

Automatic Noise Filtering with Dynamic Sparse Training in Deep Reinforcement Learning

Bram Grooten
Eindhoven University of Technology
b.j.grooten@tue.nl

Ghada Sokar
Eindhoven University of Technology
g.a.z.n.sokar@tue.nl

Shibhansh Dohare
University of Alberta
dohare@ualberta.ca

Elena Mocanu
University of Twente
e.mocanu@utwente.nl

Matthew E. Taylor
University of Alberta & Alberta
Machine Intelligence Institute (Amii)
matthew.e.taylor@ualberta.ca

Mykola Pechenizkiy
Eindhoven University of Technology
m.pechenizkiy@tue.nl

Decebal Constantin Mocanu
University of Luxembourg &
University of Twente
d.c.mocanu@utwente.nl

ABSTRACT

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 in synergy with various deep reinforcement learning algorithms. The sparse input layer learns to focus its connectivity on task-relevant features, such that ANF-SAC and ANF-TD3 outperform standard SAC and TD3 by a large margin, while using up to 95% fewer weights. Furthermore, we devise a transfer 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. Our code is available online.¹

KEYWORDS

deep reinforcement learning; noise filtering; sparse training

ACM Reference Format:

Bram Grooten, Ghada Sokar, Shibhansh Dohare, Elena Mocanu, Matthew E. Taylor, Mykola Pechenizkiy, and Decebal Constantin Mocanu. 2023. Automatic Noise Filtering with Dynamic Sparse Training in Deep Reinforcement Learning. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 25 pages.

¹See <https://github.com/bramgrooten/automatic-noise-filtering>

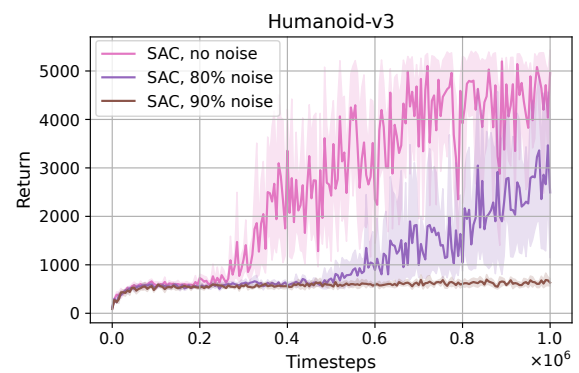


Figure 1: Performance of SAC on Humanoid-v3 environments expanded with a different number of pure noise features. Once the environment contains too much noise, SAC struggles to learn a decent policy. Standard dense networks cannot filter through the noise well enough on this problem.

1 INTRODUCTION

Future robots will likely perceive a plethora of information about the state of the world, but only parts of it are going to be relevant to their current task. For instance, a household robot receiving abundant information about all objects and processes in the house.² For its current task, e.g. making pancakes, only a small subset of these information sources, or *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) [21] fails to learn a decent policy on an environment with 90% added noise features, see Figure 1. We simulate the noisy real-world environment by adding synthetic noise features to an existing state

²For example: cleanliness of floors, furniture, cupboards, kitchen utensils; CO₂, CO levels and temperature in each room; up-to-date stock of all food and non-food items in the fridge and/or basement; mood, nourishment, and health of all inhabitants; etc.

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 [33], 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 [3, 38]. Further, [46] discovered that sparse networks can find minimal task representations in deep RL by pruning redundant input dimensions. Not long after, [39] 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 [4, 10, 35].

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 [21] and TD3 [19], which have fully dense layers throughout their networks. We devise the *extremely noisy environment* (ENE), further defined in Section 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 [9, 44].

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.

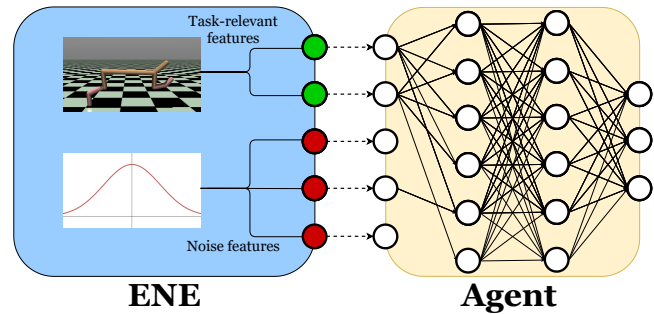


Figure 2: Extremely noisy environments (ENEs) contain many noise ● features and some relevant ● features. We use ENEs where up to 99% of the features are noise. Our method Automatic Noise Filtering (ANF) learns to predominantly connect with the input neurons that provide useful information and outperforms dense baselines by a large margin, especially in the noisiest environments.

- We devise a transfer learning setting of extremely noisy environments and 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, increasing the difficulty. ANF maintains its advantage on these challenging extensions.³

Outline. In Section 2 we formulate the problem setting. Section 3 gives an overview of the background and related work. Our method is introduced in Section 4, along with the first experiments. In Section 5 we explore the transfer learning setting. Sections 6 through 9 provide further analysis, where we perform an ablation study and discover how far we can extend our problem and algorithm. Finally, Section 10 concludes the paper. Additional results, details, and discussion are in the Appendix.⁴

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.

