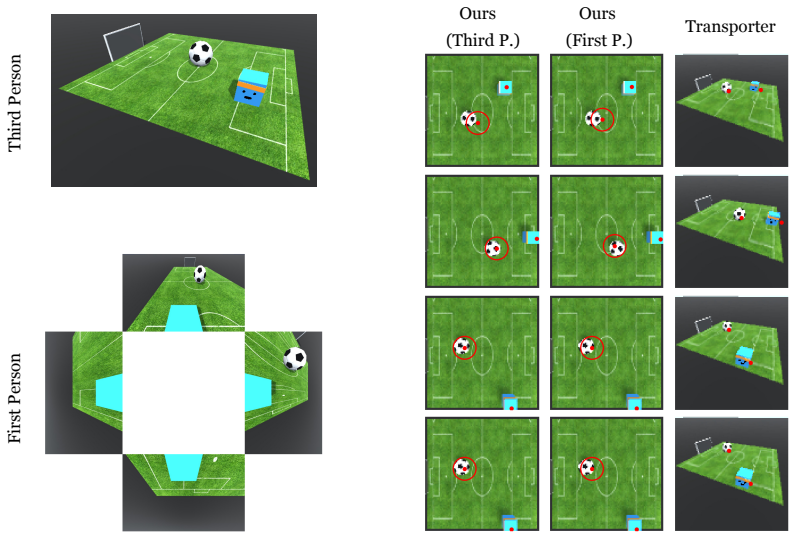


Figure B.5.3: Left: an example of the two types of observations (third and first person respectively) from the Soccer experiment. **Right**: visual comparison of z_{int}, z_{ext} (red dots) extracted via our model (from third-person view and first-person view) and the Transporter network. For our model, we overlap the representation to a view of the scene from the top instead of the original observation.

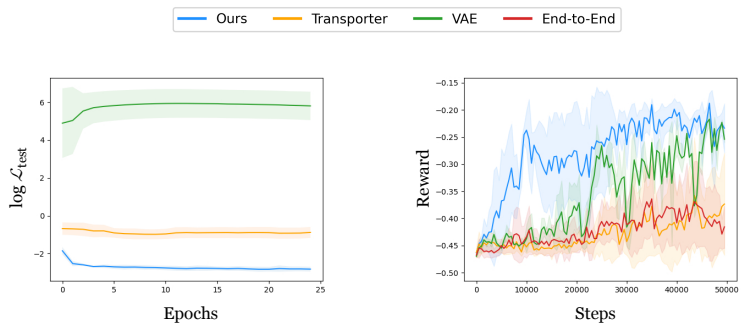


Figure B.5.4: Left: log-scale plot of the evaluation metric as the training progresses for the Soccer experiment. Observations are in third person. **Right**: plot of the reward gained via reinforcement learning on top of different representations.

Figure B.5.4 (right) displays the reward gained on test episodic runs as the training by reinforcement learning progresses. As can be seen, our geometric representation enables to solve the task more efficiently than both the competing representations (Transporter and VAE) and the end-to-end model. Note that the Transporter not only does not preserve the geometry of the state space, but has the additional disadvantage that the keypoint corresponding to the agent and the object can get swapped in the output of φ. This causes indeterminacy in the representation and has a negative impact on solving the task. Due to this, the Transporter performs similarly to the end-to-end model and is outperformed by the generic and non-geometric representation given by the VAE. In conclusion, the results show that a downstream learner can significantly benefit from geometric representations of observations in order to solve downstream control tasks.

B.6 Conclusions and Future Work

In this work we proposed a novel framework for learning representations of both an agent and an object the agent interacts with. We designed a system of losses based on a theoretical principle that guarantees isometric representations independently from the nature of observations and relying on supervision from performed actions alone. We empirically investigated our framework on multiple scenarios showcasing advantages over computer vision approaches.

Throughout the work we assumed that the agent interacts with a single object. An interesting line of future investigation is extending the framework to take multiple objects into account. In the stochastic context (see Section B.4.3) an option is to model z_{ext} via multi-modal densities, with each mode corresponding to an object. As an additional line for future investigation, our framework can be extended to actions beyond translations in Euclidean space. Lie groups other than $\mathbb{R}^n$ often arise in practice. For example, if the agent is able to rotate its body then (a factor of) the space of actions has to contain the group of rotations $\mathrm{SO}(n)$, $n = 2, 3$. Thus, a framework where actions (and consequently states) are represented in general Lie groups defines a useful and interesting extension.

Acknowledgements

This work was supported by the Swedish Research Council, the Knut and Alice Wallenberg Foundation, the European Research Council (ERC-BIRD-884807) and the European Horizon 2020 CANOPIES project. Hang Yin would like to acknowledge the support by the Pioneer Centre for AI, DNRF grant number P1.

Ethical Statement

We believe that the present work does not raise specific ethical concerns. Generally speaking, however, any system endowing artificial agents with intelligent behavior may be misused e.g., for military applications. Since we propose a representation learning method enabling an agent to locate objects in an environment, this can be poten-

tially embedded into intelligent harmful systems and deployed for unethical applications.

B.7 Appendix

B.7.1 Proofs of Theoretical Results

Theorem 4. *Suppose that the representation $\varphi : \mathcal{O} \to \mathcal{Z}$ satisfies:*

1. *φ is equivariant (Definition 3),*
2. *φ is injective,*
3. *for all $o \in \mathcal{O}$ and $a \in \mathcal{A}$ it holds that $z'_{\text{ext}} \neq z_{\text{ext}}$ if and only if $z_{\text{ext}} \in \lfloor z_{\text{int}}, z_{\text{int}} + a \rfloor$ where $(z_{\text{int}}, z_{\text{ext}}) = \varphi(o)$ and $(z'_{\text{int}}, z'_{\text{ext}}) = \varphi(a \cdot o)$.*

Then $\varphi \circ \omega$ is a translation i.e., there is a constant vector $h \in \mathbb{R}^n$ such that for all $s \in \mathcal{S}$ it holds that $\varphi(\omega(s)) = s + h$. In particular, φ is an isometry w.r.t. the Euclidean metric on both $\mathcal{S}$ and $\mathcal{Z}$.

Figure B.7.1: Graphical depiction of the proof of Theorem 3.

Proof. Pick an arbitrary state $s^0 \in \mathcal{S}$ together with its represented internal state z^0_{int} and set $h = z^0_{\text{int}} - s^0_{\text{int}}$. For any state s, consider the action $a = s_{\text{int}} - s^0_{\text{int}}$. Equivariance then implies that $z_{\text{int}} = z^0_{\text{int}} + a = s_{\text{int}} + h$. This shows that the claim holds for internal states.

To prove that the same happens for external states, suppose by contradiction that there is a state s such that $z_{\text{ext}} \neq s_{\text{ext}} + h$. Consider any path during which the agent interacts with the object without passing through z_{ext}. Formally, this means considering a sequence of actions $a_1, \cdots, a_r$ such that (see Figure B.7.1, left):

- z_{ext} and $s_{\text{ext}} + h$ do not belong to $\lfloor z_{\text{int}} + a_1 + \cdots + a_{i-1}, z_{\text{int}} + a_1 + \cdots + a_i \rfloor$ for every $i = 1, \cdots, r-1$,
- z_{ext} does not belong to $\lfloor z_{\text{int}} + a_1 + \cdots + a_{r-1}, z_{\text{int}} + a_1 + \cdots + a_r \rfloor$ but $s_{\text{ext}} + h$ does.

The existence of such a path follows from Assumptions 2 and 4. After interaction the state becomes $s' = a_r \cdot (a_{r-1} \cdots (a_1 \cdot s))$ with $s'_{\text{ext}} \neq s_{\text{ext}}$ because of Assumption 3. One can then consider a path back to the initial agent's position z_{int} i.e., another sequence of actions $a_{r+1}, \cdots, a_R$ such that (see Figure B.7.1, right):

- $s'_{\text{ext}} + h$ and z_{ext} do not belong to $\lfloor z_{\text{int}} + a_1 + \cdots + a_{i-1}, z_{\text{int}} + a_1 + \cdots + a_i \rfloor$ for every $i = r+1, \cdots, R$,
- $a_1 + \cdots + a_R = 0$.

All the conditions imply together that the representation of the object remains equal to z_{ext} during the execution of the actions $a_1, \cdots, a_R$. Since the actions sum to 0, the representation of the agent does not change as well. But then $\varphi(\omega(s)) = \varphi(\omega(s_{\text{int}}, s'_{\text{ext}}))$ while $s_{\text{ext}} \neq s'_{\text{ext}}$, contraddicting injectivity. We conclude that $z_{\text{ext}} = s_{\text{ext}} + h$ and thus $z = s + h$ as desired. □

B.7.2 Pseudocode for Loss Computation

Algorithm 1 Loss Computation

Require: Batch $\mathcal{B} \subseteq \mathcal{D}$, models $\varphi_{\text{int}}, \varphi_{\text{ext}}, \varphi_{\text{cont}}$
Ensure: Loss $\mathcal{L}$
  $\mathcal{L} = 0$
  for all $(o, a, o') \in \mathcal{B}$ **do**
    Compute $z_{\text{int}} = \varphi_{\text{int}}(o)$, $z_{\text{ext}} = \varphi_{\text{ext}}(o)$, $z'_{\text{int}} = \varphi_{\text{int}}(o')$, $z'_{\text{ext}} = \varphi_{\text{ext}}(o')$, $w = \varphi_{\text{cont}}(o)$, $w' = \varphi_{\text{cont}}(o')$
  end for
  Compute the classes C_-, C_+ via Otsu's algorithm based on $\{d_{\mathcal{W}}(w, w')\}$
  for all $(o, a, o') \in \mathcal{B}$ **do**
    Compute $\mathcal{L}_{\text{int}}(o, a, o')$ via Equation B.4.1
    Compute $A = \{d_{\mathcal{W}}(w', w''),\ d(z'_{\text{int}}, z''_{\text{int}})\}$ **for** o'' marginalized from $\mathcal{B}$
    Based on A compute $\mathcal{L}_{\text{cont}}(o, o')$ via Equation B.4.3
    if $d_{\mathcal{W}}(w, w') \in C_-$ **then**
      Compute $\mathcal{L}_{\text{ext}}(o, a, o') = \mathcal{L}_-(o, a, o')$ via Equation B.4.2 (left)
    else
      Compute $\mathcal{L}_{\text{ext}}(o, a, o') = \mathcal{L}_+(o, a, o')$ via Equation B.4.2 (right)
    end if
    $\mathcal{L} \leftarrow \mathcal{L} + \mathcal{L}_{\text{int}} + \mathcal{L}_{\text{ext}} + \mathcal{L}_{\text{cont}}$
  end for

References

[1] David Ha and Jürgen Schmidhuber. "World models". In: *arXiv preprint arXiv:1803.10122* (2018).

[2] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, et al. "Building machines that learn and think like people". In: *Behavioral and brain sciences* 40 (2017).

[3] Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation learning: A review and new perspectives". In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1798–1828.

[4] Francesco Locatello, Stefan Bauer, Mario Lucic, et al. "Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations". In: *Proceedings of the 36th International Conference on Machine Learning*. ICML. PMLR, 2019, pp. 4114–4124.

[5] Thomas Kipf, Elise van der Pol, and Max Welling. "Contrastive learning of structured world models". In: *arXiv preprint arXiv:1911.12247* (2019).

[6] Arnab Kumar Mondal, Pratheeksha Nair, and Kaleem Siddiqi. "Group equivariant deep reinforcement learning". In: *arXiv preprint arXiv:2007.03437* (2020).

[7] Jung Yeon Park, Ondrej Biza, Linfeng Zhao, et al. "Learning Symmetric Embeddings for Equivariant World Models". In: *arXiv preprint arXiv:2204.11371* (2022).

[8] Elise van der Pol, Thomas Kipf, Frans A Oliehoek, et al. "Plannable approximations to MDP homomorphisms: Equivariance under actions". In: *arXiv preprint arXiv:2002.11963* (2020).

[9] Giovanni Luca Marchetti, Gustaf Tegnér, Anastasiia Varava, et al. "Equivariant Representation Learning via Class-Pose Decomposition". In: *arXiv preprint arXiv:2207.03116* (2022).

[10] Manuel Watter, Jost Springenberg, Joschka Boedecker, et al. "Embed to control: A locally linear latent dynamics model for control from raw images". In: *Advances in neural information processing systems* 28 (2015).

[11] Xifeng Guo, En Zhu, Xinwang Liu, et al. "Affine Equivariant Autoencoder." In: *IJCAI*. 2019, pp. 2413–2419.

[12] Arnab Kumar Mondal, Vineet Jain, Kaleem Siddiqi, et al. "EqR: Equivariant Representations for Data-Efficient Reinforcement Learning". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 15908–15926.

[13] David G Lowe. "Object recognition from local scale-invariant features". In: *Proceedings of the seventh IEEE international conference on computer vision*. Vol. 2. Ieee. 1999, pp. 1150–1157.

[14] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features". In: *European conference on computer vision*. Springer. 2006, pp. 404–417.

[15] Tejas D Kulkarni, Ankush Gupta, Catalin Ionescu, et al. "Unsupervised learning of object keypoints for perception and control". In: *Advances in neural information processing systems* 32 (2019).

[16] Anand Gopalakrishnan, Sjoerd van Steenkiste, and Jürgen Schmidhuber. "Unsupervised object keypoint learning using local spatial predictability". In: *ICLR 2021* (2020).

[17] Bill Triggs, Philip F McLauchlan, Richard I Hartley, et al. “Bundle adjustment—a modern synthesis”. In: *International workshop on vision algorithms*. Springer. 1999, pp. 298–372.

[18] Johannes L Schonberger and Jan-Michael Frahm. “Structure-from-motion revisited”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4104–4113.

[19] Richard Held and Alan Hein. “Movement-produced stimulation in the development of visually guided behavior”. In: *Journal of Comparative and Physiological Psychology* 56.5 (1963), pp. 872–876.

[20] James J. Gibson. *The Senses Considered as Perceptual Systems*. Boston, MA: Houghton Mifflin, 1966.

[21] Alva Noë. *Action in Perception*. Cambridge, MA: MIT Press, 2004.

[22] Jeannette Bohg, Karol Hausman, Bharath Sankaran, et al. “Interactive Perception: Leveraging Action in Perception and Perception in Action”. In: *IEEE Transactions on Robotics* 33.6 (2017), pp. 1273–1291.

[23] Ruzena Bajcsy. “Active Perception”. In: *Proceedings of the IEEE* 76.8 (1988), pp. 966–1005.

[24] C.J. Tsikos and R.K. Bajcsy. “Segmentation via manipulation”. In: *IEEE Transactions on Robotics and Automation* 7.3 (1991), pp. 306–319.

[25] Jarmo Ilonen, Jeannette Bohg, and Ville Kyrki. “Three-dimensional object reconstruction of symmetric objects by fusing visual and tactile sensing”. In: *The International Journal of Robotics Research* 33.2 (2014), pp. 321–341.

[26] Marten Björkman, Yasemin Bekiroglu, Virgile Högman, et al. “Enhancing visual perception of shape through tactile glances”. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2013, pp. 3180–3186.

[27] David Schiebener, Jun Morimoto, Tamim Asfour, et al. “Integrating visual perception and manipulation for autonomous learning of object representations”. In: *Adaptive Behavior* 21.5 (2013), pp. 328–345.

[28] Ting Chen, Simon Kornblith, Mohammad Norouzi, et al. “A simple framework for contrastive learning of visual representations”. In: *International conference on machine learning*. PmLR. 2020, pp. 1597–1607.

[29] Nobuyuki Otsu. “A threshold selection method from gray-level histograms”. In: *IEEE transactions on systems, man, and cybernetics* 9.1 (1979), pp. 62–66.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[31] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).

[32] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. “Stochastic backpropagation and approximate inference in deep generative models”. In: *International conference on machine learning*. PMLR. 2014, pp. 1278–1286.

[33] John Schulman, Filip Wolski, Prafulla Dhariwal, et al. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

Paper C

Paper C

Goal-Conditioned Reinforcement Learning from Sub-Optimal Data on Metric Spaces

ALFREDO REICHLIN, MIGUEL VASCO, HANG YIN, DANICA KRAGIC

Preprint, 2025

Abstract:
We study the problem of learning optimal behavior from sub-optimal datasets for goal-conditioned offline reinforcement learning under sparse rewards, invertible actions and deterministic transitions. To mitigate the effects of *distribution shift*, we propose MetricRL, a method that combines metric learning for value function approximation with weighted imitation learning for policy estimation. MetricRL avoids conservative or behavior-cloning constraints, enabling effective learning even in severely sub-optimal regimes. We introduce distance monotonicity as a key property linking metric representations to optimality and design an objective that explicitly promotes it. Empirically, MetricRL consistently outperforms prior state-of-the-art goal-conditioned RL methods in recovering near-optimal behavior from sub-optimal offline data.

C.1 Introduction

Effective decision-making is an integral part of intelligent behavior. To achieve this, learning-based control methods have proven to be a viable option in complex scenarios [2, 3, 4, 5]. In particular, reinforcement learning (RL) allows learning near-optimal behavior through trial-and-error [6]. However, the online reinforcement learning framework generally requires slow, expensive (and potentially dangerous) online interactions with the environment.

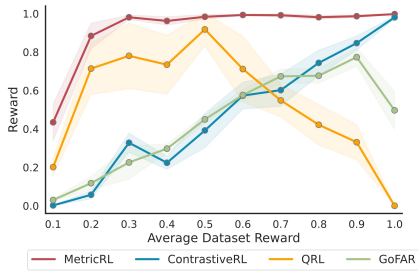


Figure C.1.1: Average reward on `Minigrid DoorKey` [1] as a function of the expected reward present in the offline dataset. We contribute MetricRL (red line), a novel goal-conditioned offline RL agent able to learn near-optimal behavior from severely sub-optimal datasets.

Offline RL, on the other hand, formalizes the learning of optimal behaviors from a *static* dataset [7]. This approach offers many advantages over its online counterpart, such as the ability to leverage large-scale datasets to learn complex behavior [8, 9] without the need of re-collecting data [10, 11]. Because of the inability to access the environment, in offline RL it is assumed that the dataset already includes suitable information to perform the given task. This data, however, may often be collected by sub-optimal agents. In this work, we address the question of how to learn different (and *better*) behaviors from the one observed in the dataset, regardless of its optimality. We focus on the challenge of *learning near-optimal behavior from severely sub-optimal datasets*, such as the ones collected by a random policy. We empirically demonstrate in Section C.4, and highlight in Figure C.1.1, how under these conditions current offline RL methods [12, 13, 14, 15] struggle significantly.

We focus on sparse-reward goal-conditioned offline reinforcement learning, which aims at learning optimal behavior to reach multiple goals within the same environment. Motivated by recent work in metric [16, 17] and quasimetric [12, 13] learning for RL, we study an approximation of the optimal value function in severely sub-optimal data conditions without explicitly relying on the Bellman operator, [18]. Paired with a weighted imitation learning actor, our approach avoids the out-of-distribution issue caused by the max operator used in dynamic programming solutions, without requiring any additional conservative (e.g., as in CQL [19]) or behavioral (e.g., as in BCQ [20], BEAR [21]) regularization. The core idea is to learn an embedding of the state space such that Euclidean distances correspond to the minimum number of actions needed to reach one state from the other. In particular, we show how symmetries can be exploited to ease the learning process in offline RL for many established environments. We introduce the notion of *distance monotonicity* (DM) as a relaxation of isometric embeddings and show that this property is sufficient to provably recover an optimal policy. Based on these insights, we propose *MetricRL*, a method to recover an optimal policy *regardless*

of the dataset quality.

We evaluate MetricRL across a wide range of literature-standard goal-conditioned reinforcement learning tasks. We show how MetricRL outperforms prior goal-conditioned offline reinforcement learning methods in learning near-optimal behavior from severely sub-optimal datasets. Additionally, we show how MetricRL easily scales to high-dimensional observations.

In summary, our contributions are the following:

- **MetricRL**: We propose a novel method that exploits symmetries in latent representation spaces for goal-conditioned offline reinforcement learning.
- **Distance Monotonicity**: We define a new property of these latent representation spaces. We show that, under invertible actions, preserving such a property provably leads to policy optimality.
- **Learning from Sub-Optimal Offline Data**: We demonstrate how MetricRL is able to recover optimal behaviors in severely sub-optimal data conditions, outperforming prior state-of-the-art offline RL methods across literature-standard environments.

C.2 Preliminaries and Assumptions

Goal-Reaching Reinforcement Learning: We consider standard Markov Decision Processes (MDPs) for goal-reaching tasks, $\mathcal{M} = (S, A, T, r, \gamma)$, where S is the state space, A is the action space, $T : S \times A \to S$ is a deterministic transition function (a common assumption in recent offline RL methods [14, 22, 13]), $r : S \times A \to \mathbb{R}$ is a goal-conditioned, sparse reward, i.e., $r(s, a) \neq 0$ iff $T(s, a) = s_g$ (where s_g is the goal-state), and $\gamma \in [0, 1)$ is a discount factor on the future rewards of the agent. We additionally define the goal states to be absorbing and consider the process terminated once these are reached.

The goal of the process is to find an optimal policy $\pi^*(a \mid s, s_g)$ that, given goal state s_g, maximizes the cumulative discounted reward of the agent for any possible starting state. To evaluate the optimality of the policy we resort to the goal-conditioned value function $V^\pi(s, s_g)$, defined as the expected discounted cumulative reward starting in a particular state and acting according to a policy π:

$$V^\pi(s, s_g) = \mathbb{E}_\pi \left[\sum_t \gamma^t r_t | s_0 = s, a = \pi(s, s_g) \right] \quad \forall s \in S. \tag{C.2.1}$$

The value function associated with the optimal policy is referred to as the optimal value function $V^* = V^{\pi^*}$ and, as such, is always greater or equal to any other value functions, i.e., $V^*(s, s_g) \geq V^\pi(s, s_g) \quad \forall s, \pi$.

Offline Reinforcement Learning: In the offline RL setting, we assume we have access to a dataset D of interactions with the environment described by the MDP and collected by some unknown policy π_β, i.e., $D_{\pi_\beta} = \{(s, a \sim \pi_\beta(a \mid s), r, s' = T(s, a))\}$. A

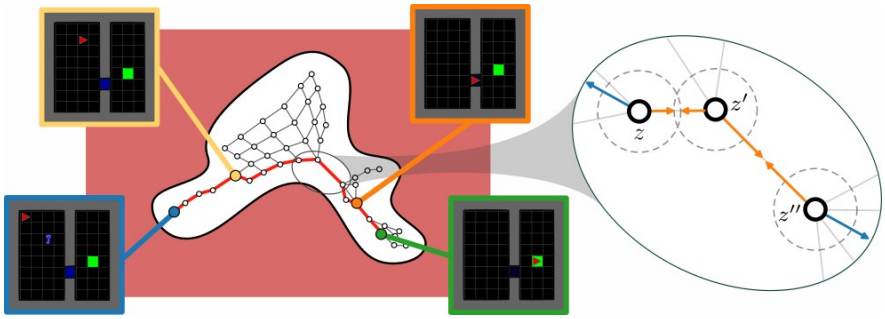


Figure C.3.1: We explore a form of symmetry in representation learning for goal-conditioned offline reinforcement learning: we learn a metric space in which Euclidean distances between the representation of states (z, z', z'') are related to the value function of the agent. We call our approach **MetricRL**. In the `Minigrid Doorkey` environment, moving greedily to adjacent states translates to the optimal policy (red line) to reach the goal (in green). Our objective is to preserve the local structure of adjacent representations (orange arrows, left) while maximizing the separation between non-adjacent ones (blue arrows, left).

major issue in Offline RL stems from the way dynamic programming methods estimate the optimal value function of the MDP. Most of the current algorithms rely on temporal difference techniques to learn the critic function, e.g., DQN [4], CQL [19], BEAR [21], BCQ [20]. However, the max operator used to estimate the target value has been shown to result in an overestimation of the expected return, [7]. While this is not an issue in Online RL, as the agent has the possibility of exploring overestimated states, it results in catastrophic effects in Offline RL. In this paper, we rely on an alternative technique for learning the critic. As discussed in prior work [13, 23], the dataset D implicitly defines a graph $G = (S, A)$ where the nodes correspond to the states and the edges to the actions of the agent. Here, we assume that this graph only has one connected component, i.e., any two states in the dataset are connected through a path on the graph. Furthermore, we assume that there always exists an *inverse* action, i.e., $\exists a' \in A : T(s', a') = s \quad \forall (s, a, s' = T(s, a))$. Note that a' doesn't necessarily need to correspond to the actual opposite action and it can be any viable action. By endowing the graph G with a metric d_S, we can define a corresponding finite metric space (G, d_S). For the tuple to be a valid metric space we need to define the metric $d_S : S \times S \to \mathbb{R}$ to respect the axioms of a metric space [24]. In particular, we can define the distance between any two states to be the number of edges on the shortest path connecting them (geodesic distance).

C.3 Method

In this work we focus on learning near-optimal goal-conditioned behavior from sub-optimal offline data. In these conditions, two main problems arise. The first is to learn an optimal behavior without depending on the quality of the distribution of the data used. The second is to avoid the out-of-distribution shift commonly faced in offline reinforcement learning [7]. We propose to address both using a metric learning approach to estimate the optimal value function. To do so, we start by defining *distance mono-*

tonicity (Section C.3.1), a novel property on representations needed to recast the problem of optimal value function estimation into metrics. We propose a loss function to learn maps that respect such a property and show how to build an approximation of the value function using distances in this learned representation (Section C.3.2). Finally, we define an actor-critic method to learn policies over this approximation and formally prove their optimality (Section C.3.3). We call our approach *MetricRL*.

C.3.1 Distance Monotonicity

Consider a continuous map between the state space of an MDP for goal-reaching tasks and a latent representation space: $\phi : S \to Z \subseteq \mathbb{R}^n$. We can equip this latent vector space with a Euclidean norm to obtain a Euclidean metric space $(Z, \|.\|_2)$. Here we consider distances in the original space, d_S as the minimum number of actions an optimal policy needs to reach one state from the other. We say ϕ is isometric if relative distances in the original state space d_S and the latent metric space d_Z are preserved, i.e., $d_Z(\phi(s), \phi(s')) = d_S(s, s'), \forall s, s' \in S$. In fact, if ϕ is isometric then the value function can be defined as simply the norm of the distance between the current state and the goal state, i.e., $V^*(s, s_g) = \gamma^{d_Z(\phi(s), \phi(s'))} r_g$ where r_g represents the reward at the goal and the expectation has been dropped as we assume deterministic transitions. We present an extended discussion of this observation in Appendix C.7.1.

However, estimating an isometry between these two metric spaces is known to be not always possible [25, 26]. To overcome such an issue, we consider a relaxation of isometries between metric spaces. Given two metric spaces (S, d_S) and (Z, d_Z) and the corresponding map ϕ between them, we can define the following property of ϕ:

Definition 4. *We say ϕ is* distance monotonic *(DM) if for all $s_1, s_2, s_3 \in S$, the following holds:*

$$d_S(s_1, s_3) < d_S(s_2, s_3) \implies d_Z(\phi(s_1), \phi(s_3)) < d_Z(\phi(s_2), \phi(s_3)).$$

We propose to parameterize the map ϕ_θ and learn it by minimizing the following objective on the dataset D:

$$\mathcal{L}_\theta(D) = \mathbb{E}_D \left[(\|\phi_\theta(s') - \phi_\theta(s)\|_2 - 1)^2 - \lambda \|\phi_\theta(s'') - \phi_\theta(s)\|_2 \right], \tag{C.3.1}$$

where (s, s') are any two states connected by an action sampled from D and s'' are other states sampled independently from the dataset at random.

Our loss function balances two requirements on the learned representation: the first term forces the representation to preserve the local distances of states in the graph, encouraging connected states to be separated by a vector of norm one in the latent representation Z. As we show in Figure C.3.1 (right, orange arrows), the representation of connected states (z, z') lies on a circumference of radius one. The second term of the loss maximizes the distance of non-directly connected states, as highlighted in Figure C.3.1 (right, blue arrows). This term of the loss is unbounded if the graph defined

by the dataset is not composed of a single connected component. As we discuss in Section C.3.4, in cases where the dataset defines multiple connected components (e.g., with image observations) we can always define a synthetic super-node to connect every termination state.

The minimization of Equation C.3.1 enforces the learning of a distance monotonic map: in Figure C.3.2 we highlight that the ratio of distance monotonic triplets significantly increases as a function of the training of the map. This can be approximated by discretizing the state space, building an ϵ-graph and the resulting distances in the state space as geodesics on the graph, comparing these distances with distances in the learned representation Z. In Appendix C.7.2 we provide a more formal intuition of this relationship as well as additional details on the evaluation of the distance monotonicity ratio.

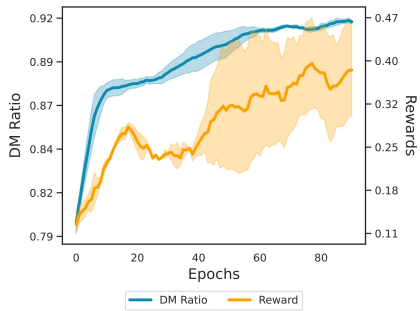


Figure C.3.2: Optimizing Equation C.3.1 increases the ratio of distance monotonic triplets (blue curve) on `Maze2D` (Large). Distance monotonicity is also correlated with an increase in the average return of the agent (orange curve).

C.3.2 Value Function Approximation

Distance monotonic representations (Definition 4) allow us to recast the original goal-conditioned RL problem as a distance one: similarly to the case of isometric maps, we can approximate the optimal value function using distances in the learned latent representation:

$$\tilde{V}(s) = \gamma^{d_Z(\phi_\theta(s), \phi_\theta(s_g))} r_g, \tag{C.3.2}$$

where s_g is the goal state and r_g is its associated sparse reward (only given at the goal state). This approximation would be identical to the true optimal value function only when the representation is an isometry of the graph.

However, distance monotonicity is enough to retain relative distances between states. Figure C.3.3 shows the estimated value function of different algorithms for a maze-like problem, using offline datasets collected with a random policy. In such problems the value function should retain the topology of the maze and estimate the value of each position in the maze based on the distance to the goal within the maze.

Figure C.3.3 (red) highlights that a distance monotonic representation (MetricRL) is the only value function able to correctly estimate the low value of the bottom left corner of the maze when the goal is on the other branch of the maze. Equations C.3.1 and C.3.2 allow us to abstract the estimate of the value function from the distribution of the policy that collected the dataset, as we highlight in Figure C.7.5 for policies of different values. Note that, for more complex topologies (Figure C.3.3, orange), distance monotonicity is not equivalent to isometries. However, as we show in the next subsection,

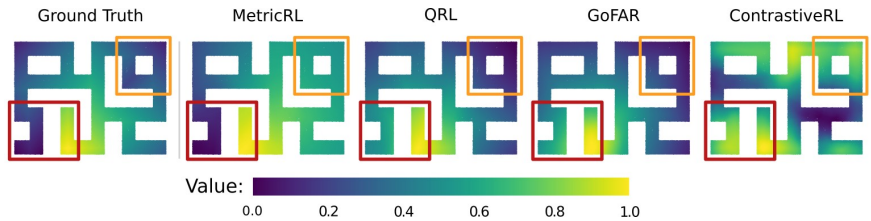


Figure C.3.3: Estimated value function for different methods using a dataset collected from a random policy in `Maze2D Large`. We highlight (in red) how MetricRL is the only method able to correctly assign low values to states that are close in the Euclidean space but not in terms of *distance to the goal*. Additionally, we highlight (in orange) that our proposed distance monotonicity in complex topologies is not equivalent to isometries, yet we are still able to recover provably optimal policies (as we discuss in Section C.3.3). All values are normalized.

our approximation still allows us to estimate a provably optimal policy.

C.3.3 MetricRL

Enforcing distance monotonicity in the latent representation allows us to learn an *approximation* of the true value function, not fully recover it. However, when distance monotonicity is preserved, a greedy policy built on this value function is optimal given some assumptions. Define the greedy policy according to the value function V as:

$$\pi_g^V(s) = \arg\max_a V(T(s,a)), \tag{C.3.3}$$

note that in general the argmax results in a set of possible actions. We have the following:

Theorem 5. *If the MDP is deterministic, sparse, and goal-conditioned, then*

$$\pi_g^{\tilde{V}}(s) \subseteq \pi_g^{V^*}(s), \quad \forall s \in S$$

holds if ϕ is distance monotonic.

Intuitively, Theorem 5 states that every policy greedy on the value function defined in Equation C.3.2 is optimal. A proof can be found in Appendix C.7.3.

To learn such policy we take a three-stage approach, which we call *MetricRL*: (i) we learn a distance monotonic map, using the loss function of Equation C.3.1; (ii) we define the value function using Equation C.3.2; (iii) we estimate a policy using the following loss function:

$$\mathcal{L}_\pi = \mathbb{E}_{s,a\sim D}\left[-(\tilde{V}(s') - \tilde{V}(s))\log(\pi(a \mid s))\right]. \tag{C.3.4}$$

This optimization procedure for the policy is akin to the family of weighted maximum likelihood [27] or weighted imitation learning [28]. A key difference is that the advantage is not exponentiated. Empirically, this didn't have particular effects on the learning of the policy and allowed the removal of the additional hyper-parameter controlling the temperature of the exponential, thus simplifying the training procedure.

Using a value function from a distance monotonic representation ensures the term $\tilde{V}(s') - \tilde{V}(s)$ is positive only for actions that bring the agent closer to the goal. This approach for the policy update is particularly suitable for offline RL: actions are sampled from the dataset which guarantees us to consider only in-distribution actions. Moreover, using the proposed distance monotonic representation instead of Temporal Difference methods for the critic solves the classic offline RL issue of out-of-distribution transitions [7]. We provide a pseudocode of our approach in Appendix C.7.5.

C.3.4 Practical Implementation

Stabilizing the loss In practice, we take the logarithm of the contrastive term (second term) in Equation C.3.1. We have found this formulation to stabilize the training, in particular for environments with longer episode horizons:

$$\mathcal{L}_\theta(D) = \mathbb{E}_D \left[(\|\phi_\theta(s') - \phi_\theta(s)\|_2 - 1)^2 - \lambda \log \|\phi_\theta(s'') - \phi_\theta(s)\|_2 \right]. \tag{C.3.5}$$

Learning with images When learning with images, often the goal information is present in the image, e.g., the position of the green square in the grid experiment in Figure C.3.4, which can break the connectivity assumption: two images with different goal positions are not connected by any path. This can be a problem in practice as the second term in Equation C.3.1 can grow indefinitely when comparing representations of images with different goals. However, in the case of finite MDPs, we can easily recover it by introducing an additional *super*-state that connects every termination state together. We present the implementation details of such a super-state in Appendix C.7.4. The introduction of this super-state and the consequent connection of the environments with different goals leads to a modification of the learned representation. Figure C.3.4 shows this learned representation. On the left is the distribution of the states for one single goal, while on the right is the distribution of all the goals connected by the super-state (red star). The solution to this representation is a radial distribution of the states connected in the middle by the super-state. All the states with the same goal (orange dots in the image) compose one ray of the overall distribution.

C.4 Results

We evaluate MetricRL against state-of-the-art baselines in offline reinforcement learning across multiple environments. In all experiments we consider three types of offline datasets: a *low* dataset, often collected using a random policy or an untrained agent; a *medium* dataset, collected using the policy of an online RL agent during training or adding stochasticity to a fully trained agent, and a *high* dataset, collected using the policy of a fully-trained online RL agent.

For baselines, we consider the following: CQL [19], BCQ [20], BEAR [21], PLAS [29], PLAS [29], IQL [15], ContrastiveRL [12], QRL [13], GoFAR [14], HIQL [22]. More details in Appendix C.7.6.

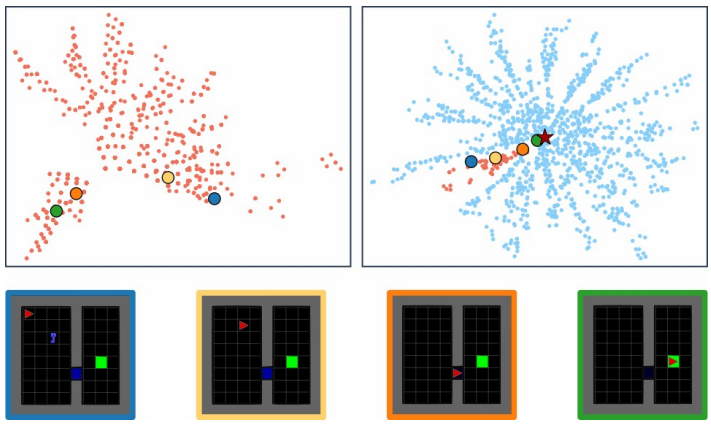


Figure C.3.4: Visualization of the two-dimensional latent space of MetricRL in the `DoorKey` environment when considering state features (left) and image (right) observations. We observe that the addition of a super-state (red star on the right figure) for image observations results in a significant change in the structure of the embedded graph as each set of states with a different (and visible) goal gets separated (orange dots). Nonetheless, in both cases, the optimal policy still follows a geodesic in the graph: from the starting state (blue) the agent needs to pick up the key (yellow) to open the locked door (orange) and move to the goal state (green).

For each model, we perform standard hyperparameter tuning or use the author's suggested hyperparameters (if available). The complete list of training hyperparameters is available in Appendix C.7.6, CQL, BCQ, BEAR, PLAS, IQL are implemented using [30]. For the remaining models, we use the author's provided code.

