Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

Local Pairwise Distance Matching for Backpropagation-Free Reinforcement Learning

Daniel Tanneberg

2025

Preprint:

This is an accepted article published in European Conference on Artificial Intelligence. The final authenticated version is available online at: https://doi.org/10.3233/FAIA251018

Local Pairwise Distance Matching for Backpropagation-Free Reinforcement Learning

Daniel Tanneberg

Honda Research Institute EU

Abstract. Training neural networks with reinforcement learning (RL) typically relies on backpropagation (BP), necessitating storage of activations from the forward pass for subsequent backward updates. Furthermore, backpropagating error signals through multiple layers often leads to vanishing or exploding gradients, which can degrade learning performance and stability. We propose a novel approach that trains each layer of the neural network using local signals during the forward pass in RL settings. Our approach introduces local, layer-wise losses leveraging the principle of matching pairwise distances from multi-dimensional scaling, enhanced with optional reward-driven guidance. This method allows each hidden layer to be trained using local signals computed during forward propagation, thus eliminating the need for backward passes and storing intermediate activations. Our experiments, conducted with policy gradient methods across common RL benchmarks, demonstrate that this backpropagation-free method achieves competitive performance compared to their classical BP-based counterpart. Additionally, the proposed method enhances stability and consistency within and across runs, and improves performance especially in challenging environments.

1 Introduction

Deep reinforcement learning (DRL) has achieved remarkable advances in recent years, trained primarily by stochastic gradient descent and leveraging the power of large models and extensive data [1–5]. While methods like tree-search have supplemented these techniques [2], the backbone of training these large neural networks remains end-to-end learning using backpropagation (BP) [6].

Despite its pivotal role in the success of deep (reinforcement) learning [5, 7], BP has known limitations. It is not considered a biologically plausible learning paradigm for the cortex, although some efforts have been made to model how real neurons might implement it [8–10]. Additionally, backpropagation requires to store all activations, entails delayed updates, demands computationally expensive full backward passes, and requires full knowledge of the model. These challenges have motivated the exploration of alternative training methods that do not rely on BP, an area of research that dates back to earlier studies [11–13] and has gained renewed interest in recent years [10, 14–17].

Several approaches have been developed to enable backpropagation-free training of neural networks. These include (or combine) alternative gradient calculation methods during the forward pass [11, 15, 18, 19], layer-wise (pre-)training [12, 20–22], using local information [20, 23], backward passes of targets [24–27], black-box optimization [13, 14, 28], and dedicated frameworks or models [17, 29–34].

In this paper, we propose a novel backpropagation-free approach that combines local loss functions with layer-wise training in a reinforcement learning (RL) setting. RL imposes an additional challenge as there is no fixed dataset. Instead, the training data is generated dynamically by the current model parameters, and all layers of the network contribute to this process, necessitating continuous updates. Therefore, we leverage the concept of matching pairwise distances from multi-dimensional scaling (MDS) [35–37], a well-established non-linear dimensionality reduction technique [38]. Layers are trained with a loss that aims to preserve the pairwise distances in the input data at its output, relying on information available at the training layer during the forward pass, eliminating the need for backward passes.

A similar concept of matching pairwise similarities was explored in [23], though it was applied in supervised learning settings and utilized additional neural structures. Additionally, predicting random distances has been used for unsupervised representation learning [39] for anomaly detection and clustering. Most layer-wise training approaches update each layer until convergence or add new layers while freezing previous ones [10, 12, 16, 20–22]. While suitable for (un)supervised learning with a static dataset, this is challenging in RL settings, where the training data is generated online and depends on the full current network's interaction with the environment.

A notable exception to the predominant supervised learning applications is found in [33], which presents a method that fits within frameworks utilizing dedicated neural models like predictive coding [30–32]. These methods primarily aim for biological plausibility and are not compatible with classical neural networks. In contrast, our proposed approach can be easily integrated into classical neural networks and is compatible with established RL algorithms.

In summary, we introduce a method that defines a local loss for each hidden layer based on matching pairwise distances. This method requires no backward pass or memorization of activations and is compatible with classical networks and any RL algorithm. We propose two variations of this pairwise distance-based local loss, with the additional possibility of integrating domain or task-specific distance measurements. Our experiments evaluated this approach with established policy gradient algorithms – REINFORCE [40], REINFORCE with learned baseline [41], and PPO [42] – across a set of common RL benchmark environments. The results demonstrate that our backpropagation-free approach can compete with classical backpropagation-based training, and can enhance performance and stability in many cases.

2 Preliminaries

Here, we provide a brief introduction into important concepts used in the paper. For a more detailed description we refer to existing literature, for example [5, 37, 43].

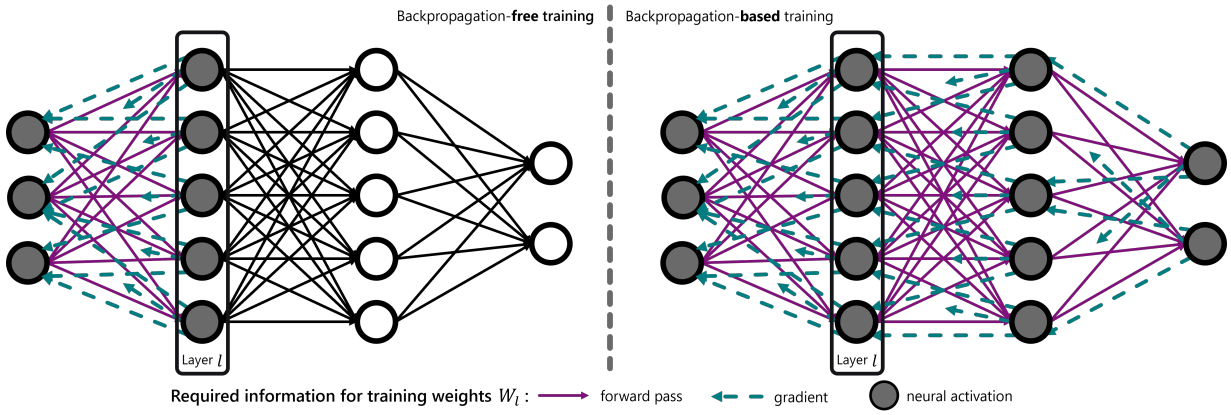


Figure 1: Overview of the required information in classical backpropagation-based training (**right**) and the proposed backpropagation-free (**left**) approach using local signals when training the weights W_l of layer l. Note that for clarity not all arrows are depicted.

