Alexander Neitz

Towards learning mechanistic models at the right level of abstraction

Towards Learning Mechanistic Models at the Right Level of Abstraction

Dissertation
der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Alexander Neitz
aus Heidelberg

Tübingen
2021

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	23.11.2022
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter:	Prof. Dr. Bernhard Schölkopf
2. Berichterstatter:	Prof. Dr. Jakob Macke

Abstract

The human brain has the ability to make predictions, plan and imagine by mental simulation. Artificial neural networks, while achieving great performance in certain domains, still seem to lack a mechanistic understanding of the world. In this thesis we look at different approaches towards making neural networks better capture the underlying mechanisms of the modeled system. We will look at *Adaptive skip intervals* (ASI), a method that allows dynamical models to choose their own temporal coarsening at every point, making long-term predictions both easier and more computationally efficient. Next, we will look into alternative ways to aggregate gradients across environments, leading to the notion of *Invariant Learning Consistency* (ILC), and the method *AND-mask*, for modified stochastic gradient descent. By filtering out inconsistent training signals from different environments, the shared mechanisms remain. Finally, we will see that learning based on meta-gradients can transform trajectories of dynamical systems so as to construct useful learning signal toward an underlying objective, such as reward in reinforcement learning. This allows the internal model to include both temporal as well as state abstraction.

Zusammenfassung

Das menschliche Gehirn ist in der Lage, Vorhersagen zu treffen, zu planen und sich durch mentale Simulationen kontrafaktische Situationen vorzustellen. Künstliche neuronale Netze sind zwar in bestimmten Bereichen brereits sehr leistungsfähig, scheinen aber immer noch ein mechanistisches Verständnis der Welt zu vermissen. In dieser Arbeit befassen wir uns mit verschiedenen Ansätzen, wie neuronale Netze die zugrundeliegenden Mechanismen des modellierten Systems besser erfassen können. Wir werden uns mit *Adaptive skip intervals* (ASI) befassen; eine Methode, die es dynamischen Modellen ermöglicht, ihre eigene zeitliche Vergröberung an jedem Punkt zu wählen. Dadurch werden langfristige Vorhersagen sowohl einfacher als auch rechnerisch effizienter. Als Nächstes werden wir uns mit alternativen Möglichkeiten zur Aggregation von Gradienten in verschiedenen Umgebungen befassen, was zum Begriff der *Invariant Learning Consistency* (ILC) und der Methode *AND-mask* für einen modifizierten stochastischen Gradientenabstieg führt. Durch das Herausfiltern inkonsistenter Trainingssignale aus verschiedenen Umgebungen bleiben die gemeinsamen Mechanismen erhalten. Schließlich werden wir sehen, dass Lernen auf der Grundlage von Meta-Gradienten Trajektorien von dynamischen Systemen transformieren kann, um nützliche Lernsignale in Richtung eines zugrunde liegenden Ziels zu konstruieren, wie z. B. Reward beim Reinforcement Learning. Dadurch kann das interne Modell sowohl eine zeitliche als auch eine Zustandsabstraktion beinhalten.

Preface

This thesis primarily consists of the following publications:

- Alexander Neitz, Giambattista Parascandolo, Stefan Bauer, and Bernhard Schölkopf (2018). "Adaptive skip intervals: Temporal abstraction for recurrent dynamical models." In: Advances in Neural Information Processing Systems 31, pp. 9816–9826.
  (NeurIPS 2018)

- Giambattista Parascandolo*, Alexander Neitz*, Antonio Orvieto, Luigi Gresele, and Bernhard Schölkopf (2021). "Learning explanations that are hard to vary." In: International Conference on Learning Representations.
  (ICLR 2021) *(*equal contribution)*

- Alexander Neitz*, Giambattista Parascandolo*, and Bernhard Schölkopf (2021). "A teacher-student framework to distill future trajectories." In: International Conference on Learning Representations.
  (ICLR 2021) *(*equal contribution)*

During my Ph.D. I additionally contributed to the following works which are not part of this thesis:

- Akilesh Badrinaaraayanan, Suriya Singh, Anirudh Goyal, Alexander Neitz, and Aaron Courville (2019). "Towards Jumpy Planning" In: Generative Modeling and Model-Based Reasoning for Robotics and AI (ICML 2019 Workshop)

- Giambattista Parascandolo, Lars Buesing, Josh Merel, Leonard Hasenclever, John Aslanides, Jessica B. Hamrick, Nicolas Heess, Alexander Neitz, and Theophane Weber (2020). "Divide-and-Conquer Monte Carlo Tree Search For Goal-Directed Planning."
  arXiv preprint (arXiv:2004.11410)

- Ossama Ahmed*, Frederik Träuble*, Anirudh Goyal, Alexander Neitz, Manuel Wüthrich, Yoshua Bengio, Bernhard Schölkopf, and Stefan Bauer (2021). "CausalWorld: A Robotic Manipulation Bench- mark for Causal Structure and Transfer Learning." In: International Conference on Learning Representations.
(ICLR 2021) *(*equal contribution)*

- Luca Biggio*, Tommaso Bendinelli*, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo, (2021). "Neural Symbolic Regression that Scales." In: Proceedings of 38th International Conference on Machine Learning (ICML). Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 936–945.
(ICML 2021) *(*equal contribution)*

- Hsiao-Ru Pan, Gürtler Nico, Alexander Neitz, and Bernhard Schölkopf (2021). "Direct Advantage Estimation."
arXiv preprint (arXiv:2109.06093)

Acknowledgments

I would like to begin by thanking my supervisor Bernhard for giving me the opportunity to do my Ph.D. in his amazing department. Thank you for your trust, and for all your insightful advice and feedback. I learned a lot from our meetings and brainstorming sessions. I am also very grateful that you gave me the freedom to explore the research directions I believe in.

I also thank my co-advisors, thesis advisory committee members and examiners Georg and Andreas for having always been available to give advice, for agreeing to review this thesis, and for being part of my examination committee. I thank Setareh for kindly agreeing to be an examiner for my defense.

Thanks a lot to Sabrina for all the support and for your kindness!

I want to thank GB (Giambattista) for the countless hours we spent brainstorming, discussing ideas, coding, writing, cooking, making music, etc. Thanks for being a great collaborator and friend.

Thank you to the "pupper group" Alessandro, Paul, Niki, Mateo, John, Matej for being a major part of this fantastic experience.

I thank the people who made this experience so valuable: the Pub quiz group (GB, Luigi, Sabrina, Timmy, Matthias, Francesco), the gaming group (Mehdi, Chantal, Partha, Soubhik, Anurag), the volleypong team (GB, Dieter, Mara, Justus), and of course to everyone in the Empirical Inference department for the friendly and collaborative atmosphere.

I want to thank my parents and my sister for their endless support.

Finally, I thank my partner Lin for believing in me and for all your encouragement which made this PhD possible.

Contents

List of Figures

List of Tables

1

Introduction

The work presented in this thesis is inspired by a fascinating human ability: learning entirely within one's own mind. A person can close their eyes, think about a problem for hours, and keep discovering new facts and insights about this problem without receiving any external input or feedback. This is likely possible because the brain uses internal *generative models* which capture essential aspects of the external world. Rather than mere statistical associations or reactive responses to stimuli, such generative models provide mechanisms which allow agents to perform computations such as *mental simulation* and *planning*. Starting from initial assumptions, these computational processes can reveal non-obvious consequences, which the person can consider as real evidence, much like they would if they were observing a physical system. The importance of mental simulation has been recognized by Kenneth Craik, who wrote:

> *"If the organism carries a "small-scale model" of external reality and of its own possible actions within its head, it is able to try out various alternatives, conclude which is the best of them, react to future situations before they arise, utilise the knowledge of past events in dealing with the present and future, and in every way to react in a much fuller, safer, and more competent manner to the emergencies which face it."*
>
> (Craik, 1952, Chapter 5)

More concisely but in a similar spirit, Konrad Lorenz described thinking as "acting in an imagined space" (Lorenz, 1973).

If we want to create artificial agents which are as adaptive and versatile as humans, a plausible hypothesis is that we have to provide these agents with the ability to perform mental simulations and reasoning as well. In fact, in the early days of artificial intelligence research, its pioneers believed that reasoning is the main challenge in building intelligent systems. It soon became clear that while symbolic reasoning at a (super-)human level is feasible in many domains, identifying useful symbols in unstructured streams of sensory data – seemingly effortlessly performed by the human brain – is much more challenging than initially thought. This phenomenon has been termed "Moravec's Paradox" after Hans Moravec, who wrote in 1988: *"it is comparatively easy to make computers exhibit adult level performance on intelligence tests or playing checkers, and difficult or impossible to give them the skills of a one-year-old when it comes to perception and mobility"* (Moravec, 1988). An intuitive conclusion could be: since we already have very powerful symbolic reasoning and planning systems, the only missing piece is to independently solve the symbol grounding problem. However, this thesis will argue that perception, symbol grounding, and reasoning should be optimized jointly, rather than in isolation.

Recent years have brought staggering progress in artificial intelligence research. There have been major advances in subfields such as object recognition (Krizhevsky et al., 2012; He et al., 2016), image generation (Goodfellow et al., 2014; Ramesh et al., 2021), speech recognition (Hinton et al., 2012), natural language processing (Brown et al., 2020) or reinforcement learning (Mnih et al., 2015; Silver et al., 2016). Much of this progress was driven by neural networks trained mostly end-to-end on large amounts of data.

While the progress of deep learning is impressive, most of these success stories could be argued to be solutions to problems which do not require reasoning, but merely consist of fast, intuitive reactions to inputs – System 1 thinking, in the taxonomy of Kahneman, 2011. AlphaGo is a notable exception: it uses Monte-Carlo Tree Search (Coulom, 2006), an

algorithm which utilizes a dynamical model of the system to plan ahead explicitly, iteratively refining the value estimates of different action sequences. However, the game of Go makes it particularly easy to apply this technique, as the dynamical system is (1) simple, (2) fully-observed, (3) deterministic, and (4) known in advance. A big open challenge is to apply techniques such as AlphaGo to real-world systems in which none of these properties are satisfied, i.e. to *complex*, *partially observed*, *stochastic* systems whose dynamics are *unknown* in advance.

Limitations of mental models In artificial domains such as Go, the agent can be equipped with an exact specification of the system dynamics. This is not possible for robots acting in the real world, for several reasons: The laws of nature are not entirely known. Even if we knew the precise physical laws governing our universe, a full real-time quantum mechanical simulation of a macroscopic system is far beyond today's computational capabilities. Moreover, as agents only receive noisy and local sensory information, they do not have access to the exact underlying state of the system. Therefore, it seems unrealistic to create intelligent agents by giving them the laws of the universe and simply letting them simulate.

However, there are several assumptions which can help us out.

Locality: The agent does not have to consider things that are happening far away, as these are less likely to have a strong influence on the current problem.

Irrelevant details: some things (e.g., the wiggling of tree leaves in the wind) might be both hard to simulate and completely uninteresting to the goals that the agent cares about.

Favorable approximations: there might be ways to approximate the system to a sufficient degree, while giving up little accuracy.

Relevant level of abstraction: There may be ways to describe the system in terms of quantities, objects and events which the agent finds

> useful, and the laws at this level of abstraction could turn out to be particularly easy to represent. For example, rather than describing the position and momentum of each molecule in a gas, it has turned out beneficial to use macroscopic thermodynamic quantities such as temperature and pressure in practice.

These assumptions have the realization in common that for most goal-directed agents, it is not *necessary* to model and simulate everything in all of its detail. Doing so would be attempting to solve a much harder problem than necessary, violating Vladimir Vapnik's principle "*When solving a problem of interest, do not solve a more general problem as an intermediate step*" (Vapnik, 2006, Chapter 3).

A common counterargument to making dynamical models utility-based is that the whole purpose of having a model is to be able to reuse it even when the goals change. While this is true, it is important to realize that there is a tradeoff. If we take the point to its extreme, we should model everything in detail, because any simplification will rule out certain goals. However, the space of reasonable goals, while large, might be much smaller than the space of theoretically conceivable goals. If this knowledge can be exploited, we should do so.

One aim of this thesis is therefore to develop learning systems that discover dynamical models at the *right level of abstraction*. Abstraction here means both a temporal coarsening as well as a representation of the state of the system in terms of useful entities. The "right level" refers to the utility towards a space of plausible downstream tasks. We take an approach that stays close to the end-to-end paradigm of deep learning, as it has shown to be surprisingly successful recently.

Relationship to causality The above research goals are closely related to those of causality (Pearl, 2009b; Peters et al., 2017b), and particularly the emerging field of *causal representation learning* (Schölkopf et al., 2021). In line with the motivation stated above, causal models can be viewed as striking a balance between physical laws in the form of differential equations and purely statistical models. One perspective on

causality is that it is not an objective property of the physical world, but a tool evolved to help intelligent agents to help achieve their goals. This is not the only point of view, and some authors believe causal representations to be objective, agent-agnostic descriptions of the world. However, if we do choose the path of utility-based representations, we have the option of manually designing objectives for intermediate representations with the hope that they will be useful, or we set up the downstream tasks explicitly and optimize everything to maximize the overall objective. In the latter case, we would hope that causal representations emerge as the most efficient solution to the problems that an agent faces. A drawback of the end-to-end approach is that it can be hard to formulate the exact utility metric one wishes to optimize, and optimizing surrogate objectives can lead to unintended solutions (Goodhart, 1984; Amodei et al., 2016).

Reinforcement learning Reinforcement learning studies agents that learn to act in a dynamic environment so as to maximize the sum of rewards (Sutton and Barto, 2018). In contrast to supervised learning, the correct actions are not known in advance, but have to be discovered through exploration. Approaches to this problem can be classified along several dimensions, including whether they are *model-based* or *model-free* (although as we will see, there exists a spectrum between the extremes). Model-free methods represent a policy and/or a value function directly, without explictly modeling the system's state or its dynamical laws. Model-based methods use a dynamical model of the system to improve the policy. This is typically achieved by using the model to generate additional, imaginary data, either in an offline manner (Sutton, 1991), or as a local search around the agent's current state, such as in AlphaGo (Silver et al., 2016). If the dynamical model of the system is not known in advance, model-based methods try to estimate it from the agent's obervations.

An advantage of model-based methods is that they make use of the vast information present in trajectories of sensory observations, while model-free methods do not use the observation trajectories as training

signal: observations are only used as the input into the model, but not as the target. On the other hand, model-free methods excel when there is plenty of data, and when representing the system dynamics is much harder than representing a good policy.

Based on our goals stated above, we should aim to take the best of both worlds: We should be model-based in the sense that agents benefit from models with which they can perform imagination-based planning. At the same time, we do not want to ask more than necessary of the agent, and learn dynamical models that are as useful as possible towards the end of achieving a better policy, which likely includes making use of heuristics and "mental shortcuts" whenever they are feasible.

Overview of the thesis

The works presented in this thesis investigate different ways to change or augment the training objective, with the goal of making the resulting models more mechanistic, as described above. We will begin with a relaxed constraint on temporal alignment. Next, we will investigate an alternative way to aggregate gradients across minibatches that reduces overfitting to spurious signals. Finally, we will look at method that meta-learns how to derive training signal from trajectories.

Chapter 2 looks at whether we can remove the constraint that the time-evolution of rollouts from a dynamical model has to align perfectly with the modeled system. We will examine the hypothesis that the most efficient temporal coarsening is not just system-dependent, but changes from state to state even within the same system, and describe a method that allows a model to pick its own level of temporal coarsening, based on what makes it easy to predict long-range trajectories. Crucially, the prediction model does not need to model explicitly how many frames were skipped. This relaxation makes deterministic long-term predictions easier and more computationally efficient.

Chapter 3 aims at achieving more mechanistic models by changing the way we aggregate gradients across training environments. We show that it can help reduce overfitting to spurious patterns, while the patterns shared across environments are preserved. This helps our research goal in situations when the mechanistic model is present in all environments.

While there are methods for temporal abstraction (Neitz et al., 2018; Parascandolo et al., 2020), as well as for state-abstraction (Watter et al., 2015a; Schrittwieser et al., 2020) in dynamical models, combining these two goals has shown to be surprisingly challenging. Chapter 4 addresses this by introducing a teacher-student framework to construct a transformed training signal from trajectories, in situations where there is an explicit utility measure, such as reward in reinforcement learning tasks. We show that we can train a teacher network to transform the trajectories – which can now be seen as *privileged information* (Vapnik and Vashist, 2009) – and present them to the student in a way that helps the student maximize its utility.

The thesis assumes familiarity with the basic concepts of Deep Learning. A comprehensive introduction to these is provided in Goodfellow et al. (2016). Specific background necessary to understand the technical content will presented individually in each chapter.

2

Adaptive skip intervals

In this chapter[1] we introduce a method which enables a recurrent dynamics model to be temporally abstract. Our approach, which we call Adaptive Skip Intervals (ASI), is based on the observation that in many sequential prediction tasks, the exact time at which events occur is irrelevant to the underlying objective. Moreover, in many situations, there exist prediction intervals which result in particularly easy-to-predict transitions. We show that there are prediction tasks for which we gain both computational efficiency and prediction accuracy by allowing the model to make predictions at a sampling rate which it can choose itself.

A core component of intelligent agents is the ability to predict certain properties of future states of their environments (Legg and Hutter, 2007). For example, model-based reinforcement learning (Daw, 2012; Arulkumaran et al., 2017) decomposes the task into the two components of learning a model and then using the learned model for planning ahead.

Despite significant recent advances, even relatively simple tasks like pushing objects is still a challenging robotic task and foresight for robot planning is still limited to relatively short horizon tasks (Finn and Levine, 2017). This is partially due to the fact that errors even from early stages

1 Adapted from: *Neitz, A., Parascandolo, G., Bauer, S., Schölkopf, B.: "Adaptive skip intervals: Temporal abstraction for recurrent dynamical models." In: Advances in Neural Information Processing Systems 31, pp. 9816–9826 (NeurIPS 2018)*. I am the main developer of idea, implementation and experiments.

in the prediction pipeline are accumulating especially when new or complex environments are considered.

Many dynamical systems have the property that long-term predictions of future states are easiest to learn if they are obtained by a sequence of incremental predictions. Our starting point is the hypothesis that at each instant of the evolution, there is an ideal *temporal step length* associated with those state transitions which are easiest to predict: Intervals which are too long correspond to complicated mechanisms that could be simplified by breaking them down into a successive application of simpler mechanisms. On the other hand, intervals which are too short do not contain much change, which means that the predictor has to represent roughly the identity – this can lead to a situation where the model makes small absolute errors δs, but a large relative error $\frac{\delta s}{\Delta t}$, which is the rate at which the prediction error accumulates. This tradeoff is illustrated in Figure 2.1. An additional drawback of too short prediction intervals is that it requires many predictions, which can be computationally expensive. Somewhere in-between the two extremes, there is an ideal step length corresponding to transitions that are easiest to represent and learn.

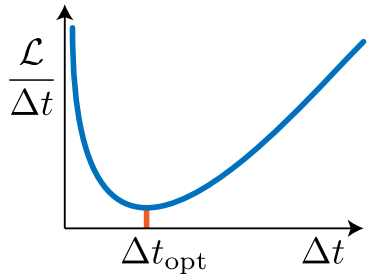


Figure 2.1: Hypothesized relationship between skip interval Δt and error accumulation rate $\frac{\mathcal{L}}{\Delta t}$. The optimum Δt_{opt} lies somewhere between the extremes, but may be state-dependent and therefore not constant across any trajectory.

We propose Adaptive Skip Intervals (ASI), a simple change to autoregressive environment simulators (Chiappa et al., 2017; Buesing et al., 2018) which can be applied to systems in which it is not necessary to predict the exact time of events. While in the literature, abstractions are often considered with respect to hierarchical components e.g. for locomotor control (Heess et al., 2016) or expanding the dynamics in a latent

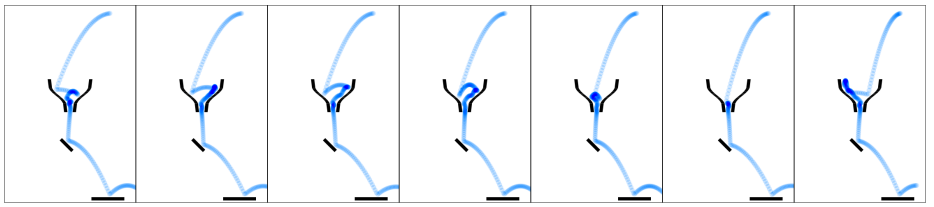


Figure 2.2: Visualization of a ball which is dropped into a funnel at different initial horizontal velocities. The part of the trajectory within the funnel can be considered *inconsequential chaos*.

space (Watter et al., 2015b), our work focuses on temporal abstractions. Our goal is to understand the dynamics of the environment in terms of robust causal mechanisms at the right level of temporal granularity. This idea is closely related to causal inference (Peters et al., 2017b) and the identification of invariances (Pearl, 2009b; Schölkopf et al., 2012; Peters et al., 2016) and mechanisms (Parascandolo et al., 2018).

ASI allows the model to dynamically adjust the temporal resolution at which predictions are made, based on the specific observed input. In other words, the model has the option to converge to the easiest-to predict transitions, with prediction intervals Δt that are not constant over the whole trajectory, but situation-dependent. Moreover, the model is more robust to certain shifts in the evolution speed at training time, and also to shifts to datasets where the trajectories are partly corrupted. For example, when some frames are missing or extremely noisy, a frame-by-frame prediction method would be forced to model the noise, especially if it is not independent of the state. Flexibly adjusting the time resolution of predictions also results in more computationally efficiency, as fewer steps need to be predicted where they are not necessary — a key requirement for real-time applications.

A type of prediction task which can especially profit from our proposed method is one which exhibits a property we call *inconsequential chaos*. To illustrate this, consider the following example: In Figure 2.2 we visualize the trajectories of a ball which falls into a funnel-shaped object at different initial horizontal velocities. The exact trajectories that

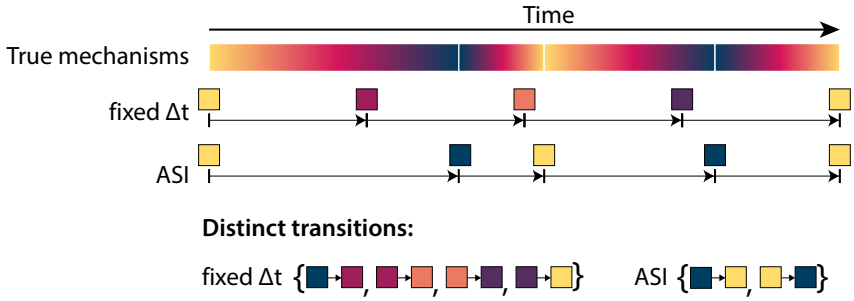


Figure 2.3: One way to motivate the need for adaptive skip intervals compared to a fixed temporal coarsening is to consider the complexity of the learned model. If the underlying true dynamics have recurring "mechanisms" which take different amounts of time, ASI enables the model to represent fewer distinct transition types, reducing the required model capacity and thus the amount of training data.

are taken within the funnel depend sensitively on the initial state and are therefore difficult to predict ahead of time. On the other hand, predicting that the ball will hit the horizontal platform on the bottom is easy because it only requires knowing that when the ball falls somewhere into the funnel, it will come out at the bottom end, irrespective of how long it bounces around. If we are only interested in predicting where the ball will ultimately land, we can skip the difficult parts, provided that they are inconsequential. Figure 2.3 explains another perspective to motivate our method.

2.1 Preliminaries

2.1.1 Problem statement

The machine learning problem we are considering is a classification problem where the labels are generated by a dynamical process, such as a Hidden Markov Model. As auxiliary data, we get access to observa-

tions of the system's internal state. The training data consists of observation sequences $\{x^{(i)}\}_{i\in 1,\dots,N}$ and labels $\{y^{(i)}\}_{i\in 1,\dots,N}$. The trajectories x are ordered sequences of elements x_t from an observation space $\mathcal{X}$. Typically, a trajectory x arises from repeatedly measuring the dynamical system's state at some fixed sampling rate. To keep the scope limited, we assume the labels $y^{(i)}$ to be categorical, i.e. belonging to a finite set $\mathcal{Y}$. In our formulation, there is only a single label for each trajectory, which intuitively corresponds to the eventual "outcome" of the particular system evolution. At test time, we are only given some initial observations $(x_0, x_1, ..., x_k)$, for some small k (e.g., $k = 0$ in the fully-observable case) and have to predict the corresponding label y.

Note that the problem does not demand the prediction of any future observations x_t. As a performance measure we use the accuracy of the label predictions. The role of the classification task is to provide a way to measure performance, as the objective is to know how well the model is suited to predict the *qualitative* outcome of each instance. We explicitly do not care about the loss in pixel space. Since frames may be skipped, video-prediction metrics are not relevant for this task. In the future we would like to use our model in latent spaces as well.

It is straightforward to generalize the classification task to a value prediction task in a (hierarchical) reinforcement learning setting, given a fixed policy (e.g. an option, as introduced in Sutton et al. (1999)). However, in this work we focus on uncontrolled tasks only.

2.1.2 Environment simulators

Environment simulators are models which approximate the conditional probability distribution

$$P(X_{t+1}, R_{t+1}|X_t) \tag{2.1}$$

where X_t is a random variable with range $\mathcal{X}$ which describes the Markovian state of the system at time t. R_t is the random variable over some real-valued *cumulant* which we want to track for our task. In order to

simplify our experiments, in this chapter we consider the special case of *fully-observable* tasks. For this reason, we use the terms "observation" and "state" interchangeably. However, note that in realistic applications, it may be desirable to predict future states given past observations, which poses the additional challenge of state inference. As an additional simplification, we consider deterministic simulators, which put a probability point mass of one on a single future state. For a recent, more detailed investigation of several efficient state-space architectures, see Buesing et al. (2018).

Note that given a distribution over an initial X_0, we can apply an environment simulator multiple times to a distribution over the initial state, yielding a probability distribution over trajectories and cumulants.

$$P(X_{0:N}, G_{0:N}) = P(X_0) \prod_{t=1}^{N} P(X_t, G_t | X_{t-1}) \tag{2.2}$$

Temporally abstract environment simulators only need to represent a relaxed version of the above conditional probability distribution:

$$P(X_{t+\tau}, R_t^\tau | X_t) \tag{2.3}$$

where τ is some arbitrary time skip interval up to the end of the trajectory, which can be chosen by the model and R_t^τ denotes the sum $\sum_{k=t}^{\tau} R_k$. In other words, a temporally abstract environment simulator must only be able to predict *some* future state of the system and additionally provide the sum of the cumulants since the last step. To address the classification problem defined in Section 2.1.1, we only consider tasks where the cumulant is zero everywhere except for the last state of the trajectory, which is a plausible restriction if the cumulant tracks some form of "outcome" of the trajectory.

The dynamical models we consider in this chapter consist at their core of a deep neural network $f : \mathcal{X} \to \mathcal{X}$ which is meant to represent the dynamical law of the environment. In order to learn to predict multiple time-steps into the future, f is iterated multiple times, which makes the architecture a recurrent neural network. As the model pre-

dicts the new state at time $t + 1$, it needs to be conditioned on the previous state at the previous time step t. During training, there is a choice for the source of the next input frame for the model: Either the ground truth (observed) frame or the model's own previous prediction can be taken. The former provides more signal when f is weak, while the latter matches more accurately the conditions during inference, when the ground truth is not known. We found the technique of scheduled sampling (Bengio et al., 2015) to be a simple and effective curriculum to address the trade-off described above. Note that other works, such as Chiappa et al. (2017) and Oh et al. (2017) have addressed the issue in different ways. The exact way of dealing with this issue is orthogonal to the use of temporal abstraction.

2.2 Method

We now introduce a method to inject temporal abstraction into deterministic recurrent environment simulators.

Training process The main idea of ASI is that the dynamical model f is not forced to predict every single time step in the sequence. Instead, it has the freedom to skip an arbitrary number of frames up to some pre-defined horizon $H \in \mathbb{N}$. We train f in such a way that it has the incentive to focus on representing those transitions which allow it to predict extended sequences which are accurate over many time steps into the future. Figure 2.4 visualizes the three steps of the ASI training procedure with a horizon of $H = 3$.

At training time, we feed the first frame x_1 into a differentiable model f, producing the output $\hat{x}_1 := f(x_1)$. In contrast to classical autoregressive modeling, $\hat{x}_1$ does not have to correspond to the next frame in the ground truth sequence, x_2, but can be matched with any frame from x_2 to x_{2+H}. Importantly, f is not required to know how many frames it is going to skip – the temporal matching is performed by a "training

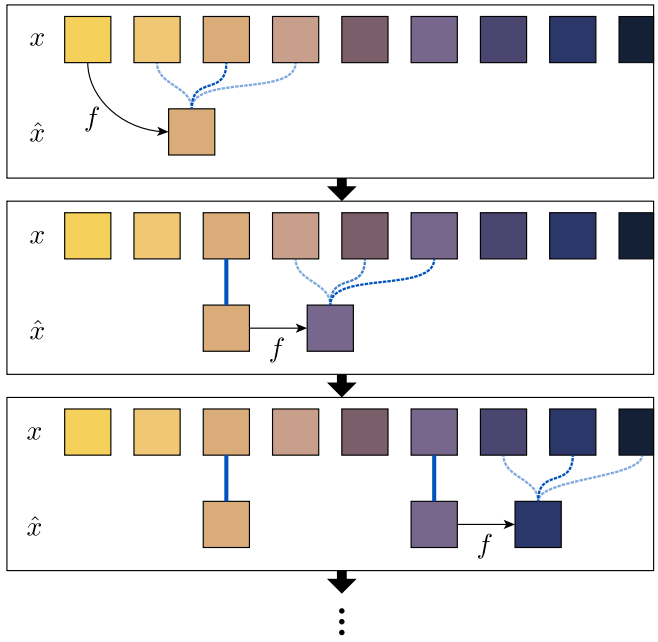


Figure 2.4: Visualization of the first three steps of ASI with a horizon of $H = 3$. The blue lines represent loss components between the ground truth frames x and predicted frames $\hat{x}$. For simplicity, we do not consider scheduled sampling here, therefore f is always applied to the previous predicted state.

supervisor" who takes f's prediction and selects the best-fitting ground-truth frame to compute the loss, which is later on reduced using gradient based optimization.