For environments, we consider:

- `Maze2D` [31]: a navigation task within a two-dimensional maze, with continuous actions and Newtonian physics. We consider three sizes of the maze: *u-maze*, medium and large. For each size, we use a uniform random policy to collect the low dataset, a policy with Ornstein-Uhlenbeck noise [32] to collect the medium dataset, and we use the Minari dataset provided in D4RL for the high dataset [31];
- `Reach` [33]: a manipulation task with a 7-DoF robot with continuous actions to reach a randomly-selected goal position in the workspace. To collect the datasets we employ a PPO agent trained on dense rewards along three different stages of training. For the low dataset, we use the policy of the randomly-initialized agent, for the medium dataset we use a PPO agent achieving half of the optimal expected reward, and for the high dataset we use the policy of the fully-converged agent;
- `Hypermaze`: a novel navigation task on a grid-like n-dimensional maze with discrete actions. To collect the offline datasets we employ a DQN agent trained online performing actions using an ϵ-greedy policy: for the low dataset we use purely uniformly random actions, for the medium dataset we sample random actions half of the time and optimal actions the other half, and for the high dataset we use the

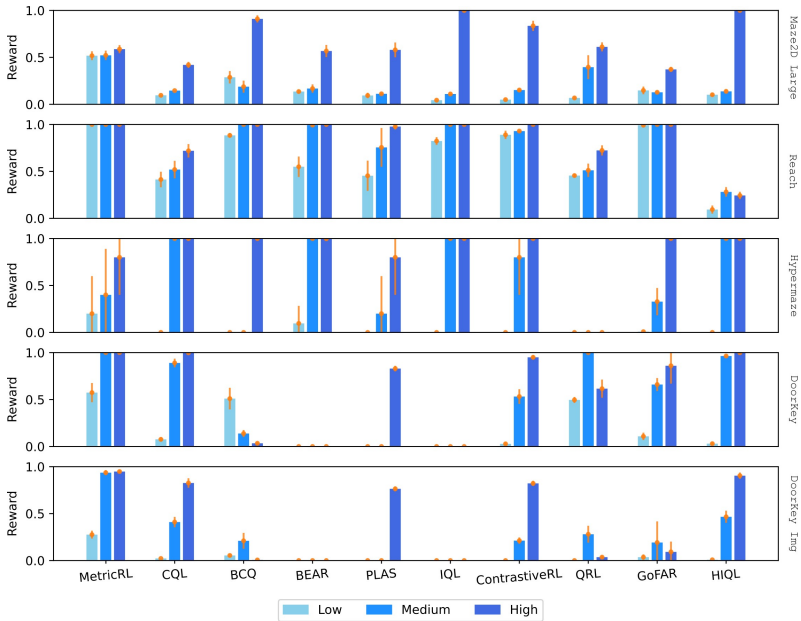


Figure C.4.1: Average reward returns on offline RL tasks with different types of datasets. All results are averaged over 5 randomly-selected seeds. Higher is better. We present extra results in Appendix C.7.9. MetricRL is the only model able to consistently learn near-optimal behavior from sub-optimal datasets (low and medium), outperforming the baselines, while performing on par with optimal datasets (high).

policy of the fully-trained agent;

- `Minigrid` [1]: a navigation task on a grid-like 2D room with discrete actions. We restrict the action space of the agent to navigation actions and remove rotations. To collect the offline datasets we employ the same strategy as above, that is ϵ-greedy on a fully trained DQN agent. We consider 2 tasks, `Minigrid Empty` consists of an open grid with external walls as the only obstacles. `Minigrid DoorKey` is a three-step task where the agent must first pick-up a key, then open a door, then reach a goal position;

We present our main results in Figure C.4.1[3] For a complete list of hyperparameters employed in the data collection procedure, please refer to Appendix C.7.6.

C.4.1 Discussion

MetricRL outperforms other methods in learning from sub-optimal datasets: The results highlight that MetricRL is the only model able to maintain a consis-

[3]We present in Appendix C.7.9 additional results. The conclusions remain the same for those additional environments.

tent level of performance, regardless of the type of dataset used for training. Additionally, MetricRL outperforms the other baseline methods when learning a policy from datasets collected using sub-optimal policies (*low* and *medium* datasets). In particular, for low datasets, MetricRL consistently outperforms all other baselines.

MetricRL provides more stable training across different data distributions in offline datasets: We conduct an additional experiment on the `Minigrid Door-Key` environment and consider a finer discretization of the distribution of rewards. We collect multiple datasets using an increasing value of ϵ for an ϵ-greedy optimal DQN agent. As shown in Figure C.1.1, MetricRL (red line) requires datasets with a smaller average reward than the baselines to achieve optimal performance. Additionally, the results show that MetricRL does not suffer from out-of-distribution data when the dataset has a very narrow distribution (almost all or all optimal trajectories).

MetricRL outperforms QRL in tasks with large action spaces: MetricRL is able to estimate good policies in both discrete action spaces and continuous ones. In particular, it manages to converge in very high-dimensional action spaces like the `Hy-permaze` where there are 81 possible actions with low datasets.

MetricRL can learn from sub-optimal datasets of images: To evaluate the performance of MetricRL when provided with high-dimensional observations (images) we reuse the MiniGrid `Empty` and `DoorKey` environments and introduce a *super*-state following the discussion in Section C.3.4. For every model, we introduce a CNN architecture that maps the images to a lower-dimensional representation. This highlights that MetricRL maintains its performance, and remains the only model able to learn optimal behavior from sub-optimal datasets. We present additional results in line with this on multiple environments, see Figure C.7.9 in Appendix.

In Appendix C.7.7 we explore the sample efficiency of MetricRL in scenarios with large state spaces. The results show that our method significantly outperforms temporal-difference (TD) methods (e.g., DQN [34]): to solve larger state space problems, we require a linear increase in the number of training iterations against the exponential increase of TD methods. In Appendix C.7.8, we show how MetricRL can be used in multi-goal tasks without modifications, considering changing multiple goals and discount factors at execution time.

C.4.2 Metric Space Ablation

Table C.4.1: Ablation on symmetrizing data in the `Maze2D Large` environment.

Dataset	QRL	QRL - sym	MetricRL
Low	0.13 ± 0.05	0.25 ± 0.04	0.51 ± 0.05
Medium	0.22 ± 0.03	0.52 ± 0.02	0.56 ± 0.05
High	0.28 ± 0.08	0.56 ± 0.12	0.60 ± 0.09

We further ablate the impact of the symmetry bias and the use of a Euclidean embedding for value function approximation. Specifically, we compare MetricRL against an equivalent model that employs a quasimetric critic, following [13]. For a fair comparison, we use the same actor optimization as in Equation C.3.4. We also evaluate a symmetrized variant of the quasimetric model by augmenting the dataset with synthetic reverse transitions, i.e., (s', a, s). Results on the `Maze2D`-large environment (Table C.4.1) show that incorporating symmetry can significantly enhance performance when it reflects a valid inductive bias. Moreover, Euclidean embeddings often prove easier to optimize, particularly in low-quality or sub-optimal data regimes.

C.4.3 Limitations

MetricRL recasts the computation of the value function as a problem of measuring distances in an appropriate learned metric space. To do so, it requires two additional assumptions on the MDPs it is applied to: the existence of inverse actions and the connectivity of the dataset used to learn the value function. As stated in Section C.3.4, the connectivity assumption can be solved using "super-states" to join the states into a unique connected component. The existence of inverse actions, on the other hand, defines a trade-off. It limits the applicability of the proposed method to MDPs where the assumption is respected. On the other hand, it greatly simplifies the learning process by imposing a relevant bias on the representation. MetricRL, in fact, requires significantly fewer data points as it can interpolate missing transitions when the inverse transition is present in the data. This allows to estimate effective policies even in severely sub-optimal data conditions.

C.5 Related Work

Offline RL: The challenge of learning a policy from a static sub-optimal dataset of transitions and rewards has been extensively studied [7]. The problem of exploration is not considered as it is assumed that the dataset given contains all the relevant information to estimate an optimal policy. Methods can be roughly divided into four main categories. The first one is constraining the learned policy to not deviate much from the policy that collected the dataset: [21] restrict policies to have the same support as the behavior policy rather than the policy itself; [29] implicitly constrain the policy by learning it using latent representations of actions with Gaussian prior (VAE), the constraint given by the KL divergence term; [35] learns the behavioral policy explicitly. The second category introduces a penalty in the reward function based on the uncertainty of the transitions or the reward function. This penalty has been defined as the uncertainty of the learned Q function, [36, 37], or a measure of pessimism (regularization of the highest Q) [19]. A third family of methods instead uses model-based RL and explicitly computes a model of the environment that can be used in different forms to regularize the learned policy [38, 39, 40, 20]. The last category includes *in-sample* algorithms which restrict the learning of the policy only on data within the provided dataset and reweighted by an estimate of the advantage function. This family of methods has been referred to as weighted supervised learning [28] or maximum likelihood [27]. It has

been extended with different forms of regularization including expectile regression for the critic [15], penalizing out-of-distribution actions in the critic [41], trust regions [42], goal relabeling [43] and Generative Adversarial Networks [44]. The policy estimate of our proposed method falls in this last category, as can be seen in Equation C.3.4. The use of Temporal Difference learning for the critic, however, subjects these methods to the problem of distribution shift in Offline RL. In this work, we propose a method that can estimate a high-return policy independently from the distribution of the data used even when the quality of the data decreases substantially.

Contrastive Learning: Representation learning techniques have been used to aid the RL problem. Several works have used contrastive learning models to speed up or improve the generalization of a classic RL algorithm, [45, 46, 47, 48]. Other applications include reward function estimation from demonstrations [49, 50] or generating intermediate goals for curriculum learning [51]. In [12, 52] contrastive learning is used to estimate the discounted state occupancy measure which is equivalent to the Q function in some particular cases. This method however works only when the reward function can be expressed as a goal reaching density and doesn't estimate the optimal Q value but rather the Q value of a current policy, thus still requiring the concurrent learning of a policy in a classic RL fashion. [53] proposes the use of contrastive learning to build a representation of states where distances are correlated with reachability in terms of actions. After the representation is learned, they propose to explicitly build the graph of the offline dataset and apply value iteration to get an estimate of the value function. The policy can be obtained by applying Dijkstra on the learned graph. More similar to this work [13, 23] estimate the optimal value function rather than the policy one. In [23] the authors use contrastive learning to find a representation where connections between states in action terms can be estimated in terms of Euclidean distances. The value function can then be recovered as the sum of the shortest path distances between the goal and the current state using classical depth-first search algorithms. [13] learns a map between pairs of states and a quasimetric representing the estimated optimal value function of a goal-conditioned MDP. This is done by setting the distance between two consecutive states to be equal to the reward between them and the distance between random pairs of states to be maximized. The optimal value function can then be defined as the distance between each state and the goal state. While being more general, quasimetrics cannot capture the appropriate bias induced by the inverse actions assumption. We show empirically that our proposed method is more effective in estimating a policy when data is severely sub-optimal. In this paper, we describe a property of representations needed to ensure the optimality of the critic function and propose a simple method to recover such a representation for a particular class of MDPs. Moreover, differently from the works above, we study the effectiveness of this methodology in handling offline datasets collected by policies that are not necessarily optimal.

Other metrics: Other measures of state similarity from a control perspective have been explored before. Bisimulation metric defines a measure of similarity between states in terms of future transitions and rewards, [54, 55]. These methods are theoretically grounded but particularly difficult to make them work in practice. This is espe-

cially true in the case of continuous spaces. Older work has explored different forms of value function approximation. By parametrically approximating the map between each state and its value, [56] approximates a notion of similarity (in a value sense) between states with an appropriate kernel and rewrite the Bellman operator as a function of this kernel. This still needs to solve the optimization problem with value iteration techniques. Proto-value functions, [57], instead express the transition and reward functions as linear matrices, the value function problem has exact solutions and can be estimated with an appropriate kernel method at the cost of expressivity.

C.6 Conclusions

In this paper, we proposed MetricRL, a novel approximation method for the optimal value function of sparse, deterministic, goal-conditioned MDPs. MetricRL relies on learning a *distance monotonic* representation of the state space, allowing it to define a value function that is correlated with the distance of each state to the goal. We have proved that, when the representation is indeed *distance monotonic*, a greedy policy on this approximated value function is optimal for the class of MDPs stated above. Experimentally, we have shown that MetricRL outperforms prior offline RL methods in learning near-optimal behavior from severely sub-optimal datasets. For future work, we plan on generalizing the notion of *distance monotonicity* to quasimetrics [58, 59], extending our method to stochastic MDPs and adapting it to online reinforcement learning problems.

C.7 Appendix

C.7.1 Sparse Goal-Conditioned Value Functions in Isometric Spaces

Recall the definition of the optimal value function described above as the maximum over the possible policies of the expectation of the discounted cumulative reward:

$$V^*(s, s_g) = \max_{\pi} \mathbb{E}_{\pi} \left[\sum_t \gamma^t r_t | s_0 = s, a = \pi(s, s_g) \right]. \tag{C.7.1}$$

In the particular case of a deterministic MDP with a sparse goal-conditioned reward function, the value function simplifies as there is only one state for which the reward is non-zero (the goal state). Moreover, in the optimal value function, the estimated discounted cumulative reward becomes equivalent to the reward at the goal discounted by $\gamma^{d_G(s,s_g)}$, where $d_G(s, s_g)$ is the geodesic. In this setting, the optimal value function estimation can be reduced to a shortest path estimation problem.

Assuming ϕ is an isometry, $d_G(s, s_g) = d_Z(\phi(s), \phi(s'))$ resulting in:

$$V^*(s, s_g) = \gamma^{d_Z(\phi(s),\phi(s'))} r_g. \tag{C.7.2}$$

C.7.2 Distance Monotonicity Measure

We provide a more formal intuition on the increase in distance monotonicity of a representation that minimizes Equation C.3.1. As such, consider the following modification of the objective:

$$\min_{\theta} \mathbb{E}_D \left[- \left\| \phi_\theta(s'') - \phi_\theta(s) \right\|_2 \right] \tag{C.7.3}$$

$$\text{subject to: } \left\| \phi_\theta(s') - \phi_\theta(s) \right\|_2 = 1. \tag{C.7.4}$$

This can be seen as the second term of the loss in Equation C.3.1 with the first term as an explicit constraint. As defined before, the distance between any two points, s_i, s_j, is the length of the geodesic on the graph defined as the offline dataset. This is equivalent to the sum of the intermediate steps within the geodesic path. Using the constraint in Equation C.7.2 we can use triangular inequality to bound the distance between points in the learned representation:

$$d_Z(z_i, z_j) = \left\| \phi_\theta(s_j) - \phi_\theta(s_i) \right\|_2 \in \left[0, d_S(s_i, s_j) \right]. \tag{C.7.5}$$

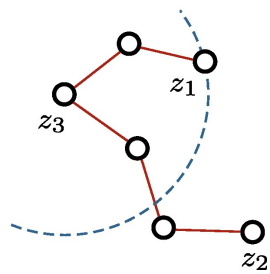


Figure C.7.1: The optimization in C.7.2 increases the distance between the representations. As long as z_2 is outside of the circumference, the triplet $s_{\{1,2,3\}}$ is distance monotonic.

The representation's ϕ_θ distance monotonicity can be measured as the ratio of triplets that respect the definition 4. That is, if $d_S(s_1, s_3) \leq d_S(s_2, s_3)$ then $d_Z(z_1, z_3) \leq d_Z(z_2, z_3)$, where $z = \phi(s)$. The distance monotonicity of ϕ increases for each triplet for which this becomes true. Graphically this can be seen in Figure C.7.1. As the distance $d_Z(z_1, z_3)$ is bounded, the distance monotonicity of ϕ increases when $d_Z(z_2, z_3) \in [d_Z(z_1, z_3), d_S(s_2, s_3)]$ (or equivalently z_2 goes outside the blue circumference in the figure). As such, distance monotonicity increases by stretching the distance between any two states. This is equivalent to the objective described in Equation C.7.2.

Experiment on Distance Monotonicity Measure

We tested quantitatively the effects of minimizing the loss defined in Equation C.3.1 and the increase in distance monotonicity of the learned representation. To do so we used the environment defined in `Maze2D` large with the high dataset. Figure C.3.2 (blue curve) shows the increase in distance monotonicity of a representation throughout the learning process. We computed the measure as follows:

- Before starting the training we compute a discretization of the positions of the maze and an adjacency matrix based on whether two positions are connected or not. This effectively defines a graph of the states of the maze.

- During every epoch of training we sample 1000 triplets of these nodes (s_1, s_2, s_3) at random and compute the distances on the graph between the two pairs using breath-first search, i.e., $d_G(s_1, s_3)$ and $d_G(s_2, s_3)$.
- Concurrently we map these three points in Z using the representation that's being learned and compute the distances using a Euclidean norm, i.e.:

$$d_Z(\phi(s_1), \phi(s_3)) \text{ and } d_Z(\phi(s_2), \phi(s_3)).$$

- We then compute the distance monotonicity (DM) ratio as the number of triplets among these 1000 for which if $d_G(s_1, s_3) < d_G(s_2, s_3)$ then:

$$d_Z(\phi(s_1), \phi(s_3)) < d_Z(\phi(s_2), \phi(s_3))$$

  or if $d_G(s_2, s_3) < d_G(s_1, s_3)$ then:

$$d_Z(\phi(s_2), \phi(s_3)) < d_Z(\phi(s_1), \phi(s_3)).$$

- The plot is also paired with the average reward a MetricRL agent achieves during training (orange curve).

More results on this are provided in Figure C.7.10.

C.7.3 Proof of Theorem 5

Proof. Here we prove that, assuming a deterministic, sparse, goal-conditioned MDP, every optimal solution defined as the greedy policy based on the value function defined as in Equation C.3.2 is contained in the greedy policy based on the optimal value function.

We start by rewriting the statement of the Theorem in the argmax form:

$$\pi_g^{\tilde{V}}(s) = \arg\max_a \tilde{V}(s' = T(s, a)), \tag{C.7.6}$$

$$\pi_g^{V^*}(s) = \arg\max_a V^*(s' = T(s, a)). \tag{C.7.7}$$

Given that the MDP is deterministic, the transition function $T(s, a)$ injectively maps (s, a) to a unique next state s'.

From the definition of $\tilde{V}$ and the Bellman equation, we have:

$$\tilde{V}(s') = \gamma^{d_Z(\phi(s'), \phi(s_g))} r_g, \tag{C.7.8}$$

$$V^*(s') = \max_{a'} \left[r(s', a') + \gamma V^*(s'') \right], \tag{C.7.9}$$

where $s'' = T(s', a')$. In a sparse, goal-conditioned MDP, the reward $r(s', a')$ is zero unless $s' = s_g$, $V^*(s')$ is equal to the discount factor raised to the power of the minimal number of actions needed to reach the goal and thus simplifies to:

$$V^*(s') = \gamma^{d_S(s', s_g)} r_g. \tag{C.7.10}$$

Substituting the value functions into the greedy policy definitions, we get:

$$\arg\max_a \gamma^{d_Z(\phi(s'),\phi(s_g))} r_g \subseteq \arg\max_a \gamma^{d_S(s',s_g)} r_g. \tag{C.7.11}$$

Since $\gamma < 1$ and r_g is constant, this reduces to:

$$\arg\min_a d_Z(\phi(s'),\phi(s_g)) \subseteq \arg\min_a d_S(s',s_g). \tag{C.7.12}$$

Assume, by contradiction, that there exists a state $\hat{s} = T(s,\hat{a})$ such that the action $\hat{a}$ is in $\pi_g^{\tilde{V}}(s)$ but not in $\pi_g^{V^*}(s)$. This implies:

$$d_Z(\phi(\hat{s}),\phi(s_g)) \leq d_Z(\phi(s'),\phi(s_g)), \quad \forall s' \in T(s,\cdot), \tag{C.7.13}$$

$$d_S(\hat{s},s_g) > d_S(\check{s},s_g), \quad \text{for some } \check{s} \in T(s,\cdot). \tag{C.7.14}$$

By the distance monotonicity of ϕ, $d_S(\hat{s},s_g) > d_S(\check{s},s_g)$ implies:

$$d_Z(\phi(\hat{s}),\phi(s_g)) > d_Z(\phi(\check{s}),\phi(s_g)),$$

which contradicts the assumption that $\hat{a} \in \pi_g^{\tilde{V}}(s)$.

Thus, the actions minimizing $d_Z(\phi(s'),\phi(s_g))$ are contained in the actions minimizing $d_S(s',s_g)$, implying:

$$\pi_g^{\tilde{V}}(s) \subseteq \pi_g^{V^*}(s), \quad \forall s \in S. \tag{C.7.15}$$

This completes the proof.

□

C.7.4 Incorporating Super-States in the Dataset

Here we provide a short description on how to add super-states in the dataset to connect it. The steps can be summarized as follows:

- **Define super-states:** These can be defined synthetically by creating a state that is not present in the dataset. In the experiments, we always defined it as a vector of all zeros of the same dimensionality of the state space.
- **Add transitions:** The offline dataset can then be augmented with synthetic transitions. For each trajectory in the dataset, we can append a new transition from the terminating state to the meta state. The action connecting these states is not relevant as it will not be used in the representation. In the experiments, we set the action value to a random (but valid) value.

C.7.5 Pseudocode

We provide the pseudocode of our approach below.

Algorithm 2 MetricRL.

Require: Initialize θ, ψ
Require: Offline dataset B, hyperparameters λ, η
repeat
  Sample batch $(s, a, s', s_g)_{\times B} \sim D$
  $s_r = \texttt{shuffle}(s)$ ▷ shuffle states in the batch
  $\mathcal{L}_\theta = (\|\phi_\theta(s) - \phi_\theta(s')\|_2 - 1)^2 - \lambda \log(\|\phi_\theta(s) - \phi_\theta(s_r)\|_2)$
  Update $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_\theta$
  $V = \|\phi_\theta(s) - \phi_\theta(s_g)\|_2$
  $V' = \|\phi_\theta(s') - \phi_\theta(s_g)\|_2$
  $\mathcal{L}_\pi = -(V' - V) \log(\pi_\psi(a|s))$
  Update $\psi \leftarrow \psi - \eta \nabla_\psi \mathcal{L}_\pi$
until convergence

C.7.6 Experimental Details

For each experiment, we provide the results for 5 runs with different seeds. Each model is trained for 100 epochs consisting of 500 batches of 256 data points each. Every model is trained using the Adam optimizer with a learning rate of 10^{-3}. All experiments have been conducted using an NVIDIA RTX 3080 GPU accelerator.

Both the policy and the value function are parameterized using a simple Multi-Layer Perception architecture consisting of 3 layers with 64 neurons each and a ReLU activation function. In the case of *MetricRL*, the policy outputs the mean of a Gaussian distribution with fixed variance when the actions are continuous and the logits of a Categorical distribution when the actions are discrete. When the observations are images we use a CNN architecture consisting of 4 layers with 64 filters each of size 3 by 3 to preprocess the images.

Models Description and Hyperparameters

MetricRL: The representation of the metric space has always dimension 128. The regularization term λ in Equation C.3.1 is 1 and the variance of the policy when actions are continuous is 1.

CQL [19], which introduces a conservative term in the estimation of the Q value. This term penalizes the highest values of the estimated Q function; hyperparameters: $\gamma = 0.95$, we use a conservative weight of 5.0.

BCQ [20], which perturbs the policy learned with a VAE with a DDPG term; hyperparameters: $\gamma = 0.95$, action flexibility: 0.5, we sample 10 actions per step and use 2 critics.

BEAR [21], that constrains the learned policy to the behavioral one estimated with BC; hyperparameters: $\gamma = 0.95$, we use an adaptive α with an initial value of 0.001 and a threshold of 0.05, we use 2 critics modules and sample 10 actions per step.

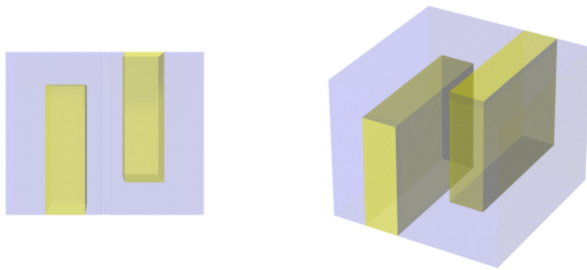


Figure C.7.2: Wall positions (yellow) in the HyperMazes in 2D and 3D, in blue are free cells.

PLAS [29], which trains the policy within the latent space of a conditional VAE trained on the Offline dataset; hyperparameters: $\gamma = 0.95$, we use 2 critics modules.

IQL [15], that avoids sampling out-of-distribution actions using a SARSA like critic update with quantile regression; hyperparameters: $\gamma = 0.95$, we use 2 critics modules and an expectile value of 0.9.

ContrastiveRL [12], which approximates the value function of the policy that collected the dataset using contrastive learning. To adapt it to offline RL, the objective is coupled with a behavioral cloning term. hyperparameters: $\gamma = 0.95$, an offline regularization of 0.05 and a fixed variance for the policy of 1.

QRL [13], that estimates a quasimetric to approximate the value function using a contrastive learning formulation. The model is paired with a learned policy regularized with a behavioral cloning term to avoid out-of-distribution state-actions pairs in offline RL conditions. hyperparameters: $\epsilon = 0.25$, initial $\lambda = 0.01$, offset softplus 500 and $\beta = 0.01$ and an offline regularization of 0.05.

GoFAR [14], that estimates an offline goal-conditioned RL policy by recasting it as a state occupancy matching problem. hyperparameters: $\gamma = 0.98$ and a discriminator gradient penalty of 0.01. For the f-divergence we use the χ^2-divergence.

HIQL [22], which reformulates IQL to an action-free hierarchical model. hyperparameters: $\gamma = 0.99$ $\beta = 1.0$, we use 2 critics modules and an expectile value of 0.7.

Evaluation Scenarios

`Maze2D`: The goal of the environment is to navigate an agent (actuated ball) to a target position inside a maze of varying complexity. The state space is 4-dimensional consisting of position and velocity in the plane, the action space is the torque applied to the ball in the two directions. The reward function is defined to be 1 when the agent reaches a position within 5 cm from the goal state and zero otherwise. Transitions take

into consideration the momentum of the ball, frictions in the environment and collisions with the walls. For the low dataset we collect a dataset of 1000 trajectories of the agent performing uniform random actions. For the medium dataset we collect actions according to an Ornstein-Uhlenbeck process with parameters: $\theta = 0.1$ and $\sigma = 0.2$. For the high dataset we rely on the Minari dataset provided by [31].

`Reach`: The task consists of moving the end-effector of a simulated 7-DoF arm to a desired position in 3D. The state space consists of positions and velocities in 3D of the end effector and gripper of the robot and the goal refers to the desired position of the end-effector and zero velocity. Actions are translations of the end-effector. We first train a PPO agent online to solve the task and collect datasets at different stages of the training to define different qualities.

`Hypermaze`: Defines a generalization of the classic Grid Maze navigation task. The environment consists of a hypercube of n dimensions of m cells per dimension where every cell can either be empty or wall. The agent occupies one cell at a time if it is empty and can translate to adjacent cells if they are not walls. The positions of the wall are initialized in an S-like shape similar to Figure C.7.2 when the hypermaze is defined in either 2 or 3 dimensions. The goal of the environment is to reach a goal placed on the other side of the maze.

For the results in Figure C.4.1 we fix the maze to be 4 dimensional with 20 cells per dimension. We collect the datasets by first training a DQN agent online to solve the task and then collect 3 datasets using an ϵ-greedy policy with ϵ respectively of values 0.9, 0.5 and 0.1.

For the sample complexity analysis our method is coupled with a learned transition function to recover the Q estimate. We vary the dimensions (from 2 to 5) and the number of cells (from 10 to 50). Here the state space is defined as the position of the agent in the maze discretized into cells plus whether there are obstacles or not in the adjacent cells. To train the agents, random states and actions are sampled from the environment. A reward of 1 is given only if the agent steps into the goal state at the end of the maze.

`Minigrid`: We experiment with two variations of the minigrid environment. The first is the `Empty` environment where an agent translates freely within a grid-like environment. The state is described by the 2D position of the agent and the actions are the 4 possible translation directions. The reward is 1 once a randomly selected cell is reached and 0 otherwise. The `DoorKey` environment introduces bottlenecks in the MDP. A wall is introduced in the center of the grid separating it into 2 rooms with a closed door in the middle. In the first room, a key is placed in a random position. The goal is to pick up the key, open the door and reach a goal cell in the other room. The state space is defined as the position in the grid of the agent plus the position of the key plus a binary value describing whether the door is open or not. Actions are translations in the grid plus a pick-up action that has an effect only when the agent is adjacent to the key plus an open door action that has an effect only when the agent has the key and is adjacent to the door. As before, The datasets are collected by training a DQN agent online to

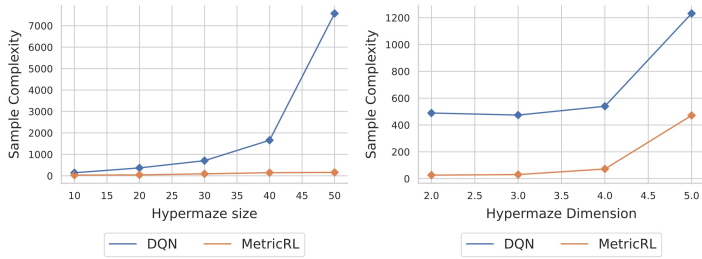


Figure C.7.3: Number of updates required to reliably solve the `Hypermaze` environment with varying number of cells (left plot) and dimensions (right plot).

convergence and collecting the datasets using ϵ-greedy strategy with the three different values for ϵ. For the high-dimensional observations case we use images rendered by the environment as the states. These are 80 by 80 pixels with 3 color channels.

C.7.7 Sample-Efficiency

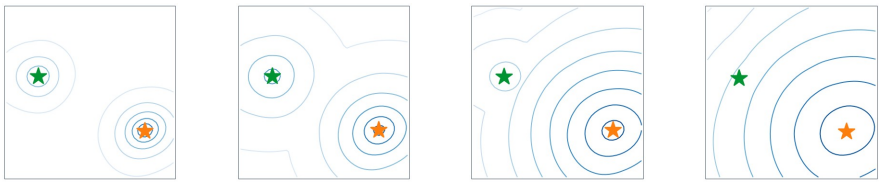
A main advantage of MetricRL stems from the nature of the loss function. Temporal difference methods (e.g., DQN) are known for their inefficiency when the time horizon grows considerably [60]. On the other hand, MetricRL mitigates this issue by using neural networks to learn a representation of a metric space.

To validate our hypothesis, we compare MetricRL against DQN [34] in the `Hypermaze` environment, considering a variable number of dimensions and cells. This allows us to control for both the dimensionality of the action space and the size of the state space of the underlying MDP.

In Figure C.7.3 we present the number of iterations required for the two methods to solve the maze at least 25 times consecutively during training as a function of the size of the maze (state space). The results show that for MetricRL the number of iterations required to learn to perform the maze grows linearly with the size of the maze. However, for DQN the number of iterations rises exponentially.

C.7.8 Multi-goal Tasks

Another advantage of the proposed representation used in Equation C.3.2 is that it does not depend explicitly on the goal. As long as goals are valid states within the training distribution, they can be arbitrarily used to estimate the value function. The explicit use of the discount factor allows for reshaping the value function to increase or decrease long-term reward delay. Moreover, with a slight modification, we can easily consider multiple competing goals. In this setting, the agent has to consider both the possible reward it can get in each goal state as well as its relative distance to each goal. As such, the discount factor of the MDP influences the optimal policy of the agent. For example when the agent has to navigate in a maze toward a door but there are multiple



(a) $\gamma = 0.9$ (b) $\gamma = 0.95$ (c) $\gamma = 0.99$ (d) $\gamma = 0.999$

Figure C.7.4: Visualization of the gradient of the value function learned by MetricRL in an environment containing two fixed goals with different rewards (r_1, r_2), as a function of the discount factor: the green star has $r_1 = 0.7$ and the orange star has $r_2 = 1.0$.

doors. In this case, the agent has to learn both how to navigate the maze as well as make the decision of which door to choose based on their reward and the length of the path.

With multiple rewards, the value function approximation can be reformulated by considering the maximum over the value function of all the possible goals (s_i, r_i):

$$\tilde{V}(s) = \max_i \{\gamma^{d_Z(\phi(s),\phi(s_i))} r_i\}. \tag{C.7.16}$$

C.7.9 Additional Results

In this section, we provide additional results on the experiments described.

Value Function Estimation in `Maze2D` Large: Following the discussion of Section C.3.2, we explore value function estimation in `Maze2D` Large for different types of datasets. The results of Figure C.7.5 highlight that MetricRL is the only method able to correctly estimate the low value of the bottom left corner of the maze when the goal is on the other branch of the maze, regardless of the quality of the policy used to collect the offline dataset.

`Maze2D`: We additionally provide the results for the u-maze and the medium maze described in [31], (Figures C.7.6 and C.7.7). Results confirm the findings described in the paper. MetricRL consistently outperforms the baselines in the case of low and medium datasets.

`Minigrid`: We provide results of the Empty environment with states and images as input, (Figures C.7.8 and C.7.9).

Below (Figure C.7.10) are additional results on the increase in distance monotonicity measure paired with an increase in the reward for different mazes and datasets of `Maze2D`.

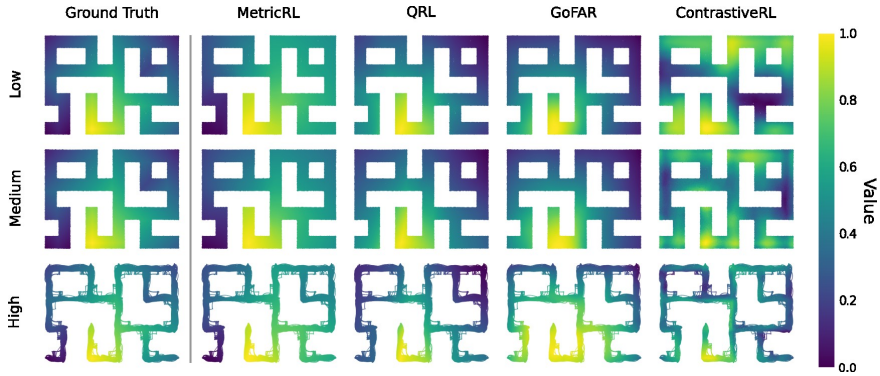


Figure C.7.5: Estimated value function for different methods using a dataset collected from different policies in `Maze2D Large`. All values are normalized.

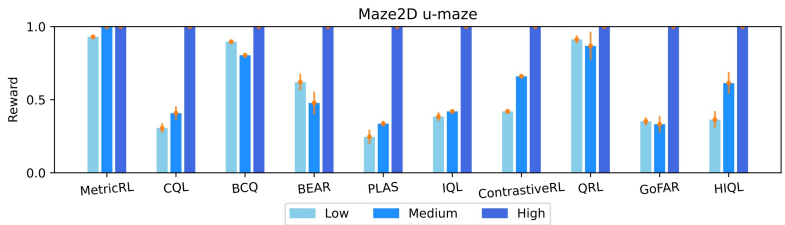


Figure C.7.6: Average reward returns on `Maze2D u-maze` with different types of datasets. All results are averaged over 5 randomly selected seeds. Higher is better.

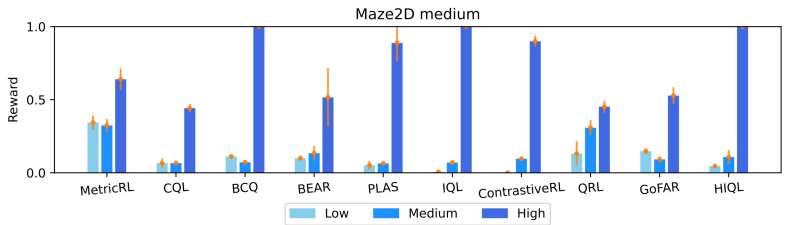


Figure C.7.7: Average reward returns on `Maze2D medium` maze with different types of datasets. All results are averaged over 5 randomly selected seeds. Higher is better.

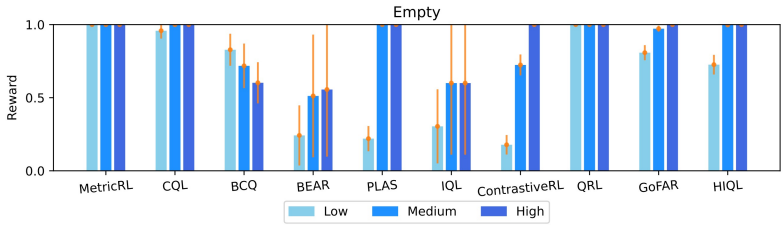


Figure C.7.8: Average reward returns on `Minigrid Empty` with different types of datasets. All results are averaged over 5 randomly selected seeds. Higher is better.

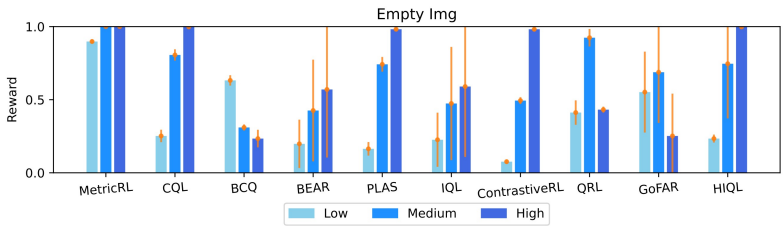


Figure C.7.9: Average reward returns on `Minigrid Empty` with images with different types of datasets. All results are averaged over 5 randomly selected seeds. Higher is better.

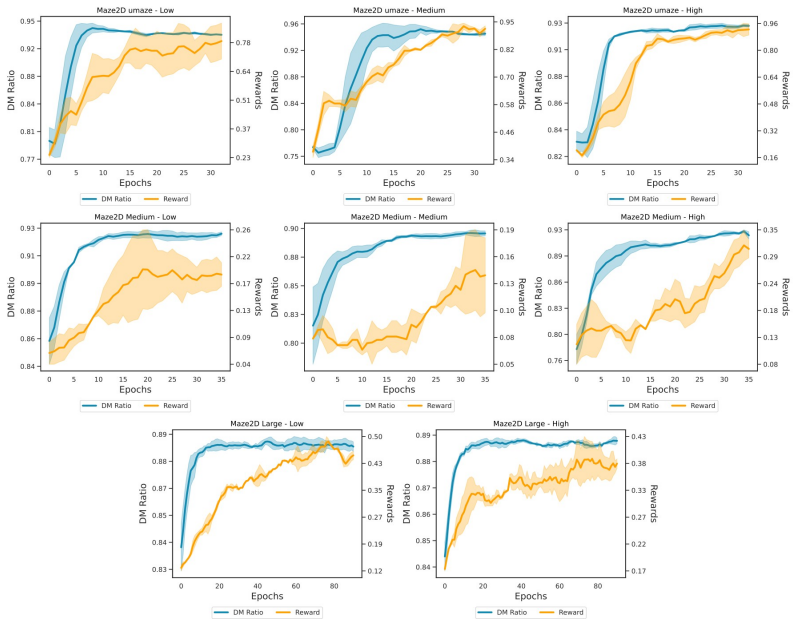


Figure C.7.10: Distance monotonicity ratio compared with average reward on `Maze2D` environments with different mazes and datasets.

