Adaptive Reinforcement Learning

LEAN AND DYNAMIC AGENTS FOR ROBUST GENERALIZATION

Bram Grooten

§

Adaptive Reinforcement Learning

Lean and Dynamic Agents for Robust Generalization

Bram Grooten

A catalogue record is available from the Eindhoven University of Technology Library.

ISBN 978-90-386-6662-4
NUR 984

Cover design by Bram Grooten.
Printed and bound by ADC Nederland.

SIKS Dissertation Series No. 2026-27.
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

Adaptive Reinforcement Learning

Lean and Dynamic Agents for Robust Generalization

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van
de rector magnificus prof. dr. S.K. Lenaerts,
voor een commissie aangewezen door het College
voor Promoties, in het openbaar te verdedigen op
8 april 2026 om 16:00 uur

door

Bram Joris Grooten
geboren te Eindhoven

Dit proefschrift is goedgekeurd door de promotoren en de samenstelling van de promotiecommissie is als volgt:

voorzitter:	prof. dr. ir. B.F. van Dongen
promotoren:	prof. dr. M. Pechenizkiy
	dr. D.C. Mocanu (University of Luxembourg)
leden:	dr. ir. J. Vanschoren
	prof. dr. M.T.J. Spaan (Technische Universiteit Delft)
	prof. dr. L. Tran-Thanh (University of Warwick)
	prof. dr. M.E. Taylor (University of Alberta)
adviseur:	dr. G.A.Z.N. Sokar (Google DeepMind, TU Eindhoven)

Het onderzoek of ontwerp dat in dit proefschrift wordt beschreven is uitgevoerd in overeenstemming met de TU/e Gedragscode Wetenschapsbeoefening.

Voor Inge.

§

"Can we wonder, then, that nature's productions should be far "truer" in character than man's productions; that they should be infinitely better adapted to the most complex conditions of life, and should plainly bear the stamp of far higher workmanship?"

—Charles Darwin
On the Origin of Species (1859)

"The survival of the fittest is a slow method for measuring advantages. The experimenter, by the exercise of intelligence, should be able to speed it up."

—Alan Turing
Computing Machinery and Intelligence (1950)

§

Summary

In a rapidly changing world, intelligent agents require the ability to adapt to the evolving environment. In this thesis, I explore various approaches to improve the generalization capacity of reinforcement learning (RL) agents.

At first, we dive into the domain of noisy environments, where only a small subset of the inputs to an agent is relevant to its goal. We show that through dynamic sparse training, the agent's neural network can learn to adapt its inner structure, focusing most connections on task-relevant features. This approach outperforms regular dense neural networks, which struggle to ignore the input noise.

Secondly, we expand our problem setting to the visual domain. We strive to develop agents that are able to perform a task well in any room or space, even though they may have learned the task in only one location. Through a simple but effective architectural adjustment, we demonstrate that deep neural networks improve their ability to adapt to varying background visuals without being distracted.

Thirdly, we examine the field of contextual RL. We aim to create agents that can adapt to any contextual setting within high-dimensional continuous control problems—e.g., drive any car—even without knowing the explicit context (car) in which it is tested. Our contribution demonstrates that by incorporating a brief history of recent observations and actions, neural networks are able to infer the environment's setting and enhance their adaptability to unseen contexts.

Finally, ensembling has proven to be a powerful approach to increase generalization capacity. We explore the use of adaptability in this direction and invent a technique that makes ensembles more efficient and effective. We share the backbone between ensemble members and dynamically adjust the sparse topology of the multi-headed architecture during training. This method shows improved performance on multiple environments as well as various vision and language tasks.

Overall, this thesis demonstrates that the skill of adaptability can be promoted inside the structure of a neural network in multiple lean and dynamic ways. I hope these contributions help to create reliable, adaptive, and intelligent agents.

§

Contents

§

CHAPTER 1
Introduction

1.1 ADAPTATION

Adapting to change is one of the core abilities of an intelligence, be it natural or artificial. In my own life, two of the most significant changes have, coincidentally, happened near the beginning and end of my PhD. The experience I have gathered through these circumstances has possibly helped me endow reinforcement learning agents with greater adaptability. This thesis will go through the details of these ideas. I hope you enjoy reading it.

The title of this book, *Adaptive Reinforcement Learning*, might be a bit of a paradox. Aren't reinforcement learning (RL) agents already able to adapt to their environment? In many cases; yes. But I will highlight circumstances in which we can improve their adaptability, while aiming to keep the algorithms simple and scalable, staying true to the *Bitter Lesson* [Sut19].

One of the first questions to ask is: what do we mean exactly, when discussing adaptability? I will highlight two distinct meanings, that are both used in this work. On the one hand, with *internal* adaptation, I refer to the ability of an agent to adjust its own software and hardware. Learning the weights of an agent's neural network through stochastic gradient descent is an example that provides some level of internal adaptability. We will see other methods, such as dynamic sparse training, that can further increase the internal adaptation ability.

On the other hand, *external* adaptation measures the ability of an agent to cope with adjustments in its environment. For instance, if a kitchen setup is changed, will a household robot still be able to complete its task of filling the dishwasher? What if we merely repaint the color of the walls? When an agent is able to maintain performance under large environment adjustments, it exhibits a high level of external adaptability.

This chapter was adapted from: Bram Grooten. Large Learning Agents: Towards Continually Aligned Robots with Scale in RL. In *Doctoral Consortium at Autonomous Agents and Multiagent Systems*, 2024.

Most of the time in this thesis, the reader may assume we are discussing the external adaptation ability, as it is often a direct objective and straightforward to measure. Internal adaptability is usually only an indirect desideratum. However, one might imagine that having a high level of internal adaptability can benefit an agent's external adaptation. We will see some examples throughout the following chapters.

1.2 RESEARCH OBJECTIVES AND APPROACH

The methodology used in this work follows the empirical practice of the deep reinforcement learning community. We construct hypotheses based on intuition and prior work, implement novel algorithms, and verify them through rigorous experimentation. These experiments span a diverse set of domains, ranging from continuous control benchmarks and high-fidelity racing simulators to physical robotic arms.

The overarching objective of this research is to enhance the adaptability of RL agents. Specifically, we aim to achieve this through the design of lean and dynamic agents that demonstrate robust generalization. As these terms form the subtitle of this thesis, let us define how they are interpreted in the context of this work.

Generalization. This is the ultimate test of external adaptability. By *generalization*, we refer to the agent's ability to perform well in environments that differ from those encountered during training. While standard RL often evaluates agents in the exact same environment they were trained in, we explicitly test on unseen scenarios.

Robustness. Closely related to generalization is *robustness*, which describes an agent's resilience against disturbances. A robust agent maintains its performance despite the presence of perturbations. Throughout this thesis, we tackle different forms of these disturbances: the noisy irrelevant inputs in Chapter 2, the visual distractions in Chapter 3, the physical context shifts in Chapter 4, and the distribution shifts on unseen datasets in Chapter 5.

We combine the previous two concepts to strive for *robust generalization*. We define this as a strong version of generalization, where an agent is capable of coping with *large* disturbances, rather than just small ones. It implies that the agent does not merely memorize a clean training environment or handle minor noise, but learns the underlying task structure sufficiently well to survive significant shifts in the environment's dynamics.

Lean. To achieve robust generalization without significantly increasing computational costs, we aim for *lean* agents. A lean agent is efficient in its parameter

usage and computational throughput. In Chapter 2 and Chapter 5, we achieve this efficiency through sparse neural networks that use a fraction of the parameters of dense models. In Chapter 3 and Chapter 4, we seek architectural simplicity, adding only lightweight modules or condensing training into a single phase.

Dynamic. Finally, our agents are *dynamic*. This term encapsulates the mechanism of internal adaptation. It refers to the ability of the neural networks to adjust their internal structure or immediate behavior on the fly. This manifests as *dynamic sparse training*—where neural network connections are pruned and regrown during learning—or as *context-aware inference*, where the agent dynamically infers the environment's setting from history. By being dynamic, the agent avoids becoming a static, brittle system and instead remains fluid enough to enable continual learning.

1.3 CONTRIBUTIONS AND OUTLINE

A standard RL agent, whose artificial brain nowadays often consists of deep, fully-connected (dense) neural networks, is not able to adapt well to noisy environments. In Chapter 2, we introduce the *Extremely Noisy Environment* (ENE). The states in these environments consist of many, many features, most of which are irrelevant to the agent's goal or task. In an ENE, the agent must learn to filter through the noise and focus on task-relevant features. We show that dense neural networks really start to struggle on such problems when the fraction of relevant features drops to around $1/5$ of the total inputs, which is not particularly low. Imagine if we could simply provide our future household robot with all the information about the house (e.g., temperature in each room, stock in the fridge, hygiene of the floors, etc.), and let it figure out by itself which few features it needs for its current task.

The algorithm we created, *Automatic Noise Filtering* (ANF), does exactly that. By making the input layer of the agent's neural network *sparse*, we already decrease the amount of distractions coming in. But what really helps the agent focus is the use of dynamic sparse training: it updates the sparse topology during training. The method prunes useless connections with small weights, while growing new connections randomly in an evolutionary way. Our analysis shows that ANF is swiftly able to adapt its network structure to focus on task-relevant features, which greatly benefits the agent's performance. Furthermore, in a continual learning setting, where the relevant inputs change from one task to another, our results demonstrate that ANF has the quickest adaptability.

This sparse approach works well in environments where states consist of a large, one-dimensional vector of features. However, what if the robot's input is vision-based? In this case a different strategy is necessary, because the relevancy of an input pixel can change from one timestep to the next. Learning from vision requires faster adaptability than the evolutionary technique of dynamic sparse training.

The problem setting that we focus on in Chapter 3 is vision-based generalization. Can we train an agent in a relatively clean environment, and deploy it zero-shot in test cases with multiple moving visual distractions in the background? Yes, our algorithm called *Masking Distractions* (MaDi), is able to adapt to such unseen circumstances.

MaDi, similar to other baseline methods, uses data augmentation to be prepared for inputs that are less clean than the training environment. But on top of this, it consists of a small architectural adjustment: MaDi adds a Masker network to the general actor-critic structure. The Masker's job is to find out which pixels are relevant to the agent's goal, and which are merely distractions. It learns to output a value for each pixel: close to 1 for relevant pixels, near 0 for the others. This soft mask is then multiplied element-wise with the original input, to create a masked version of the image where relevant parts are highlighted. This eases the task of the actor and critic modules, which can now increase their focus on learning an adequate policy and value function for the task, instead of figuring out which state the environment is in.

We test MaDi in many simulation environments, and on a real robotic arm (UR5). For the UR5, we train directly on the robot itself, in an online asynchronous manner. We attach a webcam to the tip of the arm, and reward it for reaching towards a red dot that is randomly located on a white screen. It learns to do so in about 2 hours. Then we swap the white background for a distracting video, and see if it can still find the target. MaDi demonstrates the best generalization ability in adapting to such out-of-distribution (OOD) scenarios.

Up till now, the OOD tasks only changed irrelevant aspects of the environment. But what if the adjustable parts are actually task-relevant? For example, if an RL agent is trained to drive around a racetrack in a few different cars, can it adapt to any new vehicle? We say that this strongly changes the *context* of the environment. In Chapter 4 we are working in the area of contextual reinforcement learning.

In many robotic applications, the context can be known during training, but unknown at test-time. For instance, consider a quadruped that needs to learn to walk on many different terrains. We might train it in simulation, where we can vary the exact contextual settings of friction, mass, terrain height, etc. In the real world, many of these parameters are unknown a priori, and will even change continually. We thus

aim to create an algorithm that can make use of this privileged contextual information during training, while not needing it at deployment.

Our method, *Single-Phase Adaptation for Robust Control* (SPARC), is able to do just that. It trains two policies, an expert and an adapter, in different ways. The expert policy is trained with context information as additional input (next to the standard observation). The adapter policy receives additional history (previous observations and actions) instead of the context. The adapter policy's *history* module is then tasked with trying to output the same latent encoding as the expert policy's *context* module. In this way the adapter policy is able to infer the environment's context, merely from a short window of previous timesteps.

The particularly novel part of SPARC, is that both policies can be trained simultaneously (i.e., in a single phase) in contrast to previous work. This greatly simplifies implementation and opens up the ability to learn continually. We test on the high-fidelity racing simulator *Gran Turismo 7*, and show that SPARC performs especially well on over 100 unseen vehicles.

Finally, when adapting to so many new situations, we seek to use a method that can further reduce the agent's uncertainty. In Chapter 5 we turn to the power of ensembling, which teaches us that the average of many neural networks often outperforms any individual model. We aim to increase the effectiveness and efficiency of ensembles, such that the compute required is still similar to that of a single network.

Our method, NeuroTrails, makes two impactful adjustments to a standard ensemble. First, it merges the early layers of each ensemble member into one backbone. Second, we sparsify the entire remaining multi-headed network. Through dynamic sparse training, NeuroTrails is able to find diverse pathways, or neural trails, that produce distinct outcomes in the heads. By setting the right splitting point of the backbone, we show in our analysis that we are able to tune the prediction diversity of our heads. This enables us to find a *Goldilocks zone* of prediction diversity (not too much, not too little) where the ensembling approach performs best. We show that NeuroTrails outperforms many other ensembling techniques in not only reinforcement learning, but also other fields such as computer vision (supervised learning) and language modeling (self-supervised learning).

The necessary background for each topic is included in the corresponding chapters as a section on preliminary material. Each chapter is relatively self-contained and can be read independently of the others. Let's dive in.

§

CHAPTER 2

Adapting to Noisy Environments

Tomorrow's robots will need to distinguish useful information from noise when performing different tasks. A household robot for instance may continuously receive a plethora of information about the home, but needs to focus on just a small subset to successfully execute its current chore.

Filtering distracting inputs that contain irrelevant data has received little attention in the reinforcement learning literature. To start resolving this, we formulate a problem setting in reinforcement learning called the *extremely noisy environment* (ENE), where up to 99% of the input features are pure noise. Agents need to detect which features provide task-relevant information about the state of the environment.

Consequently, we propose a new method termed *Automatic Noise Filtering* (ANF), which uses the principles of dynamic sparse training to increase its internal adaptability level. The sparse input layer learns to focus its connectivity on task-relevant features, such that ANF-SAC and ANF-TD3 outperform standard deep RL algorithms SAC and TD3 by a large margin, while using up to 95% fewer weights.

Furthermore, we devise a continual learning setting for ENEs, by permuting all features of the environment after 1M timesteps to simulate the fact that other information sources can become relevant as the world evolves. Again, ANF surpasses the baselines in final performance and sample complexity.

This chapter was adapted from: Bram Grooten, Ghada Sokar, Shibhansh Dohare, Elena Mocanu, Matthew E. Taylor, Mykola Pechenizkiy, and Decebal Constantin Mocanu. Automatic Noise Filtering with Dynamic Sparse Training in Deep Reinforcement Learning. *Int. Conf. Autonomous Agents and Multiagent Systems (AAMAS)*, 2023.

2.1 INTRODUCTION

Future robots will likely perceive ample information about the state of the world, but only parts of it will be relevant to their current task. Imagine a household robot observing the current status of all objects and processes in the house.[1] For its current task, e.g. making pancakes, only a small subset of these information sources, or

[1] For example: cleanliness of floors, cupboards, kitchen utensils; CO_2, CO levels and temperature in each room; up-to-date stock of all food and non-food items in the fridge and basement; mood, nourishment, and health of all inhabitants; etc.

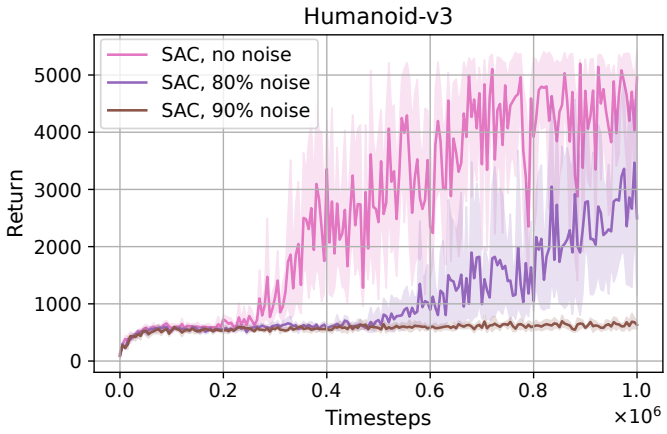


Figure 2.1. Performance of SAC on Humanoid-v3 environments expanded with a different number of pure noise features. Once the environment contains too much noise, SAC struggles to learn a decent policy. Standard dense networks cannot filter the noise well enough in this problem setting.

features, are relevant. Agents should automatically detect which features are task-relevant, without humans having to predefine this. Other examples may be: a hearing aid distinguishing between voices and auditory noise, a surgical robot receiving all possible information about the patient, or a self-driving car that needs to ignore distracting billboards.

To illustrate the current situation: Soft Actor-Critic (SAC) [HZAL18] fails to learn a decent policy on an environment with 90% added noise features, see Figure 2.1. We simulate the noisy real-world environment by adding synthetic noise features to an existing state space. This allows us to study the problem in a controlled environment to understand where we stand and what can be done. We need to invent methods that can effectively filter through the noise while learning to perform the environment's task. Our **research question** becomes: *How can we design RL agents to learn and perform well in an extremely noisy environment?*

Dynamic Sparse Training (DST), a class of methods stemming from the Sparse Evolutionary Training (SET) algorithm [MMS+18], is promising in this regard. By starting from a randomly sparsified network and subsequently pruning and growing connections (weights) during training, DST searches for the optimal network topology. DST is able to perform efficient feature selection for unsupervised learning, as shown by [ASvdL+22, SAPM22]. Further, [VLS22] discovered that sparse networks can find minimal task representations in deep RL by pruning redundant input dimensions. Not long after, [SMM+22] successfully applied DST in deep RL, reducing the number

of parameters without compromising performance.

This leads us to a plausible approach to our research question. We think that the adaptability of DST can improve an agent's sparse network structure such that task-relevant features are emphasized by receiving more connections than noise features. The combination of sparsity and adaptability enables the agent to filter through the noise more effectively, outperforming dense network approaches. The underlying hypothesis follows:

The Adaptability Hypothesis: *A sparse neural network layer can adapt the location of its connections (weights) to gain a better performance faster than a dense layer can adapt the weight values to achieve the same gain.*

Note that newly grown connections still need to adjust their weight values through gradient descent, but we hypothesize that this generally happens quicker than a dense network modifies all of its weights. Relocated weights may receive a more informative gradient when connected to task-relevant features. Briefly, the hypothesis states: dropping and growing connections is easier than adjusting the weights. This is inspired by our own brain's plasticity, which also dynamically drops and grows synapses [ByRCS+69, CLM20, Per14].

To verify our hypothesis, we propose a new algorithm called Automatic Noise Filtering (ANF), which can easily be combined with deep RL methods. It has a sparse input layer with adapting connectivity through dynamic sparse training. We compare ANF to two strong baseline deep RL algorithms: SAC [HZAL18] and TD3 [FHM18], which have fully dense layers throughout their networks. We devise the *extremely noisy environment* (ENE), further defined in Section 2.2, which expands the state space of an existing RL environment with a large number of noise features. We apply this approach to four continuous control tasks from MuJoCo Gym [TET12, BCP+16].

Contributions.

- We formulate a problem setting termed the *extremely noisy environment* (ENE), where up to 99% of the input features consist of pure noise. Agents need to detect the task-relevant features autonomously.
- We propose Automatic Noise Filtering (ANF), a dynamic sparse training method that outperforms baseline deep RL algorithms by a large margin, especially on environments with high noise levels.
- We devise a continual learning setting of extremely noisy environments and

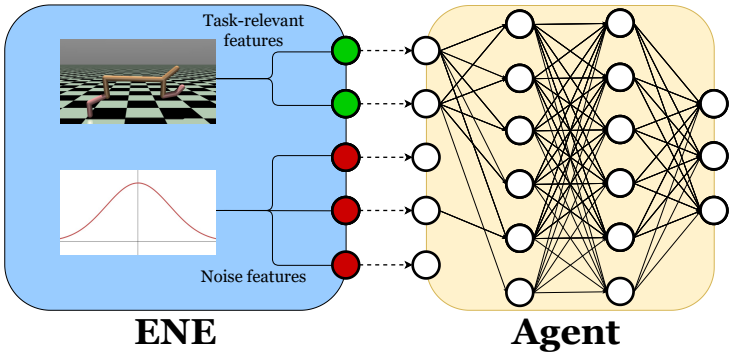


Figure 2.2. Extremely noisy environments (ENEs) contain many noise • features and some relevant • features. We use ENEs where up to 99% of the features are noise. Our method Automatic Noise Filtering (ANF) learns to predominantly connect with the input neurons that provide useful information and outperforms dense baselines by a large margin, especially in the noisiest environments.

show that ANF has better performance and forward transfer than the baselines SAC and TD3.

- We show that highly sparse ANF agents with up to 95% fewer parameters can still surpass their dense baselines on the extremely noisy environments.
- We extend the ENE by adjusting the noise distribution in two ways to increase the difficulty. ANF maintains its advantage on these challenging extensions.[2]

2.2 PROBLEM FORMULATION

We introduce a problem setting where agents have to act in environments that contain a lot of noise. As the noise features generally greatly outnumber the task-relevant features in this setting, we simply call it the extremely noisy environment (ENE).

Extremely noisy environment. To create an ENE, we take any reinforcement learning environment that generates feature vectors as states. The ENE expands this feature vector by concatenating many additional features consisting only of pure noise, sampled from any given distribution. An agent is not told which features are useful (task-relevant) and which are useless (noise), so it has to learn to ignore the distracting noise features by itself, see Figure 2.2.

In our main experiments, the noise features produce pure Gaussian noise, sam-

[2] See an illustrative video here: https://youtu.be/vS47UnsTQk8

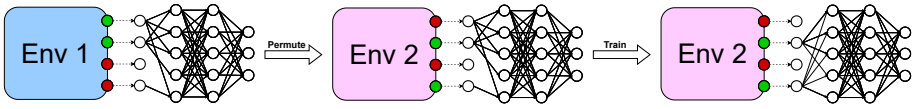


Figure 2.3. In *permutated* extremely noisy environments (PENE) the order of the input features gets shuffled after a certain number of timesteps. Our ANF agents automatically adjust their network structure to this new environment. They show superior forward transfer compared to fully dense methods, even though ANF agents might need to prune and regrow many connections in the sparse input layer. We hypothesize that adapting the location of sparse connections is easier than adjusting the weights of all connections in a dense network.

pled i.i.d. from $\mathcal{N}(0, 1)$. The fraction of noise features in an ENE is denoted by $n_f \in [0, 1)$. For example, for $n_f = 0.5$ we enlarge the original state space of a MuJoCo Gym environment by lengthening the state feature vector by a factor of 2. In general, the dimensionality of the new state space is

$$d_{ene} = \left\lceil \frac{d_{og}}{1 - n_f} \right\rceil$$

where d_{og} is the number of dimensions in the original state space. As n_f increases to 1, the dimensionality of the ENE expands.

Continual learning setting. Next to the ENE, we introduce an even more challenging problem setting where, after every T_p timesteps, all input features are permuted at random. This permutation simulates the fact that other features can become relevant over time. Previously irrelevant features might suddenly become relevant, for example, when a household robot gets a new task.[3] In our case, the change in environment is *not* announced to the agent. Agents need to detect the change and transfer their representations quickly to adapt to the new instantiation of the *permutated extremely noisy environment* (PENE), see Figure 2.3. A previously task-relevant feature may or may not still be relevant after the permutation, inducing the need to rediscover the distribution of the features and filter through the noise.

2.3 BACKGROUND AND RELATED WORK

Our proposed ANF algorithm is based on dynamic sparse training (DST). In this section, we briefly overview the related work of DST in reinforcement learning (RL) and existing noise filtering methods.

[3] Features could also gradually become more relevant, as the world evolves (i.e., concept drift). This is outside the scope of our research; we focus on the sudden change.

Sparse training. Dynamic sparse training is a subfield of the sparse training regime [MMP+21], where weights deemed superfluous are pruned away to increase the efficiency of a neural network. In dense-to-sparse training, dense networks are gradually pruned to higher sparsity levels throughout training [HPTD15, FC19, LXS+20]. In sparse-to-sparse training, where DST belongs, a network begins with a high sparsity level from scratch [MMS+18, BKML18]. The existing connections can either stay fixed (*static* sparse training) or be pruned and regrown during training (*dynamic* sparse training).

In supervised learning, especially computer vision, many promising results have been achieved with sparsity over the last few years [LYMP21, EGM+20, CCG+21]. These algorithms benefit from potential performance boosts, decreased computational costs, and better generalization [LMP19, CZW+22]. Furthermore, DST has been used successfully for an efficient feature selection algorithm [ASvdL+22], which inspired our project.

DST in RL. Applying sparse training in reinforcement learning is useful, as real-world applications often deal with latency constraints [DFB+22], which limits the number of parameters. Unfortunately, in the area of RL it seems that applying sparse training is more challenging than in supervised learning, as the achievable sparsity levels without loss in performance are generally lower [GEEC22]. Only a few works have applied sparse training to deep RL so far. In the *offline* RL setting, [AOPP21] have reached 95% sparsity with almost no performance degradation. While this is impressive, we believe that offline RL is more similar to supervised learning than online RL. Moreover, it does not support learning in changing environments [SSPS21]. Therefore, we focus on the *online* RL setting throughout this chapter, while visiting the continual learning setting in Section 2.5.

To the best of our knowledge, the first work applying DST to online RL is from [SMM+22]. They outperform dense networks with the algorithms DS-TD3 and DS-SAC, which combine sparse evolutionary training (SET) [MMS+18] with TD3 [FHM18] and SAC [HZAL18]. The methods of [SMM+22] form the foundation of our ANF algorithm.

Sokar et al. [SMM+22] reached a global sparsity level of 50%, which was later improved upon by [GEEC22, THPH23], who experimented with sparsity levels up to 99%. They showed that the sparsity level reachable without loss of performance largely depends on the environment. Graesser et al. [GEEC22] compared DST methods such as SET [MMS+18] and RigL [EGM+20] in many deep RL environments. Their performance proved to be quite similar, so we choose to use only SET.

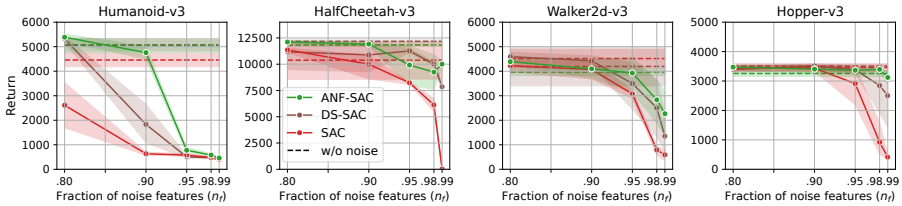


Figure 2.4. Performance of *Automatic Noise Filtering* (ANF) compared to its baselines for different noise fractions n_f. The curves show average return in the last 10% of training over 5 seeds, with shaded regions representing 95% confidence intervals. When the number of noise features in an environment increases, the performance of the standard fully-dense networks of SAC deteriorates much faster than ANF-SAC. Similar graphs for ANF-TD3 are shown in Figure A.2 of Appendix A.3.

Noise in RL. There exist different types of noise that an agent may encounter. Let us characterize the two main categories:

- Type 1: uncertainty in perception, for example when an automated vehicle cannot clearly see a traffic sign since the sun is right next to it.
- Type 2: distracting, task-irrelevant percepts, for example the bright colors of a billboard when crossing Times Square in New York City.

Type 1 noise, i.e. measurement errors, is often researched by adding noise *on top of* existing features to produce more robust agents [Bem22, VDP+20, SLZ+21, MHA+22]. This type of noise is outside the scope of this work. Instead, we focus on type 2 noise and investigate it by adding synthetic features *alongside* the existing features, creating a state space of higher dimensionality. The goal is to discover algorithms that can perform tasks well while having access to all available features, without having to pre-select the task-relevant ones. Feature selection should be carried out automatically by the RL agents.

To the best of our knowledge, the first work to make an existing RL environment noisier by adding extra features was FS-NEAT [WSS+05]. It introduced an evolutionary algorithm to select relevant features. Most of the follow-up work takes this evolutionary approach [KW09, BM13, Acu12], while we use the efficiency of deep learning, stochastic gradient descent, and dynamic sparse training.

Our work extends environments that provide the current state as a feature vector. However, it is worth mentioning that environments with visual (pixel) inputs have likewise been augmented to include a noisy challenge, such as distracting backgrounds [SRKJ21]. Other methods that have some similarities to our approach include recog-

nizing distractor signals [RAG+20], reducing state dimensionality [CBA+16, BAP+21], and identifying fake features in federated learning [LBX+21].

2.4 AUTOMATIC NOISE FILTERING

In this section, we explain how our ANF algorithm works, after which we show and interpret the results of our main experiments. ANF is a simple method that can be applied to any MLP-based deep RL algorithm. It is built upon the DS-TD3 and DS-SAC algorithms of [SMM+22], which use sparse evolutionary training (SET) from [MMS+18] as the underlying dynamic sparse training method.

In both the actor and critic networks, ANF begins by randomly pruning the input layer to the desired sparsity level s_i. During training, we drop weak connections of the input layer (weights with the smallest magnitude) after every topology-change period ΔT. After dropping a certain fraction d_f of the existing weights, ANF randomly grows the same number of connections to maintain the sparsity level s_i. By giving new connections enough time to increase their weights, ANF detects task-relevant features without explicit supervision. We provide pseudocode for ANF-SAC in Appendix A.1.

One aspect that sets ANF apart from the previous works on non-noisy settings [SMM+22, GEEC22] is that we only sparsify the input layer. This helps us to pinpoint the support of DST on our Adaptability Hypothesis. Furthermore, in extremely noisy environments it is essential to filter through the large fraction of noise. Dynamic sparse training can perform this filtering elegantly. It works well to focus the DST principle on the first layer only, as this is where the distinction between relevant and noise features is made. In Section 2.6.4 we investigate models that also have sparse hidden layers.

Another difference between ANF and DS-TD3/SAC [SMM+22] is that we mask the running averages of first and second raw moments of the gradient within the Adam optimizer [KB15] for non-existing connections. When connections are dropped and later regrown, they do not have access to previous information if implemented in a truly sparse manner. This aspect has been overlooked in the implementation of some sparsity research works that apply Adam and only simulate true sparsity with binary masks on top of the weight matrices. Our research also uses such binary masks while keeping the truly sparse implementation in mind.

Experimental setup. We integrate our ANF method in two popular deep RL algorithms: SAC and TD3. This means we compare the algorithms ANF-SAC and

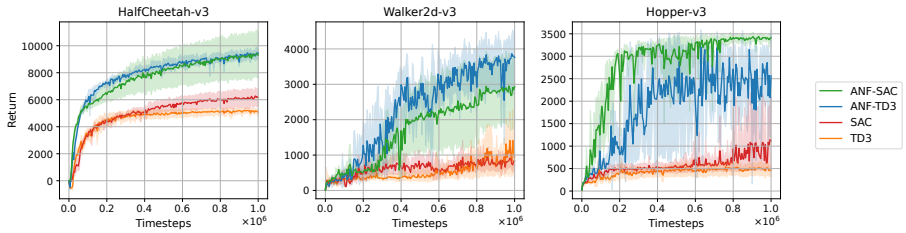


Figure 2.5. Learning curves on 3 environments with 98% noise features. This level of noise is too high for standard SAC and TD3 to learn a decent policy. ANF can still filter through the distraction and find a well-performing policy.

Table 2.1. State and action space dimensions.

Environment	State dim. *Original*	Action dim. *Original*	State dim. *ENE* ($n_f = .8$)	State dim. *ENE* ($n_f = .99$)
Humanoid-v3	376	17	1880	37600
HalfCheetah-v3	17	6	85	1700
Walker2d-v3	17	6	85	1700
Hopper-v3	11	3	55	1100

ANF-TD3 with their fully-dense counterparts as baselines. Furthermore, we compare to the closely related DS-SAC and DS-TD3, which both use their default global sparsity level of 50%. All neural networks have two hidden layers of 256 neurons with the ReLU activation function. After a hyperparameter search for ANF, we set the input layer sparsity s_i to 80%, the topology-change period $\Delta T = 1000$ timesteps, and the drop fraction $d_f = 0.05$. Further hyperparameter settings replicate prior work [SMM+22, HZAL18, FHM18]. See Appendix A.2 for additional details.

Our experiments are carried out in four continuous control environments from the MuJoCo Gym suite: Humanoid-v3, HalfCheetah-v3, Walker2d-v3, and Hopper-v3. We first run an experiment without any added noise features as a baseline and then start increasing the noise level. The fraction of noise features, n_f, ranges over the set $\{0.8, 0.9, 0.95, 0.98, 0.99\}$. Note that the state spaces of these settings increase by 5×, 10×, 20×, 50×, and 100×, respectively.

We train our agents for 1 million timesteps and evaluate them by running 10 test episodes after every 5000 timesteps. We measure the average return over the last 10% of training, as done in [GEEC22], for overview graphs such as Figure 2.4. Throughout this chapter, we run 5 random seeds for every setting. In the graphs, we show the average curve as well as a 95% confidence interval.

Results. First of all, the horizontal lines in Figure 2.4 show that even in environ-

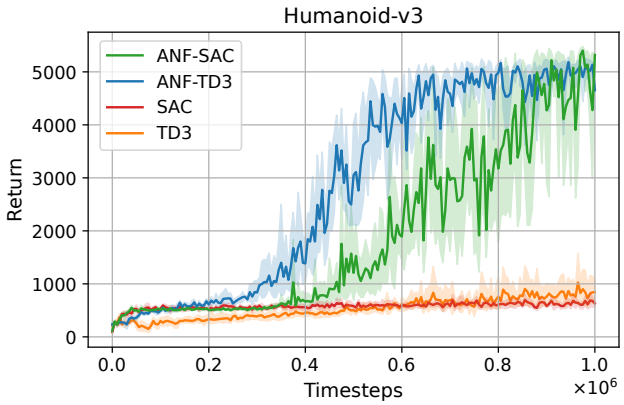


Figure 2.6. Learning curves for Humanoid-v3 with 90% noise features. While standard SAC and TD3 are too distracted by the noise to learn the task, ANF finds the task-relevant features and is able to improve.

ments without noise ANF-SAC is able to reach similar or better performance than SAC and DS-SAC for Humanoid-v3 and HalfCheetah-v3. By adjusting the connectivity of the input layer, ANF is able to select the set of most important features, the so-called *minimal task representation* [VLS22].

Furthermore, when the noise level increases our ANF method outperforms the dense baseline by a significant margin on all environments. Especially in the noisiest environments, when $n_f \geq .95$, a large gap is visible between ANF-SAC and SAC for HalfCheetah, Walker2d, and Hopper. The Humanoid environment is an exception, as ANF outperforms its baseline much earlier here but then struggles with the high noise levels as well. Table 2.1 shows that Humanoid-v3 differs noticeably from the other three environments by the size of its state space.

The learning curves in Figure 2.6 indicate that SAC and TD3 are unable to learn a decent policy within 1M timesteps in this challenging extremely noisy environment. ANF learns to ignore the distracting noise and reaches a performance level *similar even to SAC and TD3 in the environment without noise.*[4] In the environments HalfCheetah, Walker2d, and Hopper, we continue to observe this behavior up to a noise fraction of 98%, as shown in Figure 2.5. ANF outperforms its baselines by a large margin in each environment.

Topology shift. To analyze what is actually happening, we visualize the connectivity of ANF. In Figure 2.7, we present a graph that shows the development of the network's topology over time. The graph clearly demonstrates a topology shift in the input layer: on the one hand, the average number of connections to task-relevant

[4] Which is a return of ~4500, see SAC's dashed line in Figure 2.4, Humanoid-v3.

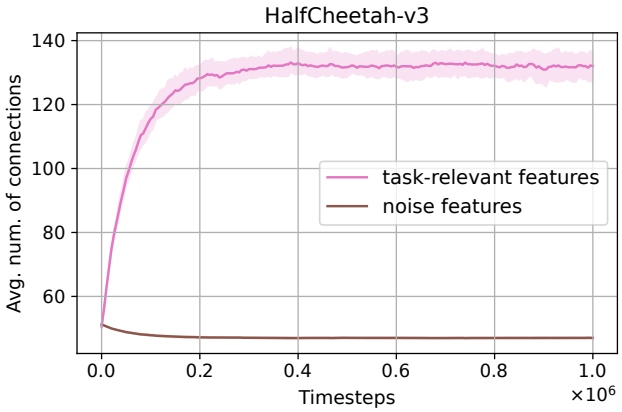


Figure 2.7. Average number of connections in the input layer of one of ANF-TD3's critic networks, on HalfCheetah-v3 with 90% noise features. At the start of training every input neuron has around $256 \cdot 0.2 \approx 51$ connections, because the input layer sparsity is 80% and connections are allocated uniformly at random. During training, ANF gradually prunes connections from the noise features and grows connections to the relevant features.

features rises, while on the other hand, noise features receive fewer weights. Together with increased performance in Figure 2.4, this supports our Adaptability Hypothesis.

2.5 CONTINUAL LEARNING

During a robot's lifetime, it may happen that other information sources become relevant to its task. Moreover, the agent may receive an entirely new task, which can require it to focus its attention on totally different state features.

We simulate this change in a *permutated extremely noisy environment* (PENE), as described in Section 2.2. The PENE rearranges all input features with a fixed permutation after every T_p timesteps. For our experiments, this means that the relevant and noise features are now mixed instead of concatenated. The agents will have to rediscover which input neurons are receiving task-relevant signals. Note that the PENE setting does *not* announce the change in environment to the agent.

Experimental setup. We set T_p to 1M timesteps, such that agents have enough time to learn. We run on the same four environments with a noise fraction of $n_f = 0.95$. In these experiments, we now train for 4 million timesteps, meaning that agents encounter four different instances (sub-environments) of feature permutations. We compare ANF-TD3 with its fully dense baseline and DS-TD3. We show 95% confidence intervals over 5 seeds.

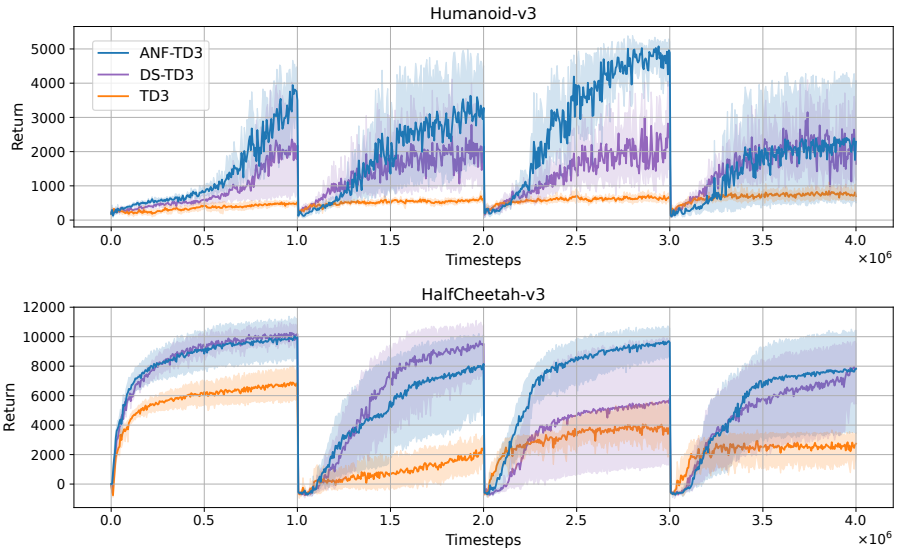


Figure 2.8. Performance of ANF-TD3 and its baselines on permuted extremely noisy environments (PENE) with 95% noise features. After every 1M timesteps, the environment's features are shuffled with a random permutation. ANF is able to cope with this challenge, while the fully dense networks of TD3 are struggling. DS-TD3 performs decently, but ANF has an advantage by focusing its sparsity on the input layer.

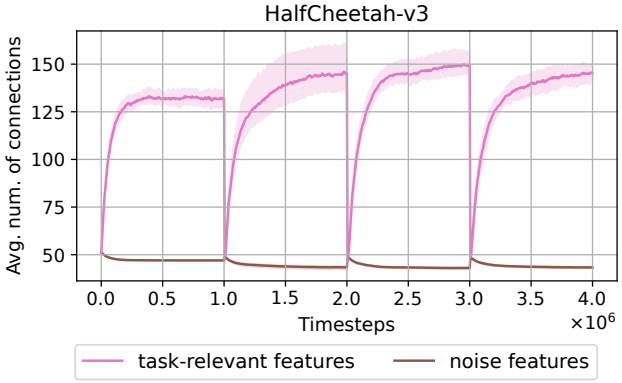


Figure 2.9. Average number of connections in the input layer of one of ANF-TD3's critic networks, on HalfCheetah-v3 with $n_f = 0.95$. Every 1M timesteps, the PENE permutes the order of the features. The ANF agent adjusts its network structure quickly, growing connections to the task-relevant features, which are now sent to different input neurons.

Results. Figure 2.8 shows the results for ANF-TD3 on Humanoid and HalfCheetah. It is evident that the performance drops considerably after each permutation of features. However, ANF is able to recover faster than the dense baselines in all environments. The method does not need to be adjusted for the challenging PENE setting; ANF keeps adapting the sparse input layer as before.

For Humanoid, some beneficial internal representations may be transferred forward, as the performance increases much earlier in the third sub-environment (between 2M and 3M timesteps) than when it is trained from scratch (between 0 and 1M timesteps). However, on the fourth sub-environment some ANF agents struggled a bit: each random seed determines not only the initialization of the agent, but also the random permutations of the environment. Thus, some sub-environments can be more challenging than others.