To soften the winner-takes-all mechanism, we use an *exploration-curriculum*. At every step, a Bernoulli trial with probability μ decides whether an exploration or an exploitation step is executed: In an exploration step, the supervisor selects a future frame at random with a frame-skip value between 1 and H; in an exploitation step, the supervisor takes the best-fitting ground-truth frame

$$x_i = \text{argmin}_{t \in \{2..2+H\}} \mathcal{L}_x(\hat{x}_b, x_t)$$

to provide the training signal. At the beginning of training, μ is high, such that exploration is encouraged. Over the course of several epochs,

Algorithm 1 Dynamical model learning with ASI

Require: i'th trajectory $\mathbf{x}^{(i)} = (x_1, x_2, ..., x_{T_i}) \in \mathcal{X}^{T_i}$
Require: Differentiable model $f : \mathcal{X} \to \mathcal{X}$ w/ params θ
Require: Loss function $\mathcal{L} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$
Require: Matching-horizon $H \in \mathbb{N}$
Require: Exploration schedule $\mu : \mathbb{N} \to [0, 1]$
Require: Scheduled sampling temperatures $\epsilon : \mathbb{N} \to [0, 1]$

$t \leftarrow 1, u \leftarrow 1$ ▷ *Data timestep t, abstract timestep u*
$l \leftarrow 0$ ▷ *Trajectory loss*
$x \leftarrow x_1$ ▷ *x is the next input to the dynamics model f*
while $t < |\mathbf{x}|$ **do**
    $\hat{x}_u \leftarrow f(x)$
    $T \leftarrow \min(t + H, |\mathbf{x}|)$ ▷ *Upper time step limit*
    if Bernoulli$(\mu(i)) = 0$ **then**
        $t \leftarrow \arg\min_{t' \in \{t+1..T\}} \mathcal{L}(\hat{x}_u, x_{t'})$
    else
        $t \sim \text{unif}\{t + 1, T\}$ ▷ *Exploration*
    end if
    $l \leftarrow l + \mathcal{L}(\hat{x}_u, x_t)$ ▷ *Accumulate trajectory loss*
    $x \leftarrow \text{binary\_choice}(\hat{x}_u, x_t; p = \epsilon(i))$ ▷ *Scheduled sampling (Bengio et al., 2015)*
    $u \leftarrow u + 1$
end while
$\theta \leftarrow$ gradient descent step on θ to reduce l

μ is gradually decreased such that f can converge to predicting sharp mechanisms. The goal of the exploration schedule is to avoid being caught in a local optimum early on during training. Over the course of the learning process, we gradually decrease the chance of picking a random frame, effectively transitioning to the winner-takes-all mechanism. We refer to this curriculum scheme *Exploration of temporal matching*.

The best fitting frame x_i is then fed into f again, iterating the same procedure as described above, but from a later starting point. At every step, we accumulate a loss l_x, leading to an overall *prediction loss* $\mathcal{L}_x$ which is simply the mean of all the step-losses. We train the model f via gradient descent to reduce the prediction loss $\mathcal{L}_x$.

In the example with the funnel, this could intuitively work as follows: the transition from the ball which falls into the funnel to the ball which is at the end of the funnel is the most robust one (let us call it the "robust transition") – it occurs virtually every time. All other positions within the funnel are visited less often. Therefore, f will tend to get most training signal from the robust transition. Hence, f will begin to predict something that resembles the robust transition, which will subsequently be reinforced because it will often be the best-fitting transition which wins in the matching process.

Instead of using a greedy matching algorithm it is conceivable to use a global optimization method which is applied to the whole sequence of iteratively predicted frames, which would then be aligned in the globally best possible way to the ground truth data. However, in this case, we would not be able to alternate randomly between the input sources for f, as we currently do with scheduled sampling, because in order to know which ground truth frame to take next, we already need to know the alignment.

Besides exploration of temporal matching,as mentioned in Section 2.1.2 we adopt another curriculum scheme, *scheduled sampling* (Bengio et al., 2015), which gradually shifts the training distribution from observation-dependent transitions towards prediction-dependent transitions.

Predicting the labels Since the learning procedure can choose to skip difficult-to-predict frames, the mean loss in pixel space would not be a fair metric to evaluate whether ASI serves a purpose. As explained in Section 2.1.1, one of our central assumptions is that we are dealing with environments which have the notion of a qualitative outcome, represented e.g. by the classification problem associated with the task. There-

fore, as a way to measure the learning success, we let a separate classifier $\psi : \mathcal{X} \to \mathcal{P}(\mathcal{Y})$ predict the label of the underlying classification task based on the frames predicted by f. At test time, f can unfold the dynamics over multiple steps and ψ is applied to the resulting frames, allowing the combined model to predict the label from the initial frame.

In principle, the classifier ψ could be trained alongside the model f, or after convergence of f – the two training processes do not interfere with each other. For the experiments described in Section 4.4, we hand-specify a classifier ψ ahead of time for each environment. Since our classification tasks are easy, given the last frame of a trajectory, the classifiers are simple functions which achieve perfect accuracy when fed the ground truth frames.

2.3 Experiments

We demonstrate the efficacy of our approach by introducing two environments for which our approach is expected to perform well. Code to reproduce our experiments is available at `https://github.com/neitzal/adaptive-skip-intervals`.

2.3.1 Domains

Room runner In the Room runner task, an agent, represented by a green dot, moves through a randomly generated map of rooms, which are observed in 2D from above. The agent follows the policy of always trying to move towards and into the next room, until it reaches a dead end. Two rooms are colored – the actual dead end which the agent will reach and another room, which is a dead end for another path. One of these two rooms is red, the other one blue, but the assignment is chosen by a fair coin flip. The underlying classification task is to predict whether the agent will end up in the red room or in the blue one. Since there is always exactly one passage between two adjacent rooms, the final room

is always well-defined and there is no ambiguity in the outcome. We add noise to the runner's acceleration at every step, simulating an imperfect controller – for example one which is still taking exploratory actions in order to improve. Figure 2.5 shows examples for the first states and the resulting trajectories.

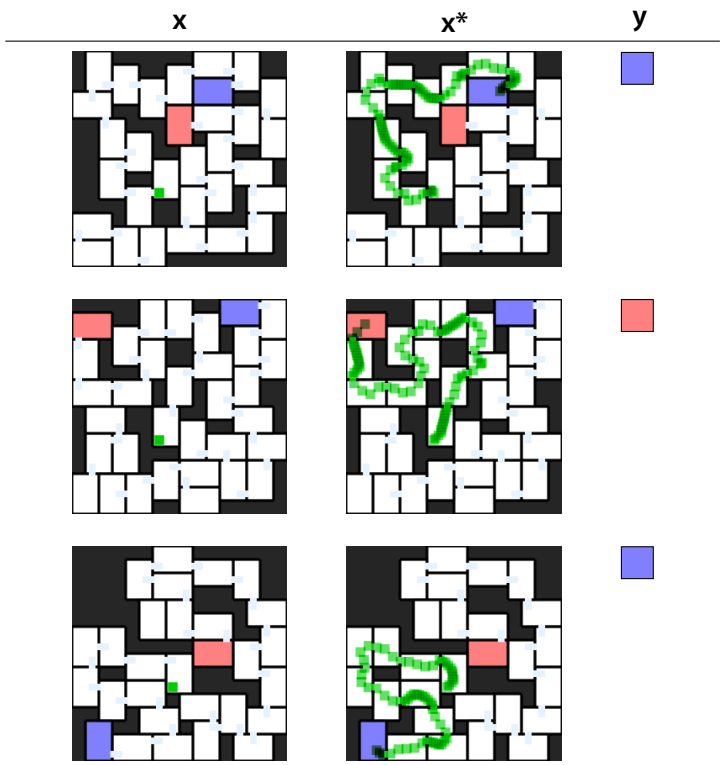


Figure 2.5: Examples of first states x of the *Room runner* domain, along with the corresponding trajectories x^* which arise from evolving the environment dynamics and the agent's policy and the correct label y. Darker regions in the trajectory correspond to parts where the agent is moving more slowly. The trajectories are merged into one image for visualization purposes only – in the dataset, every frame is separate.

Funnel board In this task, a ball falls through a grid of obstacles onto one of five platforms. Every other row of obstacles consists of funnel-

shaped objects, which are meant to capture the ball and release it at a well-defined exit position. Variety arises from the random rotations of the sliders, from the random presence or absence of funnels in every layer except for the last one, and from slight perturbations in the funnel and slider positions. The courses are generated such that the ball is always guaranteed to hit exactly one of the platforms. Figure 2.6 shows three examples for the first states and the ball's resulting paths. In order to simplify the problem, we make the states nearly fully observable by preprocessing the video frames such that they include a trace of the ball's position at the previous step.

The underlying classification task is to predict, given only access to the first frame, on which of the five platforms the ball will land eventually. Note that the task does not include predicting the time when the ball will reach its goal.

2.3.2 Experiment setup

The experiments are ablation studies of our method. We would like to investigate the efficacy of adaptive skip intervals and whether the exploration schedule is beneficial to obtain good results. For each of our two environments, we compare four methods: (a) The recurrent dynamics model with adaptive skip intervals as described in Section 2.2. (*ASI*) (b) The dynamics model with adaptive skip intervals, but without any exploration phase, i.e. $\mu = 0$. (*ASI w/o exploration*) (c) The dynamics model *without* adaptive skip intervals such that it is forced to predict every step (*fixed* ($\Delta t = 1$)). (d) The dynamics model without adaptive skip intervals such that it is forced to predict every *second* step (*fixed* ($\Delta t = 2$)). In each experiment we train with a training set of 500 trajectories, and we report validation metrics evaluated on a validation set of 500 trajectories. We perform validation steps four times per epoch in order to obtain a higher resolution in the training curves.

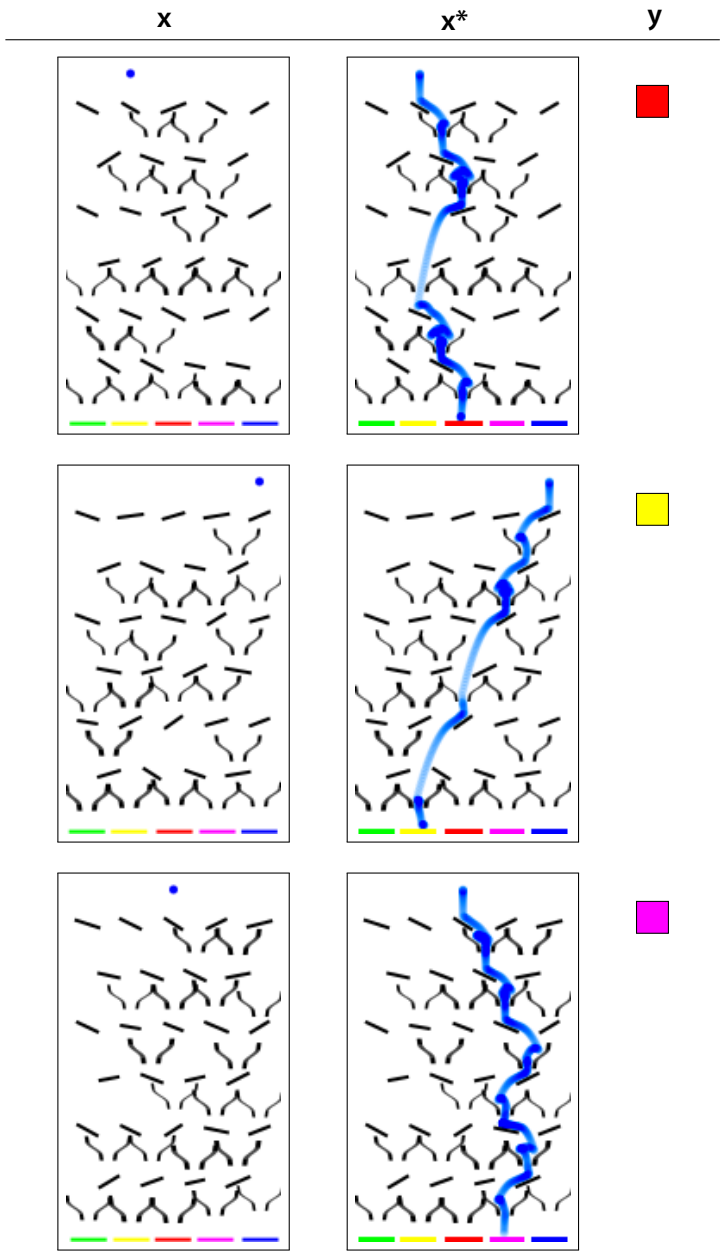


Figure 2.6: Examples of first states x of the *Funnel board* domain, along with the corresponding trajectories x^* which arise from evolving the environment dynamics, and the label y. The trajectories are merged into one image for visualization purposes only – in the dataset, every frame is separate.

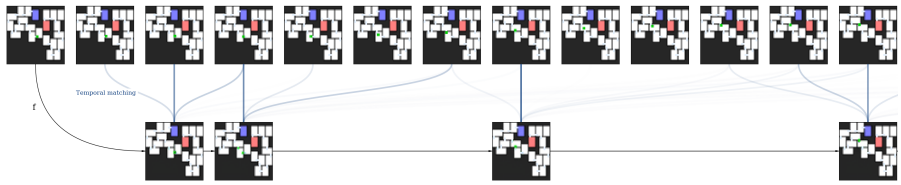


Figure 2.7: Portion of a sequence from Room runner using ASI, with ground truth frames on top and predicted, temporally aligned sequence on bottom.

For our experiments, we use a neural network with seven convolutional layers as the dynamics model f. Architectural details, which are the same in all experiments, are described in the Appendix. Like (Racanière et al., 2017), we train f using a pixel-wise binary cross entropy loss. Hyperpararameter settings such as the learning rates are determined for each method individually by using the set of parameters which led to the best result (highest maximum achieved accuracy on the validation set), out of 9 runs each. We use the same search ranges for all experiments and methods. The remaining hyperparameters, including search ranges, are provided in the Appendix. For instance, as a value for the horizon H in the ASI runs, our search yielded optimal results for values of around 20 in both experiments. After fixing the best hyperparameters, each method is evaluated 8 additional times with different random seeds, which we use to report the results. We additionally included baselines with $\Delta t > 2$, but to reduce the amount of computation did not perform another hyperparameter search for them, instead taking the best parameters for the baseline "fixed ($\Delta t = 2$)".

2.3.3 Results

We begin by visualizing how the network with adaptive skip intervals performs after training. In Figure 2.8 we show a portion of one trajectory from the Funnel board, as processed by the network. As shown, the network trained with ASI has learned to skip a variable number of frames, specifically avoiding the bouncing in the funnel, and directly

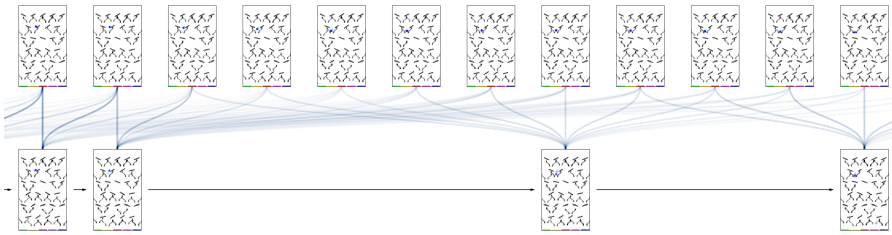


Figure 2.8: Portion of a sequence from Funnel board using ASI, with ground truth frames on top and predicted, temporally aligned sequence on bottom. Darker lines connecting a predicted frame to the ground truth frames correspond to better matching in terms of pixel loss.

predicting the exiting ball. Similarly, Figure 2.7 shows a portion of a sequence from the Room runner domain.

Quantitative results As shown in Figure 2.9, ASI outperforms the fixed-steps baselines on both datasets. On *Funnel board* the networks equipped with adaptive skip intervals achieve higher accuracy *and* in a shorter time, with exploration of adaptive skip intervals obtaining even better results. In the *Room runner* task, we observe a significant improvement of ASI with exploration over the version without exploration and the baselines. Note that some of the baselines curves get worse after an initial improvement. This can be explained by the fact that the two training curricula, scheduled sampling and exploration of temporal matching, create a nonstationary distribution for the network. We observe that ASI appears more resilient to this effect.

Computational efficiency Note that the x-axis in Figure 2.10 represents the number of forward-passes through f, which loosely corresponds to the wall clock time during the training process. Since the adaptive skip intervals methods are allowed to skip frames, they need fewer model evaluations (and therefore fewer backpropagation passes at training time) than fixed-rate training schemes. In the tasks we con-

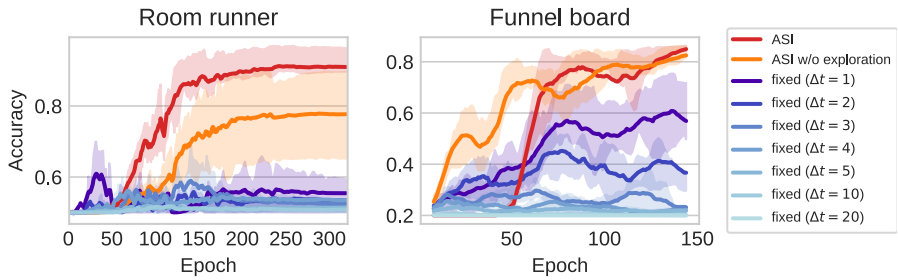


Figure 2.9: Learning progress, curves show validation accuracies on two tasks. For each task, we show on the horizontal axis the number of model evaluations and the epoch number. Curves show mean validation accuracy, evaluated on 500 trajectories. The training sets consist of 500 trajectories in each experiment. Shaded areas correspond to the interquartile range over all eight runs.

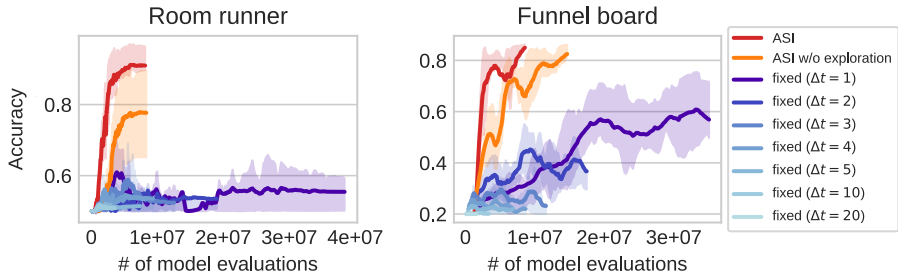


Figure 2.10: Learning progress (see Figure 2.9) with number of model evaluations on the x-axis.

sidered, not only this gain in training speed does not come at the cost of reduced accuracy, but it actually improves the overall performance.

Robustness w.r.t. perturbation of dynamics Another advantage of the temporally abstract model which we hypothesize is that the training process is more stable when the dynamical systems changes in a certain way. This is relevant because in real systems, the i.i.d. assumption is often violated. The same is true for reinforcement learning tasks, in which the distribution over observed transition changes as the agent improves its policy or due to changes in the environment over time. As a test for

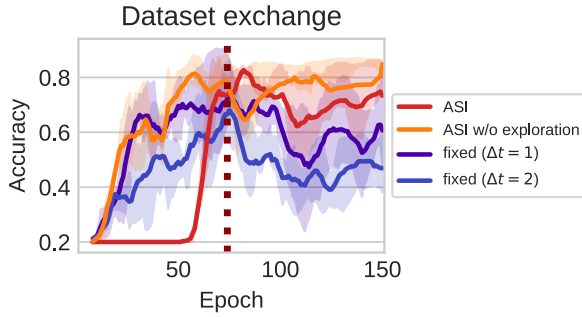


Figure 2.11: Up to epoch 75 we use a version of the Funnel board task where the funnels' bounciness is set to zero. At epoch 75 we switch the dataset for the standard one but otherwise keep the training procedure going.

our hypothesis, we prepare a second version of the Funnel board dataset with 500 trajectories of slightly altered physics: The bounciness of the funnel walls is reduced to zero. This leads to a slightly different behavior in the funnels, but the final platforms are the same in the majority of trajectories. We start with the perturbed version and before the start of the 75th epoch, we exchange it with the original one. Figure 2.11 shows the accuracy curves for this experiment. We observe that while the fixed frame-rate baselines learn the correct classification better than in the more difficult original task, after the switch the validation accuracy quickly deteriorates. Note that freezing the network at epoch 75 would leave the validation accuracy almost unchanged, since both versions of the task have similar labels.

2.4 Related work

The observation that every environment has an optimal sampling frequency has also been made for reinforcement learning. For instance, Braylan et al., 2015 investigate the effect of different frame-skip intervals on the performance of agents learning to play Atari 2600 games. A constant frame-skip value of four frames is considered standard for

Deep RL agents (Machado et al., 2018). Focusing on spatio-temporal prediction problems, (Oh et al., 2015) introduce a neural network architecture for action conditional video prediction. Their approach benefits from using curriculum learning to stabilize training of the network. Buesing et al., 2018 investigate action-conditional state-space models and explicitly consider "jumpy" models which skip a certain number of timesteps in order to be more computationally efficient. In contrast to our work they do not use adaptive skip intervals, but skip at a fixed frame rate. Belzner, 2016 introduces a time-adaptive version of model-based online planning in which the planner can optimize the step-length adaptively. Their approach focuses on temporal abstraction in the space of actions and plans. Temporal abstraction in the planning space is also a motivation of the field of hierarchical reinforcement learning (Barto and Mahadevan, 2003), often in the framework of semi-MDPs – Markov Decision Processes with temporally extended actions (e.g. Puterman, 1994).

The idea of skipping time steps has also been investigated in Ke et al. Ke et al., 2018, where the authors present a way to attack the problem of long-term credit assignment in recurrent neural networks by only propagating errors through selected states instead of every single past timestep.

Closely related to our work is the Predictron (Silver et al., 2017), which is a deep neural network architecture which is set up to perform a sequence of temporally abstract lookahead steps in a latent space. It can be trained end-to-end in order to approximate the values in a Markov Reward Process. In contrast to ASI, the outputs of the Predictron are regressed exclusively towards rewards and values, which circumvents the need for an explicit solution to the temporal alignment problem. However, by ignoring future states, the training process ignores a large amount of dynamical information from the underlying system. Similar in spirit to the Predictron, the value prediction network (VPN) (Oh et al., 2017) proposes a neural network architecture to learn a dynamics model whose abstract states make option-conditional predictions of

future values rather than of future observations. Their temporal abstraction is "grounded" by using option-termination as the skip-interval.

Ebert et al. (2017) introduced temporal skip connections for self-supervised visual planning to keep track of objects through occlusion. (Pong et al., 2018) introduce temporal difference models (TDM) which are dynamical models trained by temporal difference learning. Their approach starts with a temporally fine-grained dynamics model, which is represented with a goal-conditioned value function. The temporal resolution is successively coarsened so as to converge toward a model-free formulation. Concurrently to our work, Jayaraman et al., 2019 propose a training framework with a similar motivation to ours. They further explore ways to generalize the objective and include experiments on hierarchical planning.

2.5 Conclusion

We presented a time skipping framework for the problem of sequential predictions. Our approach builds on concepts from causal discovery (Peters et al., 2017b; Parascandolo et al., 2018) and can be included in multiple fields where planning is important. In cases where our approach fails, e.g. when the alignment of predicted and ground truth is lost and the model does not have the power to restore it, more advanced optimization methods like dynamic time warping (Müller, 2007) during the matching phase may help at the cost of the simplicity and seamless integration of the scheduled sampling, as described in Section 2.2.

An interesting direction for future work is the combination of temporal abstraction with abstractions in a latent space. As noted for instance by Oh et al., 2017, predicting future observations is a too difficult task for realistic environment due to the high dimensionality of typical observation spaces.

The idea of an optimal prediction skip interval should extend to the case of stochastic generative models, where instead of a deterministic

mapping from current to next state, the model provides a probability distribution over next states. In this case, ASI should lead to simpler distributions, allowing for simpler models and more data efficiency just as in the deterministic case. The evaluation of this claim is left for future work.

Another line of investigation which is left to future work is to integrate ASI with action-conditional models. As mentioned in Section 2.1.1, the problem could be addressed by using a separate ASI-dynamical model for each policy or option, which would make option-conditional planning possible. However, there may be a more interesting interplay between ideal skip intervals and switching points for options, which suggest that they should ideally be learned jointly.

3

Learning explanations that are hard to vary

In this chapter[1], we investigate the principle that *good explanations are hard to vary* in the context of deep learning. We show that averaging gradients across examples – akin to a logical OR ($\vee$) of patterns – can favor memorization and 'patchwork' solutions that sew together different strategies, instead of identifying invariances. To inspect this, we first formalize a notion of consistency for minima of the loss surface, which measures to what extent a minimum appears only when examples are pooled. We then propose and experimentally validate a simple alternative algorithm based on a logical AND ($\wedge$), that focuses on invariances and prevents memorization in a set of real-world tasks. Finally, using a synthetic dataset with a clear distinction between invariant and spurious mechanisms, we dissect learning signals and compare this approach to well-established regularizers.

3.1 Introduction

Consider the top of Figure 3.1, which shows a view from above of the loss surface obtained as we vary a two dimensional parameter vector $\boldsymbol{\theta} = (\theta_1, \theta_2)$, for a fictional dataset containing two observations x_A

1 Adapted from: *Parascandolo, G.*, Neitz, A.*, Orvieto, A., Gresele, L., and Schölkopf, B. (2021). "Learning explanations that are hard to vary." In: International Conference on Learning Representations. (ICLR 2021)*. I co-developed idea, implementation and experiments together with Giambattista Parascandolo.

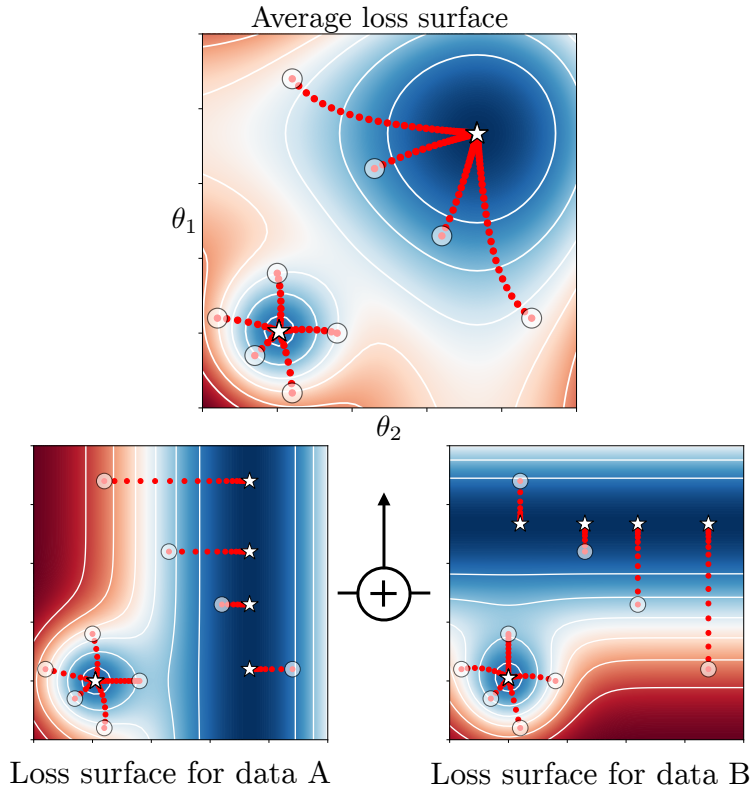


Figure 3.1: Loss landscapes of a two-parameter model. Averaging gradients forgoes information that can identify patterns shared across different environments.

and x_B. Note the two global minima on the top-right and bottom-left. Depending on the initial values of θ — marked as white circles — gradient descent converges to one of the two minima. Judging solely by the value of the loss function, which is zero in both cases, the two minima look equally good.

However, looking at the loss surfaces for x_A and x_B separately, as shown below, a crucial difference between those two minima appears: Starting from the same initial parameter configurations and following the gradient of the loss, $\nabla_\theta \mathcal{L}(\theta, x_i)$, the probability of finding the same minimum on the top-right in either case is zero. In contrast, the minimum in the lower-left corner has a significant overlap across the two

loss surfaces, so gradient descent can converge to it even if training on x_A (or x_B) only. Note that after averaging there is no way to tell what the two loss surfaces looked like: *Are we destroying information that is potentially important?*

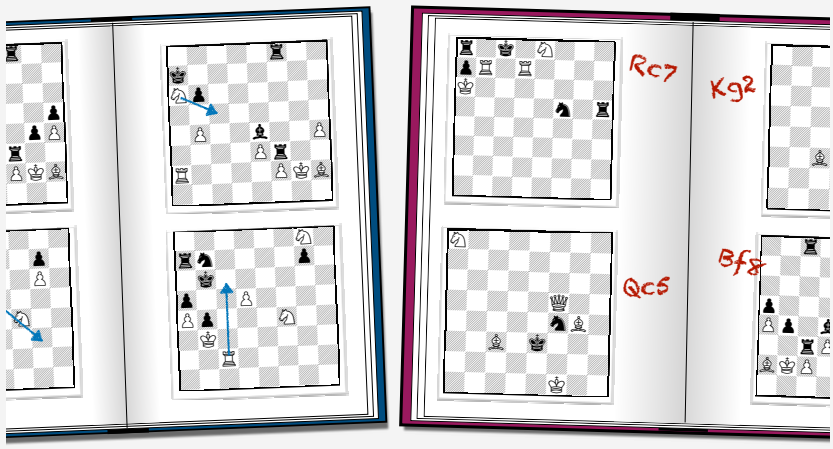
In this chapter, we argue that the answer is yes. In particular, we hypothesize that if the goal is to find *invariant mechanisms* in the data, these can be identified by finding explanations (e.g. model parameters) that are hard to vary across examples. A notion of invariance implies something that stays the same, as something else changes. We assume that data comes from different *environments*: An invariant mechanism is shared across all, generalizes out of distribution (o.o.d.), but might be hard to model; each environment also has spurious explanations that are easy to spot ('shortcuts'), but do not generalize o.o.d. From the point of view of causal modeling, such invariant mechanisms can be interpreted as conditional distributions of the targets given causal features of the inputs; invariance of such conditionals is expected if they represent *causal mechanisms*, that is — stable properties of the physical world (see e.g. Hoover, 1990). Generalizing o.o.d. means therefore that the predictor should perform equally well on data coming from different settings, as long as they share the causal mechanisms.