References

[1] Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, et al. “Minigrid & Miniworld: Modular & Customizable Reinforcement Learning Environments for Goal-Oriented Tasks”. In: *CoRR* abs/2306.13831 (2023).

[2] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, et al. “Learning dexterous in-hand manipulation”. In: *The International Journal of Robotics Research* 39.1 (2020), pp. 3–20.

[3] David Silver, Thomas Hubert, Julian Schrittwieser, et al. “Mastering chess and shogi by self-play with a general reinforcement learning algorithm”. In: *arXiv preprint arXiv:1712.01815* (2017).

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. “Playing atari with deep reinforcement learning”. In: *arXiv preprint arXiv:1312.5602* (2013).

[5] Jan Peters and Stefan Schaal. “Reinforcement learning of motor skills with policy gradients”. In: *Neural networks* 21.4 (2008), pp. 682–697.

[6] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. 2nd ed. Cambridge, MA: MIT Press, 2018.

[7] Sergey Levine, Aviral Kumar, George Tucker, et al. “Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems”. In: *arXiv preprint arXiv:2005.01643* (2020).

[8] Homer Rich Walke, Kevin Black, Tony Z Zhao, et al. “Bridgedata v2: A dataset for robot learning at scale”. In: *Conference on Robot Learning*. PMLR. 2023, pp. 1723–1736.

[9] Sudeep Dasari, Frederik Ebert, Stephen Tian, et al. “RoboNet: Large-Scale Multi-Robot Learning”. In: *Conference on Robot Learning*. PMLR. 2020, pp. 885–897.

[10] Tianyu Shi, Dong Chen, Kaian Chen, et al. “Offline reinforcement learning for autonomous driving with safety and exploration enhancement”. In: *arXiv preprint arXiv:2110.07067* (2021).

[11] Nico Gürtler, Sebastian Blaes, Pavel Kolev, et al. “Benchmarking offline reinforcement learning on real-robot hardware”. In: *arXiv preprint arXiv:2307.15690* (2023).

[12] Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, et al. “Contrastive learning as goal-conditioned reinforcement learning”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 35603–35620.

[13] Tongzhou Wang, Antonio Torralba, Phillip Isola, et al. “Optimal Goal-Reaching Reinforcement Learning via Quasimetric Learning”. In: *arXiv preprint arXiv:2304.01203* (2023).

[14] Yecheng Jason Ma, Jason Yan, Dinesh Jayaraman, et al. “How Far I’ll Go: Offline Goal-Conditioned Reinforcement Learning vis f-Advantage Regression”. In: *Workshop on Learning from Diverse, Offline Data*. 2022.

[15] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. “Offline reinforcement learning with implicit q-learning”. In: *arXiv preprint arXiv:2110.06169* (2021).

[16] Seohong Park, Tobias Kreiman, and Sergey Levine. “Foundation policies with hilbert representations”. In: *arXiv preprint arXiv:2402.15567* (2024).

[17] Seohong Park, Oleh Rybkin, and Sergey Levine. “Metra: Scalable unsupervised rl with metric-aware abstraction”. In: *arXiv preprint arXiv:2310.08887* (2023).

[18] Richard Bellman. “The theory of dynamic programming”. In: *Bulletin of the American Mathematical Society* 60.6 (1954), pp. 503–515.

[19] Aviral Kumar, Aurick Zhou, George Tucker, et al. “Conservative q-learning for offline reinforcement learning”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 1179–1191.

[20] Scott Fujimoto, David Meger, and Doina Precup. “Off-policy deep reinforcement learning without exploration”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2052–2062.

[21] Aviral Kumar, Justin Fu, Matthew Soh, et al. “Stabilizing off-policy q-learning via bootstrapping error reduction”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[22] Seohong Park, Dibya Ghosh, Benjamin Eysenbach, et al. “HIQL: Offline Goal-Conditioned RL with Latent States as Actions”. In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023.

[23] Ge Yang, Amy Zhang, Ari Morcos, et al. “Plan2vec: Unsupervised representation learning by latent plans”. In: *Learning for Dynamics and Control*. PMLR. 2020, pp. 935–946.

[24] Dmitri Burago, Yuri Burago, Sergei Ivanov, et al. *A course in metric geometry*. Vol. 33. American Mathematical Society Providence, 2001.

[25] Jean Bourgain. “On Lipschitz embedding of finite metric spaces in Hilbert space”. In: *Israel Journal of Mathematics* 52 (1985), pp. 46–52.

[26] Jiří Matoušek. “Embedding finite metric spaces into normed spaces”. In: *Lectures on Discrete Geometry*. Springer, 2002, pp. 355–400.

[27] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, et al. “Awac: Accelerating online reinforcement learning with offline datasets”. In: *arXiv preprint arXiv:2006.09359* (2020).

[28] Qing Wang, Jiechao Xiong, Lei Han, et al. “Exponentially weighted imitation learning for batched historical data”. In: *Advances in Neural Information Processing Systems* 31 (2018).

[29] Wenxuan Zhou, Sujay Bajracharya, and David Held. “Plas: Latent action space for offline reinforcement learning”. In: *Conference on Robot Learning*. PMLR. 2021, pp. 1719–1735.

[30] Takuma Seno and Michita Imai. “d3rlpy: An Offline Deep Reinforcement Learning Library”. In: *Journal of Machine Learning Research* 23.315 (2022), pp. 1–20. url: http://jmlr.org/papers/v23/22-0017.html.

[31] Justin Fu, Aviral Kumar, Ofir Nachum, et al. “D4rl: Datasets for deep data-driven reinforcement learning”. In: *arXiv preprint arXiv:2004.07219* (2020).

[32] George E Uhlenbeck and Leonard S Ornstein. “On the theory of the Brownian motion”. In: *Physical review* 36.5 (1930), p. 823.

[33] Matthias Plappert, Marcin Andrychowicz, Alex Ray, et al. “Multi-goal reinforcement learning: Challenging robotics environments and request for research”. In: *arXiv preprint arXiv:1802.09464* (2018).

[34] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. “Human-level control through deep reinforcement learning”. In: *nature* 518.7540 (2015), pp. 529–533.

[35] Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, et al. “Keep doing what worked: Behavioral modelling priors for offline reinforcement learning”. In: *arXiv preprint arXiv:2002.08396* (2020).

[36] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, et al. "Morel: Model-based offline reinforcement learning". In: *Advances in neural information processing systems* 33 (2020), pp. 21810–21823.

[37] Tianhe Yu, Garrett Thomas, Lantao Yu, et al. "Mopo: Model-based offline policy optimization". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 14129–14142.

[38] Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, et al. "Deployment-efficient reinforcement learning via model-based offline optimization". In: *arXiv preprint arXiv:2006.03647* (2020).

[39] Tianhe Yu, Aviral Kumar, Rafael Rafailov, et al. "Combo: Conservative offline model-based policy optimization". In: *arXiv preprint arXiv:2102.08363* (2021).

[40] Marc Rigter, Bruno Lacerda, and Nick Hawes. "Rambo-rl: Robust adversarial model-based offline reinforcement learning". In: *Advances in neural information processing systems* 35 (2022), pp. 16082–16097.

[41] Yevgen Chebotar, Karol Hausman, Yao Lu, et al. "Actionable Models: Unsupervised Offline Reinforcement Learning of Robotic Skills". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 1518–1528.

[42] Yixiu Mao, Hongchang Zhang, Chen Chen, et al. "Supported trust region optimization for offline reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 23829–23851.

[43] Rui Yang, Yiming Lu, Wenzhe Li, et al. "Rethinking Goal-conditioned Supervised Learning and Its Connection to Offline RL". In: *International Conference on Learning Representations* (2022).

[44] Mianchu Wang, Rui Yang, Xi Chen, et al. "GOPlan: Goal-conditioned Offline Reinforcement Learning by Planning with Learned Models". In: *Transactions on Machine Learning Research* (2024).

[45] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. "Curl: Contrastive unsupervised representations for reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 5639–5650.

[46] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. "Representation learning with contrastive predictive coding". In: *arXiv preprint arXiv:1807.03748* (2018).

[47] Ankesh Anand, Evan Racah, Sherjil Ozair, et al. "Unsupervised state representation learning in atari". In: *Advances in neural information processing systems* 32 (2019).

[48] Adam Stooke, Kimin Lee, Pieter Abbeel, et al. "Decoupling representation learning from reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 9870–9879.

[49] Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, et al. "Vip: Towards universal visual reward and representation via value-implicit pre-training". In: *arXiv preprint arXiv:2210.00030* (2022).

[50] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, et al. "Time-contrastive networks: Self-supervised learning from video". In: *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2018, pp. 1134–1141.

[51] Srinivas Venkattaramanujam, Eric Crawford, Thang Doan, et al. "Self-supervised learning of distance functions for goal-conditioned reinforcement learning". In: *arXiv preprint arXiv:1907.02998* (2019).

[52] Kyle Beltran Hatch, Benjamin Eysenbach, Rafael Rafailov, et al. “Contrastive Example-Based Control”. In: *Learning for Dynamics and Control Conference*. PMLR. 2023, pp. 155–169.

[53] Deyao Zhu, Li Erran Li, and Mohamed Elhoseiny. “Value Memory Graph: A Graph-Structured World Model for Offline Reinforcement Learning”. In: *arXiv preprint arXiv:2206.04384* (2022).

[54] Norman Ferns and Doina Precup. “Bisimulation Metrics are Optimal Value Functions.” In: *UAI*. 2014, pp. 210–219.

[55] Pablo Samuel Castro. “Scalable methods for computing state similarity in deterministic markov decision processes”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 2020, pp. 10069–10076.

[56] Dirk Ormoneit and Śaunak Sen. “Kernel-based reinforcement learning”. In: *Machine learning* 49 (2002), pp. 161–178.

[57] Ronald Parr, Lihong Li, Gavin Taylor, et al. “An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning”. In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 752–759.

[58] Ishan Durugkar, Mauricio Tec, Scott Niekum, et al. “Adversarial intrinsic motivation for reinforcement learning”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 8622–8636.

[59] Ishan Durugkar. “Estimation and control of visitation distributions for reinforcement learning”. PhD thesis. The University of Texas at Austin, 2023.

[60] Andrew W Moore and Christopher G Atkeson. “Prioritized sweeping: Reinforcement learning with less data and less time”. In: *Machine learning* 13 (1993), pp. 103–130.

Paper D

Paper D

Geometry of Uncertainty: Learning Metric Spaces for Multimodal State Estimation in RL

ALFREDO REICHLIN, ADRIANO PACCIARELLI, MIGUEL VASCO AND DANICA KRAGIC

Preprint, 2025

Abstract:
Estimating the state of an environment from high-dimensional, noisy observations is a fundamental challenge in reinforcement learning (RL). Traditional approaches rely on probabilistic models to account for the uncertainty, but often require explicit noise assumptions, in turn limiting generalization. In this work, we propose a novel method to learn a structured latent representation, in which distances between states directly correlate with the minimum number of actions required to transition between them. The proposed metric space formulation provides a geometric interpretation of uncertainty without the need for explicit probabilistic modeling. To achieve this, we introduce a multimodal latent transition model and a sensor fusion mechanism based on inverse distance weighting, allowing for the adaptive integration of multiple sensor modalities without prior knowledge of noise distributions. We empirically validate the approach on a range of RL tasks, demonstrating improved robustness to sensor noise and superior state estimation compared to baseline methods. Our experiments show enhanced performance of an RL agent via the learned representation, eliminating the need of explicit noise augmentation. The presented results suggest that leveraging transition-aware metric spaces provides a principled and scalable solution for robust state estimation in sequential decision-making.

D.1 Introduction

Estimating the state of the environment from high-dimensional observations is a key challenge in Reinforcement Learning (RL). Compact low-dimensional representations of the sensory information have been shown to dramatically improve the performances of data-driven agents in synthetic [1] and real-world [2, 3] settings. In realistic scenarios, sensory information can be unreliable due to external noises or failures. In such cases, estimating the state of the system along with a measure of uncertainty allows for robust control, [4]. When having access to multiple sensors, state estimation can become more precise as these can compensate for one another. This, however, requires a more complex design of a multimodal robust state estimation model.

Optimal solutions can be found by applying a Bayesian formulation to the problem. Bayesian filtering techniques allow for the maximum a posteriori estimate of the state space even in case of uncertainties. These, however, require restrictions in terms of modeling the state distributions [5] or using expensive Monte Carlo approaches [6]. Moreover, they generally require an observation model and prior knowledge of the functional form of the uncertainty [7]. In this regard, deep learning solutions have shown promising results on the deterministic representation of high-dimensional observations [8] and complex dynamics [9]. Regarding uncertainty estimation, approaches included either training under noisy conditions by using variational methods [10] or using a reformulation of Gaussian processes [11]. Reliable state estimation under uncertainty remains, however, an open challenge.

We address the problem of state representation from multi-sensory observations for RL. We propose to learn a representation that aligns the sensory modalities and correlates temporal distances with Euclidean distances. The key idea of this work is to recast the problem of uncertainty estimation geometrically in a metric space. This, in turn, allows to significantly simplify the problem of state estimation, Figure D.1.1. The proposed model, Metric Learning for Multimodal State Estimation (MetricMM), consists of an encoder for each modality and a latent transition model trained with simple contrastive learning and prediction losses. We empirically show that the proposed representation can be used to train an RL agent on a variety of tasks. Moreover, we demonstrate how the proposed state estimate is robust to sensor noise of arbitrary nature without being trained on any of these.

Our work makes the following contributions:

1. **Metric-consistent multimodal state space.** We introduce MetricMM, which aligns sensory modalities in a shared latent space where temporal proximity corresponds to Euclidean distance, enabling simple geometric reasoning for control.
2. **Robustness under unseen corruptions.** We empirically demonstrate how MetricMM maintains high returns under seven corruption families (unseen during training), including settings where two of three modalities are corrupted, consistently outperforming fusion and representation baselines.

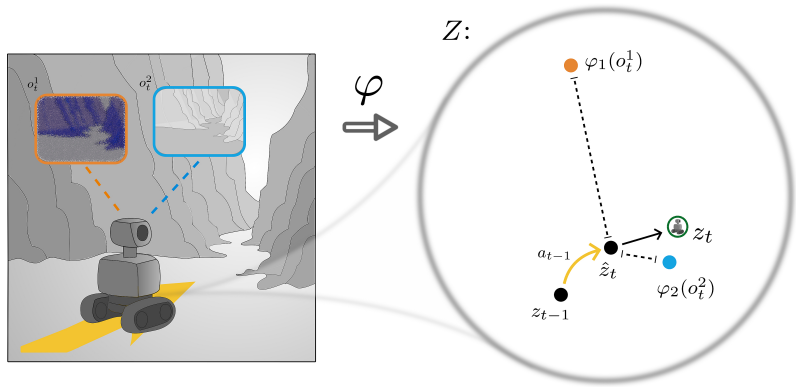


Figure D.1.1: We propose MetricMM, a novel state estimation model from noisy multimodal observations. Each observation (o_1, o_2 on the left) is mapped into a joint metric space (Z on the right) where distances from the latent dynamics prediction ($\hat{z}_t = \varphi_T(z_{t-1}, a_{t-1})$) are correlated with their uncertainty.

D.2 Related Work

Representation learning describes the problem of extracting a meaningful representation from high-dimensional observations [12]. This is of particular interest in reinforcement learning, where the dimensionality of the observations scales exponentially with the data needed for learning [13]. State representation learning, [14], refers to those representations that describe the underlying state of a sequential decision-making problem. Early work in this direction has focused on estimating a latent representation that is low dimensional and invariant to distractors by using a reconstruction loss [15, 16] or contrastive methods [17]. These approaches, however, lose the original structure of the state space and can't be generalized to other kinds of disturbances in the observation space.

In the case of reinforcement learning, data has a temporal structure due to the sequential nature of the problem. Model-based approaches make use of this structure to guide the learning of the representation and the policy. This can be done in the form of learning a transition model concurrently with the representation to either facilitate the learning of a policy [18, 19] or planning [20, 21, 22]. Moreover, this temporal bias has been used explicitly to guide the structure of the latent space of the learned representation to simplify the control problem [23, 24, 25]. While these methods use additional biases to simplify the learning problem or to recover additional properties, they do not offer a viable solution to the problem of noise robustness.

Bayesian filtering offers a solution to the problem of noise robustness in state estimation [26]. Classic methods of filtering offer provably optimal solutions in this direction but tend to be quite restrictive on the assumptions needed and the class of problem they can be applied to. Some of these assumptions have been successfully addressed by merging these models with modern versions of deep representation learning [10, 27].

These models, however, assume a noisy dataset to learn the uncertainty of the transition model and the observation model and restrict the model of the state estimate to be in a known form like a Gaussian distribution. Moreover, two major limitations are the need to learn a generative model for the update step and the inability to handle multimodal observations. In [28], they propose the use of a discriminative encoder for the observations to avoid learning the generative model. In [29], they overcome the multimodality problem by substituting the Kalman Gain with a Transformer's attention module on the learned encoding of the different modalities. Contrary to these methods, our proposed model doesn't need to estimate the uncertainty explicitly allowing us to be agnostic to the kind of noise. Moreover, multimodality is easily addressed by aligning the representation with an invariant loss.

A generalization of these methods is described with the term State Space Models (SSM) where transitions and observations as well as noise are relaxed to arbitrary functions [30]. Recurrent State Space Models (RSSM) learn explicitly the temporal relation of data using autoregressive models [31], or sequence-to-sequence models [32]. Uncertainty can be taken into account using a probabilistic version of this [33]. This has been done using variational methods and imposing a known form of the probability estimate in the latent space using Variational AutoEncoders (VAE) [34]. Alternative solutions to avoid the reconstruction loss include the use of Prototypes [35] or contrastive methods [36]. Similar to Bayesian filtering methods, uncertainty estimation remains a key challenge for these algorithms. Other forms of uncertainty estimation include Neural Processes [11, 37], and Energy functions [38]. These have, however, been applied only to supervised learning settings.

D.3 Background

Throughout the rest of the paper, we assume a Partially Observable Markov Decision Process (POMDP) defined by the tuple $(S, O^{1:N}, A, T, r, \gamma)$. Here, S denotes the true underlying Markovian state space of the environment, $O_{1:N}$ denotes the N available observation modalities, and A the action space. The transition function $T : S \times A \to S$ is assumed to be deterministic. The objective is to maximize the cumulative reward given by the function $r : S \times A \to \mathbb{R}$, scaled by the discount factor γ. Each observation o^i provides a potentially noisy measurement of the true state s. We assume that each observation is sampled from an unknown stochastic process, with independent noise affecting each modality, i.e. $o_t^i \sim p_i(o_t^i \mid s_t)$.

The overall goal of RL is to estimate a policy that maximizes the expected cumulative reward. When s is not directly accessible, the policy is generally conditioned on the available observations, i.e., $O^{1:N}$. Representation learning for RL can be defined as finding a suitable latent representation z for the sensory observations, such that the policy, $\pi : Z \to A$, achieves the maximum possible expected cumulative reward. That is, preserving the information necessary for optimal decision-making while discarding task-irrelevant noise. Stochasticity in the observations requires modeling the representation as an estimation process which can be formulated as a Bayesian filtering problem. Via the Markov assumption, we can write the estimation process recursively through

Bayes, i.e. $p(z_t \mid z_{t-1}, a_{t-1}, o_t^{1:N}) = p(o_t^{1:N} \mid z_t)p(z_t \mid z_{t-1}, a_{t-1})/p(o_t^{1:N})$. This is generally intractable when assumptions on the functional form of these probabilities cannot be made. Here, we simplify the estimation process and consider a deterministic representation of the state, as such we consider the maximum a-posteriori (MAP) estimate of the state, i.e. $z_t = \arg\max_{z_t} p(z_t \mid z_{t-1}, a_{t-1}, o_t^{1:N}) = \arg\max_{z_t} p(o_t^{1:N} \mid z_t)p(z_t \mid z_{t-1}, a_{t-1})$. In literature, the second term ($p(z_t \mid z_{t-1}, a_{t-1})$) is generally referred to as the prediction step while the first one ($p(o_t^{1:N} \mid z_t)$) as the update.

D.4 Method

We propose a novel approach to learn a latent representation of multi-sensory observations, along with a latent transition model, that enables robust state estimation independently of the nature of observation noise. Our method learns a metric space in which distances between latent states correlate with the minimum number of actions needed to transition between them in the environment. This provides a geometric interpretation of uncertainty, simplifying state estimation.

D.4.1 Latent State Representation

In an ideal scenario, having access to a perfect representation of the environment's dynamics would be sufficient to track the current state. However, in practice, approximation errors in the learned transition model and potential stochasticity in the POMDP dynamics must be accounted for. In sequential decision-making problems, these errors can compound over time, leading to divergence in the state estimate [39]. To mitigate this issue, we incorporate information from multiple sensor modalities to refine the state estimate at each step, akin to the Bayesian filtering formulation. However, traditional filtering approaches require explicit uncertainty modeling, which may not generalize well across varying noise conditions.

Instead, we propose to structure the latent space such that state transitions and sensory observations can be integrated without explicit probabilistic modeling. The key idea behind this work is that the uncertainty induced by transition model errors is generally *local* to its predictions, not in the raw observation space but in a space where distances are induced by the system's dynamics. Specifically, we argue that the transition model's uncertainty should be interpreted in a space where distances correspond to the minimum number of actions required to transition between states. This motivates the construction of a metric space that aligns with the dynamics of the environment.

To formalize this, we define a metric space $\mathcal{M} = (Z, \|\cdot\|_2)$ induced by an ideal injective map $\varphi : S \to Z$, where $Z \subseteq \mathbb{R}^m$ is a vector space, and distances are given by the Euclidean norm. We can define Z such that distances in this space are correlated with the minimum number of actions needed to transition between their corresponding environment states, i.e. temporal distances [25, 40, 41]. Injectivity between the two spaces allows us to use the formalism of MDP Homomorphisms [42, 43]. In the case of POMDPs, we do not have direct access to the state of the system and thus have to rely

on sensory observations and dynamics predictions. As such, we can define an approximation of the dynamics of the environment in this new latent space, i.e. $\varphi_T : Z \times A \to Z$. Given this structure, we make the following assumption:

Transition Error Assumption: The transition error at time t is constrained such that the predicted latent state $\varphi_T(z_{t-1}, a_{t-1})$ lies within a small ball of states that are close in terms of action-based distance. That is, there exists an ϵ-radius region in the latent space such that:

$$z_t \in \mathcal{B}(\varphi_T(z_{t-1}, a_{t-1}), \epsilon), \tag{D.4.1}$$

where $\mathcal{B}(z, \epsilon) = \{z' \in Z \mid d(z, z') \leq \epsilon\}$.

Intuitively, while the transition model may introduce small prediction errors, these errors generally remain within a neighborhood of states that require similar sequences of actions to transition between. Nevertheless, the learned transition model accumulates errors over time. This requires the integration of sensory information to correct the latent state estimate. Each modality O^i provides an independent estimate of the state. In this regard, we define a deterministic mapping from each observation space to the above-defined metric space Z:

$$\varphi_i : O^i \to Z, \quad \forall i \in [1, N] \tag{D.4.2}$$

where N represents the number of sensor modalities. We seek to learn these mappings such that the latent representation is aligned between them and respects the notion of metric space previously defined.

D.4.2 Sensor Fusion via Inverse Distance Weighting

The information from the sensors can be used to correct the prediction of the latent transition model on the state of the environment. However, due to noise, different modalities might be more or less reliable at any given time. We can make use of the geometry induced by the metric space to approximately model this uncertainty. Since we do not assume prior knowledge of noise distributions, we propose an adaptive weighting scheme based on the notion of distance induced by the metric space. The key intuitions are:

- If an observation's latent encoding $\varphi_i(o_t^i)$ is *close* to the latent transition prediction $\varphi_T(z_{t-1}, a_{t-1})$, then it is more likely to be an accurate state estimate.
- Conversely, if an observation encoding is *far* from the prediction, it is likely corrupted by noise or uninformative.

Thus, we weigh each modality's contribution using the inverse of its latent space distance to the transition model estimate. The final estimated state is:

$$z_t = \left(\sum_i \frac{1}{\|z_t^i - \hat{z}_t\|_2} \right)^{-1} \sum_i \frac{z_t^i}{\|z_t^i - \hat{z}_t\|_2}, \tag{D.4.3}$$

where $\hat{z}_t = \varphi_T(z_{t-1}, a_{t-1}))$ and $z_t^i = \varphi_i(o_t^i)$. This formulation follows a MAP principle, where more confident estimates (i.e., those that agree with the transition model) receive higher weights. This enables robustness without requiring explicit noise modeling.

D.4.3 Learning the Latent Representation

We introduce three loss functions to enforce the desired metric structure of the latent space. We do not require noisy observations during training as we do not need an explicit estimate of the uncertainty. Whenever we refer to $\bar{z}_t$ or $\bar{z}_{t+1}$ in the loss functions, we mean the average of the sensor encodings:

$$\bar{z}_t = \frac{1}{N} \sum_{i=1}^{N} \varphi_i(o_t^i), \tag{D.4.4}$$

Assuming no noise in the sensory observations, the mean is equivalent to the state estimate in Equation D.4.3.

Contrastive Temporal Distance Loss We enforce temporal consistency by structuring the latent space such that successive states remain close, while randomly sampled states are further apart:

$$\mathcal{L}_+ = \mathbb{E}[(\|\bar{z}_{t+1} - \bar{z}_t\|_2 - 1)^2] \tag{D.4.5}$$

$$\mathcal{L}_- = \mathbb{E}[-\log(\|\bar{z}_r - \bar{z}_t\|_2)] \tag{D.4.6}$$

where z_r is a randomly sampled state. A similar formulation for different applications was proposed in [40, 41]. The term $\mathcal{L}_-$ ensures that the latent representation does not collapse into a trivial solution where all states are mapped to the same point. By encouraging larger distances between random state pairs, we preserve meaningful geometry in the representation space. This term, however, is bounded by the triangular inequality given the positive term of the loss, i.e. $\mathcal{L}_+$.

Latent Transition Loss To ensure consistency between the learned transition model and observed transitions, we minimize:

$$\mathcal{L}_T = \mathbb{E}[(\varphi_T(\bar{z}_t, a_t) - \bar{z}_{t+1})^2] \tag{D.4.7}$$

This enforces that the transition model accurately captures environment dynamics.

Multimodal Invariance Loss To align representations across sensor modalities, we introduce an invariance loss:

$$\mathcal{L}_{inv} = \mathbb{E}[(\varphi_i(o_t^i) - \varphi_j(o_t^j))^2] \quad \forall i, j \in [1, N] \tag{D.4.8}$$

This ensures that each modality is mapped to the same learned latent metric space.

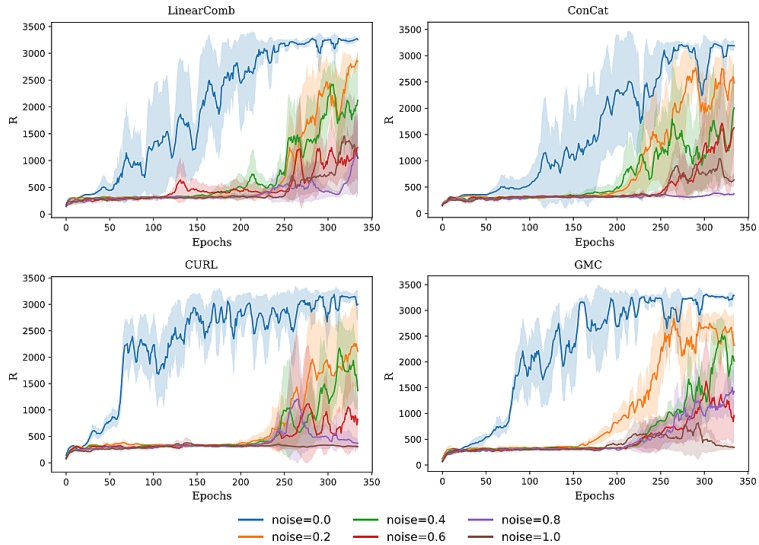


Figure D.5.1: Mean and standard deviation over 5 seeds of the training return for a SAC agent on the Hopper-v5 environment. The policies are trained with different state estimation modules (LinearComb, ConCat, CURL, GMC) and different amounts of Gaussian noise on the observations. With an increase in noise, the expected average return sensibly decreases for all the estimators. Epochs are in thousands.

D.4.4 Integration with Reinforcement Learning

By structuring the latent space around transition dynamics and sensor alignment, the representation provides a stable input for policy learning. This allows reinforcement learning agents to operate in a structured feature space where distances reflect control-relevant properties, reducing the complexity of decision-making. The proposed representation model can, thus, be integrated into standard reinforcement learning algorithms. Unlike traditional methods that require training with explicit noise augmentation, our approach naturally accounts for observation uncertainty. The final objective function is:

$$\mathcal{L} = \mathcal{L}_T + \lambda_1 \mathcal{L}_+ + \lambda_2 \mathcal{L}_- + \lambda_3 \mathcal{L}_{inv} \tag{D.4.9}$$

where $\lambda_1, \lambda_2, \lambda_3$ are hyperparameters.

D.5 Experiments

Scope and setup. We study whether multimodal representations can sustain control performance under severe observation perturbations and cross–modal mismatch. Our evaluation spans two suites. (i) `MuJoCo`: Hopper-v5, HalfCheetah-v5, Ant-v5, Walker2d-v5, Humanoid-v5, and InvertedPendulum-v5 with synchronized RGB and depth streams. (ii) `Fetch`: 7-DoF manipulation (FetchPickAndPlace-v4, FetchSlide-v4) with RGB, depth, and point clouds. Unless otherwise stated, we train Soft Actor–

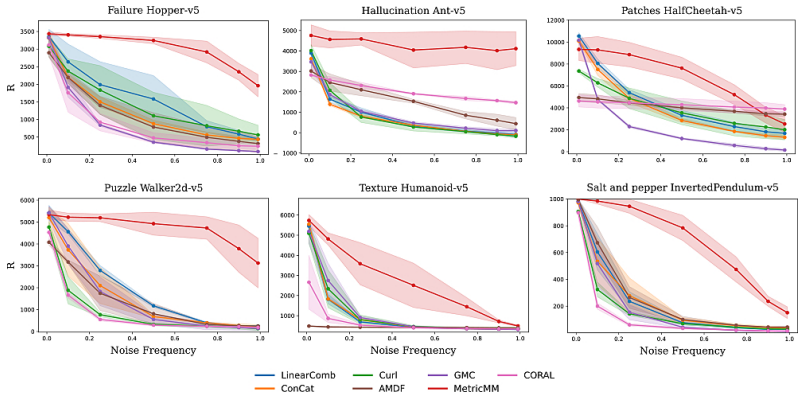


Figure D.5.2: Mean and standard deviation over 5 seeds and 50 trajectories of the testing return for a SAC agent on the Mujoco suite. The policies are trained with different state estimation modules and different amounts of noise (perturbations of one modality at a time). MetricMM is the only estimator that allows for a consistent return with high-frequency perturbations.

Table D.5.1: `Fetch` - Pick And Place — Patches noise — 2 noisy modalities.

Model	0.1	0.25	0.5	0.75	0.9	0.99
LinearComb	-0.89 ± 0.02	-1.99 ± 1.14	-1.76 ± 1.35	-2.57 ± 2.11	-2.67 ± 1.56	-2.17 ± 1.28
Concat	-0.04 ± 1.63	-1.13 ± 0.9	-2.84 ± 0.76	-2.06 ± 0.43	-2.53 ± 0.63	-2.43 ± 0.24
CURL	1.43 ± 0.32	-1.55 ± 0.93	-3.76 ± 1.94	-3.51 ± 0.59	-3.4 ± 0.46	-2.58 ± 0.38
GMC	-0.01 ± 1.16	-0.91 ± 0.27	-2.04 ± 0.38	-1.95 ± 0.67	-1.88 ± 0.89	-2.41 ± 1.24
AMDF	**2.2 ± 1.32**	0.93 ± 0.62	-1.04 ± 0.54	-1.75 ± 0.59	-2.16 ± 0.35	-2.34 ± 0.31
CORAL	-0.36 ± 0.43	-0.86 ± 0.23	-1.51 ± 0.99	-1.23 ± 0.64	-1.38 ± 0.86	**-1.47 ± 0.94**
MetricMM	1.91 ± 1.09	**1.87 ± 0.93**	**1.43 ± 1.14**	**0.92 ± 0.79**	**-0.91 ± 0.26**	**-1.47 ± 0.1**

Critic (SAC) end-to-end on top of the representation module and report mean return ± standard deviation over 5 seeds. All architectural and optimization details are deferred to the Appendix.

Corruptions and evaluation protocol. To probe robustness, we inject seven families of perturbations at *test* time (unseen during training): *Gaussian*, *Salt-and-Pepper*, *Patches*, *Puzzle*, *Texture*, *Failure*, and *Hallucination*. These are applied with different probabilities to study how fast performances deteriorate. For `MuJoCo`, we corrupt one modality at a time to isolate; for `Fetch` we additionally consider settings where two of the three modalities are simultaneously corrupted. For each (task, corruption) tuple, we evaluate 50 episodes per seed and aggregate across seeds to obtain performance–severity curves (Fig. D.5.2).

Baselines. We compare against six representative fusion/representation methods, matching encoder capacity and tuning budget:

Table D.5.2: Fetch - Slide — Failure noise — 2 noisy modalities.

Model	0.1	0.25	0.5	0.75	0.9	0.99
LinearComb	5.64 ± 0.92	3.31 ± 0.98	1.41 ± 2.14	-1.21 ± 1.74	-3.58 ± 1.17	-3.36 ± 0.35
ConCat	7.01 ± 1.64	5.5 ± 1.4	4.01 ± 0.48	0.06 ± 1.54	-1.61 ± 1.73	-1.49 ± 1.2
CURL	7.43 ± 0.43	5.34 ± 0.83	1.89 ± 2.35	-1.36 ± 3.33	-2.24 ± 1.87	-3.69 ± 3.21
GMC	5.17 ± 0.4	1.24 ± 3.33	0.12 ± 1.78	-1.77 ± 0.63	-2.72 ± 1.02	-3.11 ± 1.59
AMDF	3.05 ± 1.62	2.76 ± 1.61	-0.48 ± 1.0	-0.67 ± 1.93	-1.98 ± 1.65	-2.41 ± 2.25
CORAL	4.7 ± 2.92	2.28 ± 0.63	-1.16 ± 0.71	-1.39 ± 0.8	-1.87 ± 1.8	-1.79 ± 0.97
MetricMM	**9.26 ± 0.21**	**7.77 ± 2.02**	**5.95 ± 2.36**	**4.55 ± 2.96**	**1.33 ± 2.49**	**-0.23 ± 3.58**

- **Linear Combination** (LinearComb): a learned linear map that combines per-modality latent features.
- **Concatenation** (ConCat): feature concatenation followed by a shared projection.
- **CURL** [17]: contrastive learning on augmented views at each timestep, without explicit cross-modal fusion.
- **GMC** [44]: cross-modal contrastive alignment to learn a shared embedding.
- **α-MDF** [29]: an attention-weighted differentiable Bayesian filter that fuses modality-specific encoders into a latent state.
- **CORAL** [36]: joint latent space learned via per-modality reconstruction and temporal contrastive alignment.

All models are trained jointly with SAC under identical data budgets, replay settings, and evaluation schedules. Additional implementation details, architectures and hyperparameters are deferred to the Appendix.

Results on MuJoCo: robustness to single-modality corruption. Figure D.5.2 summarizes performance as the perturbation probability increases for six tasks and diverse corruptions. We observe three consistent trends. **(i) Robustness slope.** Our method (MetricMM) exhibits the flattest degradation, preserving a large fraction of the clean performance up to mid/high severities across tasks; in contrast, simple fusion baselines (LinearComb/Concat) degrade steeply even for low frequency of corruptions, indicating that naive aggregation does not resolve cross-modal disagreements. **(ii) Method ordering is stable across corruptions.** Across failure, hallucination, texture, puzzle, patches, and salt-and-pepper, MetricMM maintains the top curve while CURL and GMC are competitive at low frequencies but drop sharply once the corrupted modality dominates the fused signal. **(iii) High-DoF tasks are particularly sensitive.** On Humanoid-v5 and HalfCheetah-v5, the gap between MetricMM and the best baseline widens with an increased perturbation's frequency, suggesting that principled cross-modal consistency becomes increasingly important as control complexity grows.

Results on `Fetch`: robustness under multi-modality corruption. We next corrupt two of the three modalities on manipulation tasks. Tables D.5.1 and D.5.2 report returns for *patches* and *failure* corruptions, respectively, as the probability of corruption per time step increases. Two effects stand out. **(i) Majority corruptions.** Even when the majority of sensory channels are degraded, MetricMM retains substantially higher returns, e.g., on Fetch Slide with failure noise, it remains above zero reward until very high severities, while all baselines collapse much earlier (Table D.5.2). **(ii) Graceful decay vs. collapse.** On Fetch Pick-and-Place with patches, baseline returns turn negative quickly as the probability increases, while MetricMM degrades gradually and remains competitive at intermediate frequencies (Table D.5.1). These results indicate that MetricMM can prioritize and re-weight the remaining reliable modality when others fail.

Training with noisy observations is not required (and can be harmful). Figure D.5.1 shows learning curves on Hopper-v5 when injecting noise during *training* for four of the baselines. While moderate noise can induce invariances, it consistently slows exploration and lowers asymptotic returns across preprocessors; severe noise significantly delays the onset of learning. Practically, this requires knowing the corruption family a priori and increases computation, whereas MetricMM attains robustness without any noisy training.