Maintaining plasticity. Agents that have to learn continually must be able to maintain plasticity. Standard methods are unable to do so, as shown by [DSM21]. Since the connections of the input layer can drop and grow dynamically, ANF ensures that the agent has sufficient adaptability to adjust to a new environment. We analyze this plasticity by looking into the connectivity of the input layer once more, as done earlier in Figure 2.7 for the ENE experiments.

Now, in Figure 2.9, we see that the average number of connections to task-relevant features quickly recovers after an environment change in the PENE. At every 1M steps, the PENE shuffles the features, which makes the average number of connections to relevant features drop considerably, close to the initial value. This is because many task-relevant signals are now coming in at input neurons that were previously receiving noise.

The fact that ANF has not pruned all connections to the irrelevant noise features after training on the first sub-environment is actually an advantage in this PENE setting. It means that ANF may be able to reach a high number of connections to new task-relevant features faster, as they already have some 'spare' connections waiting.

2.6 ANALYSIS

In this section, we further analyze the robustness of ANF to louder (Section 2.6.1) or realistic (2.6.2) noise. We also study the benefit of the dynamic approach (2.6.3) and investigate what sparsity levels the full networks can reach before significantly degrading in performance (2.6.4).

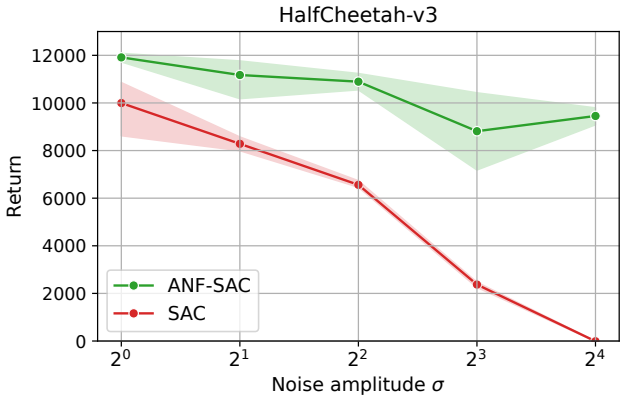


Figure 2.10. ANF-SAC and its dense baseline on ENEs with louder noise. The noise features are sampled from $\mathcal{N}(0, \sigma^2)$. Noise amplitude σ is increased exponentially; notice the log-scale on the horizontal axis. ANF tolerates louder noise much better than standard SAC, maintaining a high performance up to $\sigma = 16$. In this experiment $n_f = 0.9$.

2.6.1 Louder noise

All of our experiments so far have been executed with noise features sampled from the standard Gaussian distribution of $\mathcal{N}(0, 1)$. But what would happen if we increase the standard deviation, i.e. the noise amplitude? We expect the louder noise to be more distracting, increasing the difficulty of the ENE. We hope to discover whether ANF can cope with this additional challenge.

Experimental setup. We run ANF and its dense baselines on the ENE of HalfCheetah-v3, but the noise is now sampled from $\mathcal{N}(0, \sigma^2)$. We let the standard deviation σ increase exponentially, ranging over the set $\{1, 2, 4, 8, 16\}$. The ENE contains 90% of these louder noise features ($n_f = 0.9$).

Results. From the experimental results, we can conclude that louder noise does make the ENE more challenging. In Figure 2.10, it is clearly visible that as the noise amplitude increases, the performance decreases. Fortunately for ANF, this decrease is much less pronounced compared to its dense baseline. In fact, the final return of ANF-SAC on an ENE with noise amplitude $\sigma = 16$ is almost the same as SAC's performance on the standard $\sigma = 1$ environment. ANF can cope well with even the loudest noise.[5]

[5] See https://youtu.be/vS47UnsTQk8 for a video comparing the actual motion of HalfCheetah when controlled by ANF-SAC vs. SAC with different noise amplitudes.

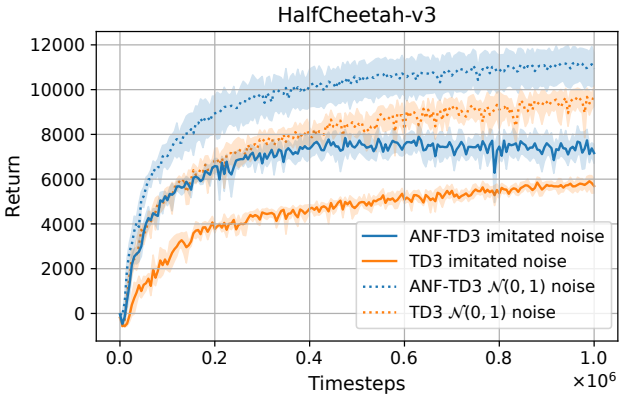


Figure 2.11. Learning curves for ANF-TD3 and its dense baseline on the challenging ENE, where the noise features imitate the task-relevant features. This increases the difficulty, but ANF still achieves the highest return. The ENE has 90% of these realistic noise features in this experiment.

2.6.2 Imitating real features

Upon closer inspection of the data distribution of the original state features, we discovered that these are far from a standard Gaussian distribution. To better reflect real-world noise, we want the noise features of our ENEs to mimic the original features. We expect that this increases the difficulty of our extremely noisy environments, as the noise is now much more similar to the task-relevant features.

Experimental setup. For each of the original features, we make a histogram of its final distribution (after training an agent in the standard noise-free environment). In the experiments of this section, the ENE samples from these histograms[6] to generate noise features for the next state. We repeatedly sample from the distribution of each original feature in sequence until we have enough noise features. We run this experiment on HalfCheetah-v3, with 90% of these noise features that imitate the task-relevant features.

Results. In Figure 2.11, we see that the imitated noise indeed raises the difficulty of the ENE. The performance of both ANF-TD3 and TD3 decreases considerably compared to the standard $\mathcal{N}(0, 1)$ noise. However, ANF is still able to outperform its dense baseline by a large margin, even in this challenging ENE.

[6] By first sampling a bin according to the histogram's probability mass function, and then sampling a value uniformly at random within the chosen bin.

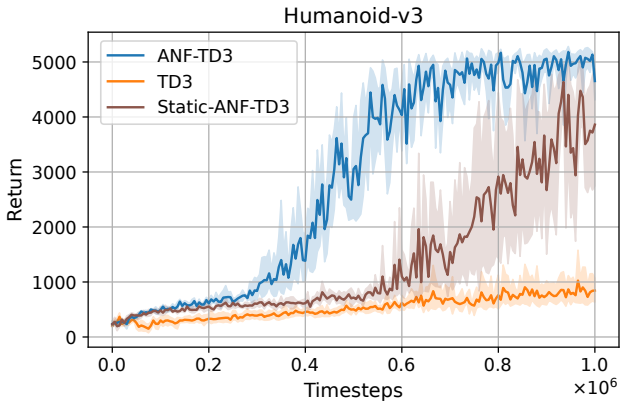


Figure 2.12. Comparison of ANF to its static sparse counterpart and standard (fully dense) TD3. ANF-TD3 dynamically updates the sparse network topology, while the topology of the other two methods remains static. The learning curves show that the dynamic updates are an essential part of ANF. This experiment is run with 90% noise features.

2.6.3 Does ANF need to be dynamic?

In this section, we perform an ablation study to show that the dynamic network topology updates of ANF are a necessary component of the algorithm. Removing these dynamic updates during training would give a static sparse training algorithm. This algorithm starts with a randomly sparsified input layer just like ANF, but it does not drop or regrow any connections during training.

Experimental setup. We compare ANF to its fixed-connectivity counterpart that uses static sparse training, which we call *Static-ANF*. In addition, we compare both with the standard dense TD3 algorithm. We run on Humanoid-v3 with 90% noise features added to the environment.

Results. The graphs in Figure 2.12 show that ANF indeed needs to be dynamic, as it significantly outperforms its static version. Intuitively, this is consistent with the concept shown in Figure 2.7, where ANF changes its connectivity to emphasize its focus on the task-relevant features. This emphasis is lost if one removes ANF's ability to dynamically adjust the network structure.

Nevertheless, it is remarkable to see that Static-ANF is able to surpass standard dense TD3 in this setting. It seems that just having fewer connections to the 90% noisy input features already helps to lower the overall distraction.

Table 2.2. Comparison of different algorithms, along with their parameter counts for the actor network. The critic networks have comparable numbers of parameters.

Algorithm	Environment	Return (↑)	# Params. (↓)
ANF-TD3	Humanoid	**4968.3**	262,400
TD3	Humanoid	817.3	1,032,448
Sparser(80%)-ANF-TD3	Humanoid	4963.1	206,489
Sparser(95%)-ANF-TD3	Humanoid	4806.4	**51,622**
ANF-TD3	HalfCheetah	**11086.4**	75,776
TD3	HalfCheetah	9452.6	110,592
Sparser(80%)-ANF-TD3	HalfCheetah	10640.3	22,118
Sparser(95%)-ANF-TD3	HalfCheetah	8357.9	**5,529**

2.6.4 How Sparse can we go?

We already showed that sparsifying the input layer can significantly improve performance in extremely noisy environments. In this section, we investigate whether we can further sparsify the agent's networks by also pruning connections in other layers. This would reduce the network size (total number of parameters) even further, while hopefully maintaining performance.

Experimental setup. Instead of only having an 80% sparse *input* layer, the networks now also have a sparse *hidden* layer for which the connectivity is frequently adjusted with DST. We keep the output layer dense, just as in [SMM+22]. The sparsity distribution is uniform, meaning that both the input layer and the hidden layer have the same sparsity level. We compute the required layer sparsity levels such that the global sparsity level (over the full network) is at s, where s ranges over $\{.80, .90, .95, .98\}$. We compare with our standard ANF algorithm, which has a global sparsity of 74.6% for $n_f = 0.9$ (actor network on Humanoid). We run on Humanoid-v3 and HalfCheetah-v3, as the achievable sparsity level before performance degradation differs significantly between these two environments.

Results. We see in Figure 2.13 that on extremely noisy environments, nearly the same performance can be reached with further sparsified networks, meaning that a large proportion of the parameters can be pruned. In Figure 2.13 (left), we see that ANF-TD3 with a global sparsity of 80% still surpasses standard dense TD3 in final return on HalfCheetah-v3. As the global sparsity increases, the performance gradually decreases. This is quite different for Humanoid-v3, in Figure 2.13 (right). Notice that ANF-TD3 can go up to a global sparsity level of 95%, with barely any performance degradation. This means it can use 20× fewer parameters than standard TD3, reducing the network size considerably, as shown in Table 2.2.

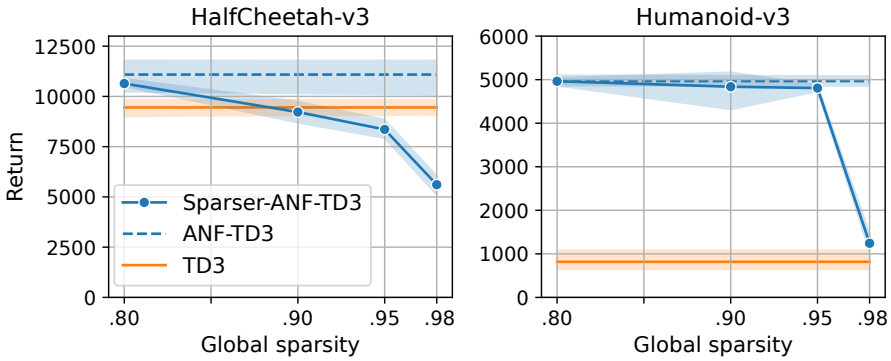


Figure 2.13. Performance of ANF-TD3 and its sparser versions, on 90% noise features. Sparser-ANF also prunes weights in the hidden layer, instead of only the input layer. Further sparsifying the network does not improve performance, but can drastically reduce the size of the network.

2.7 CONCLUSION

In this work, we formulated the problem setting of extremely noisy environments and showed that our Automatic Noise Filtering algorithm succeeds at this challenge, where standard deep RL methods struggle. By using dynamic sparse training, the ANF algorithm adapts its internal network topology to focus on task-relevant features.

Our experiments provide an initial empirical verification of our Adaptability Hypothesis, which roughly states that improving performance by dropping and growing sparse connections is easier than by modifying dense weights. Further research is necessary to provide more conclusive evidence, as we exclusively studied SAC and TD3. Integrating ANF with other deep RL methods is open for future work. The input-layer sparsity is an important hyperparameter for ANF; treating it as a learnable parameter could be beneficial.

This chapter is limited to continuous-control tasks with feature vectors as inputs. When expanding to robotic settings with noisy image or video data, other methods may be required. We will discuss one such approach in the next chapter.

CHAPTER 3

Adapting to Visual Distractions

In Chapter 2, we have explored agents that are able to adapt well in extremely noisy environments. The inputs that the agents received were all vector-based. In this chapter, we will aim to generalize the approach to vision, i.e., agents with pixel inputs. The approach is quite different here, since—in contrast to the previous chapter—each pixel can be task-relevant in one timestep, but irrelevant shortly after.

Many input pixels received by RL agents often contain distracting stimuli. Autonomous agents need the ability to distinguish useful information from task-irrelevant perceptions, enabling them to generalize to unseen environments with new distractions. Existing approaches to this problem use data augmentation or large auxiliary networks with additional loss functions.

We introduce *MaDi*, a novel algorithm that learns to **ma**sk **di**stractions by the reward signal only. In MaDi, the conventional actor-critic structure of deep reinforcement learning agents is complemented by a small third sibling, the Masker. This lightweight neural network generates a mask to determine what the actor and critic receive, allowing them to focus on learning the task. The masks are created dynamically, depending on the current input.

We run experiments on the DeepMind Control Generalization Benchmark, the Distracting Control Suite, and a real UR5 Robotic Arm. Our algorithm improves the agent's focus with useful masks, while its efficient Masker network only adds 0.2% more parameters to the original structure, in contrast to previous work. MaDi consistently achieves generalization results better than or competitive to state-of-the-art methods.

This chapter was adapted from: Bram Grooten, Tristan Tomilin, Gautham Vasan, Matthew E Taylor, A Rupam Mahmood, Meng Fang, Mykola Pechenizkiy, and Decebal Constantin Mocanu. MaDi: Learning to Mask Distractions for Generalization in Visual Deep Reinforcement Learning. *Int. Conf. Autonomous Agents and Multiagent Systems (AAMAS)*, 2024.

3.1 INTRODUCTION

Deep reinforcement learning (RL) has achieved remarkable success in a variety of complex tasks such as game playing [SAH+20, MKS+15], robotics [AAC+19, HML+23, Col23], nuclear fusion [DFB+22], and autonomous navigation [MPV+16, ZMK+17]. However, one of the major challenges faced by RL agents is their limited ability to

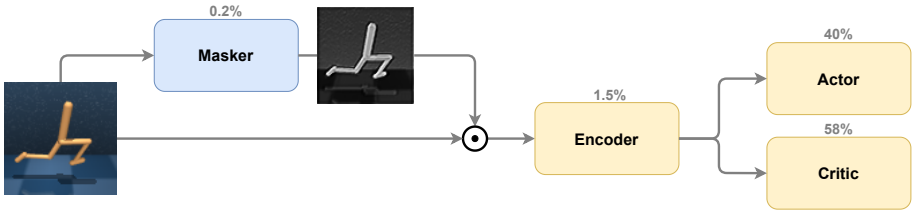


Figure 3.1. In the MaDi architecture, the Masker produces a soft mask (values between 0 and 1) for each frame, which subsequently gets multiplied element-wise with the observation. The encoder is a shared ConvNet updated only by the critic loss, which is also the objective function for the Masker. We show rounded percentages for the number of parameters used in the `walker-walk` environment. The actor and critic contain most of it ($\sim$ 98% together) as they consist of multiple fully-connected layers.

generalize to unseen environments, particularly in the presence of distracting visual noise, such as a video playing in the background [SRKJ21, HW21]. These distractions can lead to significant degradation in the performance of deep RL agents, thereby hindering their applicability in the real world. To address this, we propose a novel algorithm, **Ma**sking **Di**stractions, which learns to filter out task-irrelevant visuals, enhancing generalization capabilities.

The key idea behind MaDi is to supplement the conventional actor-critic architecture with a third lightweight component, the Masker (see Figure 3.1). This small neural network generates a mask that dims the irrelevant pixels, allowing the actor and critic to focus on learning the task at hand without getting too distracted. Unlike previous approaches that have attempted to address this issue [HW21, HSW21, BZZR22], our method increases generalization performance while introducing minimal overhead in terms of model parameters, thus preserving the efficiency of the original architecture.

Furthermore, no additional loss function is necessary for the Masker to optimize its parameters. To ensure that the Masker maintains visibility of the task-relevant pixels, it is trained on the critic's loss function. The Masker and critic networks are aligned in their objective, as pixels that are essential to determine the value of an observation should not be hidden. Figure 3.1 shows an example of a mask, visualizing the output produced by the Masker network corresponding to the current input frame. The Masker is able to learn such precise segmentations without any additional labels, bounding boxes, or other annotations. The reward alone is enough.

To evaluate the effectiveness of MaDi, we conduct experiments on multiple environments from three benchmarks: the DeepMind Control Generalization Benchmark [HW21], the Distracting Control Suite [SRKJ21], and a real UR5 Robotic Arm for

which we design a novel generalization experiment with visual distractions. Our results demonstrate that MaDi significantly improves the agent's ability to focus on relevant visual information by generating helpful masks, leading to enhanced generalization performance. Furthermore, MaDi achieves state-of-the-art performance on many environments, surpassing well-known methods in vision-based reinforcement learning [HZAL18, LLS+20, YKF21, HW21, HSW21, BZZR22].

Our main contributions are:

- We introduce a novel algorithm, MaDi, which supplements the standard actor-critic architecture of deep RL agents with a lightweight Masker. This network learns to focus on the task-relevant pixels solely from the reward signal.
- We present a comprehensive set of experiments on the DeepMind Control Generalization Benchmark and the Distracting Control Suite. MaDi consistently achieves state-of-the-art or competitive generalization performance.
- We test MaDi on a physical robot, demonstrating that our algorithm increases the performance of the UR5 Robotic Arm in a challenging VisualReacher task, even when distracting videos are playing in the background.

3.2 RELATED WORK

The problem of generalization in deep reinforcement learning has been an active area of research, with several approaches proposed to tackle the challenge of visual distractions. In this section, we review the most relevant literature, highlighting the differences between our proposed MaDi method and existing approaches.

Generalization in RL In reinforcement learning, generalization refers to an agent's ability to perform well on unseen environments or tasks [KZGR23]. This can be challenging, as RL is prone to overfit to the training environment [ZVMB18, FMB18, CKH+19, HJS+20]. Several works have focused on improving generalization capabilities by employing techniques such as domain adaptation [XNC+21], domain randomization [TFR+17, AAC+19], meta-learning [WKNT+16, DSC+16], contrastive learning [LSA20, AMCB20], imitation learning [FWH+21], bisimulation metrics [FPP11, ZMC+21], and data augmentation [HW21, YKF21, LLS+20, YMM+22, RGY+21, WKSF20]. Even using a ResNet [HZRS16] pretrained on ImageNet [DDS+09] as an encoder can improve generalization [YXY+22].

Visual Learning in RL Learning tasks from visual input, i.e., image-based or vision-based deep RL, is typically more demanding than learning from direct features in a vector. DQN [MKS+15] was the first to learn Atari games at human-level performance directly from pixels. However, it has been shown that these algorithms can be quite brittle to changes in the environment, as altering a few pixel values can significantly decrease DQN's performance [QSO+20, ZCX+20]. Using data augmentation proved to be the key in visual RL. DrQ [YKF21] and RAD [LLS+20] use light augmentations such as random shifts or crops of the observation to increase the algorithm's robustness.

Distractions in RL Several approaches have been proposed to deal with the presence of task-irrelevant noise and distractions in reinforcement learning environments. Automatic Noise Filtering (ANF) [GSD+23] works on noisy environments that provide states as feature vectors, such as the MuJoCo Gym suite [TET12, BCP+16]. We focus our work on RL agents that need to learn from image-based observations, like the pixel-wrapped DeepMind Control Suite [TDM+18]. Two benchmarks that we use are extensions of this suite.

Several works [HW21, HSW21, BZZR22, YMM+22, YXY+22] have tried to tackle the DeepMind Control Generalization Benchmark [HW21] and the Distracting Control Suite [SRKJ21]. Many of these methods use stronger[1] data augmentations than the light shifting and cropping of DrQ and RAD. Usually, they apply one of two favored augmentation techniques: a randomly initialized convolution layer (`conv` augmentation) or overlaying the observation with random images from a large dataset, such as Places365 [ZLK+17] (`overlay` augmentation).

Masking in Visual RL There exist a few works that aim to improve the generalization ability of RL agents by masking parts of the input. Yu et al. [YZL+22] randomly mask parts of the inputs and use an auxiliary loss to reconstruct these pixels. SGQN [BZZR22] and the recent InfoGating [TILB23] apply more targeted masking similar to MaDi. InfoGating has only experimented with offline RL, and it uses a large U-Net [RFB15] to determine the appropriate masks, while MaDi uses a much smaller 3-layer convolutional neural network.

Baselines We select the following set of six baselines for our experiments, as these focus on online RL and do not use any pretrained models:

[1] By *stronger*, we mean augmentations that alter an image significantly more.

- *Soft Actor-Critic* [HZAL18, **SAC**] is an off-policy actor-critic algorithm that optimizes the trade-off between exploration and exploitation by automatically tuning a temperature parameter for entropy regularization.
- *Data-regularized Q-learning* [YKF21, **DrQ**] focuses on making Q-learning more stable and sample efficient by shifting the observations by a few pixels in a random direction.
- *RL with Augmented Data* [LLS+20, **RAD**] improves the data efficiency in visual RL by randomly cropping the images.
- *Soft Data Augmentation* [HW21, **SODA**] applies data augmentation in an auxiliary task that tries to minimize the distance of augmented and non-augmented images in its feature space.
- *Stabilized Value Estimation under Augmentation* [HSW21, **SVEA**] stabilizes learning by using augmentation solely in the critic. It combines clean and augmented data in every batch used for a critic update. The actor only sees clean data.
- *Saliency Guided Q-Networks* [BZZR22, **SGQN**] is perhaps closest to our work, as it also uses masks to benefit learning. Its masks are not applied at the start of the architecture, but are learned by a third component after the encoder. This auxiliary model minimizes the difference between its masks and other masks generated by a saliency metric. By computing the gradient of the Q-function with respect to the input pixels, this saliency metric determines which pixels are important for the agent. A hyperparameter `sgqn_quantile` (often set to $95\% - 98\%$) determines how many pixels are masked.

There are multiple significant differences between SGQN and MaDi. First of all, SGQN can be quite sensitive to the quantile hyperparameter. MaDi is free from this hyperparameter tuning, as it automatically finds the right fraction of pixels to mask. Furthermore, SGQN needs to compute gradients with respect to the inputs and weights, while MaDi only requires gradients of the weights. The additional components of SGQN are heavier, as they add about 1.6M parameters (an extra 25%) to the base architecture, with MaDi just adding roughly 10K (0.2%), reducing the memory requirements. MaDi also does not introduce any additional auxiliary loss function, as it is able to learn directly from the critic's objective. In essence, SGQN does not apply a mask to every input image, but uses them to learn better representations.

MaDi tries to learn the most helpful masks such that the actor and critic receive only task-relevant information and are able to focus on the RL problem.

3.3 BACKGROUND

Problem formulation. We consider the problem of learning a policy for a Markov decision process (MDP) with the presence of visual distractions, similar to the formulation by Hansen et al. [HSW21]. Our approach, MaDi, aims to learn a policy that generalizes well across MDPs with varying state spaces.

We formulate the interaction between the environment and policy as an MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the state transition function, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and γ is the discount factor. To address the challenges of partial observability [KLC98], we define a state $\mathbf{s}_t$ as a sequence of k consecutive frames $(\mathbf{o}_t, \mathbf{o}_{t-1}, \dots, \mathbf{o}_{t-(k-1)})$, $\mathbf{o}_i \in \mathcal{O}$, where $\mathcal{O}$ is the high-dimensional image space. In the particular benchmarks we employ for evaluation, $\mathcal{O} = \mathbb{R}^{84\times84\times3}$ for the simulation environments and $\mathcal{O} = \mathbb{R}^{160\times90\times3}$ for the robotic environment, as we receive RGB colored images as input with 84×84 and 160×90 pixels respectively.

Our goal is to learn a stochastic policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, where $\Delta(\mathcal{A})$ denotes the space of probability distributions over the action space $\mathcal{A}$. This policy aims to maximize the discounted return $R_t = \mathbb{E}_{\Gamma\sim\pi,\mathcal{P}}\left[\sum_{t=0}^{T} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)\right]$ along a trajectory $\Gamma = (\mathbf{s}_0, \mathbf{s}_1, \dots, \mathbf{s}_T)$. The policy π is parameterized by a collection of learnable parameters θ. We aim to learn parameters θ such that π_θ generalizes well across MDPs with perturbed observation spaces, denoted as $\overline{\mathcal{M}} = \langle \overline{\mathcal{S}}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle$, where states $\overline{\mathbf{s}}_t \in \overline{\mathcal{S}}$ are constructed from observations $\overline{\mathbf{o}}_t \in \overline{\mathcal{O}}$. The original observation space $\mathcal{O}$ is a subset of the perturbed observation space $\overline{\mathcal{O}}$, which may contain distractions.

Distractions. We define a distraction to be any input feature that is irrelevant to the task of the MDP, meaning that an optimal policy π^* and value function Q^* remain invariant under alterations of the feature value. In our case, input features are pixels $p_i \in \mathbb{R}^3$, where a state $\boldsymbol{s}$ consists of n pixels: $\boldsymbol{s} = (p_1, p_2, \dots, p_n)$.

Suppose we encounter a state $\hat{\boldsymbol{s}}$, and we wish to determine whether pixel p_i is a distraction in that particular state. Let $\mathcal{S}_i(\hat{\boldsymbol{s}})$ be the set of states where only pixel p_i is changed in comparison to $\hat{\boldsymbol{s}}$. Then pixel p_i is considered a distraction in state $\hat{\boldsymbol{s}}$ if π^* and Q^* remain invariant across the entire set $\mathcal{S}_i(\hat{\boldsymbol{s}})$. More formally:

Definition 3.3.1. *A pixel p_i is a **distraction** in state $\hat{\boldsymbol{s}}$ if, for an optimal policy π^* and value function Q^* it holds that, for an arbitrary but fixed action $\boldsymbol{a}$, we have $\forall\, \boldsymbol{s} \in \mathcal{S}_i(\hat{\boldsymbol{s}})$:*

$$\pi^*(\boldsymbol{a}|\boldsymbol{s}) = \rho^* \qquad \text{where probability } \rho^* \in \mathbb{R} \text{ is constant,}$$
$$Q^*(\boldsymbol{s}, \boldsymbol{a}) = q^* \qquad \text{where value } q^* \in \mathbb{R} \text{ is constant.}$$

In other words: pixel p_i can take on any value, but the optimal policy will not change. In that particular state $\hat{\boldsymbol{s}}$, the pixel p_i is irrelevant to the task and thus a distraction. From this definition we can derive that the partial derivative of π^* and Q^* with respect to the input feature p_i is zero.

Corollary 3.3.2. *If p_i is a distraction in state $\hat{\boldsymbol{s}}$, then for an arbitrary action $\boldsymbol{a}$ we have that*

$$\frac{\partial}{\partial p_i}\pi^*(\boldsymbol{a}|\hat{\boldsymbol{s}}) = 0 \qquad \text{and} \qquad \frac{\partial}{\partial p_i}Q^*(\hat{\boldsymbol{s}}, \boldsymbol{a}) = 0.$$

This follows from Definition 3.3.1 since π^* and Q^* remain constant for varying p_i. Optimal policies perfectly ignore distractions, while suboptimal policies (i.e., neural networks during training) may be hindered by distractions. As distractions have no effect on the optimal policy, they can be safely masked when using π^*. This suggests that when striving to approximate π^*, it may be advantageous to mask distractions as well, a concept at the core of MaDi.

Soft Actor-Critic. In this work, we build upon the model-free off-policy reinforcement learning algorithm Soft Actor-Critic (SAC; [HZAL18]). SAC aims to estimate the optimal state-action value function Q^* with its parameterized critic Q_{θ_Q}. The actor is represented by a stochastic policy π_{θ_π}, which aims to maximize the value outputted by the critic while simultaneously maintaining high entropy. The optional shared encoder f_{θ_f} is often used for SAC in image-based environments. The critic and shared encoder have target networks that start with the same parameters $\theta^{\text{tgt}} = \theta$. These are gradually updated throughout training by an exponential moving average: $\theta^{\text{tgt}} \longleftarrow (1-\tau)\theta^{\text{tgt}} + \tau\theta$. We will often omit the implied parameter θ_N in our notation of any network N.

3.4 MASKING DISTRACTIONS

MaDi aims to mask distractions that hinder the agent from learning and performing well. We supplement the conventional actor-critic architecture of deep reinforcement

Algorithm 1 MaDi based on SAC

randomly initialize all networks: π, Q, f, M
copy parameters to target networks: Q^{tgt}, f^{tgt}
1: **for** timestep $t = 1...T$ **do**
act:
2: $\mathbf{a}_t \sim \pi\left(\cdot | f(\mathbf{s}_t \odot M(\mathbf{s}_t))\right)$ — Sample action
3: $\mathbf{s}'_t, r_t \sim \mathcal{P}(\cdot | \mathbf{s}_t, \mathbf{a}_t)$ — Perform action in env
4: $\mathcal{B} \leftarrow \mathcal{B} \cup (\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}'_t)$ — Add to replay buffer
update:
5: $\{\mathbf{s}_b, \mathbf{a}_b, r_b, \mathbf{s}'_b\} \sim \mathcal{B}$ — Sample batch $b \subset \mathcal{B}$
6: $\theta_\pi \leftarrow \theta_\pi - \eta \nabla_{\theta_\pi} \mathcal{L}_\pi(\mathbf{s}_b)$ — Update π
7: $\mathbf{s}_b \leftarrow \text{concat}(\mathbf{s}_b, \delta(\mathbf{s}_b))$ — Apply augmentation
8: **for** network N in $[Q, f, M]$ **do**
9: $\theta_N \leftarrow \theta_N - \eta \nabla_{\theta_N} \mathcal{L}_Q\left(\mathbf{s}_b, \mathbf{a}_b, r_b, \mathbf{s}'_b\right)$ — Update Q, f, M
10: **for** network N in $[Q, f]$ **do**
11: $\theta_N^{\text{tgt}} \leftarrow (1-\tau)\theta_N^{\text{tgt}} + \tau\theta_N$ — Update Q^{tgt}, f^{tgt}

learning agents by integrating a third, lightweight component, the Masker network M. The Masker adjusts the input by dimming irrelevant pixels, allowing the actor and critic networks to focus on learning the task at hand. The Masker and encoder compute internal representations using the Hadamard product and one forward call each: $f(\mathbf{s}_t \odot M(\mathbf{s}_t))$. Algorithm 1 indicates the few adjustments necessary to standard SAC (or SVEA, when using the augmentation on line 7). Note that MaDi does not need a target network for the Masker, reducing the additional number of parameters required.

3.4.1 The MaDi architecture

As shown in Figure 3.1, the Masker network is placed at the front of the agent's architecture. It produces a scalar multiplier for each pixel in the input image to determine the degree to which the pixel ought to be darkened. We refer to the output of this network as a *soft* mask.[2] These soft masks are applied element-wise to the observation frames, effectively reducing distractions (see Figure 3.2).

The Masker network is composed of three convolutional layers with ReLU non-linearities in between. The last layer outputs one channel with a Sigmoid activation function to squeeze values into the interval $[0, 1]$. The Masker receives three input channels, representing the RGB values of one frame $\boldsymbol{o}_t$. In Section 3.3, we defined

[2] We also tried *hard* (i.e., *binary*) masks, but they proved more challenging to train.

a full state $\boldsymbol{s}_t$ to be a stack of k frames, which is indeed what the actor and critic receive as input. The Masker is the only network that processes each frame separately. However, we still require only one forward pass through the Masker network for each input $\boldsymbol{s}_t$ of k frames, as we efficiently reshape the channels into the batch dimension. See Appendix B.1 for further implementation details.

3.4.2 How does the Masker learn?

One may expect that learning to output useful masks requires us to define a separate loss function, but this is not the case. The Masker can simply be updated via the critic's objective function[3] to update its parameters, as shown on line 9 of Algorithm 1. This means the masks are trained without any additional segmentation labels or saliency metrics. Our hypothesis on the surprising ability of MaDi to determine the task-relevant pixels solely from a scalar reward signal, pertains to the following:

- For *relevant* pixels, if the Masker network masks away essential pixels needed to determine an accurate Q-value, then the critic loss will presumably be high, and the Masker will thus be encouraged to leave these pixels visible.
- For *irrelevant* pixels, we believe (and empirically show in Section 3.5.3) that strong and varying data augmentation helps. It gives an irrelevant pixel in a particular state $\boldsymbol{s}$ a varying pixel-value each time state $\boldsymbol{s}$ is sampled from the replay buffer, while the pixel's contribution to the Q^*-value remains the same (none, because it is irrelevant). The Masker is thus incentivized to mask this pixel, such that the actor and critic networks always see the same pixel-value for state $\boldsymbol{s}$, no matter which augmentation is used.

The Masker is updated together with the critic, which happens once for every environment step in case of synchronous runs. The robotic experiments use an asynchronous version of each algorithm. In that case, the Masker still gets as many updates as the critic, but it is no longer equal to the number of environment steps.

[3] Future work could study whether the Masker network can also learn from the actor loss. In many SAC-based implementations with a shared ConvNet, the encoder is only updated by the critic loss, and it made sense to use these gradients for the Masker.

(a) `training_env (walker)` **(b)** `video_hard (ball_in_cup)` **(c)** `distracting_cs (cartpole)`

Figure 3.2. Examples of original observations (left) and their masked versions (right) generated by MaDi in training (a) and testing (b, c) environments. Masks from other benchmark and domain combinations are shown in Appendix B.3.

3.5 SIMULATION EXPERIMENTS

We present the experiments done on generalization benchmarks based on the DeepMind Control Suite [TDM+18] in this section, while our robotic experiments are shown in Section 3.6. We describe our experimental setup and provide the results obtained from our method, MaDi, compared with state-of-the-art approaches. Our experiments are designed to demonstrate the effectiveness of MaDi in masking distractions and improving generalization in vision-based RL.

3.5.1 Experimental Setup

Benchmarks. We evaluate the performance of MaDi on the DeepMind Control Generalization Benchmark [HW21, DMControl-GB] and the Distracting Control Suite [SRKJ21, DistractingCS]. These benchmarks consist of a range of environments with varying levels of complexity and noise, providing a comprehensive assessment of an agent's ability to generalize to unseen, distracting environments.

- **DMControl-GB** has two setups with task-irrelevant pixels in the background: `video_easy` and `video_hard`. For the easy setup, there are just 10 videos to randomly sample from, and the surface from the training environment is still shown. In the hard counterpart, the surface is no longer visible, and one of 100 videos is selected. Note that all the frames in these videos are unseen — they do not overlap with the images in the augmentation dataset we use.

- **DistractingCS** goes a step further, also adjusting the camera's orientation and the agent's color during an episode, while displaying a randomly selected video in the background. The surface remains visible, such that the agent can orient itself during a changing camera angle. The intensity of the DistractingCS determines the difficulty. With higher intensity, the environment (1) samples

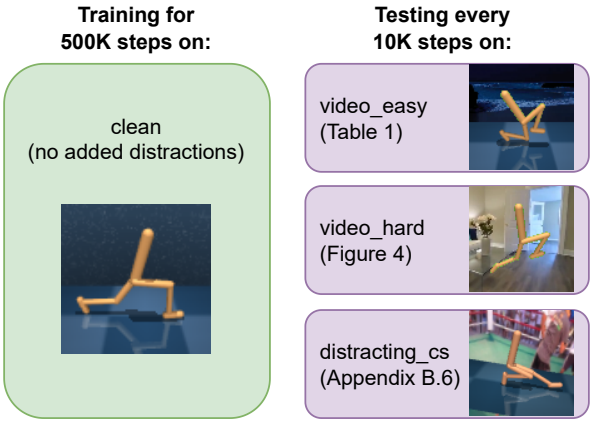


Figure 3.3. The training and testing setup.

from a larger set of videos, (2) changes the agent's color faster and to more extreme limits, and (3) adjusts the camera's orientation faster and to more severe angles. We use the default intensity level of 0.1.

Environments. Within these two benchmarks, we run all algorithms on six distinct environments, listed in Table 3.1. From the `cartpole` and `walker` domains we select two tasks, which differ by their starting positions and reward functions.

Models & Training. For a fair comparison, we use the same base actor-critic architecture for all the methods considered in this study, including MaDi. All algorithms are trained for 500K timesteps on the clean training environment without distractions. We use the default hyperparameters for all baselines, as specified by the DMControl-GB [HW21]. See Appendix B.1 for an overview of the hyperparameters.

Augmentation. As discussed in Section 3.2, all of our baselines (except SAC) use some form of data augmentation. MaDi is built on top of SVEA [HSW21], which performs best with the `overlay` augmentation for distracting video backgrounds. Therefore, we choose to apply `overlay` for MaDi as well. This strong augmentation combines an observation frame from the training environment, $\mathbf{o}_t$, with a random image $\boldsymbol{x}$ from a large dataset as follows:

$$\delta_{\boldsymbol{x}}(\mathbf{o}_t) = \alpha \cdot \mathbf{o}_t + (1 - \alpha) \cdot \boldsymbol{x}$$

Table 3.1. Generalization of MaDi and various baseline algorithms on six different environments trained for $500K$ steps. We show return on `video_easy` with mean and standard error over five seeds. MaDi outperforms or comes close to the state-of-the-art in all environments.

`video_easy`	SAC	DrQ	RAD	SODA	SVEA	SGQN	MaDi
`ball_in_cup catch`	602 ±91	714 ±131	561 ±147	750 ±98	757 ±138	761 ±171	**807 ±144**
`cartpole balance`	924 ±19	932 ±33	801 ±95	961 ±10	967 ±2	965 ±5	**982 ±4**
`cartpole swingup`	782 ±21	613 ±74	658 ±17	215 ±125	786 ±15	798 ±13	**848 ±6**
`finger spin`	227 ±26	543 ±50	479 ±65	429 ±100	645 ±39	592 ±11	**679 ±17**
`walker stand`	507 ±113	954 ±10	961 ±1	147 ±17	**977 ±3**	672 ±153	967 ±3
`walker walk`	334 ±37	821 ±38	726 ±42	479 ±168	**936 ±14**	882 ±26	895 ±24
`avg`	563	763	698	497	845	778	**863**

where δ denotes the augmentation function. We use a default overlay factor of $\alpha = 0.5$ and sample images from the same dataset as used in SVEA; Places365 [ZLK+17].

Evaluation. To assess generalization, we evaluate the trained agents zero-shot on a set of unseen environments with different levels of distractions. Specifically, we test on `video_easy` and `video_hard` from DMControl-GB, and on the Distracting Control Suite. Every 10K steps we evaluate the current policy for 20 episodes on the test environments; see Figure 3.3. We report the average undiscounted return over five random seeds during the last 10% of training, a metric often used to reduce variance [GSD+23, GEEC22]. We run statistical tests to verify significance in Appendix B.2.

3.5.2 Generalization Results

In Table 3.1 we show the results of MaDi and its baselines when generalizing to the `video_easy` setup of the DMControl-GB benchmark. MaDi is able to achieve the best or competitive performance in all environments. The learning curves of Figure 3.4 show that MaDi also generalizes well to the more challenging `video_hard` environments. Furthermore, the curves show that MaDi has a high sample efficiency, often reaching adequate performance in just 100K environment steps, by its ability to focus on the task-relevant pixels.

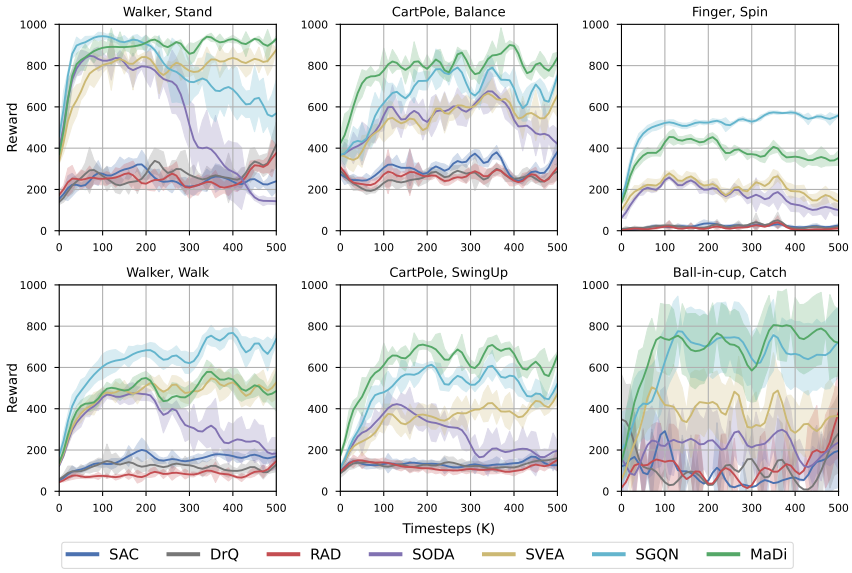


Figure 3.4. Learning curves of MaDi and six baselines on `video_hard`. Agents are trained on clean data for $500K$ steps and tested on `video_hard` every $10K$ steps. MaDi often reaches the top of the class, whereas some baselines can overfit to the training environment and lose generalizability. The curves show the mean across five seeds, with the standard error shaded.