2.1 *Markov Decision Process and Reinforcement Learning*

In reinforcement learning an agent interacts with the environment and receives a reward depending on its performance on a defined task. The aim of the agent is to learn a reward-maximizing behavior, known as policy. The problem is typically formalized as a Markov Decision Process (MDP) and represented as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \gamma, \mathcal{P})$, where $\mathcal{S}$ represents the set of environment states, $\mathcal{A}$ is the set of agent actions, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, γ is the discount factor, and $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the state transition probability distribution function. To learn a policy $\pi : \mathcal{S} \to \mathcal{A}$, that describes how to behave in a state, the agent aims to maximize the expected cumulative discounted reward (or return G_t), i.e., $\pi^* = \text{argmax}_\pi \, \mathbb{E}_{a_t \sim \pi(\cdot|s_t), s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)}\big[G_t\big]$, with $G_t = \sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t)$.

2.2 *Policy Gradient Methods*

Policy gradient methods optimize parameterized policies with respect to the return using gradient descent, i.e., $\theta = \theta + \alpha \nabla_\theta J$ with learning rate α. The stochastic policy π is parameterized by θ, typically a neural network called *actor*, and $J(\theta) = \mathbb{E}_\pi[G_t]$ is the objective that maximizes the return. Utilizing Markov Chain Monte-Carlo (MCMC) to approximate the expectation with samples and the policy gradient theorem, the episodic policy gradient algorithm REINFORCE [40] is given by $\nabla_\theta J = \mathbb{E}\big[\sum_{t=1}^{T} (G_t - b) \nabla_\theta \log \pi_\theta(a_t|s_t)\big]$, with an arbitrarily chosen baseline b to reduce variance and stabilize training [40, 44]. The baseline can be state dependent and learned, *state value baseline*, which reduces the aggressiveness of the update [41]. Often layers are shared for learning the policy and the baseline.

2.3 *Actor-Critic Methods*

Actor-critic methods [45] use a second parametrization ϕ, again typically a neural network called *critic*, to learn the baseline. Often *actor* and *critic* network share the hidden layers and the two learning objectives are weighted accordingly, similar as for learned baselines. The *actor* chooses an action a_t in a given state s_t, whereas the *critic* informs the *actor* how good the action was. Using the advantage function $A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$ for the *critic*, which can be estimated, for example, with temporal difference error as $A(s_t, a_t) = \mathcal{R}(s_t, a_t) + \gamma V(s_{t+1}) - V(s_t)$, the Advantage Actor Critic (A2C) [46] algorithm is realized as $\nabla_\theta J = \mathbb{E}\big[\sum_{t=1}^{T} A(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t)\big]$. Proximal Policy Optimization (PPO) [42] adds the idea of trust-region updates by restricting the policy update to stay *close* to the old policy and using a replay buffer with the concept of importance sampling for multiple updates, formally as $J^{CLIP}(\theta) = \mathbb{E}_t\big[\min(r_t(\theta)A(s_t, a_t), \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)A(s_t, a_t))\big]$ with $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ the ratio indicating the change with respect to the old policy.

2.4 *Multi-dimensional Scaling*

Multi-dimensional scaling (MDS) is a technique for non-linear dimensionality reduction [35, 37, 38], typically used to map high-dimensional data onto a low-dimensional representation for analyzing and visualization. There are different realizations of MDS, using different distance or similarity measures or cost functions to optimize [36, 37, 47]. The core idea, however, remains the same. Given a distance matrix with pairwise distances between the data points and a chosen dimension N of the low-dimensional representation, MDS tries to place each data point in the low-dimensional space such that the distances are preserved, i.e., maintaining the structure of the data.

3 Local Pairwise Distance Matching

Here, we first provide a high-level overview of the proposed method, before explaining the local losses in more detail. The proposed method is based on the observation that in neural networks the hidden layers typically learn higher-dimensional feature transformations, i.e, transforming the input data into higher-dimensional spaces and by this the network forms a hierarchy of features, that (ideally) make the decisions at the last layer *easier*. On the other hand, dimensionality reduction techniques try to map higher-dimensional data onto lower-dimensional representations. Reversing this mapping process, a local loss function for each layer can be formulated, that learns to map input data into higher-dimensional feature spaces, which enables the output layer to efficiently learn a policy. Importantly, these local loss functions for the hidden layers do not require backpropagation and only use local signals – see Figure 1 for a comparison of the required information in the proposed backpropagation-free training in contrast to classical backpropagation-based training.

Often, layers in neural networks *increase* the dimensionality, so we want to *reverse* the process of (non-linear) dimensionality reduction techniques, and learn a mapping from lower to higher dimensions – layers can also decrease the dimensionality, but here the same idea holds. Thus, our approach is that each hidden layer is learning a mapping from one feature space into another feature space with the

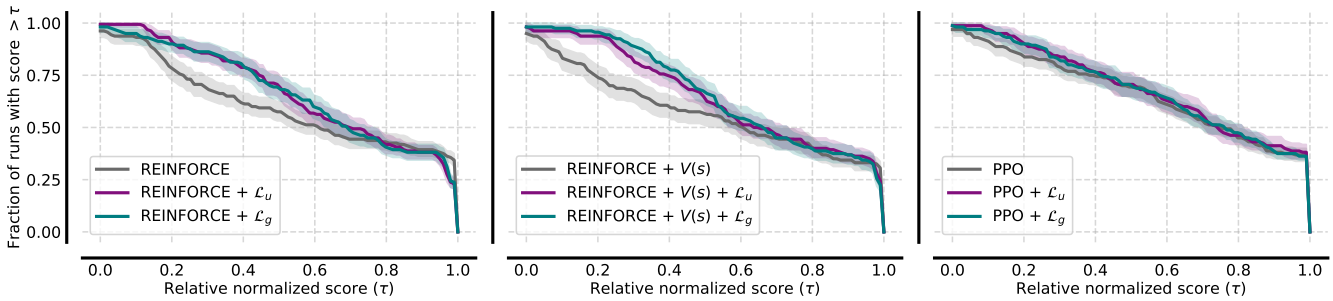


Figure 2: Performance profiles [48] of the proposed backpropagation-free method, $+\mathcal{L}_u$ and $+\mathcal{L}_g$ respectively, and their backpropagation-based baseline. Solid lines show the score distributions and shaded areas show pointwise 95% percentile stratified bootstrap CIs.

constraint that pairwise distances should be preserved – the *structure* of the data should be preserved after the feature transformation. Using only the pairwise distances between input and layer output, the learned feature transformations are unsupervised and task-agnostic, hence, enable a straightforward approach that may be beneficial for transfer, multi-task, meta, and multi-agent learning. Additionally, the pairwise distance loss can be enhanced with reward information, such that the learned feature transformations can incorporate information about the performance and are *guided* towards more *useful* transformations.