In our main experiments, the noise features produce pure Gaussian noise, sampled 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

³See an illustrative video here: <https://youtu.be/vS47UnsTQk8>

⁴The Appendix can be found here: <https://arxiv.org/abs/2302.06548>

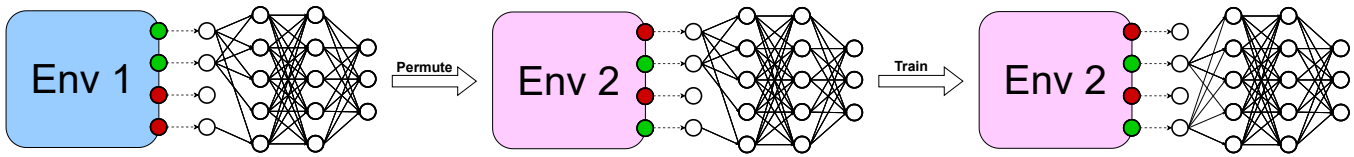


Figure 3: In *permuted extremely noisy environments (PENE)* the order of the input features gets shuffled after a certain number of timesteps. Our ANF agents automatically adjust their network structure to this new environment. They show superior forward transfer compared to fully dense methods, even 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.

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.

Transfer 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.⁵ 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 *permuted extremely noisy environment (PENE)*, see Figure 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.

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.

Sparse training. Dynamic sparse training is a subfield of the sparse training regime [32], 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 [18, 22, 28]. In sparse-to-sparse training, where DST belongs, a network begins with a high sparsity level from scratch [5, 33]. 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 [11, 17, 31]. These algorithms benefit from potential performance boosts, decreased computational costs, and better generalization [12, 30]. Furthermore, DST has been used successfully for an efficient feature selection algorithm [3], which inspired our project.

DST in RL. Applying sparse training in reinforcement learning is useful, as real-world applications often deal with latency constraints [15], which limits the number of parameters. Unfortunately,

⁵Features 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.

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 [20]. Only a few papers have applied sparse training to deep RL so far. In the *offline* RL setting, [2] 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 [37]. Therefore, we focus on the *online* RL setting throughout the paper, and even go into the transfer learning setting [43].

To the best of our knowledge, the first work applying DST to online RL is from [39]. They outperform dense networks with the algorithms DS-TD3 and DS-SAC, which combine sparse evolutionary training (SET) [33] with TD3 [19] and SAC [21]. The methods of [39] form the foundation of our ANF algorithm.

Sokar et al. [39] reached a global sparsity level of 50%, which was later improved upon by [20, 42], 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. [20] compared DST methods such as SET [33] and RigL [17] in many deep RL environments. Their performance proved to be quite similar, so we choose to use only SET.

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 [6, 34, 41, 45]. 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 [47]. It introduced an evolutionary algorithm to select relevant features. Most of the follow-up work takes this evolutionary approach [1, 7,

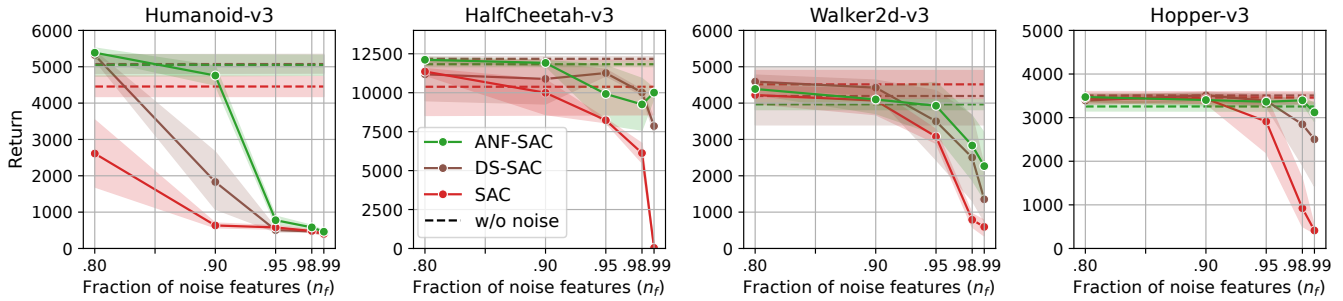


Figure 4: Performance of Automatic Noise Filtering (ANF) compared to its baselines for different noise fractions n_f . The curves show average return in the last 10% of training over 5 seeds, with shaded regions representing 95% confidence intervals. When the number of noise features in an environment 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 15 of Appendix D.1.

26], 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 [40]. Other methods that have some similarities to our approach include recognizing distractor signals [36], reducing state dimensionality [8, 14], and identifying fake features in federated learning [27].

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 [39], which use sparse evolutionary training (SET) from [33] 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.

One aspect that sets ANF apart from the previous works on non-noisy settings [20, 39] 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 9 we investigate models that also have sparse hidden layers.

Another difference between ANF and DS-TD3/SAC [39] is that we mask the running averages of first and second raw moments of the gradient within the Adam optimizer [25] 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 papers that apply Adam and only simulate true sparsity with binary masks on top of the weight matrices. Our research also utilizes such binary masks while keeping the truly sparse implementation in mind. See Appendix C for further discussion.