We present tables with results on the DistractingCS benchmark and the original training environments in Appendix B.2. MaDi also shows competitive performance in these additional settings. Note that the environments `ball_in_cup-catch` and `finger-spin` have *sparse* rewards, but even in this setting MaDi is able to perform well and generate useful masks. In Appendix B.3 we provide examples of different masks produced for each domain.

3.5.3 Ablation on Augmentation

In Section 3.4.2 we described the expectation that MaDi would perform better with augmentations, as that can help it to recognize which pixels are irrelevant. To verify whether this intuition holds, we run five seeds of MaDi without the `overlay` augmentation on all six environments. We call this variant MaDi-SAC, as it now builds on top of SAC [HZAL18] instead of SVEA.

The results are shown in Table 3.2. The generalization performance of the algorithms that use augmentation is much better than those without. MaDi indeed benefits from augmentation to become more robust against unseen distractions.

Table 3.2. Ablation study showing the effect of the `overlay` augmentation. SVEA and MaDi both use it, while SAC and MaDi-SAC do not. All algorithms are trained for $500K$ steps. We show mean undiscounted return and standard error over five seeds evaluated on `video_hard`. The results reveal that MaDi benefits from data augmentation.

`video_hard`	SAC	MaDi-SAC	SVEA	MaDi
`ball_in_cup catch`	176 ±38	190 ±52	327 ±59	758 ±135
`cartpole balance`	314 ±12	237 ±6	579 ±26	827 ±25
`cartpole swingup`	140 ±10	132 ±9	453 ±26	619 ±24
`finger spin`	21 ±4	121 ±36	154 ±31	358 ±25
`walker stand`	233 ±28	320 ±88	847 ±18	920 ±14
`walker walk`	168 ±19	95 ±24	526 ±55	504 ±33
`avg`	175	183	481	664

3.5.4 Does MaDi work with Vision Transformers?

In image-based RL, the use of Vision Transformers [DBK+21, **ViT**] has recently gained in popularity [HSW21, TRvdP22]. We set up a small experiment to verify whether MaDi still works in this setting. The shared encoder (see Figure 3.1), which is originally an 11-layer ConvNet, is now replaced by a ViT of 4 blocks with 8 attention heads each. We maintain the same architecture for the Masker network. More implementation details on the ViT encoder are in Appendix B.1.1.

We train the ViT-based versions of SVEA and MaDi for 300K timesteps on the clean environments of `walker-walk` and `cartpole-swingup`, and test on `video_hard` at every 10K steps. The results presented in Figure 3.5 show that not only does MaDi work well with a ViT encoder, but it even boosts the generalization performance.

3.6 ROBOTIC EXPERIMENTS

In this section, we describe our experiments on a robotic arm visual reacher task, showing that MaDi can learn to generalize when facing distracting backgrounds, even in real-world environments.

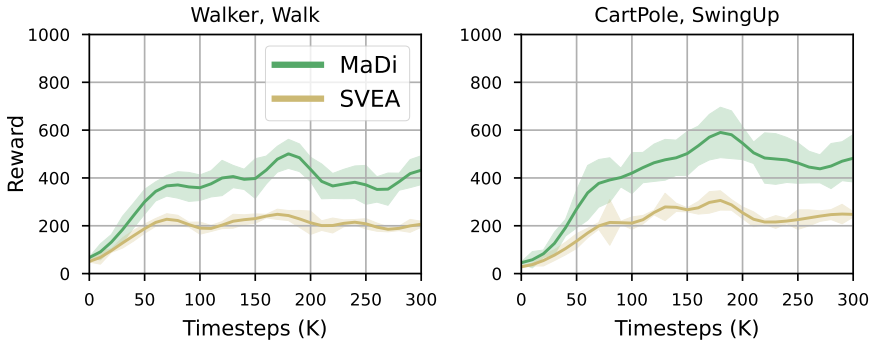


Figure 3.5. Generalization performance of MaDi and SVEA on `video_hard` when trained with a **ViT** encoder for $300K$ steps. We show the mean and standard error over five seeds.

3.6.1 Experimental Setup

Robotic Arm The UR5 industrial robot arm consists of six joints that can rotate to move the tip to a desired position. It uses the conventional TCP/IP protocol to transmit the state of the arm to the host computer and receive actuation packets in return at an interval of 8 milliseconds. A state packet includes the angles, velocities, target accelerations, and currents of all six joints. We configured the UR5 to use the velocity control mode. Since the UR5 does not come with a camera, we attached a Logitech RGB camera to the tip of the arm to facilitate vision-based tasks. We use *SenseAct*, a computational framework for robotic learning experiments to communicate with the robot [MKV+18]. This setup enables robust and reproducible learning across different locations and conditions.

Training environment We train on the UR5-VisualReacher task [YM22, WVM23] for real-world robot experiments. Figure 3.6 depicts the experimental setup. This task involves using a camera to guide a UR5 robotic arm to reach a red target on a monitor. We ensure that the arm stays within a safe bounding box to avoid collisions. The available actions are represented by the desired angular velocities for five[4] joints, ranging from -0.7 to 0.7 radians per second. The full observation that the learning agent receives includes three consecutive RGB images from the Logitech camera of dimensions 160×90 pixels, joint angles, joint angular velocities, and the previous

[4] We do not move the sixth joint because its sole purpose is to control a gripper.

(a) Training environment **(b)** Testing environment

Figure 3.6. The UR5-VisualReacher(-VideoBackgrounds) benchmark used in our experiments. The agent is rewarded for getting its camera as close as possible to the red circle randomly located on the screen. For visual demonstrations of the robot's performance with each algorithm, please refer to the corresponding video,[5] which includes clips showcasing the robot in action.

action taken. The reward function is defined as follows [YM22]:

$$r_t = \frac{c}{hw} M_t \odot W$$

where c is a scaling coefficient, h and w are the height and width of the image in pixels, M_t is a binary mask[6] of shape $h \times w$ that detects for each pixel whether it is currently red, and W is a weighting matrix of shape $h \times w$ with values decreasing from 1 at its center to 0 near the edges. These are multiplied element-wise by the Hadamard product $\odot$. The reward incentivizes the robot to move its camera closer to the target and keep the target at the center of the frame. We set the coefficient $c = 800$ for all experiments and clip the rewards to be between 0 and 4. An episode lasts 150 timesteps of 40 ms each, for 6 seconds in total. The agent sends an action at every timestep, which is repeated by the SenseAct system five times, at every 8 ms.

Real-time asynchronous learning We use the ReLoD system [WVM23] to facilitate effective, real-time learning. In simulation, environments can be internally paused while agents carry out learning updates. However, in the real world, the environment does not wait for the agent to complete its sequential computations and learning updates. We use an asynchronous implementation of SAC [YM22, WVM23] to gather data in real-time and implement learning updates through a separate process. We reimplemented MaDi and all baselines used in these experiments to their

[5] See https://youtu.be/TQMazg6dntE
[6] Note that this is a preprogrammed mask based on RGB thresholds, not made by MaDi.

asynchronous versions. Each learning update can take anywhere from 70 to 500 ms, depending on the chosen algorithm and hyperparameter configuration, meaning that there will be fewer updates than steps in the environment.

Testing environment We test the generalization performance on a similar task, but now with videos playing in the background on the screen. We define this new generalization benchmark as UR5-VisualReacher-VideoBackgrounds (see Figure 3.6b), selecting five videos from the DMControl-GB [HW21] to use as backgrounds. We test on this benchmark after every 4500 timesteps in the training environment. This evaluation is always done for 10 episodes, twice on each video.

Baselines We compare SAC [HZAL18], RAD [LLS+20], SVEA [HSW21], and MaDi. The base architecture is the same for all methods, but it differs somewhat from the simulation experiments of Section 3.5. See Appendix B.1 for details.

3.6.2 Results

In Figure 3.7, we present results on the testing environment with video backgrounds. Without being trained on these distracting visuals, MaDi is able to generalize well in this challenging task. The algorithms are trained asynchronously in this robotic environment, which means that MaDi will make fewer updates as it uses the additional Masker network. However, as Figure 3.7 shows, this only incurs a small delay in learning while gaining superior generalization capability through the increased focus on task-relevant pixels. A similar pattern is present on the original training environment, as shown in Figure B.2 of Appendix B.2.1.

In Appendix B.2.2, we present an additional experiment with a *sparse reward* function. In this UR5-VisualReacher-SparseRewards task, the agent is only rewarded with a +1 whenever its camera is close enough to the red target (according to a predefined threshold), and receives zero reward otherwise. Even in this challenging setting, MaDi is able to surpass the baselines in generalization performance.

3.6.3 Analysis

In Figure 3.8 we show that MaDi can learn to recognize the task-relevant features even in this real-world robotic task. It is able to generalize to the unseen testing environment and produce helpful masks. There seem to be fewer pixels dimmed in this environment compared to the other benchmarks, which may be because it can be

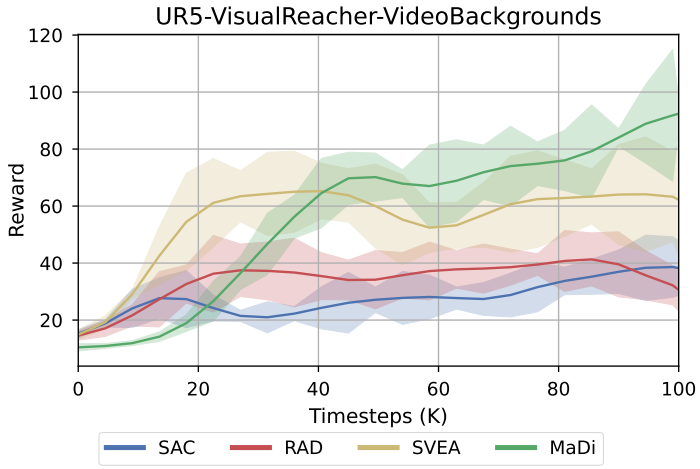


Figure 3.7. Performance of MaDi and three baselines on the UR5-VisualReacher task with random videos playing in the background. Agents are trained on the clean environment for $100K$ steps. We display the mean and standard error over five seeds. MaDi already generalizes best after $50K$ timesteps.

Table 3.3. The average reward per timestep on the UR5-VisualReacher task during the last 10% of steps in an episode. MaDi receives higher rewards in the final position of an episode in both training and testing environments, showing that it finds the red target with higher accuracy.

Reward per step	SAC	RAD	SVEA	MaDi
Training env.	1.38 ±0.09	1.32 ±0.11	1.18 ±0.37	1.95 ±0.09
Testing env.	0.32 ±0.04	0.24 ±0.07	0.47 ±0.14	0.74 ±0.07

useful for the agent to know where the entire screen is positioned.

Distinction between training and testing MaDi performs well in both the training and test environments, but there is quite a large discrepancy between the total rewards received. For all algorithms, the reward in the testing environment is substantially lower than in the training environment. Taking a qualitative look at the behavior of the robotic arm, it seems this is mostly due to the fact that the robotic arm moves slower toward the target in the testing environment than in the training environment. The agents encounter unseen observations that significantly deviate from the training environment, likely driving them to select different actions that do not match well in sequence, causing the arm to slow down. The SAC and RAD agents rarely complete the task at all when there are videos playing in the background.

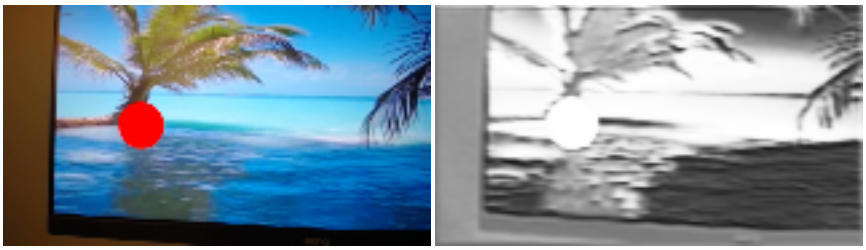


Figure 3.8. Observation and the corresponding mask generated by MaDi in the UR5-VisualReacher-VideoBackgrounds task. The mask is subtle, but clearly leaves the red dot's pixels intact while dimming other areas of the frame. See Appendix B.3 for more examples.

Even though the movement towards the goal is slower in the testing environment for all algorithms, MaDi does often reach a (near) optimal state at the end of its trajectory, similar to training. See Table 3.3 for an overview of the rewards in the last 10% of steps for each baseline. MaDi shows a higher accuracy in finding the red target near the end of an episode.[7]

3.7 CONCLUSION

In the domain of vision-based deep reinforcement learning, we formalize the problem setting of distracting task-irrelevant features. We propose a novel method, MaDi, which learns directly from the reward signal to *ma*sk *di*stractions with a lightweight Masker network, without requiring any additional segmentation labels or loss functions. Our experiments show that MaDi is competitive with state-of-the-art algorithms on the DeepMind Control Generalization Benchmark and the Distracting Control Suite, while only using 0.2% additional parameters. The masks generated by MaDi enhance the agent's focus by dimming visual distractions. Even in the sparse reward setting, the Masker network is able to learn where the task-relevant pixels are in each state. Furthermore, we test MaDi on a real UR5 Robotic Arm, showing that it can outperform the baselines not only in simulation environments but also on our newly defined UR5-VisualReacher-VideoBackgrounds generalization benchmark.

The algorithms in this work build on the model-free off-policy deep RL algorithm SAC, while other options remain open for investigation. In future work, we seek to apply MaDi to other reinforcement learning algorithms such as PPO [SWD+17] or DQN [MKS+15] and improve the clarity of its masks. We have experimented with

[7] Moreover, MaDi generally moves to the target faster, see https://youtu.be/TQMazg6dntE

MaDi on one robotic arm, it would be interesting to see whether the Masker network can produce practical masks on a diverse set of robots.

The scope of this chapter encompasses problem settings where the environment's adjustments are always irrelevant to the task at hand. However, we wish to create agents that can also adapt to task-relevant situational changes. This will be the focus of the following chapter.

CHAPTER 4

Adapting to Unseen Contexts

As we have seen in Chapter 3, we aim to create agents that can adapt to any changes in the environment. The MaDi algorithm generalizes well when the task-irrelevant background is adjusted. But can we also design algorithms that adapt well to task-relevant changes? Welcome to the field of contextual reinforcement learning, where an RL agent needs to adapt to an adjustable context within the environment.

Generalization to unseen environments is a significant challenge in the field of robotics and control. In this work, we focus on contextual reinforcement learning, where agents act within environments with varying contexts, such as self-driving cars or quadrupedal robots that need to operate in different terrains or weather conditions than they were trained for. We tackle the critical task of generalizing to out-of-distribution (OOD) settings, *without* access to explicit context information at test time. Recent work has addressed this problem by training a context encoder and a history adaptation module in separate stages. While promising, this two-phase approach is cumbersome to implement and train. We simplify the methodology and introduce SPARC: **s**ingle-**p**hase **a**daptation for **r**obust **c**ontrol. We test SPARC on varying contexts within the high-fidelity racing simulator *Gran Turismo 7* and wind-perturbed *MuJoCo* environments, and find that it achieves reliable and robust OOD generalization.

This chapter was adapted from: Bram Grooten, Patrick MacAlpine, Kaushik Subramanian, Peter Stone, and Peter R. Wurman. Out-of-Distribution Generalization with a SPARC: Racing 100 Unseen Vehicles with a Single Policy. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026. **Oral**.

4.1 INTRODUCTION

Deep reinforcement learning (RL) has demonstrated successful performance in fields such as robotics [MKV+18], nuclear fusion [DFB+22], and high-fidelity racing simulators [WBK+22]. Despite these successes, generalizing RL agents to unseen environments with varying contextual factors remains a critical challenge. In real-world applications, environmental conditions such as friction, wind speed, or vehicle dynamics can change unpredictably, often leading to catastrophic failures when the agent encounters out-of-distribution (OOD) contexts that it was not trained for.

A promising approach to tackle this issue is context-adaptive reinforcement learning [BES+21], where agents infer and adapt to latent environmental factors by leveraging past interactions. Rapid Motor Adaptation (RMA) [KFPM21] is a notable framework in this direction, introducing a two-phase learning procedure. In the first phase, a context encoder is trained using privileged information about the environment. The second phase then employs supervised learning to train a history-based adaptation module, enabling the agent to infer latent context solely from past state-action trajectories. While effective, this two-phase approach introduces complexity during implementation and training.

In this work, we introduce **SPARC** (**s**ingle-**p**hase **a**daptation for **r**obust **c**ontrol), a novel algorithm that unifies context encoding and adaptation into a single training phase, as illustrated in Figure 4.1. SPARC is straightforward to implement and naturally integrates with off-policy training as well as asynchronous distributed computation on cloud-based rollout workers. Algorithms such as SPARC and RMA are advantageous when explicit context labels are unavailable at test time, a frequent limitation in real-world robotic deployment. By collapsing adaptation into a single training loop, SPARC is naturally compatible with on-device continual learning—especially applicable in settings where retraining in the cloud is prohibitive due to privacy or latency constraints. In contrast, RMA is unable to perform continual learning in a straightforward manner.

We evaluate SPARC on two distinct domains: (1) a set of *MuJoCo* environments featuring strongly varying environment dynamics through the use of wind perturbations, and (2) a high-fidelity racing simulator, *Gran Turismo 7*, where agents must adapt to different car models on multiple tracks. SPARC achieves state-of-the-art generalization performance and consistently produces Pareto-optimal policies when evaluated across multiple desiderata.

Our contributions are summarized as follows.

- We introduce SPARC, a novel single-phase training method for context-adaptive reinforcement learning, eliminating the need for separate encoder pre-training.
- We empirically validate SPARC's generalization ability across OOD environments, demonstrating competitive or superior performance compared to existing approaches.
- We perform and analyze several ablation studies, examining key design choices such as history length and the selection of rollout policy during training.

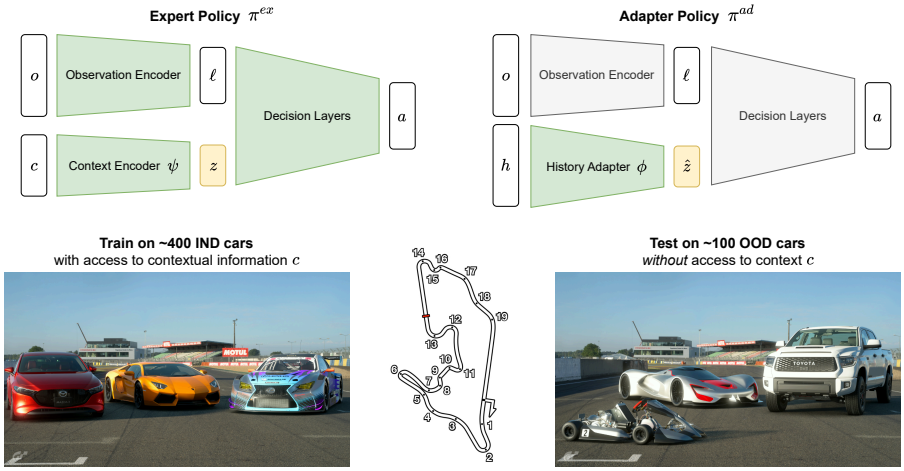


Figure 4.1. Overview of our algorithm SPARC (**top**) and the problem setting in Gran Turismo 7 (**bottom**). SPARC trains an expert policy π^{ex} and an adapter policy π^{ad} simultaneously in a single phase. The adapter policy does not require access to privileged contextual information, facilitating deployment to OOD real-world scenarios. Observations o, contextual information c, and a history of recent observation-action pairs h are passed into the networks. Latent encodings ℓ and z are concatenated and passed to the final layers, producing action a. Similar to RMA [KFPM21], π^{ex} is trained with reinforcement learning, while the History Adapter ϕ of π^{ad} is trained with supervised learning to regress its encoding $\phi(h) = \hat{z}$ to the Context Encoder's output $\psi(c) = z$. Note that since SPARC trains in one phase, the context encoding z is a moving target, instead of a traditionally fixed target in RMA. Trainable modules are in green. The black modules regularly copy weights from their counterpart in π^{ex}.

4.2 RELATED WORK

Generalization to out-of-distribution environments is a fundamental challenge in reinforcement learning, hindering its deployment in real-world applications, particularly in robotics and control tasks [KZGR23]. The learning dynamics of RL methods often struggle to adapt to novel environmental conditions [LRD+22]. Contextual reinforcement learning [Lan17, BES+21] provides a framework to address this problem by training agents capable of adapting to varying environmental factors.

4.2.1 Contextual RL

Robust RL often depends on effective contextual adaptation. Recent work has explored context-aware policies that integrate contextual cues into decision-making [BJK+23, CLZ+21, LMCZ24] or employ world models to capture environment dynam-

ics [LSL+20, PFRB24]. In addition, several studies have focused on modifying the environment itself—such as by varying gravity or adjusting agent component dimensions—to promote the development of more versatile controllers [BES+21, LRS+24].

4.2.2 What if the Agent has No Access to Context?

In many real-world scenarios, agents are deprived of explicit contextual information during deployment. In these cases, the agent must infer the relevant environmental factors indirectly. For instance, Lee et al. [LHW+20] advanced robust legged locomotion by introducing a two-phase learning process. It first trains an expert policy, which includes a context encoder using the privileged contextual information. The second phase involves an adapter policy that tries to imitate the expert's action, while a history-based adaptation component aims to minimize the difference between its history encoding and the expert's context encoding. Rapid Motor Adaptation [KFPM21] refines this methodology by only imitating the context encoding, not the action. The adapter policy can be deployed, as it does not require access to the privileged context.

4.2.3 Other Techniques for Generalization

Several complementary approaches have been proposed to enhance generalization. Domain randomization [TFR+17, PAZA18] and procedurally generated environments [CHHS20, GEG+21] introduce diversity during training, thereby encouraging robust policy behavior. We employ domain randomization by default in our experiments. System identification methods [YTLT17]—whether performed explicitly or through implicit online adaptation, as in SPARC and RMA—also contribute to improved performance under varying conditions. Moreover, techniques such as data augmentation [LLS+20, HSW21, WWH+24] and masking [GTV+24, HPZ+22] have been shown to further enhance generalization, particularly for pixel-based inputs.

Meta-reinforcement learning offers an alternative paradigm for learning adaptable policies [WKNT+16, RPS+18, DSC+16]. Foundational algorithms like Model-Agnostic Meta-Learning (MAML) [FAL17] enable rapid task adaptation, and emerging methods using hypernetworks can generate task-specific policy parameters on the fly [BJVW23, RSMH+23, BJK+23].

4.3 BACKGROUND

In this section, we formalize the underlying problem framework and examine the core techniques that form the foundation for SPARC, enabling context-adaptive behavior.

4.3.1 Problem Formulation

We consider a contextual Markov decision process (CMDP) [HDCM15, AYN14], defined by Kirk et al. [KZGR23] as a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{C}, R, T, O, p_s, p_c)$ where:

- $\mathcal{S}$ is the state space,
- $\mathcal{A}$ is the action space,
- $\mathcal{O}$ is the observation space,
- $\mathcal{C}$ is the context space,
- $R : \mathcal{S} \times \mathcal{A} \times \mathcal{C} \to \mathbb{R}$ is the reward function,
- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{C} \to \Delta(\mathcal{S})$ defines the stochastic transition dynamics conditioned on a context $c \in \mathcal{C}$,
- $O : \mathcal{S} \times \mathcal{C} \to \mathcal{O}$ is the observation function,
- $p_s : \mathcal{C} \to \Delta(\mathcal{S})$ is the distribution over initial states s_0 given a context $c \in \mathcal{C}$,
- $p_c \in \Delta(\mathcal{C})$ is the distribution over contexts.

During training, the agent will be exposed to a certain subset of contexts $\mathcal{C}_{\text{IND}} \subset \mathcal{C}$, which are in-distribution (IND), short for *within the training distribution*. To test generalization ability, we hold out a different subset of contexts $\mathcal{C}_{\text{OOD}} \subset \mathcal{C}$ that are out-of-distribution (OOD). We ensure that there is no overlap: $\mathcal{C}_{\text{IND}} \cap \mathcal{C}_{\text{OOD}} = \varnothing$. This separation defines two sub-CMDPs: $\mathcal{M}_{\text{IND}}$ and $\mathcal{M}_{\text{OOD}}$. We specify the context distributions to be uniform over their respective subsets:

$$p_c^i(c) = \begin{cases} \frac{1}{|\mathcal{C}_i|} & \text{if } c \in \mathcal{C}_i \\ 0 & \text{otherwise,} \end{cases} \tag{4.1}$$

for $i \in \{\text{IND}, \text{OOD}\}$.

In our setting, the agents do not observe c at test time and must infer it through other means, for example from their interaction history. However, for comparison,

we will also present results of an expert policy that *does* have access to the privileged context information $c \in \mathcal{C}_{\text{OOD}}$ at evaluation.

Our objective is to train a policy π that maximizes expected return across both in-distribution and out-of-distribution contexts, while only having access to privileged contextual information $c \in \mathcal{C}_{\text{IND}}$ during training.

4.3.2 Pure History-based Policies

History-based policies have emerged as a powerful approach in reinforcement learning for inferring hidden environmental context from past interactions. Instead of relying solely on the current observation $o_t \in \mathcal{O}$, these policies condition action selection on a sequence of recent observation-action pairs. Let H be the history length and $\mathcal{H} = (\mathcal{O}\times\mathcal{A})^H$, the space of possible histories. For time t we define the corresponding history h_t as

$$h_t = (o_{t-H:t-1}, a_{t-H:t-1}) \in \mathcal{H}.$$

This history input results in policies of the form $\pi : \mathcal{O} \times \mathcal{H} \to \Delta(\mathcal{A})$. Including the history may enable the agent to implicitly capture latent context information $c \in \mathcal{C}$, as the context c may influence the environment dynamics.

A pure history-based approach is presented by Lee et al. [LHW+20] as a strong baseline. In their work on quadrupedal locomotion over challenging terrains, the authors demonstrate that leveraging an extended history of proprioceptive data via a temporal convolutional network (TCN) enables robust control in diverse settings.

4.4 SINGLE-PHASE ADAPTATION FOR ROBUST CONTROL

4.4.1 Leveraging Contextual Information

Training with privileged contextual information—even if not available at test time—has been shown to be particularly useful for generalizing to OOD contexts. In that regard, the approaches by Lee et al. [LHW+20] and Kumar et al. [KFPM21] are almost equivalent; we will focus on Rapid Motor Adaptation (RMA) [KFPM21]. In RMA, two policies are trained in separate phases. First, the expert policy

$$\pi_\theta^{ex} : \mathcal{O} \times \mathcal{C} \to \Delta(\mathcal{A})$$

which includes a context encoder $\psi(\cdot)$ with access to the environment's privileged information, is trained using a reinforcement learning algorithm. While the original RMA work uses PPO [SWD+17], we employ the more sample-efficient QR-SAC, proven to work well in Gran Turismo [WBK+22].

Once training of π_θ^{ex} has converged to a sufficient level, the best model checkpoint $\pi_{\theta^*}^{ex}$ needs to be determined. This selection requires careful evaluation across multiple dimensions [MWH+23], a cumbersome intermediate step that SPARC skips, as it is trained in a single phase.

The second stage of RMA trains the adapter policy

$$\pi_\theta^{ad} : \mathcal{O} \times \mathcal{H} \to \Delta(\mathcal{A})$$

while keeping the expert policy $\pi_{\theta^*}^{ex}$ frozen. In the adapter policy, a *history adapter* ϕ_θ processes a sequence of recent observation-action pairs h_t to produce a latent representation $\hat{z}_t = \phi_\theta(h_t)$. The history adapter is trained by minimizing the distance between $\hat{z}_t = \phi_\theta(h_t)$ and $z_t = \psi_{\theta^*}(c_t)$ through the mean squared error loss:

$$\mathcal{L}_\phi(c_t, h_t) = \mathbb{E}_{c_t,h_t}[(z_t - \hat{z}_t)^2]. \tag{4.2}$$

The history-inferred latent context $\hat{z}_t$ is integrated into the policy. By conditioning on both the current observation o_t and the latent context $\hat{z}_t$, the policy can adjust its behavior to handle unseen or varying environmental conditions.

4.4.2 Single-Phase Adaptation

Our algorithm illustrated in Figure 4.1, SPARC, greatly simplifies the implementation and training of agents capable of generalizing to out-of-distribution environments without access to privileged contextual information. In SPARC, the expert policy π^{ex} and the adapter policy π^{ad} are trained simultaneously, in contrast to the two-phase approach of RMA. This means that the context encoding $\psi(c) = z$ is a non-stationary target for the history adapter ϕ, instead of a fixed target. The results in Section 4.6 demonstrate that the adapter policy is able to manage these new learning dynamics.

An important detail in RMA is which model acts in the environment to collect experience. Policy π^{ex} acts in the first training phase, while π^{ad} does so in the second. This raises the question of which policy should gather experience for SPARC, as both are trained together. One option would be to let the expert policy π^{ex} control the actions, since it is updated and improved through QR-SAC.

Table 4.1. Performance summary on IND and OOD settings across all test tracks in Gran Turismo, averaged over 3 seeds. Results show the mean built-in AI (BIAI) ratio across cars (ratio = the RL agent's lap time divided by the BIAI lap time, lower is better). If an algorithm fails to complete a lap with a specific vehicle, it receives a BIAI ratio of 2.0 for that car model. Additionally, we show the percentage of cars with a successfully completed lap (± s.e.m.). We **bold** the best out-of-distribution results across algorithms without access to context at test time (all except Oracle, see Table 4.3). We include IND results for reference. SPARC achieves the fastest OOD lap times on ⅔ tracks and completes the most laps with OOD vehicles overall.

Race Track	Method	IND		OOD	
		BIAI ratio (↓)	Success % (↑)	BIAI ratio (↓)	Success % (↑)
Grand Valley	Only Obs	0.9929 ± 0.0007	100.00 ± 0.00	1.0641 ± 0.0058	95.15 ± 0.56
	History Input	0.9904 ± 0.0001	99.68 ± 0.08	1.0826 ± 0.0203	92.56 ± 2.12
	RMA	1.0046 ± 0.0054	99.84 ± 0.16	1.0560 ± 0.0134	97.09 ± 1.12
	SPARC	0.9999 ± 0.0061	99.76 ± 0.14	**1.0491 ± 0.0055**	**98.06 ± 0.56**
	Oracle	0.9884 ± 0.0005	100.00 ± 0.00	1.1348 ± 0.0137	90.94 ± 2.27
Nürburgring	Only Obs	1.0202 ± 0.0163	95.87 ± 1.48	1.1745 ± 0.0129	81.88 ± 1.17
	History Input	0.9984 ± 0.0030	97.49 ± 0.32	1.1204 ± 0.0132	86.73 ± 1.29
	RMA	1.1085 ± 0.0195	88.03 ± 1.76	1.2995 ± 0.0306	77.99 ± 3.19
	SPARC	1.0254 ± 0.0061	95.87 ± 0.49	**1.1199 ± 0.0076**	**89.00 ± 0.86**
	Oracle	0.9804 ± 0.0027	99.27 ± 0.28	1.1182 ± 0.0215	89.64 ± 2.53
Catalunya Rallycross	Only Obs	0.9319 ± 0.0009	100.00 ± 0.00	0.9560 ± 0.0006	**100.00 ± 0.00**
	History Input	0.9294 ± 0.0001	100.00 ± 0.00	**0.9553 ± 0.0068**	99.33 ± 0.67
	RMA	0.9445 ± 0.0010	99.82 ± 0.18	0.9667 ± 0.0030	**100.00 ± 0.00**
	SPARC	0.9432 ± 0.0027	100.00 ± 0.00	0.9631 ± 0.0026	**100.00 ± 0.00**
	Oracle	0.9282 ± 0.0001	100.00 ± 0.00	1.1354 ± 0.0595	85.33 ± 5.81

However, the expert policy, π^{ex}, is not the goal of the SPARC approach. A robust adapter policy, π^{ad}, is the overall learning target, and using this policy to gather experience allows the learning algorithm to correct for any inaccuracies before final deployment. This brings the learning dynamics of π^{ad} closer to an on-policy setting, even though its history adapter ϕ is trained through supervised learning as shown in Equation 4.2. We perform an ablation study on this choice of rollout policy in Appendix C.2.1.

SPARC's critic networks—necessary to run QR-SAC on π^{ex}—have the same architecture as the expert policy and thus have access to the context c. We can still run inference without knowing c, because at test time only π^{ad} is needed.

Reducing training of SPARC to one phase provides several benefits: (i) no intermediate selection of the best model checkpoint of the first phase is necessary, (ii) training can be easily continued indefinitely, without having to retrain the second phase, (iii) the simpler implementation facilitates the use of SPARC on asynchronous distributed systems.

4.5 EXPERIMENTAL SETUP

In this section, we describe the experimental setup used to evaluate SPARC, our proposed single-phase adaptation method, in comparison with several baselines.

4.5.1 Environments

We evaluate our approach on two distinct domains:

- **MuJoCo:** A suite of continuous control tasks including *HalfCheetah*, *Hopper*, and *Walker2d* [TET12]. We induce contextual variability by perturbing the environment's wind speed in multiple dimensions and scales, creating challenging OOD scenarios.
- **Gran Turismo 7:** A high-fidelity racing simulator that features diverse car models and realistic vehicle-track dynamics [WBK+22]. The simulator's rich contextual variability makes it an ideal testbed for assessing generalization to unseen conditions.

Within Gran Turismo, we experiment on two settings: (1) generalization across car models, and (2) generalization across differing engine power and vehicle mass settings for one specific car. The in-distribution (IND) training set and OOD test set are selected as follows:

(1) *Car Models*: we sort all ~500 vehicles by their anomaly score by applying an isolation forest method[1] to the car's contextual features such as mass, length, width, weight distribution, power source type, drive train type, wheel radius, etc. We hold out the 20% most *outlier* vehicles as a test set (OOD) and train on the 80% most *inlier* cars (IND).

(2) *Power & Mass:* for a more controlled experiment, we pick a relatively standard racing car, but tune its engine power and mass in each training episode to randomly sampled values within the range [75%, 125%] of their defaults. During evaluation, we test on fixed-spaced intervals within [50%, 150%], covering IND and OOD contextual settings.

For the wind-perturbed MuJoCo environments, we similarly train all algorithms on a certain range of wind speeds, while testing on intervals twice as large. In Gran Turismo, we experiment on three different tracks, presented in Table 4.2. These tracks

[1] See the scikit-learn documentation for the IsolationForest algorithm.

Table 4.2. Gran Turismo tracks which we experiment on in the *Car Models* setting. The road type and track length pose varying challenges.

Track	Length	Road Type
Grand Valley	5.099 km	Tarmac
Nürburgring	25.378 km	Tarmac + Concrete
Catalunya Rallycross	1.133 km	Dirt + Tarmac

Table 4.3. The set of inputs that each algorithm receives.

Method	Inputs during **Training**	Inputs at **Test Time**
Only Obs	`obs`	`obs`
History Input	`obs`, `history`	`obs`, `history`
RMA	`obs`, `history`, `context`	`obs`, `history`
SPARC	`obs`, `history`, `context`	`obs`, `history`
Oracle	`obs`, `context`	`obs`, `context`

represent highly varying settings, with *Catalunya Rallycross* even including a mixed dirt and tarmac racing path.

4.5.2 Training Details

We repeat our runs with independent random seeds to ensure statistical robustness: three seeds for the compute-heavy Gran Turismo simulator, and five for MuJoCo environments. Key training hyperparameters—such as the history length H, learning rates, and network architectures—are tuned through preliminary experiments with grid search. We train all methods asynchronously, collecting experience on distributed rollout workers. Further training details and analyses are provided in Appendix C.

4.5.3 Evaluation Protocol

We evaluate policy performance under two settings:

- **In-Distribution (IND):** Environments with contextual parameters that lie within the training distribution.
- **Out-of-Distribution (OOD):** Contextual parameters that deviate significantly from the training set, testing the model's generalization capabilities.

During training, we regularly evaluate the policy on three predetermined IND settings. These training evaluations form a Pareto-front, from which we select the best model

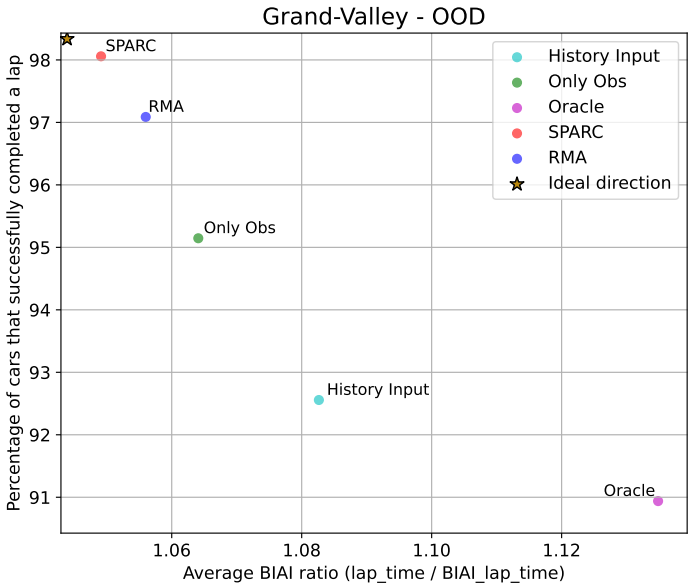


Figure 4.2. Results on Grand Valley averaged over three seeds. For each algorithm, we plot the percentage of cars that successfully completed laps, and the built-in AI ratio lap time. SPARC is able to complete the most and the fastest laps on out-of-distribution cars.

checkpoint for each run. We then test the policy on a wide range of IND & OOD contexts. For *Car Models*, this means all unique vehicles, while for *Power & Mass* and *MuJoCo* we divide the widest context ranges into fixed intervals.

Performance metrics include the return for *MuJoCo* and lap times in *Gran Turismo*. However, for particularly difficult outlier cars, some algorithms may not be able to complete any laps. For this reason, we present the racing results along two dimensions: (1) percentage of cars with a completed lap, and (2) the average lap time. Note that (2) is a biased metric, so (1) needs to be taken into account.

When averaging raw lap times, slower cars have a larger impact on the average. To avoid skewed results, we divide by the built-in AI (BIAI) lap time for each specific car. The BIAI is a classical control method implemented in Gran Turismo 7 to follow a preset driving line. This *BIAI ratio* of RL lap time over BIAI lap time provides an informative normalized value. When a car was unable to complete a lap, we set its BIAI ratio to 2.0 before averaging over all vehicles.

4.5.4 Baselines

We compare the performance of the following algorithms.

- **Only Obs:** This QR-SAC [WBK+22] policy is trained without any context information. Only the current observation is provided as input.
- **History Input:** A baseline policy [LHW+20] that additionally receives a history of observation-action pairs $h_t = (o_{t-H:t-1}, a_{t-H:t-1})$.
- **RMA:** The two-phase approach of Rapid Motor Adaptation [KFPM21], first trains an expert policy with context input, then learns the adapter policy from history.
- **SPARC:** Our single-phase adaptation technique introduced in this work. At test time it only receives an observation-action history and the current observation.
- **Oracle:** A policy that has access to the ground-truth unencoded contextual features, even at test time.

Benchmarking SPARC against the listed baselines allows us to isolate the benefits of our single-phase training paradigm, especially regarding implementation simplicity and OOD generalization. See Table 4.3 for an overview of the inputs per method.

4.6 RESULTS

We present the performance of SPARC and several baselines on *Gran Turismo* and *MuJoCo* environments, focusing on generalization to unseen contexts.

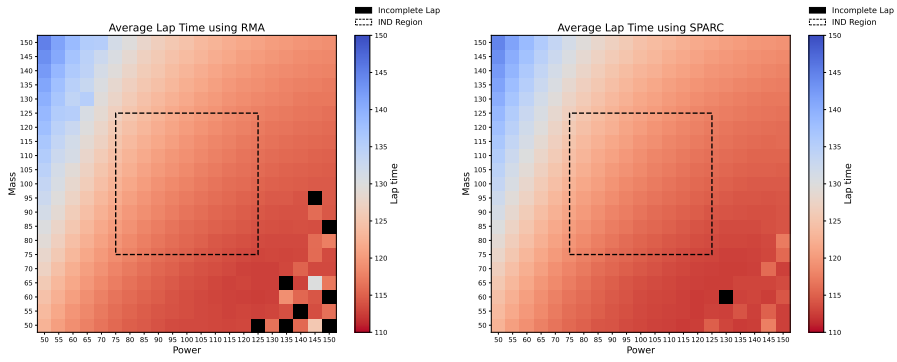
4.6.1 Gran Turismo: Car Models

The scatterplot in Figure 4.2 summarizes the performance of each algorithm averaged over all out-of-distribution cars on the race track Grand Valley. The results indicate that SPARC outperforms the baselines across unseen vehicles during training. SPARC completes laps with the most cars and with the fastest average lap time.

Table 4.1 provides a quantitative summary of our findings across all three tracks. On IND settings, SPARC is competitive, but it is particularly designed to handle OOD dynamics. When racing untrained cars, SPARC is the fastest of all algorithms without access to context at test time on 2 out of 3 tracks. Furthermore, our method manages to complete laps with the most OOD vehicles on aggregate.

Table 4.4. Performance summary of the *Power & Mass* experiments, averaged over 3 seeds. Results show the mean built-in-AI lap-time ratio (2.0 if no lap completed) across all OOD power & mass settings, and the percentage of these settings with a successfully completed lap (± s.e.m.). SPARC completes the most and has the fastest laps.