While the hidden layers of the network are trained with the proposed layer-wise pairwise distance loss, the output layer can be trained with any suitable reinforcement learning algorithm.

In the experiments we combined our approach with REINFORCE [40] without and with state value baseline ($+V(s)$), and PPO [42]. Note, in contrast to unsupervised pre-training [49–51], our approach works *online* and always trains the whole network in a single training loop. Next we describe the proposed local loss in detail.

3.1 Unsupervised Pairwise Distance Loss

For hidden layers, we define the layers' local loss with respect to the input, noted as matrix $X = [x_0, .., x_N]^\mathsf{T}$ of N stacked in-dimensional input vectors $x_n \in \mathbb{R}^{in}$. A pairwise distance matrix $D_X = \{d_{i,j} \mid i, j \in N\}$ is constructed by calculating the distances between all input vectors x_i, i.e., $d_{i,j} = \|x_i - x_j\|_1$. We used the ℓ_1-norm as distance measurement as it is better suited for higher dimensions [52] and worked best in preliminary tests, but other distance measurements are possible and may be adapted depending on the domain or model. The distance matrix D_X is normalized as $\bar{D}_X = D_X / \max D_X$, i.e., distances are normalized between $[0, 1]$ to reflect relative distances.

The out-dimensional output of a layer l for a given input h_i^{l-1} is given by $y_i^l = act(W_l^\mathsf{T} h_i^{l-1})$, with W_l the weights of the layer and a non-linear activation function act – here we used $tanh$ as activation function if not stated differently. Using the layers outputs y_i^l, the output distance matrix $\bar{D}_Y$ is created similar to $\bar{D}_X$. Importantly, the gradient between layers is stopped, i.e., using `pytorch`-like notation: $y_i^l = act(W_l^\mathsf{T} \texttt{detach}(h_i^{l-1}))$.

The hidden layers loss function is defined as minimizing the *distance* between distance matrices, i.e., the learned transformation of layer l should reflect the pairwise distances in the data. In other words, the structure of data should be preserved by matching the pairwise distances as in MDS. To learn the weights W_l, we optimize the

unsupervised loss $\mathcal{L}_u$:

$$\min_{W_l} \mathcal{L}_u(\bar{D}_X, \bar{D}_Y) = \|\bar{D}_X / out - \bar{D}_Y / in\|_F \quad , \qquad (1)$$

where $\|\cdot\|_F$ is the Frobenius norm, and both matrices are scaled by their respective data dimensionality to counter the curse of dimensionality of distances in high-dimensional spaces, i.e., guiding the learned distances to separate in high-dimensional spaces and not collapse to a relative small cluster. The proposed local loss Eq. 1 does not require backpropagation or other forms of backward passes, only forward passes are used during inference and training.

The optimization is done with stochastic gradient descent similar like *standard* training of neural networks, as

$$W_l = W_l + \alpha \nabla_{W_l} \mathcal{L}_u(\bar{D}_X, \bar{D}_Y) \quad , \qquad (2)$$

with learning rate α for the Adam optimizer [53].

This training procedure is completely unsupervised and task-independent, making it suitable for any task and setup, and the learned hidden layers, i.e., the learned feature transformations, can be directly transferred to other tasks (in the same domain) – making it interesting, for example, for multi-agent setups and meta learning.

3.2 Guided Pairwise Distance Loss

Additional information like prior knowledge or rewards may be included when constructing $\bar{D}_X$ to guide the feature learning. The target pairwise distances can be modified with such additional information by scaling the distances accordingly. One possible scaling is to add a data point dependent (learned) value to the respective rows and columns of D_X. Given the (learned) values v_i[1], we first shift and normalize them to be in the range $[0, 1]$ with $v_i = v_i - \min_i v_i$ followed by $v_i = v_i / \max_i v_i$. Next, we reverse the values to put more focus on the worse performing states by increasing their distances, and divide them by two such that the total added values for each entry in the distance matrix is in $[0, 1]$, i.e., $v_i = (1 - v_i)/2$. With D_X being the original input distance matrix and v the vector of transformed state values, the modified distance matrix $\hat{D}_X$ is created as $\hat{D}_X = D_X + v + v^\mathsf{T}$ and is normalized as before as $\hat{D}_X = \hat{D}_X / \max \hat{D}_X$. This modified distance matrix is used to create a loss similar to the unsupervised loss $\mathcal{L}_u$ in Eq. 1 but with *performance* information as the

guided loss $\mathcal{L}_g$:

$$\min_{W_l} \mathcal{L}_g(\hat{D}_X, \bar{D}_Y) = \|\hat{D}_X / out - \bar{D}_Y / in\|_F \quad , \qquad (3)$$

and is optimized as before with gradient descent as in Eq. 2. Similar as different distance measurements can be used to create the distance matrices, different modifications based on performance-based measurements can be incorporated in different ways to create different guidance using Eq. 3.

[1] In the experiments we used the normalized return (REINFORCE), the critic's state value (PPO), and the learned baseline ($+V(s)$) respectively.

4 Experiments

We evaluated the proposed method in 8 RL benchmarks using `gymnasium` [54] and `mujoco` [55]. The used environments span from rather simpler and discrete action spaces, e.g., `CartPole-v1`, up to more complex and continuous action spaces, e.g., `HalfCheetah-v4`. To show the compatibility with different RL algorithms, we evaluated our approach with and against the following algorithms: REINFORCE [40] without and with state value baseline ($+V(s)$), and PPO [42]. For each environment, we compared the three baseline algorithms using backpropagation-based training (updating all layers with the RL algorithm) against the combination of the RL algorithm (updating only the output layer) with our proposed backpropagation-free local losses $+\mathcal{L}_u$ and $+\mathcal{L}_g$ respectively.

4.1 Setup & Parameters

For all experiments we use the settings and hyperparameters listed in Table 1, with the following exceptions. Learning rate α was set to $5e^{-4}$ for `CartPole-v1`, `Acrobot-v1`, and `InvertedPendulum-v4`; to $1e^{-4}$ for `Pusher-v4`, and `HalfCheetah-v4` (PPO); and to $5e^{-5}$ for `Ant-v4` (PPO). Variance for continuous actions σ^2 set to 0.05 for `Pusher-v4`, `Ant-v4`, and `HalfCheetah-v4` (PPO). All other parameters were kept fixed, with g_{clip} for gradient clipping and discount factor γ.