We formalize a notion of *consistency*, which characterizes to what extent a minimum of the loss surface appears *only* when data from different environments are pooled. Minima with low consistency are 'patchwork' solutions, which (we hypothesize) sew together different strategies and should not be expected to generalize to new environments. An intuitive description of this principle was proposed by physicist David Deutsch: *"good explanations are hard to vary"* (Deutsch, 2011).

Using the notion of consistency, we define *Invariant Learning Consistency* (ILC), a measure of the expected consistency of the solution found by a learning algorithm on a given hypothesis class. The ILC can be improved by changing the hypothesis class or the learning algorithm, and in the last part of the paper we focus on the latter. We then analyse why current practices in deep learning provide little incentive for networks to learn invariances, and show that standard training is instead set up

with the explicit objective of greedily maximizing speed of learning, i.e., progress on the training loss. When learning "as fast as possible" is not the main objective, we show we can trade-off some "learning speed" for prioritizing learning the invariances. A practical instantiation of ILC leads to o.o.d. generalization on a challenging synthetic task where several established regularizers fail to generalize; moreover, following the memorization task from Zhang et al., 2017, ILC prevents convergence on CIFAR-10 with random labels, as no shared mechanism is present, and similarly when a portion of training labels is incorrect. Lastly, we set up a behavioural cloning task based on the game CoinRun (Cobbe et al., 2019), and observe better generalization on new unseen levels.

An example.



Take these two second-hand books of chess puzzles. We can learn the two independent shortcuts (blue arrows for the left book *OR* hand-written solutions on the right), or actually learn to play chess (the invariant mechanism). While both strategies solve other problems from the same books (i.i.d.), only the latter generalises to new chess puzzle books (o.o.d.). How to distinguish the two? We would not have learned about the red arrows had we trained on the book on the right, and vice versa with the hand-written notes.

3.2 Explanations that are hard to vary

We consider datasets $\{\mathcal{D}^e\}_{e\in\mathcal{E}}$, with $|\mathcal{E}| = d$, and $\mathcal{D}^e = (x_i^e, y_i^e)$, $i_e = 1, \dots, n^e$. Here $x_i^e \in \mathcal{X} \subseteq \mathbb{R}^m$ is the vector containing the observed inputs, and $y_i^e \in \mathcal{Y} \subseteq \mathbb{R}^p$ the targets. The superscript $e \in \mathcal{E}$ indexes some aspect of the data collection process, and can be interpreted as an environment label. Our objective is to infer a function $f : \mathcal{X} \to \mathcal{Y}$ — which we call *mechanism* — assigning a target y_i^e to each input x_i^e; as explained in the introduction, we assume that such function is shared across all environments. For estimation purposes, f may be parametrized by a neural network with continuous activations; for weights $\theta \in \Theta \subseteq \mathbb{R}^n$, we denote the neural network output at $x \in \mathcal{X}$ as $f_\theta(x)$.

Gradient-based optimization. To find an appropriate model f_θ, standard optimizers rely on gradients from a *pooled* loss function

$$\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$$

. This function measures the *average* performance of the neural network when predicting data labels, across all environments:

$$\mathcal{L}(\theta) := \frac{1}{|\mathcal{E}|} \sum_{e\in\mathcal{E}} \mathcal{L}_e(\theta),$$

with

$$\mathcal{L}_e(\theta) := \frac{1}{|\mathcal{D}^e|} \sum_{(x_i^e, y_i^e)\in\mathcal{D}^e} \ell(f(x_i^e; \theta), y_i^e);$$

where $\ell : \mathbb{R}^p \times \mathbb{R}^p \to [0, +\infty)$ is usually chosen to be the $L2$ loss or the cross-entropy loss. The parameter updates according to gradient descent (GD) are given by $\theta_{\text{GD}}^{k+1} = \theta_{\text{GD}}^k - \eta\nabla\mathcal{L}(\theta_{\text{GD}}^k)$, where $\eta > 0$ is the learning rate. Under some standard assumptions (Lee et al., 2016), $(\theta_{\text{GD}}^k)_{k\geq 0}$ converges to a local minimizer of $\mathcal{L}$, with probability one.

When do we *not* learn invariances? We start by describing what might prevent learning invariances in standard gradient-based optimization.

(i) Training stops once the loss is low enough. If the optimization has learned spurious patterns by the time it converges, invariances will not be learned anymore. This depends on the rate at which different patterns are learned. The rates at which invariant patterns emerge (and vice-versa, the spurious patterns do not) can be improved by e.g.: (a) careful architecture design, e.g. as done by hardcoding spatial equivariance in convolutional networks; (b) fine-tuning models pre-trained on large amounts of data, where strong features already emerged and can be readily selected.

(ii) Learning signals: everything looks relevant for a dataset of size 1. Due to the summation in the definition of the pooled loss $\mathcal{L}$, gradients for each example are computed independently. Informally, each signal is identical to the one for an equivalent dataset of size 1, where every pattern appears relevant to the task. To find invariant patterns across examples, if we compute our training signals on each of them independently, we have to rely on the way these are aggregated.[2]

(iii) Aggregating gradients: averaging maximizes learning speed. The default method to pool gradients is the *arithmetic mean*. GD applied to $\mathcal{L}$ is designed to minimize the pooled loss *by prioritizing descent speed*.[3] Indeed, a step of GD is equivalent to finding a tight[4] quadratic upper bound $\hat{\mathcal{L}}$ to $\mathcal{L}$, and then jumping to the minimizer of this approximation (Nocedal and Wright, 2006). While speed is often desirable, by construction GD ignores one potentially crucial piece of information: The gradient $\nabla\mathcal{L}$ is the result of averaging signals $\nabla\mathcal{L}_e$, which correspond to the patterns visible from each environment at this stage of optimiza-

2 After computing the gradients for a dataset of $n-1$ examples, if an n-th example appeared, we would just compute one more vector of gradients and add it to the sum. A Gaussian Process (Rasmussen, 2003) for example would require recomputing the entire solution from scratch, as all interactions are considered.

3 The same reasoning holds for SGD in the finite-sum optimization case $\mathcal{L} = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}_i$, where gradients from a mini-batch are seen as unbiased estimators of gradients from the pooled loss. (Bottou et al., 2018).

4 Assume that $\mathcal{L}$ has L-Lipschitz gradients (i.e. curvature bounded from above by L). Then, at any point $\tilde{\theta}$, we can construct the upper bound $\hat{\mathcal{L}}_{\tilde{\theta}}(\theta) = \mathcal{L}(\tilde{\theta}) + \nabla\mathcal{L}(\tilde{\theta})^\top(\theta-\tilde{\theta}) + L\|\theta-\tilde{\theta}\|^2/2$.

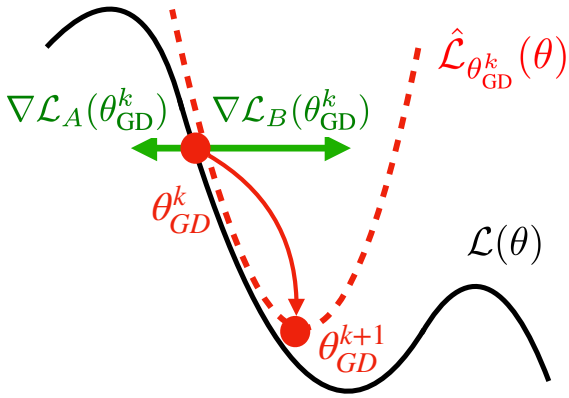


Figure 3.2: Inconsistency in gradient directions.

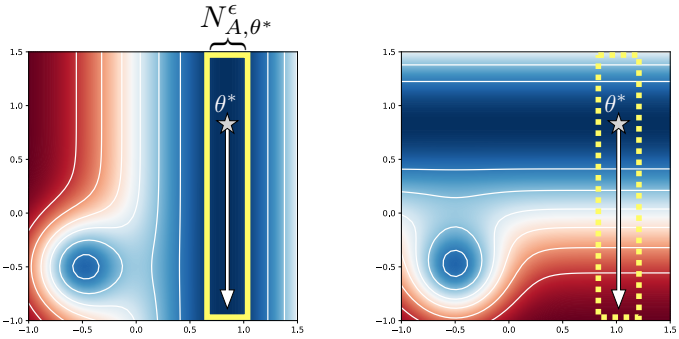
tion. In other words, GD with average gradients greedily maximizes for learning speed, but in some situations we would like to trade some convergence speed for invariance. For instance, instead of performing an arithmetic mean between gradients (logical OR), we might want to look towards a logical AND, which can be characterized as a *geometric* mean. Fig. 3.1 shows how a sum can be seen as a logical OR: the two orthogonal gradients from data A and data B at (0.5,0.5) point to different directions, yet both are kept in the combined gradient.[5] In Sec. 3.2.3 we elaborate on this idea and on implementing a logical AND between gradients. Before presenting this discussion, we take some time to better motivate the need for invariant learning consistency and to construct a precise mathematical definition of consistency.

3.2.1 Formal definition of ILC

Let $\Theta^*_{\mathcal{A}}$ be the set of convergence points of algorithm $\mathcal{A}$ when trained using all environments (pooled data): that is,

$$\Theta^*_{\mathcal{A}} = \{\theta^* \in \Theta \mid \exists\, \theta^0 \in \mathbb{R}^n \text{ s.t. } \mathcal{A}_\infty(\theta^0, \mathcal{E}) = \theta^*\}.$$

5 Loosely speaking, a sum is large if any of the summands is large, a product is large if all factors are large.



Loss surface for data A Loss surface for data B

For instance, if $\mathcal{A}$ is gradient descent, the result of Lee et al., 2016 implies that $\Theta^*_{\mathcal{A}}$ is the set of local minimizers of the pooled loss $\mathcal{L}$. To each $\theta^* \in \Theta^*_{\mathcal{A}}$, we want to associate a consistency score, quantifying the concept "*good* θ^* *are hard to vary*". In other words, we would like the score to capture the consistency of the loss landscape around θ^* across the different environments. For example, in Fig. 3.1 the loss landscape near the bottom-left minimizer is consistent across environments, while the top-right minimizer is not. Let us characterize the landscape around θ^* from the perspective of a fixed environment $e \in \mathcal{E}$. We define the set $N^{\epsilon}_{e,\theta^*}$ to be the largest path-connected region of space containing both θ^* and the set $\{\theta \in \Theta \text{ s.t.} |\mathcal{L}_e(\theta) - \mathcal{L}_e(\theta^*)| \leq \epsilon\}$, with $\epsilon > 0$. In other words, if $\theta \in N^{\epsilon}_{e,\theta^*}$ then there exist a path-connected region in parameter space including θ^* and θ where each parameter also is in $N^{\epsilon}_{e,\theta^*}$ and its loss on environment e is comparable. From the perspective of environment e, all these points are equivalent to θ^*. We would like to evaluate the elements of this set with respect to a different environment $e' \neq e$. We will say that e' is consistent with e in θ^* if $\max_{\theta \in N^{\epsilon}_{e,\theta^*}} |\mathcal{L}_{e'}(\theta) - \mathcal{L}_e(\theta)|$ is small. Repeating this reasoning for all environment pairs, we arrive at the following *inconsistency score*:

$$\mathcal{J}^{\epsilon}(\theta^*) := \max_{(e,e') \in \mathcal{E}^2} \max_{\theta \in N^{\epsilon}_{e,\theta^*}} |\mathcal{L}_{e'}(\theta) - \mathcal{L}_e(\theta^*)|. \tag{3.1}$$

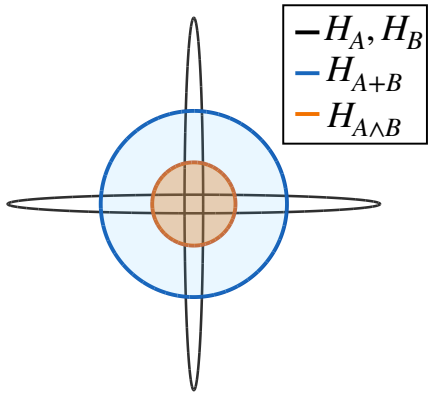


Figure 3.3: Plotted are contour lines $\theta^\top H^{-1}\theta = 1$ for $H_A = \text{diag}(0.05, 1)$ and $H_B = \text{diag}(1, 0.05)$. $H_{A\wedge B}$ retains the original volumes, while for H_{A+B} it is $5\times$ bigger. This magnification shows inconsistency of A and B.

This consistency is our formalization of the principle "*good explanations are hard to vary*". Finally, we can write down an invariant learning consistency score for $\mathcal{A}$:

$$\text{ILC}(\mathcal{A}, p_{\theta^0}) := -\mathbb{E}_{\theta^0 \sim p(\theta^0)} \left[\mathcal{I}^\epsilon(\mathcal{A}_\infty(\theta^0, \mathcal{E})) \right]. \tag{3.2}$$

That is, the learning consistency of an algorithm measures the expected consistency across environments of the minimizer it converges to on the pooled data.

Example: low consistency of a classic patchwork solution. One-hidden-layer networks with sigmoid activations and enough neurons can approximate any function $f^* : [0, 1] \to \mathbb{R}$ (Cybenko, 1989). In appendix B.1.1 we show how the construction used to obtain the weights leads to a maximally inconsistent solution according to $\mathcal{I}^\epsilon(\theta^*)$, which would not be expected to generalize o.o.d.

3.2.2 ILC as a logical AND between landscapes

Here we draw a connection between our definition of inconsistency and the local geometric properties of the loss landscapes. For the sake of

clarity, we consider two environments (A and B) and assume θ^* to be a local minimizer (with zero loss) for both environments. Using a Taylor approximation[6], we get

$$\mathcal{L}(\theta) \approx \frac{1}{2}(\theta - \theta^*)^\top H_{A+B}(\theta - \theta^*)$$

for $\|\theta - \theta^*\| \approx 0$, where $H_{A+B} = (H_A + H_B)/2$ is the *arithmetic mean* of the Hessians $H_A := \nabla^2 \mathcal{L}_A(\theta^*)$ and $H_B := \nabla^2 \mathcal{L}_A(\theta^*)$. The mean Hessian H_{A+B} does not capture the possibly conflicting geometries of landscape A or B: It performs a "logical OR" on the dominant eigendirections. In contrast, the *geometric mean*, or Karcher mean, $H_{A \wedge B}$ (Ando et al., 2004) is affected by the inconsistencies between landscapes: It performs a "logical AND". In appendix B.1.2, we give a formal definition of $H_{A \wedge B}$, and show that for diagonal Hessians,

$$\mathcal{J}^\epsilon(\theta^*) \leq 2\epsilon\left(\frac{\det(H_{A+B})}{\det(H_{A \wedge B})}\right)^2. \tag{3.3}$$

As for the geometric mean of positive numbers,

$$0 \leq \det(H_{A \wedge B}) \leq \det(H_{A+B}); \tag{3.4}$$

thus, inconsistency is lowest when *shapes* of A and B are similar – exactly as in the bottom-left minimizer of Fig. 3.1.

From Hessians to gradients. We just saw that the consistency of θ^* is linked to the geometric mean of the Hessians $\{H_e(\theta^*)\}_{e \in \mathcal{E}}$. Under the simplifying assumption that each H_e is diagonal[7] and all eigenvalues λ_i^e are positive, their geometric mean is

$$H^\wedge := \operatorname{diag}\left(\left(\prod_{e \in \mathcal{E}} \lambda_1^e\right)^{\frac{1}{|\mathcal{E}|}}, \ldots, \left(\prod_{e \in \mathcal{E}} \lambda_n^e\right)^{\frac{1}{|\mathcal{E}|}}\right). \tag{3.5}$$

6 This provides a useful simplified perspective. Indeed, this *quadratic model* is heavily used in the optimization community (see e.g. Jastrzębski et al., 2017; Zhang et al., 2019a; Mandt et al., 2017.)

7 It was shown in (Becker, Le Cun, et al., 1988) and recently in (Adolphs et al., 2019; Singh and Alistarh, 2020) that neural networks have a strong diagonal dominance of the Hessian matrix at the end of training.

The curvature of the corresponding loss in the i-th eigendirection depends on how consistent the curvatures of each environment are in that direction. Consider now optimizing from a point θ^k; gradient descent reads

$$\theta^{k+1} = \theta^k - \eta H^+(\theta^k - \theta^*)$$

where

$$H^+ := \text{diag}(\frac{1}{|\mathcal{E}|}\sum_{e\in\mathcal{E}}\lambda_1^e, \dots, \frac{1}{|\mathcal{E}|}\sum_{e\in\mathcal{E}}\lambda_n^e).$$

For η small enough[8], we have

$$|\theta_i^{k+1} - \theta_i^*| = (1 - \eta\frac{1}{|\mathcal{E}|}\sum_{e\in\mathcal{E}}\lambda_i^e)|\theta_i^k - \theta_i^*|.$$

As noted, this choice maximises the speed of convergence to θ^*, but does not take into account whether this minimizer is consistent. We can reduce the speed of convergence on directions where landscapes have different curvatures – which would lead to a high inconsistency – by following the gradients from the geometric mean of the landscapes, as opposed to the arithmetic mean. I.e, we substitute the full gradient

$$\nabla\mathcal{L}(\theta) = H^+(\theta^k - \theta^*)$$

with

$$\nabla\mathcal{L}^\wedge(\theta) = H^\wedge(\theta^k - \theta^*).$$

Also, we have that[9]

$$\nabla\mathcal{L}^\wedge(\theta) = \left(\prod_{e\in\mathcal{E}}\nabla\mathcal{L}_e(\theta)\right)^{1/|\mathcal{E}|}.$$

To reduce the speed of convergence in directions with inconsistency, we can take the element-wise geometric mean of gradients from different environments (see also Fig. B.2 in the appendix).

8 Smaller than $1/\lambda_{\text{max}}$, λ_{max} is the maximum eigenvalue of Hessians from different environments,

9 This holds if $\theta - \theta^*$ is positive, otherwise we have $\nabla\mathcal{L}^\wedge(\theta) = -\left(\prod_{e\in\mathcal{E}}|\nabla\mathcal{L}_e(\theta)|\right)^{1/|\mathcal{E}|}$.

3.2.3 Masking gradients with a logical AND

The element-wise geometric mean of gradients, instead of the arithmetic mean, increases consistency in the convex quadratic case. However, there are a few practical limitations:

(i) The geometric mean is only defined when all the signs are consistent. It is still to be defined how sign inconsistencies, which can occur in non-convex settings, should be dealt with.

(ii) It provides little flexibility for 'partial' agreement: Even a single zero gradient component in one environment stops optimization in that direction.

(iii) For numerical stability, it needs to be computed in log domain (more computationally expensive).

(iv) Adaptive step-size optimizers (e.g. Adam (Kingma and Ba, 2015)) rescale the signal component-wise for local curvature adaptation. The exact magnitude of the geometric mean would be ignored and most of the difference from arithmetic averaging will come from the zero-ed components.

(i) can be overcome by treating different signs as zeros, resulting in a geometric mean of 0 if there is any sign disagreement across environments for a gradient component. For (ii) we can allow for *some* disagreement (with a hyperparameter), by not masking out if there is a large percentage of environments with gradients in that direction. (iii) and (iv) can be addressed together: Since the final magnitude will be rescaled except for masked components, i.e. where the geometric mean is 0, we can use the average gradients (fast to compute) and mask out the components based on the sign agreement (computable avoiding the log domain).

The AND-mask. We translate the reasoning we just presented to a practical algorithm that we will refer to as the *AND-mask*. In its most simple implementation, we zero out those gradient components with respect to weights that have *inconsistent signs* across environments.

Formally, the masked gradients at iteration k are $m_t(\theta^k) \odot \nabla\mathcal{L}(\theta^k)$, where $m_t(\theta^k)$ vanishes for any component where there are less than $t \in \{d/2, d/2+1, \dots, d\}$ agreeing gradient signs across environments (d is the number of environments in the batch), and is equal to one otherwise. For convenience, our implementation of the AND-mask uses a *threshold* $\tau \in [0,1]$ as hyper-parameter instead of t, such that $t = \frac{d}{2}(\tau+1)$. Mathematically, for every component $[m_\tau]_j$ of m_τ,

$$[m_\tau]_j = \mathbb{1}\left[\tau d \leq |\sum_e \text{sign}([\nabla\mathcal{L}_e]_j)|\right].$$

Computing the AND-mask has the same time and space complexity of standard gradient descent, i.e., linear in the number of examples that we average. Due to its simplicity and computational efficiency, this is the algorithm that we will use in the experiment section. As a first result, we show that following the AND-masked gradient leads to convergence in the directions made visible by the AND-mask. The proof is presented in appendix B.1.3.

Proposition 1. Let $\mathcal{L}$ have L-Lipschitz gradients and consider a learning rate $\eta \leq 1/L$. After k iterations, AND-masked GD visits at least once a point θ where $\|m_t(\theta) \odot \nabla\mathcal{L}(\theta)\|^2 \leq \mathcal{O}(1/k)$.

Behaviour in the face of randomness. Here we put the AND-mask through a theoretical test: For gradients coming from different environments that are inconsistent (or even random), how fast does the AND-mask reduce the magnitude of the step taken in parameter space, compared to standard GD? *In case of inconsistency, the AND-mask should quickly make the gradient steps more conservative.*

To assess this property, we consider a fixed set of n parameters θ and gradients $\nabla\mathcal{L}_e$ drawn independently from a multivariate Gaussian with zero mean and unit covariance.

Proposition 2. Consider the setting we just outlined, with

$$\mathcal{L} = (1/d)\sum_{e=1}^{d}\mathcal{L}_e$$

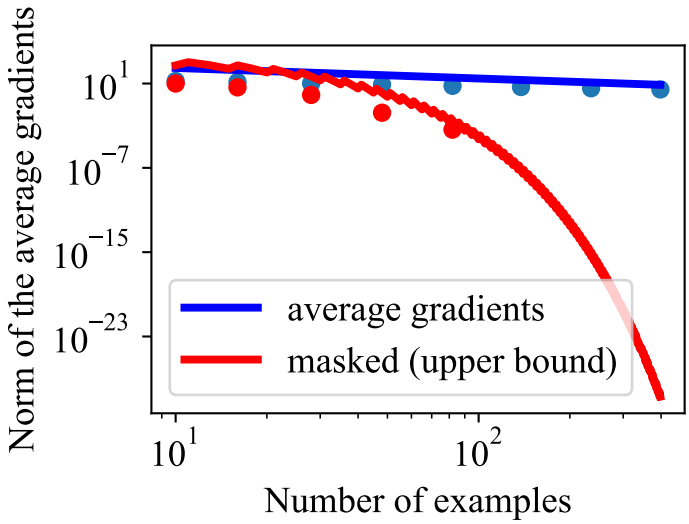


Figure 3.4: Magnitude of gradient (average or masked) on random data ($|\theta| = 3000$, $t = 0.8d$).

. While $\mathbb{E}\|\nabla\mathcal{L}(\theta)\|^2 = \mathcal{O}(n/d)$, we have that

$$\forall t \in \{d/2+1, \dots, d\}, \exists c \in (1,2] \text{s.t.} \mathbb{E}\|m_t(\theta) \odot \nabla\mathcal{L}(\theta)\|^2 \leq \mathcal{O}(n/c^d).$$

The proof is presented in Appendix B.1.4, and an illustration with numerical verification in Fig. 3.4 (the magnitudes of masked gradients (•) for more than 100 examples were always zero in the numerical verification). Intuitively, in the presence of purely random patterns, the AND-mask has a desirable property: it decreases the strength of these signals exponentially fast, as opposed to linearly.

3.3 Experiments

Real-world datasets are generated by (causal) generative processes which share mechanisms (Pearl, 2009a). However, mechanisms and spurious signals are often entangled, making it hard to assess what part of the learning signal is due to either. As the goal of this chapter is to dissect

these two components to understand how they ultimately contribute to the learning process, we create a simple synthetic dataset that allows us to control the complexity, intensity, and number of shortcuts in the data. After that, we evaluate whether spurious signals can be detected even in high-dimensional networks and datasets by testing the AND-mask on a memorization task similar to the one proposed in Zhang et al., 2017, and on a behavioral cloning task using the game CoinRun (Cobbe et al., 2019).

3.3.1 The synthetic memorization dataset

We introduce a binary classification task. The input dimensionality is $d = d_M + d_S$. While $p(y|x_{d_M})$ is the same across *all* environments (i.e. the *mechanism*), $p(y|x_{d_S}, e)$ is *not the same* across all environments (the *shortcuts*). While the mechanism is shared, it needs a nonlinear decision boundary to classify the data. The shortcuts are not shared across environments, but provide a simple way to classify the data, even when pooling all the environments together. See Figure 3.5 for a concrete example with d_M and d_S equal to 2, and two environments (A and B). The spirals (on d_M) are invariant but hard to model. The shortcuts (on d_S) are simple blobs but different in every environment: in A, linearly separable through a vertical decision boundary, in B with a horizontal one. If the two environments are pooled, a new diagonal decision boundary emerges on the shortcut dimensions as the most 'natural' one. While this perfectly classifies data in both environments A and B, critically *it would have not been found by training on either partition A or B alone*. The out-of-distribution (o.o.d.) test data has the same mechanism but random shortcuts. Therefore, any method relying exclusively on the shortcuts will have chance-level o.o.d. performance. Details about the dataset, baselines, and training curves are reported in appendix B.2.

Despite the apparent simplicity of this dataset, note that it is challenging to find the invariant mechanism. In high dimensions, even with tens of pooled environments, the shortcuts allow for a *simple* classification

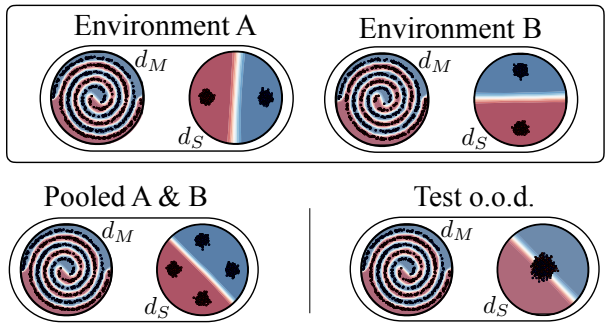


Figure 3.5: A 4-dimensional instantiation of the synthetic memorization dataset for visualization. Every example is a dot in both circles, and it can be classified by finding either of the "oracle" decision boundaries shown.

rule under almost every classical definition of 'simple': the boundary is *linear*, it has a *large margin*, it can be expressed with *small weights*, it is *fast to learn*, robust to input noise, and has *perfect accuracy and no i.i.d. generalization gap*. Finding the complex decision boundary of the spirals, instead, is a fiddly process and arguably a much slower path towards small loss.

Baselines. We evaluate several domain-agnostic baselines (all multi-layer perceptrons) with some of the most common regularizers used in deep learning — Dropout, L1, L2, Batch normalization. We also consider methods that explicitly make use of the environment labels:

(i) Domain Adversarial Neural Networks (DANN) (Ganin et al., 2016), a method specifically designed to address domain adaptation by obfuscating domain information with an adversarial classifier;

(ii) Invariant Risk Minimization (IRM) (Arjovsky et al., 2019), discussed in detail in appendix B.2. The AND-mask is trained with the same configurations in Table B.1.

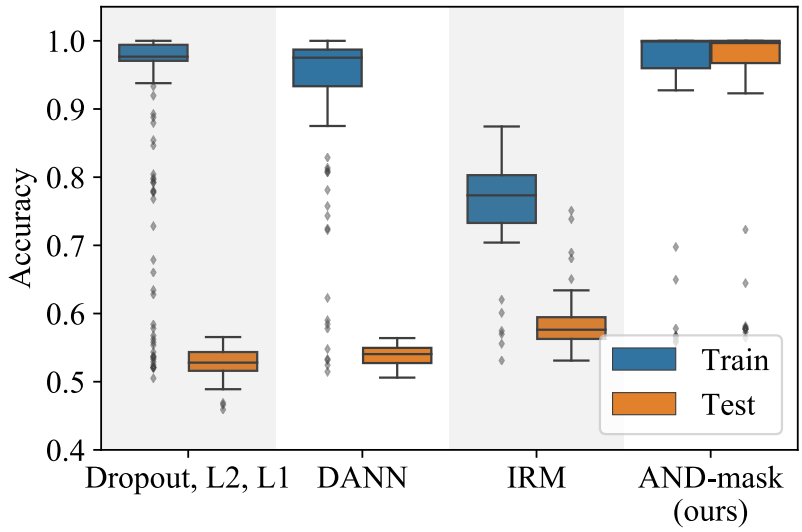


Figure 3.6: Results on the synthetic dataset.

Results. Fig. 3.6 shows training and test accuracy. DANN fails because it can align the representation-layer distributions from different environments using only shortcuts, such that they become indistinguishable to the domain-discriminating classifier. The AND-mask was the only method to achieve perfect test accuracy, by fitting the spirals instead of the shortcuts. In particular, the combination of the AND-mask with L1 or L2 regularization gave the most robust results overall, as they help suppress neurons that at initialization are tuned towards the shortcuts.