Method	Built-in-AI lap-time ratio (↓)	Success % (↑)
Only Obs	1.0131 ± 0.0136	98.75 ± 1.25
History Input	1.0135 ± 0.0013	98.33 ± 0.10
RMA	1.0004 ± 0.0030	99.17 ± 0.28
SPARC	**0.9907 ± 0.0011**	**99.90 ± 0.10**
Oracle	0.9962 ± 0.0067	99.27 ± 0.58



(a) RMA **(b)** SPARC

Figure 4.3. Lap times on the *Power & Mass* experiment. Colours denote average lap time over 3 seeds (red = fast, blue = slow); black squares indicate at least one unfinished lap. Even though both algorithms are trained only on settings within the IND region (dashed box), SPARC is able to handle challenging OOD settings in the bottom right corner (high power and low mass).

SPARC even outperforms its two-phase counterpart RMA. We believe this occurs because SPARC avoids the brittle selection of a phase-1 checkpoint, required by RMA. Training an adapter ϕ against one checkpoint of ψ can overfit to parts of the context space, while SPARC learns against multiple strong checkpoints over time.

4.6.2 Gran Turismo: Power & Mass

In Figure 4.3 we show the difference between (a) the strongest baseline and (b) our method. SPARC is able to complete laps in almost all OOD contextual settings, while RMA struggles in the most difficult scenarios of lightweight cars with high engine power. Table 4.4 provides a summary of the average results across all OOD contexts,

indicating that SPARC outperforms all baselines, including the Oracle method. The Oracle does not receive history as part of its inputs, as opposed to other baselines. We believe that SPARC is even able to outperform the Oracle in this environment because knowledge of some history may be useful to mitigate the partial observability in our contextual MDP $\mathcal{M}$ (Section 4.3.1). SPARC is the most robust in this experiment—completing laps in all but one setting—and also achieves the fastest average lap-time.

Table 4.5. Performance across MuJoCo environments, averaged over 5 seeds. Results show the mean return over all out-of-distribution wind perturbations (± s.e.m.). SPARC outperforms all baselines in 2 out of 3 environments.

Method	HalfCheetah-v5 (↑)	Hopper-v5 (↑)	Walker2d-v5 (↑)
Only Obs	5724.51 ± 1624.98	1274.13 ± 133.78	2495.77 ± 220.69
History Input	8760.12 ± 161.53	1367.09 ± 67.79	1534.86 ± 144.26
RMA	9033.87 ± 634.11	1307.96 ± 45.65	2306.23 ± 222.09
SPARC	**10017.90 ± 476.19**	1348.22 ± 53.67	**2528.25 ± 263.58**
Oracle	7821.42 ± 1156.77	**1710.14 ± 98.98**	2325.30 ± 576.48

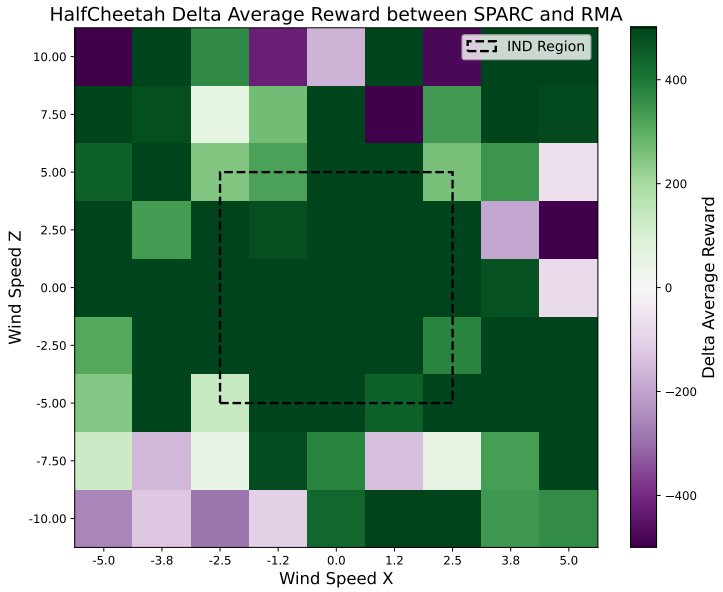


Figure 4.4. Difference in average return of SPARC versus RMA with varying wind perturbations over 5 seeds. In green SPARC is better in that wind setting, while in purple RMA scores higher. Our method outperforms the two-phase baseline across many IND and OOD contextual settings.

4.6.3 MuJoCo: Wind Perturbations

In Figure 4.4, we present results on HalfCheetah by calculating the difference in performance between SPARC and its main baseline RMA, in each wind perturbation tested. The green squares show SPARC outperforming RMA, while purple indicates the opposite. Overall, SPARC beats RMA in significantly more IND and OOD settings, demonstrating a robust performance across varying contexts.

Table 4.5 shows the OOD results for all baselines and MuJoCo environments. Again, SPARC presents strong generalization ability to unseen contexts. On Hopper, the Oracle performs best; note that this baseline has access to true context at test-time, in contrast to all others (see Table 4.3).

4.6.4 Transferability to Updated Game Dynamics

The *Gran Turismo* developers regularly deploy game updates, where the simulation physics can be adjusted. Reinforcement learning agents that are trained on previous

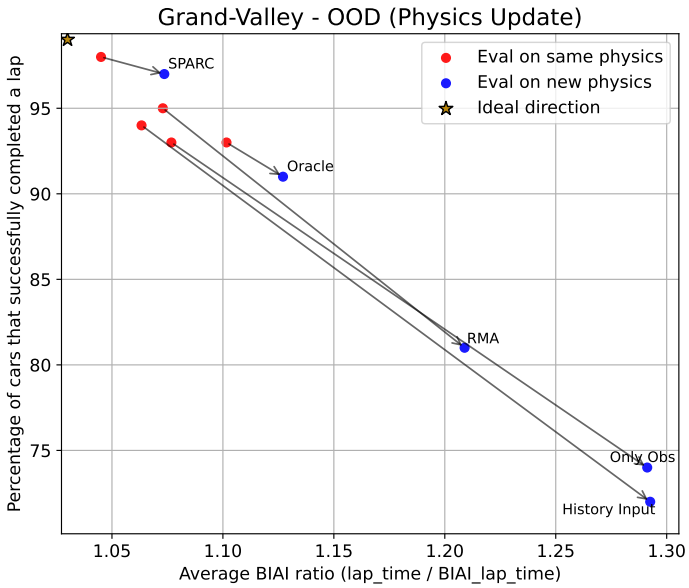


Figure 4.5. Performance difference between old and new game dynamics. These algorithms have only been trained on old physics settings, and are tested zero-shot on the new physics after a game update of Gran Turismo. SPARC shows the best OOD generalization, with only slightly slower lap times on new dynamics, while other methods degrade significantly.

game dynamics may struggle to adapt. Here we evaluate policies on the newest game dynamics, while being solely trained on a previous version of Gran Turismo.

In Figure 4.5 we show that SPARC outperforms all baselines in OOD generalization, this time not only across different car models, but also across other unseen environment dynamics. The Oracle policy with access to ground-truth context is not able to finish laps with around 10% of the OOD cars, while SPARC reduces this to less than 5%, with significantly faster lap times. Note that the context $c \in \mathcal{C}$ that we provide to the Oracle contains information about the car model only, as the exact simulator physics adjustments are unknown to us. This missing information highlights the importance of SPARC's ability to adapt to unseen contexts without access to comprehensive contextual details, e.g., when training in simulation and transferring to a real-world environment.

4.7 CONCLUSION

This chapter introduces SPARC, a novel **s**ingle-**p**hase **a**daptation method for **r**obust **c**ontrol in contextual environments. The algorithm unifies context encoding and history-based adaptation into one streamlined training procedure. By eliminating the need for separate phases, SPARC not only simplifies implementation but also facilitates continual learning and deployment in real-world scenarios.

Our extensive experiments in both the high-fidelity Gran Turismo 7 simulator and various MuJoCo tasks demonstrate that SPARC achieves competitive or superior performance in both in-distribution and OOD settings. In particular, SPARC excels at generalizing to unseen contexts while maintaining robust control, a critical capability for robotics applications where explicit contexts are unknown during deployment.

While our results are promising, the work also highlights opportunities for future research. In particular, testing SPARC on physical robotic platforms and further optimizing its training efficiency remain important next steps. Moreover, reducing the agent's prediction uncertainty when encountering extreme conditions is a difficult problem. We will attempt to tackle this issue using the approach of ensembling methods in the next chapter.

CHAPTER 5

Adapting to Multiple Fields

In this final chapter, we return to the strengths of the dynamic sparse training approach that we have seen in Chapter 2. We demonstrated that the ANF algorithm uses dynamic sparsity to increase its internal adaptability. In this research, we will verify that a similar approach can be useful to improve the effectiveness and efficiency of neural network ensembles. This time across multiple fields, not only RL but also supervised and self-supervised learning.

Model ensembles have long been a cornerstone for improving generalization and robustness in deep learning. However, their effectiveness often comes at the cost of substantial computational overhead. To address this issue, state-of-the-art methods aim to replicate ensemble-class performance without requiring multiple independently trained networks. Unfortunately, these algorithms often still demand considerable compute at inference. In response to these limitations, we introduce **NeuroTrails**, a sparse multi-head architecture with dynamically evolving topology. This unexplored model-agnostic training paradigm improves ensemble performance while reducing the required parameter count.

We analyze the underlying reason for its effectiveness and observe that the various neural trails induced by dynamic sparsity attain a *Goldilocks zone* of prediction diversity. NeuroTrails displays efficacy with convolutional and transformer-based architectures on reinforcement learning, vision, and language tasks. Experiments on DQN/Atari, ResNet-50/ImageNet, LLaMA-350M/C4 demonstrate increased performance and stronger robustness in zero-shot generalization, while requiring significantly fewer resources.

This chapter was adapted from: Bram Grooten, Farid Hasanov, Chenxiang Zhang, Qiao Xiao, Boqian Wu, Zahra Atashgahi, Ghada Sokar, Shiwei Liu, Lu Yin, Elena Mocanu, Mykola Pechenizkiy, and Decebal Constantin Mocanu. Training with Dynamic Sparse Heads as the Key to Effective Ensembling. *Under review*, 2026.

5.1 INTRODUCTION

The idea of combining the outputs of multiple models to produce a stronger predictor has been around for a long time, with foundational works on stacking linear models [Tuk77, Wol92], bagging [Bre96] and boosting [FS97] establishing the efficacy of this approach. Following these early developments, ensembling has proven to be

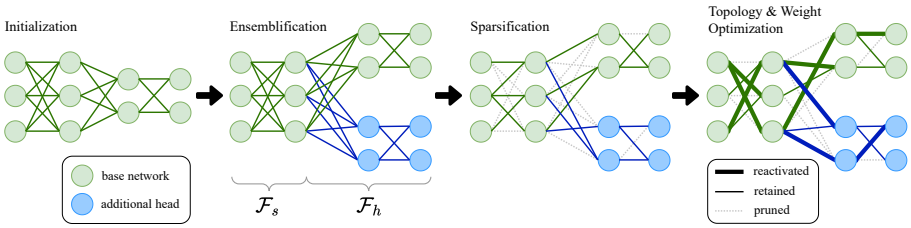


Figure 5.1. Illustration of NeuroTrails. We divide a network into a shared backbone $\mathcal{F}_s$ and multiple independent heads $\mathcal{F}_h$. Weights are initially pruned at random to a target sparsity ratio. Finally, the network topology is repeatedly refined through dynamic sparse training. The resulting sparse multi-head architecture achieves better performance than a full ensemble while using fewer resources.

Table 5.1. NeuroTrails outperforms ensembles across (self-)supervised learning and RL.

	ResNet-50/ImageNet Accuracy (↑)	LLaMA-350M/C4 Perplexity (↓)	DQN/Atari Wins (↑)
Single Network	76.1	22.8	1/10
Full Ensemble	77.5	21.3	0/10
NeuroTrails	**78.1**	**20.7**	**8/10**

a powerful technique in deep learning to increase accuracy, robustness, and generalization performance [HS90, MO11, Zho12]. A common approach involves training multiple deep neural networks independently and averaging their predictions at inference [Zho12]. Random initialization allows ensemble models to explore various local optima, diversifying their predictions [FHL20]. However, the huge increase in required compute is a significant disadvantage. Multiple works have attempted to reduce this overhead by, for example, factorizing weight matrices [WTB20], network distillation [HVD15], or with a Multi-Input Multi-Output configuration (MIMO) [HJF+21], usually reducing the number of parameters of an ensemble to be approximately similar to a single model. An alternative approach to reducing the parameter counts of neural networks lies in the extensive field of pruning [FC19, LAT18, WZG20] and dynamic sparse training [MMS+18, EGM+20]. Various studies leverage these methods to address the complexity challenges associated with ensembles [LCA+22, WW22].

In this work, we approach ensembles from the perspective of TreeNet architectures [LPC+15]. These are structures that share the early layers of neural networks, while retaining as many heads as a corresponding ensemble. While TreeNet's shared backbone reduces the parameter count, the performance may not always match a full ensemble, as the heads often fail to achieve enough separation in prediction diversity.

To resolve this, we introduce **NeuroTrails**, a novel training method enabling ensemble models to share early backbone layers while forming diverse independent trails further in the network, see Figure 5.1. We train the multi-head model using

dynamic sparse training, which allows NeuroTrails to adapt its network topology over time. By tuning the backbone length, the resulting model attains a *Goldilocks zone* of prediction diversity—neither too little nor too much (Section 5.5.2). Furthermore, the sparsity enables parameter reduction, directly translating to inference speedups (Section 5.5.5).

NeuroTrails is model-agnostic, outperforming ensembles built from both convolutional networks (DQN, ResNet-50, Wide-ResNet28-10) and transformer models (LLaMA-130M, LLaMA-350M). It surpasses them on RL, vision, and language benchmarks such as Atari, CIFAR-100, ImageNet, and the Colossal Clean Crawled Corpus, see Table 5.1. Additionally, NeuroTrails displays strong zero-shot generalization to out-of-distribution images and downstream language tasks.

In summary, our contributions are:

- We introduce NeuroTrails, a novel training methodology that improves the performance of neural network ensembles through two key mechanisms: shared early layers and dynamic sparse training.
- We validate our model-agnostic approach with extensive reinforcement learning, computer vision, and language modeling experiments on common benchmarks, showing consistent improvements.
- We provide thorough analysis on NeuroTrails' prediction diversity, real-time CPU speedups, and key design factors—including the optimal splitting point, ensemble size, and sparsity ratio.

5.2 BACKGROUND

5.2.1 Ensembling

Combining the strength of multiple models in an ensemble is widely studied in the literature, and has been shown to reduce variance and improve generalization [HS90]. Ensembles can be used for uncertainty estimation [LPB17], leading to more calibrated probability estimates, covering a larger portion of the problem space, bridging representation gaps left by individual models [Die00, Zho12]. However, the additional computational cost in training and inference of neural network ensembles severely limits their scope of application [GBEB17, Die00].

5.2.2 Sparsity

The sparsification of neural networks has been a prevalent resolution to ease this computational burden [LDS89, FC19, EGM+20]. Sparsifying a network involves removing a certain fraction of its parameters to create a lightweight model. Let an $n \times k$ dense layer be the weighted digraph $G = (V, E_{\text{dense}}, \boldsymbol{\theta})$ where $V = V_{\text{in}} \cup V_{\text{out}}$ is the set of neurons, $E_{\text{dense}} = V_{\text{in}} \times V_{\text{out}}$ the set of potential edges, and $\boldsymbol{\theta} \in \mathbb{R}^{nk}$ the corresponding weight matrix. A binary mask $\mathbf{m} \in \{0,1\}^{nk}$ selects the active edge set $E = \{ e_i \mid m_i = 1 \}$, producing the sparse weight matrix $\boldsymbol{\theta} \odot \mathbf{m}$. The *sparsity ratio* $S = 1 - \|\mathbf{m}\|_0 / nk \in [0,1]$ is the fraction of edges removed.

Pruning. Pruning methods generally involve training a *dense* network to convergence, then selecting a mask $\mathbf{m}$ with the desired sparsity, classifying these algorithms as **dense-to-sparse**. The process ranks each weight θ_i with an importance score s_i, keeping the top $(1-S)nk$ entries. Typically used scores are magnitude $s_i^{\text{mag}} = |\theta_i|$, first-order $s_i^{(1)} = |\theta_i g_i|$ with $g_i = \partial \mathcal{L} / \partial \theta_i$ [MS88], and second-order $s_i^{(2)} = \frac{1}{2} \theta_i^2 H_{ii}$ with $H_{ii} = \partial^2 \mathcal{L} / \partial \theta_i^2$ [LDS89]. A short finetuning pass can restore accuracy [HPTD15].

Sparse Training. Training neural networks with a sparse structure throughout the entire training process is the counterpart of pruning, depicting a **sparse-to-sparse** paradigm [MMP+21]. In *static* sparse training, the network topology is fixed [MMN+16], making it very sensitive to the initial choice of $\mathbf{m}$. *Dynamic* sparse training (DST) solves this issue, enabling the sparse topology to be adaptive. Popular algorithms that exemplify this methodology are Sparse Evolutionary Training (SET) [MMS+18] and Rigged Lottery Tickets (RigL) [EGM+20]. SET starts with a sparsely connected neural network and iteratively updates its structure $\mathbf{m}$ over fixed intervals ΔT. At each topology update, a drop fraction p of the active weights with the smallest magnitude $|\theta_i|$ is pruned, after which an equal number of inactive weights are regrown uniformly at random. RigL uses gradients of inactive connections to guide regrowth, always selecting the highest absolute gradients $|g_i|$ as most promising.

5.3 NEUROTRAILS

We introduce NeuroTrails, a training method to enhance the performance of ensembles, while reducing their parameter complexity. The method is model-agnostic and can be applied to any architecture. See Appendix D.1 for a concise pseudocode overview.

Architecture split. Let the base network $\mathcal{F}$ be a composition of L blocks

$$\mathcal{F}(x;\boldsymbol{\theta}) \;=\; f_L\big(f_{L-1}(\cdots f_1(x;\boldsymbol{\theta}_1)\cdots;\boldsymbol{\theta}_{L-1});\boldsymbol{\theta}_L\big)$$

where a block is a collection of neural network layers, such as a residual or transformer block. We choose a split index $1 \le \ell \le L$ and partition into

$$\mathcal{F}_s(x;\boldsymbol{\theta}_s) = f_\ell \circ \cdots \circ f_1, \qquad \mathcal{F}_h(x;\boldsymbol{\theta}_h) = f_L \circ \cdots \circ f_{\ell+1}.$$

We instantiate M independent heads $\mathcal{F}_h^{(i)}$ $(i = 1, \dots, M)$, each with separately initialized weights $\boldsymbol{\theta}_h^{(i)}$ and sparse mask $\mathbf{m}_h^{(i)}$. These unique initial conditions seed distinct "neural trails"—deep, long-range connectivity paths that give the multi-head network its diversity. The shared trunk $\mathcal{F}_s$ likewise carries a mask $\mathbf{m}_s$. We analyze the ideal backbone length ℓ in Section 5.5.1, and investigate the effect of different sparsity ratios S in Section 5.5.4. In the remainder of this chapter, we will denote the number of blocks in the backbone and heads by $|\mathcal{F}_s| = \ell$ and $|\mathcal{F}_h| = L - \ell$, respectively.

Training. On a minibatch (x, y), we compute each head's logits

$$\hat{y}^{(i)} = \mathcal{F}_h^{(i)}(F_s(x;\boldsymbol{\theta}_s);\boldsymbol{\theta}_h^{(i)}).$$

Individual losses $\mathcal{L}_i$ for each head i are calculated and averaged to form the composite loss,

$$\mathcal{L}(\Theta) = \frac{1}{M}\sum_{i=1}^{M} \mathcal{L}_i\big(\hat{y}^{(i)}, y\big), \quad \Theta = (\boldsymbol{\theta}_s, \boldsymbol{\theta}_h^{(1)}, \dots, \boldsymbol{\theta}_h^{(M)}),$$

which is used to update all active parameters through a masked version of stochastic gradient descent [RM51] or Adam [KB15]. Every ΔT steps, each component (shared or head i), performs a topology update through dynamic sparse training. This process consists of (1) layerwise pruning of p weights, and (2) reinitializing an equal number p previously inactive connections, thereby maintaining a constant density $\|\mathbf{m}\|_0/nk$ while exploring new sparse trails.

In reinforcement learning experiments, we prune and grow weights with the simplest methods: magnitude pruning $|\theta_i|$, and random growth (SET), shown to work well by Graesser et al. [GEEC22]. In computer vision, we reactivate weights with RigL and prune by standard magnitude $|\theta_i|$, as recommended by Nowak et al. [NGMT23]. We use the Erdős–Rényi (ER) approach [MMS+18, EGM+20] to distribute the global sparsity S into layerwise sparsity ratios. ER has been shown to yield

superior performance over simply setting each layer's sparsity to S, i.e., uniform sparsity [LCC+23]. In a nutshell, ER assigns higher sparsity ratios to larger layers.

For language modeling, we likewise use ER, but leave the essential attention projections dense while sparsifying all other layers. Furthermore, we also use RigL for growth, but we prune using *soft magnitude*, shown to work well for language models by Zhang et al. [ZZW+25]. In this procedure, a weight's absolute value determines a *probability* of being pruned, instead of simply pruning the smallest weights.

Dense models tend to overfit once training is prolonged, whereas sparse networks keep improving as they are still refining both weights and topology [LYMP21]. According to the schedules of Evci et al. [EGM+20], we extend the training of sparse variants by at most $1/(1-S)$, always keeping the total number of floating-point operations (FLOPs) for training below those of their dense counterparts. Exact number of environment steps, epochs, or updates appear in Section D.2.

Inference. During inference, the final prediction is computed through soft voting, averaging logits across all ensemble members:

$$\bar{y} = \frac{1}{M} \sum_{i=1}^{M} \mathcal{F}_h^{(i)}(F_s(x; \boldsymbol{\theta}_s); \boldsymbol{\theta}_h^{(i)}).$$

The shared backbone $F_s(x; \boldsymbol{\theta}_s)$ forward pass naturally only needs to be computed once. NeuroTrails ensures that while ensemble members share early feature extractors, the heads develop distinct predictive pathways through sparse connectivity patterns, thereby stimulating diversity.

5.4 EXPERIMENTS

We compare our methods against a single dense model, a full ensemble, and TreeNet [LPC+15] in all our experiments across RL, vision, and language. For computer vision, we can furthermore benchmark against performance values published for various other state-of-the-art efficient ensemble methods, including MIMO [HJF+21], Batch Ensemble [WTB20], as well as DST and EDST ensembles [LCA+22]. See Section 5.6 for detailed descriptions of these baselines. All architectures use the following base models: DQN on Atari (RL), Wide-ResNet28-10 on CIFAR-100 (vision), ResNet-50 on ImageNet (vision), and LLaMA-130M/350M on C4 (language). Details on the training regime and hyperparameters are shared in Section D.2.

Table 5.2. Reinforcement learning return (↑) on ten Atari environments with DQN as the base model. We train for 10M env steps and report IQM ± s.e.m. over 8 seeds, following [ASC+21].

	Asterix	BeamRider	Breakout	Hero	Jamesbond
Single Dense	3200.7 ±368.9	4201.7 ±77.1	126.2 ±71.7	1204.2 ±214.4	353.1 ±53.3
TreeNet (M=3)	3010.1 ±461.4	4988.9 ±35.9	268.0 ±12.4	1590.5 ±108.0	172.0 ±15.7
Full Ensemble (M=3)	4698.9 ±403.5	5072.3 ±56.5	223.8 ±3.2	1503.5 ±195.7	267.5 ±47.6
NeuroTrails (M=3, S=0.8)	**6058.5 ±142.8**	**5742.2 ±130.4**	**284.6 ±6.9**	**6659.5 ±1277.9**	**443.2 ±9.2**

	Kangaroo	Krull	Seaquest	SpaceInvaders	UpNDown
Single Dense	310.5 ±16.6	6685.8 ±57.4	605.0 ±124.4	**632.0 ±36.1**	6202.7 ±224.3
TreeNet (M=3)	1198.4 ±765.7	7448.9 ±97.8	123.5 ±6.6	478.9 ±42.4	**7241.6 ±471.3**
Full Ensemble (M=3)	2543.9 ±418.4	7091.2 ±75.5	263.7 ±83.7	539.1 ±63.7	6335.1 ±149.2
NeuroTrails (M=3, S=0.8)	**6318.0 ±118.9**	**7801.6 ±103.7**	**2331.9 ±108.2**	626.7 ±131.6	6583.9 ±292.8

In reinforcement learning experiments, we report the interquartile mean (IQM) evaluation return over the last 10% of training, as done in [GEEC22, GSD+23]. For computer vision, we present the mean test accuracy, negative log-likelihood (NLL), and expected calibration error (ECE). In language modeling, our main metric is perplexity on the C4 validation set. Next to the name of the model, we indicate the ensemble size (or number of heads) M and sparsity ratio S.

5.4.1 Reinforcement Learning

We take a standard Deep Q-Network (DQN) [MKS+13] and adjust its architecture to either NeuroTrails, TreeNet, or a Full Ensemble. We ensure that each head is trained independently, but actions in the environment are taken jointly by the ensemble. In other words, we select actions after averaging Q-values across heads, meaning heads jointly take decisions, but are independently trained on data from the replay buffer. We train for 10M steps (40M frames) on ten Atari environments [BNVB13], and report the IQM over 8 random seeds.

As shown in Table 5.2, NeuroTrails performs well across the environments, even with a sparsity level of 80%. It outperforms a full ensemble, TreeNet, and a single DQN in 8/10 cases, while also achieving decent return in the remaining two. In Appendix D.2, we demonstrate the remarkably low number of FLOPs required to train NeuroTrails:

Table 5.3. Performance on **CIFAR-100** with Wide-ResNet28-10 as the base. NeuroTrails and TreeNet have 8 blocks in the heads, with 4 remaining blocks in the shared backbone. Results marked with * are from [HJF+21], ** from [LCA+22], and *** from [LL24].

	Accuracy (↑)	NLL (↓)	ECE (↓)	Train FLOPs (↓)	Infer. FLOPs (↓)
Single Dense *	79.8	0.875	0.086	3.6e17	10.5e9
MIMO (M=3) *	82.0	0.690	0.022	1.00×	1.00×
EDST Ensemble (M=7, S=0.9) **	82.6	0.653	0.036	**0.57×**	1.17×
DST Ensemble (M=3, S=0.8) **	83.3	**0.623**	**0.018**	1.01×	1.01×
Batch Ensemble (M=4) *	81.5	0.740	0.056	1.10×	1.10×
NFE (M=3) ***	83.5	0.658	0.061	1.02×	1.02×
TreeNet (M=3)	83.2	0.673	0.052	2.91×	2.91×
Full Ensemble (M=3)	83.3	0.663	0.042	3.00×	3.00×
NeuroTrails (M=3, S=0.8)	83.8	0.681	0.044	0.85×	0.47×
NeuroTrails (M=5, S=0.9)	**83.9**	0.675	0.041	0.67×	**0.37×**

approximately 0.59× the number of a single model (while a full ensemble uses 3×).

5.4.2 Computer Vision

For computer vision experiments we are able to compare against more baselines, since ensembling methods have mostly been tested on image classification benchmarks only, in their original works. As shown in Tables 5.3 and 5.4, NeuroTrails demonstrates strong performance both on CIFAR-100 and ImageNet, while using significantly fewer FLOPs at inference time. The low FLOPs required at inference are crucial, making NeuroTrails a compelling choice for deployment in resource-constrained environments. See Section 5.5.5 for the real-time speedups that are directly available.

Robustness against Corruptions. To test NeuroTrails for its zero-shot generalization capability, we evaluate its robustness on ImageNet-C, a dataset of corrupted ImageNet samples with various severity levels [HD19]. Furthermore, we test on ImageNet-Sketch [WGLX19], a collection of black-and-white sketched illustrations, assessing the model's ability to extrapolate to out-of-domain (OOD) data. The results in Figure 5.2 show that NeuroTrails consistently outperforms the full ensemble across all severity levels and tasks, while requiring a fraction of its total inference FLOPs.

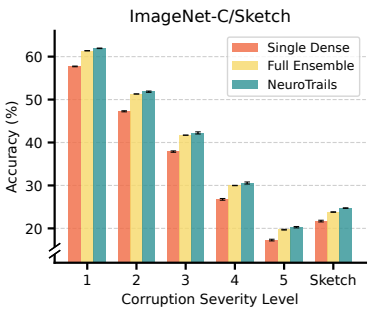


Figure 5.2. Zero-shot generalization.

Table 5.4. Performance on **ImageNet** with ResNet-50 as the baseline model. NeuroTrails and TreeNet have 10 blocks in their multi-head structure, with 6 remaining blocks in the shared backbone. Results marked with * are from [HJF+21] and ** from [LCA+22].

	Accuracy (↑)	NLL (↓)	ECE (↓)	Train FLOPs (↓)	Infer. FLOPs (↓)
Single Dense *	76.1	0.943	0.039	4.8e18	8.2e9
MIMO (M=2, ρ = 0.6) *	77.5	0.887	0.037	1.00×	1.00×
EDST Ensemble (M=4, S=0.8) **	77.7	0.935	0.064	**0.87×**	1.78×
DST Ensemble (M=2, S=0.8) **	**78.3**	0.914	0.060	1.12×	1.12×
Batch Ensemble (M=4) *	76.7	0.944	0.049	1.10×	1.10×
TreeNet (M=3)	78.1	0.886	0.053	2.91×	2.91×
Full Ensemble (M=4) *	77.5	0.877	**0.031**	4.00×	4.00×
NeuroTrails (M=3, S=0.7)	78.1	**0.861**	0.038	1.10×	**0.67×**

Table 5.5. Pretraining on the **C4** dataset with LLaMA-130M/350M as the baseline model. NeuroTrails and TreeNet use 2/3 of the transformer blocks in the heads, with 1/3 in the backbone.

	Perplexity (↓)	Training FLOPs (↓)	Inference FLOPs (↓)
LLaMA-130M			
Single Dense	29.06	3.5e18	2.2e11
TreeNet (M=3)	26.46	**2.21×**	2.21×
Full Ensemble (M=3)	26.88	3.00×	3.00×
NeuroTrails (M=3, S=0.1)	**26.00**	**2.21×**	**1.99×**
LLaMA-350M			
Single Dense	22.80	4.2e19	6.9e11
TreeNet (M=3)	21.06	**2.27×**	2.27×
Full Ensemble (M=3)	21.25	3.00×	3.00×
NeuroTrails (M=3, S=0.1)	**20.70**	**2.27×**	**2.04×**

5.4.3 Language Modeling

We pretrain variants of LLaMA-130M and LLaMA-350M on the *Colossal Clean Crawled Corpus* [RSR+20, C4]. Motivated by the work of Wu et al. [WXW+25], we use a low sparsity ratio in these experiments, but maintain the adaptive nature of dynamic sparse training. The results in Table 5.5 show that NeuroTrails performs strongly on transformer architectures, achieving the best validation perplexity. Despite using a lower sparsity ratio in the language domain, our algorithm yields a lightweight model with lower inference FLOPs than both TreeNet and the full ensemble.

Evaluation on Downstream Tasks. We test our pretrained LLAMA-350M models for zero-shot generalization to multiple downstream tasks. The results in Table 5.6 compare model accuracy across seven benchmarks: MMLU [HBB+21], BoolQ [CLC+19], ARC [CCE+18], PIQA [BZB+19], Hellaswag [ZHB+19], OpenbookQA [MCKS18], and WinoGrande [SBBC19]. These tasks span multiple domains including common sense

Table 5.6. Zero-shot accuracy (↑) of LLaMA-350M models across seven downstream tasks.

	MMLU	BoolQ	ARC	PIQA	Hellaswag	OBQA	WinoGrande	**Avg.**
Single Dense	22.92	58.47	40.24	62.51	28.31	13.60	**52.49**	39.79
TreeNet (M=3)	**22.97**	58.65	40.40	62.95	**28.45**	15.00	51.30	39.96
Full Ensemble (M=3)	**22.97**	58.23	40.36	62.68	28.18	14.40	51.70	39.78
NeuroTrails (M=3, S=0.1)	22.92	**60.49**	**41.71**	**63.28**	28.43	**15.80**	50.51	**40.45**

reasoning, multiple choice question answering, and scientific knowledge. NeuroTrails achieves the highest average accuracy, suggesting that it offers improved generalization and robustness across a wide range of language tasks.

5.5 ANALYSIS

In this section, we explore various design choices for NeuroTrails. All results reported here were obtained using Wide-ResNet28-10 on CIFAR-100, and present the mean and standard deviation over 3 independent seeds.

5.5.1 Backbone Length

An essential hyperparameter of NeuroTrails is the optimal split index l. The ideal architecture may depend on both the sparsity ratio S and the number of heads M; our analysis focuses on the configuration with 80% sparsity and 3 heads. In addition, we examine different sparsification methods on the same plot. As detailed in Section 5.3, we split the architecture between blocks, where each block in Wide-ResNet28-10 consists of two convolutional layers, two batch normalization layers and a residual connection. The base network Wide-ResNet28-10 has 12 blocks in total, so we can vary the backbone length across this depth. The results shown in Figure 5.3 reveal that performance is maximized most efficiently with a split point at 8 blocks per head. This architecture consists of four shared backbone blocks

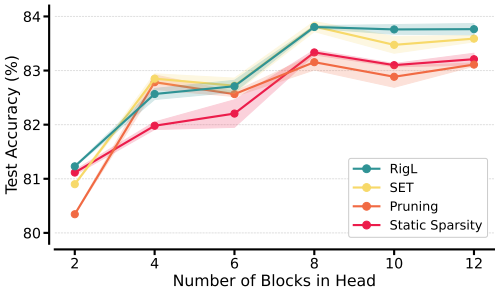


Figure 5.3. NeuroTrails models with varying backbone sizes and sparsification methods (on CIFAR-100 with Wide-ResNet28-10). **Backbone Length:** The most *effective* (optimizing accuracy and efficiency) backbone length appears around 1/3 of the network, meaning 8/12 blocks in head. **Sparsification:** RigL and SET demonstrate superior performance, confirming dynamic sparsity as the optimal approach.

Table 5.7. Prediction disagreement (PD) and test accuracy on CIFAR-100. NeuroTrails achieves peak accuracy at $|\mathcal{F}_h| = 8$, with lower PD than configurations using more head blocks, suggesting that optimal performance lies in a *Goldilocks zone* where PD is neither too low nor too high.

Blocks in head	PD (%)	Accuracy (%)
2	2.9 ± 0.17	80.89 ± 0.01
4	11.2 ± 0.41	82.85 ± 0.09
6	12.4 ± 0.28	82.71 ± 0.14
8	14.6 ± 0.36	**83.81 ± 0.10**
10	15.3 ± 0.12	83.47 ± 0.16
12	16.0 ± 0.06	83.59 ± 0.08
Full Ensemble	15.4 ± 0.34	83.62 ± 0.10

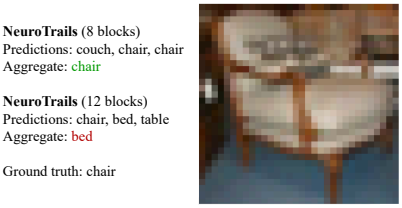


Figure 5.4. Illustration of a CIFAR-100 test-set image where too much prediction diversity between heads degrades performance. NeuroTrails with 8 blocks in each head seems to get the amount of diversity *just right* for optimal performance.

($|\mathcal{F}_s| = 12 - 8 = 4$) and eight blocks for each of the independent heads ($|\mathcal{F}_h| = 8$), resulting in approximately one-third of the network serving as the shared backbone. The different sparsification methods used have varying performance. However, the dynamic nature of RigL and SET helps them to consistently surpass static sparse training and standard one-time pruning.

5.5.2 Prediction Diversity

We analyze the effect of different NeuroTrails settings on the prediction diversity and its performance. Although numerous metrics exist for quantifying diversity [KW03], we adopt prediction disagreement (PD), one of the most widely used. PD is defined as the proportion of test samples where ensemble members produce conflicting predictions [Ska96].

Analysis of PD patterns in Table 5.7 reveals a monotonic increase in inter-head disagreement as the proportion of the NeuroTrails architecture allocated to independent heads grows. This observation aligns with intuition: as a larger portion of the network is dedicated to the heads, the extra head-only layers let each branch specialize, drifting their outputs further from the initially shared representation.

More diversity does not necessarily lead to better overall predictions. For instance, our model with $|\mathcal{F}_h| = 8$ exhibits *lower* prediction disagreement between heads (14.6) compared to a full ensemble (15.4) and configurations with more blocks in the heads (up to 16.0), while being superior in performance (83.81% accuracy).

This observation points to the existence of an optimal disagreement interval, which we refer to as the *PD Goldilocks zone* (due to the amount being 'just right').

Beyond the zone's upper threshold, excessive prediction diversity among ensemble members begins to degrade model performance. When heads make significantly divergent predictions for the same input, they cease to complement each other and instead compete. While a certain level of diversity is beneficial in ensemble learning, excessive diversity can be detrimental, see Figure 5.4. Achieving the right balance between diversity and consensus is essential to maximize performance.

5.5.3 Ensemble Size

We analyze the impact of ensemble size M on performance and efficiency. Single networks and standard ensembles are fully dense, while NeuroTrails uses 80% sparsity and $|\mathcal{F}_h| = 8$ across all experiments in this section. The results are summarized in Figure 5.5. Both traditional ensembles and NeuroTrails show significant accuracy gains as the ensemble size increases from 1 to 10, with NeuroTrails consistently outperforming the baselines across all sizes. The most substantial improvements occur between sizes 1 and 3, followed by diminishing returns as size grows.

The trade-offs are further illustrated in the right plot of Figure 5.5. Due to its high sparsity, NeuroTrails incurs significantly lower computational costs, scaling more efficiently with ensemble size. These gains could be further amplified—e.g., a 5-head NeuroTrails can support 90% sparsity without a drop in performance on CIFAR-100 (see Table 5.3)—while larger ensembles may enable even greater sparsity. Exploring such configurations is a promising avenue for future work.

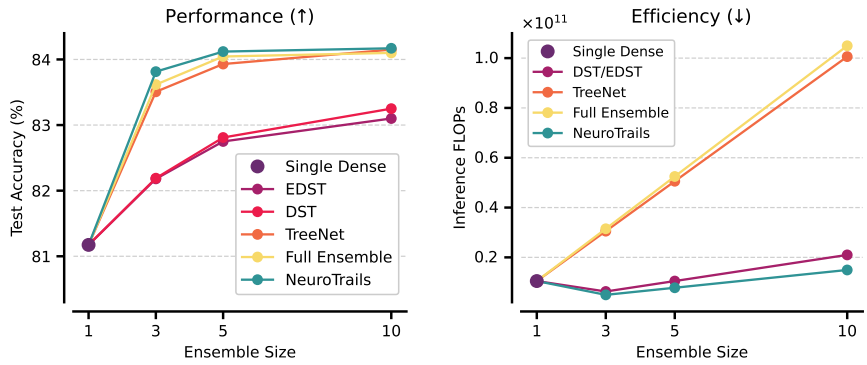


Figure 5.5. Effect of the ensemble size on CIFAR-100. NeuroTrails achieves higher accuracy than a full ensemble (**left**) while consuming only a fraction of the FLOPs (**right**).

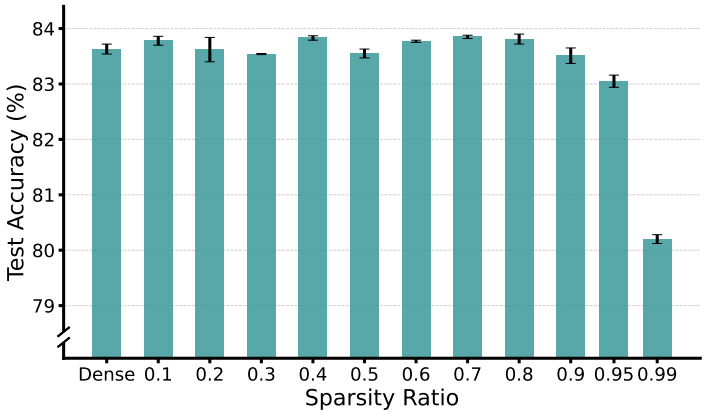


Figure 5.6. Impact of sparsity ratio on accuracy in a NeuroTrails model with three heads (M=3, Wide-ResNet28-10) on CIFAR-100. The 80% sparse configuration emerges as the optimal choice, though the performance differences across the closest competitors are notably small.

5.5.4 Sparsity Ratio Analysis

This section examines the relationship between sparsity ratios and the accuracy of the model. To ensure a controlled analysis, we fix the ensemble size at 3, set the backbone sharing factor to 8, and vary the sparsity ratio from fully dense, which corresponds to being 0% sparse, to 99% sparse.

Our results indicate that for CIFAR-100, an optimal trade-off for efficiency and performance is achieved at 80% sparsity, where the model retains only 20% of its original parameters (Figure 5.6). Interestingly, the dense model performs worse, likely due to overparameterization. The overall accuracy across sparsity ratios remains relatively stable up to a certain point (80–90%) after which performance decays.

5.5.5 Inference Speedup

While FLOPs reduction is a widely used proxy for model efficiency, achieving real-world speedups often hinges on hardware compatibility and software execution paths. Recent hardware advances, such as the Cerebras CS-2 system, have shown that unstructured sparsity can translate into substantial runtime performance gains, even on GPU-class accelerators [Cer24].

