MY BODY IS A CAGE: THE ROLE OF MORPHOLOGY IN GRAPH-BASED INCOMPATIBLE CONTROL

Vitaly Kurin
Department of Computer Science
University of Oxford
Oxford, United Kingdom
vitaly.kurin@cs.ox.ac.uk

Maximilian Igl
Department of Computer Science
University of Oxford
Oxford, United Kingdom
maximilian.igl@eng.ox.ac.uk

Tim Rocktäschel
Department of Computer Science
University College London
London, United Kingdom
t.rocktaschel@cs.ucl.ac.uk

Wendelin Böhmer
Department of Software Technology
Delft University of Technology
Delft, Netherlands
j.w.bohmer@tudelft.nl

Shimon Whiteson
Department of Computer Science
University of Oxford
Oxford, United Kingdom
shimon.whiteson@cs.ox.ac.uk

ABSTRACT

Multitask Reinforcement Learning is a promising way to obtain models with better performance, generalisation, data efficiency, and robustness. Most existing work is limited to *compatible* settings, where the state and action space dimensions are the same across tasks. Graph Neural Networks (GNN) are one way to address incompatible environments, because they can process graphs of arbitrary size. They also allow practitioners to inject biases encoded in the structure of the input graph. Existing work in graph-based continuous control uses the physical morphology of the agent to construct the input graph, i.e., encoding limb features as node labels and using edges to connect the nodes if their corresponded limbs are physically connected. In this work, we present a series of ablations on existing methods that show that morphological information encoded in the graph does not improve their performance. Motivated by the hypothesis that any benefits GNNs extract from the graph structure are outweighed by difficulties they create for message passing, we also propose AMORPHEUS, a transformer-based approach. Further results show that, while AMORPHEUS ignores the morphological information that GNNs encode, it nonetheless substantially outperforms GNN-based methods that use the morphological information to define the message-passing scheme.

1 INTRODUCTION

Multitask Reinforcement Learning (MTRL) (Vithayathil Varghese & Mahmoud, 2020) leverages commonalities between multiple tasks to obtain policies with better returns, generalisation, data efficiency, or robustness. Most MTRL work assumes *compatible* state-action spaces, where the dimensionality of the states and actions is the same across tasks. However, many practically important domains, such as robotics, combinatorial optimization, and object-oriented environments, have *incompatible* state-action spaces and cannot be solved by common MTRL approaches.

Incompatible environments are avoided largely because they are inconvenient for function approximation: conventional architectures expect fixed-size inputs and outputs. One way to overcome this limitation is to use Graph Neural Networks (GNNs) (Gori et al., 2005; Scarselli et al., 2005; Battaglia et al., 2018). A key feature of GNNs is that they can process graphs of arbitrary size and thus, in principle, allow MTRL in incompatible environments. However, GNNs also have a second key feature: they allow models to condition on structural information about how state features are related, e.g., how a robot's limbs are connected. In effect, this enables practitioners to incorporate additional domain knowledge where states are described as labelled graphs. Here, a graph is a collection of labelled nodes, indicating the features of corresponding objects, and edges, indicating the relations between them. In many cases, e.g., with the robot mentioned above, such domain knowledge is readily available. This results in a structural inductive bias that restricts the model's computation graph, determining how errors backpropagate through the network.

GNNs have been applied to MTRL in continuous control environments, a staple benchmark of modern Reinforcement Learning (RL), by leveraging both of the key features mentioned above (Wang et al., 2018; Huang et al., 2020). In these two works, the labelled graphs are based on the agent's physical morphology, with nodes labelled with the observable features of their corresponding limbs, e.g., coordinates, angular velocities and limb type. If two limbs are physically connected, there is an edge between their corresponding nodes. However, the assumption that it is beneficial to restrict the model's computation graph in this way has to our knowledge not been validated.

To investigate this issue, we conduct a series of ablations on existing GNN-based continuous control methods. The results show that removing morphological information does not harm the performance of these models. In addition, we propose AMORPHEUS, a new continuous control MTRL method based on transformers (Vaswani et al., 2017) instead of GNNs that use morphological information to define the message-passing scheme. AMORPHEUS is motivated by the hypothesis that any benefit GNNs can extract from the morphological domain knowledge encoded in the graph is outweighed by the difficulty that the graph creates for message passing. In a sparsely connected graph, crucial state information must be communicated across multiple hops, which we hypothesise is difficult in practice to learn. AMORPHEUS uses transformers instead, which can be thought of as fully connected GNNs with attentional aggregation (Battaglia et al., 2018). Hence, AMORPHEUS ignores the morphological domain knowledge but in exchange obviates the need to learn multi-hop communication. Similarly, in Natural Language Processing, transformers were shown to perform better without an explicit structural bias and even learn such structures from data (Vig & Belinkov, 2019; Goldberg, 2019; Tenney et al., 2019; Peters et al., 2018).

Our results on incompatible MTRL continious control benchmarks (Huang et al., 2020; Wang et al., 2018) strongly support our hypothesis: AMORPHEUS substantially outperforms GNN-based alternatives with fixed message-passing schemes in terms of sample efficiency and final performance. In addition, AMORPHEUS exhibits nontrivial behaviour such as cyclic attention patterns coordinated with gaits.

2 BACKGROUND

We now describe the necessary background for the rest of the paper.

2.1 REINFORCEMENT LEARNING

A Markov Decision Process (MDP) is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \rho_0 \rangle$. The first two elements define the set of states $\mathcal{S}$ and the set of actions $\mathcal{A}$. The next element defines the reward function $\mathcal{R}(s, a, s')$ with $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$. $\mathcal{T}(s'|s, a)$ is the probability distribution function over states $s' \in \mathcal{S}$ after taking action a in state s. The last element of the tuple ρ_0 is the distribution over initial states. Task and environment are synonyms for MDPs in this work.

A policy $\pi(a|s)$ is a mapping from states to distributions over actions. The goal of an RL agent is to find a policy that maximises the expected discounted cumulative return $J = \mathbb{E}\big[\sum_{t=0}^{\infty} \gamma^t r_t\big]$, where $\gamma \in [0, 1)$ is a discount factor, t is the discrete environment step and r_t is the reward at step t. In the MTRL setting, the agent aims to maximise the average performance across N tasks: $\frac{1}{N}\sum_{i=1}^{N} J_i$. We use MTRL *return* to denote the average performance across the tasks.

In this paper, we assume that states and actions are multivariate, but dimensionality remains constant for one MDP: $s \in \mathbb{R}^k, \forall s \in \mathcal{S}$, and $a \in \mathbb{R}^{k'}, \forall a \in \mathcal{A}$. We use $dim(\mathcal{S}) = k$ and $dim(\mathcal{A}) = k'$ to denote this dimensionality, which can differ amongst MDPs. We consider two tasks MDP$_1$ and MDP$_2$ as *incompatible* if the dimensionality of their state or action spaces disagree, i.e., $dim(\mathcal{S}_1) \neq$

$dim(\mathcal{S}_2)$ or $dim(\mathcal{A}_1) \neq dim(\mathcal{A}_2)$ with the subscript denoting a task index. In this case MTRL policies or value functions can not be represented by a Multi-layer Perceptron (MLP), which requires fixed input dimensions. We do not have additional assumptions on the semantics behind the state and action set elements and focus on the dimensions mismatch only.