Correlations between average, memorization and generalization gradients. Due to the synthetic nature of the dataset, we can intervene on its data-generating process in order to examine the learning signals coming from the mechanisms and from the shortcuts. We isolate the two and measure their contribution to the average gradients, as we vary the agreement threshold of the mask. More precisely, we look at the gradients computed with respect to the weights of a randomly initial-

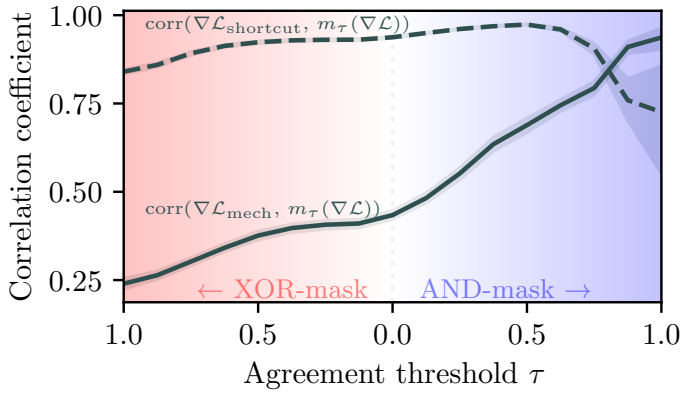


Figure 3.7: Gradient correlations.

ized network for different sets of data: (i) The original data, with mechanisms and shortcuts. (ii) Randomly permuting the dataset over the mechanisms dimensions, thus leaving the "memorization" signal of the shortcuts. (iii) Randomly permuting over the shortcuts dimensions, isolating the "generalization" signal of the mechanisms alone. Figure 3.7 shows the correlation between the components of the original average gradient (i) and the shortcut gradients ((ii), dashed line), and between the original average gradients and the mechanism gradients ((iii), solid line). While the signal from the mechanisms is present in the original average gradients (i.e. $\rho \approx 0.4$ for $\tau = 0$), its magnitude is smaller and it is 'drowned' by the memorization signal. Instead, increasing the threshold of the AND-mask (right side) suppresses memorization gradients due to the shortcuts, and for $\tau \approx 1$ most of the gradient components remaining contain signal from the mechanism. On the left side, we test the other side of our hypothesis: An XOR-mask zeroes out consistent gradients, preserves those with different signs, and results in a sharper decrease of the correlation with the mechanism gradients.

3.3.2 Experiments on CIFAR-10

Memorization in a vision task. Zhang et al., 2017 showed that neural networks trained with standard regularizers — like L2 and Dropout — can still memorize large training datasets with *shuffled* labels, i.e. reaching ≈100% training accuracy. Their experiments raised significant questions about the generalization properties of neural networks and the role of regularizers in constraining the hypothesis class. Our hypothesis is that ILC — for example implemented as the AND-mask — should prevent memorization on a similar task with the shuffled labels, as gradients will tend to largely 'disagree' in the absence of a shared mechanism. However, when the labels are *not shuffled*, ILC should have a much weaker effect, as real shared mechanisms are still present in the data.

To test our hypothesis, we ran an experiment that closely resembles the one in (Zhang et al., 2017) on CIFAR-10. We trained a ResNet on CIFAR-10 with *random labels*, with and without the AND-mask. In all experiments we used batch size 80, and treated each example as its own "environment". Recall that standard gradient averaging is equivalent to an AND-mask with threshold 0. As shown in Figure 3.8, the ResNet with standard average gradients memorized the data, while slightly increasing the threshold for the AND-mask quickly prevented memorization (dark blue line). In contrast, training the same networks on the dataset *with the original labels* resulted in both of them converging and generalizing to the test set, confirming that the mask did not significantly affect the generalization error with a general underlying mechanism in the data.

Note that there is no standard notion of environments in CIFAR-10, which is why we treated every example as coming from its own environment. This assumption is not unreasonable, as every image in the dataset was literally collected in a different physical environment. If anything, it is the standard i.i.d. assumption that hides this variety behind a notion of a single distribution encompassing all environments. The results of this experiment further support this interpretation, and can

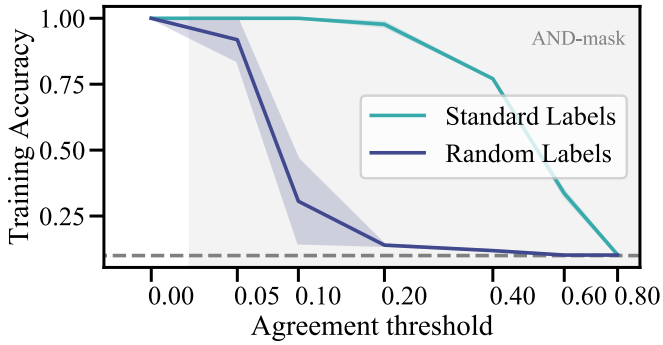


Figure 3.8: As the AND-mask threshold increases, memorization on CIFAR-10 with random labels is quickly hindered.

serve as evidence that — in some cases — we might be able to identify invariances even without an explicit partition into environments, as this can be already identified at the level of individual examples.

Label noise. Following up on this experiment, we test how the AND-mask performs in the presence of label noise, i.e. when a portion of the labels in the training set are randomly shuffled (25% here). According to our hypothesis, gradients computed on examples with random labels should disagree and get masked out by the AND-mask, while signal from correctly labeled data should contribute to update the model. As shown in Figure 3.9, the performance on the *incorrectly* labeled portion of the dataset is well *below* chance for the AND-mask (as it predicts correctly despite the wrong labels), while the baseline again memorizes the incorrect labels. On the test set (with untouched labels), the baseline peaks early then decreases as the model overfits, while the AND-mask slowly but steadily improves.

3.3.3 Behavioral Cloning on CoinRun

CoinRun (Cobbe et al., 2019) is a game introduced to test how RL agents generalize to novel situations. The agent needs to collect coins, jumping

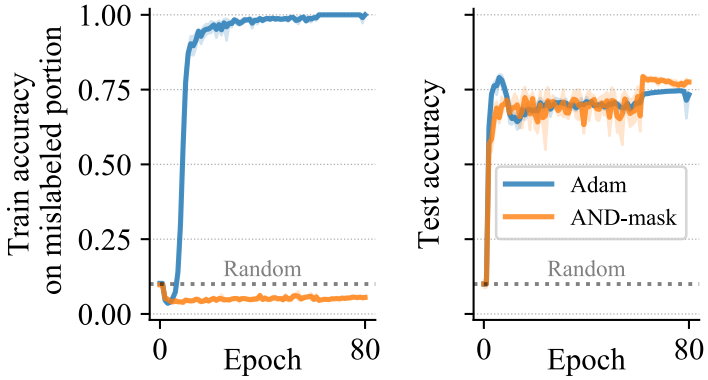


Figure 3.9: The AND-mask prevents overfitting to the incorrectly labeled portion of the training set (left) without hurting the test accuracy (right).

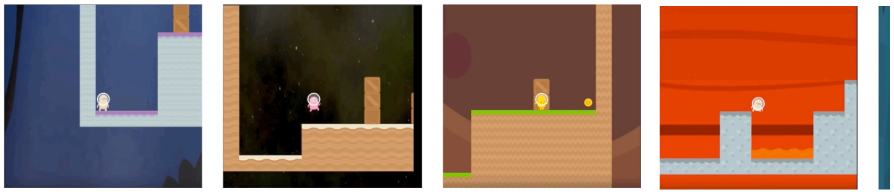


Figure 3.10: Screenshots of four levels of CoinRun (from OpenAI).

on top of walls and boxes and avoiding enemies.[10] Each level is procedurally generated — i.e. it has a different combination of sprites, background, and layout — but the physics and goals are invariant. Cobbe et al., 2019 showed that state-of-the-art RL algorithms fail to model these invariant mechanisms, performing poorly on new levels unless trained on thousands of them.

To test our hypothesis, we set up a behavioral cloning task using CoinRun.[11] We start by pre-training a strong policy π^* using standard PPO (Schulman et al., 2017) for 400M steps on the full distribution of levels.

10 See Figure 3.10 in appendix B.2.6 for a visualization of the game.

11 To obtain a robust evaluation, we preferred to approach behavioral cloning instead of the full RL problem, as it is a standard supervised learning task and has substantially fewer moving parts than most deep RL algorithms.

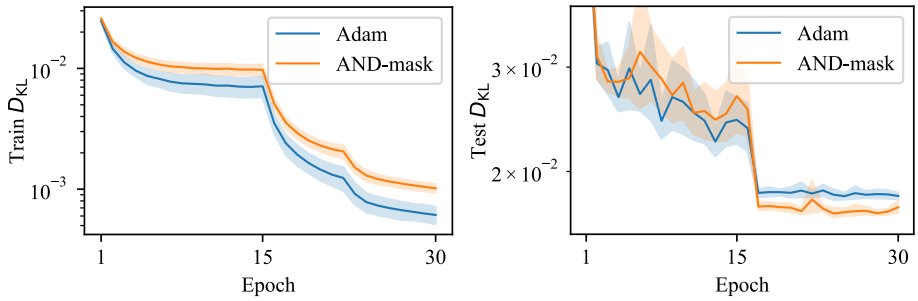


Figure 3.11: Learning curves for the behavioral cloning experiment on CoinRun. Training loss is shown on the left, test loss is shown on the right. We show the mean over the top-10 runs for each method. The shaded regions correspond to the 95% confidence interval of the mean based on bootstrapping.

We then generate a dataset of pairs $(s, \pi^*(a|s))$ from the on-policy distribution.

The training data consists of 1000 states from each of 64 levels, while test data comes from 2000 levels. A ResNet-18 $\hat{\pi}_\theta$ is then trained to minimize the loss $D_{\mathrm{KL}}(\pi^*||\hat{\pi}_\theta)$ on the training set. We compare the generalization performance of regular Adam to a version that uses the AND-mask. For each method we ran an automatic hyperparameter optimization study using Tree-structured Parzen Estimation (Bergstra et al., 2013) of 1024 trials.

Despite the theoretical computational efficiency of computing the AND-mask as presented in Section 3.2.3 (i.e., linear time and memory in the size of the mini-batch, just like classic SGD), current deep learning frameworks such as PyTorch (Paszke

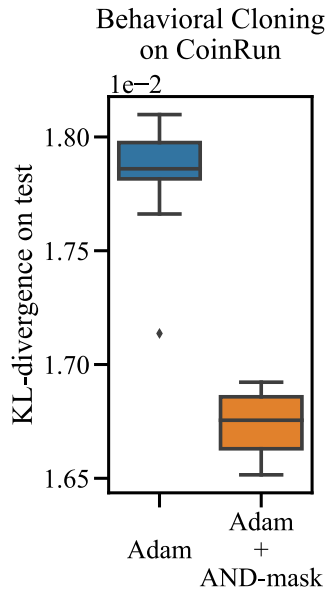


Figure 3.12: CoinRun results

et al., 2017) have optimized routines that sum gradients across examples in a mini-batch before it is possible to efficiently compute the AND-mask. We therefore test the AND-mask in a slightly different way. In training, in each iteration we sample a batch of data from a randomly chosen level out of the 64 available (and cycle through them all once per epoch). We then apply the AND-mask 'temporally', only allowing gradients that are consistent across time (and therefore across levels). See Algorithm 4 in appendix B.2.6 for a detailed description of this alternative formulation of the AND-mask. Figure 3.12 shows the minimum test loss for the 10 best runs, supporting the hypothesis that the AND-mask helps identify invariant mechanisms across different levels.

3.4 Related Work

Generalization and covariate shift. The classic formulation of statistical learning theory (Vapnik, n.d.) concerns learning from independent and identically distributed samples. The case where the distribution of the covariates at test time differs from the one observed during training is termed *covariate shift* (Sugiyama et al., 2007; Quionero-Candela et al., 2009; Sugiyama and Kawanabe, 2012). Standard solutions involve re-weighting of the training examples, but require the additional assumption of overlapping supports for train and test distributions.

Causal models and invariances. As we mentioned in the Introduction, causality provides a strong motivation for our work, based on the notion that statistical dependencies are epiphenomena of an underlying causal model (Pearl, 2009a; Peters et al., 2017a). The causal description identifies stable elements – e.g. physical *mechanisms* – connecting causes and effects, which are expected to remain *invariant* under interventions or changing external conditions (Haavelmo, 1943; Schölkopf et al., 2012)). This motivates our notion of invariant mechanisms, and inspired related notions which have been proposed for robust regression (Rojas-Carulla et al., 2018; Heinze-Deml et al., 2018; Arjovsky et

al., 2019; Hermann and Lampinen, 2020; Ahuja et al., 2020; Krueger et al., 2020). We discuss this in more detail in appendix B.3.1.

Domain generalization. ILC can be used in a setting of domain generalization (Muandet et al., 2013), but it is not limited to it: as demonstrated in the experiments in Section 3.3.2, the AND-mask can be applied even if domain labels are not available. In contrast, by treating every example as a single domain, methods relying on domain classifiers (like DANN Ganin et al., 2016 or Balaji et al. (2018)) would require as many output units as there are training examples (i.e. 50'000 for CIFAR-10).

Gradient agreement. Looking at gradient agreement to learn meaningful representations in neural networks has been explored in (Du et al., 2018; Eshratifar et al., 2018; Fort et al., 2019; Zhang et al., 2019b). These approaches mainly rely on a measure of cosine similarity between gradients, which we did not consider here for two main reasons: *(i)* It is a 'global' property of the gradients, and it would not allow us to extract precise information about different patterns in the network; *(ii)* It is unclear how to extend it beyond pairs of vectors, and for pairwise interactions its computational cost scales quadratic in the number of examples used.

3.5 Conclusions

Generalizing out of distribution is one of the most significant open challenges in machine learning, and relying on invariances across environments or examples may be key in certain contexts. In this chapter we analyzed how neural networks trained by averaging gradients across examples might converge to solutions that ignore the invariances, especially if these are harder to learn than spurious patterns. We argued that if learning signals are collected *on one example at the time* — as it is the case for gradients, e.g., computed with backpropagation — the way these signals are aggregated can play a significant role in the patterns that will ultimately be expressed: Averaging gradients in particular can

be too permissive, acting as a *logical OR* of a collection of distinct patterns, and lead to a 'patchwork' solution. We introduced and formalized the concept of Invariant Learning Consistency, and showed how to learn invariances even in the face of alternative explanations that — although spurious — fulfill most characteristics of a good solution. The AND-mask is but one of multiple possible ways to improve consistency, and it is unlikely to be a practical algorithm for all applications. However, we believe this should not distract from the general idea which we are trying to put forward — namely, that it is worthwhile to study learning of explanations that are *hard to vary*, with the longer term goal of advancing our understanding of learning, memorization and generalization.

4

Learning to distill trajectories

By[1] learning to predict trajectories of dynamical systems, model-based methods can make extensive use of all observations from past experience. However, due to partial observability, stochasticity, compounding errors, and irrelevant dynamics, training to predict observations explicitly often results in poor models. Model-free techniques try to sidestep the problem by learning to predict values directly. While breaking the explicit dependency on future observations can result in strong performance, this usually comes at the cost of low sample efficiency, as the abundant information about the dynamics contained in future observations goes unused. Here we take a step back from both approaches: Instead of hand-designing how trajectories should be incorporated, a *teacher* network learns to extract relevant information from the trajectories and to distill it into target activations which guide a *student* model that can only observe the present. The teacher is trained with meta-gradients to maximize the student's performance on a validation set. Our approach performs well on tasks that are difficult for model-free and model-based methods, and we study the role of every component through ablation studies.

1 Adapted from *Neitz, A.*, Parascandolo, G.*, and Schölkopf, B. (2021). "A teacher-student framework to distill future trajectories." In: International Conference on Learning Representations. (ICLR 2021)* I co-developed idea, implementation and experiments together with Giambattista Parascandolo.

4.1 Introduction

The ability to learn models of the world has long been argued to be an important ability of intelligent agents. An open and actively researched question is how to learn world models at the right level of abstraction. This chapter argues, as others have before, that model-based and model-free methods lie on a spectrum in which advantages and disadvantages of either approach can be traded off against each other, and that there is an optimal compromise for every task. Predicting future observations allows extensive use of all observations from previous experiences during training, and to swiftly transfer to a new reward if the learned model is accurate. However, due to partial observability, stochasticity, irrelevant dynamics and compounding errors in planning, model-based methods tend to be outperformed asymptotically (Pong et al., 2018; Chua et al., 2018). On the other end of the spectrum, purely model-free methods use the scalar reward as the only source of learning signal. By avoiding the potentially impossible task of explicitly modeling the environment, model-free methods can often achieve substantially better performance in complex environments (Vinyals et al., 2019; OpenAI et al., 2019). However, this comes at the cost of extreme sample inefficiency, as only predicting rewards throws away useful information contained in the sequences of future observations.

What is the right way to incorporate information from trajectories that are associated with the inputs? In this chapter we take a step back: Instead of trying to answer this question ourselves by hand-designing what information should be taken into consideration and how, we let a model *learn* how to make use of the data. Depending on what works well within the setting, the model should learn *if and how* to learn from the trajectories available at training time. We will adopt a teacher-student setting: a *teacher* network learns to extract relevant information from the trajectories, and distills it into target activations to guide a *student*

network.[2] A sketch of our approach can be found in Figure 4.1, next to prototypical computational graphs used to integrate trajectory information in most model-free and model-based methods. Future trajectories can be seen as being a form of *privileged information* Vapnik and Vashist, 2009, i.e. data available at training time which provides additional information but is not available at test time.

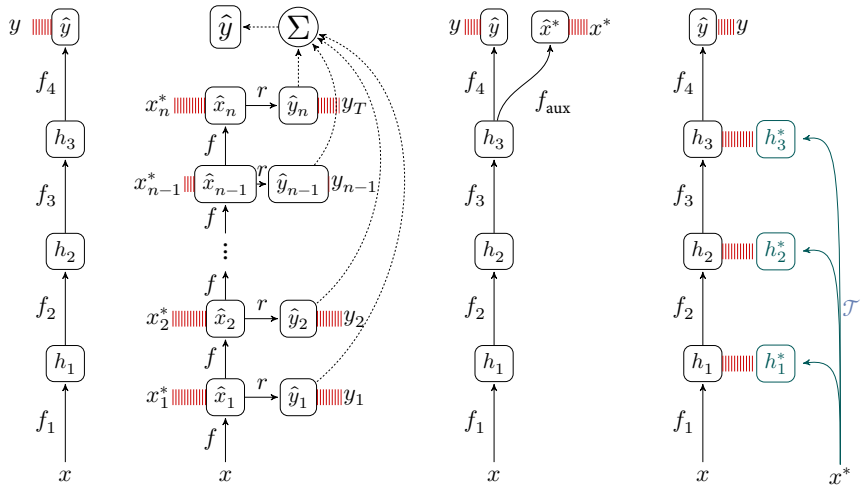
Contributions The main contribution of this chapter is the proposal of a generic method to extract relevant signal from privileged information, specifically trajectories of future observations. We present an instantiation of this approach called Learning to Distill Trajectories (LDT) and an empirical analysis of it.

4.2 Related work

Efficiently making use of signal from trajectories is the topic of a wide range of research works. The technique of bootstrapping in TD-learning (Sutton, 1988) uses future observations to reduce the variance of value function approximations. However, in its basic form, bootstrapping provides learning signal only through a scalar bottleneck, potentially missing out on rich additional sources of learning signal. Another approach to extract additional training signal from observations is the framework of Generalized Value Functions (Sutton et al., 2011), which has been argued to be able to bridge the gap between model-free and model-based methods as well. A similar interpretation can be given to the technique of successor representations (Dayan, 1993).

A number of methods have been proposed that try to leverage the strengths of both model-free and model-based methods, among them Racanière et al., 2017, who learn generative models of the environment

2 Note that the term *distillation* is often used in the context of "distilling a large model into a smaller one" (Hinton et al., 2015), but in this context we talk about distilling a trajectory into vectors used as target activations.



(a) Model-free **(b)** Vanilla model-based **(c)** Auxiliary task **(d)** Teacher-based

Figure 4.1: Comparison of architectures. The data generator is a Markov reward process (no actions) with an episode length of n. x denotes the initial observation. $y = \sum_i y_i$ is the n-step return (no bootstrapping). $x^* = (x_1^*, x_2^*, ..., x_n^*)$ is the trajectory of observations (privileged data). Model activations and predictions are displayed boxed. Losses are displayed as red lines. Solid edges denote learned functions. Dotted edges denote fixed functions.

and fuse predicted rollouts with a model-free network path. In a different line of research, Silver et al., 2017 and Oh et al., 2017 show that value prediction can be improved by incorporating dynamical structure and planning computation into the function approximators. Guez et al., 2019 investigate to what extent agents can learn implicit dynamics models which allow them to solve planning tasks effectively, using only model-free methods. Similarly to LDT, those models can learn their own utility-based state abstractions and can even be temporally abstract to some extent. One difference of these approaches to LDT is that they use reward as their only learning signal without making direct use of future observations when training the predictor.

The meta-gradient approach presented in this chapter can be used more generally for problems in the framework of *learning using privi-*

leged information (LUPI, (Vapnik and Vashist, 2009; Lopez-Paz et al., 2016)), where privileged information is additional context about the data that is available at training time but not at test time. Hindsight information such as the trajectories in a value-prediction task falls into this category.

There are a variety of representation learning approaches which can learn to extract learning signal from trajectories. Jaderberg et al., 2017 demonstrate that the performance of RL agents can be improved significantly by training the agent on additional prediction and control tasks in addition to the original task. Du et al., 2018 use gradient similarity as a means to determine whether an auxiliary loss is helpful or detrimental for the downstream task. Oord et al., 2018 introduce a method based on contrastive learning. They, as well as multiple follow-up studies, show that the representations learned in this way are helpful for downstream tasks in a variety of settings.

Buesing et al., 2018 present ways to learn efficient dynamical models which do not need to predict future observations at inference time. Recently, Schrittwieser et al., 2020 introduced an RL agent that learns an abstract model of the environment and uses it to achieve strong performance on several challenging tasks. Similarly to our motivation, their model is not required to produce future observations. Meta-learning approaches have recently been shown to be successful as a technique to achieve fast task adaptation (Finn et al., 2017), strong unsupervised learning (Metz et al., 2019), and to improve RL (Xu et al., 2018). Similar to LDT in motivation is the recent work by Guez et al., 2020 which also investigates how privileged hindsight information can be leveraged for value estimation. The difference to LDT is how the trajectory information is incorporated. Their approach has the advantage of not needing second-order gradients. At the same time, LDT naturally avoids the problem of the label being easily predictable from the hindsight data — the teacher is trained to present it to the student in such a way that it empirically improves the student's performance on held-out data. Veeriah et al., 2019 use meta-gradients to derive useful auxiliary tasks in the form of generalized value functions. In contrast, we use a teacher

network that learns to provide target activations for a student neural network based on privileged information.

4.3 Meta-Learning a Dynamics Teacher

Here we describe our approach of jointly learning a teacher and a student.[3] While our approach applies to the generic setting of *learning using privileged information* (Vapnik and Vashist, 2009), here we will focus on the special case of a prediction task with an underlying dynamical system.

4.3.1 Learning task

We are considering learning problems in which we have to make a prediction about some property of the future state of a dynamical system, given observations up to the current state. Our method particularly applies to systems in which both the function that relates the current observation to the label as well as the function that predicts the next observation from the current one are hard to learn, making it difficult for both model-free and model-based methods respectively.

To make the explanation more concrete, we will use the practical problem of medical decision-making as a running example to which we can relate the definitions we used, similar to a motivating example from Vapnik and Vashist, 2009: given the history of measurements (biopsies, blood-pressure, etc.) on a given patient and the treatment assignment, we want to predict whether the patient will recover or not.

The input $x \in \mathcal{X}$ of our learning task is some observation of the system state $s_t \in \mathbf{S}$ before and including time step[4] $t \in \mathbb{Z}$. In our running

3 Note that unlike in some related work, the teacher in our task is *not* a copy of the student network, but can have a completely different architecture.

4 For simplicity, our dynamical system is time-discrete, but this assumption is not important for what follows.

example, s_t can be considered the detailed physical state of the patient, which is not directly observable. The observations x include potentially multi-modal data such as x-ray images, vital sign measurements, oncologist reports, etc. The system is governed by an unknown dynamical law $f : \mathbf{S} \to \mathbf{S}$ — in our example, the dynamics are physical equations that determine the evolution of all cells in the body. The prediction target $y \in \mathcal{Y}$ is some function of a future state $s_T = f^{T-t}(s_t)$, separated from t by $T - t$ time steps: $y = g(s_T)$. In our running example, a prediction target could be the binary indicator of whether the patient will recover within some time frame. Note that T could vary from one example to the next. In addition to the initial observation, we have access to the trajectory $x^* = (x_\tau)_{\tau=t+1..T}$ at training (but not test) time. In our running example, the trajectory includes all measurements from the patient *after* the treatment decision has been made. This information is available in a dataset of past patients (in hindsight), but not in any novel situation.

4.3.2 Supervision of internal activations

A straightforward approach to solve the learning task which takes into account the trajectory information, would be to train a state-space-model (SSM) $\hat{f}$, consisting of a dynamical model and a decoder. The SSM is trained to maximize the likelihood of the observed trajectories in the training set, conditioned on the observed initial observation. Ideally, the induced $\hat{f}$ closely resembles f, such that at test time, we can use it to generate an estimate of the rollout and infer the label from it. A potential drawback of this approach is that learning a full SSM could be more difficult than necessary. There may be many details of the dynamics that are both difficult to model and unimportant for the classification tasks. One example for this is the precise timing of events. As argued by Neitz et al., 2018; Jayaraman et al., 2019, there are situations in which it is easy to predict a sequence of events where each event follows a previous one, but hard to predict the exact timing of those events. Moreover, an SSM

typically requires rendering observations at training time, which may be difficult to learn and computationally expensive to execute.

In the running example from Section 4.3.1, it seems challenging and wasteful to predict all future observations in detail, as it would require modeling a complicated distribution over data such as X-ray images or doctor reports written in natural language. Ideally, we would like a model to learn how to extract the relevant information from these data efficiently.

We propose to relax the requirement of fitting the dynamics precisely. The teacher can decide to omit properties of the observations that are not needed and omit time steps that can be skipped. It could also change the order of computation and let the student compute independently evolving sub-mechanisms sequentially, even if they evolved in parallel in the actual data generating process. In addition to potentially simplifying the learning problem, this could have the additional benefit of gaining computational efficiency. For example, modeling detailed pixel observations may be computationally wasteful, as argued by Buesing et al., 2018 and Oord et al., 2018.

4.3.3 Student-teacher setup

We propose a student-teacher setup with two neural networks, as shown in Figure 4.2. The *student network* $\mathcal{S}$, parameterized by weights θ, is the network that attempts to predict the quantity of interest y (for instance a cumulative reward or value). Its input is $x \in \mathcal{X}$, and its output is $\mathcal{S}(x) = \hat{y} \in \mathcal{Y}$. In computing $\hat{y}$, it produces a sequence of internal activations $(h_1, ..., h_N)$, one for each of its N hidden layers. Each h_k is a vector whose size is the number of neurons of the corresponding hidden layer. The student's goal is to minimize the generalization loss $\mathbb{E}_{x,y\sim P_{\text{test}}}[\mathcal{L}_y(S(x), y)]$ for some loss function $\mathcal{L}_y : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$.

The *teacher network* $\mathcal{T}$, parametrized by weights ϕ, is only used at training time, not at test time. It reads the observations of the rollout

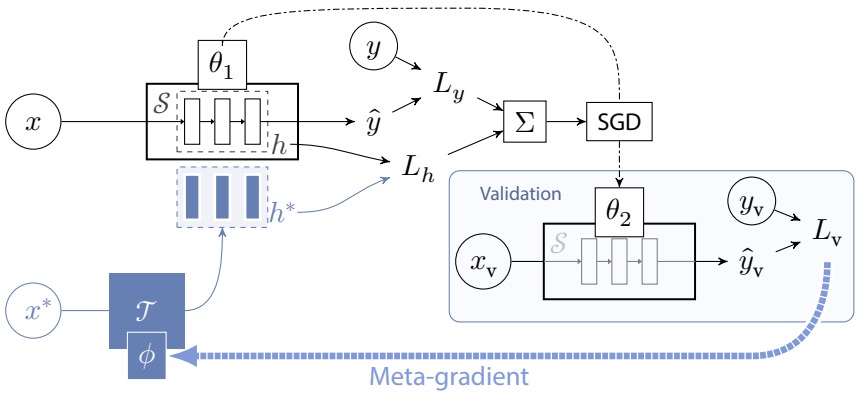


Figure 4.2: Visualization of the LDT framework for the special case of $n = 1$. Circled nodes are part of the dataset. x denotes the input, x^* is the privileged data, y is the label. $\mathcal{S}$ is the student network with parameters θ, $\mathcal{T}$ is the teacher network with parameters ϕ.

$x^* = (x_\tau)_{\tau=t+1..T}$ corresponding to the current training example, and outputs supervision signals $(h_1^*, ..., h_N^*)$.

The target activations produced by the teacher's supervision result in another loss for the student, the *teaching loss*, defined as

$$L_h = \sum_k \mathcal{L}_h(h_k, h_k^*).$$

$\mathcal{L}_h$ denotes the teaching loss function which, given a pre-activation and a supervision signal, produces a scalar value. It can be chosen to be any common loss function. Note however that in general, $\mathcal{L}_h$ could combine its inputs in an arbitrary way, as long as it is differentiable and produces a scalar. In particular, h_k and h_k^* are not required to have the same dimensionality. For example, in our specific instantiation described in Section 4.4, h^* contains masking weights to gate the teaching signal. The total student training loss is

$$L_{\text{train}} = \alpha L_h + L_y,$$

where L_y is the *label loss*, e.g. the cross-entropy error between predictions and true labels. $\alpha \in \mathbb{R}^+$ is the *teaching coefficient*, a coefficient weighting the losses against each other.