In parallel, software frameworks such as DeepSparse already deliver substantial inference-time speedups on commodity CPU hardware [Neu21]. In our experiments with CIFAR-100 and M=3, we observe that NeuroTrails models significantly outper-

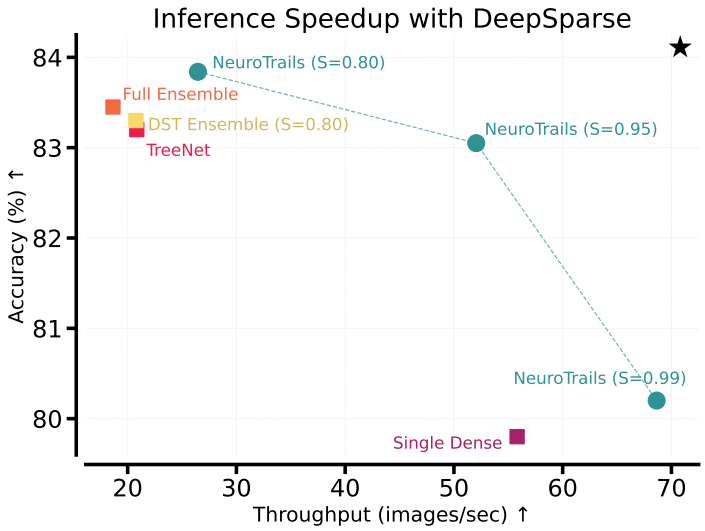


Figure 5.7. NeuroTrails forms the Pareto front of efficiency and performance on CPU hardware with DeepSparse. Measuring Wide-ResNet28-10 on CIFAR-100 with $M = 3$.

form full ensembles in terms of practical efficiency, see Figure 5.7. For example, NeuroTrails (S=0.95) achieves a throughput of 53.16 images per second, similar to a single dense model, while achieving much higher accuracy. These results position NeuroTrails on the Pareto front of efficiency and performance, giving a compelling solution for deployment on smartphones or other edge devices, where resources are constrained and GPUs are often unavailable. Software frameworks for unstructured sparsity on GPUs are likewise on the horizon.

5.6 RELATED WORK

In deep learning, multiple attempts have been made to achieve ensemble-level performance while attaining significant reductions in parameter count and FLOPs. Batch Ensemble [WTB20] introduced an efficient ensemble approach by decomposing the ensemble members into a shared matrix and rank-one personalized matrices, achieving near-single network computational costs. Multi-Input Multi-Output Ensembles (MIMO) [HJF+21] subsequently improved on this method by ensembling only input and output layers, demonstrating enhanced performance across ensemble architectures. In MIMO, the full original network is always used as the main structure, while

adding heads as additional layers at the input and output ends. NeuroTrails differs in this regard, as it only splits into heads on the output side. Furthermore, NeuroTrails has the ability to flexibly configure where our backbone splits into multiple heads, not needing to keep the full original network intact.

In TreeNets, Lee et al. [LPC+15] propose sharing early layers for ensembles. We enhance this approach with two major components: (1) by incorporating dynamic sparse training, which fosters greater diversity through sparse neural pathways and significantly reduces the number of parameters and FLOPs required; and (2) by splitting the backbone based on layer-based blocks rather than individual layers, preserving the structural integrity inherent in widely-used architectures such as ResNets [HZRS16] and Transformers [VSP+17].

Liu et al. [LCA+22] use dynamic sparse training for ensembles, but do not use a multi-headed network structure. In the DST ensemble approach, independent sparse neural networks are trained from scratch, while their Efficient-DST (EDST) method creates an ensemble from a single network by using distinct model checkpoints throughout training.

5.7 CONCLUSION

We propose **NeuroTrails**, a novel training paradigm that is straightforward to integrate into various neural network architectures. The methodology splits a network into multiple sparse heads, optimizing their topology through dynamic sparse training. Extensive experiments demonstrate significant improvements across RL, supervised and self-supervised learning settings, alongside lower inference FLOPs and practical CPU throughput gains. NeuroTrails reveals that ensembling all layers is not a necessary condition to achieve optimal performance. Early-stage representation learning is more effectively handled through a single sparse backbone.

Our analysis highlights a *Goldilocks* zone of prediction disagreement: too little diversity wastes ensemble benefits, too much disrupts aggregation. The backbone-to-heads splitting point provides a simple, general knob to repeatedly place models near this sweet spot.

§

CHAPTER 6
Conclusion

In this thesis, I have presented multiple ways to increase the adaptability of reinforcement learning agents. Let us now end with some concluding remarks on the use and limitations of the different methods, and an outlook on future work.

6.1 OVERVIEW AND FUTURE WORK

The adaptability of agents can manifest itself in many forms: from optimizing their internal neural network topology in Chapters 2 and 5, to handling visual distractions or distinct contexts through specific modules in Chapters 3 and 4. In each of these algorithms, there seems to be room for further improvement.

The *Automatic Noise Filtering* (ANF) method of Chapter 2 provides a way for reinforcement learning agents to learn in settings with extreme amounts of input noise. It is shown to be able to select the most task-relevant features, while learning the task simultaneously. I love that we were able to find another great use of sparse neural networks, besides the standard increase in parameter efficiency. For future research, there are many remaining questions on the effectiveness of ANF on different environments, real robots, or other related fields, such as reinforcement learning from human feedback (RLHF). We showed promising results of ANF-like methods in RLHF in a follow-up work [MGM+25], where we sparsified the input layers of reward models. Furthermore, I see opportunities in the area of multi-task reinforcement learning. Perhaps it is possible to learn multiple sparse parameter-masks at once, each of which selects the subset of relevant features for its corresponding task. This might enable greater generality within a single neural network.

The extension of ANF to vision input was challenging, but eventually led to a different approach to *Mask Distractions* (MaDi). In Chapter 3 we explained that the relevancy of each pixel can rapidly change from one timestep to the next, meaning

that the evolutionary search of dynamic sparse training is too slow for such cases.[1] We devised a method that can apply different masks at each timestep, learned directly from loss functions already present in reinforcement learning. The fact that this worked end-to-end still excites me. The community has already built algorithms loosely inspired by MaDi, through the use of vision-language models [HRM25] or by including a Kullback–Leibler divergence loss [STZ+25]. Potential future directions could be to investigate other masker network architectures, or masking relations between objects within a self-attention layer, instead of only masking objects directly.

Dealing with out-of-distribution environment changes that are irrelevant to the task is useful, as we have demonstrated in the first two chapters. However, handling new settings where task-relevant features are significantly adjusted is even more challenging. In Chapter 4 we showed that our *Single-Phase Adaptation* algorithm is able to generalize well in this problem setting. SPARC can drive over 100 unseen vehicles successfully around multiple tracks, and perform proper locomotion in extreme wind conditions. The fact that it updates both the expert and adapter policies simultaneously, truly simplifies implementation and training. Furthermore, this eases the application of continual learning, which is an interesting direction for future work. And of course, experimenting with SPARC on physical robots is important open research, to see if our algorithm can deal with the sim-to-real gap as well as it handles unseen contexts in simulation.

Finally, adapting to a changed environment often comes with increased uncertainty in the agent's predicted optimal actions. One way to reduce this uncertainty is using the power of ensembles. Combining the predictions of multiple experts can improve the adaptation performance to unseen situations. Chapter 5 discusses *NeuroTrails*, an effective and efficient ensembling method with a high level of internal adaptability. We have demonstrated its efficacy across reinforcement learning, computer vision, and language modeling. Especially in fields with discrete choices, such as Atari actions or image classes, ensembling seems to work well. Whether this multi-head averaging approach can increase performance in continuous control settings, remains an open question.

[1] The quote by Turing at the start of this book should have warned us.

6.2 LIMITATIONS AND SOCIAL IMPACT

The research presented in this thesis aims to advance the capabilities of artificial intelligence. In applications of AI, it is essential to consider the ethical impact of the deployed algorithms, to avoid harm in the present and the future.

When applying the methods presented in this work, take the following considerations into account. Each method is designed with the goal of being adaptable to certain changes in the environment or data. We demonstrate that our methods are able to generalize well to unseen settings or contexts. However, there is no strict guarantee that our methods will continue to work well outside of the domains we tested. It remains important to verify a robot's performance in as many different settings as possible, before deploying it in the real world.

Some methods contain specific elements that are vital to consider before deployment. For instance, the masking approach in Chapter 3 is helpful in situations we experimented on, but could be unsafe to deploy in autonomous driving if it masks too many pixels. An adjustment to the methodology, where the full input image is provided alongside the masked input, could perhaps be a way to mitigate this issue.

6.3 AFTERWORD

Recently, I became a father. I often find myself wondering whether my son will reach general intelligence before or after we have created the artificial version. Next to the scientifically thorough development of AI, I want to emphasize that it is crucial that we get AI safety and alignment right. Preferably the first time, as we do not know whether there will be a second. We need to determine what values and norms we want AI to have, and how to endow our agents with these human perspectives. Please join this discussion, which could be "the most important conversation of our time"—as Max Tegmark puts it in *Life 3.0*, a book I highly recommend. The generality, adaptability, and learning ability of agents continue to increase, thus it seems only a matter of time before we reach artificial general intelligence.

§

Bibliography

[AAC+19] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving Rubik's Cube with a Robot Hand. *arXiv preprint arXiv:1910.07113*, 2019.

[Acu12] Kevin Acunto. *Feature selection for scalable reinforcement learning*. PhD thesis, State University of New York at Binghamton, 2012.

[AMCB20] Rishabh Agarwal, Marlos C Machado, Pablo Samuel Castro, and Marc G Bellemare. Contrastive Behavioral Similarity Embeddings for Generalization in Reinforcement Learning. In *International Conference on Learning Representations*, 2020.

[AOPP21] Samin Yeasar Arnob, Riyasat Ohib, Sergey Plis, and Doina Precup. Single-Shot Pruning for Offline Reinforcement Learning. *arXiv preprint arXiv:2112.15579*, 2021.

[ASC+21] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep Reinforcement Learning at the Edge of the Statistical Precipice. *Advances in Neural Information Processing Systems*, 34:29304–29320, 2021.

[ASvdL+22] Zahra Atashgahi, Ghada Sokar, Tim van der Lee, Elena Mocanu, Decebal Constantin Mocanu, Raymond Veldhuis, and Mykola Pechenizkiy. Quick and Robust Feature Selection: the Strength of Energy-efficient Sparse Training for Autoencoders. *Machine Learning*, 111(1):377–414, 2022.

[AYN14] Yasin Abbasi-Yadkori and Gergely Neu. Online learning in MDPs with side information. *arXiv preprint arXiv:1406.6812*, 2014.

[BAP+21] Nicolò Botteghi, Khaled Alaa, Mannes Poel, Beril Sirmacek, Christoph Brune, Abeje Mersha, and Stefano Stramigioli. Low Dimensional State Representation Learning with Robotics Priors in Continuous Action Spaces. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 190–197. IEEE, 2021.

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

[Bem22] Wouter van den Bemd. Robust Deep Reinforcement Learning for Greenhouse Control and Crop Yield Optimization. Master's thesis, Eindhoven University of Technology, 2022.

[BES+21] Carolin Benjamins, Theresa Eimer, Frederik Schubert, André Biedenkapp, Bodo Rosenhahn, Frank Hutter, and Marius Lindauer. CARL: A Benchmark for Contextual and Adaptive Reinforcement Learning. *Eco. Theory RL, NeurIPS*, 2021.

[BJK+23] Michael Beukman, Devon Jarvis, Richard Klein, Steven James, and Benjamin Rosman. Dynamics Generalisation in Reinforcement Learning via Adaptive Context-Aware Policies. *Neural Information Processing Systems*, 2023.

[BJVW23] Jacob Beck, Matthew Thomas Jackson, Risto Vuorio, and Shimon Whiteson. Hypernetworks in Meta-Reinforcement Learning. In *Conference on Robot Learning*, pages 1478–1487. PMLR, 2023.

[BKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer Normalization. *Advances in Neural Information Processing Systems, Deep Learning Symposium*, 2016.

[BKML18] Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. Deep Rewiring: Training very sparse deep networks. *International Conference on Learning Representations*, 2018.

[BM13] Julian Bishop and Risto Miikkulainen. Evolutionary Feature Evaluation for Online Reinforcement Learning. In *2013 IEEE Conference on Computational Intelligence in Games (CIG)*, pages 1–8, 2013.

[BNVB13] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[Bre96] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 8 1996.

[ByRCS+69] Paul Bach-y Rita, Carter C Collins, Frank A Saunders, Benjamin White, and Lawrence Scadden. Vision Substitution by Tactile Image Projection. *Nature*, 221(5184):963–964, 1969.

[BZB+19] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about Physical Commonsense in Natural Language. *CoRR*, abs/1911.11641, 2019.

[BZZR22] David Bertoin, Adil Zouitine, Mehdi Zouitine, and Emmanuel Rachelson. Look where you look! Saliency-guided Q-networks for generalization in visual Reinforcement Learning. *Advances in Neural Information Processing Systems*, 35:30693–30706, 2022.

[CBA+16] William Curran, Tim Brys, David Aha, Matthew Taylor, and William D Smart. Dimensionality Reduced Reinforcement Learning for Assistive Robots. In *2016 AAAI Fall Symposium Series*, 2016.

[CCE+18] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *CoRR*, abs/1803.05457, 2018.

[CCG+21] Tianlong Chen, Yu Cheng, Zhe Gan, Lu Yuan, Lei Zhang, and Zhangyang Wang. Chasing Sparsity in Vision Transformers: An End-to-End Exploration. *Advances in Neural Information Processing Systems*, 34, 2021.

[Cer24] Cerebras. Sparsity Made Easy – Introducing the Cerebras PyTorch Sparsity Library - Cerebras, 2024. URL: https://www.cerebras.ai/blog/sparsity-made-easy-introducing-the-cerebras-pytorch-sparsity-library [Accessed: 2025.09.12].

[CHHS20] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging Procedural Generation to Benchmark Reinforcement Learning. In *Int. Conf. on Machine Learning*, 2020.

[CKH+19] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1282–1289. PMLR, 09–15 Jun 2019.

[CLC+19] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions. *CoRR*, abs/1905.10044, 2019.

[CLM20] Elisa Castaldi, Claudia Lunghi, and Maria Concetta Morrone. Neuroplasticity in adult human visual cortex. *Neuroscience & Biobehavioral Reviews*, 112:542–552, 2020.

[CLZ+21] Baiming Chen, Zuxin Liu, Jiacheng Zhu, Mengdi Xu, Wenhao Ding, Liang Li, and Ding Zhao. Context-Aware Safe Reinforcement Learning for Non-Stationary Environments. In *Int. Conf. on Robotics and Automation*. IEEE, 2021.

[Col23] Open X-Embodiment Collaboration. Open X-Embodiment: Robotic learning datasets and RT-X models, 2023. Source URL: https://robotics-transformer-x.github.io.

[CZW+22] Tianlong Chen, Zhenyu Zhang, Pengjun Wang, Santosh Balachandra, Haoyu Ma, Zehao Wang, and Zhangyang Wang. Sparsity Winning Twice: Better Robust Generalization from More Efficient Training. *International Conference on Machine Learning*, 2022.

[DBK+21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *International Conference on Learning Representations*, 2021.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009.

[DFB+22] Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

[Die00] Thomas G. Dietterich. Ensemble Methods in Machine Learning. In *Multiple Classifier Systems*, pages 1–15. Springer Berlin Heidelberg, 2000.

[DSC+16] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning. *arXiv preprint arXiv:1611.02779*, 2016.

[DSM21] Shibhansh Dohare, Richard S. Sutton, and A. Rupam Mahmood. Continual Backprop: Stochastic Gradient Descent with Persistent Randomness. *arXiv preprint arXiv:2108.06325*, 2021.

[DZ19] Tim Dettmers and Luke Zettlemoyer. Sparse Networks from Scratch: Faster Training without Losing Performance. *arXiv preprint arXiv:1907.04840*, 2019.

[EGM+20] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the Lottery: Making All Tickets Winners. In *International Conference on Machine Learning*, pages 2943–2952. PMLR, 2020.

[FAL17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[FC19] Jonathan Frankle and Michael Carbin. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. *International Conference on Learning Representations*, 2019.

[FHL20] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep Ensembles: A Loss Landscape Perspective, 2020.

[FHM18] Scott Fujimoto, Herke Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018.

[FMB18] Jesse Farebrother, Marlos C Machado, and Michael Bowling. Generalization and Regularization in DQN. *NeurIPS'18 Deep Reinforcement Learning Workshop*, 2018.

[FPP11] Norm Ferns, Prakash Panangaden, and Doina Precup. Bisimulation metrics for continuous markov decision processes. *SIAM Journal on Computing*, 40(6):1662–1714, 2011.

[FS97] Yoav Freund and Robert E Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[FTD+16] Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Deep Spatial Autoencoders for Visuomotor Learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 512–519. IEEE, 2016.

[FWH+21] Linxi Fan, Guanzhi Wang, De-An Huang, Zhiding Yu, Li Fei-Fei, Yuke Zhu, and Anima Anandkumar. SECANT: Self-Expert Cloning for Zero-Shot Generalization of Visual Policies. *International Conference on Machine Learning*, 2021.

[GBEB17] Heitor Murilo Gomes, Jean Paul Barddal, Fabrício Enembreck, and Albert Bifet. A Survey on Ensemble Learning for Data Stream Classification. *ACM Computing Surveys*, 50(2):1–36, March 2017.

[GEEC22] Laura Graesser, Utku Evci, Erich Elsen, and Pablo Samuel Castro. The State of Sparse Training in Deep Reinforcement Learning. In *International Conference on Machine Learning*, pages 7766–7792. PMLR, 2022.

[GEG+21] Linus Gisslén, Andy Eakins, Camilo Gordillo, Joakim Bergdahl, and Konrad Tollmar. Adversarial Reinforcement Learning for Procedural Content Generation. In *2021 IEEE Conference on Games (CoG)*. IEEE, 2021.

[GHZ+26] Bram Grooten, Farid Hasanov, Chenxiang Zhang, Qiao Xiao, Boqian Wu, Zahra Atashgahi, Ghada Sokar, Shiwei Liu, Lu Yin, Elena Mocanu, Mykola Pechenizkiy, and Decebal Constantin Mocanu. Training with Dynamic Sparse Heads as the Key to Effective Ensembling. *Under review*, 2026.

[GMS+26] Bram Grooten, Patrick MacAlpine, Kaushik Subramanian, Peter Stone, and Peter R. Wurman. Out-of-Distribution Generalization with a SPARC: Racing 100 Unseen Vehicles with a Single Policy. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026.

[Gro24] Bram Grooten. Large Learning Agents: Towards Continually Aligned Robots with Scale in RL. In *Doctoral Consortium at Autonomous Agents and Multiagent Systems*, 2024.

[GSD+23] Bram Grooten, Ghada Sokar, Shibhansh Dohare, Elena Mocanu, Matthew E. Taylor, Mykola Pechenizkiy, and Decebal Constantin Mocanu. Automatic Noise Filtering with Dynamic Sparse Training in Deep Reinforcement Learning. *Int. Conf. Autonomous Agents and Multiagent Systems (AAMAS)*, 2023.

[GTV+24] Bram Grooten, Tristan Tomilin, Gautham Vasan, Matthew E Taylor, A Rupam Mahmood, Meng Fang, Mykola Pechenizkiy, and Decebal Constantin Mocanu. MaDi: Learning to Mask Distractions for Generalization in Visual Deep Reinforcement Learning. *Int. Conf. Autonomous Agents and Multiagent Systems (AAMAS)*, 2024.

[HBB+21] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding. *International Conference for Learning Representations*, 2021.

[HD19] Dan Hendrycks and Thomas Dietterich. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. In *International Conference on Learning Representations*, 2019.

[HDCM15] Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual Markov Decision Processes. *arXiv preprint arXiv:1502.02259*, 2015.

[HDY+22] Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL: http://jmlr.org/papers/v23/21-1342.html.

[HG16] Dan Hendrycks and Kevin Gimpel. Gaussian Error Linear Units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

[HJF+21] Marton Havasi, Rodolphe Jenatton, Stanislav Fort, Jeremiah Zhe Liu, Jasper Snoek, Balaji Lakshminarayanan, Andrew Mingbo Dai, and Dustin Tran. Training independent subnetworks for robust prediction. In *International Conference on Learning Representations*, 2021.

[HJS+20] Nicklas Hansen, Rishabh Jangir, Yu Sun, Guillem Alenyà, Pieter Abbeel, Alexei A Efros, Lerrel Pinto, and Xiaolong Wang. Self-Supervised Policy Adaptation during Deployment. *International Conference on Learning Representations*, 2020.

[HML+23] Tuomas Haarnoja, Ben Moran, Guy Lever, Sandy H Huang, Dhruva Tirumala, Markus Wulfmeier, Jan Humplik, Saran Tunyasuvunakool, Noah Y Siegel, Roland Hafner, et al. Learning Agile Soccer Skills for a Bipedal Robot with Deep Reinforcement Learning. *arXiv preprint arXiv:2304.13653*, 2023.

[HPTD15] Song Han, Jeff Pool, John Tran, and William Dally. Learning both Weights and Connections for Efficient Neural Networks. *Advances in Neural Information Processing Systems*, 28, 2015.

[HPZ+22] Yangru Huang, Peixi Peng, Yifan Zhao, Guangyao Chen, and Yonghong Tian. Spectrum Random Masking for Generalization in Image-based Reinforcement Learning. *Neural Information Processing Systems*, 35, 2022.

[HRM25] Asher J Hancock, Allen Z Ren, and Anirudha Majumdar. Run-time Observation Interventions Make Vision-Language-Action Models More Visually Robust. In *International Conference on Robotics and Automation (ICRA)*, pages 9499–9506. IEEE, 2025.

[HS90] Lars Kai Hansen and Peter Salamon. Neural Network Ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

[HSW21] Nicklas Hansen, Hao Su, and Xiaolong Wang. Stabilizing Deep Q-Learning with ConvNets and Vision Transformers under Data Augmentation. *35th Conference on Neural Information Processing Systems*, 2021.

[HVD15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. *Neural Information Processing Systems (Deep Learning Workshop)*, 2015.

[HW21] Nicklas Hansen and Xiaolong Wang. Generalization in Reinforcement Learning by Soft Data Augmentation. In *International Conference on Robotics and Automation*, 2021.

[HZAL18] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[JLCvdS23] Alan Jeffares, Tennison Liu, Jonathan Crabbé, and Mihaela van der Schaar. Joint Training of Deep Ensembles Fails Due to Learner Collusion. *Advances in Neural Information Processing Systems*, 2023.

[KB15] Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *International Conference for Learning Representations*, 2015.

[KFPM21] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: Rapid Motor Adaptation for Legged Robots. *Robotics: Science and Systems*, 2021.

[KLC98] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

[KW03] Ludmila I. Kuncheva and Christopher J. Whitaker. Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy. *Kluwer Academic Publishers*, 2003.

[KW09] Mark Kroon and Shimon Whiteson. Automatic Feature Selection for Model-Based Reinforcement Learning in Factored MDPs. In *2009 International Conference on Machine Learning and Applications*, pages 324–330. IEEE, 2009.

[KZGR23] Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A Survey of Zero-shot Generalisation in Deep Reinforcement Learning. *Journal of Artificial Intelligence Research*, 76:201–264, 2023.

[Lan17] John Langford. Contextual Reinforcement Learning. In *2017 IEEE International Conference on Big Data*, pages 3–3, 2017.

[LAT18] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. SNIP: Single-shot Network Pruning based on Connection Sensitivity. *International Conference on Learning Representations*, 2018.

[LBX+21] Yuchen Li, Yifan Bao, Liyao Xiang, Junhan Liu, Cen Chen, Li Wang, and Xinbing Wang. Privacy Threats Analysis to Secure Federated Learning. *arXiv preprint arXiv:2106.13076*, 2021.

[LCA+22] Shiwei Liu, Tianlong Chen, Zahra Atashgahi, Xiaohan Chen, Ghada Sokar, Elena Mocanu, Mykola Pechenizkiy, Zhangyang Wang, and Decebal Constantin Mocanu. Deep Ensembling with No Overhead for either Training or Testing: The All-Round Blessings of Dynamic Sparsity. *International Conference on Learning Representations*, 2022.

[LCC+23] Shiwei Liu, Tianlong Chen, Xiaohan Chen, Li Shen, Decebal Constantin Mocanu, Zhangyang Wang, and Mykola Pechenizkiy. The Unreasonable Effectiveness of Random Pruning: Return of the Most Naive Baseline for Sparse Training. In *Proceedings of the 11th International Conference on Learning Representations*, 2023. URL: https://arxiv.org/abs/2202.02643.

[LDS89] Yann LeCun, John Denker, and Sara Solla. Optimal Brain Damage. In *Advances in Neural Information Processing Systems*, 1989.

[LHW+20] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47), 2020.

[LL24] Hojung Lee and Jong-Seok Lee. Network Fission Ensembles for Low-Cost Self-Ensembles. 2024.

[LLS+20] Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement Learning with Augmented Data. *Advances in Neural Information Processing Systems*, 33:19884–19895, 2020.

[LMCZ24] Seyyidahmed Lahmer, Federico Mason, Federico Chiariotti, and Andrea Zanella. Fast context adaptation in cost-aware continual learning. *IEEE Transactions on Machine Learning in Communications and Networking*, 2024.

[LMP19] Shiwei Liu, Decebal Constantin Mocanu, and Mykola Pechenizkiy. On improving deep learning generalization with adaptive sparse connectivity. *arXiv preprint arXiv:1906.11626*, 2019.

[LPB17] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. *Advances in Neural Information Processing Systems*, 2017.

[LPC+15] Stefan Lee, Senthil Purushwalkam, Michael Cogswell, David Crandall, and Dhruv Batra. Why M Heads are Better than One: Training a Diverse Ensemble of Deep Networks, 2015.

[LRD+22] Clare Lyle, Mark Rowland, Will Dabney, Marta Kwiatkowska, and Yarin Gal. Learning Dynamics and Generalization in Reinforcement Learning. *Int. Conf. on Machine Learning*, 2022.

[LRS+24] Borja G Leon, Francesco Riccio, Kaushik Subramanian, Peter R Wurman, and Peter Stone. Discovering Creative Behaviors through DUPLEX: Diverse Universal Features for Policy Exploration. In *Neural Information Processing Systems*, 2024.

[LSA20] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: Contrastive Unsupervised Representations for Reinforcement Learning. In *International Conference on Machine Learning*, pages 5639–5650. PMLR, 2020.

[LSL+20] Kimin Lee, Younggyo Seo, Seunghyun Lee, Honglak Lee, and Jinwoo Shin. Context-aware Dynamics Model for Generalization in Model-Based Reinforcement Learning. In *International Conference on Machine Learning*, 2020.

[LXS+20] Junjie Liu, Zhe Xu, Runbin Shi, Ray Cheung, and Hayden So. Dynamic Sparse Training: Find Efficient Sparse Network From Scratch With Trainable Masked Layers. *International Conference for Learning Representations*, 2020.

[LYL25] Pengxiang Li, Lu Yin, and Shiwei Liu. Mix-LN: Unleashing the Power of Deeper Layers by Combining Pre-LN and Post-LN. *International Conference on Learning Representations*, 2025.

[LYMP21] Shiwei Liu, Lu Yin, Decebal Constantin Mocanu, and Mykola Pechenizkiy. Do We Actually Need Dense Over-Parameterization? In-Time Over-Parameterization in Sparse Training. In *International Conference on Machine Learning*, pages 6989–7000. PMLR, 2021.

[MCKS18] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering, 2018.

[MGM+25] Calarina Muslimani, Bram Grooten, Deepak RS Mamillapalli, Mykola Pechenizkiy, Decebal C Mocanu, and Matthew E Taylor. Boosting Robustness in Preference-Based Reinforcement Learning with Dynamic Sparsity. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems*, pages 2687–2689, 2025.

[MHA+22] Janosch Moos, Kay Hansel, Hany Abdulsamad, Svenja Stark, Debora Clever, and Jan Peters. Robust Reinforcement Learning: A Review of Foundations and Recent Advances. *Machine Learning and Knowledge Extraction*, 4(1):276–315, 2022.

[MKS+13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. *NeurIPS Deep Learning Workshop*, 2013.

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[MKV+18] A Rupam Mahmood, Dmytro Korenkevych, Gautham Vasan, William Ma, and James Bergstra. Benchmarking Reinforcement Learning Algorithms on Real-World Robots. In *Conference on robot learning*, pages 561–591. PMLR, 2018.

[MMN+16] Decebal Constantin Mocanu, Elena Mocanu, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. A topological insight into restricted Boltzmann machines. *Machine Learning*, 104:243–270, 2016.

[MMP+21] Decebal Constantin Mocanu, Elena Mocanu, Tiago Pinto, Selima Curci, Phuong H Nguyen, Madeleine Gibescu, Damien Ernst, and Zita A Vale. Sparse Training Theory for Scalable and Efficient Agents. *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems*, 2021.

[MMS+18] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):1–12, 2018.

[MO11] Richard Maclin and David W. Opitz. Popular Ensemble Methods: An Empirical Study. *Journal of Artificial Intelligence Research*, 11:169–198, 2011.

[MPV+16] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to Navigate in Complex Environments. *arXiv preprint arXiv:1611.03673*, 2016.

[MS88] Michael C Mozer and Paul Smolensky. Skeletonization: A Technique for Trimming the Fat from a Network via Relevance Assessment. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1988.

[MWH+23] Dustin Morrill, Thomas J Walsh, Daniel Hernandez, Peter R Wurman, and Peter Stone. Composing Efficient, Robust Tests for Policy Selection. In *Uncertainty in Artificial Intelligence*, pages 1456–1466. PMLR, 2023.

[Neu21] NeuralMagic. DeepSparse Inference Engine, 2021. Source URL: https://neuralmagic.com/deepsparse.

[NGMT23] Aleksandra I. Nowak, Bram Grooten, Decebal Constantin Mocanu, and Jacek Tabor. Fantastic Weights and How to Find Them: Where to Prune in Dynamic Sparse Training. *Advances in Neural Information Processing Systems*, 36:55160–55192, 2023.

[NVI24] NVIDIA. ResNet50 v1.5 for PyTorch, 2024. Source URL: https://catalog.ngc.nvidia.com/orgs/nvidia/resources/resnet_50_v1_5_for_pytorch.

[PAZA18] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. In *Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2018.

[Per14] Alberto E Pereda. Electrical synapses and their functional interactions with chemical synapses. *Nature Reviews Neuroscience*, 15(4):250–263, 2014.

[PFRB24] Sai Prasanna, Karim Farid, Raghu Rajan, and André Biedenkapp. Dreaming of Many Worlds: Learning Contextual World Models Aids Zero-Shot Generalization. *Reinforcement Learning Conference*, 2024.

[QSO+20] Xinghua Qu, Zhu Sun, Yew-Soon Ong, Abhishek Gupta, and Pengfei Wei. Minimalistic Attacks: How Little it Takes to Fool a Deep Reinforcement Learning Policy. *IEEE Transactions on Cognitive and Developmental Systems*, 13(4):806–817, 2020.

[RAG+20] Banafsheh Rafiee, Zaheer Abbas, Sina Ghiassian, Raksha Kumaraswamy, Richard S Sutton, Elliot A Ludvig, and Adam White. From Eye-blinks to State Construction: Diagnostic Benchmarks for Online Representation Learning. *Adaptive Behavior*, 2020.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[RGY+21] Roberta Raileanu, Maxwell Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. Automatic Data Augmentation for Generalization in Reinforcement Learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 5402–5415. Curran Associates, Inc., 2021.

[RM51] Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[RPS+18] Neil Rabinowitz, Frank Perbet, Francis Song, Chiyuan Zhang, SM Ali Eslami, and Matthew Botvinick. Machine Theory of Mind. In *International Conference on Machine Learning*, pages 4218–4227. PMLR, 2018.

[RSMH+23] Sahand Rezaei-Shoshtari, Charlotte Morissette, Francois R Hogan, Gregory Dudek, and David Meger. Hypernetworks for Zero-shot Transfer in Reinforcement Learning. In *The AAAI Conference on Artificial Intelligence*, 2023.

[RSR+20] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[SAH+20] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[SAPM22] Ghada Sokar, Zahra Atashgahi, Mykola Pechenizkiy, and Decebal Constantin Mocanu. Where to Pay Attention in Sparse Training for Feature Selection? In *Advances in Neural Information Processing Systems*, 2022.

[SBBC19] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An Adversarial Winograd Schema Challenge at Scale. 2019.

[Ska96] David B. Skalak. The Sources of Increased Accuracy for Two Proposed Boosting Algorithms. In *AAAI Conference on Artificial Intelligence*, 1996.

[SLZ+21] Ke Sun, Yi Liu, Yingnan Zhao, Hengshuai Yao, Shangling Jui, and Linglong Kong. Exploring the Training Robustness of Distributional Reinforcement Learning against Noisy State Observations. *arXiv preprint arXiv:2109.08776*, 2021.

[SMM+22] Ghada Sokar, Elena Mocanu, Decebal Constantin Mocanu, Mykola Pechenizkiy, and Peter Stone. Dynamic Sparse Training for Deep Reinforcement Learning. *The 31st International Joint Conference on Artificial Intelligence*, 2022.

[SRKJ21] Austin Stone, Oscar Ramirez, Kurt Konolige, and Rico Jonschkowski. The Distracting Control Suite – A Challenging Benchmark for Reinforcement Learning from Pixels. *arXiv preprint arXiv:2101.02722*, 2021.

[SSPS21] David Silver, Satinder Singh, Doina Precup, and Richard S Sutton. Reward is enough. *Artificial Intelligence*, 299:103535, 2021.

[STZ+25] Jingbo Sun, Songjun Tu, Qichao Zhang, Ke Chen, and Dongbin Zhao. Salience-Invariant Consistent Policy Learning for Generalization in Visual Reinforcement Learning. In *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems*, 2025.

[Sut19] Richard S. Sutton. The Bitter Lesson. http://www.incompleteideas.net/IncIdeas/BitterLesson.html, 2019. Accessed: 2025-11-04.

[SWD+17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[TDM+18] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind Control Suite. *arXiv preprint arXiv:1801.00690*, 2018.

[TET12] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[TFR+17] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30. IEEE, 2017.

[THPH23] Yiqin Tan, Pihe Hu, Ling Pan, and Longbo Huang. RLx2: Training a Sparse Deep Reinforcement Learning Model from Scratch. *International Conference on Learning Representations*, 2023.

[TILB23] Manan Tomar, Riashat Islam, Sergey Levine, and Philip Bachman. Ignorance is bliss: Robust control via information gating. *arXiv preprint arXiv:2303.06121*, 2023.

[TRvdP22] Tianxin Tao, Daniele Reda, and Michiel van de Panne. Evaluating Vision Transformer Methods for Deep Reinforcement Learning from Pixels. *arXiv preprint arXiv:2204.04905*, 2022.

[Tuk77] John W. Tukey. *Exploratory Data Analysis*. Number 2 in Addison-Wesley series in Behavioral Science. Addison-Wesley Publishing Company, 1977.

[VDP+20] Eugene Vinitsky, Yuqing Du, Kanaad Parvate, Kathy Jang, Pieter Abbeel, and Alexandre Bayen. Robust Reinforcement Learning using Adversarial Populations. *arXiv preprint arXiv:2008.01825*, 2020.

[VLS22] Marc Aurel Vischer, Robert Tjarko Lange, and Henning Sprekeler. On Lottery Tickets and Minimal Task Representations in Deep Reinforcement Learning. *ICLR*, 2022.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *Advances in Neural Information Processing Systems*, 30, 2017.

[WBK+22] Peter R Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, et al. Outracing champion Gran Turismo drivers with deep reinforcement learning. *Nature*, 602(7896):223–228, 2022.

[WDS+19] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. HuggingFace's Transformers: State-of-the-art Natural Language Processing. 2019.

[WGLX19] Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning Robust Global Representations by Penalizing Local Predictive Power. In *Neural Information Processing Systems*, 2019.

[WKNT+16] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

[WKSF20] Kaixin Wang, Bingyi Kang, Jie Shao, and Jiashi Feng. Improving Generalization in Reinforcement Learning with Mixture Regularization. *Advances in Neural Information Processing Systems*, 33:7968–7978, 2020.

[Wol92] David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

[WSS+05] Shimon Whiteson, Peter Stone, Kenneth O Stanley, Risto Miikkulainen, and Nate Kohl. Automatic Feature Selection in Neuroevolution. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 1225–1232, 2005.

[WTB20] Yeming Wen, Dustin Tran, and Jimmy Ba. BatchEnsemble: An Alternative Approach to Efficient Ensemble and Lifelong Learning. *International Conference on Learning Representations*, 2020.

[WVM23] Yan Wang, Gautham Vasan, and A Rupam Mahmood. Real-Time Reinforcement Learning for Vision-Based Robotics Utilizing Local and Remote Computers. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9435–9441. IEEE, 2023.

[WW22] Tim Whitaker and Darrell Whitley. Prune and Tune Ensembles: Low-Cost Ensemble Learning with Sparse Independent Subnetworks. In *AAAI*, pages 8638–8646. AAAI Press, 2022.

[WWH+24] Shuo Wang, Zhihao Wu, Xiaobo Hu, Jinwen Wang, Youfang Lin, and Kai Lv. What Effects the Generalization in Visual Reinforcement Learning: Policy Consistency with Truncated Return Prediction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(6):5590–5598, Mar. 2024.

[WXW+25] Boqian Wu, Qiao Xiao, Shunxin Wang, Nicola Strisciuglio, Mykola Pechenizkiy, Maurice van Keulen, Decebal Constantin Mocanu, and Elena Mocanu. Dynamic Sparse Training versus Dense Training: The Unexpected Winner in Image Corruption Robustness. *International Conference on Learning Representations*, 2025.

[WZG20] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking Winning Tickets Before Training by Preserving Gradient Flow. *International Conference on Learning Representations*, 2020.

[XNC+21] Jinwei Xing, Takashi Nagata, Kexin Chen, Xinyun Zou, Emre Neftci, and Jeffrey L Krichmar. Domain Adaptation In Reinforcement Learning Via Latent Unified State Representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10452–10459, 2021.

[YKF21] Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image Augmentation Is All You Need: Regularizing Deep Reinforcement Learning from Pixels. *International Conference on Learning Representations*, 2021.

[YM22] Yufeng Yuan and A Rupam Mahmood. Asynchronous Reinforcement Learning for Real-Time Control of Physical Robots. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 5546–5552. IEEE, 2022.

[YMM+22] Zhecheng Yuan, Guozheng Ma, Yao Mu, Bo Xia, Bo Yuan, Xueqian Wang, Ping Luo, and Huazhe Xu. Don't Touch What Matters: Task-Aware Lipschitz Data Augmentation for Visual Reinforcement Learning. *International Joint Conference on Artificial Intelligence*, 2022.

[YTLT17] Wenhao Yu, Jie Tan, C Karen Liu, and Greg Turk. Preparing for the Unknown: Learning a Universal Policy with Online System Identification. *Robotics: Science and Systems*, 2017.

[YXY+22] Zhecheng Yuan, Zhengrong Xue, Bo Yuan, Xueqian Wang, Yi Wu, Yang Gao, and Huazhe Xu. Pre-Trained Image Encoder for Generalizable Visual Reinforcement Learning. *Advances in Neural Information Processing Systems*, 35:13022–13037, 2022.

[YZL+22] Tao Yu, Zhizheng Zhang, Cuiling Lan, Yan Lu, and Zhibo Chen. Mask-based Latent Reconstruction for Reinforcement Learning. *Advances in Neural Information Processing Systems*, 35:25117–25131, 2022.

[ZCX+20] Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Mingyan Liu, Duane Boning, and Cho-Jui Hsieh. Robust Deep Reinforcement Learning against Adversarial Perturbations on State Observations. *Advances in Neural Information Processing Systems*, 33:21024–21037, 2020.

[ZHB+19] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a Machine Really Finish Your Sentence? *Association for Computational Linguistics*, 2019.

[Zho12] Zhi-Hua Zhou. *Ensemble Methods: Foundations and Algorithms*. Chapman & Hall/CRC press, 2012.

[ZLK+17] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million Image Database for Scene Recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, 2017.

[ZMC+21] Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning Invariant Representations for Reinforcement Learning without Reconstruction. *International Conference on Learning Representations (ICLR)*, 2021.

[ZMK+17] Y Zhu, R Mottaghi, E Kolve, JJ Lim, A Gupta, L Fei-Fei, and A Farhadi. Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning. *International Conference on Robotics and Automation*, 2017.

[ZVMB18] Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. A Study on Overfitting in Deep Reinforcement Learning. *arXiv preprint arXiv:1804.06893*, 2018.

[ZZC+24] Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. GaLore: Memory-Efficient LLM Training by Gradient Low-Rank Projection. *International Conference on Machine Learning*, 2024.

[ZZW+25] Yingtao Zhang, Jialin Zhao, Wenjing Wu, Ziheng Liao, Umberto Michieli, and Carlo Vittorio Cannistraci. Brain-inspired sparse training enables Transformers and LLMs to perform as fully connected. *arXiv preprint arXiv:2501.19107*, 2025.

APPENDIX A

Noisy Environments

This Appendix belongs to Chapter 2. Here I provide additional details such as pseudocode for the ANF algorithm (Section A.1), experimental settings (Section A.2), and extra results (Section A.3).

Code: `https://github.com/bramgrooten/automatic-noise-filtering`

A.1 ANF ALGORITHM

In this section, we provide pseudocode of the Automatic Noise Filtering (ANF) algorithm. In Algorithm 2 we show the implementation of ANF-SAC, but keep in mind that ANF can be applied to any MLP-based deep RL method. The novel parts of our proposed method ANF are colored purple. The dynamic sparse training components, already introduced by [SMM+22], are in blue. The rest (in black) is the standard SAC algorithm [HZAL18].