Our approach, as well as the baselines in this work (Wang et al., 2018; Huang et al., 2020), use Policy Gradient (PG) methods (Peters & Schaal, 2006). PG methods optimise a policy using gradient ascent on the objective: $\theta_{t+1} = \theta_t + \alpha \nabla_\theta J|_{\theta=\theta_t}$, where θ parameterises a policy. Often, to reduce variance in the gradient estimates, one learns a critic so that the policy gradient becomes $\nabla_\theta J(\theta) = \mathbb{E}\big[\sum_t A_t^\pi \nabla_\theta \log \pi_\theta(a_t|s_t)\big]$, where A_t^π is an estimate of the advantage function (e.g., TD residual $r_t + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)$). The state-value function $V^\pi(s)$ is the expected discounted return a policy π receives starting at state s. Wang et al. (2018) use PPO (Schulman et al., 2017), which restricts a policy update to avoid instabilities from drastic changes in the policy behaviour. Huang et al. (2020) use TD3 (Fujimoto et al., 2018), a PG method based on DDPG (Lillicrap et al., 2016).

2.2 Graph Neural Networks for Incompatible Multitask RL

GNNs can address incompatible environments because they can process graphs of arbitrary sizes and topologies. A GNN is a function that takes a labelled graph as input and outputs a graph $\mathcal{G}'$ with different labels but the same topology. Here, a labelled graph $\mathcal{G} := \langle \mathcal{V}, \mathcal{E} \rangle$ consists of a set of vertices $v^i \in \mathcal{V}$, labelled with vectors $\boldsymbol{v}^i \in \mathbb{R}^{m_v}$ and a set of directed edges $e^{ij} \in \mathcal{E}$ from vertex v^i to v^j, labelled with vectors $\boldsymbol{e}^{ij} \in \mathbb{R}^{m_e}$. The output graph $\mathcal{G}'$ has the same topology but the labels can be of different dimensionality than the input, that is, $\boldsymbol{v}'^i \in \mathbb{R}^{m'_v}$ and $\boldsymbol{e}'^{ij} \in \mathbb{R}^{m'_e}$. By graph topology we mean the connectivity of the graph, which can be represented by an adjacency matrix, a binary matrix $\{a\}_{ij}$ whose elements a_{ij} equal to one iff there is an edge $e_{ij} \in \mathcal{E}$ connecting vertices $v_i, v_j \in \mathcal{V}$.

A GNN computes the output labels for entities of type k by parameterised *update functions* ϕ_ψ^k represented by neural networks that can be learnt end-to-end via backpropagation. These updates can depend on a varying number of edges or vertices, which have to be summarised first using *aggregation functions* that we denote ρ. Apart from their ability to operate on sets of elements, aggregation functions should be permutation invariant. Examples of such aggregation functions include summation, averaging and $\max$ or $\min$ operations.

Incompatible MTRL for continuous control implies learning a common policy for a set of agents with different number of limbs and connectivity of those limbs, i.e. *morphology*. To be more precise, a set of incompatible continuous control environments is a set of MDPs described in Section 2.1. When a state is represented as a graph, each node label contains features of its corresponding limb, e.g., limb type, coordinates, and angular velocity. Similarly, each factor of an action set element corresponds to a node with the label meaning the torque for a joint to emit. The typical reward function of a MuJoCo (Todorov et al., 2012) environment includes a reward for staying alive, distance covered, and a penalty for action magnitudes.

We now describe two existing approaches to incompatible control: NerveNet (Wang et al., 2018) and Shared Modular Policies (SMP) (Huang et al., 2020).

2.2.1 NerveNet

In NerveNet, the input observations are first encoded via a MLP processing each node labels as a batch element: $\boldsymbol{v}^i \leftarrow \phi_\chi\big(\boldsymbol{v}^i\big), \forall v^i \in \mathcal{V}$. After that, the message-passing part of the model block performs the following computations (in order):

$$\begin{array}{rcll} \boldsymbol{e}'^{ij} & \leftarrow & \phi_\psi^e\big(\boldsymbol{v}^i\big) & , \forall e^{ij} \in \mathcal{E}\,, \\ \boldsymbol{v}^i & \leftarrow & \phi_\xi^v\big(\boldsymbol{v}^i,\, \rho\{\boldsymbol{e}'^{ki} \,|\, e^{ki} \in \mathcal{E}\}\big) & , \forall v^i \in \mathcal{V}\,. \end{array}$$

The edge updater ϕ_ψ^e in NerveNet is an MLP which does not take the receiver's state into account. Using only one message pass restricts the learned function to local computations on the graph. The node updater ϕ_ξ^v is a Gated Recurrent Unit (GRU) (Cho et al., 2014) which maintains the internal state when doing multiple message-passing iterations, and takes the aggregated outputs of the edge updater for all incoming edges as inputs. After the message-passing stage, the MLP decoder takes the states of the nodes and, like the encoder, independently processes them, emitting scalars used as the mean for the normal distribution from which actions are sampled: $\boldsymbol{v}^i_{dec} \leftarrow \phi_\eta\big(\boldsymbol{v}^i\big), \forall v^i \in \mathcal{V}$. The standard deviation of this distribution is a separate state-independent vector with one scalar per action.

2.2.2 Shared Modular Policies

SMP is a variant of a GNN that operates only on trees. Computation is performed in two stages: top-down and bottom-up. In the first stage, information propagates level by level from leaves to the root with parents aggregating information from their children. In the second stage, information propagates from parents to the leaves with parents emitting multiple messages, one per child. The policy emits actions at the second stage of the computation together with the downstream messages.

Instead of a permutation invariant aggregation, the messages are concatenated. This, as well as separate messages for the children, also injects structural bias to the model, e.g., separating the messages for the left and right parts of robots with bilateral symmetry. In addition, its message-passing schema depends on the morphology and the choice of the root node. In fact, Huang et al. (2020) show that the root node choice can affect performance by 15%.

SMP trains a separate model for the actor and critic. An actor outputs one action per non-root node. The critic outputs a scalar per node as well. When updating a critic, a value loss is computed independently per each node with targets using the same scalar reward from the environment.