4.3.4 Training the teacher using meta-gradients

We train the teacher's weights ϕ using the the technique of *meta-gradient optimization*. This is done as follows: At the beginning of training, we split the dataset into a training and a validation set[5]. This split is kept during the entire duration of training. The split ratio is a hyperparameter.

The student's weights θ are updated n times using Stochastic Gradient Descent on randomly sampled training batches, resulting in updated weights θ_n. The student $\mathcal{S}$ is then evaluated on a validation set, producing a validation loss L_{val}. This validation loss is *optimized by the teacher*. This validation loss does not contain a term for the internal activation loss, but consists only of the label-loss $\mathcal{L}\left(\mathcal{S}(x_{val}), y_{val}\right)$. The teacher is optimized via the meta-gradient

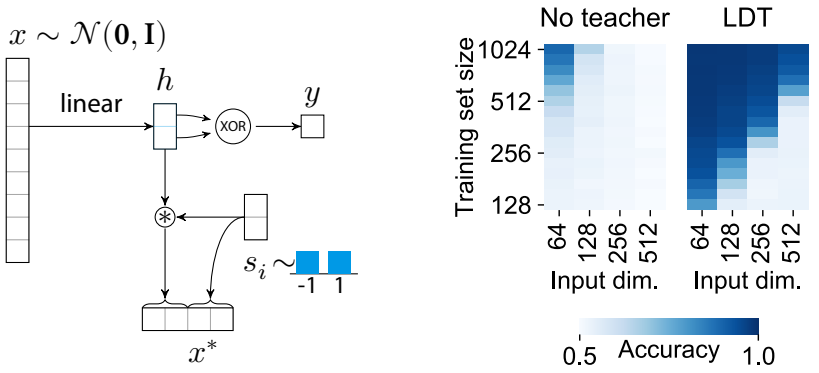
$$\frac{\mathrm{d}L_{val}}{\mathrm{d}\phi} = \frac{\partial \mathcal{L}}{\partial \widehat{y}}\Big|_{\hat{y}=\mathcal{S}(x_{val};\theta_n), y=y_{val}} \cdot \frac{\partial \mathcal{S}}{\partial \theta}\Big|_{x=x_{val};\theta=\theta_n} \cdot \frac{\mathrm{d}\theta_n}{\mathrm{d}\phi} \tag{4.1}$$

where x_{val} and y_{val} are the inputs and targets from the validation set.

We omit the summation over individual loss components to avoid cluttering the notation. The crucial quantity $\frac{\mathrm{d}\theta_n}{\mathrm{d}\phi}$ describes how the final student's weights θ_n depend on the teachers weights ϕ. It can be computed in both linear time and space in the number of steps in the inner optimization loop using automatic differentiation[6]. The meta-gradient is then used for one step of stochastic gradient descent of the teacher's weights ϕ. The student's weights are reset to what they were at the beginning of the step, since θ_n was only a hypothetical parameterization used

5 Note that the validation set is separate from the *test set*, which is an independently sampled dataset used only to evaluate the generalization performance of all methods.

6 See Baydin et al., 2018 for a survey of Automatic Differentiation in Machine Learning.



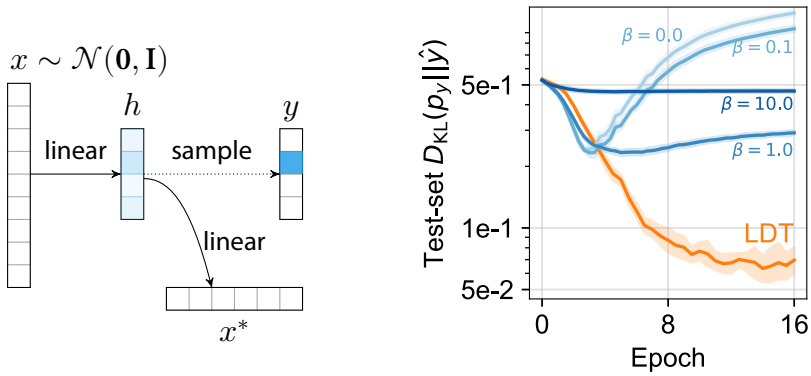
(a) (b)

Figure 4.3: (a) Data generation diagram of task A. (b) Test accuracies on task A. For every combination, we report the maximum test accuracy achieved, averaged over five random seeds.

to determine the meta-gradient. Then, the student is actually trained using the newly updated teacher for a certain number of steps. In our experiments, every step of meta-training is followed by N steps of training the student's weights where N is a hyperparameter. Alg. **??** describes the teacher update formally.

4.4 Experiments

We implemented LDT in PyTorch (Paszke et al., 2019) using *higher* by Grefenstette et al. (Grefenstette et al., 2019). Note that in all experiments, we distinguish between a *validation set* and a *test set*. The validation set is used to train the teacher's parameters. Therefore, in order to allow for fair comparison with non-meta-learning baselines, we train baselines with the full training set and for LDT we split this set into a training and a validation portion. The test set is separate from the validation set and is used only passively to track generalization metrics.



(a) (b)

Figure 4.4: (a) Data generation diagram of task B. (b) Test losses on task B achieved by LDT and the no-teacher baselines with different entropy regularization coefficients β.

4.4.1 Toy examples

Before moving on to datasets of dynamical systems, we study two toy tasks in order to give a better intuition for situations where privileged data can improve learning even though it is unavailable at test time, and at the same time to examine whether LDT can make use of the privileged data.

Task A This task demonstrates a situation where the privileged information x^* predicts the label y perfectly and is lower-dimensional than the input x. At the same time, x^* is not deterministically predictable from x. Formally, we construct the distribution over x, x^*, and y such that the conditional expectation $\mathbb{E}[x^*|x] = 0$ for all x, and the conditional entropies $H(y|x) = H(y|x^*) = 0$. This is intended to correspond to a real-world setting where we observe training-time privileged data which gives us a low-dimensional explanation of the label, but this explanation has been obfuscated by noise and is hence unneces-

sarily difficult to predict directly (here impossible with a deterministic model).

We first sample a D-dimensional input x with independent Gaussian components. This vector gets mapped to a 2-dimensional vector h using a random but fixed linear transformation $A \in \mathbb{R}^{2\times D}$. The label is obtained by applying XOR to $h > 0$. The privileged vector x^* is constructed by independently sampling another two-dimensional vector s which is multiplied with h and concatenated to it. See Figure 4.3 for a diagram and Appendix C.2.1 for a detailed description of the dataset.

Using an MLP to predict x^* from x fails because the optimal predictor of x^* from x always outputs 0. However, in principle LDT can help in this situation: the teacher could learn to invert the stochastic mapping from h to x^*. We set up a study to examine whether LDT automatically discovers a suitable inversion in practice. As student- and teacher-models we use MLPs with one hidden layer each. The teacher gets x^* as input and produces target activations for the student's hidden layer. To investigate the sample efficiency, we let both the unguided student and LDT learn for a grid of different input dimensionalities D and dataset sizes. The achieved test-set accuracies are shown in Figure 3, indicating that using a privileged data and a teacher makes this learning problem substantially more sample-efficient.

Task B In this task (Figure 4.4), instead of using deterministic labels and stochastic privileged data, we construct *deterministic* privileged data and *noisy* labels. A large neural network trained on these labels will tend to fit them exactly. As we discover empirically, this leads to ill-calibrated out-of-sample predictions.

An example for a practical situation where this applies is learning a value function in reinforcement learning using Monte Carlo or n-step temporal difference learning: environments and policies are typically stochastic, resulting in noisy empirical value targets. However, the trajectory of observations and actions as privileged data, can in principle explain away part of the influence of chance in the observed value target. We model this situation by providing as privileged data a transformed

view of the logits that were used to sample the target. This transformation is unknown to the learner, and we investigate whether LDT can still use $x^\star$ to make the student learn a well-calibrated mapping.

We again sample the D components of each input x i.i.d. from $\mathcal{N}(0, 1)$. The random linear transformation $A \in \mathbb{R}^{d_h \times D}$ now transforms x into a d_h-dimensional space. The privileged data $x^* \in \mathbb{R}^{d_p}$ is obtained via another linear transformation $B \in \mathbb{R}^{d_p \times d_h}$ of h. In our experiment we set $D = 128$, $d_h = 4$ and $d_p = 32$, and use 1000 training examples. The teacher gets x^* as input and only needs to supervise the student's output layer. As baseline we train the student without a teacher or privileged data, but regularize its output predictions by subtracting $\beta H(\hat{y})$ from the training loss of each example, where $H(\hat{y})$ is the entropy of the model's prediction, and β is a scalar coefficient. As shown in Figure 4.4, learning the mapping from stochastic labels alone never learns a well-calibrated map from x to y, while LDT learns to interpret the privileged information at training time, leading to a student that generalizes well at test time.

4.4.2 Game of Life

We performed an additional experiment aimed at evaluating whether LDT can help to extract dynamical information from trajectories. For that reason we created a dataset based on the cellular automaton Game of Life by John Conway. The input is a random initial state x, the output is the state of one particular cell after n evolution steps. The privileged data x^* consists of the trajectory of n states after the first one. To make the task more difficult, x^* is temporally permuted randomly, but consistently across examples.

Dataset As the underlying dynamical system, we use the cellular automaton *Game of Life* by John Conway. The rules of this cellular automaton are:[7]

1. Any live cell with fewer than two live neighbours dies, as if by underpopulation.
2. Any live cell with two or three live neighbours lives on to the next generation.
3. Any live cell with more than three live neighbours dies, as if by overpopulation.
4. Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.

Note that there is no linear decision boundary to determine the next cell state if the input space is the cell's neighborhood. However, with at least one hidden layer, it is possible to implement the rules. We can therefore use a convolutional neural network with $2n$ layers to represent the system's dynamics unrolled over n steps.

We generate a binary classification task from the system as follows.

- A board of size 17×17 is initialized by setting each cell to "alive" with a probability of $p_I = 0.4$, independently of other cells. Alive cells are represented with a 1, all other cells are represented with a 0. This first board state is the input $x \in \{0, 1\}^{17 \times 17}$ for the task.
- The Game-of-Life rules are applied three times in sequence, yielding three consecutive states
- The classification target is the *middle cell in the last state*, i.e. a binary label

7 Copied from https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life.

- All intermediate and final states in the trajectory are considered *privileged information*.

The initial alive-probability was chosen such that there typical rollouts are diverse. If the initial configuration is much sparser or much denser, then the population quickly dies off.

Network architecture In early experiments we found that three layers per step facilitate training of the CNN compared to two layers. Therefore, we fix three convolutional layers per step for all experiments.

The student's internal activations $\mathbf{z}$ are convolutional feature maps. We use $\mathbf{z}_{kl}$ to denote the l'th feature map of the k'th layer. Each $\mathbf{z}_{kl}$ has the same dimensions as the map of the cellular automaton (17×17 in our experiments).

Implementation of the teacher We choose a simple parameterization of the teacher. The teacher's weights are a three-dimensional tensor $\phi \in \mathbb{R}^{T \times N \times F}$, where T is the fixed number of steps in the Game-of-Life trajectory, N is the number of convolutional hidden layers in the student network, and F is the number of feature maps per hidden layer.

The internal activation targets are linear combinations of the cell states in the trajectory $(x_1, ..., x_T)$:

$$h^*_{kl} = \sum_t a_{tkl} x_t \tag{4.2}$$

where $a_{\bullet kl} = \text{softmax}(\phi_{\bullet kl})$, i.e. the mixture weights are the teacher's weights softmaxed-through-time. More explicitly,

$$a_{tkl} = \frac{\exp(\phi_{tkl})}{\sum_\tau \exp(\phi_{\tau kl})} \tag{4.3}$$

We use Adam to update the inner weights and vanilla SGD+momentum to update the meta-parameters, as we found these two choices to generally perform best.

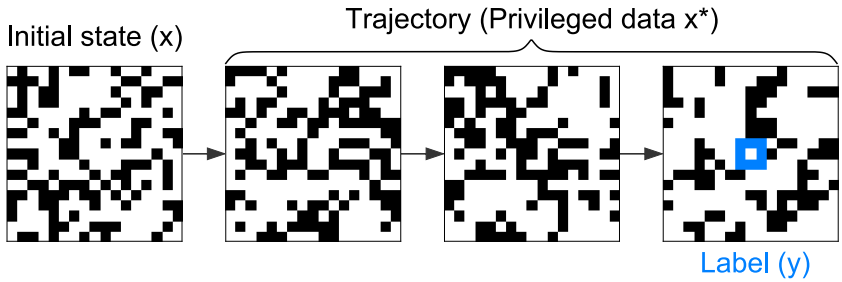


Figure 4.5: A datapoint in the Game-of-Life task. The initial state x is a randomly sampled binary pattern, the trajectory x^* is obtained from applying the transition rule multiple times. The label y is the state of the center cell.

Baselines

- **Fixed oracle teacher**: ϕ is initialized to the values such that it provides the frames in the correct sequence to every third convolutional layer. It is held fixed over the course of training.
- **Fixed random teacher**: ϕ is initialized randomly and held fixed over the course of training.
- **No teacher**: Classification task without privileged data

Details about are provided in Appendix C.3. As shown in Fig. 4.6, LDT can learn from the scrambled trajectory and help the student learn with less data than a model-free method.

4.4.3 MuJoCo

In order to evaluate whether LDT can improve learning efficiency in continuous control tasks, we set up a prediction task using the MuJoCo simulator (Todorov et al., 2012). As an objective we choose learning an n-step reward model: such a model has to predict the sum of rewards along a trajectory of n steps, given only access to the current state and n (open-loop) actions. We fix $n = 16$ in all experiments.

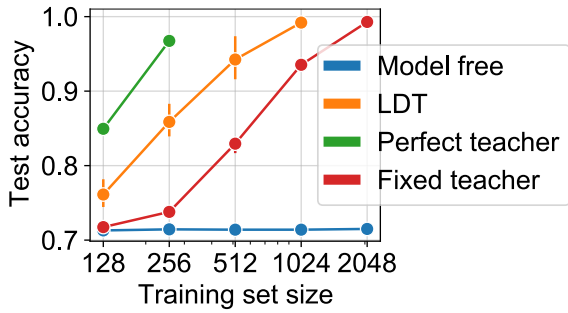


Figure 4.6: Result of Game-of-life experiment. The model-free approach cannot learn anything from training sets of the sizes we investigated. LDT improves test accuracy over using the fixed-teacher baseline.

We collect a dataset of size 4000 for each of the MuJoCo environments Swimmer-v2, Walker2d-v2, Hopper-v2, and HalfCheetah-v2. For each training example, the input $x = (s_t, a_{t:t+n})$ contains both an initial state and an action sequence. The initial state s_t is obtained by executing a random policy for a short number of time steps after resetting the simulator. The action sequence $a_{t:t+n}$ consists of random actions. The label y is the cumulative return of executing actions $a_{t:t+n}$ in state s_t. See Appendix C.2.2 for more details on the dataset.

Models We evaluate the following methods:

- **Model-free (MF)**: This baseline resembles the architecture shown in Fig. 4.1a. neurons per layer. This baseline is trained with x as input and y as output.
- **Auxiliary task (Aux)**: As shown in Fig. 4.1c, this baseline augments the model-free method with an auxiliary task-head, which is trained to fit the full trajectory x^* in order to shape the model's internal representation.
- **LDT**: Using the same MLP architecture as in the model-free baseline for the student, LDT additionally uses a teacher, which uses

a network with a 1D-convolutional torso and an MLP-head to embed the trajectory x^* and provide training signals for the student activations. The teaching loss is a gated mean squared error $L_h \propto \sum_k \sigma(m_k)(h_k - h_k^*)^2$, where σ is the logistic sigmoid function, m and h^* are the outputs by the teacher, and h are the student's internal pre-activations.

For all networks we follow Schrittwieser et al. (Schrittwieser et al., 2020) in how we turn the regression task of predicting rewards into a classification task by binning the reward space (see Appendix for details). Hyperparameters for each method are optimized independently (see Appendix for ranges) for each method and task. We select the configuration with the lowest mean-squared-error on test data and re-run it eight times with different random seeds.

Results In Figure 4.7 we report the mean squared error between predicted and true cumulative reward. The student which was trained with LDT achieved lower MSE than both the MF and Aux baselines in all tasks. Moreover, we found the *generalization gap* to be significantly smaller for LDT (Figure 4.8). This can be explained as follows: If the student is overfitting to the training set, its performance will degrade on the validation set. Since this loss only affects the teacher, the teacher can provide teaching targets h^* that steer the student away when it starts to severely overfit to the training data.

We acknowledge that there could be strong baselines from the literature that we have not considered. Many of these approaches have complex pipelines of operations (e.g. Chua et al. (Chua et al., 2018)), or use domain-specific knowledge to extract good representations (e.g. CPC (Oord et al., 2018)). As the main point of this chapter is to investigate and understand a new framework to incorporate trajectory information, we decided to keep our evaluation setting simple and consistent, e.g. by using the same student architecture for *all* models.

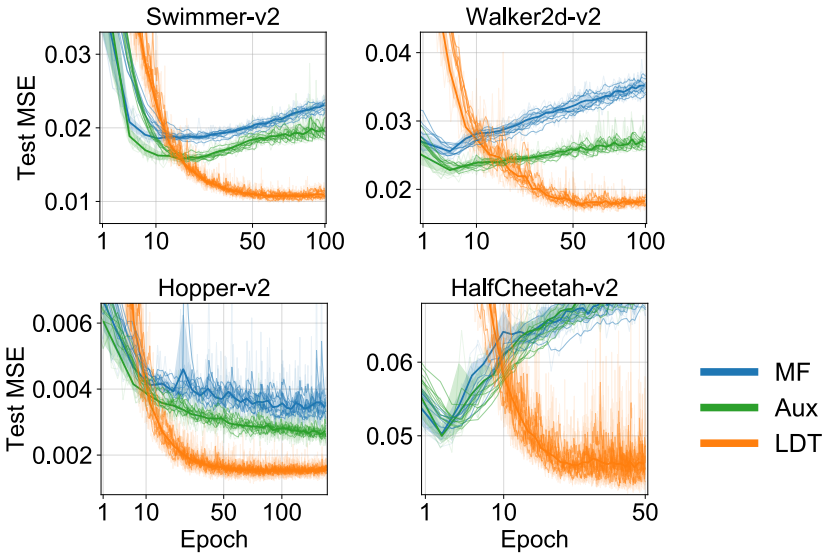


Figure 4.7: Test losses for the MuJoCo reward prediction task. Evolution of mean squared error between predicted and true normalized n-step reward on a held-out test set. We ran each configuration with 8 different random seeds and show the aggregated curves.

4.4.4 Ablations

We perform several ablation studies, in order to test the role of every component in our set-up. We describe every experiment set-up and show the results in Figure 4.9.

In the following list, we describe the different ablation studies in detail.

1. *Fixed untrained teacher (FT):* The teacher still provides target activations for the student, but we keep the teacher's weights fixed to the initial random weights. Using no meta-gradient updates, the student essentially fails to learn, validating the need to train the teacher.

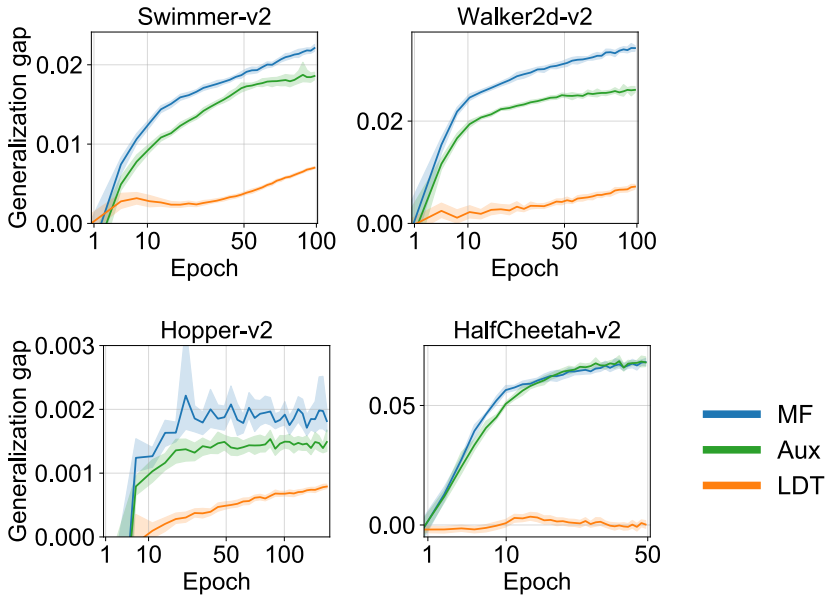


Figure 4.8: Generalization gaps (Test-set error minus Training-set error) for the different approaches and domains.)

2. *Time-permuted x^* (PT)*: we independently and randomly permute the time order of x^* for each example. Interestingly, the performance only degrades slightly. Two possible interpretations of this result are the following: either the teacher architecture does not make sufficient use of the temporal structure, or this structure is not so relevant for the task. Since we did not heavily tune our teacher network, we tend to lead towards the first explanation.

3. *Random x^* (RT-n)*: each entry of the last n frames of all trajectories x^* are replaced by noise sampled i.i.d. from $\mathcal{N}(0, 1)$. When the trajectories are completely irrelevant to the task, the teacher can in principle learn to ignore them. However, since LDT trains with a smaller effective training set (because a portion is split off for validation), we expect a slightly weaker performance. Indeed, the results show that LDT with fully randomized trajectories (RT-

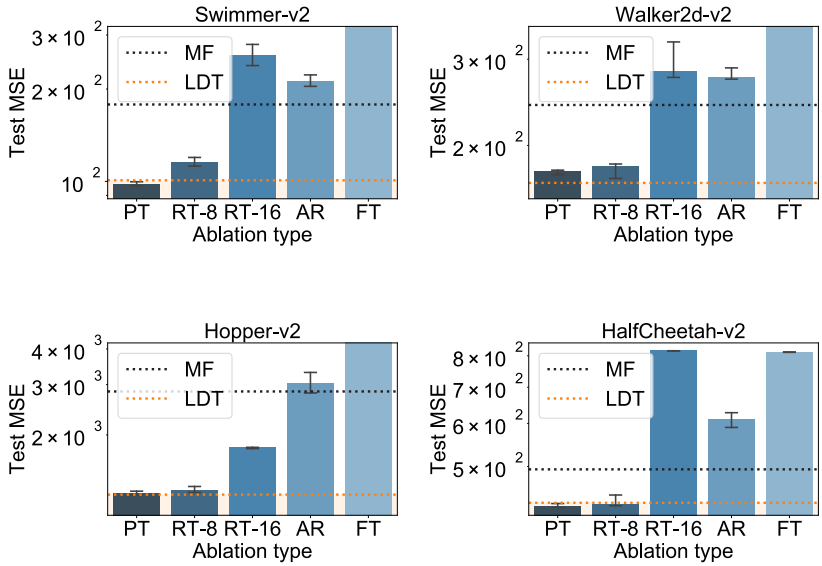


Figure 4.9: Results of ablation studies.

16) tends to be comparable to, but slightly worse than the model-free network.

4. *Same training and validation data (AR):* instead of fixing the split of the training data into disjoint subsets for the student and for the teacher, we resample the split after every step of meta-training. Results show that the performance is even slightly worse than the model-free baseline. This is consistent with expectations, as the teacher's loss can be minimized directly by the student in training.

4.5 Conclusion

In this chapter, instead of proposing a new hand-designed strategy to incorporate information from previous trajectories, we proposed to let

a model *learn if and how* to use them. A teacher network learns to extract relevant information from a trajectory, and distills targets activations for a student network that only sees the current observation. The teacher is rewarded for maximizing the student performance on validation data, but can only achieve this indirectly by supervising the student while it trains on the training data. The aim of this method is to preserve advantages from both model-based as well as model-free methods: using the rich amount of information from observations as a training signal instead of just reward signal, but is not capped in performance due to model bias. One advantage that is *not* preserved from model-based approaches is the straightforward possibility to change tasks by adapting the reward function only. As validated empirically by the experiments and ablations presented in Section 4.4, this framework allows the model to choose where to sit in the wide spectrum between model-based and model-free methods, adapting to the specifics of the task at hand. An obvious drawback of LDT is — like many algorithms that learn how to learn — that computing meta-gradients increases the time and space complexity at training time by a factor that is linear in the number of inner steps. However, the computational cost of the student at test time is exactly the same as for an equivalent student that did *not* make use of the privileged information at training time, as the teacher does not play any role and can be discarded.

We believe that the general framework of teacher-student trained with meta-gradients to incorporate privileged information can be a fruitful direction for future work, beyond learning from trajectories. As the main limitation is the linear increase in time and space complexity at training time, increases in computing power should allow for more and more complex teachers and students to be trained on large tasks.

5

Conclusion

The goal of this thesis was to present different training objectives that can help create "mechanistic" models of the world. It showed that *Adaptive skip intervals* can help make long-range sequential predictions both easier and more computationally efficient. Similarly to how causal models can be seen as a coarse view of a system that is fundamentally governed by differential equations, adaptive skip intervals abstract away details of the time evolution that are hard to predict and unnecessary for long range predictions.

We have seen how the *AND-mask* can help suppress inconsistent training signals from different environments. Whenever there are underlying mechanisms of a system which produce consistent training signals across multiple environments, this method can help moving us closer towards identifying these mechanistic models.

Finally, we investigated the goal of learning dynamical models that combine both state- and time-abstraction. While one solution to this could have been to manually construct a framework where latent embeddings and temporal matching are combined, *LDT* approached the problem from the other end – taking the downstream objective seriously and using a meta-learning approach that allows a teacher to discover useful ways to interpret the training trajectory, giving the student an internal dynamical model that is optimized towards utility. For this approach to work, we need the assumption that the trajectories do contain useful explanations about the label which can mitigate overfitting, that a useful transformation of the training signal is simple, and that this

transformation is beneficial across multiple tasks or sub-tasks within the training distribution.

Limitations ASI is limited in that frame-skipping is its only degree of freedom. If there are multiple independent processes evolving in parallel, it might be beneficial to simulate them independently from each other. ASI does not allow this, because it requires matching full ground truth frames without the possibility of splitting them into subsystems. Moreover, ASI in the form described in Chapter 2 is restricted to uncontrolled systems. LDT can bypass these limitations in principle since it gives full freedom for how a trajectory is used. On the other hand, this freedom leads to poor interpretability of the resulting implicit dynamical model: we cannot easily analyse which aspects of the modeled system are represented by the internal states.

Outlook

Some of the limitations of ASI mentioned above are addressed by Divide-and-conquer Monte-Carlo-Tree-Search (DC-MSCTS) (Parascandolo et al., 2020). It can be applied to systems with actions and does not constrain the planning to happen sequentially from present to future, but hierarchical. This comes at the expense of being a subgoal-based model, which cannot easily model uncontrollable dynamics. Neither ASI and DC-MCTS address state-abstraction.

For ideas related to LDT, we should move towards population-based training, such that the teacher is forced to provide more universally useful feedback. LDT has not shown its full potential in simple, fixed domains. While many reinforcement learning tasks consist of a sequence of multiple sub-tasks, once they are all solved, there is no learning signal left for the teacher, and any overfitting that has occurred so far will remain.

Model architectures and hypothesis classes that address generalization beyond the training distribution explicitly – for example symbolic equations (Martius and Lampert, 2016; Biggio et al., 2021) – could be combined with methods presented in this thesis.

An important impediment to research into mechanistic planning and reasoning is that often times, reasoning is not necessary in fixed, stationary domains. If an agent plays a simple Atari game over and over again, it will discover a useful policy and can represent it perfectly well with a simple neural network. Advanced reasoning and long-term planning is not necessary here. Instead, we should evaluate, and perhaps also train, agents in complex, nonstationary environments. If we want to use deep learning and train models mostly end-to-end, we need to make sure that there is sufficient pressure on the agent to be able to adapt quickly to new situations. In such systems, developing a mechanistic understanding of the world might be the only solution for the agent. A natural way to get complex, open-ended challenges is to use environments with multiple agents. These agents may have different goals, leading to an ever-changing evolution of cooperative and competitive strategies, as shown recently by (Team et al., 2021).

A

Adaptive Skip Intervals: Additional details

A.0.1 Model architecture

In all experiments, the model f consists of 7 convolutional layers with padding mode "same" and the following specifications, where $\text{Conv}(a, (b, c))$ means "a kernels of size (b, c)": [$\text{Conv}(n_k, (5, 5))$, $\text{Conv}(n_k, (5, 5))$, $\text{Conv}(n_k, (5, 5))$, $\text{Conv}(n_k, (7, 7))$, $\text{Conv}(n_k, (5, 5))$, $\text{Conv}(n_k, (1, 1))$, $\text{Conv}(3, (1, 1))$]. Before the 6th layer, the three channels of the model input are concatenated to the feature map. As part of the hyperparameter search, n_k was randomly chosen from the set $\{32, 48\}$. We added two variations of this architecture to the hyperparameter search:

- `f-strided`: the second convolutional layer performs a strided convolution with stride 2 and the 4th convolutional layer performs a transposed convolution.
- `f-dilated`: the fourth convolutional layer uses a dilation rate of 2.

We did not observe substantial difference in the performances of our architectures.

All convolutions were used with a stride of 1. The weight initialization for all layers follows He et al., 2015. We use rectified linear units (ReLU) as activation (Glorot et al., 2011).

A.0.2 Full algorithm with comments

See Algorithm 3 for the full algorithm including scheduled sampling.