A.2 EXPERIMENTAL DETAILS

Algorithms Throughout the experiments in Chapter 2 we compared our proposed dynamic sparse algorithms (ANF-SAC and ANF-TD3) with the fully-dense counterparts (SAC and TD3), and the static sparse variants (Static-ANF-SAC and Static-ANF-TD3). The complete list of hyperparameters can be found in Table A.1. Aiming for a fair comparison, we tried to maximize the number of shared hyperparameters. Table A.1 also includes the parameters used for the experiments described in Section 2.6.4, where an increasing sparsity level is incorporated in ANF beyond the first layer, leading to the Sparser-ANF-SAC and Sparser-ANF-TD3 versions.

Algorithm 2 ANF-SAC

Require: initial collect steps b_{init}, train every k env steps, minibatch size n, learning rate λ, target smoothing coefficient τ, max env steps T, topology-change period ΔT, drop fraction d_f, input layer sparsity s_i

Initialize the actor π and critics Q_1, Q_2, with weights ϕ, θ_1, θ_2.
Randomly prune the input layer of all 3 networks to sparsity level s_i.
Duplicate the critics to create target networks, with weights $\bar{\theta}_1 = \theta_1$, $\bar{\theta}_2 = \theta_2$.
Initialize the replay buffer $\mathcal{B}$ with b_{init} random actions.
for $t = 1$ **to** T **do**
    Sample action a from the policy based on current state s: $a \sim \pi_\phi(\cdot|s)$
    Take a step in the environment and observe reward r and new state s'.
    Store transition tuple (s, a, r, s') in $\mathcal{B}$.
    if t mod $k == 0$ **then**
        Sample minibatch of n transitions from $\mathcal{B}$.
        Update the weights according to SAC's objective functions J_Q, J_π:
            $\theta_i \leftarrow \theta_i - \lambda \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$
            $\phi \leftarrow \phi - \lambda \hat{\nabla}_\phi J_\pi(\phi)$
        Update the target networks:
            $\bar{\theta}_i \leftarrow \tau\theta_i + (1 - \tau)\bar{\theta}_i$ for $i \in \{1, 2\}$
    end if
    if t mod $\Delta T == 0$ **then**
        Update the topology of the networks:
            Prune fraction d_f of the smallest magnitude weights.
            Grow fraction d_f new weights randomly, initialize at value 0.
            For pruned weights, mask the running averages in Adam:
                $m \leftarrow 0, v \leftarrow 0$.
    end if
end for

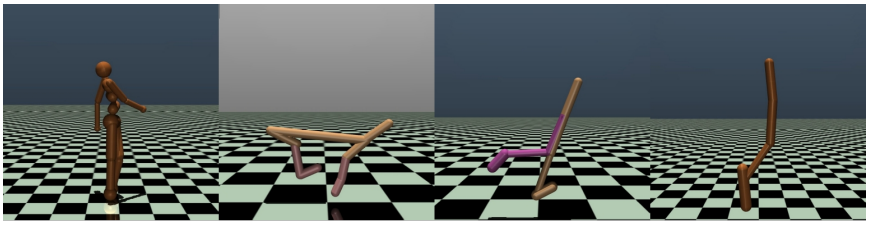


Figure A.1. The MuJoCo Gym [TET12, BCP+16] environments used as a base for our ENEs.

Environments We used as the foundation for our extremely noisy environments (ENEs) four continuous control tasks (Humanoid-v3,[1] HalfCheetah-v3, Walker-v3, and Hopper-v3) as shown in Figure A.1. See Table A.2 for the parameters corresponding to the ENEs built on top of these four worlds.

A.3 ADDITIONAL RESULTS

In this Appendix, we share additional results for Chapter 2. Some figures present outcomes of additional algorithms (Figure A.2), while other results such as Figure A.3 and Table A.3 show extra analysis of the network connectivity.

We see in Figure A.2; when the noise level increases our ANF method outperforms the dense baseline by a significant margin on most environments, similar to Figure 2.4 of Section 2.4. Especially in the noisiest environments a large gap is visible between ANF-TD3 and TD3. The sparse DS-TD3 is a strong baseline, being quite similar in methodology to ANF-TD3.

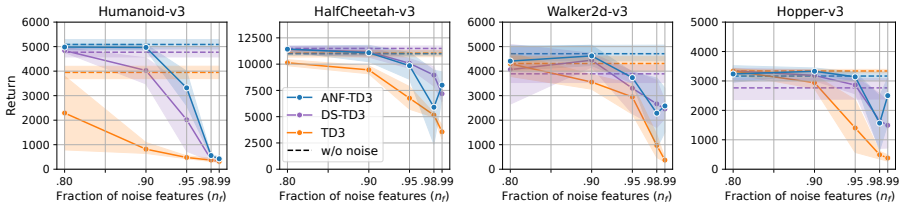


Figure A.2. Performance of ANF compared to its baselines for different fractions of noise features n_f. The standard dense networks of TD3 struggle under high n_f values. Similar graphs for ANF-SAC are shown in Figure 2.4 of Section 2.4.

[1] Please note that according to the environment's documentation (https://www.gymlibrary.dev/environments/mujoco/humanoid/#observation-space) versions 1,2,3 of Humanoid contain an issue with the contact forces (the corresponding features always give 0). Humanoid-v4 solves this, but was released too late for our research.

Table A.1. Hyperparameters for the experiments in Chapter 2.

Parameter	Value
Shared by all algorithms	
optimizer	Adam [KB15]
learning rate (λ)	$1 \cdot 10^{-3}$
weight decay	$2 \cdot 10^{-4}$
discount (γ)	0.99
nonlinearity	ReLU
replay buffer size	10^6
initial collect steps (b_{init})	25,000
network type	MLP
number of hidden layers	2
number of neurons per hidden layer	256
minibatch size (n)	100
target smoothing coefficient (τ)	0.005
train every k_c env steps (critic), $k_c =$	1
gradient steps per training step =	1
SAC and ANF-SAC	
SAC type (Gaussian / Deterministic)	Gaussian
temperature (α in [HZAL18])	0.2
automatic temperature tuning	False
train every k_a env steps (actor), $k_a =$	1
target update interval (k_{tar})	1
TD3 and ANF-TD3	
train every k_a env steps (actor), $k_a =$	2
target update interval (k_{tar})	2
std. dev. of exploration noise (σ in [FHM18])	0.1
std. dev. of sampling noise ($\tilde{\sigma}$ in [FHM18])	0.2
sampling noise clip (c in [FHM18])	0.5
ANF	
sparsity of the input layer (s_i)	0.8
drop fraction (d_f)	0.05
new weights init value	0
sparse layers	input layer
topology-change period (ΔT), env steps	1000
Static-ANF	
topology-change period (ΔT), env steps	∞ (no change)
Sparser-ANF	
global sparsity	varying (Section 2.6.4)
sparsity distribution (over sparse layers)	uniform
sparse layers	input & hidden

Table A.2. Extremely noisy environment (ENE) parameters.

Parameter	Value	Experiments in
noise fraction (n_f)	varying	Section 2.4
noise distribution	$\mathcal{N}(0, \sigma^2)$	Sections 2.4–2.6.1, 2.6.3–2.6.4
	Imitate	Section 2.6.2
noise amplitude (σ)	1	Sections 2.4–2.5, 2.6.3–2.6.4
	varying	Section 2.6.1
permutation period (T_p)	1M	Section 2.5

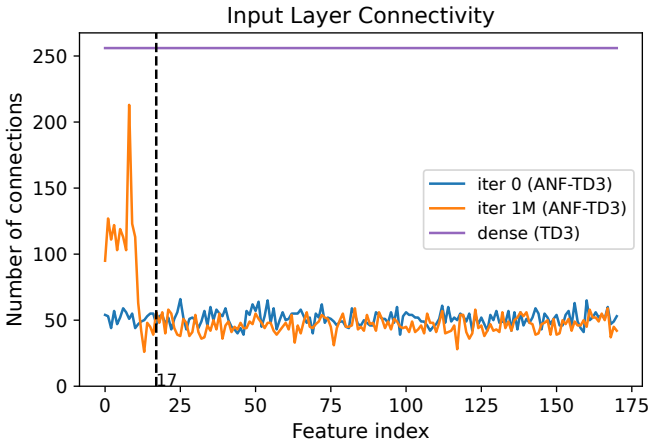


Figure A.3. Showing the connectivity of the actor network's input layer for ANF-TD3 on HalfCheetah-v3 with 90% noise features. At the start of training (iteration 0) every input neuron has around $256 \cdot 0.2 \approx 51$ connections, because the input layer sparsity is 80% and connections are allocated uniformly at random. During training, ANF-TD3 gradually removes connections from the noise features and adds connections to the relevant features (the 17 leftmost input neurons in this graph). Note that a dense network always has 256 connections to every input neuron, and thus gets distracted more easily by all the noise features.

A noteworthy observation for Figure A.3 is the fact that apparently not all 17 input features are deemed to be useful, as some received even fewer than the original ~51 connections. This phenomenon has been shown in standard RL environments recently by [VLS22], and it is interesting to see that it still holds up in extremely noisy environments. Original state features that fall outside the *minimal task representation* are considered just as irrelevant as the noise features, according to Figure A.3. We see that ANF filters not only through synthetic features, but also de-emphasizes redundant features.

Table A.3. Experiment with non-zero-centered noise $\mathcal{N}(\mu, 1)$ on HalfCheetah-v3. Noise fraction $n_f = .98$ and 5 seeds.

Avg. returns	$\mu = 0$	$\mu = 1$	$\mu = -2$	$\mu = 4$
ANF-SAC	9250.4	5642.2	5047.8	636.9
SAC	6124.3	3744.4	419.1	−35.3

A.3.1 Non-zero-centered noise

The zero-centered $\mathcal{N}(0, 1)$ noise may simplify ANF's process of pruning connections to noise-features. We ran extra experiments, see Table A.3, with non-zero-centered Gaussian noise to show empirically that: (i) non-zero-centered noise is indeed more challenging and (ii) ANF is still able to handle it better than dense networks.

Theoretical perspective. Even with non-zero-mean noise, we think the weights connected to noise-features will stay close to zero (and thus get pruned by ANF). Note that initial weights are small, and ANF's newly grown weights start at 0. Since a noise-feature is irrelevant, its connections will receive mixed signals (positive gradient for some $\langle s, a, r \rangle$ tuples, negative gradient for others). This means it barely gets a chance to grow. In a simplified setting, we can prove a stronger claim;

Conjecture: Weights connected to noise features converge to 0 with gradient descent, even for non-zero-centered noise.

Proof. (Not a full proof, only for a simple setting.) We assume to be in the local neighborhood of the optimum. Suppose we have a function approximator $f(x_1, x_2) = w_1 \cdot x_1 + w_2 \cdot x_2 = y_{\text{approx}}$, which is trying to estimate the true function $g(x_1) = a \cdot x_1 = y$. Note: $g(\cdot)$ does not depend on x_2 (noise-feature).

Suppose we use mean-squared-error loss: $L = \frac{1}{2}(f(x_1, x_2) - y)^2$. Then the gradient (partial derivative) of weight w_2 is:

$$\frac{\mathrm{d}L}{\mathrm{d}w_2} = (f(x_1, x_2) - y) \cdot x_2 = (w_1 \cdot x_1 + w_2 \cdot x_2 - a \cdot x_1) \cdot x_2.$$

Assuming we're near the optimum, i.e., $w_1 - a \approx 0$, we can rewrite $\frac{\mathrm{d}L}{\mathrm{d}w_2} \approx w_2 \cdot (x_2)^2$. For any noise x_2: if w_2 is negative, its gradient is negative, and vice versa. We minimize MSE-loss, so gradient descent moves in the direction: −gradient. Thus, w_2 will be updated toward zero. □

A.3.2 Matching the input-layer-sparsity-level with the noise-fraction

From the problem setting of extremely noisy environments (ENEs) it seems beneficial to match the algorithm's input-layer-sparsity-level with the noise-fraction of the ENE. We ran this experiment but omitted it from the main body for the following reasons:

(i) performance is similar (see Table A.4 below, it's challenging to prune all connections to noise-features, so it may be useful to have some surplus of connections for task-relevant features),

(ii) we did not want to assume that the agent knows the noise fraction. (The input-layer-sparsity-level could be an adaptive parameter; an interesting direction for future work.)

Instead, we kept the input-layer-sparsity at a well-working 80% to have a generally applicable algorithm.

Table A.4. Experiments matching input-layer-sparsity-levels with noise-fractions (policy = ANF-SAC, env = HalfCheetah-v3, num_seeds = 5).

Avg. returns	$n_f = .90$	$n_f = .95$	$n_f = .98$	$n_f = .99$
matching input-layer-sparsity	11041.1	10259.0	10090.3	8641.2
80% input-layer-sparsity	11913.8	9920.5	9250.4	10007.2

§

APPENDIX B

Visual Distractions

In this Appendix belonging to Chapter 3, I provide additional implementation details (Section B.1), results (Section B.2), and examples of masks created by the MaDi algorithm (Section B.3).

Code: `https://github.com/bramgrooten/mask-distractions`

B.1 IMPLEMENTATION DETAILS

We provide further details on our implementation of MaDi. Table B.1 presents an overview of the hyperparameters used in our study. For the simulation experiments we used the default hyperparameters as specified by the DMControl Generalization Benchmark [HW21] and the particular baselines. For the robotic task, we obtained hyperparameters from [WVM23] while aiming to keep the majority the same.

Network Architecture. All the algorithms in our experiments use the same base architecture, acquired from the DMControl Generalization Benchmark [HW21], to ensure a fair comparison. This base architecture (yellow modules in Figure 3.1) consists of:

- **Encoder:** a shared 11-layer ConvNet that takes a stack of 3 RGB frames (shape $[9, 84, 84]$) and outputs spatial features of size $[32, 21, 21]$. Each layer has 32 channels with a kernel size of 3×3, a stride of 1 and no padding. Only the first layer has a stride of 2, to reduce the channel size quicker. We use ReLU activations between all convolutional layers.
- **Actor:** a multi-layer perceptron (MLP) that outputs actions. It first projects the $32 \cdot 21 \cdot 21 = 14,112$ flattened features to a vector of just 100 through a large MLP layer. We then apply LayerNorm [BKH16] and a Tanh activation function. The

bulk of the actor network follows, with three MLP layers of hidden dimension 1024 with ReLU activations in between. The output layer has twice the size of the action dimension, to output means μ and standard deviations σ.

- **Critic:** two independent MLPs that output state-action values. Both have the same architecture as the actor. The large projection (up to and including the Tanh activation) is shared between the two critics. The critics receive the current action $\boldsymbol{a}$ as input as well; this is concatenated to the output of the large projection. Their last layers only have one output neuron, determining the Q-value estimate.

For the robotic experiments the architectures are a bit different:

- **Encoder:** a shared 4-layer ConvNet that takes a stack of 3 RGB frames (shape $[9, 90, 160]$), followed by a spatial softmax layer [FTD+16]. We use 32 channels in each layer, kernel size is 3×3, no padding, and stride 2 on all layers except the last one, which uses 1. ReLU activation functions between the layers. The encoder outputs a vector of length 64.

- **Actor:** an MLP that outputs actions. It first concatenates the encoder features with a proprioception vector of length 15, which consists of the angles, angular velocities, and previous actions of each of the five joints that we control. (The sixth joint is for a gripper, which we do not use.) After that, two MLP layers of hidden dimension 512 follow.

- **Critic:** two independent MLPs that output state-action values. Both have the same architecture as the actor, except again the concatenation of the action $\boldsymbol{a}$ in the input and an output layer of dimension 1.

And of course, the critic network has an exact replica, its *target network*, following the SAC [HZAL18] design. The algorithms **SAC**, **DrQ**, **RAD**, and **SVEA** all use this exact architecture. The other methods include an additional component:

- **SODA** has an auxiliary network that tries to minimize the distance of augmented and non-augmented images in their feature space.

- **SGQN** comes with another head attached to the critic's large projection. This auxiliary component tries to minimize the difference between the masks it outputs and the masks generated by a saliency metric.

Mask Application. In MaDi, the mask generated by the Masker is applied to the visual input by element-wise multiplication before it is passed to the actor and critic networks. Masks are generated and applied per frame, but we only need one forward pass through the Masker to get masks for all 3 frames of a stacked input observation. See the following code snippet:

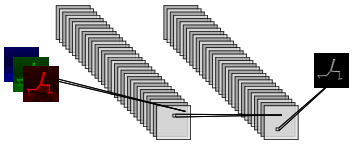


Figure B.1. The Masker network architecture contains two hidden layers with 32 channels and ReLU activations, while the output layer produces just 1 channel with a Sigmoid activation.

```
def apply_mask(self, obs):                          # obs shape: (B,9,H,W)
    """Apply masks generated by one forward pass of the Masker"""

    # split the stacked frames
    frames = obs.chunk(3, dim=1)     # [(B,3,H,W), (B,3,H,W), (B,3,H,W)]

    # concat in batch dim
    frames_cat = torch.cat(frames, dim=0)                   # (3B,3,H,W)

    # one forward pass through the Masker network
    masks_cat = self.masker(frames_cat)                     # (3B,1,H,W)

    # split the batch dim back
    masks = masks_cat.chunk(3, dim=0)  # [(B,1,H,W),(B,1,H,W),(B,1,H,W)]

    # element-wise multiplication
    # uses broadcasting over the 3 RGB channels within 1 frame
    masked_frames = [m*f for m,f in zip(masks,frames)]
                                    # [(B,3,H,W), (B,3,H,W), (B,3,H,W)]

    # concat in channel dim
    return torch.cat(masked_frames, dim=1)                   # (B,9,H,W)
```

B.1.1 Vision Transformer implementation

The design of our Vision Transformer [DBK+21, ViT] experiments closely follows the setup of SVEA [HSW21], where the ConvNet encoder of Figure 3.1 is replaced by a ViT encoder, while introducing minimal changes to the rest of the architecture. For this ViT encoder, we use RGB frames with dimensions $96 \times 96 \times 3$ as input instead

of the $84 \times 84 \times 3$ used for the ConvNet. From this, we extract a total of 144 non-overlapping $8 \times 8 \times 3k$ image patches (with k being the frame count in a stacked observation as described in Section 3.3). Each patch is mapped to a 128-dimensional embedding, which becomes a token to be processed by the ViT encoder. Consistent with the original ViT design, our model incorporates learned positional encodings and a modifiable `class` token. The architecture comprises four Transformer [VSP+17] blocks. Each block applies Multi-Head Attention encompassing eight heads, after which an MLP layer with hidden dimension 128 and a GELU [HG16] activation follows. We optimize the ViT encoder with the critic loss, just like the original ConvNet. We maintain the same optimizer (Adam) and learning rates as in Table B.1, but we double the batch size to 256. Our ViT encoder starts without any pre-trained parameters, and we do not use weight decay. The entire training process spans around three days on a single NVIDIA RTX A4000 GPU.

B.2 ADDITIONAL RESULTS

This section provides additional results from our experiments, demonstrating the performance of MaDi in the original training environments and various other environments with differing levels of distractions. Results on `video_easy` from the DMControl Generalization Benchmark [HW21] are already presented in Table 3.1 of the main text, see Section 3.5.2. Here we show the results on the original training environments in Sections B.2.1 and B.2.3, as well as evaluations on `video_hard` in Section B.2.5, and the Distracting Control Suite in Section B.2.6.

Statistical tests. We verify the statistical significance of the results in our tables by running the unequal variances t-test (also called Welch's t-test) between MaDi and the best baseline for each environment. This test does not assume similar variances, unlike the Student's t-test. All our tests are two-sided, and we consider p-values less than 0.05 as statistically significant.

Training directly on distractions. Interestingly enough, training directly on the challenging environments with video backgrounds does not work well. In our preliminary experiments, this holds for all algorithms in both the simulation and robotic environments. Some clean data seems necessary to learn an adequate policy, before agents can learn to deal with distractions. Finding the ideal curriculum could be an interesting direction for future work.

Table B.1. Overview of the hyperparameters used in the experiments of Chapter 3.

Hyperparameter	Simulation experiments	Robotic experiments (if different)
Shared by all algorithms		
optimizer	Adam [KB15]	
learning rate (actor and critic)	10^{-3}	$3 \cdot 10^{-4}$
Adam β_1, β_2 (all networks)	0.9, 0.999	
discount (γ)	0.99	
frame stack	3	
action repeat	4	5
batch size	128	
initial collect steps	1000	
replay buffer size	500K	100K
environment steps	500K	100K
train every k_c env steps (critic), k_c =	1	asynchronous
train every k_a env steps (actor), k_a =	2	once every critic update
gradient steps per training step	1	
target update interval (k_{tar})	2	once every critic update
target smoothing coefficient critic (τ)	0.01	0.005
target smoothing coefficient encoder (τ_{enc})	0.05	0.005
initial temperature (α in SAC)	0.1	
learning rate (for temperature α)	10^{-4}	$3 \cdot 10^{-4}$
Adam β_1, β_2 (for temperature α)	0.5, 0.999	
DrQ [YKF21]		
K, M (Q-target, Q-function averaging)	1, 1	
random shift	Up to ±4 pixels	
RAD [LLS+20]		
random crop	$100 \times 100 \longrightarrow 84 \times 84$	$160 \times 90 \longrightarrow 156 \times 88$
SODA [HW21]		
learning rate auxiliary network	10^{-3}	
batch size auxiliary network	256	
target smoothing coefficient aux net (τ_{aux})	0.005	
train every k_{aux} env steps (aux), k_{aux} =	2	
SVEA [HSW21]		
svea_alpha, svea_beta	0.5, 0.5	
SGQN [BZZR22]		
threshold `sgqn_quantile`[a]	0.95 or 0.98	
weight decay critic	10^{-5}	
learning rate SGQN head	$3 \cdot 10^{-4}$	
train every k_{sgqn} env steps (SGQN head), k_{sgqn} =	2	
MaDi		
learning rate masker network	10^{-3}	$3 \cdot 10^{-4}$
train every k_{m} env steps (masker), k_{m} =	1	

[a] The `sgqn_quantile` is set to 0.95 for domains with large agents (e.g., `walker` and `finger`), and 0.98 for domains with smaller agents (e.g., `cartpole` and `ball_in_cup`).

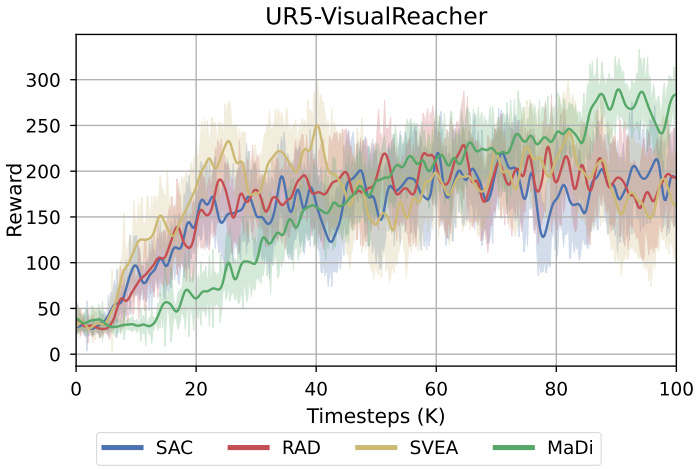


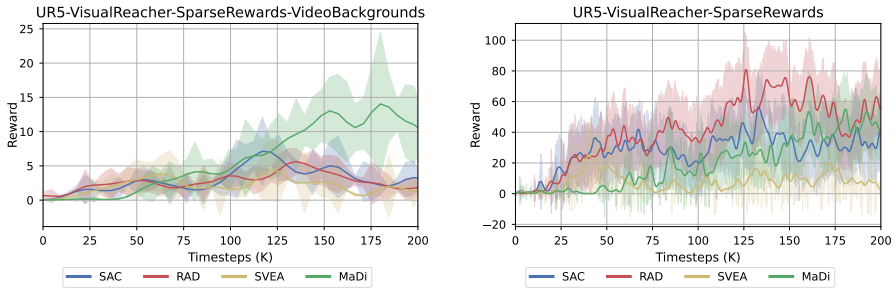
Figure B.2. Performance of MaDi and three baselines on the UR5-VisualReacher training environment. Agents are trained on this clean setup (white background with the red dot) for $100K$ steps. Curves show the mean over five seeds with standard error shaded alongside. Results on the generalization task are shown in Figure 3.7.

B.2.1 Results on training environment UR5 Robotic Arm

In the UR5 Robotic Arm experiments, the algorithms are trained asynchronously. This means that MaDi makes fewer updates than the others, as it takes some extra time to use the Masker network. However, even with fewer updates it surpasses the performance of other baselines on the training task; see Figure B.2. The corresponding results on the generalization task are presented in Section 3.6, Figure 3.7.

B.2.2 Results on UR5 Robotic Arm with *sparse* rewards

In the *sparse reward* setting of the UR5-VisualReacher task, the agent is only rewarded with a +1 whenever its camera is close enough to the red target (according to a predefined threshold). For all other situations, it receives zero reward. Even in this more challenging setting, MaDi is able to outperform the baselines in terms of generalization, as shown in Figure B.3a. On the training environment, presented in Figure B.3b, MaDi learns a bit slower but eventually reaches similar performance to SAC and RAD. SVEA seems to struggle a lot with sparse rewards. The MaDi and SVEA agents both use the default `overlay` augmentation here.



(a) testing environment **(b)** training environment

Figure B.3. Performance of MaDi and three baselines on the *sparse reward* version of the UR5-VisualReacher environment. Agents are trained on the clean setup (white background with the red dot) for $200K$ steps. Curves show the mean over five seeds with standard error shaded alongside.

B.2.3 Results on training environment DMControl

The results of MaDi and multiple baselines on the original training environments of the DeepMind Control Suite are shown in Table B.2. The best performance is achieved by DrQ, an algorithm that only uses light augmentations (random shifts). The stronger augmentations that SODA, SVEA, SGQN, and MaDi use do not improve the results on the training environment, as they are specifically designed to enhance generalization to other domains. MaDi does still come close to the state-of-the-art in the training environments as well.

B.2.4 Results on `video_easy`

The performance of MaDi and various other baseline algorithms in the `video_easy` environment is presented in Table B.3. This is the same as Table 3.1 in the main body, but here we have enough space to include p-values of our statistical tests between MaDi and the best baseline. The `video_easy` environment represents a scenario with a low level of visual distractions, making it a suitable benchmark to test the generalization performance of reinforcement learning algorithms in a still relatively clean environment.

B.2.5 Results on `video_hard`

The performance of MaDi and various other baseline algorithms in the `video_hard` environment from the DMControl Generalization Benchmark [HW21] is presented

Table B.2. Performance on the **original training environments** of MaDi and various baseline algorithms after training for $500K$ steps. Showing mean undiscounted return and standard error over five seeds. MaDi is competitive (no significant difference) with the best baseline in 5/6 cases, denoted by $\star$.

clean	SAC	DrQ	RAD	SODA	SVEA	SGQN	MaDi	p-value
ball_in_cup catch	727 ±117	**973** ±4	651 ±178	965 ±4	811 ±144	778 ±174	808 ±144	0.313*
cartpole balance	**996** ±1	993 ±3	985 ±4	981 ±4	978 ±2	966 ±7	986 ±3	0.024
cartpole swingup	861 ±8	**865** ±4	861 ±8	230 ±142	842 ±8	806 ±14	851 ±5	0.077*
finger spin	324 ±9	643 ±36	593 ±13	493 ±117	**711** ±55	597 ±13	681 ±17	0.626*
walker stand	580 ±117	**979** ±4	965 ±3	147 ±17	973 ±2	694 ±144	972 ±2	0.147*
walker walk	366 ±42	892 ±33	837 ±21	496 ±173	**937** ±13	892 ±27	893 ±27	0.193*
avg	642	891	816	552	875	789	865	

Table B.3. Generalization performance of MaDi and various baseline algorithms on six different environments after training for $500K$ steps. Evaluated on **video_easy** from the DMControl-GB benchmark, showing mean and standard error over five seeds. MaDi significantly outperforms ($\star\star$) or is competitive to ($\star$) the state-of-the-art in 5/6 environments.

video_easy	SAC	DrQ	RAD	SODA	SVEA	SGQN	MaDi	p-value
ball_in_cup catch	602 ±91	714 ±131	561 ±147	750 ±98	757 ±138	761 ±171	**807** ±144	0.841*
cartpole balance	924 ±19	932 ±33	801 ±95	961 ±10	967 ±2	965 ±5	**982** ±4	0.011**
cartpole swingup	782 ±21	613 ±74	658 ±17	215 ±125	786 ±15	798 ±13	**848** ±6	0.015**
finger spin	227 ±26	543 ±50	479 ±65	429 ±100	645 ±39	592 ±11	**679** ±17	0.461*
walker stand	507 ±113	954 ±10	961 ±1	147 ±17	**977** ±3	672 ±153	967 ±3	0.041
walker walk	334 ±37	821 ±38	726 ±42	479 ±168	**936** ±14	882 ±26	895 ±24	0.183*
avg	563	763	698	497	845	778	863	

Table B.4. Generalization performance of MaDi and various baseline algorithms on six different environments trained for $500K$ steps. Evaluated on **`video_hard`** from the DMControl-GB benchmark, showing mean undiscounted return and standard error over five seeds. MaDi significantly outperforms ($\star\star$) or is competitive to ($\star$) the state-of-the-art in 4/6 environments.

video_hard	SAC	DrQ	RAD	SODA	SVEA	SGQN	MaDi	p-value
ball_in_cup catch	176 ±38	180 ±51	261 ±58	194 ±39	327 ±59	687 ±155	**758 ±135**	0.737*
cartpole balance	314 ±12	250 ±6	263 ±18	449 ±46	579 ±26	703 ±17	**827 ±25**	0.005**
cartpole swingup	140 ±10	155 ±10	133 ±19	172 ±46	453 ±26	488 ±18	**619 ±24**	0.003**
finger spin	21 ±4	16 ±3	8 ±2	107 ±30	154 ±31	**554 ±8**	358 ±25	0.001
walker stand	233 ±28	361 ±44	343 ±31	143 ±12	847 ±18	606 ±92	**920 ±14**	0.012**
walker walk	168 ±19	111 ±16	116 ±18	214 ±68	526 ±55	**718 ±37**	504 ±33	0.003
avg	175	179	187	213	481	626	664	

Table B.5. Generalization performance of MaDi and various baseline algorithms on six different environments after training for $500K$ steps. We evaluate on the **Distracting Control Suite** with intensity 0.1, showing mean undiscounted return and standard error over five seeds. MaDi is competitive (no significant difference) with the best baseline in 5/6 environments, see $\star$.

distracting_cs	SAC	DrQ	RAD	SODA	SVEA	SGQN	MaDi	p-value
ball_in_cup catch	112 ±69	177 ±67	33 ±30	107 ±30	105 ±44	**529 ±111**	477 ±99	0.738*
cartpole balance	310 ±22	237 ±19	233 ±19	210 ±6	278 ±22	358 ±17	**377 ±12**	0.373*
cartpole swingup	146 ±8	158 ±13	162 ±6	138 ±8	219 ±18	**283 ±20**	252 ±30	0.412*
finger spin	28 ±5	55 ±13	38 ±15	52 ±18	137 ±21	**386 ±13**	211 ±30	0.002
walker stand	353 ±66	727 ±58	539 ±43	157 ±20	**871 ±22**	641 ±110	866 ±16	0.875*
walker walk	220 ±12	429 ±19	291 ±30	235 ±91	568 ±28	**642 ±44**	570 ±49	0.303*
avg	195	297	216	150	363	473	459	

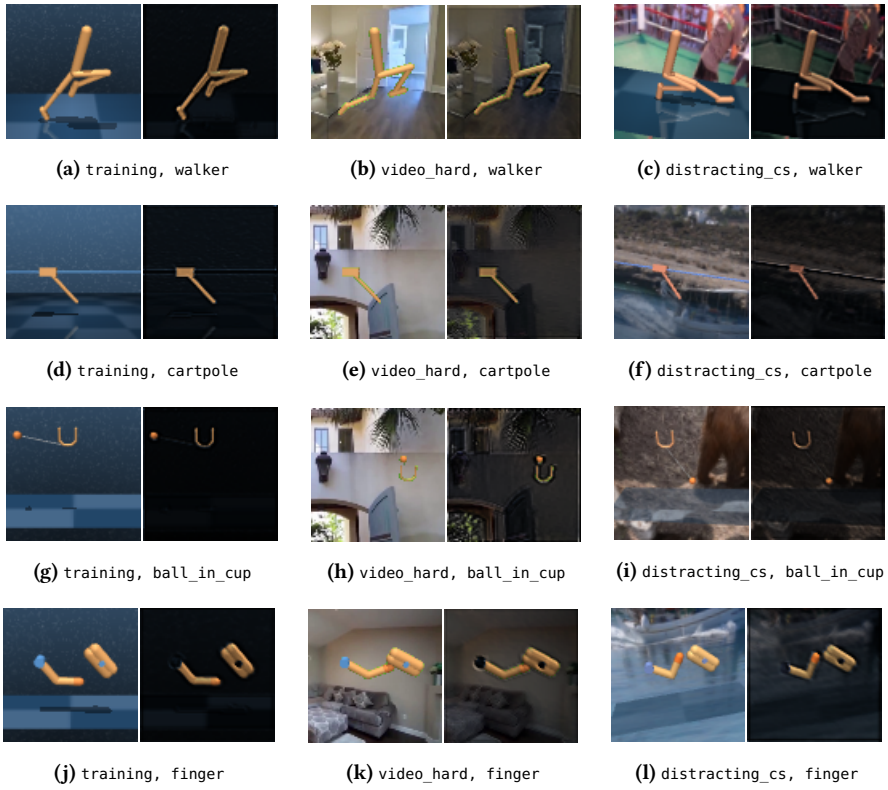
in Table B.4. The `video_hard` environment represents a scenario with a higher level of visual distractions, making it a suitable benchmark to test the generalization performance of reinforcement learning algorithms in an environment full of distractions. Similar to the results of Table 3.1 on `video_easy`, MaDi also shows the best generalization capability overall on `video_hard`.

B.2.6 Results on DistractingCS

We demonstrate the performance of MaDi and our baseline algorithms in the Distracting Control Suite [SRKJ21] with intensity 0.1 in Table B.5. DistractingCS is an extreme benchmark that features different distractions, such as varying camera angles, changing agent colors, and random videos in the background. The results indicate that MaDi performs well on this very challenging benchmark.

B.3 MASK EXAMPLES

In Figure B.4, we show examples of masked observations by the Masker network of MaDi. These masked observations are seen by the actor and critic networks, enabling them to perform well even in the presence of visual distractions.



(a) `training, walker` **(b)** `video_hard, walker` **(c)** `distracting_cs, walker`

(d) `training, cartpole` **(e)** `video_hard, cartpole` **(f)** `distracting_cs, cartpole`

(g) `training, ball_in_cup` **(h)** `video_hard, ball_in_cup` **(i)** `distracting_cs, ball_in_cup`

(j) `training, finger` **(k)** `video_hard, finger` **(l)** `distracting_cs, finger`

Figure B.4. Examples of masked observations by MaDi under the `overlay` augmentation in training and testing environments (`video_hard` and `distracting_cs`) for all domains used. The Masker network produces useful masks, displaying task-relevant information and dimming the distractions.

§

APPENDIX C

Unseen Contexts

This Appendix belongs to Chapter 4. Here I provide additional details on the algorithm and training setup (Section C.1), and further analysis (Section C.2) including an ablation study on the ideal rollout policy, and a sensitivity analysis of the history length hyperparameter.

Code: `https://github.com/bramgrooten/sparc`

C.1 TRAINING DETAILS

All experiments are conducted using the off-policy QR-SAC algorithm [WBK+22] as the base reinforcement learning method. In the *MuJoCo* experiments, we train for $3M$ policy updates. For Gran Turismo, in the *Power & Mass* setting we perform $6M$ updates, while across *Car Models* we train for $9M$ steps. The famously long and difficult *Nürburgring* track is an exception, where we perform additional updates: $12M$, see Table C.1.

We tune the history length (see Section C.2.2) and the history adapter's learning rate (values: $\{2.5 \times 10^{-4}, 2.5 \times 10^{-5}\}$). The list of hyperparameters used for training our Gran Turismo and MuJoCo experiments is presented in Table C.2.

In each episode, a new IND setting is sampled according to Equation 4.1, determining either the wind speed, the car's power & mass, or the car model. Generalization complexity is further raised for the *Car Model* experiment by sampling uniformly over 9 tire types, from least traction *Comfort Hard* to most traction *Racing Soft*.

C.1.1 Reproducibility

The code for *Gran Turismo 7* is proprietary. The game runs exclusively on PlayStation (PS4 and PS5), which we use for training. However, for reproducibility we do provide

Table C.1. Number of training steps for each environment.

Environment	Training steps
MuJoCo	$3M$
Gran Turismo (Power & Mass)	$6M$
Gran Turismo (Car Models)	$9M$ (Nürburgring track: $12M$)

open-source code for the multiple wind-perturbed MuJoCo environments that we designed. Furthermore, see Algorithm 3 for pseudocode of our algorithm SPARC. The resulting adapter policy $\pi_\theta^{ad}(o, h)$ no longer requires the context c as input, and is the model deployed at test time.

C.1.2 Model Architecture

The architecture of SPARC used in our experiments is detailed in Table C.3. We use QR-SAC as the base reinforcement learning algorithm for the expert policy, which involves two critic networks and two critic target networks. All four of these have the same architecture, with access to the true context c. This is possible since the critics are not needed at deployment, only during training. We use 32 quantiles within the critic networks of QR-SAC.

The observation o is a large vector of relevant information, similar to the one used by [WBK+22]. The action a has a dimension of 2 for Gran Turismo, and varying

Table C.2. Training hyperparameters for Gran Turismo and MuJoCo experiments.

Hyperparameter	Gran Turismo	MuJoCo
Optimizer	Adam [KB15]	Adam
Batch size (B)	1024	32
History length (H)	50	50
History adapter learning rate (α_{ad})	2.5×10^{-4}	3×10^{-4}
SAC learning rate for π^{ex} and critics (α_{SAC})	2.5×10^{-5}	3×10^{-4}
Target critics update rate (τ)	0.005	0.005
Global norm of critics gradient clipping	10	10
Discount factor (γ)	0.9896	0.99
SAC entropy temperature [HZAL18]	0.01	0.01
Number of quantiles	32	32

Algorithm 3 SPARC: **S**ingle-**P**hase **A**daptation for **R**obust **C**ontrol

Require: Environment $\mathcal{M}$ with latent context c (observable *only during training*), history length H, replay buffer $\mathcal{D}$, target update rate τ, learning rates $\alpha_{\text{SAC}}, \alpha_{ad}$

Initialise:

Expert policy π^{ex} with context encoder ψ
Adapter policy π^{ad} with history adapter ϕ
Critic networks Q^{ex} and target critics $\tilde{Q}^{ex} \leftarrow Q^{ex}$
History $h \leftarrow \mathbf{0}^H$
Sample IND context c for environment $\mathcal{M}$
for $t \leftarrow 1, 2, \ldots$ **do**

Act:

Observe o_t from environment $\mathcal{M}_c$
Sample action $a_t \sim \pi_\theta^{ad}(o_t, h_t)$
Concat (o_t, a_t) to history $h_t = (o_{t-H:t-1}, a_{t-H:t-1})$; (and pop last entry)
Execute a_t in $\mathcal{M}_c$, get reward r_t, next obs. o_{t+1}, done flag d_t
Store transition $(o_t, a_t, r_t, o_{t+1}, d_t, h_t, c) \rightarrow \mathcal{D}$

Update:

Sample minibatch $\mathcal{B}$ from $\mathcal{D}$
$\theta^{\mathcal{Q}} \leftarrow \theta^{\mathcal{Q}} - \alpha_{\text{SAC}} \nabla_{\theta^{\mathcal{Q}}} \mathcal{L}_{\text{SAC}}\left(\pi^{ex}, Q^{ex}, \tilde{Q}^{ex}\right)$
$\theta^{ex} \leftarrow \theta^{ex} + \alpha_{\text{SAC}} \nabla_{\theta^{ex}} J_{\text{SAC}}(\pi^{ex})$ Expert Policy update
$\theta^{\phi} \leftarrow \theta^{\phi} - \alpha_{ad} \nabla_{\theta^{\phi}} \left\| \phi(h) - \psi(c) \right\|_2^2$ History Adapter regression
$\tilde{Q}^{ex} \leftarrow (1-\tau)\tilde{Q}^{ex} + \tau Q^{ex}$ Polyak avg. of target critics
$\theta^{ad} \leftarrow \text{copy\_modules}(\theta^{ex})$ Copy obs. enc. & decision layers
if d_t **then**
Reset environment $\mathcal{M}$; sample new context $c \in \mathcal{C}_{\text{IND}}$ uniformly
Reset history $h \leftarrow \mathbf{0}^H$

numbers for the MuJoCo environments. For our MuJoCo experiments, we decrease the width of the FC layers from 2048 and 64 to 256 and 32, respectively.

C.1.3 Rollouts and Distributed Training

We train our algorithms asynchronously, performing rollouts on up to 20 distributed PlayStations simultaneously for Gran Turismo runs. The collected experience is added to a central replay buffer, from which a single GPU trains the neural networks. We use NVIDIA A100 GPUs with 40GB of memory. The wall-clock training time of an average Gran Turismo run is about 6 days, for MuJoCo it is approximately 14 hours. See [WBK+22] for further details on the Gran Turismo training setup.

At the start of each episode, we sample a contextual setting uniformly at random from the in-distribution (IND) set. For Gran Turismo, the agent's initial position is

Table C.3. Model architecture of SPARC. See Figure 4.1 for a schematic overview.