2.3 Transformers

Transformers can be seen as GNNs applied to fully connected graphs with the attention as an edge-to-vertex aggregation operation (Battaglia et al., 2018). Self-attention used in transformers is an associative memory-like mechanism that first projects the feature vector of each node $\boldsymbol{v}^i \in \mathbb{R}^{m_v}$ into three vectors: query $\boldsymbol{q}_i := \boldsymbol{\Theta}\boldsymbol{v}^i \in \mathbb{R}^d$, key $\boldsymbol{k}_i := \bar{\boldsymbol{\Theta}}\boldsymbol{v}^i \in \mathbb{R}^d$ and value $\hat{\boldsymbol{v}}_i := \hat{\boldsymbol{\Theta}}\boldsymbol{v}^i \in \mathbb{R}^{m_v}$. Parameter matrices $\boldsymbol{\Theta}, \bar{\boldsymbol{\Theta}}$, and $\hat{\boldsymbol{\Theta}}$ are learnt. The query of the receiver v_i is compared to the key value of senders using a dot product. The resulting values $\boldsymbol{w}_i$ are used as weights in the weighted sum of all the value vectors in the graph. The computation proceeds as follows:

$$\begin{array}{rcl} \boldsymbol{w}_i & := & \text{softmax}\big(\frac{[\boldsymbol{k}_1,\ldots,\boldsymbol{k}_n]^\top \boldsymbol{q}_i}{\sqrt{d}}\big) \\ \boldsymbol{v}'_i & := & [\hat{\boldsymbol{v}}_1,\ldots,\hat{\boldsymbol{v}}_n]\boldsymbol{w}_i \end{array} , \forall v_i \in \mathcal{V}\,, \tag{1}$$

with $[x_1, x_2, ..., x_n]$ being a $\mathbb{R}^{k\times n}$ matrix of concatenated vectors $x_i \in \mathbb{R}^k$. Often, multiple attention heads, i.e., $\boldsymbol{\Theta}, \bar{\boldsymbol{\Theta}}$, and $\hat{\boldsymbol{\Theta}}$ matrices, are used to learn different interactions between the nodes and mitigate the consequences of unlucky initialisation. The output of multiple heads is concatenated and later projected to respect the dimensions.

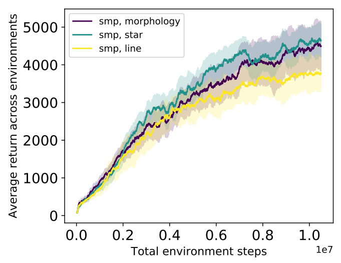
A transformer block is a combination of an attention block and a feedforward layer with a possible normalisation between them. In addition, there are residual connections from the input to the attention output and from the output of the attention to the feedforward layer output. Transformer blocks can be stacked together to take higher order dependencies into account, i.e., reacting not only to the features of the nodes, but how the features of the nodes change after applying a transformer block.

3 The Role of Morphology in Existing Work

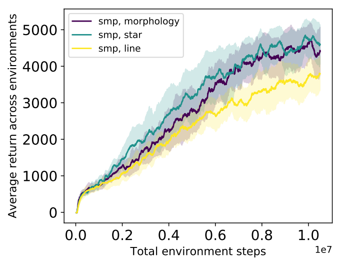
In this section, we provide evidence against the assumption that GNNs improve performance by exploiting information about physical morphology (Huang et al., 2020; Wang et al., 2018). Here and in all of the following sections, we run experiments for three random seeds and report the average undiscounted MTRL return and the standard error across the seeds.

To determine if information about the agent's morphology encoded in the relational graph structure is essential to the success of SMP, we compare its performance given full information about the structure (morphology), given no information about the structure (star), and given a structural bias unrelated to the agent's morphology (line). Ideally, we would test a fully connected architecture as well, but SMP only works with trees. Figure 9 in Appendix B illustrates the tested topologies.

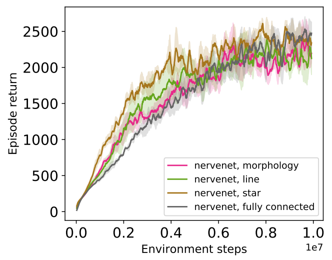
The results in Figure 1a and 1b demonstrate that, surprisingly, performance is not contingent on having information about the physical morphology. A `star` agent performs on par with the



(a) SMP, `Walker++`



(b) SMP, `Humanoid++`



(c) NERVENET, `Walkers`

Figure 1: Neither SMP nor NERVENET leverage the agent's morphological information, or the positive effects are outweighted by their negative effect on message passing.

`morphology` agent, thus refuting the assumption that the method learns because it exploits information about the agent's physical morphology. The `line` agent performs worse, perhaps because the network must propagate messages even further away, and information is lost with each hop due to the finite size of the MLPs causing information bottlenecks (Alon & Yahav, 2020).

We also present similar results for NERVENET. Figure 1c shows that all of the variants we tried perform similarly well on `Walkers` from (Wang et al., 2018), with `star` being marginally better. Since NERVENET can process non-tree graphs, we also tested a fully connected variant. This version learns more slowly at the beginning, probably because of difficulties with differentiating nodes at the aggregation step. Interestingly, in contrast to SMP, in NERVENET `line` performs on par with `morphology`. This might be symptomatic of problems with the message-passing mechanism of SMP, e.g., bottlenecks leading to information loss.

4 AMORPHEUS

Inspired by the results above, we propose AMORPHEUS, a transformer-based method for incompatible MTRL in continuous control. AMORPHEUS is motivated by the hypothesis that any benefit GNNs can extract from the morphological domain knowledge encoded in the graph is outweighed by the difficulty that the graph creates for message passing. In a sparse graph, crucial state information must be communicated across multiple hops, which we hypothesise is difficult to learn in practice.

AMORPHEUS belongs to the encode-process-decode family of architectures (Battaglia et al., 2018) with a transformer at its core. Since transformers can be seen as GNNs operating on fully connected graphs, this approach allows us to learn a message passing schema for each state and each pass separately, and limits the number of message passes needed to propagate sufficient information through the graph. Multi-hop message propagation in the presence of aggregation, which could cause problems with gradient propagation and information loss, is no longer required. We implement both actor and critic in the SMP codebase (Huang et al., 2020) and made our implementation available online at `https://github.com/yobibyte/amorpheus`. Like in SMP, there is no weight

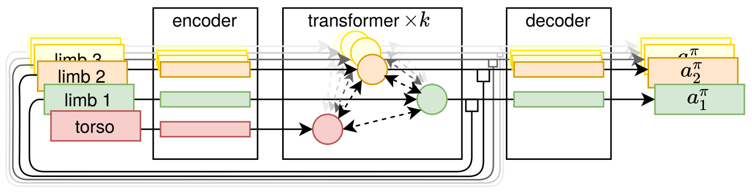


Figure 2: AMORPHEUS architecture. Lines with squares at the end denote concatenation. Arrows going separately through encoder and decoder denote that rows of the input matrix are processed independently as batch elements. Dashed arrows denote message-passing in a transformer block. The diagram depicts the policy network, the critic has an identical architecture, with the decoder outputs interpreted as value function values.

sharing between the actor and the critic. Both of them consist of three parts: a linear encoder, a transformer in the middle, and the output decoder MLP.

Figure 2 illustrates the AMORPHEUS architecture (policy). The encoder and decoder process each node independently, as if they are different elements of a mini-batch. Like SMP, the policy network has one output per graph node. The critic has the same architecture as on Figure 2, and, as in Huang et al. (2020), each critic node outputs a scalar with the value loss independently computed per node.