Algorithm 3 Dynamical model learning with ASI

Require: i'th trajectory $\mathbf{x}^{(i)} = (x_1, x_2, ..., x_{T_i}) \in \mathcal{X}^{T_i}$
Require: Differentiable model $f : \mathcal{X} \to \mathcal{X}$ w/ params θ
Require: Loss function $\mathcal{L} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$
Require: Matching-horizon $H \in \mathbb{N}$
Require: Exploration schedule $\mu : \mathbb{N} \to [0, 1]$
Require: Scheduled sampling temperatures $\epsilon : \mathbb{N} \to [0, 1]$

$t \leftarrow 1, u \leftarrow 1$ ▷ *Data timestep t, abstract timestep u*
$l \leftarrow 0$ ▷ *Trajectory loss*
$x \leftarrow x_1$ ▷ *x is the next input to the dynamics model f*
while $t < |\mathbf{x}|$ **do**
  $\hat{x}_u \leftarrow f(x)$
  $T \leftarrow \min(t + H, |\mathbf{x}|)$ ▷ *Upper time step limit*
  if Bernoulli$(\mu(i)) = 0$ **then**
    $t \leftarrow \arg\min_{t' \in \{t+1..T\}} \mathcal{L}(\hat{x}_u, x_{t'})$
  else
    $t \sim \text{unif}\{t + 1, T\}$ ▷ *Exploration*
  end if
  $l \leftarrow l + \mathcal{L}(\hat{x}_u, x_t)$ ▷ *Accumulate trajectory loss*
  if Bernoulli$(\epsilon(i)) = 0 = 0$ **then**
    $x \leftarrow \hat{x}_u$ ▷ *Scheduled sampling (Bengio et al., 2015)*
  else
    $x \leftarrow x_t$ ▷ *Take ground truth frame as next model input*
  end if
  $u \leftarrow u + 1$
end while
$\theta \leftarrow$ gradient descent step on θ to reduce l

A.0.3 Training details

For all experiments, the Adam optimizer (Kingma and Ba, 2015) was used. For hyperparameter search, learning rates for the model f were sampled from the set $\{1.0\times10^{-3}, 7.5\times10^{-4}, 5.0\times10^{-4}\}$. The learning rate was decayed by a factor of 0.2 after n_D steps, where n_D was sampled from the set $\{7500, 10000, 15000\}$. The maximum ASI horizon H was sampled from the set $\{15, 18, 21, 25\}$. The number of trajectories per training batch was chosen to be 2 in all experiments. As schedule of exploration for temporal matching we choose $\mu(t) = \max(0, 1 - \frac{t}{K})$, where K was sampled from the set $\{7500, 10000, 15000\}$.

The hyperparameter search described in Section 4.4 resulted in the parameters shown in Tables A.1 and A.2, which were used to produce the resulting plots.

	ASI	**ASI w/o expl.**	**fixed** ($\Delta t = 1$)	**fixed** ($\Delta t = 2$)
learning rate	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}
steps to LR decay	15000	15000	15000	15000
ASI horizon	21	18	-	-
Exploration steps	7500	-	-	-
f-architecture	`f-strided`	`f-simple`	`f-simple`	`f-simple`
f: # of kernels	48	48	48	48

Table A.1: Hyperparameters found for Room Runner

	ASI	**ASI w/o expl.**	**fixed** ($\Delta t = 1$)	**fixed** ($\Delta t = 2$)
learning rate	5×10^{-4}	7.5×10^{-4}	7.5×10^{-4}	7.5×10^{-4}
steps to LR decay	15000	10000	15000	15000
ASI horizon	21	18	-	-
Exploration steps	15000	-	-	-
f-architecture	`f-dilated`	`f-strided`	`f-dilated`	`f-strided`
f: # of kernels	48	32	32	32

Table A.2: Hyperparameters found for Funnel Board

B

ILC: Additional details

B.1 Appendix to Section 3.2

B.1.1 A classic example of a patchwork solution

Consider a neural network with one hidden layer consisting of two neurons and sigmoidal activations:

$$f_\theta(x) = \theta_5\sigma(\theta_1 x+\theta_2)+\theta_6\sigma(\theta_3 x+\theta_4), \quad \sigma(z) := 1/(1+e^{-z}). \tag{B.1}$$

We want to learn the continuous function $f^* : [0,1] \to [0,2]$ defined as

$$f^*(x) = \begin{cases} 0 & x \in [0, 0.4); \\ 10(x-0.4) & x \in [0.4, 0.5); \\ 1 & x \in [0.5, 0.7); \\ 10(x-0.7)+1 & x \in [0.7, 0.8); \\ 2 & x \in [0.8, 1]. \end{cases}$$

To perform this task, we have access to (noiseless) data from two environments:

$$A : \{(x, f(x)) \mid x \in [0, 0.5)\}, \quad B : \{(x, f(x)) \mid x \in [0.5, 1]\}.$$

There is a simple *constructive* way, provided by the universal function approximation theorem Cybenko, 1989 to fit this function[1] using f_θ up

1 For a graphical description, the reader can check http://neuralnetworksanddeeplearning.com/chap4.html

to an arbitrarily small mean squared error $\mathcal{L}_{A+B}(\theta^*)$. Leaving out the details of such a construction (Cybenko, 1989 for details), the reader can check on the left panel of Figure B.1 that

$$\theta^* = (100, -50, 100, -75, 1, 1)$$

provides a good fit for *both environments* A and B — both $\mathcal{L}_A(\theta^*)$ and $\mathcal{L}_B(\theta^*)$ are small.

However, it is easy to realize that θ^* — while being a solution which can be returned by gradient descent using the pooled data A+B — is not *consistent* (formal definition given in the main paper in Section 3.2). Indeed, it is possible to modify $\tilde{\theta}^*$ such that the loss in environment A remains almost unchanged, while the loss in environment B gets larger. In particular, on the right panel of Figure B.1, we show that

$$\tilde{\theta}^* = (100, -50, 100, -75, 1, -0.5)$$

is such that $\mathcal{L}_A(\theta^*) \leq \mathcal{L}_A(\tilde{\theta}^*) + \epsilon$ (with ϵ very small) but $\mathcal{L}_B(\theta^*) \ll \mathcal{L}_B(\tilde{\theta}^*)$. According to our definition in Equation 3.1 (see main paper), we have $\mathcal{I}^\epsilon(\theta^*) \leq |\mathcal{L}_B(\theta^*) - \mathcal{L}_B(\tilde{\theta}^*)|$ — that is a large number (low consistency).

Remark 1 (Connection to out of distribution generalization). The main point of this analysis was to show an example of where our measure of *consistency* behaves according to expectations: A typical implementation of the universal approximation theorem — which one would *not* expect to generalize out of distribution, due to its *'patchwork'* behavior — leads indeed to a very low consistency score.

B.1.2 Section 3.2.2: Consistency as arithmetic/geometric mean of landscapes

Geometric mean of matrices. Given an n-tuple of $d \times d$ positive definite matrices $(A_j)_{j=1}^n$, the geometric (Karcher) mean Ando et al., 2004 is the unique positive definite solution X to the equation

$$\sum_{i=1}^{m} \log(A_i^{-1} X) = 0,$$

where log is the matrix logarithm. This matrix average has many desirable properties, which make it relevant to signal processing and medical imaging. The Karcher mean can also be written as

$$\underset{X \in \mathcal{S}^{++}(d)}{\arg\min}\ f(X) = \frac{1}{2m} \sum_{i=1}^{m} d(A_i, X)^2,$$

where d is the Riemannian distance in the manifold of SPD matrices $\mathcal{S}^{++}(d)$.

Link between consistency and geometric means. Here we show how the consistency score introduced in Equation 3.1 can be linked (in a simplified setting) to a comparison between the arithmetic and geometric means of the Hessians approximating the landscapes of two separate environments A and B.

At the local minimizer $\theta^* = 0$, we assume that $\mathcal{L}_A = \mathcal{L}_B = 0$ and consider the local quadratic approximations $\mathcal{L}_A(\theta) = \frac{1}{2}\theta^\top H_A \theta$ and $\mathcal{L}_B(\theta) = \frac{1}{2}\theta^\top H_B \theta$. Here, we make the additional simplifying assumption that H_A and H_B are diagonal (or, more broadly, co-diagonalizable): $H_A = \text{diag}(\lambda_1^A, \cdots, \lambda_n^A)$, $H_B = \text{diag}(\lambda_1^B, \cdots, \lambda_n^B)$, with $\lambda_i^A \geq 0$ and $\lambda_i^B \geq 0$ for all $i = 1, \ldots, n$. The *arithmetic* and *geometric* means (noted as H_{A+B} and $H_{A \wedge B}$) of these matrices are defined in this simplified setting as follows:

$$H_{A+B} = \text{diag}\left(\frac{1}{2}(\lambda_1^A + \lambda_1^B), \cdots, \frac{1}{2}(\lambda_n^A + \lambda_n^B)\right),$$

$$H_{A \wedge B} = \text{diag}\left(\sqrt{\lambda_1^A \lambda_1^B}, \cdots, \sqrt{\lambda_n^A \lambda_n^B}\right).$$

As motivated in the main paper and in Figure B.3, one can link the consistency of two landscapes to a comparison between the geometric and arithmetic means of the corresponding Hessians.

Proposition 3. In the setting we just described, the consistency score in Equation 3.1 can be estimated as follows:

$$\mathcal{J}^\epsilon(\theta^*) \leq 2\epsilon \left(\frac{\det(H_{A+B})}{\det(H_{A \wedge B})}\right)^2.$$

Before showing the proof, we note that the proposition gives a *lower bound* on the consistency. That is, it provides a *pessimistic* estimate. Yet, as we motivated, this estimate has a nice geometric interpretation. However, as we outline in a remark after the proof, this estimate is tight in two important limit cases.

Proof. In this setting, Equation 3.1 gives

$$\mathcal{J}^\epsilon(\theta^*) := \max\left\{\max_{\mathcal{L}_A(\theta)\leq\epsilon} \mathcal{L}_B(\theta), \max_{\mathcal{L}_B(\theta)\leq\epsilon} \mathcal{L}_A(\theta)\right\}.$$

Recall that

$$\mathcal{L}_A(\theta) = \frac{1}{2}\theta^\top H_A \theta = \frac{1}{2}\sum_i \lambda_i^A \theta_i^2.$$

Hence, this is a simple quadratic program with quadratic constraints, and

$$\max_{\mathcal{L}_A(\theta)\leq\epsilon} \mathcal{L}_B(\theta) = \max_{\frac{1}{2}\sum_i \lambda_i^A\theta_i^2\leq\epsilon} \frac{1}{2}\sum_i \lambda_i^B\theta_i^2.$$

Further, we can change variables and introduce $\tilde{\theta}_i = \theta_i\sqrt{\lambda_i^A/2}$. The problem gets even simpler:

$$\max_{\mathcal{L}_A(\theta)\leq\epsilon} \mathcal{L}_B(\theta) = \max_{\|\tilde{\theta}\|^2\leq\epsilon} \sum_i \frac{\lambda_i^B}{\lambda_i^A}\tilde{\theta}_i^2 = \epsilon\cdot\max_i \frac{\lambda_i^B}{\lambda_i^A}.$$

All in all, we get

$$\begin{aligned}
\mathcal{J}^\epsilon(\theta^*) &= \epsilon\max\left\{\max_i \frac{\lambda_i^B}{\lambda_i^A}, \max_i \frac{\lambda_i^A}{\lambda_i^B}\right\} \\
&= \epsilon\cdot\max_i \max\left\{\frac{\lambda_i^B}{\lambda_i^A}, \frac{\lambda_i^A}{\lambda_i^B}\right\} \\
&\leq \epsilon\cdot\max_i \left(\frac{\lambda_i^B}{\lambda_i^A} + \frac{\lambda_i^A}{\lambda_i^B}\right) \\
&= \epsilon\cdot\max_i \left\{\frac{(\lambda_i^B)^2 + (\lambda_i^A)^2}{\lambda_i^B\lambda_i^A}\right\} \\
&\leq \epsilon\cdot\max_i \left\{\frac{(\lambda_i^B + \lambda_i^A)^2}{\lambda_i^B\lambda_i^A}\right\}.
\end{aligned}$$

This means

$$\begin{aligned}\sqrt{\mathcal{J}^{\epsilon}(\theta^*)} &\leq \epsilon \max_i \frac{\lambda_i^B + \lambda_i^A}{\sqrt{\lambda_i^B \lambda_i^A}} \\ &= 2\epsilon \max_i \frac{(\lambda_i^B + \lambda_i^A)/2}{\sqrt{\lambda_i^B \lambda_i^A}} \\ &\leq 2\epsilon \frac{\prod_i (\lambda_i^B + \lambda_i^A)/2}{\prod_i \sqrt{\lambda_i^B \lambda_i^A}} \\ &= 2\epsilon \frac{\det(H_{A+B})}{\det(H_{A \wedge B})},\end{aligned}$$

where the first inequality comes from the monotonicity of the square root function, and the second inequality comes from the fact that (i) the geometric mean is always smaller or equal than the arithmetic mean and (ii) for any sequence of numbers $\alpha_i > 1$, $\max_i \alpha_i \leq \prod_i \alpha_i$. □

Remark 2 (Sanity check). There are two important cases where we can test the bound above. First, if $H_A = H_B$, then $\mathcal{J}^{\epsilon}(\theta^*) = \epsilon$, and the bound returns $\mathcal{J}^{\epsilon}(\theta^*) \leq 2\epsilon$, since the geometric and arithmetic mean are the same. Next, say $\lambda_i^A = 0$ but $\lambda_i^B > 0$; then, both the bound and the inconsistency score are ∞ (highest possible inconsistency).

B.1.3 Proof of Proposition 1

In this appendix section we consider the AND-masked GD algorithm, introduced at the end of Section 3.2. We recall that the masked gradients at iteration k are $m_t(\theta^k) \odot \nabla \mathcal{L}(\theta^k)$, where $m_t(\theta^k)$ vanishes for any component where there are less than $t \in \{d/2 + 1, \dots, d\}$ agreeing gradient signs across environments, and is equal to one otherwise. In a full-batch setting, the algorithm is

$$\theta^{k+1} = \theta^k - \eta\, m_t(\theta^k) \odot \nabla \mathcal{L}(\theta^k), \qquad \text{(AND-masked GD)}$$

where $\eta > 0$ is the learning rate.

Proposition 1. Let $\mathcal{L}$ have L-Lipschitz gradients and consider a learning rate $\eta \leq 1/L$. After k iterations, AND-masked GD visits at least once a point θ where $\|m_t(\theta) \odot \nabla\mathcal{L}(\theta)\|^2 \leq \mathcal{O}(1/k)$.

Proof. Thanks to the component-wise L-smoothness and using a Taylor expansion around θ^i we have

$$\begin{aligned}\mathcal{L}(\theta^{i+1}) &\leq \mathcal{L}(\theta^i) - \eta\langle\nabla\mathcal{L}(\theta^i), m_t(\theta^i) \odot \nabla\mathcal{L}(\theta^i)\rangle \\ &\quad + \frac{L\eta^2}{2}\|m_t(\theta^i) \odot \nabla\mathcal{L}(\theta^i)\|^2 \\ &= \mathcal{L}(\theta^i) - \left(\eta - \frac{L\eta^2}{2}\right)\|m_t(\theta^i) \odot \nabla\mathcal{L}(\theta^i)\|^2.\end{aligned}$$

If we seek $\eta - L\eta^2/2 \geq \eta/2$, then $\eta \leq \frac{1}{L}$, as we assumed in the proposition statement. Therefore, $\mathcal{L}(\theta^{i+1}) \leq \mathcal{L}(\theta^i) - (\eta/2)\|m_t(\theta^i) \odot \nabla\mathcal{L}(\theta^i)\|^2$, for all $i \geq 0$. Summing over i from 0 to a desired iteration k, we get

$$\sum_{i=0}^{k-1}(\eta/2)\|m_t(\theta^i) \odot \nabla\mathcal{L}(\theta^i)\|^2 \leq \mathcal{L}(\theta^0) - \mathcal{L}(\theta^k) \leq \mathcal{L}(\theta^0).$$

Therefore,

$$\min_{i=0,\ldots,k}\|m_t(\theta^i)\odot\nabla\mathcal{L}(\theta^i)\|^2 \leq \frac{1}{k}\sum_{i=0}^{k-1}(\eta/2)\|m_t(\theta^i)\odot\nabla\mathcal{L}(\theta^i)\|^2 \leq \frac{2\mathcal{L}(\theta^0)}{\eta k}.$$

Hence, there exist an iteration $i^* \in \{0, \ldots, k\}$ such that $\|m_t(\theta^{i^*}) \odot \nabla\mathcal{L}(\theta^{i^*})\|^2 \leq \mathcal{O}(1/k)$. □

B.1.4 Proof of Proposition 2

Here we fix parameters $\theta \in \mathbb{R}^n$ and assume gradients $\nabla\mathcal{L}_e(\theta) \in \mathbb{R}^n$ coming from environments $e \in \mathcal{E}$ are drawn independently from a multivariate Gaussian with zero mean and $\sigma^2 I$ covariance. We want to show that, in this random setting, the AND-mask introduced in Section 3.2.3 decreases the magnitude of the gradient step.

Proposition 2. Consider the setting we just outlined, with

$$\mathcal{L} = (1/d)\sum_{e=1}^{d} \mathcal{L}_e$$

. While $\mathbb{E}\|\nabla\mathcal{L}(\theta)\|^2 = \mathcal{O}(n/d)$, we have that

$$\forall t \in \{d/2+1, \dots, d\}, \exists c \in (1,2] \text{s.t.} \mathbb{E}\|m_t(\theta)\odot\nabla\mathcal{L}(\theta)\|^2 \leq \mathcal{O}(n/c^d).$$

Proof. Let us drop the argument θ for ease of notation. First, let us consider $\nabla\mathcal{L}$ (no gradient AND-mask):

$$\mathbb{E}\left\|\frac{1}{d}\sum_{i=1}^{d}\nabla\mathcal{L}_{e_i}\right\|^2 = \frac{1}{d^2}\sum_{i=1}^{d}\mathbb{E}\|\nabla\mathcal{L}_{e_i}\|^2 = \frac{n\sigma^2}{d},$$

where in the first equality we used the fact that the $\nabla\mathcal{L}_{e_i}$ are uncorrelated and in the second the fact that $\mathbb{E}[\|\nabla\mathcal{L}_{e_i}\|^2]$ is the trace of the covariance of $\nabla\mathcal{L}_{e_i}$.

Next, assume we apply the element-wise AND-mask m_t to the gradients, which puts to zero the components (dimensions) where there are less than $t \in \{d/2, \dots, d\}$ equal signs. Since Gaussians are symmetric around zero, the probability of having *exactly* u positive j-th gradient component among d environments is $Pr(p_j = u) = \left(\frac{1}{2}\right)^d \binom{d}{u}$. Hence, the probability to keep the j-th gradient direction (considering also negative consistency) is

$$\begin{aligned}
\Pr[[m_t]_j = 1] &= \sum_{u=t}^{d}\Pr(p_j = u) + \sum_{u=0}^{d-t}\Pr(p_j = u) \\
&= \left(\frac{1}{2}\right)^d \sum_{k=t}^{d}\binom{d}{k} + \left(\frac{1}{2}\right)^d \sum_{k=0}^{d-t}\binom{d}{k} \\
&= 2\left(\frac{1}{2}\right)^d \sum_{k=t}^{d}\binom{d}{k}. \qquad \text{(B.2)}
\end{aligned}$$

We would now like to compute $\mathbb{E}\left\|m_t \odot \left(\frac{1}{d}\sum_{i=1}^{d}\nabla\mathcal{L}_{e_i}\right)\right\|^2$. The difficulty lies in the fact that the event $m_t = 1$ makes gradients conditionally *dependent*. Indeed, conditioning on both $m_t = 1$ and $[\nabla\mathcal{L}_e]_j > 0$

changes the distribution of $[\nabla \mathcal{L}_{e'}]_j$: this gradient entry is going to be more likely to be positive or negative, depending on the value of $[\nabla \mathcal{L}_e]_j$ and on the details of the gradient mask. To solve the issue, we our strategy is to reduce the discussion (without loss in generality and with no additional assumption) to the case where gradient entries have all the same sign and hence conditional independence is restored.

We consider the following writing for the quantity we are interested in:

$$
\begin{aligned}
&\mathbb{E}\left\| m_t \odot \left(\frac{1}{d} \sum_{i=1}^{d} \nabla \mathcal{L}_{e_i} \right) \right\|^2 \\
&= \sum_{j=1}^{n} \mathbb{E}\left[[m_t]_j \left(\frac{1}{d} \sum_{i=1}^{d} [\nabla \mathcal{L}_{e_i}]_j \right)^2 \right] \\
&= \sum_{j=1}^{n} \sum_{\hat{p}_j=0}^{d} \mathbb{E}\left[[m_t]_j \left(\frac{1}{d} \sum_{i=1}^{d} [\nabla \mathcal{L}_{e_i}]_j \right)^2 \middle| p_j = \hat{p}_j \right] \Pr[p_j = \hat{p}_j] \\
&= \sum_{j=1}^{n} \sum_{\hat{p}_j=0}^{(d-t)} \sum_{\hat{p}_j=t}^{d} \mathbb{E}\left[\left(\frac{1}{d} \sum_{i=1}^{d} [\nabla \mathcal{L}_{e_i}]_j \right)^2 \middle| p_j = \hat{p}_j \right] \Pr[p_j = \hat{p}_j] \\
&= 2 \sum_{j=1}^{n} \sum_{\hat{p}_j=t}^{d} \mathbb{E}\left[\left(\frac{1}{d} \sum_{i=1}^{d} [\nabla \mathcal{L}_{e_i}]_j \right)^2 \middle| p_j = \hat{p}_j \right] \left(\frac{1}{2} \right)^d \binom{d}{\hat{p}_j},
\end{aligned}
$$

where we used the definition of 2-norm, the law of total expectation, and the symmetry of the problem with respect to positive and negative numbers. Finally, since the gradient components within the same envi-

ronment are conditionally independent, for any $j \in \{1, \dots, n\}$ we can write

$$\mathbb{E}\left\| m_t \odot \left(\frac{1}{d} \sum_{i=1}^{d} \nabla \mathcal{L}_{e_i} \right) \right\|^2 = 2n \sum_{\hat{p}_j = t}^{d} \mathbb{E}\left[\left(\frac{1}{d} \sum_{i=1}^{d} [\nabla \mathcal{L}_{e_i}]_j \right)^2 \middle| p_j = \hat{p}_j \right] \left(\frac{1}{2}\right)^d \binom{d}{\hat{p}_j}.$$

Finally, we note that the following bound holds:

$$\mathbb{E}\left[\left(\frac{1}{d} \sum_{i=1}^{d} [\nabla \mathcal{L}_{e_i}]_j \right)^2 \middle| p_j = \hat{p}_j \leq d \right] \leq \mathbb{E}\left[\left(\frac{1}{d} \sum_{i=1}^{d} [\nabla \mathcal{L}_{e_i}]_j \right)^2 \middle| p_j = d \right].$$

Indeed, if *all* environments lead to positive (or, symmetrically, negative) and *non-interacting* gradients in the j-th direction, the average will be the biggest in norm. Moreover — crucially — conditioned on the event $p_j = d$, gradients coming from different environments are distributed as a positive half-normal distributions. Moreover, they are *conditionally independent*; this because, since they are all positive, the value of a gradient in one environment cannot influence the value of the gradient in another one. We remark that conditional independence on the right-hand side is therefore *not an assumption*, but is intrinsic to the upper bound.

Putting it all together, we have

$$\begin{aligned}
&\mathbb{E}\left\| m_t \odot \left(\frac{1}{d} \sum_{i=1}^{d} \nabla \mathcal{L}_{e_i} \right) \right\|^2 \\
&\leq 2n \sum_{\hat{p}_j = t}^{d} \mathbb{E}\left[\left(\frac{1}{d} \sum_{i=1}^{d} [\nabla \mathcal{L}_{e_i}]_j \right)^2 \middle| p_j = d \right] \left(\frac{1}{2}\right)^d \binom{d}{\hat{p}_j} \\
&\leq 2n \sum_{\hat{p}_j = t}^{d} \sigma^2 \left(\frac{1}{2}\right)^d \binom{d}{\hat{p}_j} \\
&\leq \sigma^2 n (d-t) \binom{d}{t} \left(\frac{1}{2}\right)^{d-1},
\end{aligned}$$

where in the second line we bounded the squared average of a sum of half normal distributions: let $\{X_i\}_{i=1}^d$ be a family of uncorrelated positive half-normal distributions derived from a Gaussians with mean zero and variance σ^2, we have[2] that $\mathbb{E}[X_i] = \sigma\sqrt{2/\pi}$ and $\mathbb{E}[X_i^2] = \sigma^2$. Also, $\mathbb{E}[X_i X_j] = \mathbb{E}[X_i]\mathbb{E}[X_j] \leq \sigma^2$. Therefore,

$$\mathbb{E}\left[\left(\frac{1}{d}\sum_{i=1}^{d} X_i\right)^2\right] = \frac{1}{d^2}\sum_{i,j=1}^{d} \mathbb{E}[X_i X_j] \leq \sigma^2.$$

Finally, if we set $r = t/d \in (0.5, 1]$, we have[3]

$$\binom{d}{t} \sim \left(\frac{1}{r^r(1-r)^{1-r}}\right)^d$$

as $d \to \infty$ (discarding all polynomial terms). Hence $\binom{d}{t}$ is of the form q^d, with $1 \leq q < 2$. So, the quantity $\sigma^2 n(d-t)\binom{d}{t}\left(\frac{1}{2}\right)^{d-1}$ will be exponentially decreasing at a rate $\mathcal{O}(n/(2-q)^d)$. Notably, if $t = d/2$, then we lose the exponential rate and get back to $\mathcal{O}(n/d)$. □

2 https://en.wikipedia.org/wiki/Half-normal_distribution

3 Theorem 1 in Burić, Tomislav, and Neven Elezović. "Asymptotic expansions of the binomial coefficients." Journal of applied mathematics and computing 46.1-2 (2014): 135-145.

B.2 Appendix to Section 3.3

We used PyTorch (Paszke et al., 2017) to implement all experiments in this chapter. Our codebase is publicly available[4].

B.2.1 Section 3.3.1

Table B.1: Hyperparameter ranges for synthetic data experiments. The regularizers L1 and L2 are never combined; instead, one weight regularization type out of L1, L2 and none is selected and we sample from the respective range afterwards.

Hyperparameter	Ranges
No. hidden units	$\{256, 512\}$
No. hidden layers	$\{3, 5\}$
Batch-size	$\{64, 128, 256\}$
Optimizer	$\{\text{Adam}_{\beta_1=0.9,\beta_2=0.999}, \text{SGD + momentum}_{0.9}\}$
Learning rate	{1e-3, 1e-2, 1e-1}
Batch-normalization	{Yes, No}
Dropout	{0.0, 0.5}
L2 regularization	{1e-5, 1e-4, 1e-3}
L1 regularization	{1e-6, 1e-5, 1e-4}

B.2.2 Dataset

Here we report more technical details about the synthetic dataset described in Section 3.3. Each example is constructed as follows: we first choose the label randomly to be either $+1$ or -1, with equal probability.

4 https://github.com/gibipara92/learning-explanations-hard-to-vary

The example is a vector with $d_S + d_M$ entries, consisting of the *shortcut* and the *mechanism*. In our experiments, $d_M = 2$ and $d_S = 32$.

The Gaussian *shortcuts* are obtained by first sampling one random vector $\mathbf{x}_s \in \mathbb{R}^{d_S}$ per environment. Its components $x_{s,i}$ are sampled independently from a Normal distribution: $x_{s,i} \sim \mathcal{N}(0, 0.1)$. We use $\mathbf{x}_s$ for class 1, and $-\mathbf{x}_s$ for class -1. In the test set, all shortcut components are sampled i.i.d. from the same Normal distribution. Effectively, each example of the test set belongs to a different domain. The *mechanism* is implemented as the two interconnected spirals shown in Figure B.4 by sampling the radius $r \sim \text{Unif}(0.08, 1.0)$ and then computing the angle as $\alpha = 2\pi nr$ where n is the number of revolutions of the spiral. We add uniform noise in the range $[-0.02, 0.02]$ to the radii afterwards.

The training dataset consists of 1280 examples per environment and we use $D = 32$ environments unless otherwise mentioned. The training datasets consists of 2000 examples.

B.2.3 Experiment

We train all networks for $\lfloor 3000/D \rfloor$ epochs, dropping the learning rate by a factor 10 halfway through, and again at three-quarters of training. For computational reason, we stop each trial before completion if the training accuracy exceeds 97% and the test accuracy is below 60%. All networks are MLPs with LeakyReLU activation functions and a cross-entropy loss on the output. We run a hyperparameter search over the ranges shown in Table B.1. For IRM and the AND-mask, we select the best-performing run and re-run it 50 times with different random seeds. For DANN and the standard baselines nothing produced results significantly better than chance.

B.2.3.1 Standard regularizers and AND-mask

The networks with the L1, L2, Dropout and Batch-normalization regularizers, have hyperparameters that were randomly selected from Table B.1. For the AND-mask we used the very same ranges. The regu-

larizers L1 and L2 are never combined; instead, one weight regularization type out of L1, L2 and none is selected and we sample from the respective range afterwards. The parameters found to work best from the grid search were: agreement threshold of 1, 256 hidden units, 3 hidden layers, batch size 128, Adam with learning rate 1e-2, no batch norm, no dropout, L2-regularization with a coefficient of 1e-4, no L1-regularization. In practice, we often found it helpful to rescale the gradients after masking to compensate for the decreasing overall magnitude. We add the option for gradient rescaling as an additional hyperparameter, as we found it to help in several experiments. It rescales gradient components layer-wise after masking, by multiplying the remaining gradient components by c, where c is the ratio of the number of components in that layer over the number of non-masked components in that layer (i.e. the sum of the binary elements in the mask).[5]. We speculate that for very large layers, a less extreme normalization scheme or the additional use of gradient clipping might be appropriate.

B.2.3.2 Domain Adversarial Neural Networks

The experiments using DANN follow a similar pattern. The model consists of an embedding network, a classification network, and a "domain discrimination" network. All three modules are two-layer multi-layer perceptrons (MLP). The number of hidden units of all MLPs are sampled from the range specified in Table B.1, and we trained 100 models. Both label classifier and domain discriminator are applied to the output of the embedding network. The label classifier is trained to minimize the cross-entropy-loss between the predicted and the true label. Similarly, the domain discriminator is trained to minimize the loss between predicted and true domain-label. The embedding network is trained to minimize the regular task classification loss and at the same time to maximize the the domain-loss achieved by the domain discriminator.