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 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 [19, 21, 39]. See Appendix B 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 $5\times$, $10\times$, $20\times$, $50\times$, and $100\times$, 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 [20], for overview graphs such as Figure 4. Throughout the paper, we run 5 random seeds for every setting. In the graphs, we show the average curve as well as a 95% confidence interval.

Table 1: State and action space dimensions.

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

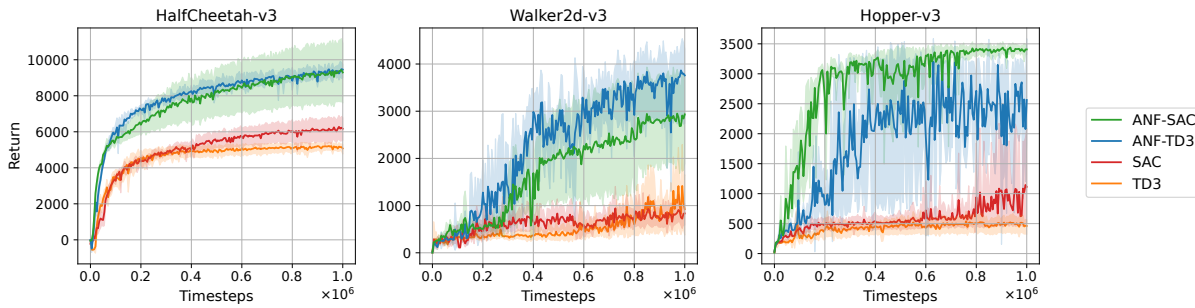


Figure 5: Learning curves on 3 environments with 98% noise features. This level of noise is too high for standard SAC and TD3 to learn a decent policy. ANF can still filter through the distraction and find a well-performing policy.

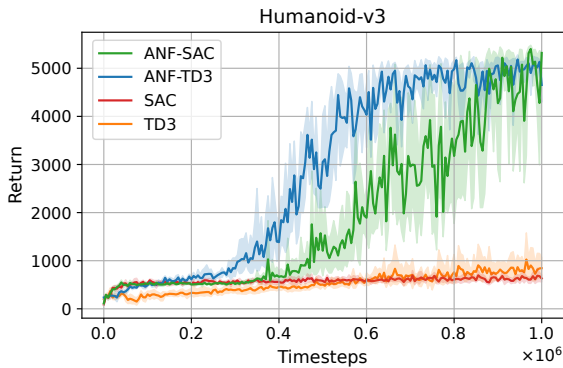


Figure 6: Learning curves for Humanoid-v3 with 90% noise features. While standard SAC and TD3 are too distracted by the noise to learn the task, ANF finds the task-relevant features and is able to improve.

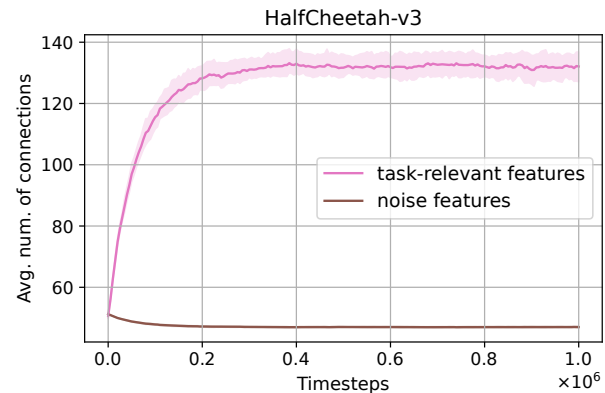


Figure 7: Average number of connections in the input layer of one of ANF-TD3’s critic networks, on HalfCheetah-v3 with 90% noise features. At the start of training every input neuron has around $256 \cdot 0.2 \approx 51$ connections, because the input layer sparsity is 80% and connections are allocated uniformly at random. During training, ANF gradually prunes connections from the noise features and grows connections to the relevant features. A similar graph for ANF-SAC is shown in Figure 16 of Appendix D.1.

Results. First of all, the horizontal lines in Figure 4 show that even in environments 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* [46].

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 1 shows that Humanoid-v3 differs noticeably from the other three environments by the size of its state space.

The learning curves in Figure 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*.⁶ In the environments HalfCheetah, Walker2d, and Hopper, we continue to observe this behavior up to a noise fraction of 98%, as shown in Figure 5. ANF outperforms its baselines by a large margin in each environment.

⁶Which is a return of ~ 4500 , see SAC’s dashed line in Figure 4, Humanoid-v3.

Topology shift. To analyze what is actually happening, we visualize the connectivity of ANF. In Figure 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 features rises, while on the other hand, noise features receive fewer weights. Together with the increased performance shown in Figure 4, this fully supports our Adaptivity Hypothesis.