Similarly to NERVENET and SMP, AMORPHEUS is modular and can be used in incompatible environments, including those not seen in training. In contrast to SMP which is constrained by the maximum number of children per node seen at the model initialisation in training, AMORPHEUS can be applied to any other morphology with no constraints on the physical connectivity.

Instead of one-hot encoding used in natural language processing, we apply a linear layer on node observations. Each node observation uses the same state representation as SMP and includes a limb type (e.g. hip or shoulder), position with a relative x coordinate of the limb with respect to the torso, positional and rotational velocities, rotations, angle and possible range of the values for the angle normalised to $[0, 1]$. We add residual connections from the input features to the decoder output to avoid the nodes forgetting their own features by the time the decoder independently computes the actions. Both actor and critic use two attention heads for each of the three transformer layers. Layer Normalisation (Ba et al., 2016) is a crucial component of transformers which we also use in AMORPHEUS. See Appendix A for more details on the implementation.

4.1 EXPERIMENTAL RESULTS

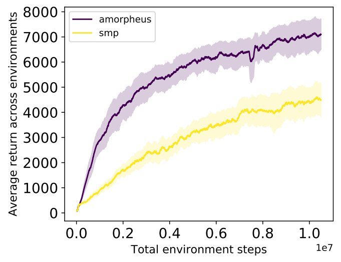
We first test AMORPHEUS on the set of MTRL environments proposed by Huang et al. (2020). For `Walker++`, we omit flipped environments, since Huang et al. (2020) implement flipping on the model level. For AMORPHEUS, the flipped environments look identical to the original ones. Our experiments in this Section are built on top of the TD3 implementation used in Huang et al. (2020).

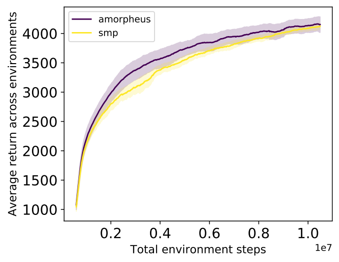
Figure 3 supports our hypothesis that explicit morphological information encoded in graph topology is not needed to yield a single policy achieving high average returns across a set of incompatible continuous control environments. Free from the need to learn multi-hop communication and equipped with the attention mechanism, AMORPHEUS clearly outperforms SMP, the state-of-the-art algorithm for incompatible continuous control. Huang et al. (2020) report that training SMP on `Cheetah++` together with other environments makes SMP unstable. By contrast, AMORPHEUS has no trouble learning in this regime (Figure 3g and 3h).

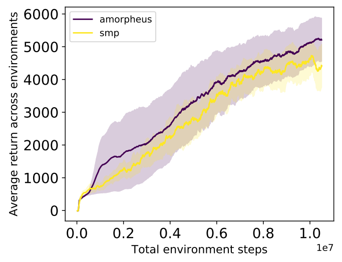
Our experiments demonstrate that node features have enough information for AMORPHEUS to perform the task and limb discrimination needed for successful MTRL continuous control policies. For example, a model can distinguish left from right, not from structural biases as in SMP, but from the relative position of the limb w.r.t. the root node provided in the node features.

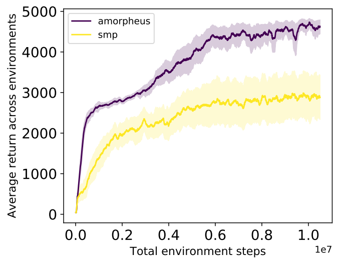
While the total number of tasks in the SMP benchmarks is high, they all share one key characteristic. All tasks in a benchmark are built using subsets of the limbs from an archetype (e.g., `Walker++` or `Cheetah++`). To verify that our results hold more broadly, we adapted the `Walkers` benchmark (Wang et al., 2018) and compared AMORPHEUS with SMP and NERVENET on it. This benchmark includes five agents with different morphologies: a Hopper, a HalfCheetah, a FullCheetah, a Walker, and an Ostrich. The results in Figure 4 are consistent[1] with our previous experiments, demonstrating the benefits of AMORPHEUS' fully-connected graph with attentional aggregation.

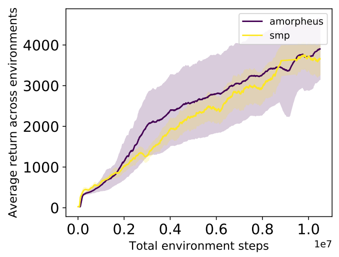
[1]Note that the performance of NERVENET is not directly comparable, as the observational features and the learning algorithm differ from AMORPHEUS and SMP. We do not test NERVENET on SMP benchmarks because the codebases are not compatible and comparing NERVENET and SMP is not the focus of the paper. Even if we implemented NERVENET in the SMP training loop, it is unclear how the critic of NERVENET would perform in a new setting. The original paper considers two options for the critic: one GNN-based and one MLP-based. We use the latter in Figure 4 as the former takes only the root node output labels as an input and is thus most likely to face difficulty in learning multi-hop message-passing. The MLP critic should perform better because training an MLP is easier, though it might be sample-inefficient when the number of tasks is large. For example, in `Cheetah++` an agent would need to learn 12 different critics. Finally, NERVENET learns a separate MLP encoder per task, partially defeating the purpose of using GNN for incompatible environments.

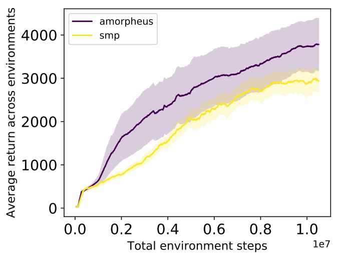

(a) Walker++

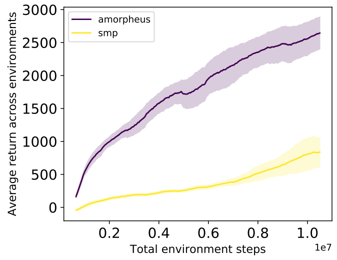

(b) Cheetah++


(c) Humanoid++


(d) Hopper++


(e) Walker-Humanoid++


(f) Walker-Humanoid-Hopper++


(g) Cheetah-Walker-Humanoid++

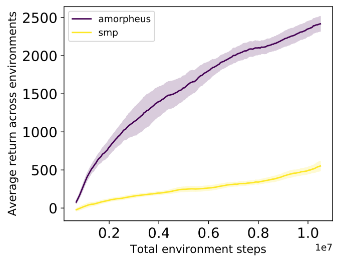

(h) Cheetah-Walker-Humanoid-Hopper++

Figure 3: AMORPHEUS consistently outperforms SMP on MTRL benchmarks from Huang et al. (2020), supporting our hypothesis that no explicit structural information is needed to learn a successful MTRL policy and that facilitated message-passing procedure results in faster learning.