Every modality provides useful information. Not all observations are equally easy to exploit, and without explicit cross-modal alignment a policy often latches onto the cheapest signal. In `Fetch`, the point-cloud stream is the most informative for precise geometry but also the most demanding to encode, making it especially vulnerable to perturbations. To quantify reliance, Table D.5.3 reports performance when we corrupt *only one* modality at a high probability (0.99). When point clouds are the sole corrupted stream, most baselines exhibit little to no degradation, revealing a systematic over-reliance on RGB/depth and under-utilization of 3D structure. In contrast, our aligned representation distributes credit across modalities and shows a more uniform sensitivity profile, indicating that each sensor contributes meaningfully to the learned state.

Takeaway. Across locomotion and manipulation, MetricMM maintains strong control performance under severe and diverse corruptions, including cases where two of three modalities are degraded. The method remains robust without noise-aware training, offering a practical way to resilient multimodal RL.

D.6 Conclusion

We introduced a novel approach for robust state estimation in reinforcement learning by learning a structured latent space where distances reflect the minimum number of actions required to transition between states. This metric space formulation enables a geometric interpretation of uncertainty, eliminating the need for explicit

Table D.5.3: `Fetch - Slide` — Failure noise — 1 noisy modality with probability 1. Noise applied to one type of modality only

Model	all modalities	image only	depth only	point cloud only
Linear Comb	-1.14 $\pm$ 2.41	-2.94 $\pm$ 1.26	-1.96 $\pm$ 2.14	9.27 $\pm$ 0.17
ConCat	1.53 $\pm$ 2.91	0.28 $\pm$ 6.19	-2.38 $\pm$ 0.67	7.37 $\pm$ 2.11
CURL	0.16 $\pm$ 3.09	-2.17 $\pm$ 1.51	-0.06 $\pm$ 8.26	8.15 $\pm$ 0.94
GMC	-1.76 $\pm$ 1.31	-1.13 $\pm$ 1.10	-4.52 $\pm$ 4.01	7.67 $\pm$ 0.53
AMDF	0.1 $\pm$ 1.53	-2.62 $\pm$ 2.87	-1.63 $\pm$ 2.91	3.80 $\pm$ 2.28
CORAL	-1.32 $\pm$ 0.67	-3.87 $\pm$ 4.63	-2.53 $\pm$ 1.15	7.56 $\pm$ 2.03
MetricMM	8.46 $\pm$ 0.92	7.29 $\pm$ 1.39	8.90 $\pm$ 0.96	7.87 $\pm$ 0.65

probabilistic noise modeling. Our method integrates multi-sensory observations via inverse distance weighting, ensuring adaptive sensor fusion without prior knowledge of noise distributions. Additionally, contrastive and transition-consistency losses enforce temporal structure, while an invariance loss aligns representations across modalities. We empirically demonstrated the effectiveness of our approach on a diverse set of tasks, showing that it improves state estimation robustness in the presence of sensor noise and significantly enhances RL agent performance. Results confirm that the learned representation generalizes across different environments without requiring explicit noise augmentation. By leveraging the environment's dynamics within the latent space, our approach provides a scalable and principled solution for robust state estimation. Future directions include adapting the method to non-stationary environments, evaluating adversarial robustness, and extending it to real-world robotics.

D.7 Appendix

D.7.1 Experimental Details

Here, we provide the necessary details on the conducted experiments. For the `MuJoCo` suite, we augmented the observations to include both RGB images and depth images. For both, we used a resolution of 84×84. To ensure Markovianity in the observations, we stack 3 consecutive frames together for both the RGB images and the depth images. We do not use proprioceptive information. We keep the reward function to be the default one of the different environments. For the `Fetch` suite, we augmented the observations to include RGB images, depth images and point clouds. Both RGB and depth images have a resolution of 128×128. We generate the point cloud observations by projecting rays from the camera position, given the depth of the environment. This results in a variable number of points, each consisting of a 3D position and an RGB value. Again, for all the modalities, we stack 3 consecutive frames together. For these environments, we changed the reward function to be a linear combination of the negative Euclidean distance between the end-effector of the arm and the object and the negative Euclidean distance between the object and the goal (the latter scaled by the

initial distance and multiplied by a factor of 10).

For each model and each environment, we learn a Soft Actor-Critic agent to maximize the reward of the environment. The list of relevant hyperparameters is described in Table D.7.1, these are the same for every experiment.

Table D.7.1: SAC hyperparameters.

Hyperparameter	Value
γ	0.99
τ	0.005
Learning rate actor	0.0003
Learning rate critic	0.001
Replay buffer size	100000
Batch size	256
Initial α	0.1
Number of critics	2
Number of simulations between updates	1
Number of workers	2
Target entropy	- dim of actions
Number of hidden layers critic	3
Number of neurons per layer critic	256
Non-linear activation critic	ReLU
Number of hidden layers actor	3
Number of neurons per layer actor	256
Non-linear activation actor	ReLU

For every experiment, we train the SAC agent together with its representation module end-to-end by minimizing the linear combination of all its losses. We train the agent until convergence, around 200 thousand steps for the `MuJoCo` environments and 400 thousand for the `Fetch` environments. After convergence, we test the best-performing models for 50 trajectories. Noise is applied at random to any modality at every step (except the initial observation) with a different probability.

For every state estimation method we use a Convolutional Neural Network (CNN) architecture for the RGB and the depth images and a SetTransformer for the point cloud. The CNN consists of 3 convolutional layers with 32 filters each and a kernel size and stride respectively of [4, 2], [4, 2], [3, 1] with a fully connected output layer. Each layer is followed by a ReLU non-linear activation function. We encode each point cloud frame with a compact SetTransformer: 6-D points (XYZ+RGB) are linearly projected to width $d_{\text{model}}{=}64$, processed by two self-attention blocks (4 heads, LayerNorm+residuals, feed-forward width $2\times$), and pooled via multihead attention with a single learned seed ($k{=}1$). Over a sequence of $T{=}3$ frames, the pooled 64-D summaries are concatenated

and mapped with a linear layer to a latent z of size d_z=64, then refined by a small MLP ($64 \to 64 \to d_z$) to produce the final embedding.

Below is the list of hyperparameters used for MetricMM and the baselines:

- MetricMM: latent size $d_z = 64$. Per-modality encoders (CNN for images, Set Transformer for point clouds). The loss hyperparameters are all 1, i.e., $\lambda_T = \lambda_1 = \lambda_2 = \lambda_{inv} = 1$.
- LinearComb: per-modality encoders (CNN/Set Transformer) projecting to d_z, learned linear fusion over modality latents, capacity matched to ours (same d_z, same encoder widths).
- ConCat: same per-modality encoders to d_z; fusion by feature concatenation (final latent dimension scales with number of modalities).
- CURL: per-modality encoders to d_z with a contrastive head, InfoNCE temperature $\tau = 0.2$, momentum target encoder with EMA m=0.99; standard cropping as image augmentations and gaussian noise as point cloud augmentations.
- GMC: per-modality encoders to a shared embedding with cross-modal contrastive alignment; InfoNCE temperature τ=0.3; same d_z and encoder widths as ours for fairness.
- AMDF: attention-weighted differentiable filter over modality encoders with recurrent latent state; latent size d_z and filter MLP widths matched to ours; attention temperature/weights from the implementation defaults.
- CORAL: per-modality encoders with joint latent space trained via reconstruction + contrastive objectives; InfoNCE temperature τ=0.1; same d_z and encoder capacity as ours.

D.7.2 Noises

For all the environments we experiment with the following families of corruptions as noise. Figure D.7.1 shows an example of the different noises for the `MuJoCo` Ant-v5 environment.

- Gaussian: linear combination with samples from a Gaussian distribution. For RGB and depth images: $\mathcal{N}(0, 25)$, for point clouds: $\mathcal{N}(0, 0.03)$. Can be applied to all the modalities.
- Salt-and-pepper: with probability 0.3 for `MuJoCo` and 0.8, each dimension of the observation is transformed to either the minimum or maximum possible value. Can be applied to all the modalities.
- Patches: a portion at random of the image is masked, 30% for `MuJoCo` observations and 50% for `Fetch`. Can be applied to RGB and depth images only.
- Puzzle: the image is divided into a 3 by 3 grid and reshuffled. Can be applied to RGB and depth images only.

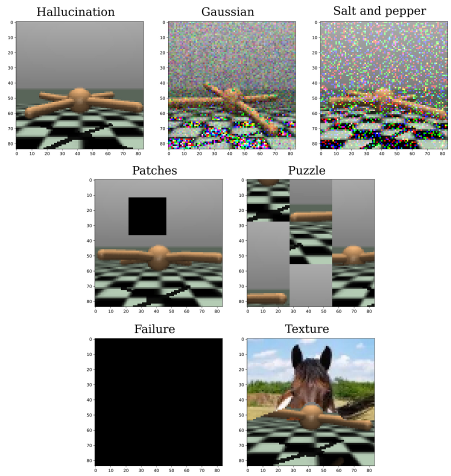


Figure D.7.1: Examples of noises on the RGB images for the MuJoCo Ant-v5 environment.

- Texture: the background of the image is segmented and replaced with another image. Can be applied to RGB images only.
- Failure: the entire observation is set to 0. Can be applied to all modalities.
- Hallucination: the entire observation is substituted with another in-distribution observation from another trajectory. Can be applied to all modalities.

D.7.3 Additional Mujoco Results

In Figures D.7.2 and D.7.3, we provide additional results on the MuJoCo environment. They illustrate the average reward and the standard deviation for every combination of noise for all the environments in the suite. Noise is applied to 1 modality at a time. Results are obtained from 50 evaluation trajectories after the models have fully trained.

D.7.4 Additional Fetch Results

Below, we provide additional results on the Fetch environment. The tables describe the average reward and the standard deviation for every combination of noise for the Pick and Place and the Slide environments. Noise is applied to either 1 or 2 modalities at a time. Results are obtained from 50 evaluation trajectories after the models have fully trained.

Table D.7.2: `Fetch` - Pick And Place — Failure noise — 1 noisy modality.

Model	0.1	0.25	0.5	0.75	0.9	0.99
Linear Comb	-0.82 ± 0.16	-1.2 ± 1.38	-1.65 ± 2.14	-1.53 ± 1.68	-1.68 ± 0.62	-1.37 ± 0.76
ConCat	0.8 ± 1.14	0.95 ± 1.59	0.44 ± 0.84	-0.49 ± 0.5	-0.17 ± 0.37	-0.92 ± 0.47
CURL	1.58 ± 1.0	1.08 ± 1.11	-1.56 ± 1.0	-2.08 ± 0.2	-2.29 ± 0.46	-2.56 ± 1.36
GMC	-0.01 ± 1.15	-1.34 ± 0.87	-1.51 ± 0.77	-1.62 ± 0.63	-1.63 ± 0.54	-1.6 ± 0.57
AMDF	**2.73 ± 1.11**	2.09 ± 1.08	1.02 ± 0.93	-0.3 ± 0.56	-0.83 ± 1.98	-1.52 ± 1.55
CORAL	-0.81 ± 0.14	-0.84 ± 0.12	-0.85 ± 0.1	-0.87 ± 0.26	-1.04 ± 0.38	-1.11 ± 0.56
MetricMM	2.31 ± 1.34	**2.35 ± 1.08**	**2.03 ± 0.68**	**2.35 ± 1.18**	**2.21 ± 1.2**	**1.89 ± 1.2**

Table D.7.3: `Fetch` - Pick And Place — Gaussian noise — 1 noisy modality.

Model	0.1	0.25	0.5	0.75	0.9	0.99
LinearComb	-0.84 ± 0.54	-1.02 ± 0.32	-0.78 ± 0.64	-1.37 ± 0.8	-1.42 ± 0.85	-1.36 ± 1.07
ConCat	1.36 ± 1.06	0.28 ± 0.36	-0.39 ± 0.92	-1.89 ± 0.44	-1.68 ± 1.11	-1.51 ± 0.66
CURL	**2.97 ± 1.32**	1.33 ± 1.23	0.13 ± 0.99	-0.75 ± 0.41	-1.11 ± 0.41	-1.43 ± 0.16
GMC	-0.24 ± 1.86	-0.15 ± 0.78	-0.55 ± 0.71	-1.42 ± 0.9	-1.94 ± 1.77	-0.85 ± 0.4
AMDF	2.57 ± 1.69	1.97 ± 0.94	1.07 ± 1.41	-0.04 ± 1.08	-0.24 ± 0.35	-0.56 ± 0.28
CORAL	-0.36 ± 0.59	-0.58 ± 0.23	-0.57 ± 0.24	-0.78 ± 0.13	-0.61 ± 0.19	-0.5 ± 0.38
MetricMM	2.29 ± 1.3	**2.52 ± 1.67**	**2.28 ± 0.87**	**1.97 ± 1.33**	**1.95 ± 1.17**	**2.19 ± 1.2**

Table D.7.4: `Fetch` - Pick And Place — Hallucination noise — 1 noisy modality.

Model	0.1	0.25	0.5	0.75	0.9	0.99
LinearComb	-0.95 ± 0.21	-1.44 ± 0.58	-1.28 ± 0.33	-1.57 ± 0.53	-1.82 ± 0.7	-1.93 ± 0.81
ConCat	-0.08 ± 0.93	-0.77 ± 0.47	-1.84 ± 0.99	-2.38 ± 0.46	-2.67 ± 0.6	-2.44 ± 1.27
CURL	2.26 ± 0.81	**2.44 ± 0.65**	0.69 ± 0.3	0.32 ± 0.26	-0.48 ± 0.84	-0.83 ± 0.48
GMC	0.28 ± 1.29	0.12 ± 0.81	-1.72 ± 1.39	-1.42 ± 1.17	-1.37 ± 0.82	-1.15 ± 0.64
AMDF	**2.43 ± 1.19**	1.03 ± 0.78	0.25 ± 0.34	-0.41 ± 0.7	-1.36 ± 0.55	-1.21 ± 0.16
CORAL	-0.43 ± 0.52	-0.58 ± 0.3	-0.59 ± 0.33	-1.02 ± 0.21	-1.24 ± 0.35	-0.96 ± 0.13
MetricMM	2.17 ± 0.73	1.99 ± 1.23	**1.85 ± 1.09**	**1.85 ± 0.96**	**2.11 ± 0.84**	**2.08 ± 1.05**

Table D.7.5: `Fetch` - Pick And Place — Patches noise — 1 noisy modality.

Model	0.1	0.25	0.5	0.75	0.9	0.99
LinearComb	-0.75 ± 0.57	-1.07 ± 0.87	-1.42 ± 1.13	-1.36 ± 0.94	-2.13 ± 0.97	-2.59 ± 1.68
ConCat	0.14 ± 0.18	-0.55 ± 0.29	-1.73 ± 1.0	-1.62 ± 0.64	-1.47 ± 1.11	-1.97 ± 0.58
CURL	1.52 ± 0.83	-0.56 ± 1.15	-2.43 ± 0.93	-2.8 ± 1.53	-3.14 ± 1.16	-2.67 ± 0.32
GMC	-0.1 ± 0.43	-1.78 ± 0.78	-1.8 ± 0.94	-1.37 ± 0.7	-1.92 ± 0.92	-1.28 ± 0.41
AMDF	**2.76 ± 1.44**	1.28 ± 1.03	0.56 ± 0.89	-1.3 ± 0.61	-1.09 ± 0.37	-1.98 ± 1.28
CORAL	-0.59 ± 0.19	-1.22 ± 0.73	-1.26 ± 0.77	-1.23 ± 0.75	-1.5 ± 1.18	-1.13 ± 0.62
MetricMM	2.33 ± 1.16	**2.13 ± 1.32**	**1.89 ± 1.21**	**2.22 ± 1.51**	**2.13 ± 1.39**	**2.23 ± 1.56**

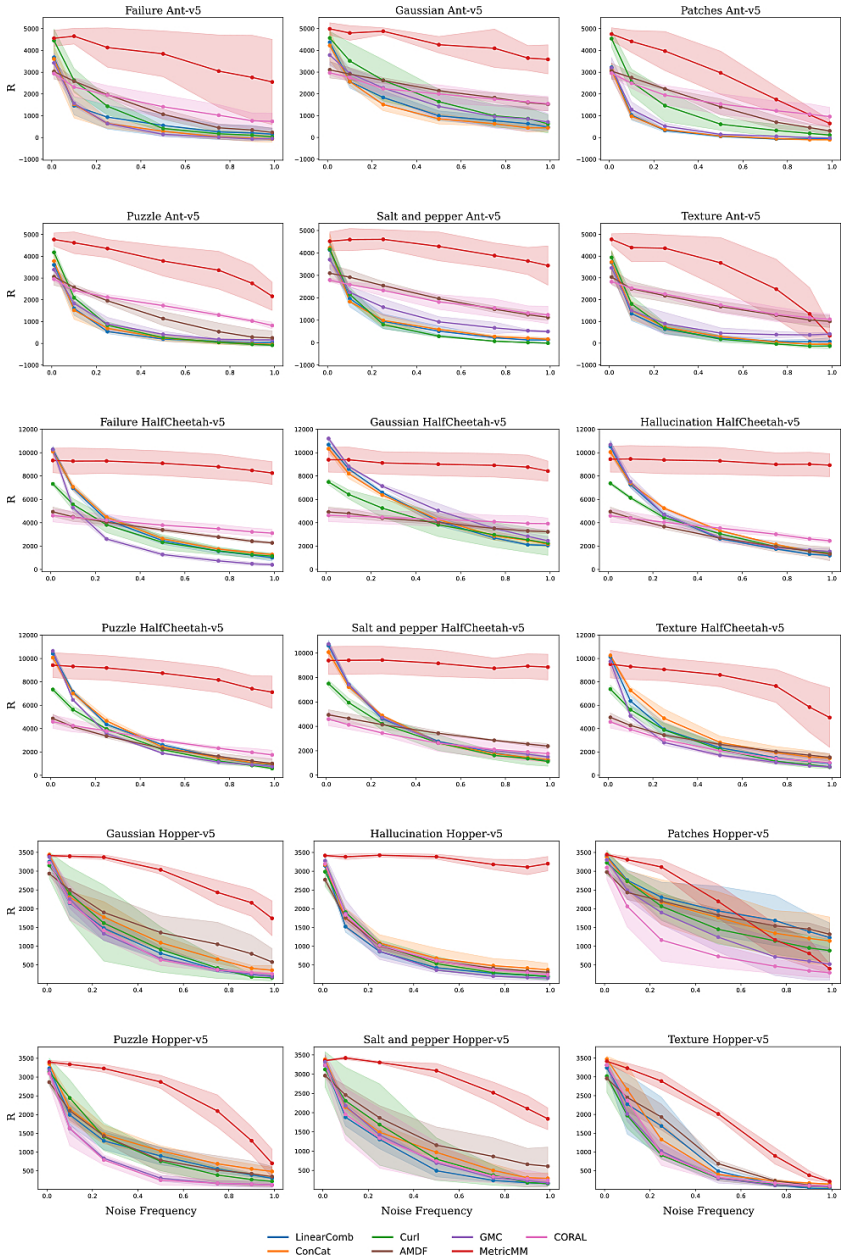


Figure D.7.2: Mean and standard deviation over 5 seeds and 50 trajectories of the testing return for a SAC agent on the Mujoco suite. The policies are trained with different state estimation modules and different amounts of noise (perturbations of one modality at a time).

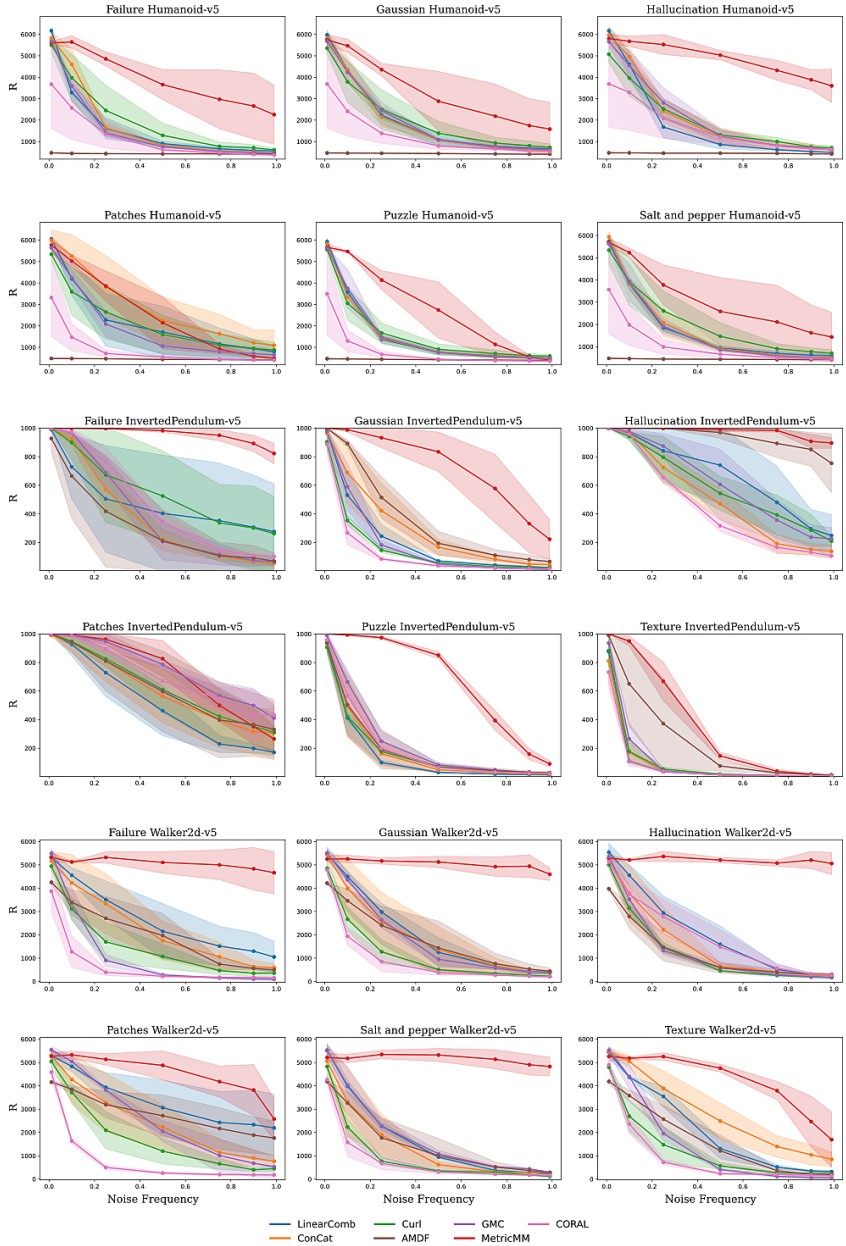


Figure D.7.3: Mean and standard deviation over 5 seeds and 50 trajectories of the testing return for a SAC agent on the Mujoco suite. The policies are trained with different state estimation modules and different amounts of noise (perturbations of one modality at a time).

Table D.7.6: `Fetch` - Pick And Place — Puzzle noise — 1 noisy modality.

Model	0.1	0.25	0.5	0.75	0.9	0.99
LinearComb	-0.29 ± 0.4	-0.5 ± 0.58	-1.39 ± 0.47	-1.23 ± 0.74	-1.03 ± 0.22	-1.66 ± 0.78
ConCat	0.64 ± 1.05	-0.25 ± 1.48	-0.97 ± 0.21	-1.18 ± 0.52	-2.62 ± 1.45	-1.71 ± 0.31
CURL	**2.79 ± 1.14**	**2.79 ± 1.7**	1.02 ± 0.19	-0.83 ± 0.47	-1.69 ± 0.63	-2.26 ± 0.32
GMC	-0.87 ± 1.02	-0.97 ± 0.55	-1.54 ± 0.61	-2.23 ± 1.3	-1.31 ± 0.42	-1.4 ± 0.49
AMDF	2.4 ± 1.67	1.69 ± 1.14	-0.67 ± 1.79	-2.02 ± 0.47	-2.61 ± 0.6	-3.89 ± 1.4
CORAL	-0.55 ± 0.32	-0.64 ± 0.11	-1.36 ± 0.93	-1.38 ± 0.65	-1.53 ± 0.99	-1.39 ± 0.62
MetricMM	2.24 ± 0.89	2.26 ± 0.7	**1.71 ± 1.27**	**2.37 ± 1.33**	**1.85 ± 1.4**	**1.98 ± 1.39**

Table D.7.7: `Fetch` - Pick And Place — Salt and pepper noise — 1 noisy modality.

Model	0.1	0.25	0.5	0.75	0.9	0.99
LinearComb	-0.33 ± 0.56	-0.68 ± 0.19	-0.68 ± 0.1	-0.83 ± 0.02	-1.08 ± 0.22	-0.72 ± 0.1
ConCat	0.56 ± 1.5	1.08 ± 0.95	0.62 ± 0.67	-0.76 ± 1.26	-0.79 ± 0.56	-0.89 ± 0.43
CURL	**3.56 ± 0.93**	**2.33 ± 0.88**	1.64 ± 0.85	0.49 ± 0.97	0.15 ± 1.29	-0.66 ± 0.85
GMC	-0.15 ± 0.96	-0.92 ± 1.3	-1.64 ± 1.06	-1.24 ± 0.98	-1.9 ± 1.1	-1.43 ± 0.57
AMDF	2.54 ± 1.07	1.83 ± 0.7	**1.68 ± 0.55**	-0.12 ± 0.23	0.01 ± 0.09	-0.83 ± 0.43
CORAL	-0.39 ± 0.48	-0.53 ± 0.3	-0.62 ± 0.16	-0.83 ± 0.1	-0.88 ± 0.12	-0.83 ± 0.1
MetricMM	2.43 ± 1.61	1.83 ± 1.22	1.67 ± 0.51	**2.21 ± 1.24**	**1.89 ± 0.58**	**2.26 ± 1.04**

Table D.7.8: `Fetch` - Pick And Place — Texture noise — 1 noisy modality.

Model	0.1	0.25	0.5	0.75	0.9	0.99
LinearComb	-0.22 ± 0.97	-0.63 ± 0.76	-0.42 ± 1.38	-0.55 ± 0.5	-0.36 ± 0.41	-0.02 ± 0.69
ConCat	1.56 ± 0.78	0.89 ± 0.96	0.71 ± 1.69	-0.14 ± 1.36	-0.62 ± 2.25	0.86 ± 1.91
CURL	2.39 ± 1.46	0.09 ± 1.02	-1.97 ± 0.38	-2.9 ± 0.35	-2.89 ± 0.2	-3.2 ± 0.31
GMC	-0.1 ± 0.71	-1.12 ± 0.86	-1.41 ± 1.45	-2.02 ± 1.33	-1.6 ± 1.01	-1.65 ± 0.97
AMDF	**3.21 ± 2.03**	**2.36 ± 1.66**	1.26 ± 1.34	0.1 ± 0.67	-0.79 ± 0.19	0.09 ± 0.22
CORAL	-0.35 ± 0.54	-0.67 ± 0.14	-0.99 ± 0.3	-1.8 ± 1.38	-1.54 ± 1.08	-1.09 ± 0.41
MetricMM	1.84 ± 0.94	2.1 ± 1.09	**2.52 ± 1.36**	**2.19 ± 1.01**	**2.27 ± 1.22**	**1.95 ± 1.31**

Table D.7.9: `Fetch` - Pick And Place — Failure noise — 2 noisy modalities.

Model	0.1	0.25	0.5	0.75	0.9	0.99
LinearComb	-1.32 ± 1.36	-1.54 ± 0.82	-1.36 ± 0.54	-2.28 ± 1.32	-2.04 ± 1.22	-2.37 ± 1.16
ConCat	0.17 ± 1.29	-0.25 ± 0.1	-1.17 ± 0.79	-2.37 ± 1.99	-2.1 ± 0.19	-2.78 ± 0.99
CURL	1.32 ± 0.43	-0.15 ± 1.5	-2.39 ± 0.83	-2.27 ± 0.77	-2.55 ± 0.65	-2.64 ± 0.72
GMC	0.05 ± 1.06	-1.64 ± 0.63	-1.68 ± 0.59	-2.18 ± 0.38	-2.15 ± 0.47	-2.28 ± 0.45
AMDF	**2.76 ± 1.39**	1.93 ± 1.72	-0.18 ± 1.06	-1.27 ± 0.87	-2.39 ± 1.21	**-1.77 ± 0.59**
CORAL	-0.6 ± 0.14	-1.09 ± 0.43	-1.1 ± 0.53	-1.34 ± 0.82	**-1.35 ± 0.81**	-1.83 ± 1.51
MetricMM	2.22 ± 1.74	**2.44 ± 1.48**	**1.65 ± 0.64**	**-0.47 ± 1.2**	-1.51 ± 1.25	-3.0 ± 2.27

Table D.7.10: `Fetch` - Pick And Place — Gaussian noise — 2 noisy modalities.

Model	0.1	0.25	0.5	0.75	0.9	0.99
LinearComb	-0.54 ± 0.67	-0.99 ± 0.54	-1.64 ± 0.8	-1.21 ± 0.31	-1.91 ± 0.78	-1.38 ± 0.36
ConCat	-0.05 ± 0.58	-1.05 ± 0.66	-1.96 ± 1.91	-2.63 ± 0.47	-3.09 ± 0.98	-3.27 ± 0.23
CURL	1.89 ± 0.74	0.29 ± 0.6	-1.5 ± 0.81	-1.94 ± 0.24	-2.11 ± 0.47	-2.04 ± 0.54
GMC	-0.1 ± 1.09	-0.88 ± 0.91	-1.59 ± 1.1	-1.57 ± 1.08	-1.22 ± 0.44	-1.4 ± 0.52
AMDF	**2.21 ± 1.56**	1.56 ± 1.24	-0.27 ± 0.45	-2.15 ± 0.69	-2.65 ± 0.96	-2.64 ± 0.5
CORAL	-0.48 ± 0.38	-0.59 ± 0.25	-0.92 ± 0.25	-0.81 ± 0.03	-1.1 ± 0.44	**-0.88 ± 0.12**
MetricMM	2.1 ± 1.49	**2.46 ± 1.23**	**1.76 ± 1.5**	**0.46 ± 0.63**	**-0.83 ± 0.39**	-2.4 ± 0.5

Table D.7.11: `Fetch` - Pick And Place — Hallucination noise — 2 noisy modalities.

Model	0.1	0.25	0.5	0.75	0.9	0.99
LinearComb	-0.24 ± 0.61	-1.4 ± 0.88	-1.73 ± 0.65	-1.48 ± 0.48	-1.92 ± 1.1	**-1.81 ± 0.74**
ConCat	-0.13 ± 0.7	-0.92 ± 1.23	-2.55 ± 0.79	-2.69 ± 0.62	-3.52 ± 0.62	-3.95 ± 0.83
CURL	2.19 ± 0.59	0.43 ± 0.66	-0.76 ± 0.36	-2.22 ± 0.1	-2.24 ± 0.24	-2.83 ± 0.32
GMC	-0.12 ± 0.58	-1.31 ± 0.68	-1.36 ± 0.73	-2.01 ± 1.02	-2.26 ± 1.15	-2.88 ± 1.63
AMDF	1.43 ± 0.91	-0.2 ± 0.53	-0.87 ± 0.69	-1.79 ± 0.38	-2.85 ± 0.23	-2.75 ± 0.23
CORAL	-0.6 ± 0.23	-0.92 ± 0.38	-1.32 ± 0.46	-1.46 ± 0.54	-1.72 ± 0.8	-2.25 ± 1.28
MetricMM	**2.36 ± 1.18**	**2.13 ± 1.33**	**1.86 ± 1.07**	**0.71 ± 0.46**	**-0.87 ± 0.54**	-3.46 ± 1.15

Table D.7.12: `Fetch` - Pick And Place — Puzzle noise — 2 noisy modalities.

Model	0.1	0.25	0.5	0.75	0.9	0.99
LinearComb	-0.77 ± 0.81	-1.09 ± 0.35	-1.07 ± 0.32	-1.97 ± 0.9	-1.74 ± 0.72	-1.63 ± 0.61
ConCat	0.09 ± 2.02	-1.83 ± 1.13	-2.94 ± 1.32	-2.6 ± 0.65	-1.92 ± 0.51	-2.01 ± 0.67
CURL	**2.61 ± 1.13**	1.34 ± 0.86	-1.41 ± 0.47	-1.8 ± 0.77	-1.77 ± 0.03	**-1.35 ± 0.0**
GMC	-1.15 ± 1.29	-1.27 ± 0.52	-2.43 ± 1.63	-1.37 ± 0.4	-1.53 ± 0.52	-2.22 ± 1.2
AMDF	1.94 ± 1.22	-0.04 ± 0.64	-1.9 ± 1.32	-2.4 ± 0.6	-2.38 ± 0.78	-2.31 ± 0.74
CORAL	-0.34 ± 0.56	-1.12 ± 0.51	-1.22 ± 0.52	-1.49 ± 0.67	-1.26 ± 0.3	-1.36 ± 0.39
MetricMM	1.94 ± 0.51	**2.18 ± 1.01**	**1.79 ± 1.64**	**0.38 ± 0.69**	**-1.05 ± 0.49**	-1.86 ± 0.71

Table D.7.13: `Fetch` - Pick And Place — Salt and pepper noise — 2 noisy modalities.

Model	0.1	0.25	0.5	0.75	0.9	0.99
LinearComb	-0.55 ± 0.37	-0.57 ± 0.2	-1.03 ± 0.36	-1.21 ± 0.46	-0.99 ± 0.2	-0.96 ± 0.16
ConCat	-0.02 ± 1.22	0.25 ± 1.43	-0.91 ± 0.99	-2.06 ± 1.24	-1.94 ± 0.51	-1.38 ± 1.37
CURL	**2.53 ± 0.66**	**2.08 ± 0.79**	0.12 ± 1.15	-0.7 ± 0.28	-1.17 ± 0.58	-1.26 ± 0.7
GMC	-0.45 ± 0.34	-1.34 ± 0.76	-1.34 ± 0.4	-1.27 ± 0.53	-1.42 ± 0.48	-1.72 ± 0.92
AMDF	2.22 ± 1.1	0.46 ± 0.21	0.17 ± 0.26	-0.73 ± 0.34	-1.31 ± 0.36	-1.38 ± 0.26
CORAL	-0.56 ± 0.26	-0.56 ± 0.19	-0.66 ± 0.08	-0.83 ± 0.1	-1.25 ± 0.61	**-0.83 ± 0.12**
MetricMM	2.24 ± 0.62	1.78 ± 1.07	**1.59 ± 0.88**	**0.27 ± 0.6**	**-0.8 ± 0.83**	-1.54 ± 0.26

Table D.7.14: `Fetch - Slide` — Failure noise — 1 noisy modality.

Model	0.1	0.25	0.5	0.75	0.9	0.99
LinearComb	6.36 ± 0.71	4.79 ± 2.13	4.58 ± 0.47	1.15 ± 1.23	0.5 ± 1.76	-1.14 ± 2.41
ConCat	6.5 ± 0.17	5.49 ± 0.12	3.49 ± 1.86	2.8 ± 0.96	2.48 ± 2.0	1.53 ± 2.91
CURL	6.95 ± 0.95	7.41 ± 0.82	5.68 ± 0.92	2.72 ± 1.97	1.27 ± 1.93	0.16 ± 3.09
GMC	5.65 ± 1.48	4.39 ± 1.92	1.6 ± 2.39	-0.42 ± 1.81	-1.53 ± 2.62	-1.76 ± 1.31
AMDF	4.03 ± 1.96	2.67 ± 1.64	2.1 ± 1.47	-0.21 ± 0.58	0.17 ± 0.51	0.1 ± 1.53
CORAL	4.97 ± 3.43	4.63 ± 2.64	0.79 ± 0.66	-0.35 ± 1.21	-1.47 ± 0.48	-1.32 ± 0.67
MetricMM	**8.98 ± 1.49**	**9.31 ± 0.84**	**9.12 ± 0.51**	**8.25 ± 1.1**	**7.92 ± 1.29**	**8.46 ± 0.92**

Table D.7.15: `Fetch - Slide` — Gaussian noise — 1 noisy modality.

Model	0.1	0.25	0.5	0.75	0.9	0.99
LinearComb	6.96 ± 1.16	6.55 ± 1.46	4.02 ± 0.4	3.64 ± 1.64	3.6 ± 1.24	2.17 ± 2.3
ConCat	6.33 ± 0.36	6.74 ± 1.48	4.79 ± 1.24	4.76 ± 1.49	4.41 ± 2.19	4.35 ± 2.38
CURL	9.11 ± 0.72	6.46 ± 0.86	4.92 ± 1.53	3.29 ± 0.45	2.57 ± 1.91	1.77 ± 0.86
GMC	6.88 ± 0.92	4.33 ± 0.25	2.41 ± 0.78	0.97 ± 0.71	-1.24 ± 1.5	-0.81 ± 0.99
AMDF	4.56 ± 2.47	2.7 ± 1.05	1.65 ± 0.82	0.32 ± 0.43	0.26 ± 1.18	-1.02 ± 0.69
CORAL	5.25 ± 3.29	4.04 ± 2.01	2.88 ± 2.27	1.59 ± 0.32	-0.16 ± 1.25	-0.24 ± 0.74
MetricMM	**9.34 ± 0.61**	**9.54 ± 1.26**	**9.7 ± 0.38**	**8.14 ± 1.14**	**8.99 ± 1.35**	**8.28 ± 1.01**

Table D.7.16: `Fetch - Slide` — Hallucination noise — 1 noisy modality.

Model	0.1	0.25	0.5	0.75	0.9	0.99
LinearComb	6.76 ± 0.67	6.27 ± 0.91	5.78 ± 0.72	5.62 ± 0.71	4.21 ± 0.44	4.16 ± 1.37
ConCat	7.14 ± 0.05	6.61 ± 1.72	5.57 ± 1.23	5.21 ± 1.12	4.04 ± 0.99	2.57 ± 1.11
CURL	8.96 ± 0.8	7.99 ± 1.5	7.58 ± 0.9	5.96 ± 0.64	4.87 ± 0.87	4.72 ± 0.75
GMC	7.07 ± 1.01	6.53 ± 0.72	3.9 ± 1.0	2.95 ± 0.82	3.53 ± 0.83	2.71 ± 0.09
AMDF	3.62 ± 1.73	3.75 ± 1.69	4.06 ± 1.92	2.28 ± 1.17	2.25 ± 0.99	1.62 ± 1.18
CORAL	5.72 ± 3.19	5.13 ± 2.82	4.02 ± 3.01	3.54 ± 2.7	2.62 ± 1.68	2.28 ± 2.19
MetricMM	**9.2 ± 1.15**	**8.31 ± 0.56**	**8.58 ± 0.57**	**9.27 ± 0.88**	**9.18 ± 0.45**	**8.48 ± 0.34**

Table D.7.17: `Fetch - Slide` — Patches noise — 1 noisy modality.