5 TRANSFER 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 *permuted extremely noisy environment* (PENE), as described in Section 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

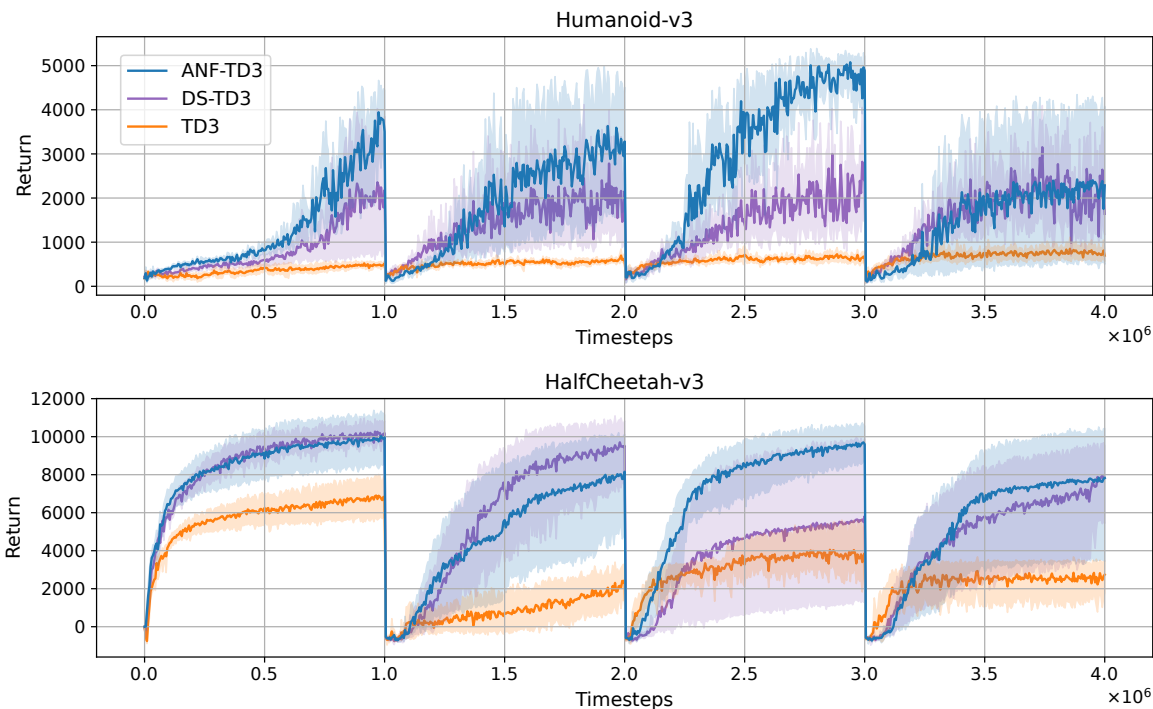


Figure 8: Performance of ANF-TD3 and its baselines on permuted extremely noisy environments (PENE) with 95% noise features. After every 1M timesteps, the environment’s features are shuffled with a random permutation. ANF is able to cope with this challenge, while the fully dense networks of TD3 are struggling. DS-TD3 performs decently, but ANF has an advantage by focusing its sparsity on the input layer. Similar graphs for ANF-SAC and other environments are shown in Appendix D.2.

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. Similar to the experiments of Section 4, we compare ANF-SAC and ANF-TD3 with their fully dense baselines and DS-SAC/TD3. We show 95% confidence intervals over 5 seeds.

Results. Figure 8 shows the results for ANF-TD3 on Humanoid and HalfCheetah. See Appendix D.2 for the graphs of the remaining algorithms and environments. 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 [16]. 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 7 for the ENE experiments.

Now, in Figure 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.

6 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.

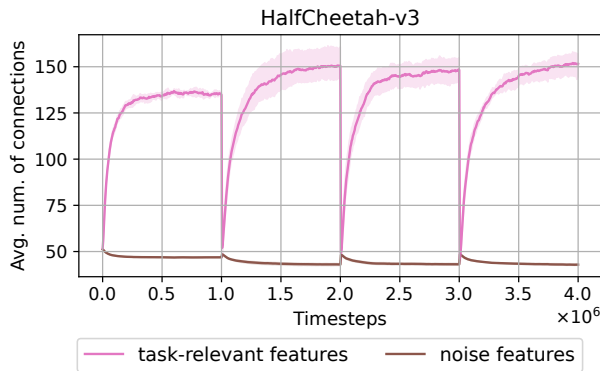


Figure 9: Average number of connections in the input layer of an ANF-SAC critic network, on HalfCheetah-v3 with $n_f = 0.95$. Every 1M timesteps, the PENE permutes the order of the features. The ANF agent adjusts its network structure quickly, growing connections to the task-relevant features, which are now sent to different input neurons. A similar graph for ANF-TD3 is shown in Figure 21 of Appendix D.2.

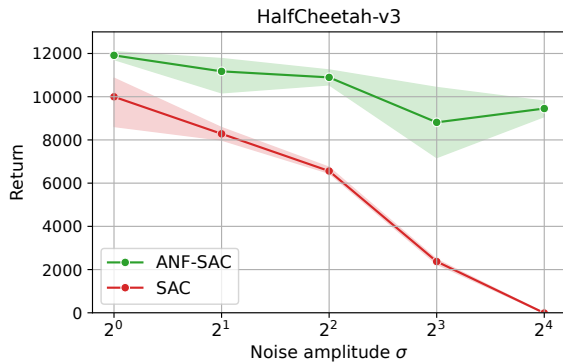


Figure 10: ANF-SAC and its dense baseline on ENEs with louder noise. The noise features are sampled from $\mathcal{N}(0, \sigma^2)$. Noise amplitude σ is increased exponentially, notice the log-scale on the horizontal axis. ANF tolerates louder noise much better than standard SAC, maintaining a high performance up to $\sigma = 16$. In this experiment $n_f = 0.9$. The graph for ANF-TD3 is given in Figure 23 of Appendix D.3.