While we focused on MTRL in this work, we also evaluated AMORPHEUS in a zero-shot generalisation setting. Table 3 in Appendix D provides initial results demonstrating AMORPHEUS's potential.

4.2 ATTENTION MASK ANALYSIS

GNN-based policies, especially those that use attention, are more interpretable than monolithic MLP policies. We now analyse the attention masks that AMORPHEUS learns. Having an implicit structure that is state dependent is one of the benefits of AMORPHEUS (every node has access to other nodes' annotations, and the aggregation weights depend on the input as shown in Equation 1). By contrast, NERVENET and SMP have a rigid message-passing structure that does not change throughout training or throughout a rollout. Indeed, Figure 5 shows a variety of masks a `Walker++` model exhibits within a `Walker-7` rollout, confirming that AMORPHEUS attends to different parts of the state space based on the input.

Both Wang et al. (2018) and Huang et al. (2020) notice periodic patterns arising in their models. Smilarly, AMORPHEUS demonstrates cycles in attention masks, usually arising for the first layer of the transformer. Figure 6 shows the column-wise sum of the attention masks coordinated with an upper-leg limb of a `Walker-7` agent. Intuitively, the column-wise sum shows how much other nodes are interested in the node corresponding to that column.

Interestingly, attention masks in earlier layers change more slowly within a rollout than those of the downstream layers. Figure 13 in Appendix C.2 demonstrates this phenomenon for three different

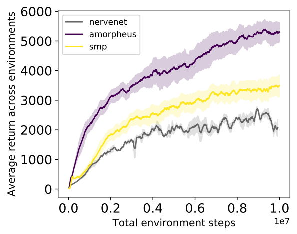


Figure 4: MTRL performance on `Walkers` (Wang et al., 2018).

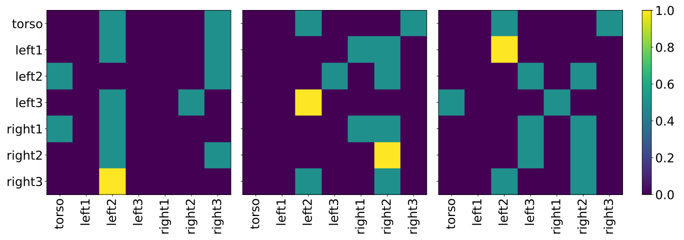


Figure 5: State-dependent masks of AMORPHEUS (3rd attention layer) within a `Walker-7` rollout.

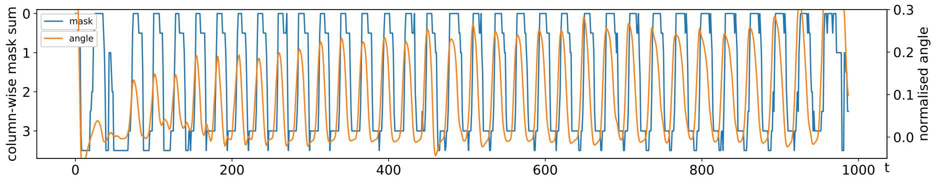


Figure 6: In the first attention layer of a `Walker-7` rollout, nodes attend to an upper leg (column-wise mask sum ~ 3) when the leg is closer to the ground (normalized angle ~ 0).

`Walker++` models tested on `Walker-7`. This shows that AMORPHEUS might, in principle, learn a rigid structure (as in GNNs) if needed.

Finally, we investigate how attention masks evolve over time. Early in training, the masks are spread across the whole graph. Later on, the mask weights distributions become less uniform. Figures 10, 11 and 12 in Appendix C.1 demonstrate this phenomenon on `Walker-7`.

5 RELATED WORK

Most MTRL research considers the compatible case (Rusu et al., 2016; Parisotto et al., 2016; Teh et al., 2017; Vithayathil Varghese & Mahmoud, 2020). MTRL for continuous control is often done from pixels with CNNs solving part of the compatibility issue. DMLab (Beattie et al., 2016) is a popular choice when learning from pixels with a compatible action space shared across the environments (Hessel et al., 2019; Song et al., 2020).

GNNs started to stretch the possibilities of RL allowing MTRL in incompatible environments. Khalil et al. (2017) learn combinatorial optimisation algorithms over graphs. Kurin et al. (2020) learn a branching heuristic of a SAT solver. Applying approximations schemes typically used in RL to these settings is impossible, because they expect input and output to be of fixed size. Another form of (potentially incompatible) RL using message passing are coordination graphs (e.g. DCG, Boehmer et al., 2020), that use the max-plus algorithm (Pearl, 1989) to coordinate action selection between multiple agents. One can apply DCG in single-agent RL using ideas of Tavakoli et al. (2021).

Several methods for incompatible continuous control have also been proposed. Chen et al. (2018) pad the state vector with zeros to have the same dimensionality for robots with different number of joints, and condition the policy on the hardware information of the agent. D'Eramo et al. (2020) demonstrate a positive effect of learning a common network for multiple tasks, learning a specific encoder and a decoder one per task. We expect this method to suffer from sample-inefficiency because it has to learn separate input and output heads per each task. Moreover, Wang et al. (2018) have a similar implementation of their MTRL baseline showing that GNNs have benefits over MLPs for incompatible control. Huang et al. (2020), whose work is the main baseline in this paper, apply a GNN-like approach and study its MTRL and generalisation properties. The method can be used only with trees, its aggregation function is not permutation invariant, and the message-passing schema stays fixed throughout the training procedure. Wang et al. (2018) and Huang et al. (2020) attribute the effectiveness of their methods to the ability of the GNNs to exploit information about agent morphology. In this work, we present evidence against this hypothesis, showing that existing approaches do not exploit morphological information as was previously believed.

Attention mechanisms have also been used in the RL setting. Zambaldi et al. (2018) consider self-attention to deal with an object-oriented state space. They further generalize this to variable action spaces and test generalisation on Starcraft-II mini-games that have a varying number of units and other environmental entities. Duan et al. (2017) apply attention for both temporal dependency and a factorised state space (different objects in the scene) keeping the action space compatible. Parisotto et al. (2020) use transformers as a replacement for a recurrent policy. Loynd et al. (2020) use transformers to add history dependence in a POMDP as well as for factored observations, having a node per game object. The authors do not consider a factored action space, with the policy receiving the aggregated information of the graph after the message passing ends. Baker et al. (2020) use self-attention to account for a factored state-space to attend over objects or other agents in the scene. AMORPHEUS does not use a transformer for recurrency but for the factored state and action spaces, with each non-torso node having an action output. Iqbal & Sha (2019) apply attention to generalise MTRL multi-agent policies over varying environmental objects and Iqbal et al. (2020) extend this to a factored action space by summarising the values of all agents with a mixing network (Rashid et al., 2020). Li et al. (2020) learn embeddings for a multi-agent actor-critic architecture by generating the weights of a graph convolutional network (GCN, Kipf & Welling, 2017) with attention. This allows a different topology in every state, similar to AMORPHEUS, which goes one step further and allows to change the topology in every round of message passing.