Table 1: Parameters.

parameter	value
network size	$[128, 256]$
non-linearity	*tanh*
γ	0.99
σ^2	0.1
α	$3e^{-4}$
g_{clip}	1
w_{val}	0.5
PPO specific	
replayBuffer	2000
batchSize	128
epochs	10
ϵ	0.2
w_{KL}	0.01
ϵ_{value}	0.5

For REINFORCE $+V(s)$ and PPO, we used a shared network for the *actor* and *critic* with w_{val} weighting of the value loss for the backpropagation baseline, the proposed method trains all layers independently and does not need this weighting. The additional PPO hyperparameters are ϵ for clipping, w_{KL} the KL penalty weighting, and ϵ_{value} for clipping the *critic* objective. REINFORCE and REINFORCE $+V(s)$ use episode-based learning, i.e., one episode = one iteration, while PPO uses a replay buffer for updates (one iteration = one PPO update using #*epochs*). Here we use a basic PPO implementation without optimizations [56] and improvements like GAE [57], i.e., the PPO performance does not reflect its best known performance. Importantly here, however, we are interested in the relative performances and both the backpropagation baseline as well as our approaches use this same implementation. All parameters were set by empirical pre-evaluations and were not optimized for individual environments or algorithms as the scope of this paper is not creating new state of the art results, but to compare the relative performance between backpropagation-based and -free training. Hence, we used the same settings for all setups (environment + algorithm), with the aforementioned variations.

4.2 Results

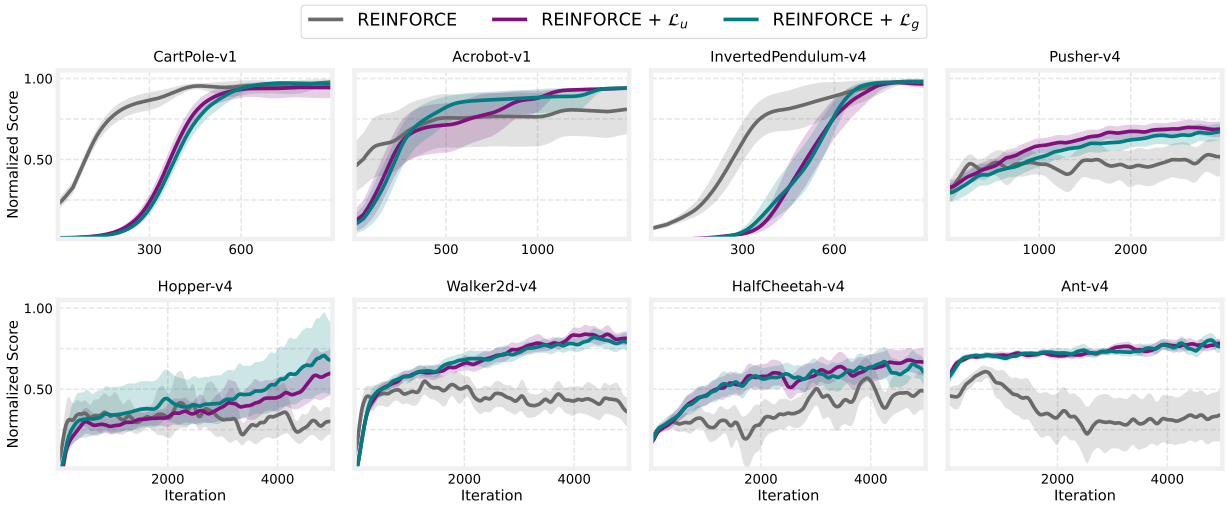
All settings, each algorithm variation for each environment, were run 20 times. The results are summarized in Figures 2 and Table 2. In Figure 2 we show the performance profiles [48] of the different algorithms, which is a recommended summary measure to compare RL methods on multiple environments/tasks. The plot shows the fraction of runs that achieve a certain relative normalized score using all runs in all environments, which allows a qualitative comparison and shows all score percentiles. The relative normalized score is calculated for each environment by using min-max normalization, with min and max computed over all algorithms and the min/max taken over the max scores averaged over 100 iterations. As these performance profiles show a summarized evaluation over all runs in all environments, we can see that the (1) the proposed backpropagation-free method can compete with the backpropagation-based baselines in terms of performance, and (2) that the proposed methods improve stability and consistency during training as their score distributions are higher especially in the lower score regimes, i.e., fewer runs got stuck in bad local optima. In the high performance regions ($\tau > 0.98$) we see that the backpropagation-based REINFORCE achieves a slightly higher peak performance. Indicating having few higher performing runs at the cost of more bad performing runs and stability.

In Figure 3 we show the performance for each environment and algorithm. Normalized scores are calculated by averaging the smoothed raw scores (cumulative reward) and using the 5% and 95% percentiles as min and max for min-max normalization. This normalization allows for relative comparisons of the methods in the environments. Across the different comparisons and environments, we see that the proposed method can achieve at least comparable performance with their BP-counterparts in all settings (as summarized before with the performance profiles). In addition, especially in the more complex environments (`Hopper-v4`, `Walker2d-v4`, `HalfCheetah-v4`, and `Ant-v4`) where there is no defined maximal performance, we often see an increased performance. The improvement in stability and consistency is also reflected in the lower spread. In some environments, it does come at the cost of speed in terms of required iterations. However, this gap is most prominent in the easier environments (`CartPole-v1`, `Acrobot-v1`, and `InvertedPendulum-v4`) that do not require many iterations in general, whereas in the remaining environments that gap is not present.