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 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.⁷

⁷See <https://youtu.be/vS47UnsTQk8> for a video comparing the actual motion of HalfCheetah when controlled by ANF-SAC vs. SAC with different noise amplitudes.

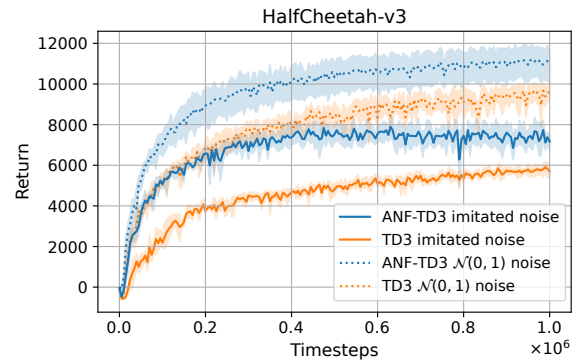


Figure 11: Learning curves for ANF-TD3 and its dense baseline on the challenging ENE, where the noise features imitate the task-relevant features. This increases the difficulty, but ANF still achieves the highest return. The ENE has 90% of these realistic noise features in this experiment. See Figure 24 in Appendix D.4 for the graph of ANF-SAC.

7 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. In Appendix E, we present visualizations of the distributions of task-relevant features before and after training.⁸ To get closer to real-world noise, we want the noise features of our ENEs to imitate 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 environment), as shown in Appendix E. In the experiments of this section, the ENE samples from these histograms⁹ to generate noise features for the next state. We repeatedly sample from the distribution of each original feature 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 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.

8 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, which we call *Static-ANF*. In addition, we compare the standard dense algorithms of SAC and TD3. We run on Humanoid-v3 with 90% noise features.

⁸The challenging distribution shift of RL is clearly visible!

⁹By first sampling a bin according to the histogram’s probability mass function, and then sampling a value uniformly at random within the chosen bin.

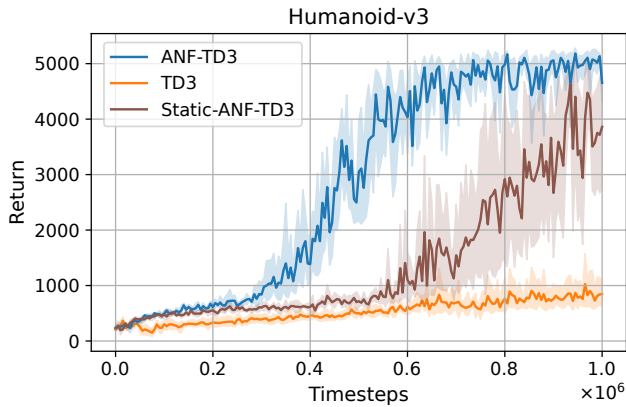


Figure 12: Comparison of ANF to its static sparse counterpart and standard (fully dense) TD3. ANF-TD3 dynamically updates the sparse network topology, while the topology of the other two methods remains static. The learning curves show that the dynamic updates are an essential part of ANF. This experiment is run with 90% noise features. A similar graph for SAC is shown in Figure 25 of Appendix D.5.

Results. The graphs in Figure 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 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.

9 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 [39]. 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 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 13 (left), we see that ANF-TD3 with a global sparsity of 80% still surpasses standard dense TD3 in

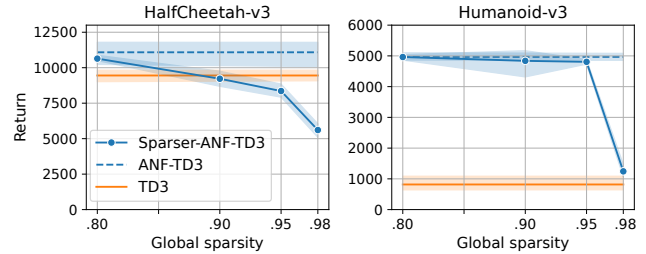


Figure 13: Performance of ANF-TD3 and its sparser versions, on 90% noise features. Sparser-ANF also prunes weights in the hidden layer, instead of only the input layer. Further sparsifying the network does not improve performance, but can drastically reduce the size of the network. The graphs for ANF-SAC are shown in Figure 26 of Appendix D.6.

Table 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 (\uparrow)	# Params. (\downarrow)
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

final return on HalfCheetah-v3. As the global sparsity increases, the performance gradually decreases. This is quite different for Humanoid-v3, in Figure 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.

10 CONCLUSION & LIMITATIONS

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 adjusts its network topology to focus on task-relevant features.

Our experiments provide an initial empirical verification of our Adaptability Hypothesis, which roughly states that dropping and growing sparse connections is easier than adjusting dense weights. Further research is necessary to grant more conclusive evidence, as our work is limited to continuous control tasks that have feature vectors as states. We exclusively studied SAC and TD3; integrating ANF in other deep RL methods is open for future work.