5 Therefore, c is 1 if the AND-mask has only 1s, and infinite if all components are masked out (which we then keep as 0.)

B.2.3.3 Invariant Risk Minimization

For the experiments using IRM we used the authors' PyTorch implementation[6]. We perform a random hyperparameter search over with the ranges shown in Table B.2

Table B.2: Hyperparameter ranges for IRM.

Hyperparameter	Ranges
No. hidden units	$\{256, 512\}$
No. hidden layers	$\{3, 5\}$
Batch-size	$\{64, 128, 256\}$
Optimizer	$\{\text{Adam}_{\beta_1=0.9,\beta_2=0.999}, \text{SGD+momentum}_{0.9}\}$
Batch-normalization	{Yes, No}
Penalty weight	$\{10.0, 100.0, 1000.0\}$
Annealing iterations	$\{0, 1, 2, 4, 8\}$
Learning rate	{1e-3, 1e-2, 1e-1, 1}

B.2.3.4 Curves for all experiments

In Figure B.5 we show the learning curves of training and test accuracy for the different methods.

B.2.3.5 Correlation plots

For the correlation plots in Figure 3.7 we used a randomly initialized MLP with the following configuration: 3 hidden layers, 256 hidden units. The dataset was using 16 environments and batches of size 1024. The lines in Figure 3.7 are linear least-squares regressions to the gradient data shown as scatter plots. We repeat the experiment 10 times with different network weight seeds, resulting in the 10 regression lines. Zero

6 https://github.com/facebookresearch/InvariantRiskMinimization

gradients are excluded from the regression computation, as most gradients are masked out by the product mask in both cases.

B.2.4 Further visualizations and experiments

In Figure B.6 we show how many environments need to be present for the baseline without AND-mask to switch the decision boundary from the shortcuts to the mechanism. Under the same experimental condition as in the main paper, the baseline first succeeds at 1024 environments.

B.2.5 Section 3.3.2: CIFAR-10 memorization and label noise experiments

Memorization experiment In Figure B.7, we report the test performance (dashed lines) corresponding to the curves presented in the main paper for the CIFAR-10 memorization experiment. The test performance with standard labels decreases slower than the training performance as the threshold increases, and they eventually reach the same value. This is consistent with the hypothesis that by training on the consistent directions, the AND-mask selects the invariant patterns and prunes out the signals that are not invariant.

Network architecture and training details Each trial trains the ResNet "`FastResNet`" from the PyTorch-Ignite example[7] for 80 epochs on the full CIFAR-10 training set. We use the Adam optimizer with a learning rate of 5e−4, and a 0.1 learning rate decay at epoch 40 and 60. We fix the batch size to 80. We set up 14 trials by evaluating each of the AND-mask-thresholds $\{0, 0.05, 0.1, 0.2, 0.4, 0.6, 0.8\}$ for two datasets:

(a) unchanged CIFAR-10,

7 https://github.com/pytorch/ignite/blob/master/examples/contrib/cifar10/fastresnet.py

(b) CIFAR-10 with the training labels replaced by random labels. Note that a threshold of 0 corresponds to not using the AND-mask. Each trial is run twice with separate random seeds.

Label noise experiment We trained the same ResNet as for the experiment above, once with and once without the AND-mask. We ran each experiment with three different starting learning rates $\{5e{-}4, 1e{-}3, 5e{-}3\}$ and a learning rate decay at epoch 60. The baseline worked best with a learning rate of 1e−3, while the AND-mask with 5e−3, likely to compensate for the masked out gradients. The AND-mask threshold that worked best was 0.2, which is consistent with the results obtain in the experiment above.

B.2.6 Section 3.3.3: Behavioral Cloning on CoinRun

The target policy π^* is obtained by training PPO (Schulman et al., 2017) for 400M time steps using the code[8] for the paper Cobbe et al., 2020. This policy is trained on the full distribution of levels in order to maximize its generality. We use π^* to generate a behavioral cloning (BC) dataset, consisting of pairs $(s, \pi^*(a|s))$, where s are the input-images (64×64 RGB) and $\pi^*(a|s)$ is the discrete probability distribution over actions output by π^*.

The states are sampled randomly from trajectories generated by π^*. In order to test for generalization performance, the BC training dataset is restricted to 64 distinct levels. We generate 1000 examples per training level. The test set consists of 2000 examples, each from a different level which does not appear in the training set.

A ResNet-18 $\hat{\pi}_\theta$ is trained to minimize the loss $D_{\mathrm{KL}}(\pi^*||\hat{\pi}_\theta)$. We ran two automatic hyperparameter optimization studies using Tree-structured Parzen Estimation (TPE) (Bergstra et al., 2013) of 1024 trials each, with and without the AND-mask. The learning rate was decayed by a factor of 10 half-way at at $3/4$ of the training epochs.

8 https://github.com/openai/train-procgen

The "temporal" version of the AND-mask used for this experiment is reported in Algorithm 4.

Algorithm 4 Temporal AND-mask.

$\mathbf{m} \leftarrow \beta_1 \cdot \mathbf{m} + (1 - \beta_1) \cdot \mathbf{g}$

$\mathbf{v} \leftarrow \beta_2 \cdot \mathbf{v} + (1 - \beta_2) \cdot (\mathbf{g} \circ \mathbf{g})$

$\mathbf{a} \leftarrow \beta_3 \cdot \mathbf{a} + (1 - \beta_3) \cdot \texttt{elemwise\_sign}(\mathbf{g})$

$\mathbf{b} \leftarrow \mathbf{1}[|\mathbf{a}| \geq \tau]$ ▷ All operations are element-wise

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha(\mathbf{m} \circ \mathbf{b}) \oslash \sqrt{\mathbf{v} + \epsilon}$ ▷ Element-wise division

In blue we highlight the additional lines compared to traditional Adam. The threshold τ and β_3 are hyperparameters that we included in the 1024 trials of the search using Tree-structured Parsen Estimators. For the top 10 runs, hyperparameter values that were selected via the TPE search for the AND-mask are the following.

Table B.3: Hyperparameters for the 5 best runs using the AND-mask, from the TPE search.

Test KL div	lr	β_1	β_3	τ	weight decay
1.652E-2	0.0078	0.21	0.79	0.36	0.057
1.656E-2	0.0072	0.26	0.86	0.40	0.041
1.662E-2	0.0080	0.23	0.84	0.41	0.045
1.665E-2	0.0068	0.33	0.72	0.47	0.077
1.672E-2	0.0063	0.67	0.65	0.47	0.080

We found that applying weight decay as a second independent update *after* the AND-mask routine improved performance. To keep the comparison fair, we added this as a switch in the hyperparameter search for the Adam baseline as well, and it improved performance there as well.

B.3 Appendix to Section 3.4

B.3.1 Related work in causal inference

Causal graphs and causal factorizations The formalization of causality through directed acyclic graphs (Pearl, 2009a) is a key element informing our exposition. According to such formalization, a causal model gives rise to each observed distribution. It is thereby possible to exploit properties of the causal factorization of the joint probability distribution over the observed variables. Clearly, there are many ways to factorize a joint distribution into conditionals; a distinguishing feature of the causal factorization is that many of the conditionals, which we can think of as physical mechanisms underlying the statistical dependencies represented, are expected to remain *invariant* under interventions or changing external conditions. This postulate has appeared in various forms in the literature (Haavelmo, 1943; Simon, 1953; Hurwicz, 1962; Pearl, 2009a; Schölkopf et al., 2012).[9]

Causal models and robust regression Based on this insight, it was proposed that regression based on causal features should presents desirable invariance and robustness properties (Mooij et al., 2009; Schölkopf et al., 2012; Peters et al., 2016; Rojas-Carulla et al., 2018; Heinze-Deml et al., 2018; Kügelgen et al., 2019; Parascandolo et al., 2018). In this view, the mechanisms can be considered as features of the patterns such that they support stable conditional probabilities. Thus learning the mechanisms may help achieve a stable performance across a number of conditions. Other works connecting causality and learning through invariances are (Subbaswamy et al., 2019; Heinze-Deml and Meinshausen, 2017), and perhaps – most related to our work – (Arjovsky et al., 2019): we presented a comparison with this method in the following section.

9 This would be different for a non-causal factorization of the joint distribution, see Schölkopf, 2019

Causal regularization Recently (Janzing, 2019) showed that biasing learning towards models of lower complexity might in some cases be beneficial for a notion of generalization from observational to interventional regimes. Our proposed solution is however different, in that we only indirectly deal with penalizing model complexity, and rather focus on our proposed notion of consistency.

B.3.2 Learning invariances in the data

Here we are going to compare ILC to other approaches for learning invariances in the data with neural networks, and in particular to Invariant Risk Minimization (IRM) Arjovsky et al., 2019. The authors of IRM analyze a set up where minimizing training error might lead to models which absorb all the correlations found within the training data, thus failing to recover the relevant causal explanation. They consider a multi-environment setting and focus on the objective of extracting data representations that lead to invariant prediction across environments.

While the high level objective is close to the one we focused on, the differences become clear when considering the definition of *invariant predictors* presented in Arjovsky et al., 2019:

Definition 1. A data representation $\Phi : \mathcal{X} \to \mathcal{H}$ elicits an invariant predictor $w \circ \Phi$ across environments $\mathcal{E}$ if there is a classifier $w : \mathcal{H} \to \mathcal{Y}$ simultaneously optimal for all environments, i.e.,

$$w \in \arg\min_{\bar{w}:\mathcal{H}\to y} R^e(\bar{w} \circ \Phi) \; \forall e \in \mathcal{E}.$$

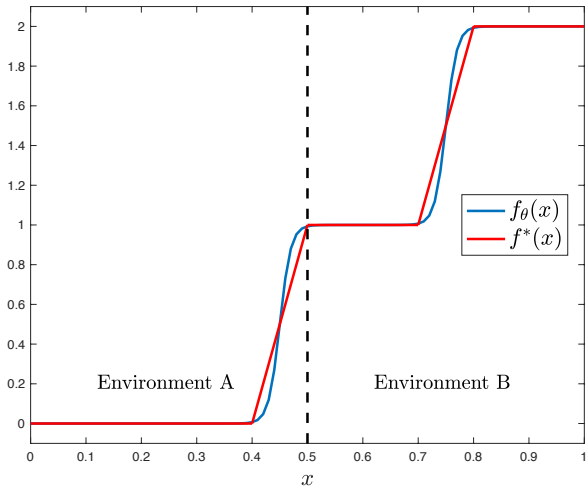
In particular, the objective minimized by IRM is:

$$\min_{\Phi:\mathcal{X}\to\mathcal{Y}} \sum_{e\in\mathcal{E}_{\text{tr}}} R^e(\Phi) + \lambda \cdot \left\|\nabla_{w|w=1.0} R^e(w \cdot \Phi)\right\|^2 \tag{B.3}$$

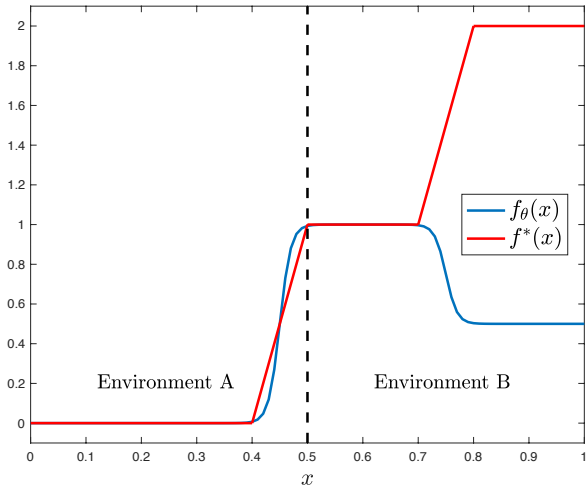
where Φ are the logits predicted by the neural network and w is a dummy scaling variable (see Arjovsky et al., 2019). The relevant part is the penalty term $\lambda \cdot \left\|\nabla_{w|w=1.0} R^e(w \cdot \Phi)\right\|^2$: One way to interpret it, is that the

penalty is large on every environment where the distribution outputted by Φ could be made 'closer' to the distribution of the labels by either sharpening ($w > 1$) or softening it (i.e., closer to uniform $w < 1$).

Let us consider the example from IRM, where the authors describe two datasets of images that each contain either a cow or a camel: In one of the datasets, there is grass on 80% of the images with cows, while in the other dataset there is grass on 90% of them. IRM then makes the point that we can learn to ignore grass as a feature, because its correlation with the label cow is inconsistent (80% vs 90%). The setting we consider in this chapter is slightly different: take our example from the CIFAR-10 experiments. Under our concept of invariance, we expect that (depending on the data generating process) even a single dataset where we treat every image as coming from its own 'environment' should be sufficient to discover invariances. Drawing a connection to the setting from IRM, we would argue that the second dataset should not be necessary to learn that 'grass' is not 'cow'. If one treats every example as coming from its own environment, there is already sufficient information in the first dataset to realize that cows are not grass: Grass is predictive of cows only in 80% of the data, so grass cannot be 'cow'. The actual cow on the other hand, should be present in 100% of the images, and as such it is the invariance we are looking for. Note that this is of course a much more strict definition of invariance: If our dataset contains images labeled as 'cows' but that have no cows within them, we might start to discard the features of cows as well.



$$\theta = \theta^* = (100, -45, 100, -75, 1, 1)$$



$$\theta = \tilde{\theta}^* = (100, -45, 100, -75, 1, -0.5)$$

Figure B.1: Performance of the neural network in Equation B.1 for two different parameters. Any reasonable modification on θ_6 (say ± 1) leaves the performance on environment A unchanged, while the performance on environment B quickly degrades.

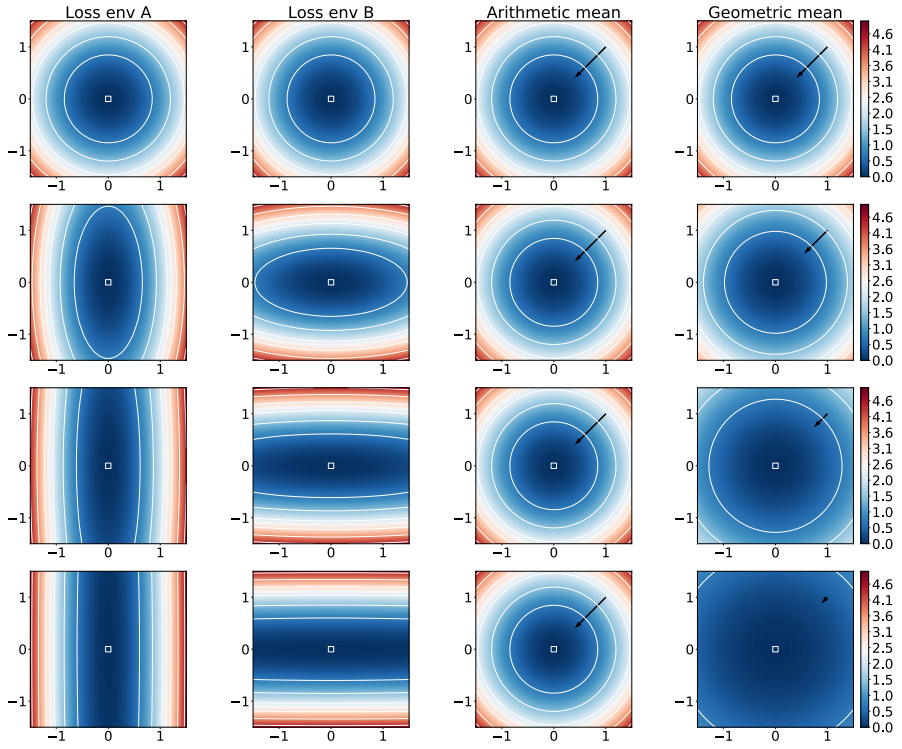


Figure B.2: While the arithmetic mean of the two loss surfaces on the left is identical in all three cases (third column), the geometric mean has weaker and weaker gradients (black arrow) the more inconsistent the two loss surfaces become.

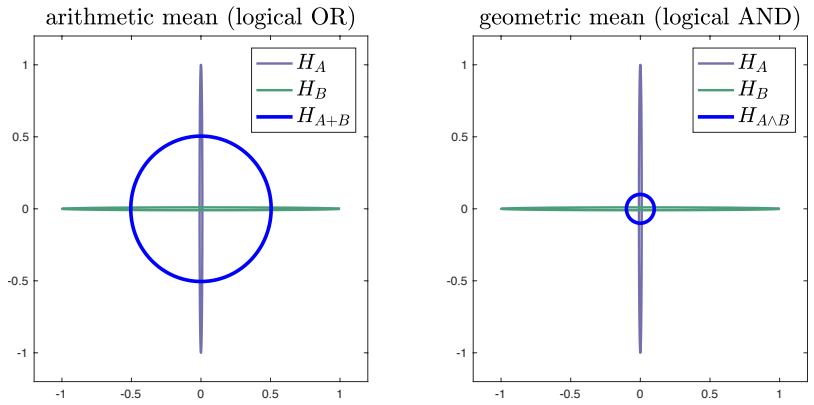


Figure B.3: Plotted are contour lines $\theta^\top H^{-1}\theta = 1$ for $H_A = \text{diag}(0.01, 1)$ and $H_B = \text{diag}(1, 0.01)$. It is convenient to provide this visualization because it is linked to the matrix determinant: $\text{Vol}(\{\theta^\top H^{-1}\theta = 1\}) = \pi\sqrt{\det(H)}$. The geometric average retains the volume of the original ellipses, while the volume of H_{A+B} is 25 times bigger. This magnification indicates that landscape A is not consistent with landscape B.

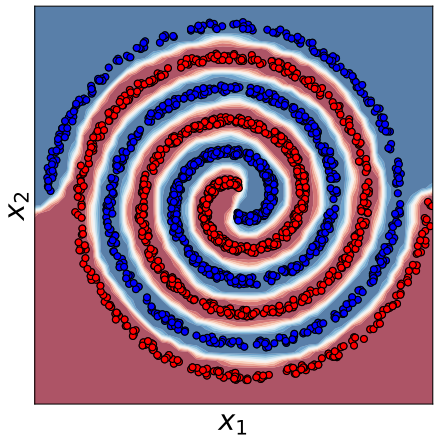


Figure B.4: The spirals used as the *mechanism* in the synthetic memorization dataset.

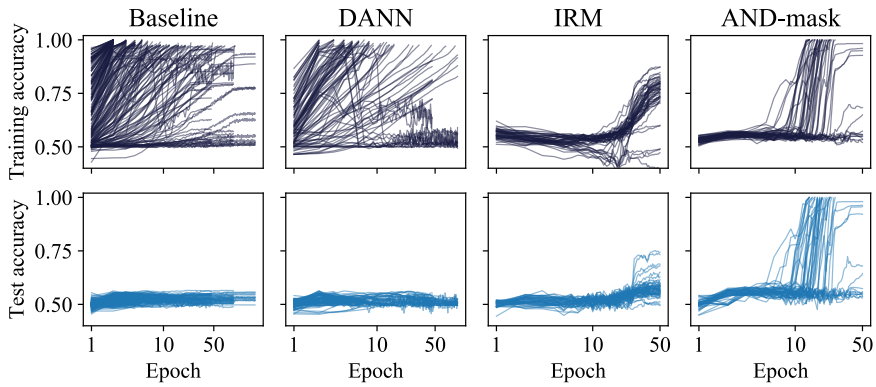


Figure B.5: Learning curves for the evaluated methods. The top row shows the accuracy on the training set, the bottom row shows the accuracy on the test set.

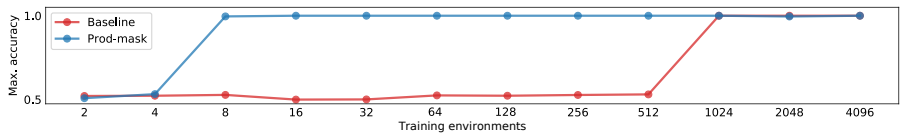


Figure B.6: Relationship between number of training environments and test accuracy for the AND-mask method compared to the baseline. We show the best performance out of five runs using the settings that were used for the experiment in the main text.

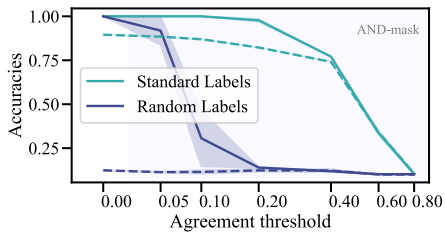


Figure B.7: Dashed lines show test acc, solid lines show training acc.

C

LDT: additional details

C.1 Implementation details

We implemented LDT using PyTorch (Paszke et al., 2019) and used the library *higher* (Grefenstette et al., 2019) to compute meta-gradients.

Our implementation of LDT contains a number of hyperparameters and settings which we describe here.

- **Inner-loop optimizer**: The optimizer used on the student's weights θ_k in the inner loop of determining the meta-gradient (includes **inner learning rate** and **inner momentum**)
- **Student optimizer**: The optimizer used to actually update the student's weights θ. (Includes **student learning rate** and **student momentum**)
- **Meta-optimizer**: The optimizer used to optimize the teacher's weights ϕ. (Includes **meta learning rate** and **meta momentum**)
- **Validation split**: The fraction of the training set that is used to compute the meta-loss.
- **Teacher architecture**: The architecture of the teacher network.

C.2 Experiment details

C.2.1 Toy datasets

Task A Construction of a training example:

- Sample $x \in \mathbb{R}^D$. $x \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- Sample $A \in \mathbb{R}^{2\times D}$ with $A_{ij} \overset{i.i.d.}{\sim} \text{Uniform}(\frac{1}{\sqrt{D}}, \frac{1}{\sqrt{D}})$ (default for `torch.nn.Linear`)
- Sample $s \in \{-1, 1\}^2$. $s \overset{i.i.d.}{\sim} 2 \cdot \text{Bernoulli}(0.5) - 1$
- $h = Ax$
- $y = \mathbf{I}[h_1 > 0] \oplus \mathbf{I}[h_2 > 0]$ where $\oplus$ denotes logical XOR.
- $x^* = [s_1,\ s_2,\ s_1 h_1,\ s_2 h_2]$
- Observed at training time: (x, x^*, y)
- Observed at test time: only x.

We perform an experiment over a grid of values for D and n, where n is the number of training examples in the dataset. The range of values for D is the set $\{64, 128, 256, 512\}$. The range of values for n are 16 exponentially-spaced integers between 128 and 1024. The student is a multi-layer-perceptron (MLP) with 32 and 128 neurons in the hidden layers, respectively. The architecture mirrors the data-generating process with a bottleneck after the first linear transformation - in particular, there is no activation function after the first hidden layer. We set up a baseline without a teacher, letting the student learn to predict y from x. For the teacher $\mathcal{T}$ in LDT, we use an MLP with $[256, 256]$ neurons in the hidden layers, ReLU activations, and an output size of 32, matching the size of the student's first hidden layer. The *teaching loss* is the mean squared error between the teacher's output and the student's first hidden activation. Additionally, the student is always trained to minimize

binary cross-entropy between its prediction and the training label. We use Adam with a learning rate of 10^{-3} as the optimizer for the student in both methods. For LDT, we use a validation split of 0.5.

Task B Construction of a training example:

- Sample $x \in \mathbb{R}^D$. $x \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- Sample $A \in \mathbb{R}^{d_h \times D}$ with $A_{ij} \overset{i.i.d.}{\sim}$ Uniform$(-\frac{1}{\sqrt{D}}, \frac{1}{\sqrt{D}})$
- Sample $B \in \mathbb{R}^{d_p \times d_h}$ with $B_{ij} \overset{i.i.d.}{\sim}$ Uniform$(-\frac{1}{\sqrt{d_h}}, \frac{1}{\sqrt{d_h}})$
- $h = Ax$
- Sample $y \in \{0..d_h - 1\}$: $y \sim$ Categorical(Softmax(h))
- $x^* = Bh$
- Observed at training time: (x, x^*, y)
- Observed at test time: only x.

For this experiment, we set $D = 128$ $d_h = 4$, $d_p = 32$, and use 1000 training examples.

The student is an MLP with input dimension D, two layers of 256 hidden units each, ReLU activations, and output dimension d_h. The teacher is an MLP with input dimension d_p, two layers of 256 hidden units each, ReLU activations, and output dimension d_h. As teaching loss, we use the KL-divergence between the softmax of the teacher's output and the softmax of the student's output activations.

Other hyperparameters used in this experiment are summarized in Table C.1.

C.2.2 MuJoCo

We now describe additional details for the MuJoCo reward-prediction experiments.

Hyperparameter	Value
Inner optimizer	Adam
Inner learning rate	1e−3
Batch size	32
Meta-optimizer	Adam
Meta-learning rate	1e−3
Teaching coefficient α	10^4
Inner loop optimization steps n	64
Validation split	0.5

Table C.1: Hyperparameters for LDT in toy task B

C.2.2.1 Dataset details

The datasets for the MuJoCo reward prediction task are generated as follows.

We use the MuJoCo environments from OpenAI gym (Brockman et al., 2016), Each training set consists of 4000 examples generated as follows.

- Reset the environment to an initial state drawn from the initial-state-distribution
- Sample an integer n uniformly from the range $\{10..30\}$
- Perform n steps following random policy π.
- Record the current state s.
- Sample a 16 step open-loop action sequence $a_{1:16}$ from π.
- Set the example's input $x = (s, a_{1:16})$
- Execute $a_{1:16}$ starting from the current state s

- Record the trajectory of states $s_{1:16}$ and rewards $r_{1:16}$ as privileged data x^*
- Record the sum of rewards as label $y = \sum_t r_t$

As random policy π we choose the policy that ignores the state and at each step and for each action dimension, samples one of the numbers $\{-1, 0, 1\}$ with equal probability, independently of each other.

We did not investigate the effects of using a different policy π to generate the dataset.

The test sets consist of 10k examples distributed identically to their respective training sets.

Similarly to Schrittwieser et al., 2020 we turn the regression task of predicting rewards into a classification task by binning the reward space. We first obtain a transformation $\psi : \mathbb{R} \to [0, 1]$ in such a way that it transforms training rewards to a uniform distribution in $[0, 1]$, interpolating linearly in-between values from the training set and clipping values outside the training-reward-range to lie between 0 and 1. We apply this transformation to both training and test labels and afterwards distribute the resulting values into 32 equally spaced bins between 0 and 1 to obtain categorical values. This leads to an even label distribution on the training set, and a roughly-even distribution on the test set. The label-loss-function is then the cross-entropy between predicted label probabilities and the one-hot distribution of the true label. The normalized mean-squared-error reported in the curves is obtained by first obtaining the expected value of the prediction by weighting the output-bucket with the predicted probability and then determining the squared difference to the true bucket value between 0 and 1.

Before training, we standardize all inputs along the dimensions individually, using empirical means and standard deviations found in the training set. Each state along the trajectory is standardized with the same normalization parameters.

C.2.2.2 Choosing hyperparameters

In order to determine good general ranges of hyperparameters[1], we first performed manual hyperparameter investigations and used a Tree-structured Parzen Estimator (Bergstra et al., 2011) search to find good ranges for the hyperparameters. Hyperparameter searches are performed using different random seeds from the final evaluation in order to reduce overfitting due to the hyperparameter-optimization. The curves for LDT shown in Figure 4.7 are the result of re-running the best hyperparameter configurations eight times with different random seeds. The curves for the baselines are obtained by performing a grid search on the hyperparameters shown in Tables C.2 and C.3, selecting the configuration that yielded the lowest test-MSE at any point in training and re-running it eight times with different random seeds.

We fix as architecture for the prediction model (the student in the LDT framework) a five-layer fully connected MLP with ReLU activations and 128 neurons per layer.

LDT parameters Tables C.5 and C.6 show hyperparameters we determined to work well for the MuJoCo reward prediction task.

We use Adam (Kingma and Ba, 2015) as optimizer for both the teacher's and student's weights. For the inner-loop student-optimizer we use SGD with momentum.

As the teacher network we use a 1D-convolutional Neural Network followed by a fully connected network as follows: The 16 states of the trajectory x^* (including rewards at every step) are fed into n_{conv} 1D-convolutional layers, each followed by a ReLU activation. The first $n_{\text{conv}}-1$ convolutional layers have c_1 output-channels, the last one has c_2 output-channels. The output of the last convolutional layer is flattened and fed into a fully connected downstream model with one layer of c_3 neurons.

1 By *hyperparameters* we mean those parameters which are fixed over the course of a training run. They do not consist of the teacher's *meta-parameters*.

In our experiments we found that it did not help to feed the actions along the trajectory to the teacher (additionally to the student who always gets to see the actions).

The output of the teacher is a vector that contains two entries for every neuron in the student network. One of these values, h_k^*, is the target-pre-activation, and m_k is a gating parameter to weight that particular neuron's loss. We found it helpful to scale and shift the input to the gating-sigmoid by two constant scalars that are used for the entire network: the loss for a given pre-activation is computed as $\sigma(\frac{m_k}{2} - 1)(h_k - h_k^*)^2$, where σ is the logistic sigmoid function, h_k is the pre-activation in the student's network, h_k^* is the target activation given by the teacher, and m_k is the gating signal output by the teacher.

We use batch-normalization (Ioffe and Szegedy, 2015) in the teacher network and weight-normalization (Salimans and Kingma, 2016) in the student network, following the findings of Such et al., 2020.

As a precaution against exploding meta-gradients, we clip each component of the meta-gradient to the range $[-1e8, 1e8]$, but did not investigate whether this was necessary.

Table C.2: Hyperparameter ranges for the method 'model-free' (MF)

Hyperparameter	Range
Learning rate	$\{1e{-}3, 1e{-}2\}$
Weight decay	$\{0, 1e{-}5, 1e{-}4\}$
Batch size	$\{8, 16, 32\}$

Performances When using the hyperparameters described above, we obtain the minimum test MSEs in the eight evaluation runs shown in Table C.7.