Another line of work aims to infer graph topology instead of hardcoding one. Differentiable Graph Module (Kazi et al., 2020) predicts edge probabilities doing a continuous relaxation of k-nearest neighbours to differentiate the output with respect to the edges in the graph. Johnson et al. (2020) learn to augment a given graph with additional edges to improve the performance of a downstream task. Kipf et al. (2018) use variational autoencoders (Kingma & Welling, 2014) using a GNN for reconstruction. Notably, the authors notice that message passing on a fully connected graph might work better than when restricted by skeleton when evaluated on human motion capture data.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we investigated the role of explicit morphological information in graph-based continous control. We ablated existing methods SMP and NERVENET, providing evidence against the belief that these methods improve performance by exploiting explicit morphological structure encoded in graph edges. Motivated by our findings, we presented AMORPHEUS, a transformer-based method for MTRL in incompatible environments. AMORPHEUS obviates the need to propagate messages far away in the graph and can attend to different regions of the observations depending on the input and the particular point in training. As a result, AMORPHEUS clearly outperforms existing work in incompatible continuous control. In addition, AMORPHEUS exhibits non-trivial behaviour such as periodic cycles of attention masks coordinated with the gait. The results show that information in the node features alone is enough to learn a successful MTRL policy. We believe our results further push the boundaries of incompatible MTRL and provide valuable insights for further progress.

One possible drawback of AMORPHEUS is its computational complexity. Transformers suffer from quadratic complexity in the number of nodes with a growing body of work addressing this issue (Tay et al., 2020). However, the number of the nodes in continuous control problems is relatively low compared to much longer sequences used in NLP (Devlin et al., 2019). Moreover, Transformers are higly parallelisable, compared to SMP with the data dependency across tree levels (the tree is processed level by level with each level taking the output of the previous level as an input).

We focused on investigating the effect of injecting explicit morphological information into the model. However, there are also opportunities to improve the learning algorithm itself. Potential directions of improvement include averaging gradients instead of performing sequential task updates, or balancing tasks updates with multi-armed bandits or PopArt (Hessel et al., 2019).

ACKNOWLEDGMENTS

VK is a doctoral student at the University of Oxford funded by Samsung R&D Institute UK through the AIMS program. SW has received funding from the European Research Council under the European Union's Horizon 2020 research and innovation programme (grant agreement number 637713). The experiments were made possible by a generous equipment grant from NVIDIA. The authors would like to thank Henry Kenlay and Marc Brockschmidt for useful discussions on GNNs.

REFERENCES

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. *CoRR*, abs/2006.05205, 2020. URL `https://arxiv.org/abs/2006.05205`.

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016. URL `http://arxiv.org/abs/1607.06450`.

Bowen Baker, Ingmar Kanitscheider, Todor M. Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=SkxpxJBKwS`.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çaglar Gülçehre, H. Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey R. Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018. URL `http://arxiv.org/abs/1806.01261`.

Charles Beattie, Joel Z. Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, Julian Schrittwieser, Keith Anderson, Sarah York, Max Cant, Adam Cain, Adrian Bolton, Stephen Gaffney, Helen King, Demis Hassabis, Shane Legg, and Stig Petersen. Deepmind lab. *CoRR*, abs/1612.03801, 2016. URL `http://arxiv.org/abs/1612.03801`.

Wendelin Boehmer, Vitaly Kurin, and Shimon Whiteson. Deep coordination graphs. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 980–991. PMLR, 2020. URL `http://proceedings.mlr.press/v119/boehmer20a.html`.

Tao Chen, Adithyavairavan Murali, and Abhinav Gupta. Hardware conditioned policies for multi-robot transfer learning. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 9355–9366, 2018. URL `https://proceedings.neurips.cc/paper/2018/hash/b8cfbf77a3d250a4523ba67a65a7d031-Abstract.html`.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans (eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1724–1734. ACL, 2014. doi: 10.3115/v1/d14-1179. URL `https://doi.org/10.3115/v1/d14-1179`.

Carlo D'Eramo, Davide Tateo, Andrea Bonarini, Marcello Restelli, and Jan Peters. Sharing knowledge in multi-task deep reinforcement learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=rkgpv2VFvr`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL `https://doi.org/10.18653/v1/n19-1423`.

Yan Duan, Marcin Andrychowicz, Bradly C. Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 1087–1098, 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/ba3866600c3540f67c1e9575e213be0a-Abstract.html`.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1582–1591. PMLR, 2018. URL `http://proceedings.mlr.press/v80/fujimoto18a.html`.

Yoav Goldberg. Assessing bert's syntactic abilities. *CoRR*, abs/1901.05287, 2019. URL `http://arxiv.org/abs/1901.05287`.

Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pp. 729–734. IEEE, 2005.

Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado van Hasselt. Multi-task deep reinforcement learning with popart. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 3796–3803. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33013796. URL `https://doi.org/10.1609/aaai.v33i01.33013796`.

Wenlong Huang, Igor Mordatch, and Deepak Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 4455–4464. PMLR, 2020. URL `http://proceedings.mlr.press/v119/huang20d.html`.

Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2961–2970. PMLR, 2019. URL `http://proceedings.mlr.press/v97/iqbal19a.html`.

Shariq Iqbal, Christian A. Schröder de Witt, Bei Peng, Wendelin Böhmer, Shimon Whiteson, and Fei Sha. AI-QMIX: attention and imagination for dynamic multi-agent reinforcement learning. *CoRR*, abs/2006.04222, 2020. URL `https://arxiv.org/abs/2006.04222`.

Daniel D. Johnson, Hugo Larochelle, and Daniel Tarlow. Learning graph structure with A finite-state automaton layer. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/1fdc0ee9d95c71d73df82ac8f0721459-Abstract.html`.

Anees Kazi, Luca Cosmo, Nassir Navab, and Michael M. Bronstein. Differentiable graph module (DGM) graph convolutional networks. *CoRR*, abs/2002.04999, 2020. URL `https://arxiv.org/abs/2002.04999`.

Elias B. Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6348–6358, 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/d9896106ca98d3d05b8cbdf4fd8b13a1-Abstract.html`.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun (eds.), *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL `http://arxiv.org/abs/1312.6114`.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL `https://openreview.net/forum?id=SJU4ayYgl`.

Thomas N. Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard S. Zemel. Neural relational inference for interacting systems. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2693–2702. PMLR, 2018. URL `http://proceedings.mlr.press/v80/kipf18a.html`.

Vitaly Kurin, Saad Godil, Shimon Whiteson, and Bryan Catanzaro. Can q-learning with graph networks learn a generalizable branching heuristic for a SAT solver? In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/6d70cb65d15211726dcce4c0e971e21c-Abstract.html`.

Sheng Li, Jayesh K. Gupta, Peter Morales, Ross E. Allen, and Mykel J. Kochenderfer. Deep implicit coordination graphs for multi-agent reinforcement learning. *CoRR*, abs/2006.11438, 2020. URL `https://arxiv.org/abs/2006.11438`.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In Yoshua Bengio and Yann LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL `http://arxiv.org/abs/1509.02971`.