The input layer sparsity is an important hyperparameter for ANF. Making this an adaptive parameter could be beneficial. Further, we believe expanding ANF’s compatibility towards other neural network types is a promising future research direction. Combining ANF with convolutional NNs or transformers would open the possibility of operating on noisy image or video data.

ACKNOWLEDGMENTS

This publication is part of the project AMADeUS (with project number 18489) of the Open Technology Programme, which is partly financed by the Dutch Research Council (NWO). This research used the Dutch national e-infrastructure with the support of the SURF Cooperative, using grant no. EINF-3098. Part of this work has taken place in the Intelligent Robot Learning (IRL) Lab at the University of Alberta, which is supported in part by research grants from the Alberta Machine Intelligence Institute (Amii); a Canada CIFAR AI Chair, Amii; Compute Canada; Huawei; Mitacs; and NSERC. We thank Joan Falcó Roget, Mickey Beurskens, Anne van den Berg, and Rik Grooten for the fruitful discussions. Finally, we thank the anonymous reviewers and Antonie Bodley for their thorough proof-reading and useful comments.

REFERENCES

- [1] Kevin Acunto. 2012. *Feature selection for scalable reinforcement learning*. Ph.D. Dissertation. State University of New York at Binghamton. URL: <https://www.proquest.com/openview/73f1a695503d3e32f0fa49cae57cb411>. (Cited in Section 3)
- [2] Samin Yeasar Arnob, Riyasat Ohib, Sergey Plis, and Doina Precup. 2021. Single-Shot Pruning for Offline Reinforcement Learning. *arXiv preprint arXiv:2112.15579* (2021). URL: <https://arxiv.org/abs/2112.15579>. (Cited in Section 3)
- [3] Zahra Atashgahi, Ghada Sokar, Tim van der Lee, Elena Mocanu, Decebal Constantin Mocanu, Raymond Veldhuis, and Mykola Pechenizkiy. 2022. Quick and Robust Feature Selection: the Strength of Energy-efficient Sparse Training for Autoencoders. *Machine Learning* 111, 1 (2022), 377–414. URL: <https://link.springer.com/article/10.1007/s10994-021-06063-x>. (Cited in Section 1, 3)
- [4] Paul Bach-y Rita, Carter C Collins, Frank A Saunders, Benjamin White, and Lawrence Scadden. 1969. Vision Substitution by Tactile Image Projection. *Nature* 221, 5184 (1969), 963–964. URL: <https://www.nature.com/articles/221963a0>. (Cited in Section 1)
- [5] Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. 2018. Deep Rewiring: Training very sparse deep networks. *International Conference on Learning Representations* (2018). URL: <https://arxiv.org/abs/1711.05136>. (Cited in Section 3)
- [6] Wouter van den Bemd. 2022. *Robust Deep Reinforcement Learning for Greenhouse Control and Crop Yield Optimization*. Master's thesis. Eindhoven University of Technology. URL: <https://research.tue.nl/en/studentTheses/robust-deep-reinforcement-learning-for-greenhouse-control-and-cro>. (Cited in Section 3)
- [7] Julian Bishop and Risto Miikkulainen. 2013. Evolutionary Feature Evaluation for Online Reinforcement Learning. In *2013 IEEE Conference on Computational Intelligence in Games (CI-G)*. 1–8. <https://doi.org/10.1109/CI-G.2013.6633648> (Cited in Section 3)
- [8] Nicolò Botteghi, Khaled Alaa, Mannes Poel, Beril Sirmacek, Christoph Brune, Abeje Mersha, and Stefano Stramigioli. 2021. Low Dimensional State Representation Learning with Robotics Priors in Continuous Action Spaces. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 190–197. URL: <https://arxiv.org/abs/2107.01667>. (Cited in Section 3)
- [9] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. *arXiv preprint arXiv:1606.01540* (2016). URL: <https://www.gymlibrary.dev/>. (Cited in Section 1, 14)
- [10] Elisa Castaldi, Claudia Lunghi, and Maria Concetta Morrone. 2020. Neuroplasticity in adult human visual cortex. *Neuroscience & Biobehavioral Reviews* 112 (2020), 542–552. URL: <https://www.sciencedirect.com/science/article/pii/S0149763419303288>. (Cited in Section 1)
- [11] Tianlong Chen, Yu Cheng, Zhe Gan, Lu Yuan, Lei Zhang, and Zhangyang Wang. 2021. Chasing Sparsity in Vision Transformers: An End-to-End Exploration. *Advances in Neural Information Processing Systems* 34 (2021). URL: <https://arxiv.org/abs/2106.04533>. (Cited in Section 3)
- [12] Tianlong Chen, Zhenyu Zhang, Pengjun Wang, Santosh Balachandra, Haoyu Ma, Zehao Wang, and Zhangyang Wang. 2022. Sparsity Winning Twice: Better Robust Generalization from More Efficient Training. *International Conference on Machine Learning* (2022). URL: <https://openreview.net/forum?id=SYuJXrXq8tw>. (Cited in Section 3)
- [13] Selima Curci, Decebal Constantin Mocanu, and Mykola Pechenizkiy. 2021. Truly Sparse Neural Networks at Scale. *arXiv preprint arXiv:2102.01732* (2021). URL: <https://arxiv.org/abs/2102.01732>. (Cited in Section C)
- [14] William Curran, Tim Brys, David Aha, Matthew Taylor, and William D Smart. 2016. Dimensionality Reduced Reinforcement Learning for Assistive Robots. In *2016 AAAI Fall Symposium Series*. URL: <https://www.aaai.org/ocs/index.php/FSS/FSS16/paper/download/14076/13660>. (Cited in Section 3)