Table C.3: Hyperparameter ranges for the method 'auxiliary' (AUX)

Hyperparameter	Range
Learning rate (LR)	$\{1e{-}3, 1e{-}2\}$
Weight decay	$\{0, 1e{-}5, 1e{-}4\}$
Auxiliary task weight	$\{1, 2, 4\}$

Table C.4: Hyperparameter ranges for our proposed method (LDT)

Hyperparameter	Range
Adam β_1 (Meta)	$\{0.0, 0.9\}$
Inner-loop LR (SGD)	$\{1e{-}3, 5e{-}3, 1e{-}2\}$
Teaching coefficient ($\log_{10}$)	$\{2.0, 2.5, 3.0\}$
Validation split	$\{0.3, 0.5, 0.7\}$

Table C.5: Best hyperparameters found in grid search for LDT

Hyperparameter	Swimmer-v2	Walker2d-v2	Hopper-v2	HalfCheetah-v2
Adam β_1 (Meta)	0.9	0.9	0.0	0.0
Inner-loop LR (SGD)	1e−2	1e−2	1e−2	1e−2
Teaching coef. ($\log_{10}$)	2.5	3.0	2.5	2.5
Validation split	0.3	0.5	0.3	0.7

C.2.2.3 Model-based-baselines

A naive implementation of model-based could not fit the data. We trained the model within the same teacher-student framework, with a fixed dummy teacher that supervises the output of the student with the next observation (essentially mimicking a model-based setting). We 'stacked' the student such that it takes as input its output observation on the previous timestep, and by feeding it only the first observation of the se-

Table C.6: Other hyperparameters for LDT (all environments)

Hyperparameter	Value (all envs)
Learning rate of Meta-optimizer (Adam)	5e−4
Learning rate of Student-optimizer (Adam)	1e−3
Batch size	24
Momentum of inner-loop optimizer (SGD)	0.75
Weight decay (L2) coefficient in inner-loop	1e−8
Number of inner-loop steps (n_{inner})	96
Teacher c_1	96
Teacher c_2	256
Teacher c_3	768
Teacher n_{conv}	4
Number of student training steps per meta-step	24

quence, we supervise all of its intermediate targets. However, without introducing the extra inductive bias of training the student on every step *independently* of the others (instead of a single trajectory), the student's output would tend to diverge as it got deeper into the number of steps.

Training the student on pairs of consecutive transitions independently of the whole trajectory, makes the model work much better. However, making a fair comparison to the model-free, Aux, and LDT students is difficult, since the model-based student effectively uses $n = 16$ times more computation. Note that the main objective of this chapter is to compare to what extent the *abstract* models implicitly learned by the *same* student architectures but with different techniques, can learn to incorporate the trajectory information. On this basis, we exclude the model-based baselines from our comparisons.

Table C.7: Minimum mean squared errors on all environments.

	no-teacher	auxiliary	LDT
HalfCheetah-v2	0.0485	0.0477	**0.0427**
Hopper-v2	0.00270	0.00224	**0.00132**
Swimmer-v2	0.0169	0.0146	**0.00964**
Walker2d-v2	0.0238	0.0211	**0.0164**

C.3 Game of Life experiment

Here we provide details about the Game of Life experiment mentioned in Section 4.4.2.

Hyperparameter	Distribution	Min	Max	Methods
`learning_rate`	LogUniform	10^{-4}	10^{-2}	all
`adam_momentum`	1.0 - LogUniform	10^{-1}	10^{0}	all
`weight_decay`	LogUniform	10^{-8}	10^{-3}	all
`batch_size`	DiscreteUniform	32	128	all
`weight_init_multiplier`	LogUniform	0.1	2.0	all
`n_filters`	DiscreteUniform	16	32	all
`internal_coef`	LogUniform	10^{2}	10^{4}	LDT, fixed
`teacher_weight_scale`	LogUniform	10^{-2}	10^{0}	fixed
`teacher_attention_scale`	LogUniform	10^{-3}	10^{-1}	fixed
`n_inner`	DiscreteUniform	16	32	LDT
`meta_learning_rate`	LogUniform	10^{-6}	10^{-3}	LDT
`meta_momentum`	LogUniform	10^{-2}	10^{0}	LDT
`validation_split`	LogUniform	10^{-2}	10^{0}	LDT

Hyperparameter ranges The parameters for the initial teacher weights are used exclusively by the fixed-teacher method.

Note that while the meta-learning approach has more hyperparameters than the baselines, the overall computational budget given to each method is the same - more hyperparameters are not advantageous by default.

We run a hyperparameter search over the specified range for each combination of (method, n-training-examples). As hyperparameter optimization algorithm we use a Tree-structured Parzen Estimator (Bergstra et al., 2011) for each method.

Bibliography

Adolphs, Leonard, Jonas Köhler, and Aurélien Lucchi (2019). "Ellipsoidal Trust Region Methods and the Marginal Value of Hessian Information for Neural Network Training." In: *CoRR* abs/1905.09201.

Ahmed, Ossama, Frederik Träuble, Anirudh Goyal, Alexander Neitz, Manuel Wuthrich, Yoshua Bengio, Bernhard Schölkopf, and Stefan Bauer (2021). "CausalWorld: A Robotic Manipulation Benchmark for Causal Structure and Transfer Learning." In: *International Conference on Learning Representations*.

Ahuja, Kartik, Karthikeyan Shanmugam, Kush R. Varshney, and Amit Dhurandhar (2020). "Invariant Risk Minimization Games." In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 145–155.

Amodei, Dario, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané (2016). "Concrete problems in AI safety." In: *arXiv preprint arXiv:1606.06565*.

Ando, Tsuyoshi, Chi-Kwong Li, and Roy Mathias (2004). "Geometric means." In: *Linear algebra and its applications* 385, pp. 305–334.

Arjovsky, Martin, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz (2019). "Invariant risk minimization." In: *arXiv preprint arXiv:1907.02893*.

Arulkumaran, Kai, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath (2017). "A brief survey of deep reinforcement learning." In: *arXiv preprint arXiv:1708.05866*.

Badrinaaraayanan, Akilesh, Suriya Singh, Anirudh Goyal, Alexander Neitz, and Aaron Courville (2019). *Towards Jumpy Planning*. Generative Modeling and Model-Based Reasoning for Robotics and AI (ICML 2019 Workshop).

Balaji, Yogesh, Swami Sankaranarayanan, and Rama Chellappa (2018). "MetaReg: Towards Domain Generalization using Meta-Regularization." In: *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. Ed. by Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, pp. 1006–1016.

Barto, Andrew G and Sridhar Mahadevan (2003). "Recent advances in hierarchical reinforcement learning." In: *Discrete event dynamic systems* 13.1-2, pp. 41–77.

Baydin, Atilim Gunes, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind (2018). "Automatic Differentiation in Machine Learning: a Survey." In: *Journal of Machine Learning Research* 18.153, pp. 1–43.

Becker, Sue, Yann Le Cun, et al. (1988). "Improving the convergence of back-propagation learning with second order methods." In: *Proceedings of the 1988 connectionist models summer school*, pp. 29–37.

Belzner, Lenz (2016). "Time-adaptive cross entropy planning." In: *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. ACM, pp. 254–259.

Bengio, Samy, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer (2015). "Scheduled sampling for sequence prediction with recurrent neural networks." In: *Advances in Neural Information Processing Systems*, pp. 1171–1179.

Bergstra, James, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl (2011). "Algorithms for Hyper-Parameter Optimization." In: *Proceedings of the 24th International Conference on Neural Information Processing Systems*. NIPS'11. Granada, Spain: Curran Associates Inc., pp. 2546–2554.

Bergstra, James, Daniel Yamins, and David D. Cox (2013). "Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures." In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June*

2013. Vol. 28. JMLR Workshop and Conference Proceedings. JMLR.org, pp. 115–123.

Biggio, L., T. Bendinelli, A. Neitz, A. Lucchi, and G. Parascandolo (July 2021). "Neural Symbolic Regression that Scales." In: *Proceedings of 38th International Conference on Machine Learning (ICML)*. Vol. 139. Proceedings of Machine Learning Research. *equal contribution. PMLR, pp. 936–945.

Bottou, Léon, Frank E Curtis, and Jorge Nocedal (2018). "Optimization methods for large-scale machine learning." In: *Siam Review* 60.2, pp. 223–311.

Braylan, Alexander, Mark Hollenbeck, Elliot Meyerson, and Risto Miikkulainen (2015). "Frame Skip Is a Powerful Parameter for Learning to Play Atari." In: *AAAI Workshop: Learning for General Competency in Video Games*.

Brockman, Greg, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba (2016). *OpenAI Gym*. eprint: arXiv:1606.01540.

Brown, Tom et al. (2020). "Language Models are Few-Shot Learners." In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., pp. 1877–1901.

Buesing, Lars, Theophane Weber, Sébastien Racaniere, SM Eslami, Danilo Rezende, David P Reichert, Fabio Viola, Frederic Besse, Karol Gregor, Demis Hassabis, et al. (2018). "Learning and querying fast generative models for reinforcement learning." In: *arXiv preprint arXiv:1802.03006*.

Chiappa, Silvia, Sébastien Racanière, Daan Wierstra, and Shakir Mohamed (2017). "Recurrent Environment Simulators." In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Chua, Kurtland, Roberto Calandra, Rowan McAllister, and Sergey Levine (2018). "Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models." In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., pp. 4754–4765.

Cobbe, Karl, Christopher Hesse, Jacob Hilton, and John Schulman (2020). "Leveraging Procedural Generation to Benchmark Reinforcement Learning." In:

Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 2048–2056.

Cobbe, Karl, Oleg Klimov, Christopher Hesse, Taehoon Kim, and John Schulman (2019). "Quantifying Generalization in Reinforcement Learning." In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 1282–1289.

Coulom, Rémi (2006). "Efficient selectivity and backup operators in Monte-Carlo tree search." In: *International conference on computers and games*. Springer, pp. 72–83.

Craik, Kenneth James Williams (1952). *The nature of explanation*. Vol. 445. CUP Archive.

Cybenko, George (1989). "Approximation by superpositions of a sigmoidal function." In: *Mathematics of control, signals and systems* 2.4, pp. 303–314.

Daw, Nathaniel D (2012). "Model-based reinforcement learning as cognitive search: neurocomputational theories." In: *Cognitive search: Evolution, algorithms and the brain*, pp. 195–208.

Dayan, Peter (July 1993). "Improving Generalization for Temporal Difference Learning: The Successor Representation." In: *Neural Comput.* 5.4, pp. 613–624.

Deutsch, David (2011). *The beginning of infinity: Explanations that transform the world*. Penguin UK.

Du, Yunshu, Wojciech M Czarnecki, Siddhant M Jayakumar, Razvan Pascanu, and Balaji Lakshminarayanan (2018). "Adapting auxiliary losses using gradient similarity." In: *arXiv preprint arXiv:1812.02224*.

Ebert, Frederik, Chelsea Finn, Alex X. Lee, and Sergey Levine (2017). "Self-Supervised Visual Planning with Temporal Skip Connections." In: *1st Annual Conference on Robot Learning, CoRL 2017, Mountain View, California, USA, November 13-15, 2017, Proceedings*. Vol. 78. Proceedings of Machine Learning Research. PMLR, pp. 344–356.

Eshratifar, Amir Erfan, David Eigen, and Massoud Pedram (2018). "Gradient Agreement as an Optimization Objective for Meta-Learning." In: *arXiv preprint arXiv:1810.08178*.

Finn, Chelsea, Pieter Abbeel, and Sergey Levine (2017). "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks." In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, pp. 1126–1135.

Finn, Chelsea and Sergey Levine (2017). "Deep visual foresight for planning robot motion." In: *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, pp. 2786–2793.

Fort, Stanislav, Paweł Krzysztof Nowak, and Srini Narayanan (2019). "Stiffness: A New Perspective on Generalization in Neural Networks." In: *arXiv preprint arXiv:1901.09491*.

Ganin, Yaroslav, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky (2016). "Domain-adversarial training of neural networks." In: *The Journal of Machine Learning Research* 17.1, pp. 2096–2030.

Glorot, Xavier, Antoine Bordes, and Yoshua Bengio (2011). "Deep sparse rectifier neural networks." In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323.

Goodfellow, Ian J., Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. Cambridge, MA, USA: MIT Press.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). "Generative Adversarial Nets." In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger. Vol. 27. Curran Associates, Inc.

Goodhart, Charles AE (1984). "Problems of monetary management: the UK experience." In: *Monetary theory and practice*. Springer, pp. 91–121.

Grefenstette, Edward, Brandon Amos, Denis Yarats, Phu Mon Htut, Artem Molchanov, Franziska Meier, Douwe Kiela, Kyunghyun Cho, and Soumith

Chintala (2019). "Generalized Inner Loop Meta-Learning." In: *arXiv preprint arXiv:1910.01727*.

Guez, Arthur, Mehdi Mirza, Karol Gregor, Rishabh Kabra, Sebastien Racaniere, Theophane Weber, David Raposo, Adam Santoro, Laurent Orseau, Tom Eccles, Greg Wayne, David Silver, and Timothy Lillicrap (2019). "An Investigation of Model-Free Planning." In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, pp. 2464–2473.

Guez, Arthur, Fabio Viola, Théophane Weber, Lars Buesing, Steven Kapturowski, Doina Precup, David Silver, and Nicolas Heess (2020). "Value-driven Hindsight Modelling." In: *arXiv preprint arXiv:2002.08329*.

Haavelmo, T. (1943). "The statistical implications of a system of simultaneous equations." In: *Econometrica* 11.1.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2015). "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." In: *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034.

– (2016). "Deep residual learning for image recognition." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Heess, Nicolas, Greg Wayne, Yuval Tassa, Timothy Lillicrap, Martin Riedmiller, and David Silver (2016). "Learning and transfer of modulated locomotor controllers." In: *arXiv preprint arXiv:1610.05182*.

Heinze-Deml, Christina and Nicolai Meinshausen (2017). "Conditional variance penalties and domain shift robustness." In: *arXiv preprint arXiv:1710.11469*.

Heinze-Deml, Christina, Jonas Peters, and Nicolai Meinshausen (2018). "Invariant causal prediction for nonlinear models." In: *Journal of Causal Inference* 6.2.

Hermann, Katherine L. and Andrew K. Lampinen (2020). "What shapes feature representations? Exploring datasets, architectures, and training." In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-*

12, 2020, virtual. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin.

Hinton, Geoffrey, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury (2012). "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups." In: *IEEE Signal Processing Magazine* 29.6, pp. 82–97.

Hinton, Geoffrey, Oriol Vinyals, and Jeffrey Dean (2015). "Distilling the Knowledge in a Neural Network." In: *NIPS Deep Learning and Representation Learning Workshop*.

Hoover, Kevin D (1990). "The logic of causal inference: Econometrics and the conditional analysis of causation." In: *Economics & Philosophy* 6.2, pp. 207–234.

Hurwicz, L (1962). "On the structural form of interdependent systems." In: *Logic, Methodology and Philosophy of Science, Proceedings of the 1960 International Congress*. Ed. by E. Nagel, P. Suppes, and A. Tarski. Stanford, CA: Stanford University Press, pp. 232–239.

Ioffe, Sergey and Christian Szegedy (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France: JMLR.org, pp. 448–456.

Jaderberg, Max, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu (2017). "Reinforcement Learning with Unsupervised Auxiliary Tasks." In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Janzing, Dominik (2019). "Causal Regularization." In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, pp. 12683–12693.

Jastrzębski, Stanisław, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey (2017). "Three factors influencing minima in sgd." In: *arXiv preprint arXiv:1711.04623*.

Jayaraman, Dinesh, Frederik Ebert, Alexei Efros, and Sergey Levine (2019). "Time-Agnostic Prediction: Predicting Predictable Video Frames." In: *International Conference on Learning Representations*.

Kahneman, Daniel (2011). *Thinking, fast and slow*. New York: Farrar, Straus and Giroux.

Ke, Nan Rosemary, Anirudh Goyal, Olexa Bilaniuk, Jonathan Binas, Michael C Mozer, Chris Pal, and Yoshua Bengio (2018). "Sparse Attentive Backtracking: Temporal Credit Assignment Through Reminding." In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc.

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization." In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks." In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Vol. 25. Curran Associates, Inc.

Krueger, David, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Remi Le Priol, and Aaron Courville (2020). "Out-of-distribution generalization via risk extrapolation (rex)." In: *arXiv preprint arXiv:2003.00688*.

Kügelgen, Julius von, Alexander Mey, and Marco Loog (2019). "Semi-Generative Modelling: Covariate-Shift Adaptation with Cause and Effect Features." In: *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*. Ed. by Kamalika Chaudhuri and Masashi Sugiyama. Vol. 89. Proceedings of Machine Learning Research. PMLR, pp. 1361–1369.

Lee, Jason D, Max Simchowitz, Michael I Jordan, and Benjamin Recht (2016). "Gradient descent converges to minimizers." In: *arXiv preprint arXiv:1602.04915*.

Legg, Shane and Marcus Hutter (2007). "Universal intelligence: A definition of machine intelligence." In: *Minds and Machines* 17.4, pp. 391–444.

Lopez-Paz, D., B. Schölkopf, L. Bottou, and V. Vapnik (Nov. 2016). "Unifying distillation and privileged information." In: *International Conference on Learning Representations (ICLR)*.

Lorenz, Konrad (1973). *Die Rückseite des Spiegels: Versuch einer Naturgeschichte menschlichen Erkennens*. Piper.

Machado, Marlos C., Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling (Jan. 2018). "Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents." In: *J. Artif. Int. Res.* 61.1, pp. 523–562.

Mandt, Stephan, Matthew D Hoffman, and David M Blei (2017). "Stochastic gradient descent as approximate bayesian inference." In: *The Journal of Machine Learning Research* 18.1, pp. 4873–4907.

Martius, Georg and Christoph H Lampert (2016). "Extrapolation and learning equations." In: *arXiv preprint arXiv:1610.02995*.

Metz, Luke, Niru Maheswaranathan, Brian Cheung, and Jascha Sohl-Dickstein (2019). "Learning Unsupervised Learning Rules." In: *International Conference on Learning Representations*.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis (Feb. 2015). "Human-level control through deep reinforcement learning." In: *Nature* 518.7540, pp. 529–533.

Mooij, Joris M., Dominik Janzing, Jonas Peters, and Bernhard Schölkopf (2009). "Regression by dependence minimization and its application to causal inference in additive noise models." In: *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*. Ed. by Andrea Pohoreckyj Danyluk, Léon Bottou, and Michael L. Littman. Vol. 382. ACM International Conference Proceeding Series. ACM, pp. 745–752.

Moravec, Hans (1988). *Mind children: The future of robot and human intelligence*. Harvard University Press.

Muandet, Krikamol, David Balduzzi, and Bernhard Schölkopf (2013). "Domain Generalization via Invariant Feature Representation." In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*. Vol. 28. JMLR Workshop and Conference Proceedings. JMLR.org, pp. 10–18.

Müller, Meinard (2007). "Dynamic time warping." In: *Information retrieval for music and motion*, pp. 69–84.

Neitz, Alexander, Giambattista Parascandolo, Stefan Bauer, and Bernhard Schölkopf (2018). "Adaptive skip intervals: Temporal abstraction for recurrent dynamical models." In: *Advances in Neural Information Processing Systems 31*, pp. 9816–9826.

Neitz*, Alexander, Giambattista Parascandolo*, and Bernhard Schölkopf (2021). "A teacher-student framework to distill future trajectories." In: *International Conference on Learning Representations*.

Nocedal, Jorge and Stephen Wright (2006). *Numerical optimization*. Springer Science & Business Media.

Oh, Junhyuk, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh (2015). "Action-conditional video prediction using deep networks in atari games." In: *Advances in Neural Information Processing Systems*, pp. 2863–2871.

Oh, Junhyuk, Satinder Singh, and Honglak Lee (2017). "Value Prediction Network." In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., pp. 6118–6128.

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

OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang (2019).

"Solving Rubik's Cube with a Robot Hand." In: *CoRR* abs/1910.07113. arXiv: 1910.07113.

Pan, Hsiao-Ru, Nico Gürtler, Alexander Neitz, and Bernhard Schölkopf (2021). *Direct Advantage Estimation*. arXiv: 2109.06093 [cs.LG].

Parascandolo, Giambattista, Lars Buesing, Josh Merel, Leonard Hasenclever, John Aslanides, Jessica B. Hamrick, Nicolas Heess, Alexander Neitz, and Theophane Weber (2020). "Divide-and-Conquer Monte Carlo Tree Search For Goal-Directed Planning." In: *CoRR* abs/2004.11410. arXiv: 2004.11410.

Parascandolo, Giambattista, Niki Kilbertus, Mateo Rojas-Carulla, and Bernhard Schölkopf (2018). "Learning Independent Causal Mechanisms." In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 4033–4041.

Parascandolo*, Giambattista, Alexander Neitz*, Antonio Orvieto, Luigi Gresele, and Bernhard Schölkopf (2021). "Learning explanations that are hard to vary." In: *International Conference on Learning Representations*.

Paszke, Adam, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer (2017). *Automatic differentiation in pytorch*.

Paszke, Adam et al. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library." In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., pp. 8024–8035.

Pearl, J. (2009a). *Causality: Models, Reasoning, and Inference*. 2nd. Cambridge University Press.

Pearl, Judea (2009b). *Causality*. Cambridge university press.

Peters, J., D. Janzing, and B. Schölkopf (2017a). *Elements of Causal Inference - Foundations and Learning Algorithms*. Cambridge, MA, USA: MIT Press.

Peters, Jonas, Peter Bühlmann, and Nicolai Meinshausen (2016). "Causal inference by using invariant prediction: identification and confidence intervals." In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 78.5, pp. 947–1012.

Peters, Jonas, Dominik Janzing, and Bernhard Schölkopf (2017b). *Elements of causal inference: foundations and learning algorithms*. MIT Press.

Pong, Vitchyr, Shixiang Gu, Murtaza Dalal, and Sergey Levine (2018). "Temporal Difference Models: Model-Free Deep RL for Model-Based Control." In: *International Conference on Learning Representations*.

Puterman, Martin L (1994). "Markov Decision Processes." In: *Wiley and Sons*.

Quionero-Candela, Joaquin, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence (2009). *Dataset shift in machine learning*. The MIT Press.

Racanière, Sébastien, Theophane Weber, David Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adrià Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, Razvan Pascanu, Peter Battaglia, Demis Hassabis, David Silver, and Daan Wierstra (2017). "Imagination-Augmented Agents for Deep Reinforcement Learning." In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., pp. 5690–5701.

Ramesh, Aditya, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever (2021). "Zero-shot text-to-image generation." In: *arXiv preprint arXiv:2102.12092*.

Rasmussen, Carl Edward (2003). "Gaussian processes in machine learning." In: *Summer School on Machine Learning*. Springer, pp. 63–71.

Rojas-Carulla, Mateo, Bernhard Schölkopf, Richard Turner, and Jonas Peters (2018). "Invariant models for causal transfer learning." In: *The Journal of Machine Learning Research* 19.1, pp. 1309–1342.

Salimans, Tim and Durk P Kingma (2016). "Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks." In: *Advances in Neural Information Processing Systems 29*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett. Curran Associates, Inc., pp. 901–909.

Schölkopf, B., D. Janzing, J. Peters, E. Sgouritsa, K. Zhang, and J. Mooij (2012). "On Causal and Anticausal Learning." In: *Proceedings of the 29th International Conference on Machine Learning (ICML)*. Ed. by J. Langford and J. Pineau. New York, NY, USA: Omnipress, pp. 1255–1262.

Schölkopf, B., F. Locatello, S. Bauer, N. R. Ke, N. Kalchbrenner, A. Goyal, and Y. Bengio (2021). "Toward Causal Representation Learning." In: *Proceedings of the IEEE - Advances in Machine Learning and Deep Neural Networks* 109.5, pp. 612–634.

Schölkopf, Bernhard (2019). "Causality for Machine Learning." In: *CoRR* abs/1911.10500. arXiv: 1911.10500.

Schölkopf, Bernhard, Dominik Janzing, Jonas Peters, Eleni Sgouritsa, Kun Zhang, and Joris M. Mooij (2012). "On causal and anticausal learning." In: *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress.

Schrittwieser, Julian, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. (2020). "Mastering atari, go, chess and shogi by planning with a learned model." In: *Nature* 588.7839, pp. 604–609.

Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov (2017). "Proximal policy optimization algorithms." In: *arXiv preprint arXiv:1707.06347*.

Silver, David, Hado van Hasselt, Matteo Hessel, Tom Schaul, Arthur Guez, Tim Harley, Gabriel Dulac-Arnold, David Reichert, Neil Rabinowitz, Andre Barreto, and Thomas Degris (2017). "The Predictron: End-To-End Learning and Planning." In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, pp. 3191–3199.

Silver, David, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. (2016). "Mastering the game of Go with deep neural networks and tree search." In: *nature* 529.7587, pp. 484–489.

Simon, H. A. (1953). "Causal Ordering and Identifiability." In: *Studies in Econometric Methods*. Ed. by W. C. Hood and T. C. Koopmans. Cowles Commission for Research in Economics, Monograph No. 14. New York, NY: John Wiley & Sons, pp. 49–74.

Singh, Sidak Pal and Dan Alistarh (2020). "WoodFisher: Efficient second-order approximations for model compression." In: *arXiv preprint arXiv:2004.14340*.

Subbaswamy, Adarsh, Peter Schulam, and Suchi Saria (2019). "Preventing Failures Due to Dataset Shift: Learning Predictive Models That Transport." In: *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*. Ed. by Kamalika Chaudhuri and Masashi Sugiyama. Vol. 89. Proceedings of Machine Learning Research. PMLR, pp. 3118–3127.

Such, Felipe Petroski, Aditya Rawal, Joel Lehman, Kenneth Stanley, and Jeffrey Clune (2020). "Generative Teaching Networks: Accelerating Neural Architecture Search by Learning to Generate Synthetic Training Data." In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 9206–9216.

Sugiyama, Masashi and Motoaki Kawanabe (2012). *Machine learning in non-stationary environments: Introduction to covariate shift adaptation*. MIT press.

Sugiyama, Masashi, Matthias Krauledat, and Klaus-Robert Müller (2007). "Covariate shift adaptation by importance weighted cross validation." In: *Journal of Machine Learning Research* 8.May, pp. 985–1005.

Sutton, Richard S. (Aug. 1988). "Learning to Predict by the Methods of Temporal Differences." In: *Mach. Learn.* 3.1, pp. 9–44.

Sutton, Richard S (1991). "Dyna, an integrated architecture for learning, planning, and reacting." In: *ACM Sigart Bulletin* 2.4, pp. 160–163.

Sutton, Richard S and Andrew G Barto (2018). *Reinforcement learning: An introduction*. MIT press.

Sutton, Richard S., Joseph Modayil, Michael Delp, Thomas Degris, Patrick M. Pilarski, Adam White, and Doina Precup (2011). "Horde: A Scalable Real-Time Architecture for Learning Knowledge from Unsupervised Sensorimotor Interaction." In: *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*. AAMAS '11. Taipei, Taiwan: International Foundation for Autonomous Agents and Multiagent Systems, pp. 761–768.

Sutton, Richard S, Doina Precup, and Satinder Singh (1999). "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning." In: *Artificial intelligence* 112.1-2, pp. 181–211.

Team, Open Ended Learning, Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michaël Mathieu, Nat McAleese, Nathalie Bradley-Schmieg, Nathaniel Wong, Nicolas Porcel, Roberta Raileanu, Steph Hughes-Fitt, Valentin Dalibard, and Wojciech Marian Czarnecki (2021). "Open-Ended Learning Leads to Generally Capable Agents." In: *CoRR* abs/2107.12808. arXiv: 2107.12808.

Todorov, Emanuel, Tom Erez, and Yuval Tassa (2012). "MuJoCo: A physics engine for model-based control." In: *IROS*. IEEE, pp. 5026–5033.

Vapnik, Vladimir N. (n.d.). *The nature of statistical learning theory*. Springer-Verlag New York, Inc.

Vapnik, Vladimir (2006). *Estimation of dependences based on empirical data*. Springer Science & Business Media.

Vapnik, Vladimir and Akshay Vashist (2009). "A new learning paradigm: Learning using privileged information." In: *Neural networks* 22.5-6, pp. 544–557.

Veeriah, Vivek, Matteo Hessel, Zhongwen Xu, Janarthanan Rajendran, Richard L Lewis, Junhyuk Oh, Hado P van Hasselt, David Silver, and Satinder Singh (2019). "Discovery of Useful Questions as Auxiliary Tasks." In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett. Curran Associates, Inc., pp. 9310–9321.

Vinyals, Oriol, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. (2019). "Grandmaster level in StarCraft II using multi-agent reinforcement learning." In: *Nature* 575.7782, pp. 350–354.

Watter, Manuel, Jost Tobias Springenberg, Joschka Boedecker, and Martin Riedmiller (2015a). "Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images." In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'15. Montreal, Canada: MIT Press, pp. 2746–2754.

Watter, Manuel, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller (2015b). "Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images." In: *Advances in Neural Information Processing*

Systems. Ed. by C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett. Vol. 28. Curran Associates, Inc.

Xu, Zhongwen, Hado P van Hasselt, and David Silver (2018). "Meta-Gradient Reinforcement Learning." In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., pp. 2396–2407.

Zhang, Chiyuan, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals (2017). "Understanding deep learning requires rethinking generalization." In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Zhang, Guodong, Lala Li, Zachary Nado, James Martens, Sushant Sachdeva, George E. Dahl, Christopher J. Shallue, and Roger B. Grosse (2019a). "Which Algorithmic Choices Matter at Which Batch Sizes? Insights From a Noisy Quadratic Model." In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, pp. 8194–8205.

Zhang, Yunbo, Wenhao Yu, and Greg Turk (2019b). "Learning Novel Policies For Tasks." In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, pp. 7483–7492.