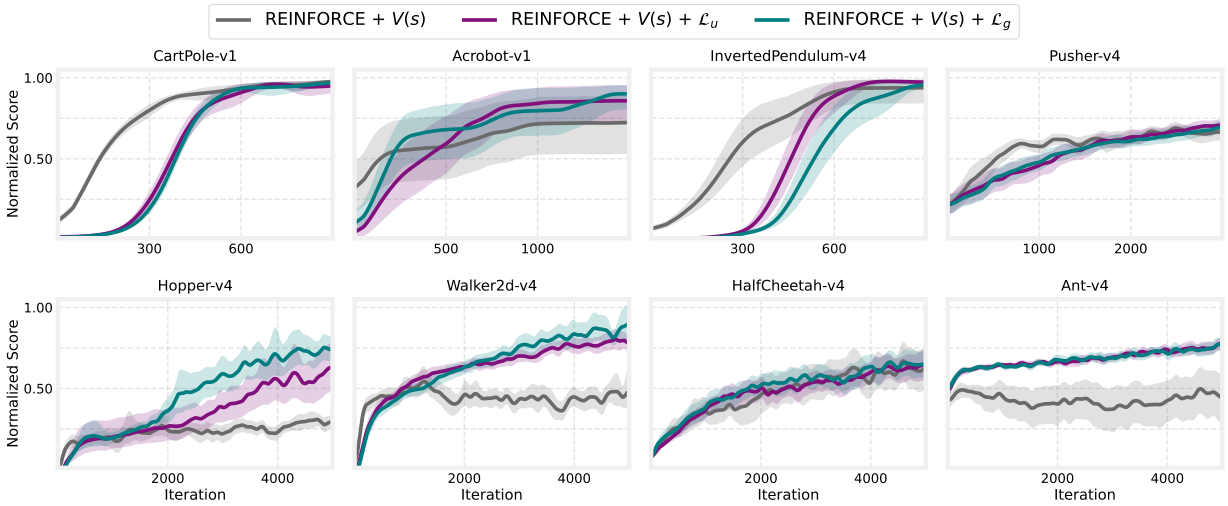
The aforementioned improvement in stability and consistency as well as the speed differences are evaluated in more detail in Table 2. Here we show the (1) *Max score*: the maximal mean score averaged over 100 iterations, (2) *Rel. spread*: the relative spread in % around this max score as the 95% CI divided by *Max score*, and (3) *Rel. iter.*: the relative iteration in which the max score was achieved in relation to the baseline. Additionally we show and highlight the relative changes in the these metrics for easier comparison and colored them in *green* if the proposed method is at least as good as the baseline, and *red* if it is more than *1.1x* worse.

Supporting the previous discussion, the *Max score* is equal or higher than the baseline in $38/48$ cases, with 8 cases only slightly worse $>=$ *0.98x*, and two *outliers* with *0.96x* and *0.92x*. Comparing the *Relative spread* we see that in $33/48$ cases the spread is smaller. Where the *relative spread* is higher, we either see low absolute spread (e.g., `CartPole-v1` or `InvertedPendulum-v4`) or a combination with stronger performance in unbounded environments (e.g., `Hopper-v4` or `Walker2d-v4`). The comparisons of *Relative iterations*, i.e., the learning speed, supports the statement from above, that the learning speed gap is mostly not existent in the more challenging environments and relative small in the easier ones.

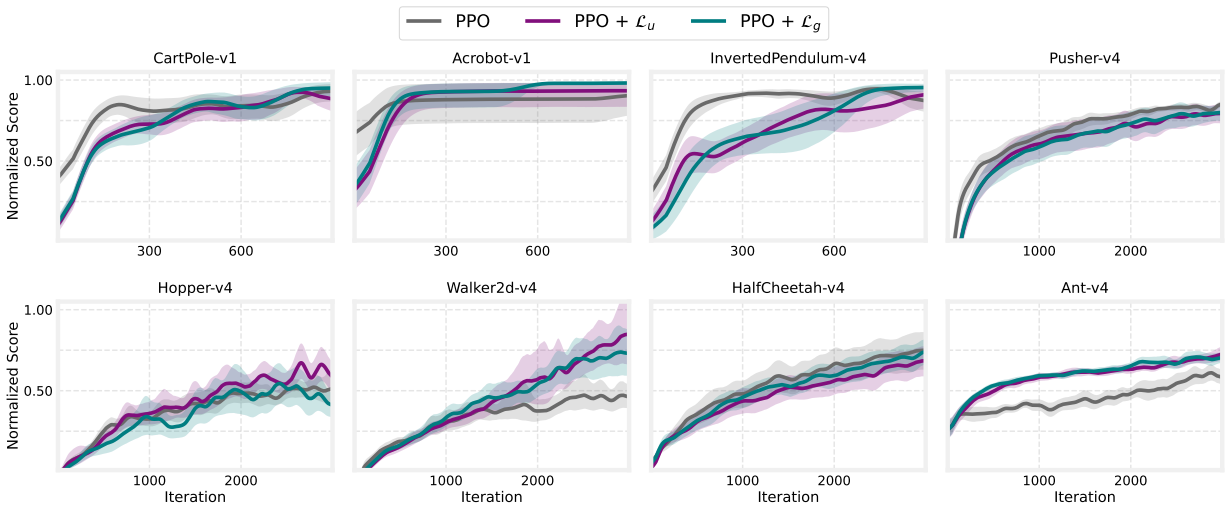
In summary, the presented evaluations show that the proposed backpropagation-free method can compete with the backpropagation-based baselines, while often improving performance and stability – especially improving consistency of the algorithms, having fewer runs stuck in bad local optima.



(a) Comparing against and with REINFORCE.



(b) Comparing against and with REINFORCE $+V(s)$.



(c) Comparing against and with PPO.

Figure 3: Comparing (a) REINFORCE, (b) REINFORCE $+V(s)$, and (c) PPO with backpropagation against the proposed backpropagation-free method with the two loss variations, $+\mathcal{L}_u$ and $+\mathcal{L}_g$ respectively. Plots show the mean (solid line) and 95% CI (shaded area) of the normalized score over 20 runs for each method and smoothed over 100 iterations.

Table 2: Comparing the proposed backpropagation-free methods ($+\mathcal{L}_u$ and $+\mathcal{L}_g$) with and against the backpropagation-based baselines of REINFORCE (REI.), REINFORCE $+V(s)$ (REI. $+V(s)$), and PPO. *Max score* indicates the maximal reached cumulative reward averaged over 100 iterations, *Rel. spread* measures the relative spread around this maximal score as the 95% CI in relation to the maximal score (in %), and *Rel. iter.* shows the iteration in which the maximal score was reached (for the baselines) and when this maximal score was matched by the proposed methods (if it was not reached, it is just their respective best iteration). The relative changes are colored *green* if the proposed method was at least as good as the baseline and *red* if it was more than *1.1x* worse.