- [15] Jonas Degraeve, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. 2022. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature* 602, 7897 (2022), 414–419. URL: <https://www.nature.com/articles/s41586-021-04301-9>. (Cited in Section 3)
- [16] Shibhansh Dohare, Richard S. Sutton, and Pablo Mahmoed. 2021. Continual Backprop: Stochastic Gradient Descent with Persistent Randomness. *arXiv preprint arXiv:2108.06325* (2021). URL: <https://arxiv.org/abs/2108.06325>. (Cited in Section 5)
- [17] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. 2020. Rigging the Lottery: Making All Tickets Winners. In *International Conference on Machine Learning*. PMLR, 2943–2952. URL: <https://arxiv.org/abs/1911.11134>. (Cited in Section 3)
- [18] Jonathan Frankle and Michael Carbin. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. *International Conference on Learning Representations*. URL: <https://arxiv.org/abs/1803.03635>. (Cited in Section 3)
- [19] Scott Fujimoto, Herke Hoof, and David Meger. 2018. Addressing Function Approximation Error in Actor-Critic Methods. In *International Conference on Machine Learning*. PMLR, 1587–1596. URL: <https://arxiv.org/abs/1802.09477>. (Cited in Section 1, 3, 4, 3)
- [20] Laura Graesser, Utku Evci, Erich Elsen, and Pablo Samuel Castro. 2022. The State of Sparse Training in Deep Reinforcement Learning. In *International Conference on Machine Learning*. PMLR, 7766–7792. URL: <https://arxiv.org/abs/2206.10369>. (Cited in Section 3, 4)
- [21] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *International Conference on Machine Learning*. PMLR, 1861–1870. URL: <https://arxiv.org/abs/1801.01290>. (Cited in Section 1, 3, 4, A, 3)
- [22] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both Weights and Connections for Efficient Neural Networks. *Advances in Neural Information Processing Systems* 28 (2015). URL: <https://arxiv.org/abs/1506.02626>. (Cited in Section 3)
- [23] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. 2021. Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research* 22, 241 (2021), 1–124. URL: <https://arxiv.org/abs/2102.00554>. (Cited in Section C)
- [24] Sara Hooker. 2021. The Hardware Lottery. *Commun. ACM* 64, 12 (2021), 58–65. URL: <https://dl.acm.org/doi/10.1145/3467017>. (Cited in Section C)
- [25] Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *International Conference for Learning Representations* (2015). URL: <https://arxiv.org/abs/1412.6980>. (Cited in Section 4, 3)
- [26] Mark Kroon and Shimon Whiteson. 2009. Automatic Feature Selection for Model-Based Reinforcement Learning in Factored MDPs. In *2009 International Conference on Machine Learning and Applications*. IEEE, 324–330. URL: <https://ieeexplore.ieee.org/document/5381529>. (Cited in Section 3)
- [27] Yuchen Li, Yifan Bao, Liyao Xiang, Junhan Liu, Cen Chen, Li Wang, and Xinbing Wang. 2021. Privacy Threats Analysis to Secure Federated Learning. *arXiv preprint arXiv:2106.13076* (2021). URL: <https://arxiv.org/abs/2106.13076>. (Cited in Section 3)
- [28] Junjie Liu, Zhe Xu, Runbin Shi, Ray Cheung, and Hayden So. 2020. Dynamic Sparse Training: Find Efficient Sparse Network From Scratch With Trainable Masked Layers. *International Conference for Learning Representations* (2020). URL: <https://arxiv.org/abs/2005.06870>. (Cited in Section 3)
- [29] Shiwei Liu, Decebal Constantin Mocanu, Amarsagar Reddy Ramapuram Matavalam, Yulong Pei, and Mykola Pechenizkiy. 2020. Sparse evolutionary Deep Learning with over one million artificial neurons on commodity hardware. *Neural Computing and Applications* 33 (2020), 2589–2604. URL: <https://arxiv.org/abs/1901.09181>. (Cited in Section C)
- [30] Shiwei Liu, Decebal Constantin Mocanu, and Mykola Pechenizkiy. 2019. On improving deep learning generalization with adaptive sparse connectivity. *arXiv preprint arXiv:1906.11626* (2019). URL: <https://arxiv.org/abs/1906.11626>. (Cited in Section 3)
- [31] Shiwei Liu, Lu Yin, Decebal Constantin Mocanu, and Mykola Pechenizkiy. 2021. Do We Actually Need Dense Over-Parameterization? In-Time Over-Parameterization in Sparse Training. In *International Conference on Machine Learning*. PMLR, 6989–7000. URL: <https://arxiv.org/abs/2102.02887>. (Cited in Section 3)
- [32] Decebal Constantin Mocanu, Elena Mocanu, Tiago Pinto, Selima Curci, Phuong H Nguyen, Madeleine Gibescu, Damien Ernst, and Zita A Vale. 2021. Sparse Training Theory for Scalable and Efficient Agents. *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems* (2021). URL: <https://arxiv.org/abs/2103.01636>. (Cited in Section 3, C)
- [33] Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. 2018. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications* 9, 1 (2018), 1–12. URL: <https://arxiv.org/abs/1707.04780>. (Cited in Section 1, 3, 4)