Ricky Loynd, Roland Fernandez, Asli Çelikyilmaz, Adith Swaminathan, and Matthew J. Hausknecht. Working memory graphs. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6404–6414. PMLR, 2020. URL `http://proceedings.mlr.press/v119/loynd20a.html`.

Emilio Parisotto, Lei Jimmy Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. In Yoshua Bengio and Yann LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL `http://arxiv.org/abs/1511.06342`.

Emilio Parisotto, H. Francis Song, Jack W. Rae, Razvan Pascanu, Çaglar Gülçehre, Siddhant M. Jayakumar, Max Jaderberg, Raphaël Lopez Kaufman, Aidan Clark, Seb Noury, Matthew Botvinick, Nicolas Heess, and Raia Hadsell. Stabilizing transformers for reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7487–7498. PMLR, 2020. URL `http://proceedings.mlr.press/v119/parisotto20a.html`.

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

Judea Pearl. *Probabilistic reasoning in intelligent systems - networks of plausible inference*. Morgan Kaufmann series in representation and reasoning. Morgan Kaufmann, 1989.

Jan Peters and Stefan Schaal. Policy gradient methods for robotics. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2006, October 9-15, 2006, Beijing, China*, pp. 2219–2225. IEEE, 2006. doi: 10.1109/IROS.2006.282564. URL `https://doi.org/10.1109/IROS.2006.282564`.

Matthew E. Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. Dissecting contextual word embeddings: Architecture and representation. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 1499–1509. Association for Computational Linguistics, 2018. doi: 10.18653/v1/d18-1179. URL `https://doi.org/10.18653/v1/d18-1179`.

Tabish Rashid, Mikayel Samvelyan, Christian Schröder de Witt, Gregory Farquhar, Jakob N. Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *J. Mach. Learn. Res.*, 21:178:1–178:51, 2020. URL `http://jmlr.org/papers/v21/20-081.html`.

Andrei A. Rusu, Sergio Gomez Colmenarejo, Çaglar Gülçehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. In Yoshua Bengio and Yann LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL `http://arxiv.org/abs/1511.06295`.

Franco Scarselli, Sweah Liang Yong, Marco Gori, Markus Hagenbuchner, Ah Chung Tsoi, and Marco Maggini. Graph neural networks for ranking web pages. In Andrzej Skowron, Rakesh Agrawal, Michael Luck, Takahira Yamaguchi, Pierre Morizet-Mahoudeaux, Jiming Liu, and Ning Zhong (eds.), *2005 IEEE / WIC / ACM International Conference on Web Intelligence (WI 2005), 19-22 September 2005, Compiegne, France*, pp. 666–672. IEEE Computer Society, 2005. doi: 10.1109/WI.2005.67. URL `https://doi.org/10.1109/WI.2005.67`.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL `http://arxiv.org/abs/1707.06347`.

Sequence-to-Sequence Modeling, Pytorch Tutorial. Sequence-to-sequence modeling with nn.transformer and torchtext. URL `https://pytorch.org/tutorials/beginner/transformer_tutorial.html`. [Online; accessed 8-August-2020].

H. Francis Song, Abbas Abdolmaleki, Jost Tobias Springenberg, Aidan Clark, Hubert Soyer, Jack W. Rae, Seb Noury, Arun Ahuja, Siqi Liu, Dhruva Tirumala, Nicolas Heess, Dan Belov, Martin A. Riedmiller, and Matthew M. Botvinick. V-MPO: on-policy maximum a posteriori policy optimization for discrete and continuous control. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL `https://openreview.net/forum?id=SylOlp4FvH`.

Arash Tavakoli, Mehdi Fatemi, and Petar Kormushev. Learning to represent action values as a hypergraph on the action vertices. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=Xv_s64FiXTv`.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *CoRR*, abs/2009.06732, 2020. URL `https://arxiv.org/abs/2009.06732`.

Yee Whye Teh, Victor Bapst, Wojciech M. Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 4496–4506, 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/0abdc563a06105aee3c6136871c9f4d1-Abstract.html`.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. What do you learn from context? probing for sentence structure in contextualized word representations. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL `https://openreview.net/forum?id=SJzSgnRcKX`.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109. URL `https://doi.org/10.1109/IROS.2012.6386109`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html`.

Jesse Vig and Yonatan Belinkov. Analyzing the structure of attention in a transformer language model. *CoRR*, abs/1906.04284, 2019. URL `http://arxiv.org/abs/1906.04284`.

Nelson Vithayathil Varghese and Qusay H. Mahmoud. A survey of multi-task deep reinforcement learning. *Electronics*, 9(9):1363–1384, Aug 2020. ISSN 2079-9292. doi: 10.3390/electronics9091363. URL `http://dx.doi.org/10.3390/electronics9091363`.

Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. Nervenet: Learning structured policy with graph neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL `https://openreview.net/forum?id=S1sqHMZCb`.

Vinícius Flores Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David P. Reichert, Timothy P. Lillicrap, Edward Lockhart, Murray Shanahan, Victoria Langston, Razvan Pascanu, Matthew Botvinick, Oriol Vinyals, and Peter W. Battaglia. Relational deep reinforcement learning. *CoRR*, abs/1806.01830, 2018. URL `http://arxiv.org/abs/1806.01830`.

A Reproducibility

We initially took the transformer implementation from the Official Pytorch Tutorial (Sequence-to-Sequence Modeling, Pytorch Tutorial) which uses `TransformerEncoderLayer` from Pytorch (Paszke et al., 2017). We modified it for the regression task instead of classification, and removed masking and the positional encoding. Table 1 provides all the hyperparameters needed to replicate our experiments.

Table 1: Hyperparameters of our experiments

Hyperparameter	**Value**	**Comment**
Amorpheus		
– Learning rate	0.0001	
– Gradient clipping	0.1	
– Normalisation	LayerNorm	As an argument to `TransformerEncoder` in `torch.nn`
– Attention layers	3	
– Attention heads	2	
– Attention hidden size	256	
– Encoder output size	128	
Training		
– runs	3	per benchmark

Amorpheus makes use of gradient clipping and a smaller learning rate. We found, that SMP also performs better with the decreased learning rate (0.0001) as well and we use it throughout the work. Figure 7 demonstrates the effect of a smaller learning rate on `Walker++`. All other SMP hyperparameters are as reported in the original paper with the two-directional message passing.

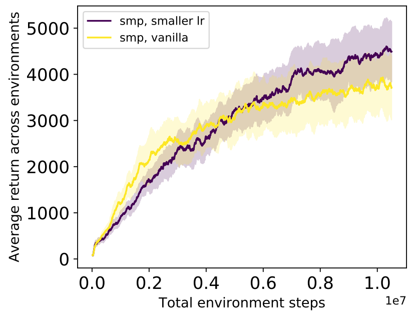


Figure 7: Smaller learning rate make SMP to yield better results on `Walker++`.