	CartPole-v1			Acrobot-v1			InvertedPendulum-v4			Pusher-v4		
	Max score ↑	Rel. spread ↓	Rel. iter. ↓	Max score ↑	Rel. spread ↓	Rel. iter. ↓	Max score ↑	Rel. spread ↓	Rel. iter. ↓	Max score ↑	Rel. spread ↓	Rel. iter. ↓
REI.	499.3	0.24%	1485	-154.0	81.88%	1499	993.7	1.13%	1347	-53.4	8.43%	2848
$+\mathcal{L}_u$	489.5 *0.98x*	1.84% *7.65x*	1028 *0.69x*	-98.1 *1.57x*	4.89% *0.06x*	803 *0.54x*	975.8 *0.98x*	2.73% *2.42x*	803 *0.60x*	-49.6 *1.08x*	4.23% *0.50x*	763 *0.27x*
$+\mathcal{L}_g$	493.7 *0.99x*	0.95% *3.96x*	1021 *0.69x*	-98.5 *1.56x*	4.47% *0.05x*	479 *0.32x*	979.6 *0.99x*	2.01% *1.78x*	870 *0.65x*	-50.2 *1.06x*	4.58% *0.54x*	1100 *0.39x*
REI. $+V(s)$	492.6	1.64%	1299	-190.3	80.03%	1499	993.0	1.14%	1464	-47.0	3.83%	2489
$+\mathcal{L}_u$	492.3 *1.00x*	1.73% *1.05x*	1302 *1.00x*	-132.8 *1.43x*	77.41% *0.97x*	649 *0.43x*	991.7 *1.00x*	1.47% *1.29x*	1466 *1.00x*	-46.6 *1.01x*	3.65% *0.95x*	2702 *1.09x*
$+\mathcal{L}_g$	490.6 *1.00x*	1.37% *0.83x*	1374 *1.06x*	-114.5 *1.66x*	55.98% *0.70x*	723 *0.48x*	990.0 *1.00x*	1.02% *0.90x*	1358 *0.93x*	-46.9 *1.00x*	3.84% *1.00x*	2950 *1.19x*
PPO	474.5	8.13%	871	-98.3	65.31%	1497	961.2	4.09%	1356	-40.0	1.25%	2999
$+\mathcal{L}_u$	474.9 *1.00x*	4.99% *0.61x*	778 *0.89x*	-98.5 *1.00x*	65.99% *1.01x*	1479 *0.99x*	963.8 *1.00x*	3.70% *0.91x*	1225 *0.90x*	-40.8 *0.98x*	3.43% *2.75x*	2966 *0.99x*
$+\mathcal{L}_g$	484.7 *1.02x*	4.39% *0.54x*	768 *0.88x*	-77.0 *1.28x*	1.30% *0.02x*	494 *0.33x*	969.7 *1.01x*	3.46% *0.85x*	929 *0.69x*	-40.7 *0.98x*	2.95% *2.36x*	2997 *1.00x*
	Hopper-v4			Walker2d-v4			HalfCheetah-v4			Ant-v4		
REI.	331.5	52.64%	2235	251.3	27.26%	1280	993.5	60.25%	3934	570.7	14.09%	688
$+\mathcal{L}_u$	423.2 *1.28x*	33.58% *0.64x*	3620 *1.62x*	364.1 *1.45x*	11.92% *0.44x*	798 *0.62x*	1276.3 *1.28x*	29.37% *0.49x*	1731 *0.44x*	754.4 *1.32x*	7.99% *0.57x*	71 *0.10x*
$+\mathcal{L}_g$	474.8 *1.43x*	50.88% *0.97x*	1877 *0.84x*	356.2 *1.42x*	12.13% *0.44x*	702 *0.55x*	1251.8 *1.26x*	38.11% *0.63x*	1806 *0.46x*	776.6 *1.36x*	11.22% *0.80x*	96 *0.14x*
REI. $+V(s)$	397.1	16.39%	4711	295.5	26.77%	1270	1609.8	40.37%	4589	524.4	59.82%	4604
$+\mathcal{L}_u$	642.5 *1.62x*	35.42% *2.16x*	2430 *0.52x*	416.0 *1.41x*	13.99% *0.52x*	1056 *0.83x*	1542.0 *0.96x*	31.98% *0.79x*	4987 *1.09x*	828.7 *1.58x*	9.87% *0.17x*	34 *0.01x*
$+\mathcal{L}_g$	734.6 *1.85x*	16.58% *1.01x*	1835 *0.39x*	461.5 *1.56x*	22.19% *0.83x*	1447 *1.14x*	1593.2 *0.99x*	24.70% *0.61x*	4352 *0.95x*	825.3 *1.57x*	10.15% *0.17x*	5 *0.00x*
PPO	814.0	20.57%	2794	518.4	24.42%	2508	2021.4	30.50%	2999	775.2	9.16%	2874
$+\mathcal{L}_u$	980.0 *1.20x*	24.84% *1.21x*	1780 *0.64x*	741.0 *1.43x*	34.51% *1.41x*	1670 *0.67x*	1851.0 *0.92x*	26.65% *0.87x*	2999 *1.00x*	857.3 *1.11x*	7.27% *0.79x*	1337 *0.47x*
$+\mathcal{L}_g$	838.6 *1.03x*	32.73% *1.59x*	2361 *0.85x*	676.1 *1.30x*	26.77% *1.10x*	1706 *0.68x*	1998.1 *0.99x*	20.79% *0.68x*	2999 *1.00x*	844.2 *1.09x*	6.11% *0.67x*	1366 *0.48x*

5 Limitations & Discussion

The evaluations demonstrated that our proposed method offers a promising approach to backpropagation-free training of neural policies, compatible with various reinforcement learning algorithms and improving their learning behavior. However, there are open questions, potential limitations regarding the scalability and transferability of this approach, and the assumption of a global target.

Scalability: One potential limitation is scaling in terms of network depth and batch size N. The pairwise distance matrices, which are $N \times N$, may become computationally expensive as N increases, especially with full episode based updates. A potential solution to this issue is the use of sparse matrices, only considering a certain distance neighborhood, to reduce computational complexity and focus on small neighborhoods. See the extended paper [58] for initial results applying this concept. With increasing network depth, i.e., more layers trained with the pairwise distance based loss, it needs to be investigated if useful representations are still learned.

Global target: For calculating the local losses, we used $\bar{D}_X$ as the target distance matrix for all layers, assuming a global target available to all layers. This assumption may be circumvented by, for example, exploring random transformations as targets [23] or additional forward-error propagation [59]. See the extended paper [58] for initial results applying the idea of forward-error propagation.

Frozen Random Layers To evaluate if the observed benefit is due to learning meaningful representations in higher dimensions, and not due to the mapping to a higher dimension itself, we also trained agents using frozen random hidden layers. The results (see the extended paper [58]) indicate that the proposed approach is able to learn meaningful transformations into higher dimensions.