Model	0.1	0.25	0.5	0.75	0.9	0.99
LinearComb	5.56 ± 0.6	4.16 ± 0.98	2.81 ± 1.12	1.96 ± 2.34	0.35 ± 1.86	-0.36 ± 1.16
ConCat	7.5 ± 0.94	4.7 ± 0.56	3.54 ± 0.9	2.75 ± 1.69	1.3 ± 2.3	1.3 ± 2.44
CURL	8.42 ± 1.34	5.95 ± 1.43	4.53 ± 1.76	1.41 ± 4.01	-0.01 ± 3.13	-0.68 ± 4.36
GMC	5.39 ± 0.65	3.49 ± 1.84	-0.25 ± 2.09	-2.62 ± 2.24	-1.57 ± 1.11	-3.09 ± 1.44
AMDF	3.82 ± 2.01	3.29 ± 1.49	0.87 ± 0.41	-0.06 ± 1.41	-0.62 ± 1.0	-0.4 ± 1.7
CORAL	3.94 ± 2.84	2.08 ± 1.11	-0.88 ± 0.42	-0.92 ± 1.4	-2.23 ± 1.46	-1.85 ± 0.41
MetricMM	**8.75 ± 0.7**	**9.19 ± 1.06**	**9.37 ± 0.95**	**8.13 ± 0.09**	**7.26 ± 1.01**	**8.51 ± 0.79**

Table D.7.18: `Fetch` - Slide — Puzzle noise — 1 noisy modality.

Model	0.1	0.25	0.5	0.75	0.9	0.99
LinearComb	6.83 ± 1.01	5.47 ± 0.87	4.65 ± 1.06	0.7 ± 0.69	-0.18 ± 0.6	-1.47 ± 1.56
ConCat	6.35 ± 1.87	5.95 ± 1.73	4.38 ± 1.99	2.26 ± 2.15	1.56 ± 2.42	-0.05 ± 2.35
CURL	8.44 ± 0.25	6.84 ± 0.53	5.42 ± 1.47	3.55 ± 2.58	1.13 ± 2.33	0.59 ± 3.19
GMC	5.55 ± 1.65	4.7 ± 1.86	1.29 ± 1.77	-1.75 ± 1.02	-3.12 ± 1.66	-2.94 ± 1.66
AMDF	3.88 ± 1.79	3.11 ± 1.55	2.32 ± 1.47	0.51 ± 0.97	-0.85 ± 1.75	-0.89 ± 1.61
CORAL	4.95 ± 2.95	3.87 ± 2.33	0.56 ± 1.75	-1.41 ± 1.09	-2.0 ± 0.96	-1.81 ± 0.57
MetricMM	**8.69 ± 0.62**	**9.44 ± 0.44**	**9.19 ± 1.29**	**7.97 ± 0.3**	**8.55 ± 0.82**	**7.9 ± 1.44**

Table D.7.19: `Fetch` - Slide — Salt and pepper noise — 1 noisy modality.

Model	0.1	0.25	0.5	0.75	0.9	0.99
LinearComb	6.48 ± 0.14	5.76 ± 0.24	5.64 ± 1.01	5.34 ± 0.81	4.82 ± 1.31	3.37 ± 1.21
ConCat	6.41 ± 1.52	6.05 ± 1.48	6.48 ± 1.03	5.18 ± 1.59	3.49 ± 2.32	3.6 ± 2.02
CURL	8.55 ± 1.54	7.14 ± 0.76	6.51 ± 0.27	5.73 ± 1.55	3.95 ± 0.79	3.15 ± 0.85
GMC	7.34 ± 1.55	6.36 ± 0.85	4.87 ± 1.99	4.21 ± 2.5	2.46 ± 1.51	1.16 ± 2.65
AMDF	4.13 ± 2.24	3.01 ± 1.32	2.85 ± 1.1	1.65 ± 1.31	0.91 ± 0.47	0.2 ± 0.44
CORAL	5.04 ± 3.53	4.3 ± 2.26	5.29 ± 2.91	4.61 ± 2.6	2.46 ± 1.97	2.33 ± 1.59
MetricMM	**9.57 ± 0.9**	**8.69 ± 0.79**	**9.64 ± 0.65**	**8.54 ± 1.41**	**8.03 ± 0.86**	**8.71 ± 0.7**

Table D.7.20: `Fetch` - Slide — Texture noise — 1 noisy modality.

Model	0.1	0.25	0.5	0.75	0.9	0.99
LinearComb	6.54 ± 0.73	7.09 ± 0.08	4.63 ± 1.19	2.07 ± 1.11	2.69 ± 2.55	1.02 ± 1.74
ConCat	7.32 ± 1.39	5.64 ± 1.38	5.32 ± 2.07	4.93 ± 2.86	4.59 ± 2.54	3.52 ± 3.85
CURL	8.01 ± 0.92	5.71 ± 0.92	2.85 ± 1.27	-0.52 ± 0.52	-2.22 ± 0.22	-2.48 ± 0.37
GMC	6.36 ± 0.8	4.43 ± 0.41	1.9 ± 1.58	-1.27 ± 1.59	-2.15 ± 2.31	-2.65 ± 0.57
AMDF	3.28 ± 1.83	2.69 ± 1.71	1.03 ± 0.35	-0.65 ± 0.52	-1.5 ± 0.77	-1.83 ± 0.68
CORAL	4.49 ± 2.76	3.16 ± 2.03	1.71 ± 1.34	-1.02 ± 0.28	-1.75 ± 0.79	-2.43 ± 0.86
MetricMM	**9.45 ± 0.49**	**9.01 ± 0.97**	**9.4 ± 0.65**	**7.54 ± 0.18**	**9.02 ± 1.12**	**7.6 ± 0.47**

Table D.7.21: `Fetch` - Slide — Gaussian noise — 2 noisy modalities.

Model	0.1	0.25	0.5	0.75	0.9	0.99
LinearComb	6.65 ± 1.03	4.15 ± 0.62	3.28 ± 0.66	-0.25 ± 1.3	-0.15 ± 1.23	-1.22 ± 1.08
ConCat	6.05 ± 0.65	5.69 ± 0.73	4.63 ± 1.14	2.64 ± 1.41	0.87 ± 1.13	0.31 ± 1.46
CURL	7.06 ± 0.71	6.68 ± 0.11	3.93 ± 0.39	-0.08 ± 1.13	-1.12 ± 1.51	-1.46 ± 0.77
GMC	5.28 ± 0.58	3.67 ± 1.57	1.05 ± 1.06	-2.04 ± 1.34	-2.45 ± 1.56	-3.75 ± 2.28
AMDF	4.02 ± 1.71	2.14 ± 0.89	-0.11 ± 0.61	-1.1 ± 1.56	-2.45 ± 2.66	-2.81 ± 1.7
CORAL	4.45 ± 2.76	3.0 ± 1.74	1.14 ± 0.57	-2.03 ± 1.51	-1.61 ± 0.74	-2.37 ± 0.57
MetricMM	**10.1 ± 0.61**	**8.71 ± 0.56**	**7.15 ± 0.76**	**6.13 ± 1.63**	**4.38 ± 2.15**	**2.47 ± 1.89**

Table D.7.22: Fetch - Slide — Hallucination noise — 2 noisy modalities.

Model	0.1	0.25	0.5	0.75	0.9	0.99
LinearComb	6.64 ± 1.36	5.88 ± 0.37	2.73 ± 0.68	1.51 ± 0.36	-0.01 ± 0.3	-0.71 ± 0.4
ConCat	6.68 ± 1.49	5.88 ± 0.4	3.37 ± 1.53	0.77 ± 0.64	-1.2 ± 1.42	-0.46 ± 0.6
CURL	8.23 ± 0.94	7.85 ± 0.65	4.08 ± 0.57	1.48 ± 1.29	-0.15 ± 0.69	0.21 ± 0.68
GMC	6.62 ± 0.29	4.99 ± 1.0	1.82 ± 1.34	0.03 ± 0.39	-1.02 ± 1.24	-1.86 ± 0.47
AMDF	3.74 ± 1.24	3.19 ± 1.4	1.71 ± 0.48	0.07 ± 0.8	-0.29 ± 0.2	-0.51 ± 1.08
CORAL	5.5 ± 3.2	4.39 ± 2.65	1.96 ± 2.1	-0.03 ± 0.8	-1.66 ± 0.57	-1.36 ± 0.56
MetricMM	**8.67 ± 1.48**	**8.88 ± 0.55**	**8.15 ± 1.19**	**6.94 ± 0.87**	**5.94 ± 0.97**	**3.32 ± 1.65**

Table D.7.23: Fetch - Slide — Patches noise — 2 noisy modalities.

Model	0.1	0.25	0.5	0.75	0.9	0.99
LinearComb	5.31 ± 0.75	5.32 ± 1.59	1.85 ± 2.11	-0.65 ± 1.31	-2.1 ± 0.32	-2.05 ± 0.92
ConCat	6.85 ± 1.73	5.96 ± 1.41	2.14 ± 0.89	-0.66 ± 1.37	-1.77 ± 0.87	-1.93 ± 0.61
CURL	8.31 ± 1.27	5.58 ± 1.18	2.58 ± 2.41	-1.27 ± 0.75	-2.74 ± 1.79	-2.16 ± 1.2
GMC	4.4 ± 1.32	2.45 ± 1.09	-0.65 ± 1.59	-1.02 ± 0.49	-2.23 ± 1.04	-1.97 ± 0.22
AMDF	3.24 ± 1.94	2.33 ± 0.72	0.74 ± 1.77	-1.37 ± 1.86	-2.06 ± 2.96	-2.54 ± 2.7
CORAL	3.69 ± 1.61	1.53 ± 1.32	-0.91 ± 0.71	-2.46 ± 0.85	-2.31 ± 0.81	-2.1 ± 1.27
MetricMM	**8.62 ± 0.73**	**7.81 ± 0.74**	**4.03 ± 0.74**	**1.74 ± 1.77**	**-0.84 ± 1.57**	**-0.12 ± 1.46**

Table D.7.24: Fetch - Slide — Puzzle noise — 2 noisy modalities.

Model	0.1	0.25	0.5	0.75	0.9	0.99
LinearComb	5.82 ± 0.77	5.18 ± 0.76	1.76 ± 0.79	-1.67 ± 0.55	-2.74 ± 0.62	-1.96 ± 0.58
ConCat	7.14 ± 0.98	5.31 ± 1.1	2.02 ± 1.54	0.5 ± 2.03	-1.9 ± 1.54	-2.73 ± 1.48
CURL	7.42 ± 0.74	5.5 ± 0.57	2.67 ± 1.25	0.96 ± 0.69	-0.27 ± 0.66	-0.76 ± 0.7
GMC	6.25 ± 0.32	2.82 ± 1.22	0.08 ± 0.94	-1.44 ± 1.25	-2.63 ± 0.91	-2.37 ± 1.9
AMDF	4.09 ± 2.54	3.36 ± 2.0	0.55 ± 2.42	-1.5 ± 2.04	-2.03 ± 1.51	-1.76 ± 1.74
CORAL	5.19 ± 3.33	3.62 ± 1.92	0.71 ± 0.33	-1.01 ± 0.66	-2.71 ± 0.14	-2.28 ± 1.06
MetricMM	**9.12 ± 0.87**	**7.05 ± 1.52**	**6.17 ± 0.06**	**3.79 ± 0.77**	**1.43 ± 0.58**	**1.21 ± 1.53**

Table D.7.25: Fetch - Slide — Salt and pepper noise — 2 noisy modalities.

Model	0.1	0.25	0.5	0.75	0.9	0.99
LinearComb	6.3 ± 0.8	6.5 ± 1.13	5.02 ± 1.15	1.6 ± 1.72	-0.03 ± 1.06	0.79 ± 1.85
ConCat	5.56 ± 1.4	4.1 ± 0.62	3.58 ± 1.86	1.11 ± 2.76	0.29 ± 1.74	-0.23 ± 1.32
CURL	8.33 ± 1.67	6.01 ± 0.28	4.61 ± 1.34	1.83 ± 2.3	-0.12 ± 0.29	-1.18 ± 0.43
GMC	6.29 ± 0.45	5.36 ± 1.72	1.97 ± 1.76	0.75 ± 2.56	-1.74 ± 2.96	-1.08 ± 3.09
AMDF	2.6 ± 1.88	2.74 ± 1.04	1.98 ± 1.55	-1.35 ± 1.55	-1.32 ± 1.97	-2.3 ± 2.12
CORAL	5.22 ± 2.71	4.64 ± 2.74	3.3 ± 2.01	0.77 ± 1.75	-0.14 ± 0.24	0.11 ± 0.52
MetricMM	**9.2 ± 0.94**	**7.18 ± 2.01**	**6.79 ± 1.56**	**5.79 ± 1.27**	**2.97 ± 2.65**	**3.23 ± 2.2**

References

[1] Ankesh Anand, Evan Racah, Sherjil Ozair, et al. “Unsupervised state representation learning in atari”. In: *Advances in neural information processing systems* 32 (2019).

[2] Chelsea Finn, Xin Yu Tan, Yan Duan, et al. “Deep spatial autoencoders for visuomotor learning”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 512–519.

[3] Carlos Florensa, Jonas Degrave, Nicolas Heess, et al. “Self-supervised learning of image embedding for continuous control”. In: *arXiv preprint arXiv:1901.00943* (2019).

[4] Jürgen Ackermann, Andrew Bartlett, Dieter Kaesbauer, et al. *Robust control: Systems with uncertain physical parameters*. Springer, 1993.

[5] RE Kalman. “A new approach to linear filtering and prediction problems”. In: *Trans. ASME, D* 82 (1960), pp. 35–44.

[6] Neil J Gordon, David J Salmond, and Adrian FM Smith. “Novel approach to nonlinear/non-Gaussian Bayesian state estimation”. In: *IEE proceedings F (radar and signal processing)*. Vol. 140. IET. 1993, pp. 107–113.

[7] Sebastian Thrun. “Probabilistic robotics”. In: *Communications of the ACM* 45.3 (2002), pp. 52–57.

[8] Yoshua Bengio, Aaron Courville, and Pascal Vincent. “Representation learning: A review and new perspectives”. In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1798–1828.

[9] S Hochreiter. “Long Short-term Memory”. In: *Neural Computation MIT-Press* (1997).

[10] Rahul G Krishnan, Uri Shalit, and David Sontag. “Deep kalman filters”. In: *arXiv preprint arXiv:1511.05121* (2015).

[11] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, et al. “Neural processes”. In: *arXiv preprint arXiv:1807.01622* (2018).

[12] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), pp. 436–444.

[13] Jens Kober, J Andrew Bagnell, and Jan Peters. “Reinforcement learning in robotics: A survey”. In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1238–1274.

[14] Timothée Lesort, Natalia Díaz-Rodríguez, Jean-Franois Goudou, et al. “State representation learning for control: An overview”. In: *Neural Networks* 108 (2018), pp. 379–392.

[15] Jelle Munk, Jens Kober, and Robert Babuška. “Learning state representation for deep actor-critic control”. In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE. 2016, pp. 4667–4673.

[16] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, et al. “Asynchronous methods for deep reinforcement learning”. In: *International conference on machine learning*. PMLR. 2016, pp. 1928–1937.

[17] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. “Curl: Contrastive unsupervised representations for reinforcement learning”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 5639–5650.

[18] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, et al. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2015).
[19] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, et al. "Soft actor-critic algorithms and applications". In: *arXiv preprint arXiv:1812.05905* (2018).
[20] David Ha and Jürgen Schmidhuber. "World models". In: *arXiv preprint arXiv:1803.10122* (2018).
[21] Danijar Hafner, Timothy Lillicrap, Ian Fischer, et al. "Learning latent dynamics for planning from pixels". In: *International conference on machine learning*. PMLR. 2019, pp. 2555–2565.
[22] Michael Janner, Qiyang Li, and Sergey Levine. "Offline reinforcement learning as one big sequence modeling problem". In: *Advances in neural information processing systems* 34 (2021), pp. 1273–1286.
[23] Manuel Watter, Jost Springenberg, Joschka Boedecker, et al. "Embed to control: A locally linear latent dynamics model for control from raw images". In: *Advances in neural information processing systems* 28 (2015).
[24] Marvin Zhang, Sharad Vikram, Laura Smith, et al. "Solar: Deep structured representations for model-based reinforcement learning". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 7444–7453.
[25] Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, et al. "Contrastive learning as goal-conditioned reinforcement learning". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 35603–35620.
[26] Simon Haykin. *Kalman filtering and neural networks*. John Wiley & Sons, 2004.
[27] Maximilian Karl, Maximilian Soelch, Justin Bayer, et al. "Deep variational bayes filters: Unsupervised learning of state space models from raw data". In: *arXiv preprint arXiv:1605.06432* (2016).
[28] Tuomas Haarnoja, Anurag Ajay, Sergey Levine, et al. "Backprop kf: Learning discriminative deterministic state estimators". In: *Advances in neural information processing systems* 29 (2016).
[29] Xiao Liu, Yifan Zhou, Shuhei Ikemoto, et al. "α-MDF: An Attention-based Multimodal Differentiable Filter for Robot State Estimation". In: *7th Annual Conference on Robot Learning*. 2023.
[30] Stephen A Billings. *Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains*. John Wiley & Sons, 2013.
[31] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, et al. "Dream to control: Learning behaviors by latent imagination". In: *arXiv preprint arXiv:1912.01603* (2019).
[32] Philipp Becker, Niklas Freymuth, and Gerhard Neumann. "KalMamba: Towards Efficient Probabilistic State Space Models for RL under Uncertainty". In: *arXiv preprint arXiv:2406.15131* (2024).
[33] Andreas Doerr, Christian Daniel, Martin Schiegg, et al. "Probabilistic recurrent state-space models". In: *International conference on machine learning*. PMLR. 2018, pp. 1280–1289.
[34] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[35] Fei Deng, Ingook Jang, and Sungjin Ahn. “Dreamerpro: Reconstruction-free model-based reinforcement learning with prototypical representations”. In: *International conference on machine learning*. PMLR. 2022, pp. 4956–4975.

[36] Philipp Becker, Sebastian Mossburger, Fabian Otto, et al. “Combining reconstruction and contrastive methods for multimodal representations in RL”. In: *arXiv preprint arXiv:2302.05342* (2023).

[37] Myong Chol Jung, He Zhao, Joanna Dipnall, et al. “Beyond unimodal: Generalising neural processes for multimodal uncertainty estimation”. In: *Advances in Neural Information Processing Systems* 36 (2024).

[38] Qingyang Zhang, Haitao Wu, Changqing Zhang, et al. “Provable dynamic fusion for low-quality multimodal data”. In: *International conference on machine learning*. PMLR. 2023, pp. 41753–41769.

[39] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. “A reduction of imitation learning and structured prediction to no-regret online learning”. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 627–635.

[40] Tongzhou Wang, Antonio Torralba, Phillip Isola, et al. “Optimal Goal-Reaching Reinforcement Learning via Quasimetric Learning”. In: *arXiv preprint arXiv:2304.01203* (2023).

[41] Seohong Park, Oleh Rybkin, and Sergey Levine. “Metra: Scalable unsupervised rl with metric-aware abstraction”. In: *arXiv preprint arXiv:2310.08887* (2023).

[42] Balaraman Ravindran and Andrew G Barto. *Symmetries and model minimization in markov decision processes*. 2001.

[43] Elise Van der Pol, Daniel Worrall, Herke van Hoof, et al. “Mdp homomorphic networks: Group symmetries in reinforcement learning”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 4199–4210.

[44] Petra Poklukar, Miguel Vasco, Hang Yin, et al. “Geometric multimodal contrastive representation learning”. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 17782–17800.

Paper E

Paper E

Walking on the Fiber: A Simple Geometric Approximation for Bayesian Neural Networks

ALFREDO REICHLIN, MIGUEL VASCO AND DANICA KRAGIC

Published in
Transactions on Machine Learning Research (TMLR), 2025

Abstract:
Bayesian Neural Networks provide a principled framework for uncertainty quantification by modeling the posterior distribution of network parameters. However, exact posterior inference is computationally intractable, and widely used approximations like the Laplace method struggle with scalability and posterior accuracy in modern deep networks. In this work, we revisit sampling techniques for posterior exploration, proposing a simple variation tailored to efficiently sample from the posterior in over-parameterized networks by leveraging the low-dimensional structure of loss minima. Building on this, we introduce a model that learns a deformation of the parameter space, enabling rapid posterior sampling without requiring iterative methods. Empirical results demonstrate that our approach achieves competitive posterior approximations with improved scalability compared to recent refinement techniques. These contributions provide a practical alternative for Bayesian inference in deep learning.

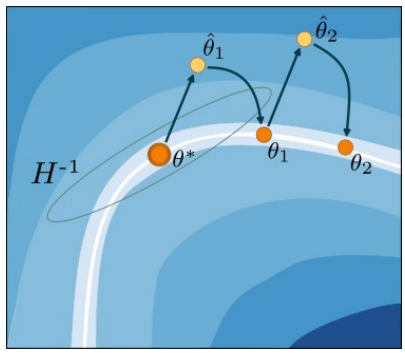


Figure E.0.4: We propose a novel sampling scheme to approximate the posterior of a trained network. When the loss landscape of a trained network over its parameters (blue regions) is low on a very low-dimensional curve (white region), a simple Hessian (H^{-1}) fails in capturing its distribution. Our sampling scheme is composed of two iterative steps: randomly perturb a given solution (yellow points) then refine to minimize the loss function over the given dataset (orange points). This proposed scheme allows to estimate posteriors of arbitrary shapes and it is well-suited when the space of solutions in a network is much lower dimensional than the parameter space.

E.1 Introduction

Bayesian Neural Networks (BNNs) have emerged as a principled framework for uncertainty quantification in deep learning by treating model parameters as random variables and inferring their posterior distribution. This Bayesian perspective is crucial for tasks requiring reliable decision-making under uncertainty, including medical diagnosis [1], autonomous driving [2], and weather forecasting [3]. Despite their appeal, the high dimensionality and non-linearity of modern neural network parameter spaces render exact posterior inference intractable, requiring the development of efficient approximation techniques.

A widely used method to make a deterministic, already-trained neural network Bayesian is the Laplace approximation, which estimates the posterior distribution of the parameters as a Gaussian centered at the Maximum a Posteriori (MAP) estimate [4, 5]. This approach is particularly convenient due to its simplicity and post-hoc applicability. However, the Laplace approximation faces significant challenges in modern deep networks. The computation and inversion of the Hessian scale poorly with network size [6], and its reliance on local curvature limits its ability to capture the complex, non-linear geometry of the posterior in over-parameterized models [7, 8].

In contrast, sampling techniques, though less frequently used in recent years, can be surprisingly effective in exploring the posterior when the parameter space of loss minima resides on a much lower-dimensional manifold than the full parameter space [9]. By leveraging this property, sampling methods can efficiently explore the posterior with fewer samples, especially in over-parameterized settings. However, relying on sampling for posterior estimation can still be computationally expensive if every new

inference requires a fresh sampling process. These limitations highlight the need for a method that combines the efficiency of sampling-based exploration with a more scalable and flexible posterior representation.

In this work, we propose a variation of a sampling method to explore the geometry of the loss landscape in the parameter space of neural networks. Our approach perturbs parameters around the MAP estimate using a set of drift directions and refines them with gradient updates, effectively maintaining computational efficiency as the network size increases. Using the data collected from our sampling process, we introduce a novel objective function to learn a structured latent representation of the posterior by deforming the original parameter space. This learned representation enables the direct generation of new parameter samples without relying on computationally expensive iterative sampling or restrictive variational inference approximations. We refer to these two components collectively as **MetricBNN**. Empirically, we demonstrate the efficacy of our method in estimating uncertainty from trained models in both regression and classification tasks. In particular, our approach produces better-calibrated uncertainty estimates than Gaussian approximations, especially on real-world datasets and high-dimensional tasks.

E.2 Related Work

Bayesian Neural Networks. BNNs provide a principled approach to uncertainty quantification by treating the network parameters θ as random variables and modeling their posterior distribution $p(\theta \mid \mathcal{D})$. However, exact inference is intractable due to the high dimensionality of parameter spaces in modern neural networks. Various methods have been proposed to approximate the posterior. Variational Inference (VI) approximates the posterior by optimizing a surrogate distribution, often in the form of a mean-field Gaussian [10, 11]. Extensions to hierarchical and amortized variational methods have improved scalability but often struggle to capture complex posterior structures [12]. The Laplace Approximation (LA) defines the posterior locally around the MAP estimate using a Gaussian distribution [13, 5]. Despite its simplicity and efficiency, it is limited by its reliance on local curvature and the Gaussian assumption. Sampling-based methods such as Stochastic Gradient Langevin Dynamics (SGLD) [14] and Hamiltonian Monte Carlo (HMC) [15] generate posterior samples by simulating stochastic dynamics. While these methods are more flexible, their computational cost often makes them impractical for large-scale neural networks.

Improving Posterior Samples. To address these limitations, recent works have sought to refine and extend the Laplace approximation by introducing more expressive approximate posteriors. [16] combines Laplace approximation with normalizing flows for a non-Gaussian posterior, refining the base Gaussian distribution. Similarly, [17] refines Laplace approximation by leveraging Gaussian variational methods and Gaussian processes, improving linearized Laplace posterior accuracy. [18] iteratively builds a mixture model to improve the posterior approximation by adding components to the variational distribution. [19] combines multiple pre-trained models to form a weighted sum of Laplace approximations, improving posterior flexibility. [20] introduced auxil-

iary variables to locally refine mean-field variational approximations, achieving better fit in regions of interest. [8] extends the Laplace approximation by leveraging Riemannian geometry to model the posterior distribution on a manifold, improving accuracy for non-linear loss landscapes.

In this paper, we re-evaluate the efficiency of sampling methods for posterior estimation, particularly in the context of over-parameterized neural networks. Through empirical results, we demonstrate that our proposed simple sampling framework can outperform modern posterior refinement techniques in efficiency and accuracy. Furthermore, our novel use of an autoencoder moves beyond the Gaussian formulation of the posterior, enabling a flexible, easy-to-sample representation that significantly improves performance. This approach allows us to capture the non-linear geometry of the posterior while maintaining computational simplicity, addressing key limitations of both classical and modern refinement methods. Differently from classic hypernetwork approaches, MetricBNN learns a structured latent representation of near-optimal parameter regions explicitly for posterior approximation. This enables us to generate parameter samples efficiently while preserving a geometry-informed posterior structure, directly targeting calibrated Bayesian uncertainty estimation rather than task-conditioned adaptation.

E.3 Preliminaries

BNNs provide a principled framework for quantifying uncertainty in neural network predictions by treating the model parameters as random variables. This section introduces the relevant background on BNNs, discusses the Laplace approximation for posterior estimation, and highlights its limitations.

Consider an independent and identically distributed (i.i.d.) dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ where $x_i \in \mathbb{R}^D$ and $y_i \in \mathbb{R}^C$. Let $f_\theta : \mathbb{R}^D \to \mathbb{R}^C$ denote a parametric function (e.g., a neural network) with parameters $\theta \in \mathbb{R}^K$. The goal is to model the predictive distribution: $p(y' \mid x', \mathcal{D}) = \int p(y' \mid x', \theta) p(\theta \mid \mathcal{D})\, d\theta$, where $p(\theta \mid \mathcal{D})$ is the posterior distribution of the parameters given the dataset. Bayesian inference provides a framework for estimating the posterior $p(\theta \mid \mathcal{D})$ using Bayes' theorem: $p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}$, where $p(\mathcal{D} \mid \theta)$ is the likelihood of the data, $p(\theta)$ is the prior distribution over the parameters, and $p(\mathcal{D})$ is the marginal likelihood or evidence. In practice, computing the evidence $p(\mathcal{D})$ is often intractable, making direct posterior computation challenging.

E.3.1 Laplace Approximation

The Laplace approximation is a classic method for approximating the posterior distribution $p(\theta \mid \mathcal{D})$ using a Gaussian centered at the MAP estimate. Let $\mathcal{L}(\theta)$ denote the regularized negative log-likelihood:

$$\mathcal{L}(\theta) = -\log p(\mathcal{D} \mid \theta) - \alpha \log p(\theta), \tag{E.3.1}$$

where α is a regularization coefficient and depends on the choice of prior for the parameters. Assuming this to be the Normal distribution, the regularization can be rewritten

as an Euclidean norm.

One of the main advantages of the Laplace approximation is that it allows to define a posterior distribution given an already fully trained network, *post-hoc*. When given a fully trained network we assume access to the set of parameters, θ^*, that minimize the loss function of Equation E.3.1. This is the MAP solution. The exponential of this loss function is proportional to the posterior distribution as $p(\theta \mid \mathcal{D}) \propto p(\mathcal{D} \mid \theta)p(\theta)$. Laplace approximation methods propose to approximate the posterior distribution with a second-order Taylor expansion around the MAP of the loss function. This results in a Gaussian approximation of the parameters with the mean being the MAP solution and the covariance being the inverse of the Hessian computed in the MAP. The posterior can then be defined as:

$$q(\theta) = \mathcal{N}(\theta^*, H^{-1}), \tag{E.3.2}$$

where $H = \nabla^2 \mathcal{L}(\theta^*)$ is the Hessian of the loss function at θ^*.

E.3.2 Limitations of the Laplace Approximation

While widely used, the Laplace approximation has several limitations:

1. **Local Approximation:** It captures only the local curvature of the loss landscape around θ^*, ignoring the broader structure of the posterior, which can be highly non-Gaussian in high-dimensional spaces [21].
2. **Scalability:** Computing and inverting the Hessian is computationally expensive for large-scale neural networks, limiting its applicability to small models [6].
3. **Positive-Definiteness:** In over-parameterized networks, the Hessian is often not positive definite, making it difficult to define a valid Gaussian approximation. Regularization or approximate methods are sometimes used to mitigate this issue, but these approaches can introduce biases [5].

The limitations of the Laplace approximation motivate the need for alternative methods that better capture the true posterior distribution. Specifically, the posterior for BNNs often lies on a complex, non-linear manifold in parameter space, which the Laplace approximation fails to represent. This motivates our approach, which combines efficient exploration of the parameter space with a flexible latent representation to better approximate the posterior.

E.4 Method

To address the limitations of the Laplace approximation and improve posterior estimation in BNNs, we propose *MetricBNN*, a two-step framework. The first step involves locally exploring the parameter space around the MAP estimate using a simple sampling method. The second step learns a latent representation to construct a flexible posterior distribution that captures the complex geometry of the parameter space.

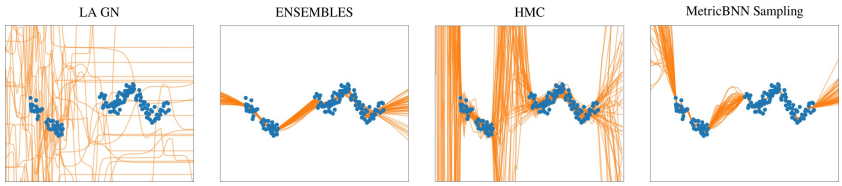


Figure E.4.1: Posterior samples for Regression task. The blue points represent the dataset, the orange lines are samples from the estimated posteriors. Our proposed sampling method correctly captures the uncertainty in the data gap.

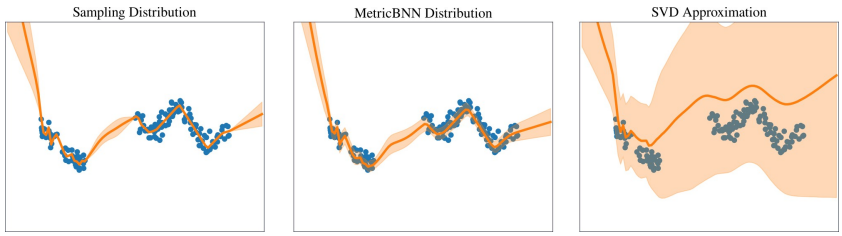


Figure E.4.2: Estimated posterior for the regression task. The blue points represent the dataset and in orange the mean and standard deviation of posteriors sampled from the estimated distributions. A naive SVD approximation of the solutions found with the sampling scheme fails in correctly representing the true posterior. Our proposed MetricBNN posterior correctly approximates it.

E.4.1 Exploring Neighbor Solutions

Given a trained neural network with parameters θ^* that minimize the regularized loss function (Equation E.3.1), the parameter space of deep networks is known to exhibit a high degree of redundancy due to overparameterization and reparametrization invariances [7, 9]. These properties create a connected, lower-dimensional manifold of near-optimal solutions surrounding the MAP estimate θ^*. This phenomenon, often referred to as the *linear connectivity assumption*, suggests that different sets of parameters can achieve similar performance, even when interpolating between them [9, 7, 22].

In particular, empirical studies on the loss landscapes of neural networks have shown that minima are often connected by low-loss paths, forming smooth and structured regions in the parameter space [9]. This implies that the posterior distribution is not confined to a single mode but instead spans a broader, non-linear manifold. Capturing this geometric complexity is crucial for accurately modeling the posterior. By exploring neighboring solutions around θ^*, we aim to gather representative samples that reflect the underlying structure of this manifold, providing a richer and more accurate approximation of the posterior distribution.

Estimating the posterior distribution of the parameters amounts to identifying the distribution of these solutions. To achieve this, we propose collecting a set of such solutions using the following sampling technique (MetricBNN sampling):

1. **Initialization:** Start with N particles, each initialized at the MAP estimate θ^*, i.e., $\theta_{i,0}$.
2. **Random Drift Assignment:** Assign each particle a random drift vector d_i, sampled as a unit vector with uniform orientation in the parameter space.
3. **Iterative Exploration:** For T time steps, update each particle's position as:
   (a) **Drift:** Perturb the particle by adding the corresponding random drift.

$$\hat{\theta}^0_{i,t+1} = \theta_{i,t} + \alpha \cdot d_i. \tag{E.4.1}$$

   (b) **Refinement:** After each drift update, refine the particle's position using M steps of gradient descent to ensure alignment with the loss landscape.

$$\theta^{m+1}_{i,t+1} = \hat{\theta}^m_{i,t+1} - \eta \nabla_\theta \mathcal{L}(\hat{\theta}^m_{i,t+1}) \tag{E.4.2}$$

$$\theta_{i,t+1} = \hat{\theta}^M_{i,t+1} \tag{E.4.3}$$

This procedure generates a set of $N \times T$ viable solutions near θ^*. These samples represent a local exploration of the posterior distribution of interest, capturing the diversity of solutions around the MAP estimate. Figure E.4.1(on the right) illustrates an example of these solutions for a two-dimensional regression problem.

While sampling methods have traditionally been considered computationally expensive for high-dimensional neural networks, leading to their relative neglect in favor of variational inference and other approximation techniques [13, 11], we argue that these concerns warrant re-evaluation in light of the linear connectivity assumption. Specifically, the high redundancy and structured geometry of neural network parameter spaces suggest that meaningful posterior exploration can be achieved through efficient sampling on a lower-dimensional, connected manifold of solutions [9, 7]. This implies that the computational complexity of sampling methods may scale more favorably with the dimensionality of modern networks than previously assumed.

The empirical covariance matrix of the collected solutions provides an improved local Gaussian approximation of the posterior distribution. However, due to the extreme non-linearity of the solution manifold, a Gaussian approximation alone fails to accurately capture the true structure of the posterior distribution. Addressing this limitation requires moving beyond the Gaussian assumption.

E.4.2 Defining a Posterior

While the empirical covariance of the collected samples provides a basic Gaussian approximation, it fails to capture the non-linear geometry of the posterior, see Figure E.4.2. Addressing this limitation requires moving beyond the Gaussian assumption. Given that the drift applied in the Drift step (3a) is small enough, we can assume that every element in between two sets of parameters is a viable solution. The curve defined

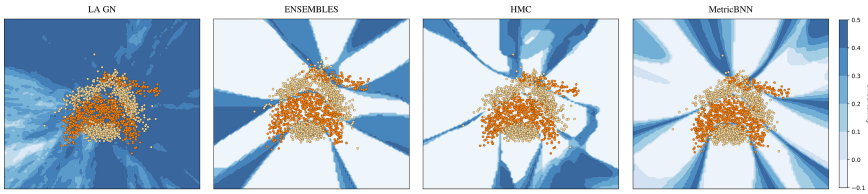


Figure E.4.3: Estimated posterior for the Banana classification task. The orange and yellow points represent the data of the two classes in the classification dataset. In blue is the value of the uncertainty of the posteriors sampled from the estimated distributions.

by all the samples can, however, be arbitrarily non-linear. To overcome this, we propose to learn a representation that maps the parameters into a structured latent space where the collected trajectories of parameters are linearly distributed.

We define a latent representation through an autoencoder framework:

- $\varphi : \Theta \to Z$: An encoder that maps neural network parameters θ to a latent space $Z \subseteq \mathbb{R}^k$.
- $\varphi^{-1} : Z \to \Theta$: A decoder that reconstructs parameters θ from the latent space.

The training dataset for the autoencoder consists of the collected samples, i.e., D_θ. The goal is to learn a latent space where the posterior distribution is well-structured and linearly interpolable. We achieve this by optimizing the following loss function:

$$\mathcal{L} = \lambda_+ \mathcal{L}_+ + \lambda_- \mathcal{L}_- + \mathcal{L}_d, \tag{E.4.4}$$

with:

$$\mathcal{L}_+ = \mathbb{E}_{D_\theta}\left[\left(||\varphi(\theta) - \varphi(\theta')|| - \frac{1}{T}\right)^2\right], \tag{E.4.5}$$

$$\mathcal{L}_- = \mathbb{E}_{D_\theta}\left[-\log\left(||\varphi(\theta) - \varphi(\theta'')||\right)\right], \tag{E.4.6}$$

$$\mathcal{L}_d = \mathbb{E}_{D_\theta}\left[||\varphi^{-1}(\varphi(\theta)) - \theta||^2\right], \tag{E.4.7}$$

where θ and θ' are two sets of parameters of the same trajectory and successive time step while θ'' is another randomly sampled set of parameters. Both λ_+ and λ_- are scalar values. The role of these terms is as follows:

- $\mathcal{L}_+$: encourages local distances between successive samples are preserved.
- $\mathcal{L}_-$: encourages these sets of parameters to stretch in the learned latent space by maximizing every pair-wise distance.
- $\mathcal{L}_d$: encourages reconstruction, mapping latent representations back to their original parameter space.