Layer Description	Input Dimensions	Output Dimensions
Expert Policy π^{ex}		
Observation Encoder		
FC: 2048 units, ReLU	o	2048
FC: 2048 units, ReLU	2048	2048
Context Encoder		
FC: 64 units, ReLU	c	64
FC: 64 units, ReLU	64	64
Decision Layers		
FC: 2048 units, ReLU	2112 (2048 + 64)	2048
FC: 2048 units, ReLU	2048	2048
FC: $2a$ units, Tanh	2048	$2a$
Adapter Policy π^{ad}		
Observation Encoder		
FC: 2048 units, ReLU	o	2048
FC: 2048 units, ReLU	2048	2048
History Adapter		
FC: 64 units, ReLU	$(o + a)$x50	64x50
Conv1D: 32 filters, kernel 8, stride 4, ReLU	64x50	32x13
Conv1D: 32 filters, kernel 5, stride 1, ReLU	32x13	32x13
Conv1D: 32 filters, kernel 5, stride 1, ReLU	32x13	32x13
Flatten	32x13	416
FC: 64 units, ReLU	416	64
Decision Layers		
FC: 2048 units, ReLU	2112 (2048 + 64)	2048
FC: 2048 units, ReLU	2048	2048
FC: $2a$ units, Tanh	2048	$2a$
Critic Networks		
Observation + Action Encoder		
FC: 2048 units, ReLU	$o + a$	2048
FC: 2048 units, ReLU	2048	2048
Context Encoder		
FC: 64 units, ReLU	c	64
FC: 64 units, ReLU	64	64
Decision Layers		
FC: 2048 units, ReLU	2112 (2048 + 64)	2048
FC: 2048 units, ReLU	2048	2048
FC: 32 units, identity	2048	32

(a) IND cars sampled from the ~400-vehicle training set. The Mazda, Lamborghini, and Lexus (from left to right) are representative, standard racing cars in *Gran Turismo 7*.

(b) OOD cars drawn from the ~100-vehicle test set. The racing kart, the Tomahawk, and the Toyota pickup truck (from left to right) are highly unusual outlier vehicles.

Figure C.1. Illustrative in-distribution (IND) versus out-of-distribution (OOD) vehicles used in our *Gran Turismo 7* experiments.

randomly chosen from on-track regions as well as from off-track zones to the left and right, extending up to 5% of the track width. The agent is oriented to face a point 30 meters ahead along the center line, and its initial speed is drawn uniformly from 0 to 104.607 km/h. Each episode runs for up to 150 seconds before being reset.

C.1.4 Race Cars

As detailed in Section 4.5, we hold out the 20% most *outlier* vehicles as a test set (OOD) and train on the 80% most *inlier* cars (IND). This means we train on about ~400 vehicles, while testing on ~100 unseen cars. In Figures C.1a and C.1b we present a few examples of IND and OOD vehicles, respectively. The sampled cars in these figures are from left to right:

Example cars IND (Figure C.1a)

- Mazda3 ’19
- Lamborghini Aventador LP 700-4
- Lexus RC F GT3 ’17

Example cars OOD (Figure C.1b)

- GT Racing Kart 125 Shifter
- Dodge SRT Tomahawk X Vision
- Toyota Tundra TRD Pro ’19

The **kart** (on the left in Figure C.1b) is one of the most outlier vehicles: very tiny and over-responsive. The **Tomahawk** (in the middle) is extremely fast and also famously difficult to control. The Toyota **pickup truck** (on the right) is an outlier in its mass, being significantly heavier and bulkier than most IND cars.

(a) Grand Valley **(b)** Nürburgring **(c)** Catalunya Rallycross

Figure C.2. Map layouts of the three tracks in Gran Turismo 7 used in our experiments. Note that these are not in the same scale, Nürburgring is by far the longest track. Catalunya Rallycross is the shortest and features a separate start ramp, which is not a part of our experiments. See Table 4.2 for the exact lengths of the tracks.

(a) Grand Valley **(b)** Nürburgring **(c)** Catalunya Rallycross

Figure C.3. In-game pictures of the three tracks in Gran Turismo 7 used in our experiments. The Grand Valley track is placed on a regular road, going over a bridge and through a tunnel. Nürburgring is a replica of the real-world race track located in Germany. Catalunya Rallycross is a mixed surface track, with tarmac and dirt areas, where specific dirt tires are required.

C.1.5 Race Tracks

We evaluate our approach on three diverse tracks in Gran Turismo 7: Grand Valley Highway 1, Nürburgring 24h, and Circuit de Barcelona - Catalunya Rallycross. These tracks vary widely in length, layout, and surface composition, enabling robust testing of our model's generalization capabilities.

- **Grand Valley** is a medium-length road track featuring tight turns, a tunnel section, and a bridge crossing. It provides a balanced mix of technical and high-speed sections.
- **Nürburgring** is the longest and one of the most complex tracks in the game, combining the Grand Prix circuit with the Nordschleife. Its high difficulty and track length pose a significant generalization challenge.

- **Catalunya Rallycross** is the shortest of the three. It introduces mixed terrain, with both asphalt and dirt sections. Driving here requires a distinct policy due to the lower traction and rapid transitions between surface types.

The map layouts of these tracks are shown in Figure C.2, while in-game screenshots can be found in Figure C.3. The characteristics are summarized in Table 4.2.

C.2 FURTHER ANALYSIS

To gain a deeper understanding of the contributions of our design choices, we perform an analysis of key factors, including the ideal rollout policy (Section C.2.1) and the optimal history length (Section C.2.2).

C.2.1 Selecting the Ideal Rollout Policy

In this ablation, we compare the performance of SPARC under two scenarios: (i) when experience is collected with the expert policy (π^{ex}), or (ii) when this is done by the adapter policy (π^{ad}). As detailed in Section 4.4.2, by default SPARC performs rollouts with the adapter policy to ensure a more on-policy style of learning.

Table C.4 presents results across all IND and OOD cars on three tracks. Although the differences are small, the results demonstrate that naively using π^{ex} for rollouts leads to slower lap times on 2 out of 3 tracks when looking at OOD generalization. Moreover, SPARC finishes every track with at least as many OOD cars as the alternative method.

C.2.2 Optimal History Length

We perform a sensitivity analysis of SPARC to different lengths of the observation-action history H. Recall from Figure 4.1 that SPARC's history adapter ϕ uses this recent experience to recognize its current contextual environment. The history input is defined as $h_t = (o_{t-H:t-1}, a_{t-H:t-1}) \in \mathcal{H}$, where $\mathcal{H} = (\mathcal{O} \times \mathcal{A})^H$ is the space of all possible observation-action histories.

The results in Figure C.4 show that a history length of $H = 50$ timesteps seems to be optimal for SPARC. Too few observation-action pairs do not provide enough information to robustly distinguish between contexts, whereas too many may distract the agent and waste computational resources.

Table C.4. Ablation on rollout policy. Results across all IND/OOD cars and 3 seeds. Mean ± s.e.m. of lap-time BIAI ratio and % successful laps. Especially on OOD, naively collecting experience with the expert policy (π^{ex}) generally does not perform as well as directly using the adapter policy (π^{ad}) for rollouts.

Method	IND		OOD	
	BIAI ratio (↓)	Success % (↑)	BIAI ratio (↓)	Success % (↑)
Grand Valley				
SPARC (π^{ex} rollouts)	**0.9922 ± 0.0034**	**100.0 ± 0.00**	**1.0417 ± 0.0024**	**98.06 ± 0.00**
SPARC	0.9999 ± 0.0061	99.76 ± 0.14	1.0491 ± 0.0055	**98.06 ± 0.56**
Nürburgring				
SPARC (π^{ex} rollouts)	**1.0107 ± 0.0150**	**96.93 ± 1.21**	1.1531 ± 0.0158	85.44 ± 1.12
SPARC	1.0254 ± 0.0061	95.87 ± 0.49	**1.1199 ± 0.0076**	**89.00 ± 0.86**
Catalunya Rallycross				
SPARC (π^{ex} rollouts)	0.9434 ± 0.0012	**100.00 ± 0.00**	0.9659 ± 0.0032	**100.00 ± 0.00**
SPARC	**0.9432 ± 0.0027**	**100.00 ± 0.00**	**0.9631 ± 0.0026**	**100.00 ± 0.00**

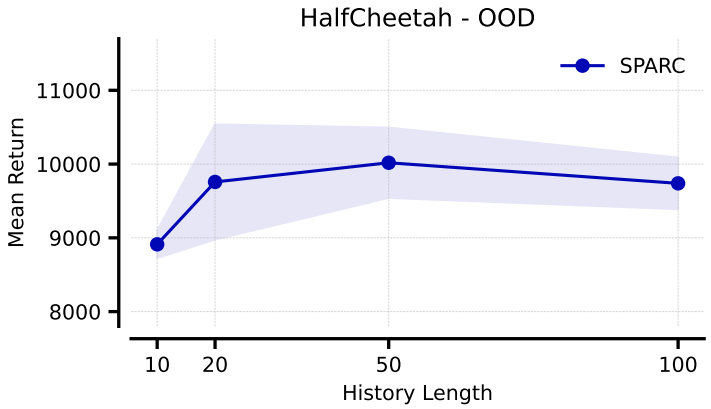


Figure C.4. Analysis of the optimal history length for SPARC, averaged over 5 seeds (± s.e.m.). We show the mean return over all tested out-of-distribution wind perturbations. A history length of 50 yields the highest average return.

APPENDIX D

Multiple Fields

In this Appendix belonging to Chapter 5, I provide additional details on the algorithm (Section D.1), hyperparameters (Section D.2), and extra results on Tiny-ImageNet and a parameter-matched language model study (Section D.3).

Code: `https://github.com/bramgrooten/neurotrails`

D.1 NEUROTRAILS ALGORITHM

The NeuroTrails algorithm, detailed in Algorithm 4, aims to efficiently enhance the performance of neural network ensembles while significantly reducing their parameter footprint. The approach splits a given base architecture into a shared backbone and multiple sparse heads, initializing each part with a target sparsity ratio. Throughout training, each head independently processes outputs from the backbone, enabling diverse predictions while leveraging common representation learning.

Periodically (every ΔT weight updates), the algorithm adjusts the network's topology through pruning and growing of parameters, controlled by a drop fraction p. During inference, predictions from individual heads are combined by averaging their output probabilities, resulting in a final aggregated prediction that leverages the strengths of each sparse pathway. See also Section 5.3 for further details.

Algorithm 4 NeuroTrails

Input: Base architecture $\mathcal{F}$, number of heads M, splitting point ℓ, sparsity ratio S, drop fraction p

Initialization Phase:

Split $\mathcal{F}$ at block ℓ into shared blocks $\mathcal{F}_s$ and independent heads $\mathcal{F}_h$

Initialize $\mathcal{F}$ to sparsity ratio S

Training Phase:

for each training iteration **do**

  $h_s = \mathcal{F}_s(x)$

  for each head $i \in \{1, \dots, M\}$ **do**

    $\hat{y}^i = \mathcal{F}_h^i(h_s)$

    $\mathcal{L}_i = \mathcal{L}(\hat{y}^i, y)$

  $\mathcal{L} = \frac{1}{M} \sum_{i=1}^{M} \mathcal{L}_i$

  $\theta_e \leftarrow \theta_e - \eta \nabla_{\theta_e} \mathcal{L}$

  if current iteration $\%\Delta T = 0$ **then**

    Prune p fraction of parameters layerwise

    Grow p fraction of parameters layerwise

Inference Phase:

Compute NeuroTrails prediction $\hat{y}$ by averaging the class probabilities predicted by each head j:

$$\hat{y} = \arg\max_i \left(\frac{1}{M} \sum_{j=1}^{M} \mathcal{F}_h^j(\mathcal{F}_s(x)) \right)_i$$

D.2 TRAINING SETTINGS

D.2.1 Code Repositories

Vision. For computer vision experiments, we build upon the codebases from Liu et al. [LYMP21] and Dettmers and Zettlemoyer [DZ19], implementing our method throughout their existing sparse training library. The codebase from Dettmers and Zettlemoyer [DZ19] is released under the MIT license.

Language. For language experiments, we use the codebases from Li et al. [LYL25] and Zhao et al. [ZZC+24] as a foundation, implementing NeuroTrails in conjunction with the LLaMA architectures based on HuggingFace's reproduction [WDS+19]. The codebase from Zhao et al. [ZZC+24] is licensed under Apache 2.0.

RL. In the reinforcement learning setup, we build upon the CleanRL codebase [HDY+22], adjusting the DQN architecture for NeuroTrails, TreeNet, and the Full Ensemble, as well as enabling independent training of these models. CleanRL is released under the MIT license.

D.2.2 Hyperparameters

We describe the hyperparameters for our experiments. Tables D.1 to D.3 present the settings of our computer vision, language modeling, and reinforcement learning experiments, respectively.

Computer Vision. For CIFAR-100 and Tiny-ImageNet, we train Wide-ResNet28-10 using SGD with momentum 0.9 and an initial learning rate of 0.1. The batch size is 128, and the L2 regularization constant is set to 0.0005. For ImageNet, we follow the standard training regime [NVI24]. We train ResNet-50 using SGD with a momentum of 0.875 and an initial learning rate of 0.256. The batch size is set to 256. We use L2 regularization with a fixed constant of 3.05e-05. For all computer vision datasets, the learning rate decreases by a factor of 10 after 25%, 50%, and 75% of training.

DST hyperparameters are set as follows: pruning and regrowing 50% of the available parameters at the beginning, with a cosine decay to 0 by the end of training. The ΔT topology update interval is set to 100 for CIFAR-100 and 1000 for ImageNet.

For the static sparse baseline of Section 5.5.1, we instantiate the model with the desired sparsity ratio at the start and subsequently train without adjusting the

Table D.1. Hyperparameters and settings for **computer vision** experiments.

Parameter	Value
Shared by all experiments	
optimizer	SGD with momentum
learning rate schedule	0.1× step decay at 25%, 50%, 75% of training
ensemble aggregation	soft voting (mean of probabilities)
CIFAR-100 and Tiny-ImageNet (all baselines)	
model	Wide-ResNet28-10
momentum	0.9
initial learning rate	0.1
batch size	128
weight decay (L2)	$5 \cdot 10^{-4}$
training device CIFAR-100	1 × NVIDIA V100 (16GB memory)
training device Tiny-ImageNet	4 × NVIDIA A100 (40GB memory)
approx. training time	6.25h (CIFAR-100), 5.5h (Tiny-ImageNet)
ImageNet (all baselines)	
model	ResNet-50
momentum	0.875
initial learning rate	0.256
batch size	256
weight decay (L2)	$3.05 \cdot 10^{-5}$
training device	4 × NVIDIA A100 (40GB memory)
approx. training time	53h
Static Sparse baseline	
sparsity ratio	varying (Section 5.5.1)
sparsity initialization	ER
topology update interval (ΔT)	∞ (no change)
NeuroTrails	
sparsity ratio	varying (Section 5.4)
sparsity initialization	ER
DST drop fraction	$0.5 \cdot \text{cosine\_decay}(t)$
DST grow method	gradient (RigL)
DST prune method	magnitude-based
topology update interval (ΔT)	100 (CIFAR-100), 1000 (ImageNet)
blocks in head	8 (CIFAR-100), 10 (ImageNet)
Pruning baseline	
sparsity ratio	varying (Section 5.5.1)
pruning method	global magnitude
pre-pruning phase	250 epochs
fine-tuning phase	250 epochs
TreeNet	
blocks in head	8 (CIFAR-100), 10 (ImageNet)
Full Ensemble	
training paradigm	Independent training

topology. For the pruning baseline, we employ global magnitude pruning to achieve the target sparsity ratio after the first 50% of training. After reaching the desired sparsity, we fine-tune the model for the remaining duration without making any further changes to its topology.

For ensemble training, we use independent training, shown to work well by Jeffares et al. [JLCvdS23], and train networks separately. At test time, we gather each network's predictions and average their raw probabilities, i.e., soft voting.

Table D.2. Hyperparameters and settings for **language modeling** experiments.

Parameter	Value
Shared by all experiments	
model size	130M, 350M
optimizer	Adam
learning rate	$1.5e-3$ for 130M, $5e-4$ for 350M
learning rate schedule	cosine decay (min LR: 0.1×base) + warmup
learning rate warmup	10% of update steps
weight decay	0 (no decay)
ensemble aggregation	soft voting (mean of probabilities)
batch size	512
vocabulary size	32,000
max sequence length	1024
data type	bfloat16
training device	4 × NVIDIA A100 (40GB memory)
approx. training time	7.5h (130M), 40h (350M)
NeuroTrails	
sparsity ratio	0.1
sparsity initialization	ER with attention projections dense
DST drop fraction	$0.5 \cdot \text{cosine\_decay}(t)$
DST grow method	gradient (RigL)
DST prune method	soft magnitude
soft magnitude temperature	3.0
topology update interval (ΔT)	50 steps
blocks in head	8 (130M), 16 (350M)
TreeNet	
blocks in head	8 (130M), 16 (350M)
Full Ensemble	
training paradigm	Independent training

Language Modeling. For our language modeling experiments on the C4 corpus with LLaMA-130M/350M, we present the hyperparameters in Table D.2. We train with Adam using a learning rate cosine-decay schedule (minimum LR set to 10% of the base) and a linear warmup over the first 10% of update steps. The batch size is 512 tokens, and we run on four A100 GPUs.

Our NeuroTrails models use dynamic sparse training: every $\Delta T = 50$ steps we drop and regrow a fraction p of the weights, decaying p to zero by the end of training. We allocate 8 or 16 transformer blocks per head, matching the $\sim 1/3$ shared-backbone setting used in our vision experiments. This configuration lets each head discover and adapt its own topology while respecting the overall FLOP budget.

As a reference, we split the TreeNet baseline likewise into a shared backbone and 8 or 16-block heads. For the dense-ensemble baseline, we train three independent LLaMA-130M models from scratch under the same schedule and aggregate their outputs via the same soft-voting scheme.

Table D.3. Hyperparameters and settings for **reinforcement learning** experiments.

Parameter	Value
Shared by all experiments	
optimizer	Adam
learning rate	1e−4
discount γ	0.99
batch size	32
replay buffer	1,000,000 transitions
learning starts	80,000 steps
train frequency	every 4 env steps
target net update period	every 1,000 env steps
target net update rate τ	1.0 (hard copy)
ϵ-greedy	linear decay $1.0 \rightarrow 0.01$ over 10% of total steps
NeuroTrails	
sparsity ratio	0.8
sparsity initialization	ERK
DST drop fraction	$0.5 \cdot \text{cosine\_decay}(t)$
DST grow method	random (SET)
DST prune method	magnitude
topology update interval (ΔT)	2000 grad steps
blocks in head	4
TreeNet	
blocks in head	4

Reinforcement Learning. We train for 10M steps (40M frames) on the Atari environments: `Asterix`, `BeamRider`, `Breakout`, `Seaquest`, `SpaceInvaders`, and `UpNDown`. Every 100K steps we evaluate the model on 10 episodes. We average these evaluation returns over the last 10% of training (100 eval episodes in total) for more reliable results, following [GEEC22, GSD+23]. We take the interquartile mean to be more robust against outliers, as proposed by Agarwal et al. [ASC+21]. We train with standard Atari preprocessing (no-op starts, frameskip 4, reward clipping) as is default in CleanRL [HDY+22]. Hyperparameters are presented in Table D.3.

As mentioned in Section 5.4.1, we train the heads independently, but sample actions in the environment jointly as we aim to collect the best data. We find that training independently is crucial, but joint or independent sampling does not make a large difference. For NeuroTrails and TreeNet we split the backbone after the three convolutional layers. NeuroTrails adjusts the sparse topology after every $\Delta T = 2000$ weight updates, with an initial drop fraction $p = 0.5$ annealed through cosine decay. We simply use magnitude pruning and random growth, as SET [MMS+18] has been shown to work just as well as RigL [EGM+20] in RL [GEEC22].

D.2.3 Training Schedules

Building on the observation that dense models tend to overfit once training is prolonged, whereas sparse networks keep improving as they are still refining both weights and topology [LYMP21], we follow the recipe of Evci et al. [EGM+20]. We extend the training of sparse variants by at most $1/(1-S)$, so that the total FLOPs never exceed those of the dense counterpart. Exact training schedules appear in Tables D.4 to D.6.

Table D.4. Training cost comparison on CIFAR-100 (Wide-ResNet28-10). Baselines marked with * are from [HJF+21], ** from [LCA+22], and *** from [LL24].

	Train Epochs	Train FLOPs (↓)
Single Dense *	250	3.6e17
MIMO ($M = 3$) *	250	1.00×
EDST Ensemble ($M = 7$, $S = 0.9$) **	850	0.57×
DST Ensemble ($M = 3$, $S = 0.8$) **	3×250	1.01×
Batch Ensemble ($M = 4$) *	250	1.10×
NFE ($M = 3$) ***	200	1.02×
TreeNet ($M = 3$)	250	2.91×
Full Ensemble ($M = 3$)	3×250	3.00×
NeuroTrails ($M = 3$, $S = 0.8$)	450	0.85×
NeuroTrails ($M = 5$, $S = 0.9$)	450	0.67×

Table D.5. Training cost comparison on ImageNet (ResNet-50). Baselines marked with * are from [HJF+21], ** from [LCA+22].

	Train Epochs	Train FLOPs (↓)
Single Dense *	90	4.8e18
MIMO ($M = 2$, $\rho = 0.6$) *	150	1.00×
EDST Ensemble ($M = 4$, $S = 0.8$) **	310	0.87×
DST Ensemble ($M = 2$, $S = 0.8$) **	2×200	1.12×
Batch Ensemble ($M = 4$) *	4×135	1.10×
TreeNet ($M = 3$)	180	2.91×
Full Ensemble ($M = 4$) *	4×90	4.00×
NeuroTrails ($M = 3$, $S = 0.7$)	270	1.10×

In Table D.4 we report the number of epochs and relative training FLOPs on CIFAR-100 with Wide-ResNet28-10. The dense model runs for 250 epochs (3.6e17 FLOPs). Sparse methods such as EDST ($S = 0.9$) and DST ($S = 0.8$) run up to 850 or 750 total epochs to compensate for their reduced per-epoch cost, while other baselines (MIMO, BatchEnsemble, NFE, TreeNet) stay close to 250. NeuroTrails is trained for 450 epochs, calibrated to maintain training FLOPs significantly below a single network.

Table D.5 shows the analogous schedule on ImageNet (ResNet-50). The single-model baseline uses 90 epochs (4.8e18 FLOPs); EDST and DST extend to 310 and 400 total epochs respectively, whereas NeuroTrails ($S = 0.7$) runs for 270 epochs—staying close to a single dense model's compute budget. Notably, most baselines tune their schedules so that total training FLOPs remain close to the 1.0× dense reference. We recognize that reported training lengths vary substantially across papers; to ensure fidelity to each comparison, we simply report each method's published schedule when taking values from the original works. We encourage the ensembling field to always publish the full training schedules of all baselines and adopt more consistent training protocols, to enable clearer comparisons.

For C4 pretraining with LLaMA models, Table D.6 compares the number of weight-update steps, total tokens seen, and training FLOPs across different methods. The dense baseline for LLaMA-130M performs 10,000 updates (1.0B tokens), while NeuroTrails ($S = 0.1$) scales to 11,111 steps (1.1B tokens), precisely matching TreeNet's training FLOPs via the $1/(1-S)$ rule. A similar scaling applies to the larger LLaMA-350M model, where NeuroTrails with sparsity 0.1 takes 44,444 steps (4.4B tokens), again equating to the training FLOPs of the dense counterpart.

Finally, Table D.7 provides the environment steps and FLOPs of our reinforcement learning experiments. We use 10M environment steps (40M frames) for all methods, achieving 0.59× training FLOPs for NeuroTrails compared to a single dense model.

Table D.6. Compute comparison on C4 pretraining (LLaMA-130M/350M).

	Train Updates	Tokens seen	Training FLOPs (↓)
LLaMA-130M			
Single Dense	10,000	1.0B	3.5e18
TreeNet ($M = 3$)	10,000	1.0B	2.21×
Full Ensemble ($M = 3$)	3×10,000	1.0B	3.00×
NeuroTrails ($M = 3$, $S = 0.1$)	11,111	1.1B	2.21×
LLaMA-350M			
Single Dense	40,000	4.0B	3.5e18
TreeNet ($M = 3$)	40,000	4.0B	2.27×
Full Ensemble ($M = 3$)	3×40,000	4.0B	3.00×
NeuroTrails ($M = 3$, $S = 0.1$)	44,444	4.4B	2.27×

Table D.7. Training cost comparison on Atari (DQN).

	Environment steps	Train FLOPs (↓)
Single Dense	10M	8.6e15
TreeNet ($M = 3$)	10M	2.97×
Full Ensemble ($M = 3$)	3×10M	3.00×
NeuroTrails ($M = 3$) ($S = 0.8$)	10M	0.59×

D.3 ADDITIONAL RESULTS

Parameter-Matched Language Model Study To isolate the effect of *sparsity + multi-head sharing* from sheer model size, we build a parameter-matched NeuroTrails variant whose total parameter budget is essentially identical to that of a larger dense model. We compare a single dense LLaMA-250M model with a NeuroTrails variant of LLaMA-130M, which we designed to match the number of parameters. It uses 3 heads, with just 7 blocks per head (instead of our default 8), and a sparsity ratio of 13%. We train both for exactly the same number of update steps (10k), meaning both models see approximately 1B training tokens. As shown in Table D.8, NeuroTrails delivers a modestly lower validation perplexity (26.48 vs. 26.59), despite having slightly fewer parameters than the 250M dense baseline.

Table D.8. Size–efficiency comparison: a 250M-parameter single dense model versus a NeuroTrails variant with approximately the same parameter budget.

	Perplexity (↓)	**Parameters (↓)**	**Train FLOPs (↓)**	**Infer. FLOPs (↓)**
Single Dense	26.59	247.37M	7.0e18	4.56e11
NeuroTrails ($M = 3$, $S = 0.13$)	**26.48**	245.66M	1.0×	1.0×

§

SIKS Dissertation Series

2016
01 Syed Saiden Abbas (RUN), Recognition of Shapes by Humans and Machines
02 Michiel Christiaan Meulendijk (UU), Optimizing medication reviews through decision support: prescribing a better pill to swallow
03 Maya Sappelli (RUN), Knowledge Work in Context: User Centered Knowledge Worker Support
04 Laurens Rietveld (VUA), Publishing and Consuming Linked Data
05 Evgeny Sherkhonov (UvA), Expanded Acyclic Queries: Containment and an Application in Explaining Missing Answers
06 Michel Wilson (TUD), Robust scheduling in an uncertain environment
07 Jeroen de Man (VUA), Measuring and modeling negative emotions for virtual training
08 Matje van de Camp (TiU), A Link to the Past: Constructing Historical Social Networks from Unstructured Data
09 Archana Nottamkandath (VUA), Trusting Crowdsourced Information on Cultural Artefacts
10 George Karafotias (VUA), Parameter Control for Evolutionary Algorithms
11 Anne Schuth (UvA), Search Engines that Learn from Their Users
12 Max Knobbout (UU), Logics for Modelling and Verifying Normative Multi-Agent Systems
13 Nana Baah Gyan (VUA), The Web, Speech Technologies and Rural Development in West Africa - An ICT4D Approach
14 Ravi Khadka (UU), Revisiting Legacy Software System Modernization
15 Steffen Michels (RUN), Hybrid Probabilistic Logics - Theoretical Aspects, Algorithms and Experiments
16 Guangliang Li (UvA), Socially Intelligent Autonomous Agents that Learn from Human Reward
17 Berend Weel (VUA), Towards Embodied Evolution of Robot Organisms
18 Albert Meroño Peñuela (VUA), Refining Statistical Data on the Web
19 Julia Efremova (TU/e), Mining Social Structures from Genealogical Data
20 Daan Odijk (UvA), Context & Semantics in News & Web Search
21 Alejandro Moreno Célleri (UT), From Traditional to Interactive Playspaces: Automatic Analysis of Player Behavior in the Interactive Tag Playground
22 Grace Lewis (VUA), Software Architecture Strategies for Cyber-Foraging Systems
23 Fei Cai (UvA), Query Auto Completion in Information Retrieval
24 Brend Wanders (UT), Repurposing and Probabilistic Integration of Data; An Iterative and data model independent approach
25 Julia Kiseleva (TU/e), Using Contextual Information to Understand Searching and Browsing Behavior
26 Dilhan Thilakarathne (VUA), In or Out of Control: Exploring Computational Models to Study the Role of Human Awareness and Control in Behavioural Choices, with Applications in Aviation and Energy Management Domains
27 Wen Li (TUD), Understanding Geo-spatial Information on Social Media

28 Mingxin Zhang (TUD), Large-scale Agent-based Social Simulation - A study on epidemic prediction and control
29 Nicolas Höning (TUD), Peak reduction in decentralised electricity systems - Markets and prices for flexible planning
30 Ruud Mattheij (TiU), The Eyes Have It
31 Mohammad Khelghati (UT), Deep web content monitoring
32 Eelco Vriezekolk (UT), Assessing Telecommunication Service Availability Risks for Crisis Organisations
33 Peter Bloem (UvA), Single Sample Statistics, exercises in learning from just one example
34 Dennis Schunselaar (TU/e), Configurable Process Trees: Elicitation, Analysis, and Enactment
35 Zhaochun Ren (UvA), Monitoring Social Media: Summarization, Classification and Recommendation
36 Daphne Karreman (UT), Beyond R2D2: The design of nonverbal interaction behavior optimized for robot-specific morphologies
37 Giovanni Sileno (UvA), Aligning Law and Action - a conceptual and computational inquiry
38 Andrea Minuto (UT), Materials that Matter - Smart Materials meet Art & Interaction Design
39 Merijn Bruijnes (UT), Believable Suspect Agents; Response and Interpersonal Style Selection for an Artificial Suspect
40 Christian Detweiler (TUD), Accounting for Values in Design
41 Thomas King (TUD), Governing Governance: A Formal Framework for Analysing Institutional Design and Enactment Governance
42 Spyros Martzoukos (UvA), Combinatorial and Compositional Aspects of Bilingual Aligned Corpora
43 Saskia Koldijk (RUN), Context-Aware Support for Stress Self-Management: From Theory to Practice
44 Thibault Sellam (UvA), Automatic Assistants for Database Exploration
45 Bram van de Laar (UT), Experiencing Brain-Computer Interface Control
46 Jorge Gallego Perez (UT), Robots to Make you Happy
47 Christina Weber (UL), Real-time foresight - Preparedness for dynamic innovation networks
48 Tanja Buttler (TUD), Collecting Lessons Learned
49 Gleb Polevoy (TUD), Participation and Interaction in Projects. A Game-Theoretic Analysis
50 Yan Wang (TiU), The Bridge of Dreams: Towards a Method for Operational Performance Alignment in IT-enabled Service Supply Chains

---

2017 01 Jan-Jaap Oerlemans (UL), Investigating Cybercrime
02 Sjoerd Timmer (UU), Designing and Understanding Forensic Bayesian Networks using Argumentation
03 Daniël Harold Telgen (UU), Grid Manufacturing; A Cyber-Physical Approach with Autonomous Products and Reconfigurable Manufacturing Machines
04 Mrunal Gawade (CWI), Multi-core Parallelism in a Column-store
05 Mahdieh Shadi (UvA), Collaboration Behavior
06 Damir Vandic (EUR), Intelligent Information Systems for Web Product Search
07 Roel Bertens (UU), Insight in Information: from Abstract to Anomaly
08 Rob Konijn (VUA), Detecting Interesting Differences:Data Mining in Health Insurance Data using Outlier Detection and Subgroup Discovery

09 Dong Nguyen (UT), Text as Social and Cultural Data: A Computational Perspective on Variation in Text
10 Robby van Delden (UT), (Steering) Interactive Play Behavior
11 Florian Kunneman (RUN), Modelling patterns of time and emotion in Twitter #anticipointment
12 Sander Leemans (TU/e), Robust Process Mining with Guarantees
13 Gijs Huisman (UT), Social Touch Technology - Extending the reach of social touch through haptic technology
14 Shoshannah Tekofsky (TiU), You Are Who You Play You Are: Modelling Player Traits from Video Game Behavior
15 Peter Berck (RUN), Memory-Based Text Correction
16 Aleksandr Chuklin (UvA), Understanding and Modeling Users of Modern Search Engines
17 Daniel Dimov (UL), Crowdsourced Online Dispute Resolution
18 Ridho Reinanda (UvA), Entity Associations for Search
19 Jeroen Vuurens (UT), Proximity of Terms, Texts and Semantic Vectors in Information Retrieval
20 Mohammadbashir Sedighi (TUD), Fostering Engagement in Knowledge Sharing: The Role of Perceived Benefits, Costs and Visibility
21 Jeroen Linssen (UT), Meta Matters in Interactive Storytelling and Serious Gaming (A Play on Worlds)
22 Sara Magliacane (VUA), Logics for causal inference under uncertainty
23 David Graus (UvA), Entities of Interest — Discovery in Digital Traces
24 Chang Wang (TUD), Use of Affordances for Efficient Robot Learning
25 Veruska Zamborlini (VUA), Knowledge Representation for Clinical Guidelines, with applications to Multimorbidity Analysis and Literature Search
26 Merel Jung (UT), Socially intelligent robots that understand and respond to human touch
27 Michiel Joosse (UT), Investigating Positioning and Gaze Behaviors of Social Robots: People's Preferences, Perceptions and Behaviors
28 John Klein (VUA), Architecture Practices for Complex Contexts
29 Adel Alhuraibi (TiU), From IT-BusinessStrategic Alignment to Performance: A Moderated Mediation Model of Social Innovation, and Enterprise Governance of IT"
30 Wilma Latuny (TiU), The Power of Facial Expressions
31 Ben Ruijl (UL), Advances in computational methods for QFT calculations
32 Thaer Samar (RUN), Access to and Retrievability of Content in Web Archives
33 Brigit van Loggem (OU), Towards a Design Rationale for Software Documentation: A Model of Computer-Mediated Activity
34 Maren Scheffel (OU), The Evaluation Framework for Learning Analytics
35 Martine de Vos (VUA), Interpreting natural science spreadsheets
36 Yuanhao Guo (UL), Shape Analysis for Phenotype Characterisation from High-throughput Imaging
37 Alejandro Montes Garcia (TU/e), WiBAF: A Within Browser Adaptation Framework that Enables Control over Privacy
38 Alex Kayal (TUD), Normative Social Applications
39 Sara Ahmadi (RUN), Exploiting properties of the human auditory system and compressive sensing methods to increase noise robustness in ASR

40 Altaf Hussain Abro (VUA), Steer your Mind: Computational Exploration of Human Control in Relation to Emotions, Desires and Social Support For applications in human-aware support systems
41 Adnan Manzoor (VUA), Minding a Healthy Lifestyle: An Exploration of Mental Processes and a Smart Environment to Provide Support for a Healthy Lifestyle
42 Elena Sokolova (RUN), Causal discovery from mixed and missing data with applications on ADHD datasets
43 Maaike de Boer (RUN), Semantic Mapping in Video Retrieval
44 Garm Lucassen (UU), Understanding User Stories - Computational Linguistics in Agile Requirements Engineering
45 Bas Testerink (UU), Decentralized Runtime Norm Enforcement
46 Jan Schneider (OU), Sensor-based Learning Support
47 Jie Yang (TUD), Crowd Knowledge Creation Acceleration
48 Angel Suarez (OU), Collaborative inquiry-based learning

2018 01 Han van der Aa (VUA), Comparing and Aligning Process Representations
02 Felix Mannhardt (TU/e), Multi-perspective Process Mining
03 Steven Bosems (UT), Causal Models For Well-Being: Knowledge Modeling, Model-Driven Development of Context-Aware Applications, and Behavior Prediction
04 Jordan Janeiro (TUD), Flexible Coordination Support for Diagnosis Teams in Data-Centric Engineering Tasks
05 Hugo Huurdeman (UvA), Supporting the Complex Dynamics of the Information Seeking Process
06 Dan Ionita (UT), Model-Driven Information Security Risk Assessment of Socio-Technical Systems
07 Jieting Luo (UU), A formal account of opportunism in multi-agent systems
08 Rick Smetsers (RUN), Advances in Model Learning for Software Systems
09 Xu Xie (TUD), Data Assimilation in Discrete Event Simulations
10 Julienka Mollee (VUA), Moving forward: supporting physical activity behavior change through intelligent technology
11 Mahdi Sargolzaei (UvA), Enabling Framework for Service-oriented Collaborative Networks
12 Xixi Lu (TU/e), Using behavioral context in process mining
13 Seyed Amin Tabatabaei (VUA), Computing a Sustainable Future
14 Bart Joosten (TiU), Detecting Social Signals with Spatiotemporal Gabor Filters
15 Naser Davarzani (UM), Biomarker discovery in heart failure
16 Jaebok Kim (UT), Automatic recognition of engagement and emotion in a group of children
17 Jianpeng Zhang (TU/e), On Graph Sample Clustering
18 Henriette Nakad (UL), De Notaris en Private Rechtspraak
19 Minh Duc Pham (VUA), Emergent relational schemas for RDF
20 Manxia Liu (RUN), Time and Bayesian Networks
21 Aad Slootmaker (OU), EMERGO: a generic platform for authoring and playing scenario-based serious games
22 Eric Fernandes de Mello Araújo (VUA), Contagious: Modeling the Spread of Behaviours, Perceptions and Emotions in Social Networks
23 Kim Schouten (EUR), Semantics-driven Aspect-Based Sentiment Analysis
24 Jered Vroon (UT), Responsive Social Positioning Behaviour for Semi-Autonomous Telepresence Robots
25 Riste Gligorov (VUA), Serious Games in Audio-Visual Collections

26 Roelof Anne Jelle de Vries (UT),Theory-Based and Tailor-Made: Motivational Messages for Behavior Change Technology
27 Maikel Leemans (TU/e), Hierarchical Process Mining for Scalable Software Analysis
28 Christian Willemse (UT), Social Touch Technologies: How they feel and how they make you feel
29 Yu Gu (TiU), Emotion Recognition from Mandarin Speech
30 Wouter Beek (VUA), The "K" in "semantic web" stands for "knowledge": scaling semantics to the web

---

2019 01 Rob van Eijk (UL),Web privacy measurement in real-time bidding systems. A graph-based approach to RTB system classification
02 Emmanuelle Beauxis Aussalet (CWI, UU), Statistics and Visualizations for Assessing Class Size Uncertainty
03 Eduardo Gonzalez Lopez de Murillas (TU/e), Process Mining on Databases: Extracting Event Data from Real Life Data Sources
04 Ridho Rahmadi (RUN), Finding stable causal structures from clinical data
05 Sebastiaan van Zelst (TU/e), Process Mining with Streaming Data
06 Chris Dijkshoorn (VUA), Nichesourcing for Improving Access to Linked Cultural Heritage Datasets
07 Soude Fazeli (TUD), Recommender Systems in Social Learning Platforms
08 Frits de Nijs (TUD), Resource-constrained Multi-agent Markov Decision Processes
09 Fahimeh Alizadeh Moghaddam (UvA), Self-adaptation for energy efficiency in software systems
10 Qing Chuan Ye (EUR), Multi-objective Optimization Methods for Allocation and Prediction
11 Yue Zhao (TUD), Learning Analytics Technology to Understand Learner Behavioral Engagement in MOOCs
12 Jacqueline Heinerman (VUA), Better Together
13 Guanliang Chen (TUD), MOOC Analytics: Learner Modeling and Content Generation
14 Daniel Davis (TUD), Large-Scale Learning Analytics: Modeling Learner Behavior & Improving Learning Outcomes in Massive Open Online Courses
15 Erwin Walraven (TUD), Planning under Uncertainty in Constrained and Partially Observable Environments
16 Guangming Li (TU/e), Process Mining based on Object-Centric Behavioral Constraint (OCBC) Models
17 Ali Hurriyetoglu (RUN),Extracting actionable information from microtexts
18 Gerard Wagenaar (UU), Artefacts in Agile Team Communication
19 Vincent Koeman (TUD), Tools for Developing Cognitive Agents
20 Chide Groenouwe (UU), Fostering technically augmented human collective intelligence
21 Cong Liu (TU/e), Software Data Analytics: Architectural Model Discovery and Design Pattern Detection
22 Martin van den Berg (VUA),Improving IT Decisions with Enterprise Architecture
23 Qin Liu (TUD), Intelligent Control Systems: Learning, Interpreting, Verification
24 Anca Dumitrache (VUA), Truth in Disagreement - Crowdsourcing Labeled Data for Natural Language Processing
25 Emiel van Miltenburg (VUA), Pragmatic factors in (automatic) image description
26 Prince Singh (UT), An Integration Platform for Synchromodal Transport
27 Alessandra Antonaci (OU), The Gamification Design Process applied to (Massive) Open Online Courses

28 Esther Kuindersma (UL), Cleared for take-off: Game-based learning to prepare airline pilots for critical situations
29 Daniel Formolo (VUA), Using virtual agents for simulation and training of social skills in safety-critical circumstances
30 Vahid Yazdanpanah (UT), Multiagent Industrial Symbiosis Systems
31 Milan Jelisavcic (VUA), Alive and Kicking: Baby Steps in Robotics
32 Chiara Sironi (UM), Monte-Carlo Tree Search for Artificial General Intelligence in Games
33 Anil Yaman (TU/e), Evolution of Biologically Inspired Learning in Artificial Neural Networks
34 Negar Ahmadi (TU/e), EEG Microstate and Functional Brain Network Features for Classification of Epilepsy and PNES
35 Lisa Facey-Shaw (OU), Gamification with digital badges in learning programming
36 Kevin Ackermans (OU), Designing Video-Enhanced Rubrics to Master Complex Skills
37 Jian Fang (TUD), Database Acceleration on FPGAs
38 Akos Kadar (OU), Learning visually grounded and multilingual representations