- [34] Janosch Moos, Kay Hansel, Hany Abdulsamad, Svenja Stark, Debora Clever, and Jan Peters. 2022. Robust Reinforcement Learning: A Review of Foundations and Recent Advances. *Machine Learning and Knowledge Extraction* 4, 1 (2022), 276–315. URL: <https://www.mdpi.com/2504-4990/4/1/13>. (Cited in Section 3)
- [35] Alberto E Pereda. 2014. Electrical synapses and their functional interactions with chemical synapses. *Nature Reviews Neuroscience* 15, 4 (2014), 250–263. URL: <https://www.nature.com/articles/nrn3708>. (Cited in Section 1)
- [36] Banafsheh Rafiee, Zaheer Abbas, Sina Ghiassian, Raksha Kumaraswamy, Richard S Sutton, Elliot A Ludvig, and Adam White. 2020. From Eye-blinks to State Construction: Diagnostic Benchmarks for Online Representation Learning. *Adaptive Behavior* (2020). URL: <https://arxiv.org/abs/2011.04590>. (Cited in Section 3)
- [37] David Silver, Satinder Singh, Doina Precup, and Richard S Sutton. 2021. Reward is enough. *Artificial Intelligence* 299 (2021), 103535. URL: <https://www.sciencedirect.com/science/article/pii/S0004370221000862>. (Cited in Section 3)
- [38] Ghada Sokar, Zahra Atashgahi, Mykola Pechenizkiy, and Decebal Constantin Mocanu. 2022. Where to Pay Attention in Sparse Training for Feature Selection?. In *Advances in Neural Information Processing Systems*. URL: <https://openreview.net/forum?id=xWvI9z37Xd>. (Cited in Section 1)
- [39] Ghada Sokar, Elena Mocanu, Decebal Constantin Mocanu, Mykola Pechenizkiy, and Peter Stone. 2022. Dynamic Sparse Training for Deep Reinforcement Learning. *The 31st International Joint Conference on Artificial Intelligence* (2022). URL: <https://arxiv.org/abs/2106.04217>. (Cited in Section 1, 3, 4, 9, A, C)
- [40] Austin Stone, Oscar Ramirez, Kurt Konolige, and Rico Jonschkowski. 2021. The Distracting Control Suite – A Challenging Benchmark for Reinforcement Learning from Pixels. *arXiv preprint arXiv:2101.02722* (2021). URL: <https://arxiv.org/abs/2101.02722>. (Cited in Section 3)
- [41] Ke Sun, Yi Liu, Yingnan Zhao, Hengshuai Yao, Shangling Jui, and Linglong Kong. 2021. Exploring the Training Robustness of Distributional Reinforcement Learning against Noisy State Observations. *arXiv preprint arXiv:2109.08776* (2021). URL: <https://arxiv.org/abs/2109.08776>. (Cited in Section 3)
- [42] Yiqin Tan, Pihe Hu, Ling Pan, and Longbo Huang. 2023. RLx2: Training a Sparse Deep Reinforcement Learning Model from Scratch. *International Conference on Learning Representations* (2023). URL: <https://arxiv.org/abs/2205.15043>. (Cited in Section 3)
- [43] Matthew E Taylor and Peter Stone. 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research* 10, 7 (2009). URL: <https://www.jmlr.org/papers/v10/taylor09a.html>. (Cited in Section 3)
- [44] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 5026–5033. URL: <https://mujoco.org/>. (Cited in Section 1, 14)
- [45] Eugene Vinitzky, Yuqing Du, Kanaad Parvate, Kathy Jang, Pieter Abbeel, and Alexandre Bayen. 2020. Robust Reinforcement Learning using Adversarial Populations. *arXiv preprint arXiv:2008.01825* (2020). URL: <https://arxiv.org/abs/2008.01825>. (Cited in Section 3)
- [46] Marc Aurel Vischer, Robert Tjarko Lange, and Henning Sprekeler. 2022. On Lottery Tickets and Minimal Task Representations in Deep Reinforcement Learning. *ICLR* (2022). URL: <https://arxiv.org/abs/2105.01648>. (Cited in Section 1, 4, D.1)
- [47] Shimon Whiteson, Peter Stone, Kenneth O Stanley, Risto Miikkilainen, and Nate Kohl. 2005. Automatic Feature Selection in Neuroevolution. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*. 1225–1232. URL: <https://dl.acm.org/doi/10.1145/1068009.1068210>. (Cited in Section 3)
- [48] Aojun Zhou, Yukun Ma, Junnan Zhu, Jianbo Liu, Zhijie Zhang, Kun Yuan, Wenxiu Sun, and Hongsheng Li. 2020. Learning N:M Fine-grained Structured Sparse Neural Networks From Scratch. *International Conference on Learning Representations* (2020). URL: <https://arxiv.org/abs/2102.04010>. (Cited in Section C)