Transferability: Another potential limitation is the transferability of the approach to different network architectures, such as convolutional layers. The choice of the underlying distance metric plays a crucial role in this transferability. Here, we used the ℓ_1-norm as a distance measurement suitable for higher dimensions [52] and which worked best in preliminary tests. Calculating all pairwise distances can become expensive with larger batch sizes (e.g., the proposed approach was on average $1.5x$ slower per iteration) and investigating metrics that are suitable for fast computation via matrix operations like ℓ_2-norm or gram matrix based metrics is a useful future direction. Additionally, for other model architectures, specialized distance or similarity measurements might be more beneficial. Task-specific adap-

tations can also be incorporated, similar to our proposed guided loss $\mathcal{L}_g$, which scales the pairwise distances with rewards or performance-related feedback. The flexibility of our method allows for the use of various distance measurements tailored to specific tasks or domains.

Hyperparameters: As with all deep (reinforcement) learning methods, hyperparameters are critical. Besides the general parameters influencing the underlying RL algorithm (Section 4.1), the scaling of the distance matrices can also be considered hyperparameters. The choice of normalization and scaling should be adapted to different environments or tasks, much like the selection of distance measurements. The here used scaling might not be the best fit in all scenarios. Our approach's generality allows for such adaptations, enhancing its applicability across various settings.

Potential Benefits and Future Directions: Beyond the demonstrated benefits, the layer-wise unsupervised loss introduces a promising avenue for transferring or sharing learned representations. This could be particularly useful in multi-agent setups [60–62], meta-RL [63], or continual learning [64], where learned hidden layers can be shared while training different output layers accordingly. In settings where an offline dataset is available, the proposed method may be used for offline pre-training of the representation layers [65]. This might help the RL agent to learn more efficiently and robustly. Additionally, layer-wise training allows for different learning rates for learning representations (hidden layers) and behavior (output layer), accommodating various learning speeds [66]. Initial results on mixing learning speeds can be found in the extended paper [58]. Since our approach does not require backpropagation, the entire network does not need to be fully differentiable or even fully known. This opens up interesting possibilities for incorporating black-box operations between layers [10], further expanding the method's versatility.

In summary, while our backpropagation-free method shows promising results and a flexible framework, future research should challenge its scalability and transferability, explore different metrics and scalings for different environments, and investigate its application within other network architectures and reinforcement learning algorithms, e.g., value-based methods.

6 Conclusion

In this paper, we introduced a novel approach for backpropagation-free training of neural networks in reinforcement learning settings. Our method focuses on learning feature transformations by minimizing the difference between pairwise distance matrices in the hidden layers, leveraging the concept of multi-dimensional scaling. We demonstrated the effectiveness of our approach across a set of common RL benchmarks, showing that it is compatible with various RL algorithms. Our method not only matched but often improved the performance of traditional backpropagation-based training, while also enhancing training stability and consistency.

The versatility of the proposed method invites future research into utilizing different distance metrics and non-linear dimensionality reduction techniques. Additionally, it offers potential for leveraging the task-independent representation learning for transfer. As the interest in backpropagation-free alternatives for training neural networks regains attraction [10, 16, 17, 33], we hope this work inspires new ideas and research in alternative learning approaches, particularly within the underexplored domain of reinforcement learning.

References

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2015.

[2] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 2016.

[3] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 2017.

[4] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 2019.

[5] Xu Wang, Sen Wang, Xingxing Liang, Dawei Zhao, Jincai Huang, Xin Xu, Bin Dai, and Qiguang Miao. Deep Reinforcement Learning: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[6] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 1986.

[7] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 2015.

[8] Benjamin Scellier and Yoshua Bengio. Equilibrium Propagation: Bridging the Gap between Energy-Based Models and Backpropagation. *Frontiers in Computational Neuroscience*, 2017.

[9] Timothy P. Lillicrap, Adam Santoro, Luke Marris, Colin J. Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 2020.

[10] Geoffrey Hinton. The Forward-Forward Algorithm: Some Preliminary Investigations. *arXiv*, 2022.

[11] Andrew Barto and Michael Jordan. Gradient following without backpropagation in layered networks. In *International Conference Neural Nets*, 1987.

[12] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy Layer-Wise Training of Deep Networks. In *Neural Information Processing Systems*, 2007.

[13] Daan Wierstra, Tom Schaul, Jan Peters, and Juergen Schmidhuber. Natural Evolution Strategies. *Journal of Machine Learning Research*, 2014.

[14] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution Strategies as a Scalable Alternative to Reinforcement Learning. *arXiv*, 2017.

[15] Atılım Güneş Baydin, Barak A. Pearlmutter, Don Syme, Frank Wood, and Philip Torr. Gradients without Backpropagation. *arXiv*, 2022.

[16] Alexander Ororbia and Ankur Mali. The Predictive Forward-Forward Algorithm. *arXiv*, 2023.

[17] Jonas Guan, Shon Verch, Claas Voelcker, Ethan Jackson, Nicolas Papernot, and William Cunningham. Temporal-difference learning using distributed error signals. In *Neural Information Processing Systems*, 2024.

[18] Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, David Silver, and Koray Kavukcuoglu. Decoupled Neural Interfaces using Synthetic Gradients. In *International Conference on Machine Learning*, 2017.

[19] Mengye Ren, Simon Kornblith, Renjie Liao, and Geoffrey Hinton. Scaling Forward Gradient With Local Losses. In *International Conference on Learning Representations*, 2023.

[20] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, 2006.

[21] Heiko Wersing and Edgar Körner. Learning Optimized Features for Hierarchical Models of Invariant Object Recognition. *Neural Computation*, 2003.

[22] Mandar Kulkarni and Shirish Karande. Layer-wise training of deep networks using kernel similarity. *arXiv*, 2017.

[23] Arild Nøkland and Lars Hiller Eidnes. Training Neural Networks with Local Error Signals. In *International Conference on Machine Learning*, 2019.

[24] Yoshua Bengio. How Auto-Encoders Could Provide Credit Assignment in Deep Networks via Target Propagation. *arXiv*, 2014.

[25] Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. Difference Target Propagation. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2015.

[26] Alexander G. Ororbia and Ankur Mali. Biologically Motivated Algorithms for Propagating Local Target Representations. In *AAAI Conference on Artificial Intelligence*, 2019.