Using the trained autoencoder, we define the posterior distribution in the latent space Z, MetricBNN posterior. Each trajectory of samples is treated as independent, with

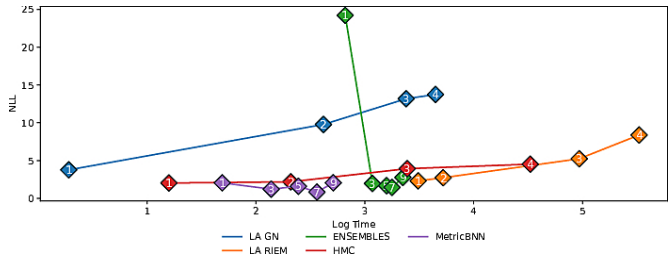


Figure E.5.1: Trade-off between network size and computational complexity on the regression task. Each marker shows NLL (lower is better) for a network with the indicated number of hidden layers. MetricBNN's cost grows only mildly with depth.

a uniform probability assigned to each trajectory. Along each trajectory, we place a uniform probability between the MAP solution and the last sample of such trajectory, i.e., $\theta_{i,T}$. Sampling from the posterior involves: Sample a trajectory index i uniformly, Sample a scaling factor $\epsilon \sim \text{Uniform}(0, 1)$. Compute the parameter sample:

$$\theta = \varphi^{-1}\left(\varphi(\theta^*) + \epsilon \cdot (\varphi(\theta_{i,T}) - \varphi(\theta^*))\right). \tag{E.4.8}$$

This method captures the broader geometry of the posterior while maintaining computational efficiency. We provide a pseudo-code of the whole method in Appendix E.8.2

E.5 Experiments

In this section, we evaluate the ability of our proposed method to approximate the posterior efficiently. We first assess the quality of our sampling technique in capturing model uncertainty using simple regression and classification tasks. We then analyze the computational complexity of our approach as the network size increases. Finally, we present quantitative results on real-world datasets, evaluating the negative log-likelihood (NLL) and the expected calibration error (ECE) of our posterior distribution.

We benchmark our method against Bayesian inference baselines that collectively span local–Gaussian, stochastic–sampling, and deterministic–ensembling paradigms.

Gauss–Newton Laplace (LA GN) [5] fits a Gaussian around the MAP solution using the Gauss–Newton curvature, offering a lightweight post-hoc posterior of any trained network. For some experiments, we also include a low-rank estimate of the Laplace approximation (LA LowRank) and a linearized version (Lin LA).

Riemannian Laplace (Riem LA) [8] extends this idea by replacing the Euclidean metric with an appropriate Riemannian version, yielding a curvature-aware Gaussian that better respects the manifold structure of the parameter space. We also include the linearized version of this approach (Riem Lin LA).

For fully stochastic references we include Hamiltonian Monte Carlo (HMC) [15] and its scalable stochastic-gradient variant (SGLD) [14], both of which provide asymptotically exact posterior samples at the cost of increased computation.

Complementing these, Deep Ensembles [23] (ENSEMBLES) approximate the posterior with a set of independently trained networks, while (SWAG) [24] captures the first two moments of SGD iterates to form a low-rank Gaussian that can be sampled cheaply.

Finally, (ESPRO) [25] exploits mode-connecting simplices to generate functionally diverse checkpoints without extra training epochs, yielding a fast ensembling baseline.

We conduct experiments across five different settings:

Simple Regression: We evaluate our method on the one-dimensional regression problem introduced in [26]. The dataset consists of 200 points, with 50 held out to assess the model's ability to estimate uncertainty. We use a fully connected neural network with three hidden layers of 32 units and ReLU activations.

Simple Classification: We test our method on the Banana dataset, a two-dimensional binary classification task with 5,300 points, of which 30% are reserved for testing. We use a fully connected network with two hidden layers of 6 units and Tanh activations.

Classification on UCI Datasets: To evaluate performance on real-world structured data, we experiment with six classification datasets from the UCI repository [27]. We use a fully connected network with two hidden layers of 32 units and ReLU activations.

Image Classification: For high-dimensional problems, we experiment with the MNIST [28], FashionMNIST [29] and CIFAR10 [30] datasets. For MNIST and FMNIST we use a shallow convolutional neural network with two convolutional layers followed by three fully connected layers with Tanh activations. For CIFAR10 we instead use a ResNet18 architecture [31] and compute the posterior on the last two layers of the network.

OOD Classification: We test the ability of the model in classifying MNIST images with an increasing amount of rotations [32] and CIFAR10 images with 15 different realistic noises [33].

E.5.1 Scalability of the Proposed Sampling Method

Approximating the posterior distribution in neural networks is challenging due to the high-dimensional and non-linear nature of the parameter space. The standard Laplace approximation has been shown to struggle in these settings [34, 35], as it assumes a Gaussian posterior, which may not align well with the actual distribution.

Our proposed sampling method provides a more flexible alternative, allowing for a tighter and better-calibrated posterior that is independent of the loss landscape's curva-

Table E.5.1: Quantitative results on the UCI datasets, including NLL, lower is better, and ECE, lower is better.

	Model	Australian	Breast	Glass	Ionosphere	Vehicle	Waveform
NLL	LA GN	0.71 ± 0.06	0.73 ± 0.07	2.28 ± 0.28	0.72 ± 0.09	0.8 ± 0.16	0.88 ± 0.09
	LA LowRank	0.69 ± 0.03	0.64 ± 0.10	2.06 ± 0.15	1.08 ± 0.21	0.76 ± 0.05	0.96 ± 0.04
	Riem LA	0.54 ± 0.07	0.59 ± 0.1	1.42 ± 0.21	0.17 ± 0.02	0.65 ± 0.02	0.3 ± 0.01
	Lin LA	0.69 ± 0.04	0.61 ± 0.04	3.59 ± 1.17	0.42 ± 0.04	0.68 ± 0.01	0.4 ± 0.02
	Riem Lin LA	0.74 ± 0.05	2.32 ± 0.74	15.92 ± 0.05	13.5 ± 0.66	0.65 ± 0.01	0.38 ± 0.02
	ENSEMBLES	0.51 ± 0.07	0.62 ± 0.10	0.90 ± 0.08	0.25 ± 0.05	0.63 ± 0.00	0.30 ± 0.01
	HMC	1.30 ± 0.31	1.52 ± 0.61	1.49 ± 0.40	0.80 ± 0.30	0.65 ± 0.02	0.43 ± 0.03
	SGLD	0.64 ± 0.16	0.65 ± 0.10	1.36 ± 0.03	0.26 ± 0.07	0.73 ± 0.02	0.72 ± 0.09
	ESPRO	0.76 ± 0.25	0.63 ± 0.12	0.94 ± 0.17	0.22 ± 0.10	0.64 ± 0.03	0.32 ± 0.01
	SWAG	0.70 ± 0.04	0.59 ± 0.06	1.56 ± 0.08	0.25 ± 0.08	0.68 ± 0.00	0.30 ± 0.01
	MetricBNN	0.66 ± 0.05	0.57 ± 0.04	1.07 ± 0.07	0.17 ± 0.05	0.69 ± 0.02	0.3 ± 0.01
ECE	LA GN	13.61 ± 4.34	18.53 ± 9.51	12.25 ± 3.56	28.03 ± 7.23	21.36 ± 9.26	21.68 ± 6.40
	LA LowRank	17.12 ± 3.17	16.66 ± 9.26	17.72 ± 10.25	49.68 ± 5.82	18.88 ± 6.29	13.82 ± 4.38
	Riem LA	10.28 ± 4.83	13.60 ± 2.91	21.30 ± 7.77	5.57 ± 0.78	5.80 ± 1.10	5.62 ± 0.70
	Lin LA	10.62 ± 1.71	18.45 ± 4.57	15.32 ± 7.21	28.80 ± 1.67	7.31 ± 1.30	23.00 ± 1.82
	Riem Lin LA	8.47 ± 2.71	20.90 ± 2.58	14.10 ± 2.14	6.84 ± 2.31	10.75 ± 1.88	2.51 ± 0.72
	ENSEMBLES	8.09 ± 1.37	15.46 ± 3.77	13.47 ± 1.57	8.24 ± 1.14	2.90 ± 1.30	1.39 ± 0.26
	HMC	16.03 ± 2.87	26.04 ± 4.98	18.32 ± 2.43	9.24 ± 2.10	8.46 ± 1.17	6.53 ± 0.96
	SGLD	13.38 ± 3.36	15.12 ± 6.35	18.40 ± 3.09	8.12 ± 2.58	12.12 ± 3.03	8.70 ± 4.52
	ESPRO	11.62 ± 3.33	15.41 ± 3.37	14.03 ± 2.91	6.47 ± 1.72	8.29 ± 1.62	2.51 ± 0.96
	SWAG	9.80 ± 6.35	14.29 ± 2.49	3.87 ± 2.75	7.44 ± 2.12	1.18 ± 0.98	2.39 ± 0.59
	MetricBNN	27.33 ± 8.08	12.42 ± 4.38	11.51 ± 5.46	5.74 ± 0.97	10.18 ± 10.20	1.50 ± 0.53

ture. Figure E.4.1 compares our method with the Laplace approximation on the toy regression task, demonstrating a significantly improved uncertainty estimation.

Beyond accuracy, scalability is a key factor in Bayesian inference for deep networks. Many posterior approximation techniques rely on computing second-order derivatives, making them computationally expensive as the network size grows. To assess the computational trade-off, we analyze the inference time and NLL performance as the number of layers in the network increases. Figure E.5.1 presents these results for our method, LA GN, Riem LA, ENSEMBLES, HMC. Results show that Hessian-based methods and HMC scale poorly with size. ENSEMBLES have a higher temporal cost as they require the training of multiple models. Moreover, they suffer from a very low-dimensional network as they converge to the same solution and thus result in low variance solutions. Our sampling approach maintains a reasonable computational cost while preserving accuracy. Furthermore, our learned latent posterior model enables rapid parameter sampling, further reducing the overhead compared to iterative sampling.

E.5.2 Quality of the Proposed Geometric Posterior

A potential concern is whether the posterior obtained via our sampling method could be approximated using a naive covariance computation, similar to the Laplace approximation. However, as shown in Figure E.4.2, this approach still does not yield accurate uncertainty estimates. By contrast, our latent posterior representation effectively cap-

Table E.5.2: Quantitative results on image datasets, including NLL, lower is better, and ECE, lower is better.

	Model	MNIST	FMNIST	CIFAR10
NLL	LA GN	2.42 ± 0.03	2.24 ± 0.08	2.29 ± 0.09
	LA LowRank	1.35 ± 0.04	1.67 ± 0.06	2.19 ± 0.08
	Riem LA	1.11 ± 0.11	1.20 ± 0.02	0.84 ± 0.03
	Lin LA	0.96 ± 0.08	1.27 ± 0.13	1.93 ± 0.12
	Riem Lin LA	0.62 ± 0.09	0.87 ± 0.07	0.83 ± 0.06
	ENSEMBLES	0.17 ± 0.00	0.47 ± 0.00	1.55 ± 0.02
	SGLD	1.60 ± 0.20	1.98 ± 0.26	0.78 ± 0.06
	ESPRO	0.15 ± 0.02	0.51 ± 0.02	0.84 ± 0.05
	SWAG	0.24 ± 0.01	0.89 ± 0.11	1.97 ± 0.33
	MetricBNN	0.12 ± 0.01	0.58 ± 0.09	0.70 ± 0.04
ECE	LA GN	4.78 ± 2.97	12.31 ± 2.39	4.59 ± 1.88
	LA LowRank	50.58 ± 3.20	18.70 ± 3.07	9.74 ± 5.82
	Riem LA	47.14 ± 9.19	26.78 ± 7.83	33.91 ± 8.23
	Lin LA	29.43 ± 1.32	19.63 ± 1.18	35.46 ± 12.67
	Riem Lin LA	21.20 ± 2.00	15.19 ± 1.92	32.10 ± 3.15
	ENSEMBLES	10.52 ± 0.13	11.33 ± 0.29	8.94 ± 0.17
	SGLD	13.83 ± 7.47	14.55 ± 6.06	7.45 ± 1.25
	ESPRO	1.56 ± 0.20	4.36 ± 0.79	10.40 ± 0.33
	SWAG	13.47 ± 1.07	25.25 ± 4.95	3.47 ± 2.76
	MBNN	1.57 ± 0.31	13.57 ± 6.89	9.26 ± 0.45

tures the true posterior structure, demonstrating the benefits of learning a structured parameter space.

E.5.3 Performance on UCI Regression and Image Classification

We further evaluate our approach to real-world structured data by testing it on standard UCI regression benchmarks. Table E.5.1 reports NLL and ECE results, showing that our method remains competitive with the baselines. In Appendix we provide details on the computational cost as well demonstrating that our method offers a good balance between accuracy and computational cost.

To assess performance in high-dimensional settings, we train convolutional neural networks on MNIST and FashionMNIST and CIFAR10. Table E.5.2 presents NLL and ECE results.

E.5.4 OOD Classification

We evaluate all methods under two canonical OOD settings. For every corruption we report the log-likelihood. All networks are first trained on clean data only. Posterior approximations are then fit *post-hoc*.

Geometric shift: progressively rotating MNIST test images by 0°–90° [32]. Figure E.5.3 shows that MetricBNN has competitive log-likelihood performances for rotations

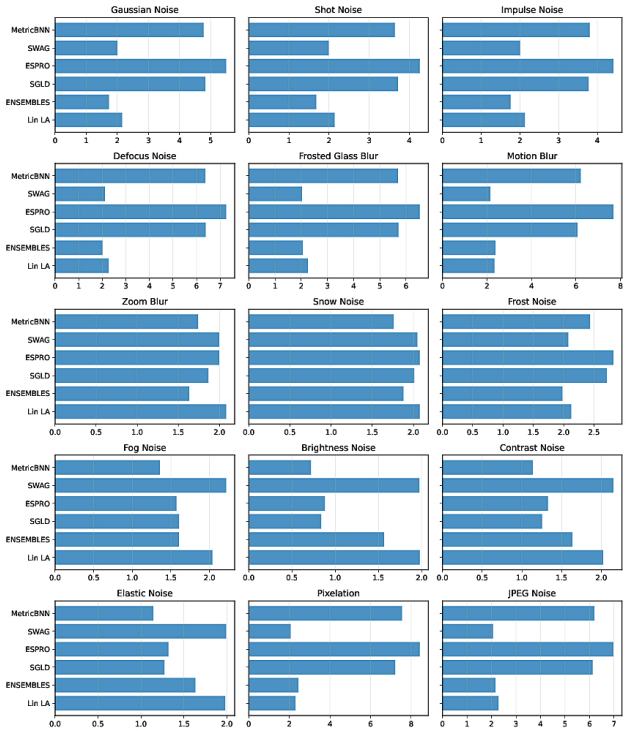


Figure E.5.2: Robustness to semantic and photometric shift CIFAR10. Each panel reports test NLL across each corruption.

up to 30 degrees. Performances drop when the rotations become more severe as distinguishing between the numbers is not always possible.

Semantic + photometric shift: applying the fifteen corruption families of CIFAR10 [33]. Figure E.5.2 shows competitive performances for MetricBNN for some of the perturbations.

Our experiments highlight three key takeaways: Our sampling-based posterior exploration provides a more flexible and well-calibrated uncertainty estimate, particularly as network size increases. The proposed latent posterior model enables efficient posterior sampling, avoiding the computational overhead of iterative methods. Our approach remains competitive in terms of NLL and ECE on real-world datasets while being significantly more computationally effi-

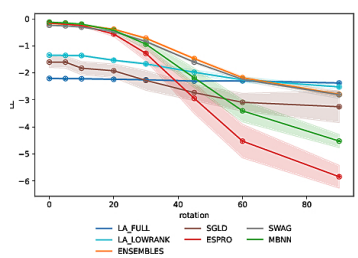


Figure E.5.3: Robustness to geometric shift (Rotated MNIST). Test NLL and standard deviations of MNIST image classification versus rotation angle; lower is better.

cient.

E.6 Discussion

In this work, we revisited scalable Bayesian inference in deep learning, benchmarking our method, MetricBNN, against a broad range of baselines including Laplace approximations, variational methods, deep ensembles, and sampling-based techniques. Across structured and high-dimensional tasks, MetricBNN consistently achieves competitive or superior negative log-likelihood and calibration while maintaining efficiency as network size increases. We observe that while Laplace-based methods and their geometric extensions can perform well in certain settings, their performance is notably sensitive to the parameter count, network capacity, and the quality of the MAP solution used for centering the approximation. Differences in MAP training conditions and model size can therefore lead to the lower scores observed in our experiments compared to those reported in other Laplace-focused studies. Our sampling-based exploration combined with a learned latent posterior helps mitigate these sensitivities, capturing non-Gaussian, non-linear posterior structures more robustly across diverse regimes.

Across our experiments, MetricBNN demonstrates strong empirical performance on structured datasets, high-dimensional image classification, and out-of-distribution - (OOD) robustness benchmarks. On structured UCI datasets, MetricBNN achieves negative log-likelihood and calibration scores competitive with or better than Laplace (LA GN, Riem LA) and variational baselines, while providing a more flexible posterior representation. On high-dimensional datasets like MNIST, FashionMNIST, and CIFAR10, MetricBNN maintains low NLL and ECE scores, matching or outperforming deep ensembles and methods such as SWAG, while avoiding the computational overhead of training multiple models or computing Hessians at scale. In OOD settings, MetricBNN shows consistent robustness under geometric shifts (rotated MNIST) and photometric and semantic corruptions (CIFAR10), providing well-calibrated uncertainty estimates across perturbation levels.

Compared to fully sampling-based methods such as Hamiltonian Monte Carlo (HMC) and SGLD, MetricBNN achieves comparable uncertainty estimates while being significantly more computationally efficient, making it practical for large-scale neural network applications. In contrast to deep ensembles, which require training multiple networks to achieve uncertainty calibration, MetricBNN provides comparable calibration with a single network, improving efficiency and scalability. Compared to variational inference methods, which often rely on restrictive mean-field approximations, MetricBNN captures the non-Gaussian structure of the posterior by leveraging the geometry of the parameter space through structured latent representations.

Finally, while hypernetworks typically generate weights conditioned on input or task information for adaptation, our approach fundamentally differs by using an autoencoder to learn a structured latent space of near-optimal parameter regions for scalable posterior sampling. This allows MetricBNN to efficiently generate parameter samples

that reflect the geometry of the loss landscape without the use of expensive iterative techniques or variational methods.

E.7 Conclusions

In this work, we proposed a simple variation of sampling-based techniques tailored to explore the posterior geometry of Bayesian Neural Networks efficiently, even in over-parameterized settings. By leveraging the low-dimensional structure of loss minima, our method achieves competitive posterior approximations while maintaining scalability as network size increases. Additionally, we introduced a model that learns a deformation of the parameter space based on the collected samples, enabling rapid posterior sampling without requiring iterative methods. Our empirical results demonstrate that this approach improves posterior accuracy and computational efficiency compared to recent refinement techniques. These contributions provide a practical and flexible framework for Bayesian inference in deep learning, offering new directions for scalable uncertainty quantification in complex models.

Acknowledgments

This work was supported by the Swedish Research Council (2019-00748), the Knut and Alice Wallenberg Foundation, the European Research Council (ERC-BIRD-884807) and the European Horizon 2020 CANOPIES project.

E.8 Appendix

E.8.1 Experimental Details

This section provides additional details regarding the experimental setup, including the models, hyper-parameters, and computational settings.

Experimental Setup

We evaluate our method across four types of tasks: toy regression, toy classification, structured classification from the UCI repository, high-dimensional image classification and OOD classification.

Toy Regression. We consider the *Snelson 1D regression dataset* [26], consisting of 200 points, with 50 held out for evaluating uncertainty estimation. We use a fully connected neural network with three hidden layers of 32 units and ReLU activations. The model is trained using the Adam optimizer with a learning rate of 1×10^{-3}, batch size of 200, and L2 regularization of 0.01 for 50,000 epochs.

For the sampling procedure of our approach, we used a drift-scaling factor of $\alpha = 0.1$ and performed $T = 100$ sampling steps. At each step, we updated the parameters using $M = 10$ gradient refinement steps to improve posterior estimates. We initialized $N = 10$ particles, each following a perturbed trajectory in the parameter space, with

a drift step scaled by an inner learning rate of $\eta = 0.001$. For the autoencoder-based posterior model, we projected the sampled parameter trajectories into a structured latent space of $k = 32$ dimensions. The autoencoder was trained using batch size 1024 for 1000 epochs, optimizing the contrastive loss with weight coefficients $\lambda_+ = 1.0$ and $\lambda_- = 1.0$. The architecture consists of an encoder with two fully connected hidden layers of 256 units and ReLU activations, and a decoder with a symmetric structure of two hidden layers with 256 units and ReLU activations. Training was performed using the Adam optimizer.

Toy Classification. For classification, we use the *Banana dataset*, a 2D binary classification problem with 5300 points, of which 30% are used for testing. The network consists of two hidden layers with 16 units and Tanh activations. The training procedure follows the regression setup, except with a smaller batch size (32) and 2500 training epochs. For our model, we use the same parameters as in the regression task.

Structured Data (UCI Datasets). To evaluate our method in structured classification settings, we test on six datasets from the *UCI repository* [27], using a fully connected architecture with two hidden layers of 32 units and ReLU activations. The training follows the same schedule as before, with a batch size of 32 and 1000 training epochs. Our method is estimated with the same parameters as in the classification experiment except for $T = 50$ steps.

High-Dimensional Image Classification. For large-scale experiments, we train Bayesian neural networks on *MNIST* [28], *FashionMNIST* [29] and CIFAR10 [30]. For MNIST and FAMNIST we use a shallow convolutional neural network consisting of two convolutional layers, followed by three fully connected layers, with Tanh activations. For CIFAR10 we use a standard ResNet18 architecture. Training follows the same setup as UCI experiments, but with a reduced number of epochs (100). Our method follows the same parameters as in the UCI experiment.

OOD Classification. We use the same networks described above for both MNIST and CIFAR10 as well as the same parameters for our method.

Scalability and Computational Complexity

To conduct the experiment on the computational complexity depicted in Figure E.5.1 we used a fully connected network with hidden layers from 1 to 9. Each layer with 64 units and ReLu as activation function. For each architecture, we measure the inference time and negative log-likelihood (NLL) performance. To compute the posterior of each model we use 100 samples of the parameters. All experiments were conducted on an NVIDIA RTX3080 GPU.

E.8.2 Pseudocode

Algorithm 3 Proposed Posterior Approximation Method

Require: Trained neural network f_θ, dataset $\mathcal{D}$, drift scaling α, sampling steps T, gradient steps M, particles N, inner learning rate η, autoencoder latent dimension k, training epochs E

Ensure: Approximate posterior $q(\theta)$

Phase 1: Parameter Sampling
Initialize N particles at θ^* (MAP estimate)
for each particle $i = 1, \dots, N$ **do**
    Sample random drift direction $d_i \leftarrow \frac{v}{||v||}, \quad v \sim \mathcal{N}(0, I)$
    for each step $t = 1, \dots, T$ **do**
        Apply drift: $\theta_{i,t} = \theta_{i,t-1} + \alpha d_i$
        for each gradient step $m = 1, \dots, M$ **do**
            Refine using gradient descent:
            $\theta_{i,t} \leftarrow \theta_{i,t} - \eta \nabla_\theta \mathcal{L}(\theta_{i,t})$
        end for
    end for
end for
Store all sampled parameters $\Theta = \{\theta_{i,t}\}$
Phase 2: Learning Latent Posterior Representation
Train an autoencoder $\varphi : \Theta \to Z$ using:
for epoch $e = 1, \dots, E$ **do**
    Sample minibatch of subsequent parameters $\theta, \theta' \sim \Theta$
    Compute latent embeddings: $z = \varphi(\theta)$ and $z' = \varphi(\theta')$
    Compute negative parameters by reshuffling the batch: z''
    Compute contrastive loss:
    $\mathcal{L}_+ = (||z - z'|| - 1/T)^2$
    $\mathcal{L}_- = -\log(||z - z''||)$
    $\mathcal{L}_d = ||\varphi^{-1}(z) - \theta||$
    Update autoencoder using $\mathcal{L} = \lambda_+ \mathcal{L}_+ + \lambda_- \mathcal{L}_- + \lambda_d \mathcal{L}_d$
end for
Posterior Approximation and Sampling
Sample trajectory index $j \sim$ Categorical$(\{z_{\max,j}\}_{j=1}^N)$
Sample interpolation factor $\epsilon \sim \mathcal{U}(0, 1)$
Compute latent posterior sample: $z = z_{\text{MAP}} + \epsilon(z_{\max,j} - z_{\text{MAP}})$
Map to parameter space: $\theta = \varphi^{-1}(z)$
return Posterior sample θ

E.8.3 Time Complexity Results

We provide a more detailed comparison of the computational complexity and the accuracy of our method and the baselines for the UCI experiment and the images experiment.

Table E.8.1: Logarithmic time complexity in seconds and accuracy for the UCI datasets.

	Model	Australian	Breast	Glass	Ionosphere	Vehicle	Waveform
Log Time	LA GN	1.88 ± 0.98	1.54 ± 0.8	1.56 ± 0.67	1.88 ± 0.72	3.05 ± 1.95	2.68 ± 0.48
	LA LowRank	2.0 ± 0.22	1.24 ± 0.28	0.98 ± 0.37	1.47 ± 0.48	4.69 ± 0.1	3.68 ± 0.16
	Riem LA	4.36 ± 4.06	2.42 ± 1.61	3.69 ± 3.63	2.38 ± 1.61	3.33 ± 3.0	3.26 ± 2.45
	Lin LA	1.86 ± 1.0	1.52 ± 0.4	1.55 ± 0.48	1.89 ± 0.65	3.05 ± 1.95	2.68 ± 0.31
	Riem Lin LA	1.9 ± 1.04	1.63 ± 0.6	1.67 ± 0.54	1.93 ± 0.65	3.08 ± 1.42	2.69 ± 1.53
	ENSEMBLES	2.5 ± 0.02	2.14 ± 0.03	2.01 ± 0.02	2.2 ± 0.02	3.76 ± 0.01	3.32 ± 0.02
	HMC	3.3 ± 0.2	0.54 ± 0.21	1.73 ± 0.09	0.36 ± 0.24	2.62 ± 0.42	1.43 ± 0.16
	SGLD	1.38 ± 0.03	1.06 ± 0.07	0.95 ± 0.05	1.11 ± 0.04	2.58 ± 0.03	2.16 ± 0.03
	ESPRO	0.61 ± 0.1	0.33 ± 0.16	0.19 ± 0.22	0.31 ± 0.18	1.66 ± 0.01	1.27 ± 0.03
	SWAG	0.95 ± 0.01	0.62 ± 0.07	0.51 ± 0.08	0.68 ± 0.08	2.16 ± 0.02	1.72 ± 0.01
	MetricBNN	1.66 ± 0.34	1.61 ± 0.29	1.64 ± 0.32	1.76 ± 0.35	1.8 ± 0.36	1.82 ± 0.41
Accuracy	LA GN	0.49 ± 0.12	0.57 ± 0.21	0.17 ± 0.11	0.36 ± 0.09	0.45 ± 0.04	0.57 ± 0.14
	LA LowRank	0.55 ± 0.05	0.60 ± 0.23	0.12 ± 0.06	0.33 ± 0.06	0.46 ± 0.04	0.57 ± 0.05
	Riem LA	0.57 ± 0.04	0.72 ± 0.01	0.38 ± 0.01	0.93 ± 0.02	0.61 ± 0.01	0.86 ± 0.00
	Lin LA	0.56 ± 0.06	0.62 ± 0.08	0.11 ± 0.09	0.91 ± 0.00	0.60 ± 0.01	0.84 ± 0.01
	Riem Lin LA	0.75 ± 0.01	0.68 ± 0.03	0.48 ± 0.18	0.91 ± 0.03	0.58 ± 0.02	0.86 ± 0.01
	ENSEMBLES	0.82 ± 0.02	0.71 ± 0.04	0.72 ± 0.04	0.92 ± 0.02	0.72 ± 0.00	0.83 ± 0.00
	SGLD	0.73 ± 0.03	0.69 ± 0.01	0.52 ± 0.08	0.92 ± 0.02	0.50 ± 0.02	0.64 ± 0.03
	ESPRO	0.82 ± 0.02	0.77 ± 0.01	0.67 ± 0.08	0.96 ± 0.01	0.70 ± 0.00	0.86 ± 0.01
	SWAG	0.57 ± 0.10	0.71 ± 0.04	0.34 ± 0.05	0.92 ± 0.02	0.58 ± 0.01	0.86 ± 0.01
	MetricBNN	0.50 ± 0.04	0.76 ± 0.01	0.57 ± 0.04	0.95 ± 0.01	0.58 ± 0.01	0.85 ± 0.01

E.8.4 Sensitivity of the Hyper-Parameters for MetricBNN

We additionally provide experiments on the sensitivity of the hyper-parameters of MetricBNN. Figures E.8.1 and E.8.2 show the difference in the sampling distribution of MetricBNN when varying the parameters α, N, T. As expected the variance increases with an increase in the number and length of the trajectories as well as the norm of the drift term. Figures E.8.3 and E.8.4 provide the negative loglikelihood for the same varying parameters plus the number of hidden dimensions of the autoencoder k for the posterior distribution.

Table E.8.2: Logarithmic time complexity in seconds and accuracy for the image datasets.

	Model	MNIST	FMNIST	CIFAR10
Log Time	LA GN	3.57 ± 0.02	2.24 ± 0.08	4.79 ± 0.16
	LA LowRank	2.86 ± 0.11	1.67 ± 0.06	3.83 ± 0.02
	Riem LA	5.22 ± 3.16	5.01 ± 4.09	5.31 ± 0.08
	Lin LA	3.64 ± 2.80	3.67 ± 2.96	4.81 ± 0.06
	Riem Lin LA	3.90 ± 3.18	3.91 ± 3.29	5.50 ± 0.09
	ENSEMBLES	4.07 ± 0.04	4.03 ± 0.05	5.35 ± 0.01
	SGLD	2.97 ± 0.05	2.97 ± 0.05	4.17 ± 0.08
	ESPRO	2.12 ± 0.05	2.10 ± 0.05	3.93 ± 0.10
	SWAG	2.51 ± 0.06	2.51 ± 0.06	4.20 ± 0.10
	MetricBNN	3.61 ± 0.05	3.67 ± 0.09	4.22 ± 0.10
Accuracy	LA GN	0.19 ± 0.02	0.19 ± 0.04	0.16 ± 0.04
	LA LowRank	0.77 ± 0.04	0.45 ± 0.03	0.26 ± 0.06
	Riem LA	0.71 ± 0.04	0.38 ± 0.03	0.45 ± 0.05
	Lin LA	0.85 ± 0.03	0.67 ± 0.04	0.52 ± 0.04
	Riem Lin LA	0.92 ± 0.02	0.75 ± 0.06	0.82 ± 0.07
	ENSEMBLES	0.98 ± 0.01	0.87 ± 0.02	0.86 ± 0.02
	SGLD	0.38 ± 0.14	0.25 ± 0.09	0.85 ± 0.10
	ESPRO	0.96 ± 0.02	0.83 ± 0.03	0.86 ± 0.03
	SWAG	0.97 ± 0.02	0.77 ± 0.02	0.16 ± 0.03
	MetricBNN	0.97 ± 0.03	0.82 ± 0.02	0.85 ± 0.40

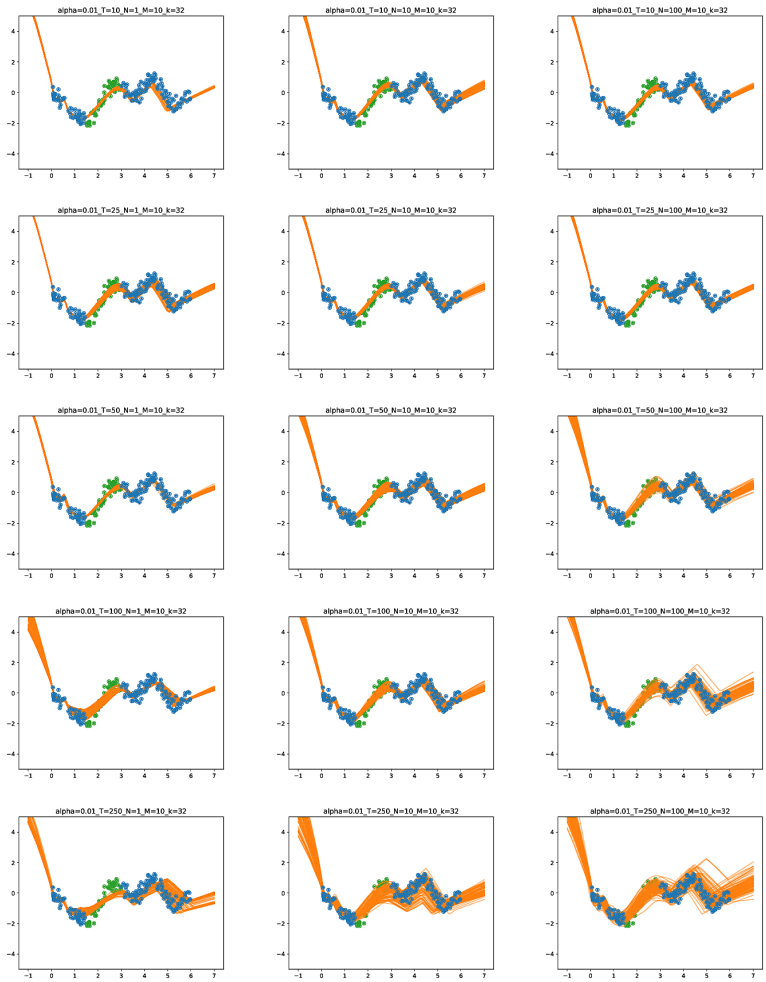


Figure E.8.1: Samples from MetricBNN on the Toy regression problem with varying length of sampling (T) and number of particles (N).

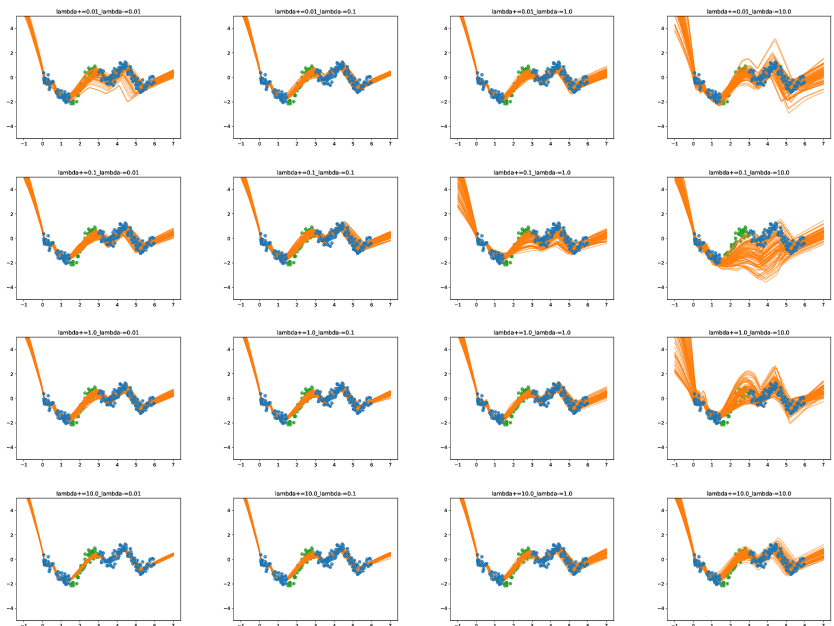


Figure E.8.2: Samples from MetricBNN on the Toy regression problem with varying λ_+ and λ_- for training the autoencoder (λ_d fixed to 1).

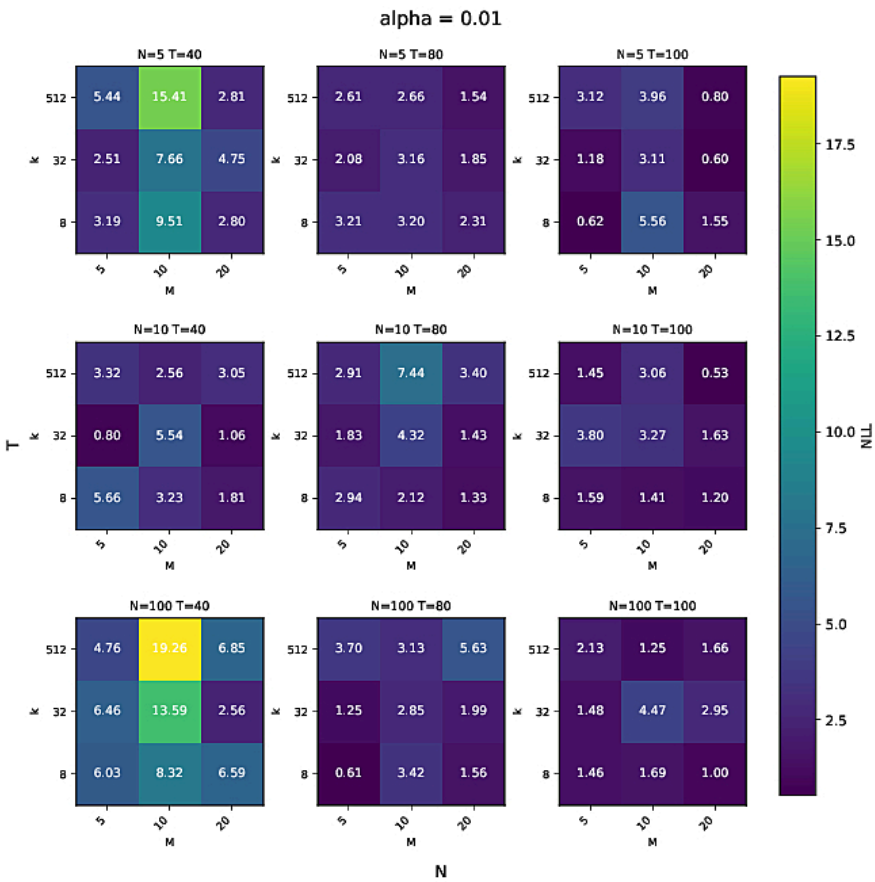


Figure E.8.3: Negative log-likelihood on the Toy regression experiment for MetricBNN with α = 0.01 and different values of N and T (different subplots) and M and k (within each subplot).