---

2020 01 Armon Toubman (UL), Calculated Moves: Generating Air Combat Behaviour
02 Marcos de Paula Bueno (UL), Unraveling Temporal Processes using Probabilistic Graphical Models
03 Mostafa Deghani (UvA), Learning with Imperfect Supervision for Language Understanding
04 Maarten van Gompel (RUN), Context as Linguistic Bridges
05 Yulong Pei (TU/e), On local and global structure mining
06 Preethu Rose Anish (UT), Stimulation Architectural Thinking during Requirements Elicitation - An Approach and Tool Support
07 Wim van der Vegt (OU), Towards a software architecture for reusable game components
08 Ali Mirsoleimani (UL),Structured Parallel Programming for Monte Carlo Tree Search
09 Myriam Traub (UU), Measuring Tool Bias and Improving Data Quality for Digital Humanities Research
10 Alifah Syamsiyah (TU/e), In-database Preprocessing for Process Mining
11 Sepideh Mesbah (TUD), Semantic-Enhanced Training Data AugmentationMethods for Long-Tail Entity Recognition Models
12 Ward van Breda (VUA), Predictive Modeling in E-Mental Health: Exploring Applicability in Personalised Depression Treatment
13 Marco Virgolin (CWI), Design and Application of Gene-pool Optimal Mixing Evolutionary Algorithms for Genetic Programming
14 Mark Raasveldt (CWI/UL), Integrating Analytics with Relational Databases
15 Konstantinos Georgiadis (OU), Smart CAT: Machine Learning for Configurable Assessments in Serious Games
16 Ilona Wilmont (RUN), Cognitive Aspects of Conceptual Modelling
17 Daniele Di Mitri (OU), The Multimodal Tutor: Adaptive Feedback from Multimodal Experiences
18 Georgios Methenitis (TUD), Agent Interactions & Mechanisms in Markets with Uncertainties: Electricity Markets in Renewable Energy Systems
19 Guido van Capelleveen (UT), Industrial Symbiosis Recommender Systems
20 Albert Hankel (VUA), Embedding Green ICT Maturity in Organisations
21 Karine da Silva Miras de Araujo (VUA), Where is the robot?: Life as it could be
22 Maryam Masoud Khamis (RUN), Understanding complex systems implementation through a modeling approach: the case of e-government in Zanzibar

23 Rianne Conijn (UT), The Keys to Writing: A writing analytics approach to studying writing processes using keystroke logging
24 Lenin da Nóbrega Medeiros (VUA/RUN), How are you feeling, human? Towards emotionally supportive chatbots
25 Xin Du (TU/e), The Uncertainty in Exceptional Model Mining
26 Krzysztof Leszek Sadowski (UU), GAMBIT: Genetic Algorithm for Model-Based mixed-Integer opTimization
27 Ekaterina Muravyeva (TUD), Personal data and informed consent in an educational context
28 Bibeg Limbu (TUD), Multimodal interaction for deliberate practice: Training complex skills with augmented reality
29 Ioan Gabriel Bucur (RUN), Being Bayesian about Causal Inference
30 Bob Zadok Blok (UL), Creatief, Creatiever, Creatiefst
31 Gongjin Lan (VUA), Learning better – From Baby to Better
32 Jason Rhuggenaath (TU/e), Revenue management in online markets: pricing and online advertising
33 Rick Gilsing (TU/e), Supporting service-dominant business model evaluation in the context of business model innovation
34 Anna Bon (UM), Intervention or Collaboration? Redesigning Information and Communication Technologies for Development
35 Siamak Farshidi (UU), Multi-Criteria Decision-Making in Software Production

---

2021 01 Francisco Xavier Dos Santos Fonseca (TUD),Location-based Games for Social Interaction in Public Space
02 Rijk Mercuur (TUD), Simulating Human Routines: Integrating Social Practice Theory in Agent-Based Models
03 Seyyed Hadi Hashemi (UvA), Modeling Users Interacting with Smart Devices
04 Ioana Jivet (OU), The Dashboard That Loved Me: Designing adaptive learning analytics for self-regulated learning
05 Davide Dell'Anna (UU), Data-Driven Supervision of Autonomous Systems
06 Daniel Davison (UT), "Hey robot, what do you think?" How children learn with a social robot
07 Armel Lefebvre (UU), Research data management for open science
08 Nardie Fanchamps (OU), The Influence of Sense-Reason-Act Programming on Computational Thinking
09 Cristina Zaga (UT), The Design of Robothings. Non-Anthropomorphic and Non-Verbal Robots to Promote Children's Collaboration Through Play
10 Quinten Meertens (UvA), Misclassification Bias in Statistical Learning
11 Anne van Rossum (UL), Nonparametric Bayesian Methods in Robotic Vision
12 Lei Pi (UL), External Knowledge Absorption in Chinese SMEs
13 Bob R. Schadenberg (UT), Robots for Autistic Children: Understanding and Facilitating Predictability for Engagement in Learning
14 Negin Samaeemofrad (UL), Business Incubators: The Impact of Their Support
15 Onat Ege Adali (TU/e), Transformation of Value Propositions into Resource Re-Configurations through the Business Services Paradigm
16 Esam A. H. Ghaleb (UM), Bimodal emotion recognition from audio-visual cues
17 Dario Dotti (UM), Human Behavior Understanding from motion and bodily cues using deep neural networks

18 Remi Wieten (UU), Bridging the Gap Between Informal Sense-Making Tools and Formal Systems - Facilitating the Construction of Bayesian Networks and Argumentation Frameworks
19 Roberto Verdecchia (VUA), Architectural Technical Debt: Identification and Management
20 Masoud Mansoury (TU/e), Understanding and Mitigating Multi-Sided Exposure Bias in Recommender Systems
21 Pedro Thiago Timbó Holanda (CWI), Progressive Indexes
22 Sihang Qiu (TUD), Conversational Crowdsourcing
23 Hugo Manuel Proença (UL), Robust rules for prediction and description
24 Kaijie Zhu (TU/e), On Efficient Temporal Subgraph Query Processing
25 Eoin Martino Grua (VUA), The Future of E-Health is Mobile: Combining AI and Self-Adaptation to Create Adaptive E-Health Mobile Applications
26 Benno Kruit (CWI/VUA), Reading the Grid: Extending Knowledge Bases from Human-readable Tables
27 Jelte van Waterschoot (UT), Personalized and Personal Conversations: Designing Agents Who Want to Connect With You
28 Christoph Selig (UL), Understanding the Heterogeneity of Corporate Entrepreneurship Programs

2022 01 Judith van Stegeren (UT), Flavor text generation for role-playing video games
02 Paulo da Costa (TU/e), Data-driven Prognostics and Logistics Optimisation: A Deep Learning Journey
03 Ali el Hassouni (VUA), A Model A Day Keeps The Doctor Away: Reinforcement Learning For Personalized Healthcare
04 Ünal Aksu (UU), A Cross-Organizational Process Mining Framework
05 Shiwei Liu (TU/e), Sparse Neural Network Training with In-Time Over-Parameterization
06 Reza Refaei Afshar (TU/e), Machine Learning for Ad Publishers in Real Time Bidding
07 Sambit Praharaj (OU), Measuring the Unmeasurable? Towards Automatic Co-located Collaboration Analytics
08 Maikel L. van Eck (TU/e), Process Mining for Smart Product Design
09 Oana Andreea Inel (VUA), Understanding Events: A Diversity-driven Human-Machine Approach
10 Felipe Moraes Gomes (TUD), Examining the Effectiveness of Collaborative Search Engines
11 Mirjam de Haas (UT), Staying engaged in child-robot interaction, a quantitative approach to studying preschoolers' engagement with robots and tasks during second-language tutoring
12 Guanyi Chen (UU), Computational Generation of Chinese Noun Phrases
13 Xander Wilcke (VUA), Machine Learning on Multimodal Knowledge Graphs: Opportunities, Challenges, and Methods for Learning on Real-World Heterogeneous and Spatially-Oriented Knowledge
14 Michiel Overeem (UU), Evolution of Low-Code Platforms
15 Jelmer Jan Koorn (UU), Work in Process: Unearthing Meaning using Process Mining
16 Pieter Gijsbers (TU/e), Systems for AutoML Research
17 Laura van der Lubbe (VUA), Empowering vulnerable people with serious games and gamification
18 Paris Mavromoustakos Blom (TiU), Player Affect Modelling and Video Game Personalisation
19 Bilge Yigit Ozkan (UU), Cybersecurity Maturity Assessment and Standardisation
20 Fakhra Jabeen (VUA), Dark Side of the Digital Media - Computational Analysis of Negative Human Behaviors on Social Media
21 Seethu Mariyam Christopher (UM), Intelligent Toys for Physical and Cognitive Assessments

22 Alexandra Sierra Rativa (TiU), Virtual Character Design and its potential to foster Empathy, Immersion, and Collaboration Skills in Video Games and Virtual Reality Simulations
23 Ilir Kola (TUD), Enabling Social Situation Awareness in Support Agents
24 Samaneh Heidari (UU), Agents with Social Norms and Values - A framework for agent based social simulations with social norms and personal values
25 Anna L.D. Latour (UL), Optimal decision-making under constraints and uncertainty
26 Anne Dirkson (UL), Knowledge Discovery from Patient Forums: Gaining novel medical insights from patient experiences
27 Christos Athanasiadis (UM), Emotion-aware cross-modal domain adaptation in video sequences
28 Onuralp Ulusoy (UU), Privacy in Collaborative Systems
29 Jan Kolkmeier (UT), From Head Transform to Mind Transplant: Social Interactions in Mixed Reality
30 Dean De Leo (CWI), Analysis of Dynamic Graphs on Sparse Arrays
31 Konstantinos Traganos (TU/e), Tackling Complexity in Smart Manufacturing with Advanced Manufacturing Process Management
32 Cezara Pastrav (UU), Social simulation for socio-ecological systems
33 Brinn Hekkelman (CWI/TUD), Fair Mechanisms for Smart Grid Congestion Management
34 Nimat Ullah (VUA), Mind Your Behaviour: Computational Modelling of Emotion & Desire Regulation for Behaviour Change
35 Mike E.U. Ligthart (VUA), Shaping the Child-Robot Relationship: Interaction Design Patterns for a Sustainable Interaction

---

2023 01 Bojan Simoski (VUA), Untangling the Puzzle of Digital Health Interventions
02 Mariana Rachel Dias da Silva (TiU), Grounded or in flight? What our bodies can tell us about the whereabouts of our thoughts
03 Shabnam Najafian (TUD), User Modeling for Privacy-preserving Explanations in Group Recommendations
04 Gineke Wiggers (UL), The Relevance of Impact: bibliometric-enhanced legal information retrieval
05 Anton Bouter (CWI), Optimal Mixing Evolutionary Algorithms for Large-Scale Real-Valued Optimization, Including Real-World Medical Applications
06 António Pereira Barata (UL), Reliable and Fair Machine Learning for Risk Assessment
07 Tianjin Huang (TU/e), The Roles of Adversarial Examples on Trustworthiness of Deep Learning
08 Lu Yin (TU/e), Knowledge Elicitation using Psychometric Learning
09 Xu Wang (VUA), Scientific Dataset Recommendation with Semantic Techniques
10 Dennis J.N.J. Soemers (UM), Learning State-Action Features for General Game Playing
11 Fawad Taj (VUA), Towards Motivating Machines: Computational Modeling of the Mechanism of Actions for Effective Digital Health Behavior Change Applications
12 Tessel Bogaard (VUA), Using Metadata to Understand Search Behavior in Digital Libraries
13 Injy Sarhan (UU), Open Information Extraction for Knowledge Representation
14 Selma Čaušević (TUD), Energy resilience through self-organization
15 Alvaro Henrique Chaim Correia (TU/e), Insights on Learning Tractable Probabilistic Graphical Models
16 Peter Blomsma (TiU), Building Embodied Conversational Agents: Observations on human nonverbal behaviour as a resource for the development of artificial characters
17 Meike Nauta (UT), Explainable AI and Interpretable Computer Vision – From Oversight to Insight

18 Gustavo Penha (TUD), Designing and Diagnosing Models for Conversational Search and Recommendation
19 George Aalbers (TiU), Digital Traces of the Mind: Using Smartphones to Capture Signals of Well-Being in Individuals
20 Arkadiy Dushatskiy (TUD), Expensive Optimization with Model-Based Evolutionary Algorithms applied to Medical Image Segmentation using Deep Learning
21 Gerrit Jan de Bruin (UL), Network Analysis Methods for Smart Inspection in the Transport Domain
22 Alireza Shojaifar (UU), Volitional Cybersecurity
23 Theo Theunissen (UU), Documentation in Continuous Software Development
24 Agathe Balayn (TUD), Practices Towards Hazardous Failure Diagnosis in Machine Learning
25 Jurian Baas (UU), Entity Resolution on Historical Knowledge Graphs
26 Loek Tonnaer (TU/e), Linearly Symmetry-Based Disentangled Representations and their Out-of-Distribution Behaviour
27 Ghada Sokar (TU/e), Learning Continually Under Changing Data Distributions
28 Floris den Hengst (VUA), Learning to Behave: Reinforcement Learning in Human Contexts
29 Tim Draws (TUD), Understanding Viewpoint Biases in Web Search Results

---

2024 01 Daphne Miedema (TU/e), On Learning SQL: Disentangling concepts in data systems education
02 Emile van Krieken (VUA), Optimisation in Neurosymbolic Learning Systems
03 Feri Wijayanto (RUN), Automated Model Selection for Rasch and Mediation Analysis
04 Mike Huisman (UL), Understanding Deep Meta-Learning
05 Yiyong Gou (UM), Aerial Robotic Operations: Multi-environment Cooperative Inspection & Construction Crack Autonomous Repair
06 Azqa Nadeem (TUD), Understanding Adversary Behavior via XAI: Leveraging Sequence Clustering to Extract Threat Intelligence
07 Parisa Shayan (TiU), Modeling User Behavior in Learning Management Systems
08 Xin Zhou (UvA), From Empowering to Motivating: Enhancing Policy Enforcement through Process Design and Incentive Implementation
09 Giso Dal (UT), Probabilistic Inference Using Partitioned Bayesian Networks
10 Cristina-Iulia Bucur (VUA), Linkflows: Towards Genuine Semantic Publishing in Science
11 withdrawn
12 Peide Zhu (TUD), Towards Robust Automatic Question Generation For Learning
13 Enrico Liscio (TUD), Context-Specific Value Inference via Hybrid Intelligence
14 Larissa Capobianco Shimomura (TU/e), On Graph Generating Dependencies and their Applications in Data Profiling
15 Ting Liu (VUA), A Gut Feeling: Biomedical Knowledge Graphs for Interrelating the Gut Microbiome and Mental Health
16 Arthur Barbosa Câmara (TUD), Designing Search-as-Learning Systems
17 Razieh Alidoosti (VUA), Ethics-aware Software Architecture Design
18 Laurens Stoop (UU), Data Driven Understanding of Energy-Meteorological Variability and its Impact on Energy System Operations
19 Azadeh Mozafari Mehr (TU/e), Multi-perspective Conformance Checking: Identifying and Understanding Patterns of Anomalous Behavior
20 Ritsart Anne Plantenga (UL), Omgang met Regels
21 Federica Vinella (UU), Crowdsourcing User-Centered Teams

22 Zeynep Ozturk Yurt (TU/e), Beyond Routine: Extending BPM for Knowledge-Intensive Processes with Controllable Dynamic Contexts
23 Jie Luo (VUA), Lamarck's Revenge: Inheritance of Learned Traits Improves Robot Evolution
24 Nirmal Roy (TUD), Exploring the effects of interactive interfaces on user search behaviour
25 Alisa Rieger (TUD), Striving for Responsible Opinion Formation in Web Search on Debated Topics
26 Tim Gubner (CWI), Adaptively Generating Heterogeneous Execution Strategies using the VOILA Framework
27 Lincen Yang (UL), Information-theoretic Partition-based Models for Interpretable Machine Learning
28 Leon Helwerda (UL), Grip on Software: Understanding development progress of Scrum sprints and backlogs
29 David Wilson Romero Guzman (VUA), The Good, the Efficient and the Inductive Biases: Exploring Efficiency in Deep Learning Through the Use of Inductive Biases
30 Vijanti Ramautar (UU), Model-Driven Sustainability Accounting
31 Ziyu Li (TUD), On the Utility of Metadata to Optimize Machine Learning Workflows
32 Vinicius Stein Dani (UU), The Alpha and Omega of Process Mining
33 Siddharth Mehrotra (TUD), Designing for Appropriate Trust in Human-AI interaction
34 Robert Deckers (VUA), From Smallest Software Particle to System Specification - MuDForM: Multi-Domain Formalization Method
35 Sicui Zhang (TU/e), Methods of Detecting Clinical Deviations with Process Mining: a fuzzy set approach
36 Thomas Mulder (TU/e), Optimization of Recursive Queries on Graphs
37 James Graham Nevin (UvA), The Ramifications of Data Handling for Computational Models
38 Christos Koutras (TUD), Tabular Schema Matching for Modern Settings
39 Paola Lara Machado (TU/e), The Nexus between Business Models and Operating Models: From Conceptual Understanding to Actionable Guidance
40 Montijn van de Ven (TU/e), Guiding the Definition of Key Performance Indicators for Business Models
41 Georgios Siachamis (TUD), Adaptivity for Streaming Dataflow Engines
42 Emmeke Veltmeijer (VUA), Small Groups, Big Insights: Understanding the Crowd through Expressive Subgroup Analysis
43 Cedric Waterschoot (KNAW Meertens Instituut), The Constructive Conundrum: Computational Approaches to Facilitate Constructive Commenting on Online News Platforms
44 Marcel Schmitz (OU), Towards learning analytics-supported learning design
45 Sara Salimzadeh (TUD), Living in the Age of AI: Understanding Contextual Factors that Shape Human-AI Decision-Making
46 Georgios Stathis (Leiden University), Preventing Disputes: Preventive Logic, Law & Technology
47 Daniel Daza (VUA), Exploiting Subgraphs and Attributes for Representation Learning on Knowledge Graphs
48 Ioannis Petros Samiotis (TUD), Crowd-Assisted Annotation of Classical Music Compositions

---

2025 01 Max van Haastrecht (UL), Transdisciplinary Perspectives on Validity: Bridging the Gap Between Design and Implementation for Technology-Enhanced Learning Systems
02 Jurgen van den Hoogen (JADS), Time Series Analysis Using Convolutional Neural Networks
03 Andra-Denis Ionescu (TUD), Feature Discovery for Data-Centric AI

04 Rianne Schouten (TU/e), Exceptional Model Mining for Hierarchical Data
05 Nele Albers (TUD), Psychology-Informed Reinforcement Learning for Situated Virtual Coaching in Smoking Cessation
06 Daniël Vos (TUD), Decision Tree Learning: Algorithms for Robust Prediction and Policy Optimization
07 Ricky Maulana Fajri (TU/e), Towards Safer Active Learning: Dealing with Unwanted Biases, Graph-Structured Data, Adversary, and Data Imbalance
08 Stefan Bloemheuvel (TiU), Spatio-Temporal Analysis Through Graphs: Predictive Modeling and Graph Construction
09 Fadime Kaya (VUA), Decentralized Governance Design - A Model-Based Approach
10 Zhao Yang (UL), Enhancing Autonomy and Efficiency in Goal-Conditioned Reinforcement Learning
11 Shahin Sharifi Noorian (TUD), From Recognition to Understanding: Enriching Visual Models Through Multi-Modal Semantic Integration
12 Lijun Lyu (TUD), Interpretability in Neural Information Retrieval
13 Fuda van Diggelen (VUA), Robots Need Some Education: on the complexity of learning in evolutionary robotics
14 Gennaro Gala (TU/e), Probabilistic Generative Modeling with Latent Variable Hierarchies
15 Michiel van der Meer (UL), Opinion Diversity through Hybrid Intelligence
16 Monika Grewal (TU Delft), Deep Learning for Landmark Detection, Segmentation, and Multi-Objective Deformable Registration in Medical Imaging
17 Matteo De Carlo (VUA), Real Robot Reproduction: Towards Evolving Robotic Ecosystems
18 Anouk Neerincx (UU), Robots That Care: How Social Robots Can Boost Children's Mental Wellbeing
19 Fang Hou (UU), Trust in Software Ecosystems
20 Alexander Melchior (UU), Modelling for Policy is More Than Policy Modelling (The Useful Application of Agent-Based Modelling in Complex Policy Processes)
21 Mandani Ntekouli (UM), Bridging Individual and Group Perspectives in Psychopathology: Computational Modeling Approaches using Ecological Momentary Assessment Data
22 Hilde Weerts (TU/e), Decoding Algorithmic Fairness: Towards Interdisciplinary Understanding of Fairness and Discrimination in Algorithmic Decision-Making
23 Roderick van der Weerdt (VUA), IoT Measurement Knowledge Graphs: Constructing, Working and Learning with IoT Measurement Data as a Knowledge Graph
24 Zhong Li (UL), Trustworthy Anomaly Detection for Smart Manufacturing
25 Kyana van Eijndhoven (TiU), A Breakdown of Breakdowns: Multi-Level Team Coordination Dynamics under Stressful Conditions
26 Tom Pepels (UM), Monte-Carlo Tree Search is Work in Progress
27 Danil Provodin (JADS, TU/e), Sequential Decision Making Under Complex Feedback
28 Jinke He (TU Delft), Exploring Learned Abstract Models for Efficient Planning and Learning
29 Erik van Haeringen (VUA), Mixed Feelings: Simulating Emotion Contagion in Groups
30 Myrthe Reuver (VUA), A Puzzle of Perspectives: Interdisciplinary Language Technology for Responsible News Recommendation
31 Gebrekirstos Gebreselassie Gebremeskel (RUN), Spotlight on Recommender Systems: Contributions to Selected Components in the Recommendation Pipeline
32 Ryan Brate (UU), Words Matter: A Computational Toolkit for Charged Terms

33 Merle Reimann (VUA), Speaking the Same Language: Spoken Capability Communication in Human-Agent and Human-Robot Interaction
34 Eduard C. Groen (UU), Crowd-Based Requirements Engineering
35 Urja Khurana (VUA), From Concept To Impact: Toward More Robust Language Model Deployment
36 Anna Maria Wegmann (UU), Say the Same but Differently: Computational Approaches to Stylistic Variation and Paraphrasing
37 Chris Kamphuis (RUN), Exploring Relations and Graphs for Information Retrieval
38 Valentina Maccatrozzo (VUA), Break the Bubble: Semantic Patterns for Serendipity
39 Dimitrios Alivanistos (VUA), Knowledge Graphs & Transformers for Hypothesis Generation: Accelerating Scientific Discovery in the Era of Artificial Intelligence
40 Stefan Grafberger (UvA), Declarative Machine Learning Pipeline Management via Logical Query Plans
41 Mozhgan Vazifehdoostirani (TU/e), Leveraging Process Flexibility to Improve Process Outcome - From Descriptive Analytics to Actionable Insights
42 Margherita Martorana (VUA), Semantic Interpretation of Dataless Tables: a metadata-driven approach for findable, accessible, interoperable and reusable restricted access data
43 Krist Shingjergji (OU), Sense the Classroom - Using AI to Detect and Respond to Learning-Centered Affective States in Online Education
44 Robbert Reijnen (TU/e), Dynamic Algorithm Configuration for Machine Scheduling Using Deep Reinforcement Learning
45 Anjana Mohandas Sheeladevi (VUA), Occupant-Centric Energy Management: Balancing Privacy, Well-being and Sustainability in Smart Buildings
46 Ya Song (TU/e), Graph Neural Networks for Modeling Temporal and Spatial Dimensions in Industrial Decision-making
47 Tom Kouwenhoven (UL), Collaborative Meaning-Making. The Emergence of Novel Languages in Humans, Machines, and Human-Machine Interactions
48 Evy van Weelden (TiU), Integrating Virtual Reality and Neurophysiology in Flight Training
49 Selene Báez Santamaría (VUA), Knowledge-centered conversational agents with a drive to learn
50 Lea Krause (VUA), Contextualising Conversational AI
51 Jiaxu Zhao (TU/e), Understanding and Mitigating Unwanted Biases in Generative Language Models
52 Qiao Xiao (TU/e), Model, Data and Communication Sparsity for Efficient Training of Neural Networks
53 Gaole He (TUD), Towards Effective Human-AI Collaboration: Promoting Appropriate Reliance on AI Systems
54 Go Sugimoto (VUA), MISSING LINKS Investigating the Quality of Linked Data and its Tools in Cultural Heritage and Digital Humanities
55 Sietze Kai Kuilman (TUD), AI that Glitters is Not Gold: Requirements for Meaningful Control of AI Systems
56 Wijnand van Woerkom (UU), A Fortiori Case-Based Reasoning: Formal Studies with Applications in Artificial Intelligence and Law
57 Syeda Amna Sohail (UT), Privacy-Utility Trade-Off in Healthcare Metadata Sharing and Beyond: A Normative and Empirical Evaluation at Inter and Intra Organizational Levels
58 Junhan Wen (TUD), "From iMage to Market": Machine-Learning-Empowered Fruit Supply

59 Mohsen Abbaspour Onari (TU/e), From Explanation to Trust: Modeling and Measuring Trust in Explainable Decision Support
60 Marcel Jurriaan Robeer (UU), Beyond Trust: A Causal Approach to Explainable AI in Law Enforcement
61 Shuai Wang (VUA), Links in Large Integrated Knowledge Graphs: Analysis, Refinement, and Domain Applications
62 Khaleel Asyraaf Mat Sanusi (OU), Augmenting a learning model within immersive learning environments for psychomotor skills
63 Rashid Zaman (TU/e), Online Conformance Checking on Degraded Data
64 Jens d'Hondt (TU/e), Effective and Efficient Multivariate Similarity Search
65 Aswin Balasubramaniam (UT), Disentangling Runner Drone Interaction Potentialities

---

2026 01 Pei-Yu Chen (TUD), Human-Agent Alignment Dialogues: Eliciting User Information at Runtime for Personalized Behavior Support
02 Hezha Hassan Mohammedkhan (TiU), Estimating Body Measurements of Children from 2D Images: Towards the Automatic Detection of Malnutrition
03 Kyriakos Psarakis (TUD), Democratizing Scalable Cloud Applications: Transactional Stateful Functions on Streaming Dataflows
04 Boyu Xu (UU), Exploring Indirect Relations Between Topics in Neuroscience Literature Using Augmented Reality to Inform Experimental Design
05 Koen Minartz (TU/e), Stochastic Simulation with Geometric Deep Generative Models
06 Azim Afroozeh (CWI, VUA), FastLanes: A Next-Gen File Format
07 Inès Blin (VUA), Narrative Understanding with Knowledge Graphs
08 Paul van Vulpen (UU), Debating Digital Dominance: Decentralized Technology Governance For Strategic Autonomy
09 Afrizal Doewes (TU/e), Rethinking Automated Essay Scoring: Agreement, Fairness, and Feedback
10 Nikolaos Delapaschos Kondylidis (VUA), Establishing Task-Oriented Understanding between Agents
11 Işıl Baysal Erez (UT), Handling Missing Data with Meta-Learning and Large Language Models
12 Xue Li (UvA), From Fine-tuning to Prompting: A Paradigm Shift in Knowledge Graph Construction
13 Isaac da Silva Torres (VUA), Guidelines To Flux Between Conceptual Models: Understanding Complex Digital Business Ecosystems
14 Philip Lippmann (TUD), Synthetic Data for Robust Language Modelling
15 Rashmi Khazanchi (OU), Artificial Intelligence in Education: Impact of AI-Based Systems on Mathematics Achievement
16 Carolina Ferreira Gomes Centeio Jorge (TUD), Modelling Artificial Trust for Effective Human-AI Teamwork
17 Maria Tsfasman (TUD), Towards Predicting Memory in Multimodal Group Interactions
18 Riccardo Lo Bianco (TU/e), Deep Reinforcement Learning for Automated Decision-Making in Process Management Systems
19 Israel Campero Jurado (TU/e), Innovations in Optimization and Applications in Healthcare
20 Iftitahu Ni'mah (TU/e), Contrastive Learning and Evaluation in Low Resource Scenario of Natural Language Processing
21 Francisco N.F.Q. Simoes (UU), Causality, Information, and Decision-Making
22 Ruben Verhagen (TUD), Transparent and Explainable Agents for Human-Agent Teaming

23 Eduardo Calò (UU), Automatically Expressing the Meaning of Logical Formulae in Natural Language

24 Shuai Han (UU), Improving Sample Efficiency of Reinforcement Learning: Exploiting Structural Knowledge for Decision Making

25 Francis Saa-Dittoh (VUA), From Radio to AI: African Community-Driven Development of Sustainable Information Systems

26 Mohammed Al Owayyed (TUD), Interactive Simulation-Based Learning Tools for Training Children's Helpline Counsellors

27 **Bram Grooten** (TU/e), Adaptive Reinforcement Learning: Lean and Dynamic Agents for Robust Generalization

§

List of Publications

Conference proceedings:

1. **Bram Grooten**, Patrick MacAlpine, Kaushik Subramanian, Peter Stone, and Peter R. Wurman. Out-of-Distribution Generalization with a SPARC: Racing 100 Unseen Vehicles with a Single Policy. *Oral at the AAAI Conference on Artificial Intelligence*, 2026.

2. Calarina Muslimani, **Bram Grooten**, Deepak R.S. Mamillapalli, Mykola Pechenizkiy, Decebal C. Mocanu, Matthew E. Taylor. Boosting Robustness in Preference-Based Reinforcement Learning with Dynamic Sparsity. *Autonomous Agents and Multiagent Systems (AAMAS)*, 2025.

3. **Bram Grooten**, Tristan Tomilin, Gautham Vasan, Matthew E. Taylor, A. Rupam Mahmood, Meng Fang, Mykola Pechenizkiy, and Decebal C. Mocanu. MaDi: Learning to Mask Distractions for Generalization in Visual Deep Reinforcement Learning. *Autonomous Agents and Multiagent Systems (AAMAS)*, 2024.

4. **Bram Grooten**. Large Learning Agents: Towards Continually Aligned Robots with Scale in RL. *Doctoral Consortium at Autonomous Agents and Multiagent Systems (AAMAS)*, 2024.

5. Aleksandra I. Nowak, **Bram Grooten**, Decebal C. Mocanu, and Jacek Tabor. Fantastic Weights and How to Find Them: Where to Prune in Dynamic Sparse Training. *Neural Information Processing Systems (NeurIPS)*, 2023.

6. **Bram Grooten**, Ghada Sokar, Shibhansh Dohare, Elena Mocanu, Matthew E. Taylor, Mykola Pechenizkiy, and Decebal C. Mocanu. Automatic Noise Filtering with Dynamic Sparse Training in Deep Reinforcement Learning. *Autonomous Agents and Multiagent Systems (AAMAS)* and *Spotlight at the Sparse Neural Networks workshop of the International Conference on Learning Representations (ICLR)*, 2023.

Workshop papers:

7. Camiel Oerlemans*, **Bram Grooten***, Michiel Braat, Alaa Alassi, Emilia Silvas, Decebal C. Mocanu. LiMTR: Time Series Motion Prediction for Diverse Road Users through Multimodal Feature Integration. *Time Series in the Age of Large Models workshop at Neural Information Processing Systems (NeurIPS)*, 2024.

8. Wieger Wesselink, **Bram Grooten**, Qiao Xiao, Cassio de Campos, and Mykola Pechenizkiy. Nerva: a Truly Sparse Implementation of Neural Networks. *Sparse Neural Networks workshop at the International Conference on Learning Representations (ICLR)*, 2023.

9. **Bram Grooten**, Jelle Wemmenhove, Maurice Poot, Jim Portegies. Is Vanilla Policy Gradient Overlooked? Analyzing Deep Reinforcement Learning for Hanabi. *Adaptive and Learning Agents workshop at Autonomous Agents and Multiagent Systems (AAMAS)*, 2022.

Currently under review:

10. **Bram Grooten***, Farid Hasanov*, Chenxiang Zhang, Qiao Xiao, Boqian Wu, Zahra Atashgahi, Ghada Sokar, Shiwei Liu, Lu Yin, Elena Mocanu, Mykola Pechenizkiy, and Decebal C. Mocanu. Training with Dynamic Sparse Heads as the Key to Effective Ensembling. *Under review*, 2026.

11. Tristan Tomilin, Luka van den Boogaard, Samuel Garcin, **Bram Grooten**, Meng Fang, Yali Du, Mykola Pechenizkiy. MEAL: A Benchmark for Continual Multi-Agent Reinforcement Learning. *Under review*, 2026.

12. Wieger Wesselink, **Bram Grooten**, Huub van de Wetering, Qiao Xiao, Decebal C. Mocanu. Batch Matrix-form Equations and Implementation of Multilayer Perceptrons. *Under review*, 2026.

List of Invited Talks

The author has been invited to speak at the following venues:

1. **Openmind Winter School**
   Lean & Adaptive Agents for Generalization in Deep RL.
   Universiti Tunku Abdul Rahman. Selangor, Malaysia. Jan 27th, 2026.

2. **EAISI**
   How to Make Sure your Robot is Not Distracted.
   Eindhoven AI Systems Institute, Netherlands. Dec 17th, 2025.

3. **VU Amsterdam**
   Lean & Adaptive Agents for Generalization in Deep RL.
   Robotics lab. Amsterdam, Netherlands. Dec 9th, 2025.

4. **TU Darmstadt**
   Lean & Adaptive Agents for Generalization in Deep RL.
   Jan Peters' IAS lab. Darmstadt, Germany. Dec 5th, 2025.

5. **TU Delft**
   Lean & Adaptive Agents for Generalization in Deep RL.
   Delft, Netherlands. Dec 4th, 2025.

6. **Massachusetts Institute of Technology**
   Efficient Focus for Autonomous Agents.
   LIDS: Cathy Wu's lab. Cambridge, MA, United States. Aug 19th, 2024.

7. **Sony AI**
   Efficient Focus for Autonomous Agents: Generalization in Deep RL.
   Tech Talk Series. Zürich, Switzerland. Jul 24th, 2024.

8. **ETH Zürich**
   Efficient Focus for Autonomous Agents: Generalization in Deep RL.
   Computational Robotics Lab. Zürich, Switzerland. Jul 17th, 2024.

9. **ML Collective**
   Efficient Focus for Autonomous Agents: Generalization in Deep RL.
   Online reading group by Rosanne Liu: "DL: Classics and Trends." Mar 15th, 2024.

10. **Leiden University**
    Efficient Focus for Autonomous Agents.
    Leiden, Netherlands. Feb 13th, 2024.

11. **University of Calgary**
    Efficient Focus for Autonomous Agents.
    Calgary AB, Canada. Oct 25th, 2023.

12. **LIFE at MIT**
    MaDi: Learning to Mask Distractions from Pixels.
    Online reading group "Learning in Foundation Environments." Oct 23rd, 2023.

13. **University of Alberta**
    Efficient Focus for Autonomous Agents.
    Edmonton AB, Canada. Aug 25th, 2023.

14. **PyData**
    Automatic Noise Filtering.
    Eindhoven, Netherlands. Apr 26th, 2023.

15. **Jagiellonian University**
    Efficient AI for Autonomous Agents.
    Kraków, Poland. Jul 5th, 2022.

Acknowledgements

Throughout my PhD journey, I have received support from numerous people. I'd like to thank all of you for helping me complete this grand project.

First of all, thank you Decebal, for all our enjoyable weekly meetings and conference trips together. You have a great sense of when to motivate or calm your students, and I was lucky to have you as my supervisor. Thank you Mykola, for providing additional guidance where necessary. I appreciate your dry humor and look back on our safari in Rwanda with pleasure.

Thanks to the doctoral committee for evaluating my thesis. Long, it was lovely to meet you at AAMAS, always smiling and happy to discuss RL. Matthijs, thanks for inviting me to Delft, it is always great to chat with you and your group. Matt, I immensely enjoyed my research visit to your lab in 2023. It was one of the best parts of my PhD. I'm glad we found a weekly timeslot to meet, which wasn't too hard once we realized we're both morning people. Ghada, thank you so much for your support, not only in research, but also in other parts of life. Joaquin, I appreciate your kindness and broad expertise. Looking forward to working more with you as a postdoc!

I want to thank my office mates in MF 7.086 for providing a pleasant balance between work and chat, making research extra enjoyable. We jokingly called our office "The Foster Home", as at one point all 6 of us had our main supervisor affiliated at a different university. Thanks Shiwei, Ghada, Tristan, Jiaxu, Qiao, Yudi, Danil, Mickey, Yannis.

Tristan and Farid, thank you for being the paranymphs at my defense! You are probably the two TU/e colleagues with whom I've collaborated most intensely. Thanks for making our research together so pleasant, and for the fun outside of work as well. I cherish our basketball sessions, tennis matches, birthday parties, and PSV games.

Thank you to all my dear colleagues of the BlueNN lab and the researchers who joined along: Farid, Christopher, Inês, Chen, Qiao, Boqian, Aleksandra, Lu, Shiwei, Zahra, Ghada, Elena. I greatly value the many interesting group discussions we've had.

I wish to send gratitude across the ocean to the kind people at the University of Alberta, whom I had the fortune of meeting. Gautham and Callie especially, thank you for our fascinating collaborations that made my research visit deeply worthwhile. Abhishek, Shibhansh, Fernando, Adrian, and others, thanks for many interesting discussions, playing ice hockey and cricket, and for lending me your mountainbike.

Thanks also to the many people I've interacted with at Sony AI. Florian, for helping me get started; Patrick, for supervising me closely and getting up early for our weekly online meeting; Kaushik, for making me feel part of the team. Also Yunshu, Peter, Pete, thank you for this opportunity!

I greatly appreciate the wonderful people in our Data and AI cluster, which has grown too large to name everyone. However, I'd like to give a shoutout to our team that won 2nd place at the TU/e Sports Day 2025! Fahad, Jiaxu, Shruti, Tristan, Daan, Wieger Punter, Ziwei, Hilde, Wouter van Lil, Lieke, Gizem; you made me a proud captain. Wouter Duivesteijn, thanks for giving me the first and most important piece of feedback I've received on poster design: "Way too much text!" I am now also a fierce proponent of minimizing the amount of text on a scientific poster. Wieger Wesselink, thank you for diving into truly sparse implementations with me. I appreciate your thorough way of thinking. Gizem and Julie, thank you for helping so much in organizing our regular DAI Seminar. Thanks also to other secretaries, including Ine and José, who were always open for a chat and welcomed me warmly into the group. A big thank you to Emine and all others who helped keep our workplace clean. I greatly appreciate your unlimited friendliness in the hallway.

Thanks to all my friends for providing welcome distractions in life, be it baseball, other outdoor activities, or Serpentine events. Thanks for coming to my annual birthday boardgame tournament (which I finally won this year, after organizing it 9 times).

Thank you to all my family, close and far. From Roermond to Nij Beets and beyond. *Bedankt mama*, who's glancing from above, for teaching me "*liefde is het belangrijkste wat er is.*" *Bedankt papa*, for showing me what ambition and determination look like. *Bedankt Rik, voor het zijn van mijn beste maatje.* Good luck in your own PhD!

En lieve Anne Lynn, my soon-to-be wife, thank you for all your love and warmth throughout the years. You are the kindest, most patient, and coolest person I know. Thank you for the greatest gift of all, *onze blije gup David.*

About the Author

Bram Grooten is a PhD researcher at Eindhoven University of Technology (TU/e). In 2017, he obtained his bachelor's degree in applied mathematics, having studied at Wentworth Institute of Technology and TU/e. He completed a double master's in applied mathematics and science education, acquiring the Dutch high school teacher's license and graduating cum laude in 2021.

That year, Bram began his current position, researching reliable, adaptive, and intelligent agents. His work focuses on reinforcement learning, generalization, robotics, and sparse neural networks and has been published in various top-level scientific conference proceedings (e.g., AAMAS, NeurIPS, AAAI). Throughout his PhD, he has gained research experience through internships at Sony AI (Zürich), joining the *Gran Turismo* team developing RL for PlayStation games, and the University of Alberta (Edmonton), being part of the *Intelligent Robot Learning* lab, working with the UR5 robotic arm. Both visits have contributed to chapters in this thesis.

He has given invited talks at leading institutions worldwide, including MIT, ETH Zürich, University of Calgary, Jagiellonian University, and was a finalist in the *Talking Science 2025* pitch competition. He was accepted at top deep learning summer schools, including DLRL, MLSS, EEML, M2L, and has served as a reviewer for scientific conferences and journals such as NeurIPS, ICML, AAAI, Nature Communications. Bram is a co-applicant on a successful grant for 80,000 GPU hours on MeluXina, Luxembourg's largest HPC, and has obtained multiple personal compute grants from NWO. He has co-supervised numerous bachelor's students—including an Honors Academy group on robotics with the SO-ARM101—and master's students, some of whom graduated cum laude or published their work at a NeurIPS workshop.

In an ever-changing environment, intelligent agents need to be adaptive to maintain their performance. Can a robotic arm still reach to a target when its vision is distracted by new, unseen backgrounds? Is a reinforcement learning policy, trained on a diverse set of cars, able to safely drive another vehicle outside of the initial distribution?

Generalization problems such as these are investigated in this thesis. The corresponding algorithmic solutions aim to be lean and dynamic. Neural networks in this research are efficient in their parameter usage and promote a high level of adaptiveness: both internally in the network's structure, as well as externally in the agent's resulting behaviour. The outcome provides a step towards reliable, adaptive, and intelligent robots.

TU/e
EINDHOVEN
UNIVERSITY OF
TECHNOLOGY