[27] Alexander Meulemans, Francesco S Carzaniga, Johan A K Suykens, João Sacramento, and Benjamin F Grewe. A Theoretical Framework for Target Propagation. In *Neural Information Processing Systems*, 2020.

[28] Daniel Tanneberg, Elmar Rueckert, and Jan Peters. Evolutionary training and abstraction yields algorithmic generalization of neural computers. *Nature Machine Intelligence*, 2020.

[29] James E. Kostas, Chris Nota, and Philip S. Thomas. Asynchronous Coagent Networks. *arXiv*, 2020.

[30] Matin Hosseini and Anthony Maida. Hierarchical Predictive Coding Models in a Deep-Learning Framework. *arXiv*, 2020.

[31] Alexander Ororbia and Daniel Kifer. The neural coding framework for learning generative models. *Nature Communications*, 2022.

[32] Beren Millidge, Tommaso Salvatori, Yuhang Song, Rafal Bogacz, and Thomas Lukasiewicz. Predictive Coding: Towards a Future of Deep Learning beyond Backpropagation? *arXiv*, 2022.

[33] Alexander Ororbia and Ankur Mali. Backprop-Free Reinforcement Learning with Active Neural Generative Coding. In *AAAI Conference on Artificial Intelligence*, 2022.

[34] Marcel Graetz, Abdullah Makkeh, Andreas C. Schneider, David A. Ehrlich, Viola Priesemann, and Michael Wibral. Infomorphic networks: Locally learning neural networks derived from partial information decomposition. *arXiv*, 2023.

[35] Warren S. Torgerson. Multidimensional scaling: I. Theory and method. *Psychometrika*, 1952.

[36] Mark Steyvers. Multidimensional scaling. *Encyclopedia of cognitive science*, 2002.

[37] Nasir Saeed, Haewoon Nam, Mian Imtiaz Ul Haq, and Dost Bhatti Muhammad Saqib. A Survey on Multidimensional Scaling. *ACM Computing Surveys*, 2018.

[38] John A. Lee and Michel Verleysen. Nonlinear Dimensionality Reduction. *Springer*, 2007.

[39] Hu Wang, Guansong Pang, Chunhua Shen, and Congbo Ma. Unsupervised Representation Learning by Predicting Random Distances. In *International Joint Conference on Artificial Intelligence*, 2020.

[40] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 1992.

[41] Jincheng Mei, Wesley Chung, Valentin Thomas, Bo Dai, Csaba Szepesvari, and Dale Schuurmans. The role of baselines in policy gradient optimization. In *Neural Information Processing Systems*, 2022.

[42] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv*, 2017.

[43] Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 2013.

[44] Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 2008.

[45] Ivo Grondman, Lucian Busoniu, Gabriel A. D. Lopes, and Robert Babuska. A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients. *IEEE Transactions on Systems, Man, and Cybernetics*, 2012.

[46] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P Lillicrap, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. In *International Conference on Machine Learning*, 2016.

[47] Jigang Sun, Malcolm Crowe, and Colin Fyfe. Extending metric multidimensional scaling with Bregman divergences. *Pattern Recognition*, 2011.

[48] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep Reinforcement Learning at the Edge of the Statistical Precipice. In *Neural Information Processing Systems*, 2021.

[49] Nicolò Botteghi, Mannes Poel, and Christoph Brune. Unsupervised Representation Learning in Deep Reinforcement Learning: A Review. *arXiv*, 2022.

[50] Zhihui Xie, Zichuan Lin, Junyou Li, Shuai Li, and Deheng Ye. Pretraining in Deep Reinforcement Learning: A Survey. *arXiv*, 2022.

[51] Tingting Zhao, Ying Wang, Wei Sun, Yarui Chen, Gang Niub, and Masashi Sugiyama. Representation Learning for Continuous Action Spaces is Beneficial for Efficient Policy Learning. *arXiv*, 2022.

[52] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. On the Surprising Behavior of Distance Metrics in High Dimensional Space. In *Database Theory: International Conference*, 2001.

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

[54] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv*, 2024.

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

[56] Shengyi Huang, Rousslan Fernand Julien Dossa, Antonin Raffin, Anssi Kanervisto, and Weixun Wang. The 37 implementation details of proximal policy optimization. In *ICLR Blog Track*, 2022. URL https://iclr-blog-track.github.io/2022/03/25/ppo-implementation-details/.

[57] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv*, 2015.

[58] Daniel Tanneberg. Local pairwise distance matching for backpropagation-free reinforcement learning. *arXiv*, 2025. URL https://arxiv.org/abs/2507.11367.

[59] Adam A. Kohan, Edward A. Rietman, and Hava T. Siegelmann. Error Forward-Propagation: Reusing Feedforward Connections to Propagate Errors in Deep Learning. *arXiv*, 2018.

[60] Felipe Leno Da Silva and Anna Helena Reali Costa. A Survey on Transfer Learning for Multiagent Reinforcement Learning Systems. *Journal of Artificial Intelligence Research*, 2019.

[61] Xiaobai Ma, David Isele, Jayesh K. Gupta, Kikuo Fujimura, and Mykel J. Kochenderfer. Recursive Reasoning Graph for Multi-Agent Reinforcement Learning. In *AAAI Conference on Artificial Intelligence*, 2022.

[62] David Rother, Thomas H. Weisswange, and Jan Peters. Disentangling Interaction Using Maximum Entropy Reinforcement Learning in Multi-Agent Systems. In *European Conference on Artificial Intelligence*, 2023.

[63] Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shimon Whiteson. A Survey of Meta-Reinforcement Learning. *arXiv*, 2023.

[64] Eli Verwimp, Rahaf Aljundi, Shai Ben-David, Matthias Bethge, Andrea Cossu, Alexander Gepperth, Tyler L. Hayes, Eyke Hüllermeier, Christopher Kanan, Dhireesha Kudithipudi, Christoph H. Lampert, Martin Mundt, Razvan Pascanu, Adrian Popescu, Andreas S. Tolias, Joost van de Weijer, Bing Liu, Vincenzo Lomonaco, Tinne Tuytelaars, and Gido M. Ven. Continual Learning: Applications and the Road Forward. *arXiv*, 2023.

[65] Mengjiao Yang and Ofir Nachum. Representation matters: Offline pretraining for sequential decision making. In *International Conference on Machine Learning*, 2021.

[66] Pierre Marion and Raphaël Berthier. Leveraging the two-timescale regime to demonstrate convergence of neural networks. In *Neural Information Processing Systems*, 2023.