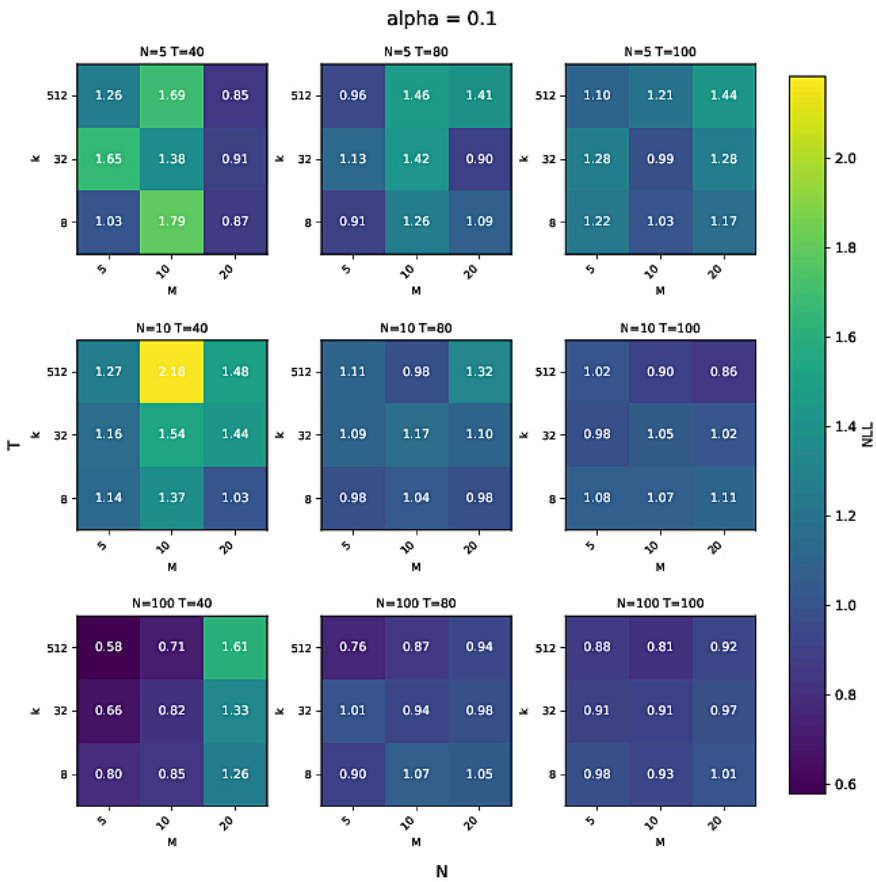


Figure E.8.4: Negative log-likelihood on the Toy regression experiment for MetricBNN with α = 0.1 and different values of N and T (different subplots) and M and k (within each subplot).

E.8.5 Principal Components Analysis on Estimated Posterior

We additionally present the two principal components of the posterior samples of both MetricBNN and HMC for the Banana dataset with varying amounts of steps T in Figure E.8.5. Given enough steps, the presented method covers a large part of the parameter space.

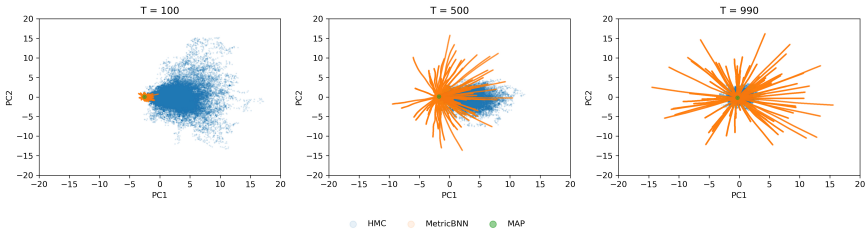


Figure E.8.5

References

[1] Christian Leibig, Vaneeda Allken, Murat Seçkin Ayhan, et al. "Leveraging uncertainty information from deep neural networks for disease detection". In: *Scientific reports* 7.1 (2017), pp. 1–14.

[2] Di Feng, Lars Rosenbaum, and Klaus Dietmayer. "Towards safe autonomous driving: Capture uncertainty in the deep neural network for lidar 3d vehicle detection". In: *2018 21st international conference on intelligent transportation systems (ITSC)*. IEEE. 2018, pp. 3266–3273.

[3] Antonio S Cofino, Rafael Cano, Carmen Sordo, et al. "Bayesian networks for probabilistic weather prediction". In: *15th European Conference on Artificial Intelligence (ECAI)*. Citeseer. 2002.

[4] David JC MacKay. "A practical Bayesian framework for backpropagation networks". In: *Neural computation* 4.3 (1992), pp. 448–472.

[5] Erik Daxberger, Agustinus Kristiadi, Alexander Immer, et al. "Laplace reduxeffortless bayesian deep learning". In: *Advances in neural information processing systems* 34 (2021), pp. 20089–20103.

[6] James Martens. "Deep learning via Hessian-free optimization". In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. 2010, pp. 735–742.

[7] Stanislav Fort and Stanislaw Jastrzebski. "Large scale structure of neural network loss landscapes". In: *Advances in Neural Information Processing Systems* 32 (2019).

[8] Federico Bergamin, Pablo Moreno-Muñoz, Søren Hauberg, et al. "Riemannian Laplace approximations for Bayesian neural networks". In: *Advances in Neural Information Processing Systems* 36 (2024).

[9] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, et al. "Loss surfaces, mode connectivity, and fast ensembling of dnns". In: *Advances in neural information processing systems* 31 (2018).

[10] Alex Graves. "Practical variational inference for neural networks". In: *Advances in neural information processing systems* 24 (2011).

[11] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, et al. "Weight uncertainty in neural network". In: *International conference on machine learning*. PMLR. 2015, pp. 1613–1622.

[12] Cheng Zhang, Judith Bütepage, Hedvig Kjellström, et al. "Advances in variational inference". In: *IEEE transactions on pattern analysis and machine intelligence* 41.8 (2018), pp. 2008–2026.

[13] David JC MacKay. "Choice of basis for Laplace approximation". In: *Machine learning* 33.1 (1998), pp. 77–86.

[14] Max Welling and Yee W Teh. "Bayesian learning via stochastic gradient Langevin dynamics". In: *Proceedings of the 28th international conference on machine learning (ICML-11)*. 2011, pp. 681–688.

[15] Radford M Neal et al. "MCMC using Hamiltonian dynamics". In: *Handbook of markov chain monte carlo* 2.11 (2011), p. 2.

[16] Agustinus Kristiadi, Runa Eschenhagen, and Philipp Hennig. "Posterior refinement improves sample efficiency in bayesian neural networks". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 30333–30346.

[17] Alexander Immer, Maciej Korzepa, and Matthias Bauer. “Improving predictions of Bayesian neural nets via local linearization”. In: *International conference on artificial intelligence and statistics*. PMLR. 2021, pp. 703–711.

[18] Andrew C Miller, Nicholas J Foti, and Ryan P Adams. “Variational boosting: Iteratively refining posterior approximations”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 2420–2429.

[19] Runa Eschenhagen, Erik Daxberger, Philipp Hennig, et al. “Mixtures of Laplace approximations for improved post-hoc uncertainty in deep learning”. In: *arXiv preprint arXiv:2111.03577* (2021).

[20] Marton Havasi, Jasper Snoek, Dustin Tran, et al. “Sampling the variational posterior with local refinement”. In: *Entropy* 23.11 (2021), p. 1475.

[21] Andrew G Wilson and Pavel Izmailov. “Bayesian deep learning and a probabilistic perspective of generalization”. In: *Advances in neural information processing systems* 33 (2020), pp. 4697–4708.

[22] Johanni Brea, Berfin Simsek, Bernd Illing, et al. “Weight-space symmetry in deep networks gives rise to permutation saddles, connected by equal-loss valleys across the loss landscape”. In: *arXiv preprint arXiv:1907.02911* (2019).

[23] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. “Simple and scalable predictive uncertainty estimation using deep ensembles”. In: *Advances in neural information processing systems* 30 (2017).

[24] Wesley J Maddox, Pavel Izmailov, Timur Garipov, et al. “A simple baseline for bayesian uncertainty in deep learning”. In: *Advances in neural information processing systems* 32 (2019).

[25] Gregory Benton, Wesley Maddox, Sanae Lotfi, et al. “Loss surface simplexes for mode connecting volumes and fast ensembling”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 769–779.

[26] Edward Snelson and Zoubin Ghahramani. “Sparse Gaussian processes using pseudo-inputs”. In: *Advances in neural information processing systems* 18 (2005).

[27] K Markelle, L Rachel, and N Kolby. *UCI Dataset. The UCI Machine Learning Repository*. 2023.

[28] Yann LeCun. “The MNIST database of handwritten digits”. In: *http://yann. lecun. com/exdb/mnist/* (1998).

[29] Han Xiao, Kashif Rasul, and Roland Vollgraf. “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms”. In: *arXiv preprint arXiv:1708.07747* (2017).

[30] Alex Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. Technical Report TR-2009. Toronto, Ontario: University of Toronto, 2009. url: `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`.

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[32] Erik Daxberger, Eric Nalisnick, James U Allingham, et al. “Bayesian deep learning via subnetwork inference”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 2510–2521.

[33] Dan Hendrycks and Thomas Dietterich. “Benchmarking neural network robustness to common corruptions and perturbations”. In: *arXiv preprint arXiv:1903.12261* (2019).

[34] Hippolyt Ritter, Aleksandar Botev, and David Barber. “A scalable laplace approximation for neural networks”. In: *6th international conference on learning representations, ICLR 2018-conference track proceedings*. Vol. 6. International Conference on Representation Learning. 2018.

[35] Neil David Lawrence. “Variational inference in probabilistic models”. PhD thesis. Citeseer, 2001.

Paper F

Paper F

Reducing Variance in Meta-Learning via Laplace Approximation for Regression Tasks

ALFREDO REICHLIN*, GUSTAF TEGNÉR*, MIGUEL VASCO, HANG YIN, MÅRTEN BJÖRKMAN AND DANICA KRAGIC

Published in
Transactions on Machine Learning Research (TMLR), 2024

Abstract:
Given a finite set of sample points, meta-learning algorithms aim to learn an optimal adaptation strategy for new, unseen tasks. Often, this data can be ambiguous as it might belong to different tasks concurrently. This is particularly the case in meta-regression tasks. In such cases, the estimated adaptation strategy is subject to high variance due to the limited amount of support data for each task, which often leads to sub-optimal generalization performance. In this work, we address the problem of variance reduction in gradient-based meta-learning and formalize the class of problems prone to this, a condition we refer to as *task overlap*. Specifically, we propose a novel approach that reduces the variance of the gradient estimate by weighing each support point individually by the variance of its posterior over the parameters. To estimate the posterior, we utilize the Laplace approximation, which allows us to express the variance in terms of the curvature of the loss landscape of our meta-learner. Experimental results demonstrate the effectiveness of the proposed method and highlight the importance of variance reduction in meta-learning.

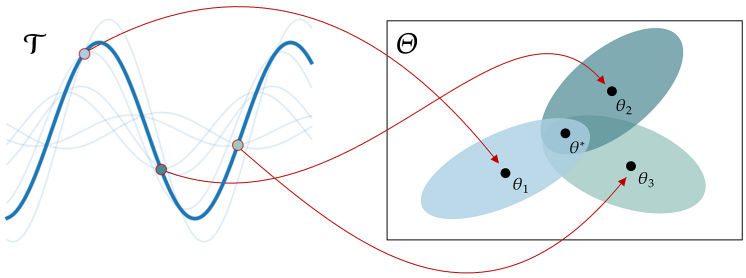


Figure F.1.1: The Problem of Task Overlap in Regression Tasks: *On the left*: Three support points of a single task are marked out. These points are shared between different tasks which are marked out by the opaque curves. *On the right:* Each support point induces a distribution in the parameter space. The true function parameters θ^* lie at the *intersection* over the possible function values.

F.1 Introduction

Meta-learning, also known as learning-to-learn, is concerned with the development of intelligent agents capable of adapting to changing conditions in the environment. The central idea of meta-learning is to learn a prior over a distribution of similar learning tasks, enabling fast adaptation to novel tasks given only a limited number of data points. This approach has proven successful in many domains, such as few-shot learning [1], image completion [2], and imitation learning tasks [3].

One instance of this is Gradient-Based Meta-Learning (GBML), which was introduced in [4]. GBML methods employ a bi-level optimization procedure, where the learner first adapts its parameters given a few samples denoted the *support-set*, which it then evaluates on another, more complete, *query-set*. The shared prior is defined as the parameter initialization. A central issue arises in identifying accurate parameters from the support set. The limited amount of support points used for adaptation, measurement error, or ambiguity between the tasks inevitably leads to uncertainty in this parameter identification. To enable more efficient adaptation, more effective priors must be learned.

When learning over a continuous space of tasks in the context of meta-regression, ambiguity often arises. Consider the problem of regressing sinusoidal waves with varying amplitudes and phases as depicted in Fig. F.1.1 on the left. Each pair of (x, y) used to adapt to the specific wave can correspond to infinite other waves and, as such, doesn't uniquely identify the task at hand. We refer to this condition as *task overlap*. In this case, the aggregation of the information conveyed by each data point in the adaptation dataset has to consider this uncertainty. GBML models struggle in this regard as this uncertainty is not taken into account.

In this work, we propose **L**aplace **A**pproximation for **V**ariance-reduced **A**daptation **(LAVA)**, a method that introduces a novel strategy for aggregating the information of the support-set in the adaptation process. The key idea of our method is to recognize each support point as inducing a unique posterior distribution over the task

parameters (Fig. F.1.1 on the right). The complete posterior distribution can then be expressed as the joint posterior over the samples. Thus, our model is a form of Bayesian model-averaging [5], where each model is induced by a single point from the support. We approximate the posterior distribution w.r.t each support point through *Laplace's approximation*, which constructs a Gaussian distribution where the variance of the estimator is a function of the Hessian of the negative log-likelihood [6, Chapter 27]. In turn, the joint posterior is again Gaussian, from which the optimal value can be expressed as its mean. In contrast to other Bayesian GBML methods, the posterior approximation is not built on the full posterior but rather on the single posteriors induced by every point in the support data. This allows us to optimally aggregate the information these points share and reduce the variance of this estimate in the case of ambiguity.

Our contributions include an insight into the adaptation process of GBML methods by identifying a class of continuous regression problems where GBML methods are particularly subject to high variance in their adaptation procedure. We introduce a method for modeling the variance that each data point carries over the parameters and optimally aggregate these posterior distributions into the task-adapted parameters. Finally, we demonstrate the performance of our method on regression of dynamical systems and real-world experiments and showcase state-of-the-art performance compared to standard GBML methods.

F.2 Preliminaries

F.2.1 Problem Formulation

Let $p(\mathcal{T})$ be a distribution of tasks that share a common generative process. Each task $\tau \in \mathcal{T}$ defines a function $f_\tau : \mathcal{X} \to \mathcal{Y}$ represented as subsets $\tau \in \mathcal{X} \times \mathcal{Y}$ of inputs $\mathcal{X} \subseteq \mathbb{R}^d$ and outputs $\mathcal{Y} \subseteq \mathbb{R}^k$. For each task, assume we are given a finite dataset sampled i.i.d. i.e., $D_\tau \subseteq \mathcal{X} \times \mathcal{Y}$. These points can be further divided into two sets, the support set D_τ^S and the query set D_τ^Q such that $D_\tau = D_\tau^S \cup D_\tau^Q$. We denote $N = |D_\tau^S|$, $M = |D_\tau^Q|$ as the size of the support and query set respectively, which we assume, for simplicity, is the same across all tasks.

From the support set, D_τ^S, we are interested in estimating the underlying function defined by each task, f_τ. Following previous work, [7, 8], this can be described as estimating a map from the support data to a set of parameters that can be used to approximate the true task's function i.e., $\mathcal{A} : \mathcal{X} \times \mathcal{Y} \to \Theta$ with $\mathcal{A}(D_\tau^S) = \theta_\tau$ such that $f_{\theta_\tau} \approx f_\tau$. We define similarity in terms of mean-squared error and compute it using the M points of the query set with the following loss function:

$$\mathcal{L}(\theta_\tau, D_\tau^Q) = \frac{1}{M} \sum_{i=1}^{M} \left\| f_{\theta_\tau}(x_\tau^i) - y_\tau^i \right\|_2^2, \quad \text{s.t.} \quad \theta_\tau = \mathcal{A}(D_\tau^S). \tag{F.2.1}$$

The performance is tightly bound to the error in the estimator $\mathcal{A}$ which in part arises from uncertainty in the support data. In this work, we consider the uncertainty that

is induced when data points can belong to multiple tasks, which we denote as *task overlap*.

For ease of notation, let $f_\tau(x) = f(\tau, x)$ denote the true data-generating process.
Definition 5 (Task Overlap)**.** *We define task overlap as the condition for which* $\forall x \in \mathcal{X}$ *the map* $f(\cdot, x) : \mathcal{T} \to \mathcal{Y}$ *is non-injective.*

From a probabilistic perspective, this property is equivalent in stating that the conditional task-distribution $p(\tau \mid x, y)$ will have a non-zero covariance for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$.

An example of *task overlap* can be seen by considering a sine wave regression problem. Let the distribution of tasks be sinusoidal waves with changing amplitude and phase i.e., $y = A_\tau \sin(x + \phi_\tau)$. In this case, the space of tasks is defined by the union of possible amplitudes and phases and thus has dimension 2. A single sample in the form of a tuple (x, y) is, however, insufficient to identify this task unambiguously. In fact, there exists an infinite number of viable amplitudes and phases for which the sine can pass through such a point, Fig. F.1.1 on the left. In the terms above, there is no injective map between (x, y) and (A_τ, ϕ_τ). The set of all possible solutions induced by this single point defines a distribution of solutions in the task-adapted parameter's space. All of these solutions are equally probable if conditioned on this single point. When multiple (and different) samples are considered for the adaptation step, the inference of the sine wave can be exact. There exists a unique sine wave that passes through all of these points at the same time. In terms of task parameters, this can be seen as considering the intersection of the solutions induced by each point; see Fig. F.1.1 on the right. When using a single-point estimate, the information about this distribution of possible optimal task parameters is lost.

F.2.2 Gradient-Based Meta-Learning

A notable family of methods to solve the problem described in (F.2.1) are GBML algorithms. In these, learning the adaptation process is formulated as a bi-level optimization procedure. A set of meta-parameters θ_0 is learned such that the inference process $\mathcal{A}$ corresponds to a single gradient descent step on the loss computed using the support data and θ_0:

$$\theta_\tau = \theta_0 - \alpha \nabla_\theta \mathcal{L}(\theta, D_\tau^S)\big|_{\theta=\theta_0}, \tag{F.2.2}$$

where α is a scalar value denoting the learning rate. The resulting estimate of the task-adapted parameters is then optimized to approximate the underlying task function using (F.2.1). This, effectively, defines an overall optimization procedure on the meta-parameters θ_0. Gradient-based Meta-Learning does not require additional parameters for the adaption procedure, as it operates in the same parameter space as the learner. Moreover, it is proven to be a universal function approximator [9], making it one of the most common models for meta-learning.

As noted in previous work [10], the bi-level formulation of GBML can be interpreted from a Bayesian view. We are interested, for each task, in finding a set of adapted

parameters that maximize the likelihood of the query data i.e., $\max p(D_\tau^Q \mid \theta_\tau)$. This is achieved by estimating a prior in the form of θ_0 such that, when combined with the evidence from the support data, it leads to the highest probability estimate. GBML simplifies the process by defining the estimate of the adapted parameters as the solution to (F.2.2). This, in fact, is equivalent to estimating the most probable set of parameters given the prior θ_0 and the support data D_τ^S:

$$\hat{\theta}_\tau = \arg\max_\theta p(\theta \mid \theta_0, D_\tau^S). \tag{F.2.3}$$

A full derivation can be found in Section F.7.2. The quality of the estimate of the adapted parameters is, however, limited by the finiteness of the support data. In fact, while it is unbiased (Section F.7.5), it may suffer from high variance.

F.2.3 Variance Reduction

In this section we discuss the variance problem related to meta-learning and techniques to reduce it.

Assuming the support set is composed of N i.i.d. samples from the task, the variance of the estimated adapted parameters can be expressed as follows:

$$\mathrm{Var}\left[\hat{\theta}\right] = \frac{1}{N^2}\sum_{i=1}^{N}\mathrm{Var}\left[\hat{\theta}_i\right], \tag{F.2.4}$$

which in the case of task overlap is non-zero. In Fig. F.5.1 we depict the variance of this estimate for a simple sinusoidal regression task, together with a comparison to our proposed variance-reduced estimation described in Section F.3. As can be seen, for standard GBML, the variance of the estimated parameters cannot get below the variance induced by the finite sampling of the support data. On the other hand, LAVA's estimation has a decreasing variance as training progresses.

To address the problem of high variance, we leverage upon a general method for reducing the variance of a sum of random variables [11]. Let $\theta_1, \dots, \theta_N$ be a set of random variables with the same mean θ^* and different covariance Σ_i for each $i = 1, \dots, N$. Let $\hat{\theta} = \sum_{i=1}^N W_i\theta_i$ denote a weighted average with weights $W_i \in \mathbb{R}^{d\times d}$. We wish to estimate W_i^* that yield a $\hat{\theta}$ with the minimum variance. We can define this variance reduction problem as:

$$\min_W \quad \mathrm{Var}\left[\sum_{i=1}^N W_i\theta_i\right], \quad \textit{subject to} \quad \sum_{i=1}^N W_i = I. \tag{F.2.5}$$

Proposition 1 (Variance Reduction)**.** *The solution to* (F.2.5) *is given by:*

$$W_i^* = \left[\sum_{i=1}^N \Sigma_i^{-1}\right]^{-1}\Sigma_i^{-1}. \tag{F.2.6}$$

We provide a proof in Section F.7.6 for completeness.

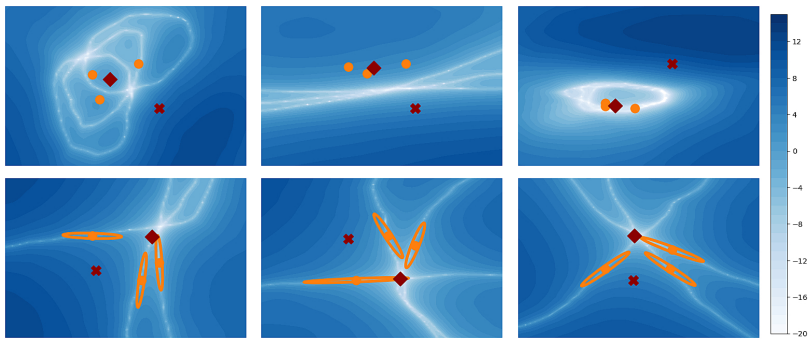


Figure F.3.1: The space of task parameters adapts to the Hessian. The sum of the logarithm of the loss for 3 support points over different parameters for the sine experiment. Values increase from white to dark blue. The red cross and the red diamond indicate the prior and the posterior, orange points are the single task-adapted parameters. *Top row:* Results for CAVIA. *Bottom row:* Results for LAVA, included is also the covariance for each support point.

F.3 Method

In this section, we describe a novel method to model the estimation of the task-adapted parameters more accurately in the task overlap regime. Given a finite set of support points D^S, we want to approximate the optimal posterior of (F.2.3). Let $p(\theta|x_i, y_i)$ denote the posterior w.r.t to one data point. Given the conditions of task overlap, (Definition 5), this induces a distribution in parameter space. Another way of interpreting this posterior is that each data point provides evidence of what the possible true function could be. The full posterior $p(\theta|D^S)$ will thus lie at the *intersection* of all marginal posteriors. Given the i.i.d. assumption, this implies that:

$$p\left(\theta|D^S\right) \propto \prod_{i=1}^{N} p(\theta|x_i, y_i). \tag{F.3.1}$$

A derivation can be found in Section F.7.4. We propose to model the probability of the adapted parameters given each support point as a Gaussian distribution, $\mathcal{N}(\hat{\theta}_i, \Sigma_i)$, by means of Laplace's approximation [12]. In particular, we define the maximum a posteriori (MAP) estimate to be the GBML original formulation i.e., $\hat{\theta}_i = \theta_0 - \alpha \nabla_{\theta_0} \mathcal{L}(\theta_0, D_i^S)$. The variance of the adaptation can then be defined as the inverse of the Hessian of the loss function evaluated in the adapted parameters i.e., $\Sigma_i = H_i^{-1}$. This approximation allows us to rewrite the estimate of the overall adapted parameters in (F.3.1) as the product of these Gaussians. The most probable set of parameters given the provided support data will thus be the mean of the resulting Gaussian distribution:

$$\hat{\theta} = \left(\sum_{i=1}^{N} H_i\right)^{-1} \sum_{i=1}^{N} H_i \hat{\theta}_i. \tag{F.3.2}$$

One interpretation of GBML methods such as MAML [9] is that the posterior estimate assumes an equal covariance across all marginal posteriors, which in turn implies that

the MAP estimate equals the average of the parameters. In fact, in the case of no task overlap, (F.3.2) becomes equivalent to (F.2.2). This follows from the fact that we perform *a single gradient step* to approximate $\arg\max_\theta p(\theta|D^S)$. Due to the linearity of the gradient operator, $\arg\max_\theta p(\theta|D^S)$ equals the average of $\{\arg\max_\theta p(\theta|x_i, y_i)\}_{i=1}^N$. However, in the case of anisotropic uncertainty in the adapted parameters, as in the task overlap regime, this leads to a sub-optimal estimation. On the other hand, when the Laplace approximation faithfully describes the underlying distribution, the estimate proposed in (F.3.2) corresponds to the minimum variance estimator. This can be seen from Proposition 1, where $\Sigma_i = H_i^{-1}$. The final assumption to make the results coincide is that the expectation over the adapted parameters is the same, i.e. $\mathbb{E}_{p(\tau)}[\widehat{\theta}_i] = \theta^*$ for all $i = 1, \dots, N$. This θ^* can be seen as the optimal for the given task but is not necessary for the proof.

Compared to simple parameter averaging defined in (F.2.4), we achieve a lower variance for any support set $D^S \sim p(\tau)$. Thus, on expectation over all tasks, we achieve a variance-reduced estimate.

The overall training procedure follows from GBML methods. We optimize the set of parameters θ_0 through the objective of (F.2.1) as a bi-level optimization procedure where θ_τ is now defined according to (F.3.2). Differently from GBML, the Laplace approximation allows for reshaping of the parameter space of the learned model. Embedding this new adaptation process in the optimization allows for a precise approximation. To minimize the loss, the model has to shape the parameter space such that, locally, the Laplace approximation can exactly estimate the posterior. This imposes certain constraints on the structure of the parameter space arising from task overlap. This learned parameter space can be precisely shaped to allow for an accurate Laplace Approximation due to the flexible nature of parametric neural networks. Consider once again the sine wave regression example. Each support point gives evidence for a space of possible solutions. When considering a model with a parameter space of dimension 2, the space of solutions in the adapted parameters is a one-dimensional curve. The optimal posterior corresponds to an intersection of the curves induced by each support point. In contrast to other GBML methods, LAVA allows for straightening out the region of parameters that minimize the inner loss function. This in turn allows for shaping of the parameter space that enables a precise Laplace Approximation. We illustrate this in Fig. F.3.1.

F.3.1 Computational Implementation

A limitation of the described method lies in an increased time complexity. Computing the Hessian on each single support point can severely affect the training time of methods that already require complex second-order calculations like in GBML, especially for over-parameterized neural networks. To minimize the computational burden, we consider performing adaptation in a subspace of the parameter space. In most of the experiments, we opt for the implementation of ANIL [13], which performs adaptation over the last layer of a neural network as they argue that most of the adaptation takes place in the final layers. This allows us to compute the Hessian in closed form, drasti-

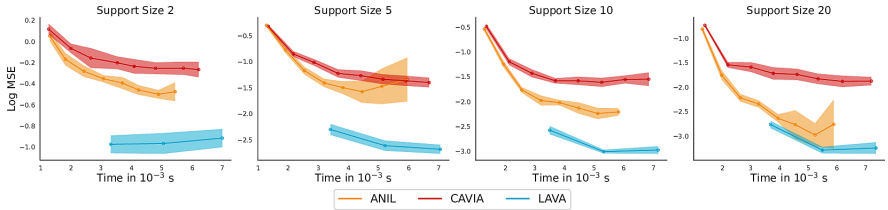


Figure F.3.2: Evaluation of Computation Time: We evaluate the performance of the different methods (measured in terms of MSE) as a function of the computational time (in seconds) across different support sizes. The results show that, considering the same execution time (x-axis), LAVA outperforms both CAVIA and ANIL, with lower MSE. This result holds across all evaluated support sizes.

cally reducing the computational cost and keeping it independent of the dimensionality of the input space.

Nevertheless, the Hessian computation can result in an increase in computational time and LAVA is, in fact, more expensive than the standard GBML model. However, this computational complexity increase is paired with stronger performances. LAVA provides a more effective adaptation technique as one of its main features is the efficient use of the limited information given by the support. In this regard, LAVA provides a better trade-off between performances and computational complexity. To better analyze this trade-off we compared LAVA's performances against two GBML baselines in the Sine regression experiment by varying the number of inner loop adaptation steps. As shown in Fig. F.3.2, LAVA has a computational complexity comparable to CAVIA [14] with 2 inner loop gradient steps and ANIL with 3. In Section F.7.8 we provide a description of how to compute the Hessian when the loss is in the form of (F.2.1).

Computing second-order derivatives, however, is known to be numerically unstable for neural networks, [15]. We found that a simple regularization considerably stabilizes the training. Following [16], we take a weighted average between the computed Hessian and an identity matrix before aggregating the posteriors. For all of our experiments, we substitute each Hessian H_i in (F.3.2) with the following:

$$\tilde{H}_i = \frac{1}{1+\epsilon}\left(H_i + \epsilon I\right), \tag{F.3.3}$$

where ϵ is a scalar value and I is the identity matrix of the same dimensionality of H_i. In our experiments, we consider $\epsilon = 0.1$.

F.4 Related Work

Gradient-Based Meta-Learning methods were first introduced with MAML [4] and then expanded into many variants. Among these, Meta-SGD [17] includes the learning rate as a meta parameter to modulate the adaptation process, Reptile [18] gets rid of the inner gradient and approximates the gradient descent step to the first-order. In CAVIA [14], the adaptation is performed over a set of conditioning parameters of the

base learner rather than on the entire parameter space of the network. Other works instead make use of a meta-learned preconditioning matrix in various forms to improve the expressivity of the inner optimization step [19, 20, 21].

Bayesian Meta-Learning formulates meta-learning as learning a prior over model parameters. Most of the work in this direction is concerned with the approximation of the intractable integral resulting from the marginalization of the task parameters. This has been attempted using a second-order Laplace approximation of the distribution [10], variational methods [22], MCMC methods [23], or Diffusion models [24]. While these Bayesian models can provide a better trade-off between the posterior distribution of the task-adapted parameters and the likelihood of the data [25], they require approximating the full posterior and marginalizing over it. LLAMA [10] proposes to approximate the integral around the MAP estimate through the Laplace Approximation on integrals. Given one gradient step, the posterior estimate is still performed by averaging. As we have shown, the averaging fails to account for inter-dependencies between the support points arising from task overlap.

Model-based Meta-Learning relies on using another model for learning the adaptation. One such approach is HyperNetwork [26], which learns a separate model to directly map the entire support data to the task-adapted parameters of the base learner. In [27], this is implemented using amortized inference, while in [28], the task-adapted parameters are context to the base learner. Alternatively, the HyperNetwork can be used to define a distribution of candidate functions using the few-shot adaptation data [2, 29] and additionally extend it using an attention module [30]. Lastly, memory modules can be iteratively used to store information about similar seen tasks [31] or to define a new optimization process for task-adapted parameters [32]. All of these methods can potentially solve the aggregation of information problem implicitly as the support data are processed concurrently. However, the learned model is not model-agnostic and introduces additional parameters.

F.5 Experiments

To begin with, we test the validity of using the Laplace approximation to compute the task-adapted parameters for a simple sine regression problem. Additionally, we show how LAVA exhibits a much lower variance in the posterior estimation in comparison to standard GBML. In Section F.7.5, we demonstrate the unbiasedness of GBML and evaluate the noise robustness of GBML and LAVA against a model-based approach.

We further evaluate our proposed model on dynamical systems tasks of varying complexity in regard to the family of functions and dimensionality of the task space as well as regression of two real-world datasets. We compare the results of our model against other GBML models. In particular, our baselines include ANIL [13], CAVIA, [14] as a context-based GBML method, LLAMA [10], VFML and VR-MAML [33, 34] as variance-reduced meta-learning methods, and MetaMix [35] as a meta-data augmented method. Experimental details are given in the Section F.7.1.

F.5.1 Sine Wave Regression

To develop a further intuition of our model, we conduct experiments on the sine-wave regression problem introduced in [9]. We investigate two aspects of our model. The first is in regards to the geometry of the learned parameters space (Section F.7.7) and how well the Laplace approximation can accurately capture distributions in parameter space. The other aspect we investigate is measuring the variance reduction in our estimate. In the sine experiment, the dimensionality of the true task parameters is two, allowing us to visualize the learned parameter space. To this end, we train both LAVA and GBML with a conditional model (akin to CAVIA [14]) on the sine-wave regression problem with a context vector of dimension two.

Given a single (x, y) tuple, there exists a one-dimensional space of sine waves that pass through that point. This makes the aggregation challenging and thus allows us to test the benefits of approximating this subspace with the Laplace Approximation.

Laplace Approximation Assumption The first ablation aims at testing the quality of the Laplace approximation in modeling the distribution of the task parameters given each single data point. After the models have converged, we visualize the loss landscape induced by different support data points when using different task-adapted parameters. In particular, we sample a support dataset of three (x, y) tuples and compute the logarithm of the mean squared error between the prediction of both models and the true y sampled from a grid of tasks. The idea is that each data point's loss is minimized by a continuous space of parameters. When summing up the losses of these support points, in fact, very well-defined valleys can be noticed in the loss landscape. We illustrate the distribution of this sum of losses for a grid of task-adapted parameters in Fig. F.3.1 as a heat map for CAVIA (top row) and LAVA (bottom row). Additionally, the prior context (red cross), the single task adapted parameters (orange dots) and the aggregated final posterior (red diamond) are also shown. For our method, we provide a visualization of the Hessian matrix for each single task-adapted parameter.

Variance Estimation For the second ablation, we evaluate the variance of the posterior estimate for both CAVIA and LAVA using the sine regression framework. The variance of the estimator describes the difference between the task-adapted parameters from the sampling of different support sets from the same task. For this experiment, we fix the sine task and sample 100 different support sets each with 10 (x, y) tuples. For each of these support points, we compute the resulting task-adapted parameters. We depict the spread of these distributions in Fig. F.5.1 (left and center) for CAVIA and LAVA respectively. Note that the scale of the parameter space as well as the inner learning rate is the same for the two methods. The right of Fig. F.5.1 illustrates the log variance of the task-adapted parameters during training for different support sizes. As can be seen in the figure, at the beginning of training, the variance increases suddenly as the models learn to use these parameters to solve the meta-learning problem. However, while CAVIA's context parameters variance remains high during the rest of the training, our method learns to reduce it consistently.

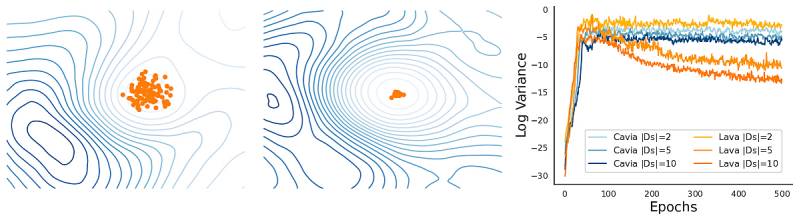


Figure F.5.1: Estimator's variance. Variance of the task-adapted parameters given the same task but different support data points. *Left:* For CAVIA. *Center:* For LAVA. *Right:* Log variance of the distribution of the adapted parameters during training.

F.5.2 Differential Equations

We evaluate further on a set of complex regression tasks in the form of dynamical systems prediction. We consider 5 sets of Ordinary Differential Equations (ODEs):

- **FitzHugh-Nagumo**: A model for excitable systems, e.q. modeling the spike of a neuron.
- **Mass-Spring System**: A classic mass-spring dynamics model for prediction of the effects of gravity upon a mass connected to a spring.
- **Pendulum**: Describes the motion of an object of mass m connected to a rod of length L suspended from a pivot.
- **Van Der Pol Oscillator**: An oscillating system that undergoes non-linear damping, determined by the parameter μ.
- **Cartpole**: An inverse dynamics problem of an actuated cartpole with varying mass.

For each system, we consider samples from its vector field as our target data. We train all models for 300 epochs and compare their MSE on reconstructing the vector field, we present the results in Table F.5.1. For CAVIA and ANIL, we perform a grid search over up to 8 inner optimization steps and select the best-performing one. For a single gradient step, LAVA outperforms both CAVIA and ANIL even with a large number of adaptation steps. This increase in performance can be attributed to LAVA's use of second-order information through the curvature of the loss landscape. Further details about the dataset construction and choice of parameters are given in Section F.7.1.

To further evaluate the top-performing models, we visualize roll-out trajectories devised from integrating the learned vector field in Fig. F.5.2. We can note that LAVA provides strong roll-out prediction in almost all cases.

F.5.3 Real World Datasets

In the next experiment, we evaluate our model on two real-world datasets. The first is a regression task on the Beijing Air Quality Dataset [36] while in the second we con-

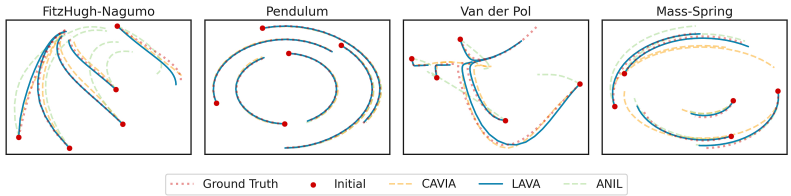


Figure F.5.2: ODEs qualitative results. Rollout Trajectories for the dynamical systems' prediction with CAVIA, LAVA and ANIL. We consider the dynamics of systems with random parameters for 5 initial conditions. LAVA (blue line) is the only model that consistently predicts the evolution of the system (red dotted line).

Table F.5.1: MSE $\downarrow$ by Dynamical System and Model for Support Size 10, including Cartpole results

Model Name	FitzHugh -Nagumo	Mass-Spring $(\times 10^{-2})$	Pendulum $(\times 10^{-2})$	Van der Pol	Cartpole
CAVIA	0.19 ± 0.05	1.00 ± 1.36	0.34 ± 0.11	3.57 ± 1.18	1.14 ± 0.07
ANIL	0.64 ± 0.25	0.09 ± 0.04	0.22 ± 0.07	12.29 ± 6.84	1.81 ± 0.57
LLAMA	1.18 ± 0.57	0.51 ± 0.05	3.29 ± 0.37	6.78 ± 1.10	1.77 ± 0.32
VFML	1.27 ± 0.25	0.53 ± 0.07	4.62 ± 0.81	6.36 ± 1.34	1.86 ± 0.62
Metamix	3.56 ± 1.21	0.65 ± 0.11	3.68 ± 0.85	46.01 ± 8.29	4.49 ± 0.15
VR	3.88 ± 0.95	3.28 ± 0.28	21.55 ± 8.52	32.26 ± 6.42	2.34 ± 0.46
LAVA	$\mathbf{0.17 \pm 0.05}$	$\mathbf{0.02 \pm 0.00}$	$\mathbf{0.07 \pm 0.01}$	$\mathbf{1.50 \pm 0.87}$	$\mathbf{1.02 \pm 0.25}$

sider regressing a real-world and synthetic Frequency-modulated radio signal subject to noise in the RadioML 2018.01A dataset [37].

Air Quality: We consider the Beijing Air Quality Dataset [36] which is a time-series dataset containing recordings of air quality across 12 monitoring sites. The dataset contains hourly measurements during the time period from 01-03-2014 to 28-2-2017. We consider each monitoring site a separate task. Each task can be viewed as an interpolation task. A random contiguous subsequence of size $n_s + n_q$ is selected which is then split randomly into a support- and query-set. For each measurement, a time-variable t is appended, indicating a positional embedding in the time-series. The task is then predicting the air-quality measurement from the given time t. We present the MSE in the left-most column of Table F.5.2. In this task, LAVA clearly outperform the baselines.

RadioML: Lastly, we evaluate the performances of LAVA compared with the baselines on regressing frequency-modulated radio signals. To do so, we make use of the RadioML 2018.01A dataset described in [37]. These are both synthetic and real-world radio signals recorded over-the-air and subject to noise. We devise the task as being able to regress each signal given a few data point recordings. For testing, we withhold 25% of the signals. We present the MSE in Table F.5.2 on the right-most column. Regression of these signals poses a considerable challenge for meta-learning methods.

The change in frequency between signals exacerbates the task overlap conditions of the task. The results highlight this, as LAVA is the only method able to correctly regress the signals.

Table F.5.2: MSE ↓ of model-predictions on the Air-Quality dataset and RadioML dataset.

Model Name	Air-Quality ($\times 10^{-2}$)	RadioML
ANIL	7.63 ± 0.31	0.047 ± 0.0139
CAVIA	8.25 ± 0.16	0.5016 ± 0.0
VFML	9.21 ± 0.46	0.156 ± 0.1351
LLAMA	15.37 ± 7.36	0.5017 ± 0.0
Metamix	56.76 ± 13.52	0.0471 ± 0.0018
VR	NaN	0.491 ± 0.0081
LAVA	$\mathbf{4.65 \pm 0.30}$	$\mathbf{0.0038 \pm 0.0002}$

F.6 Discussion and Conclusion

In this paper, we characterized the problem of *task overlap* for under-determined inference frameworks. We have shown how this is a cause of high variance in the posterior parameters estimate for GBML models. In this regard, we have proposed LAVA, a novel method to address this issue that generalizes the original formulation of the adaptation step in GBML. In particular, the task-adapted parameters are reformulated as the average of the gradient step of each single support point weighted by the inverse of the Hessian of the negative log-likelihood. This formulation follows from the Laplace approximation of every single posterior given by each support data point, resulting in the posterior being the mean of a product of Gaussian distributions. Empirically we have shown how our proposed adaptation process suffers from a much lower variance and overall increased performance for a number of experiments.

For regression tasks, the assumption of task overlap is of particular interest. On the other hand, classification tasks are inherently discrete and do not suffer from the problem of task overlap to the same extent as regression-like problems. Nonetheless, for completion, we provide in Section F.7.10 the results of a few-shot classification experiment on *mini-Imagenet*. In this experiment, LAVA performs similarly to the other GBML baselines, confirming the different nature of the overall problem. However, the discrete nature of classification problems presents an avenue for future work in the possibility of incorporating adaptation over a categorical distribution of parameters. A second limitation is the computational burden of computing the Hessian. As described in Section F.3.1, and further discussed in Section F.7.8, we overcome this limitation by restricting the adapted parameters to the last layer only (akin to ANIL [13]). This allows us to compute the Hessian in closed form. We show that the time complexity of our model is comparable to 2 inner steps of CAVIA and 3 inner steps of ANIL while showcasing superior performance.

An interesting extension to the proposed method would be to explore techniques to approximate the Hessian. This would allow us to extend the adapted parameters to the whole model. Possible approximations could include the Fisher information matrix or the Kronecker-factored approximate curvature [38] to estimate the covariance of the Laplace approximation. Alternatively, it might be interesting to explore the direction of fully learning this covariance by following an approach similar to model-based methods.

Acknoledgments

This work was supported by the Swedish Research Council, the Knut and Alice Wallenberg Foundation, the European Research Council (ERC-BIRD-884807) and the European Horizon 2020 CANOPIES project. Hang Yin would like to acknowledge the support by the Pioneer Centre for AI, DNRF grant number P1.

F.7 Appendix

F.7.1 Experimental Details

For all of the experiments and all the baselines, we fix the architecture of the meta-learner f_θ to be a multi-layer perceptron with 3 hidden layers of 64 hidden units together with ReLU activations. We use a meta batch size of 10 tasks and train all the models with Adam [39] optimizer with a learning rate of 10^{-3}. We use the inner learning rate $\alpha = 0.1$ for the adaptation step and MSE as the adaptation loss. All experiments were run for 5 different seeds to compute mean and standard deviations. For LLAMA we use $\eta = 10^{-6}$, for PLATIPUS we scale the KL loss by 0.1, for BMAML we use 10 particles and use MSE rather than the chaser loss for a fair comparison. Other experiment-specific details include:

Differential Equations

- **FitzHugh-Nagumo**:

$$\frac{du}{dt} = c(u - u^3/3 + v),$$
$$\frac{dv}{dt} = -\frac{1}{c}(u - a + bv),$$

  with $a, b, c \sim \mathcal{U}[0.1, 2.0]$. The initial positions are sampled as $u, v \in \mathcal{U}[-2.5, 2.5]$.

- **Mass-Spring**:

$$\frac{dx}{dt} = -\frac{\dot{x}}{m},$$
$$\frac{d\dot{x}}{dt} = -kx,$$

  with mass m and spring constant k sampled from $\mathcal{U}[0.5, 1.5]$ with initial conditions $x, \dot{x} \sim \mathcal{U}[-1, 1]$.

- **Pendulum**: The variables modeled is the angle of the pendulum and its angular velocity. The dynamics are described as:

$$\frac{d\theta}{dt} = \frac{\dot{\theta}}{ml^2},$$
$$\frac{d\dot{\theta}}{dt} = -mgl\sin(\theta).$$

  We sample mass m, length l and gravity g as $\mathcal{U}[0.5, 1.5]$ and initial conditions $\theta \sim \mathcal{U}[-\frac{\pi}{2}, \frac{\pi}{2}]$ and $\dot{\theta} \sim \mathcal{U}[-1, 1]$.

- **Van-Der Pol Oscillator** We can describe through a first-order ODE as:

$$\frac{dx}{dt} = y,$$
$$\frac{dy}{dt} = \mu(1 - x^2)y - x.$$

  We let $\mu \sim \mathcal{U}[0.1, 5.0]$ and $x, y \sim \mathcal{U}[-3, 3]$.

- **Cartpole:** The dynamics of a rigid body can be described as follows:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = Bu.$$

  Here, q is a generalized coordinate vector, M is a matrix describing the mass, C is a force matrix and $g(q)$ is the gravity vector. The system takes external force as input contained in u which gets mapped into general forces by the matrix B. The inverse dynamics task involves predicting the control inputs u given a target trajectory $\{q(s)\}$. We generate trajectories of an actuated cartpole with varying mass M and train on the inverse-dynamics task.

RadioML

The RadioML 2018.01A dataset [37] consists of measurements of over-the-air radio communications signals. The original dataset consists of signals with different modulation techniques and varying signal-to-noise (SNR) ratios. For the experiment presented in the paper, we restrict the dataset to signals with frequency modulation and an SNR greater or equal to 20 dB. Each signal is composed of two channels describing the In-phase and Quadrature components of the signal (I/Q). Here we consider the first 100 time-steps of each of these signals as the curve to be regressed.

F.7.2 Bayesian View on GBML

GBML can be formulated as a probabilistic inference problem from an empirical Bayes perspective [10]. The objective of GBML involves inferring a set of meta parameters θ_0 that maximize the likelihood of the data $\mathbf{D} = \bigcup_\tau D_\tau$ for all tasks. Keeping the notation above and marginalizing over the task-adapted parameters, the GBML inference problem can be written as follows:

$$\theta_0 = \arg\max_\theta p(\mathbf{D}|\theta) = \arg\max_\theta \prod_\tau \int p(D_\tau^Q|\theta_\tau)p(\theta_\tau|D_\tau^S, \theta)d\theta_\tau, \tag{F.7.1}$$

where $p(D_\tau^Q|\theta_\tau)$ corresponds to the likelihood of each task's data given the adapted parameters and $p(\theta_\tau|D_\tau^S,\theta)$ is the posterior probability of the task-adapted parameters.

The integral in (F.7.1) can be simplified by considering a *maximum a posteriori* (MAP) estimate of the posterior:

$$\theta_\tau^* = \arg\max_{\theta_\tau} p(\theta_\tau|D_\tau^S,\theta).$$

This simplifies the intractable integral to the following optimization problem:

$$\theta_0 = \arg\max_{\theta} \prod_\tau p(D_\tau^Q|\theta_\tau = \theta_\tau^*).$$

F.7.3 Variance of parameters

Here we provide a short account of the variance of the GBML parameters estimator as reported in (F.2.4):

$$\begin{aligned}
\text{Var}\left[\widehat{\theta}\right] &= \text{Var}\left[\theta_0 - \alpha\nabla_{\theta_0}\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(\theta_0, D_i^S)\right] \\
&= \text{Var}\left[\frac{1}{N}\sum_{i=1}^{N}\theta_0 - \alpha\nabla_{\theta_0}\mathcal{L}(\theta_0, D_i^S)\right] \\
&= \frac{1}{N^2}\sum_{i=1}^{N}\text{Var}\left[\widehat{\theta}_i\right].
\end{aligned}$$

F.7.4 Intersection of the Posteriors

We show that the posterior over the parameters conditioned on the support set is proportional to the intersection of the posteriors given each single point in the support. The main idea here is applying the Bayes rule twice and making use of the i.i.d. assumption on the support data.

$$
\begin{aligned}
p\left(\theta \mid D^S\right) &= \frac{p(D^S \mid \theta)p(\theta)}{p(D^S)} \\
&= \frac{\left(\prod_{i=1}^N p(x_i, y_i \mid \theta)\right) p(\theta)}{p(D^S)} \\
&= \frac{\left(\prod_{i=1}^N \frac{p(\theta|x_i,y_i)p(x_i,y_i)}{p(\theta)}\right) p(\theta)}{p(D^S)} \\
&= \frac{\prod_{i=1}^N p(\theta \mid x_i, y_i)}{p(\theta)^{N-1}} \\
&\propto \prod_{i=1}^N p(\theta \mid x_i, y_i).
\end{aligned}
$$

F.7.5 Gradient-Based Meta-Learning is an Unbiased Estimator

Here, we show that GBML with one gradient step is an unbiased estimator. Define the loss for one task as:

$$\mathcal{L}(\theta, \tau) = \mathbb{E}_{x \sim p(x|\tau)} \left[\mathcal{L}(\theta, x)\right].$$

Then the gradient w.r.t θ is an unbiased estimator:

$$\mathbb{E}_{x \sim p(x|\tau)} \left[\nabla_\theta \mathcal{L}(\theta, x)\right] = \nabla_\theta \mathbb{E}_{x \sim p(x|\tau)} \left[\mathcal{L}(\theta, x)\right] = \nabla_\theta \mathcal{L}(\theta, \tau).$$

Moreover, we measure empirically the bias of GBML and LAVA estimators. As a comparison, we include a fully learned network implemented as a HyperNetwork [26] that takes as input the entire support dataset and outputs the adapted parameters directly. Both the adaptation and the aggregation are learned end-to-end together with the downstream task.

We train these three models until convergence on the sine regression dataset. Then, we measure their performance on each task corrupted by Gaussian noise with a standard deviation of 3 on the support labels. The experiment is designed to test how the performance changes when increasing the support size. We show the difference in the loss between adaptation with and without noise for the three models and for different support sizes in Fig. F.7.1. Thus, we are effectively testing the ability of these estimators to recover the performances of the noiseless adaptation. Ideally, an unbiased estimator converges to the correct posterior with enough samples as long as the noise has zero mean. As can be seen in the figure, GBML methods are much more robust to these kinds of perturbations, while learned networks are not unbiased.

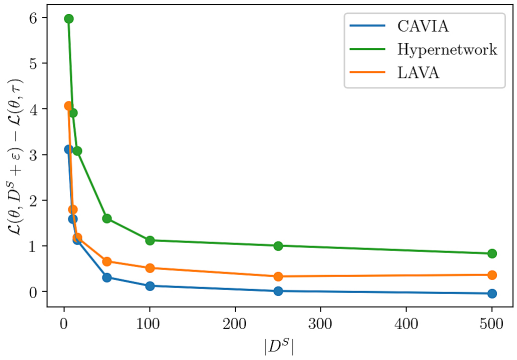


Figure F.7.1: GBML is an unbiased estimator. Scaled MSE when adding noise to the support labels for the sine experiment with increasing support size.

F.7.6 Variance reduction

Below we give an account of the proof of Proposition 1. Consider the variance reduction problem defined in (F.2.5)

$$\min_W \quad \text{Var}\left[\sum_{i=1}^n W_i \theta_i\right], \quad \textit{subject to} \quad \sum_{i=1}^n W_i = I. \tag{F.7.2}$$

We have that

$$\text{Var}\left[\sum_{i=1}^n W_i \theta_i\right] = \sum_{i=1}^n W_i \Sigma_i W_i^T.$$

By introducing a Lagrange multiplier λ, we reach the following optimization problem:

$$\begin{aligned} &\min_W \quad F(\theta, W), \\ \text{with} \quad &F(\theta, W) = \sum_{i=1}^n W_i \Sigma_i W_i^T + \lambda \left(\sum_{i=1}^n W_i - I\right). \end{aligned}$$

By taking the derivative w.r.t W_i and using the fact that Σ_i is symmetric we find that

$$\frac{dF}{dW_i} = 2W_i \Sigma_i - \lambda.$$

Setting this equal to 0, we get

$$W_i = \frac{\lambda}{2} \Sigma_i^{-1}. \tag{F.7.3}$$

Algorithm 4 LAVA Pseudo-Code

Require: $p(\mathcal{T})$ distribution of tasks
Require: α, η, ϵ hyperparameters.
Ensure: Output results

Randomly initialize θ_0
while not done **do**
  Sample batch of tasks $\mathcal{T} \sim p(\mathcal{T})$
  for all $\tau \in \mathcal{T}$ **do**
    Sample $D_\tau^S, D_\tau^Q \sim \tau$
    for all $(x_i, y_i) \in D_\tau^S$ **do**
      Evaluate $\widehat{\theta}_i = \theta_0 - \alpha \nabla_\theta \mathcal{L}(\theta_0, x_i, y_i)$
      Evaluate $H_i = \frac{d^2}{d\theta^2} \mathcal{L}(\hat{\theta}_i, x_i, y_i)$
      Evaluate $\tilde{H}_i = \frac{1}{1+\epsilon} (H_i + \epsilon I)$
    end for
    Evaluate $\tilde{H} = \sum_i \tilde{H}_i$
    Evaluate $\hat{\theta}_\tau = \tilde{H}^{-1} \sum_i \tilde{H}_i \hat{\theta}_i$
  end for
  Update $\theta_0 = \theta_0 - \eta \nabla_{\theta_0} \sum_{\tau \in \mathcal{T}} \mathcal{L}(\hat{\theta}_\tau, D_\tau^Q)$ using each D_τ^Q
end while

From the condition defined in (F.7.2), we have

$$\frac{\lambda}{2} \sum_{i=1}^{n} \Sigma_i^{-1} = I,$$

$$\lambda = 2 \left[\sum_{i=1}^{n} \Sigma_i^{-1} \right]^{-1}.$$

Plugging this into (F.7.3), it follows that

$$W_i = \left[\sum_{i=1}^{n} \Sigma_i^{-1} \right]^{-1} \Sigma_i^{-1}.$$

Given these weights, W_i, the distribution of $\sum_{i=1}^{n} W_i \theta_i$ follows a normal distribution equivalent to the estimate in (F.3.2). □

F.7.7 Geometry of the Parameter Space

In order to minimize the loss described in (F.2.1) using LAVA's adaptation ((F.3.2)), we implicitly make an assumption over the geometry of the task space. In this section, we expand on this assumption and provide an example of the limitations of using Gaussians to represent the distribution of solutions in parameter space. In fact, the model

has to shape the parameter space such that, locally, the Laplace approximation can exactly estimate the posterior. This imposes certain constraints on the structure of the parameter space arising from task overlap.

From the assumption of task overlap (Definition 5), we have that $f(\cdot, x) : \mathcal{T} \to \mathcal{Y}$ is non-injective, or that for each $(x, y) \in \mathcal{X} \times \mathcal{Y}$, the pre-image $\mathcal{M}_{x,y} = f_x^{-1}(y)$ defines a subspace of the task-manifold and $\bigcup_{x,y} \mathcal{M}_{x,y}$ defines a *cover* of the task space $\mathcal{T}$. As the Laplace approximation estimates the space with a normal distribution, we require the underlying fiber $\mathcal{M}_{x,y}$ to be simply connected, or in other words, each element (x, y) induces a path-connected space of model parameterizations in which every path can be continuously deformed into another one.

An example of a space of tasks with a non-trivial topology could be an annulus or a disk with a hole inside it. Such a space can arise in problems of goal-oriented navigation. Consider an agent in a 2-dimensional plane P where the task provides the control inputs to reach a given goal position specified by a coordinate $(x, y) \in P$. Given trajectories provided by expert demonstrations, infer the goal-conditioned policy. In this setting, each support point corresponds to a position of the agent, and the loss is defined as the L_2 distance to the goal. If the goal can be situated anywhere on the plane P, then a single support point (x_s, y_s) defines a space of possible goal positions as a circle around the current position. By continuity, this implies that the space of possible parameters M_{x_s,y_s} that yield the possible policies must be non-simply connected as well. This poses problems for the Laplace approximation, as the Gaussian distribution implicitly holds an assumption on the topological properties.

F.7.8 A Note on Computational Complexity

Here we provide a description of how we compute The Hessian in closed form with respect to the loss defined in (F.2.1) and how to implement it in practice on a standard automatic differentiation framework.

Assuming the dimensions on the output to be: $\mathcal{Y} \subseteq \mathbb{R}^{k \times 1}$. In the case of a multi-layer perception, define $f_\psi \in \mathbb{R}^{d \times 1}$ the network up to the last layer excluded and augment it with an appropriate one-padding to vectorize the bias i.e., $z = \begin{bmatrix} f_\psi(x) \\ 1 \end{bmatrix} \in \mathbb{R}^{(d+1) \times 1}$. Moreover, define the adaptation parameters to be the weights and the bias of the last layer i.e., $\theta = [W, b] \in \mathbb{R}^{k \times (d+1)}$. We can rewrite the loss of (F.2.1) for one data point of the support as:

$$\mathcal{L}(\theta, x_i, y_i) = \|\theta \cdot z - y\|_2^2 .$$

The corresponding Hessian can be written as:

$$H_i = 2I_{k \times k} \otimes (z \cdot z^T) .$$

The notation on the most standard automatic differentiation frameworks like PyTorch [40] differs somehow from the above way of computing the Hessian. In Pseudo-code Algorithm 5 we provide the necessary code for computing the Hessian.

Algorithm 5 Hessian Pseudo-Code in PyTorch.

Require: $x_i, y_i, f_\psi, W_\theta, b_\theta$

$z = f_\psi(x_i)$

$H_W = I_{k\times k} \otimes (z \cdot z^T)$

$H_b = I_{k\times k} \otimes z$

$$H = 2 \begin{bmatrix} H_W & H_b \\ H_b^T & I_{k\times k} \end{bmatrix}$$

F.7.9 Additional Experiments

Condition Number

We measure the condition number of the Hessian for LAVA under both the conditional setting and when only updating the final layer. We calculate the condition number $\kappa(H)$ for $H = \sum_{i=1}^{N} H_i$ and $\tilde{H}$ calculated similarly with the approximate Hessians defined in (F.3.3). We present the results across different support sizes in Table F.7.1 below.

	Condition Number $\kappa(H)$		Regularized Condition Number $\kappa(\tilde{H})$	
Model	Epoch=1	Epoch=200	Epoch=1	Epoch=200
LAVA + CAVIA (dim=2)	4.96×10^0	1.94×10^0	1.04×10^0	1.96×10^0
LAVA + CAVIA (dim=4)	1.31×10^2	6.26×10^2	1.04×10^0	41.03×10^0
LAVA + CAVIA (dim=8)	5.28×10^6	1.78×10^6	1.03×10^0	66.73×10^0
LAVA + CAVIA (dim=16)	1.35×10^{10}	5.31×10^9	1.04×10^0	36.00×10^0
LAVA + ANIL	1.89×10^{11}	1.36×10^{11}	74.36×10^0	217.86×10^0

Table F.7.1: Condition and Approximate Condition Numbers for Support Size 10

The condition number increases with the number of adaptation parameters increases. A high condition number signifies that the matrix is close to being singular, meaning it is difficult to invert. In the context of matrix inversion, this implies that small errors in the input data can lead to disproportionately large errors in the output, making the inversion process highly sensitive and potentially unreliable. In the table, we also present the condition number at convergence. This approximate condition number is derived from the approximate Hessian, as defined in (F.3.3). Initially, the regularization promotes an even distribution of singular values, but as training progresses, the matrices become increasingly skewed. This behavior is expected, as the Hessian for each support point H_i should be elongated in specific directions, resulting in a relatively high condition number.

Additional Sine Results

Here we provide additional results for the sine regression experiment. Using the same experimental settings described in Section F.5.1 and Section F.7.1, we present MSE and

Models	$\lvert D^S\rvert = 1$	$\lvert D^S\rvert = 2$	$\lvert D^S\rvert = 10$	$\lvert D^S\rvert = 20$
ANIL	171.07 ± 10.19	46.05 ± 6.42	1.32 ± 0.1	0.37 ± 0.02
CAVIA	247.6 ± 4.02	56.75 ± 2.5	2.96 ± 0.46	1.44 ± 0.23
VFML	286.0 ± 0.95	106.31 ± 7.14	31.72 ± 4.21	18.63 ± 1.8
LLAMA	248.48 ± 4.12	109.75 ± 12.41	29.08 ± 4.81	18.31 ± 2.01
Metamix	329.9 ± 13.53	120.12 ± 7.44	42.04 ± 4.27	24.85 ± 2.25
VR	318.21 ± 17.05	338.3 ± 22.91	283.05 ± 21.38	292.28 ± 32.98
LAVA	$\mathbf{169.38 \pm 9.18}$	$\mathbf{7.01 \pm 2.02}$	$\mathbf{0.11 \pm 0.02}$	$\mathbf{0.09 \pm 0.02}$

Table F.7.2: Model performance MSE (10^{-2}) based on different support sizes on the Sine wave regression problem.

standard deviations for 5 seeds for LAVA and baselines in Table F.7.2. In particular, we provide results for support size equal to 1 as an ablation. The MSE value is noticeably higher as one support point is not sufficient to identify the underlying task accurately. Nevertheless, results show comparable performances between LAVA and ANIL when $|D^S| = 1$ as (F.3.2) becomes equivalent to the adaptation described in (F.2.2). Qualitative results comparing the effect of different support points and the addition of Gaussian noise on the support are shown in Fig. F.7.3. The figure shows how the prior of the model (Green dotted line) evolves into the posterior for LAVA (orange curve) and ANIL (blue curve), this is compared with the ground truth curve (green dashed line). The effects of task overlap can be seen in the plots in the left column. The support size is small, GBML models like ANIL manage to fit the support points (red crosses) but fail to identify the true underlying distribution.

F.7.10 Mini-Imagenet

Model	5-ways 1-shot	5-ways 5-shot
ANIL	45.94 ± 0.94	62.86 ± 0.26
CAVIA	47.84 ± 0.41	63.09 ± 0.51
LLAMA	40.19 ± 0.85	56.50 ± 0.15
VFML	49.60 ± 0.5	$\mathbf{66.20 \pm 0.80}$
Metamix	$\mathbf{50.51 \pm 0.86}$	65.73 ± 0.72
VR	49.20 ± 1.40	63.60 ± 0.80
LAVA	46.69 ± 1.45	61.51 ± 0.97

Table F.7.3: Results Mini-Imagenet with support sizes 1 and 5

We further experiment with classification on the Mini-Imagenet dataset [41]. We use the training-set split as used in [32] which leaves 64 classes for training, 16 for validation and 20 for test. We experiment with 5-way classification in either a 1-shot or

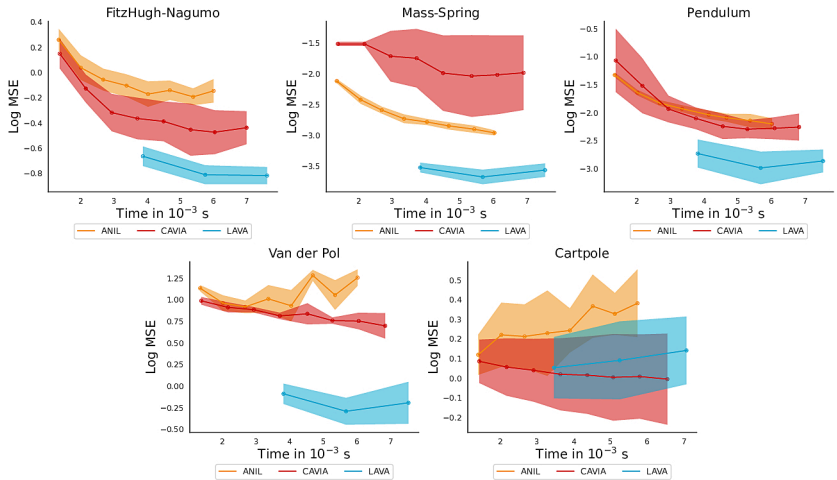


Figure F.7.2: Further Evaluation of Computational Time. Computational Complexity results for ODEs regression with support size 5 and a different number of adaptation steps.

5-shot setting. We train the models for 1000 epochs and perform model selection by choosing the one with the best performance on the validation set. We present results on the test set in Table F.7.3.

Standard classification benchmarks such as Mini-Imagenet test the capability of the model to incorporate high-dimensional data in the form of images. Some of the methods are optimized toward image data and attempt to efficiently learn a well-structured representation space of images, such that the adaptation reduces to modifying decision boundaries. In particular, few-shot image classification problems in this form are inherently discrete problems that do not suffer as extensively from the task overlap assumption as outlined in Definition 5. In this context, the Gaussian assumption on the parameter distribution is not an accurate one, as each element of the support induces a *multi-modal* distribution in the parameter space. Thus variance reduction becomes inaccurate, and unbiased methods such as ANIL prove better. Nevertheless, LAVA manages to get competitive performances on this benchmark as well.

F.7.11 ODEs Computational Complexity

We provide additional computational complexity analysis for the ODEs experiments. As it can be noticed in Fig. F.7.2, for most of the experiments LAVA provides a comparative advantage in terms of computational complexity and general performances.

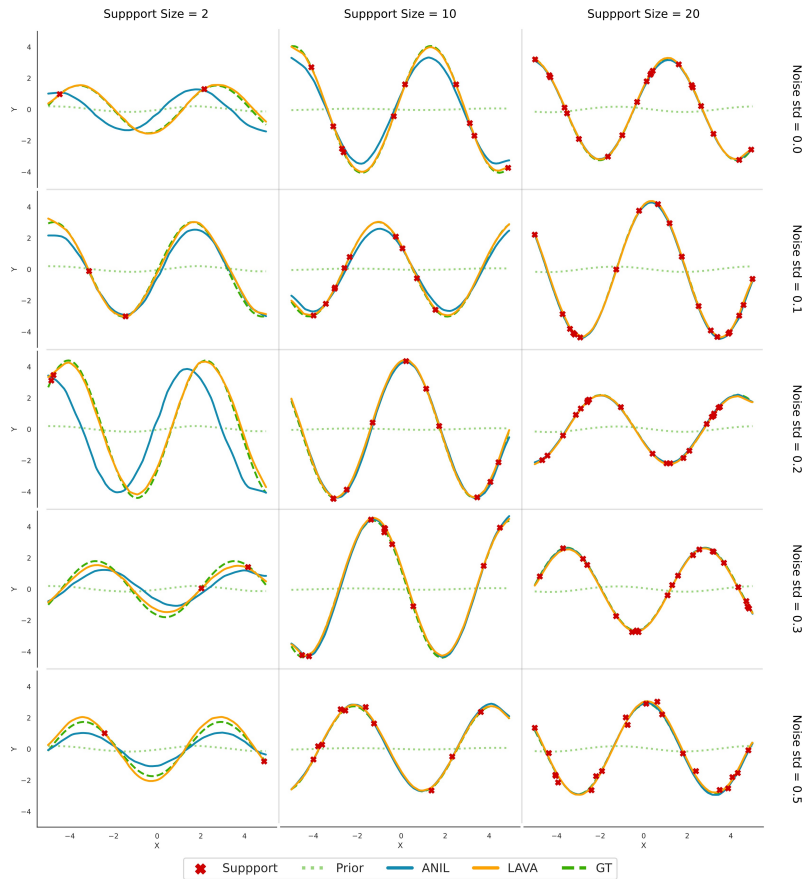


Figure F.7.3: Qualitative Results for Sine Regression. Estimated sinusoidal curves with varying amounts of support size and noise in the support data compared with GBML.

References

[1] Jake Snell, Kevin Swersky, and Richard Zemel. "Prototypical networks for few-shot learning". In: *Advances in neural information processing systems* 30 (2017).

[2] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, et al. "Neural processes". In: *arXiv preprint arXiv:1807.01622* (2018).

[3] Chelsea Finn, Tianhe Yu, Tianhao Zhang, et al. "One-shot visual imitation learning via meta-learning". In: *Conference on robot learning*. 2017, pp. 357–368.

[4] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks". In: *International conference on machine learning*. PMLR. 2017, pp. 1126–1135.

[5] Jennifer A Hoeting, David Madigan, Adrian E Raftery, et al. "Bayesian model averaging: a tutorial (with comments by M. Clyde, David Draper and EI George, and a rejoinder by the authors". In: *Statistical science* 14.4 (1999), pp. 382–417.

[6] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[7] Marcin Andrychowicz, Misha Denil, Sergio Gomez, et al. "Learning to learn by gradient descent by gradient descent". In: *Advances in neural information processing systems* 29 (2016).

[8] Sachin Ravi and Hugo Larochelle. "Optimization as a model for few-shot learning". In: *International conference on learning representations*. 2016.

[9] Chelsea Finn and Sergey Levine. "Meta-Learning and Universality: Deep Representations and Gradient Descent can Approximate any Learning Algorithm". In: *International Conference on Learning Representations*. 2018.

[10] Erin Grant, Chelsea Finn, Sergey Levine, et al. "Recasting gradient-based meta-learning as hierarchical bayes". In: *arXiv preprint arXiv:1801.08930* (2018).

[11] Joachim Hartung, Guido Knapp, and Bimal K Sinha. *Statistical meta-analysis with applications*. John Wiley & Sons, 2011.

[12] Robert E Kass, Luke Tierney, and Joseph B Kadane. "Laplace's method in Bayesian analysis". In: *Contemporary Mathematics* 115 (1991), pp. 89–99.

[13] Aniruddh Raghu, Maithra Raghu, Samy Bengio, et al. "Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML". In: *International Conference on Learning Representations*. 2020.

[14] Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, et al. "Fast context adaptation via meta-learning". In: *International Conference on Machine Learning*. 2019, pp. 7693–7702.

[15] James Martens. *Second-order optimization for neural networks*. University of Toronto (Canada), 2016.

[16] David I Warton. "Penalized normal likelihood and ridge regularization of correlation and covariance matrices". In: *Journal of the American Statistical Association* 103.481 (2008), pp. 340–349.

[17] Zhenguo Li, Fengwei Zhou, Fei Chen, et al. "Meta-sgd: Learning to learn quickly for few-shot learning". In: *arXiv preprint arXiv:1707.09835* (2017).

[18] Alex Nichol and John Schulman. "Reptile: a scalable metalearning algorithm". In: *arXiv preprint arXiv:1803.02999* 2.3 (2018), p. 4.

[19] Yoonho Lee and Seungjin Choi. “Gradient-based meta-learning with learned layerwise metric and subspace”. In: *International Conference on Machine Learning*. 2018, pp. 2927–2936.

[20] Eunbyung Park and Junier B Oliva. “Meta-curvature”. In: *Advances in neural information processing systems* 32 (2019).

[21] Sebastian Flennerhag, Andrei A Rusu, Razvan Pascanu, et al. “Meta-Learning with Warped Gradient Descent”. In: *International Conference on Learning Representations*. 2020.

[22] Cuong Nguyen, Thanh-Toan Do, and Gustavo Carneiro. “Uncertainty in model-agnostic meta-learning using variational inference”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2020, pp. 3090–3100.

[23] Jaesik Yoon, Taesup Kim, Ousmane Dia, et al. “Bayesian model-agnostic meta-learning”. In: *Advances in neural information processing systems* 31 (2018).

[24] Krunoslav Lehman Pavasovic, Jonas Rothfuss, and Andreas Krause. “MARS: Meta-learning as Score Matching in the Function Space”. In: *The Eleventh International Conference on Learning Representations*. 2023.

[25] Lisha Chen and Tianyi Chen. “Is Bayesian Model-Agnostic Meta Learning Better than Model-Agnostic Meta Learning, Provably?” In: *International Conference on Artificial Intelligence and Statistics*. 2022, pp. 1733–1774.

[26] David Ha, Andrew M Dai, and Quoc V Le. “HyperNetworks”. In: *International Conference on Learning Representations*. 2022.

[27] Jonathan Gordon, John Bronskill, Matthias Bauer, et al. “Meta-Learning Probabilistic Inference for Prediction”. In: *International Conference on Learning Representations*. 2019.

[28] Matthieu Kirchmeyer, Yuan Yin, Jérémie Donà, et al. “Generalizing to new physical systems via context-informed dynamics model”. In: *International Conference on Machine Learning*. 2022, pp. 11283–11301.

[29] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, et al. “Conditional neural processes”. In: *International conference on machine learning*. 2018, pp. 1704–1713.

[30] Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, et al. “Attentive Neural Processes”. In: *International Conference on Learning Representations*. 2019.

[31] Adam Santoro, Sergey Bartunov, Matthew Botvinick, et al. “Meta-learning with memory-augmented neural networks”. In: *International conference on machine learning*. 2016, pp. 1842–1850.

[32] Sachin Ravi and Hugo Larochelle. “Optimization as a model for few-shot learning”. In: *International conference on learning representations*. 2017.

[33] Ling Xiao Wang, Kevin Huang, Tengyu Ma, et al. “Variance-reduced first-order meta-learning for natural language processing tasks”. In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2021, pp. 2609–2615.

[34] Hansi Yang and James Kwok. “Efficient Variance Reduction for Meta-Learning”. In: *International Conference on Machine Learning*. 2022, pp. 25070–25095.

[35] Yangbin Chen, Yun Ma, Tom Ko, et al. “Metamix: Improved meta-learning with interpolation-based consistency regularization”. In: *2020 25th International Conference on Pattern Recognition (ICPR)*. 2021, pp. 407–414.

[36] Shuyi Zhang, Bin Guo, Anlan Dong, et al. “Cautionary tales on air-quality improvement in Beijing”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 473.2205 (2017), p. 20170457.

[37] Timothy James O’Shea, Tamoghna Roy, and T Charles Clancy. “Over-the-air deep learning based radio signal classification”. In: *IEEE Journal of Selected Topics in Signal Processing* 12.1 (2018), pp. 168–179.

[38] James Martens and Roger Grosse. “Optimizing neural networks with kronecker-factored approximate curvature”. In: *International conference on machine learning*. 2015, pp. 2408–2417.

[39] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[40] Adam Paszke, Sam Gross, Francisco Massa, et al. “PyTorch: An imperative style, high-performance deep learning library”. In: *Advances in neural information processing systems* 32 (2019).

[41] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, et al. “Matching networks for one shot learning”. In: *Advances in neural information processing systems* 29 (2016).