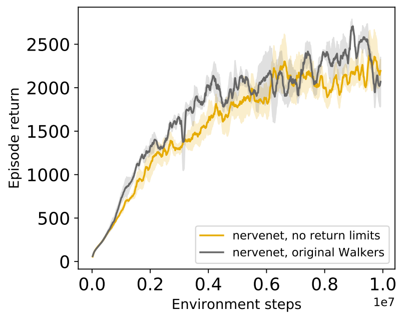


Figure 8: Removing the return limit slightly deteriorates the performance of NerveNet on Walkers.

Wang et al. (2018) add an artificial return limit of 3800 for their Walkers environment. We remove this limit and compare the methods without it. For NerveNet, we plot the results with the option best for it. Figure 8 compares the two options.

Table 2: Full list of environments used in this work.

Environment	**Training**	**Zero-shot testing**
`Walker++`		

	`walker_2_main`
	`walker_4_main`
	`walker_5_main`
	`walker_7_main`
		`walker_3_main`
		`walker_6_main`

`humanoid++`		

	`humanoid_2d_7_left_arm`
	`humanoid_2d_7_lower_arms`
	`humanoid_2d_7_right_arm`
	`humanoid_2d_7_right_leg`
	`humanoid_2d_8_left_knee`
	`humanoid_2d_9_full`
		`humanoid_2d_7_left_leg`
		`humanoid_2d_8_right_knee`

`Cheetah++`		

	`cheetah_2_back`
	`cheetah_2_front`
	`cheetah_3_back`
	`cheetah_3_front`
	`cheetah_4_allback`
	`cheetah_4_allfront`
	`cheetah_4_back`
	`cheetah_4_front`
	`cheetah_5_balanced`
	`cheetah_5_front`
	`cheetah_6_back`
	`cheetah_7_full`
		`cheetah_3_balanced`
		`cheetah_5_back`
		`cheetah_6_front`

`Cheetah-Walker-` `-Humanoid`		
	All in the column above	All in the column above
`Hopper++`		

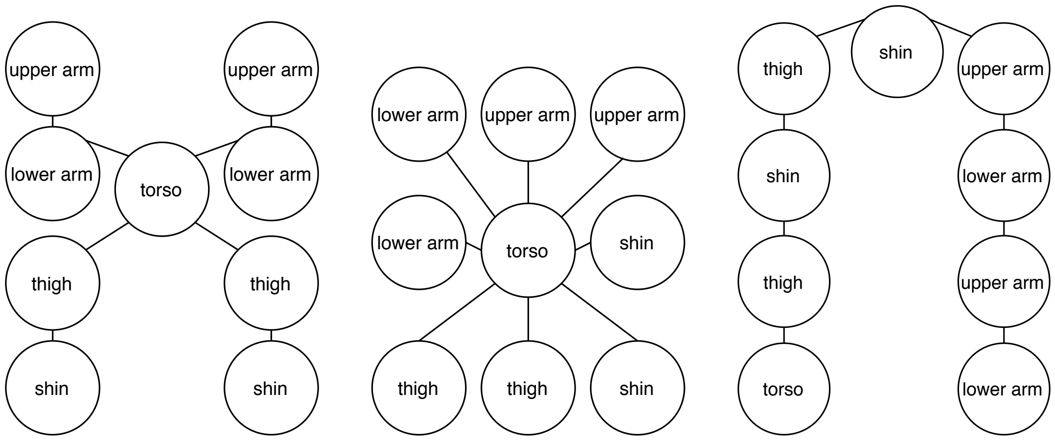
	`hopper_3`
	`hopper_4`
	`hopper_5`	

`Cheetah-Walker-` `-Humanoid-Hopper`		
	All in the column above	All in the column above
`Walkers` from Wang et al. (2018)		

	`Ostrich`
	`HalfCheetah`
	`FullCheetah`
	`Hopper`
	`HalfHumanoid`	

B MORPHOLOGY ABLATIONS

Figure 9 shows examples of graph topologies we used in structure ablation experiments.



(a) Morphology (b) Star (c) Line

Figure 9: Examples of graph topologies used in the structure ablation experiments.

C Attention Mask Analysis

C.1 Evolution of masks throughout the training process

Figures 10, 11 and 12 demonstrate the evolution of Amorpheus attention masks during training.

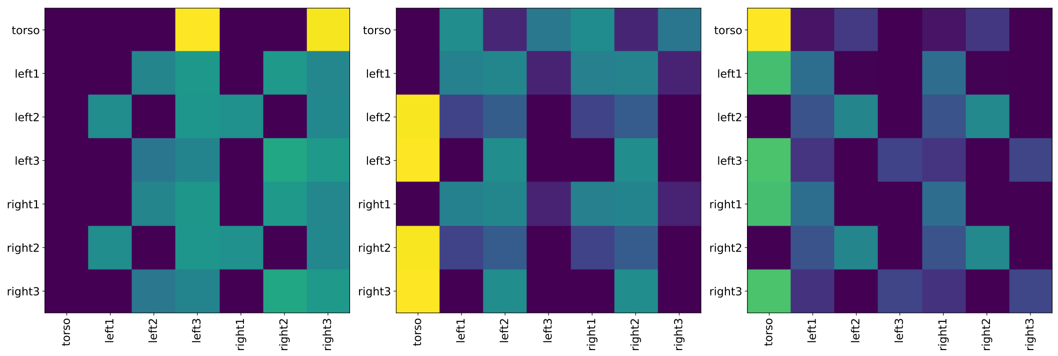


Figure 10: `Walker++` masks for the 3 attention layers on `Walker-7` at the beginning of training.

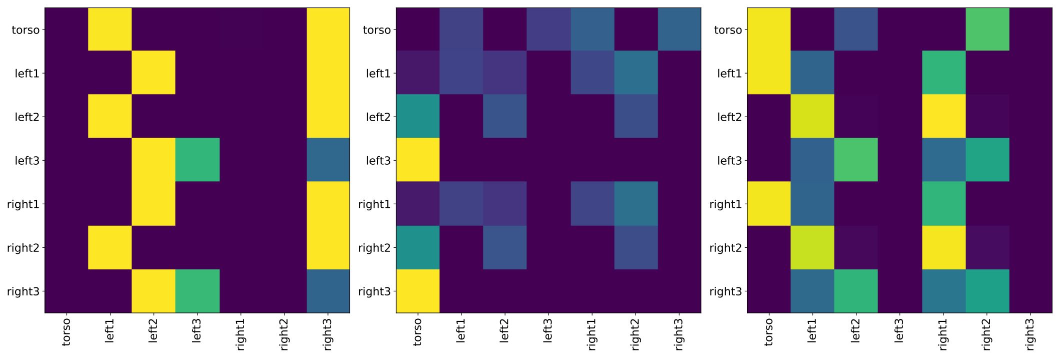


Figure 11: `Walker++` masks for the 3 attention layers on `Walker-7` after 2.5 mil frames.

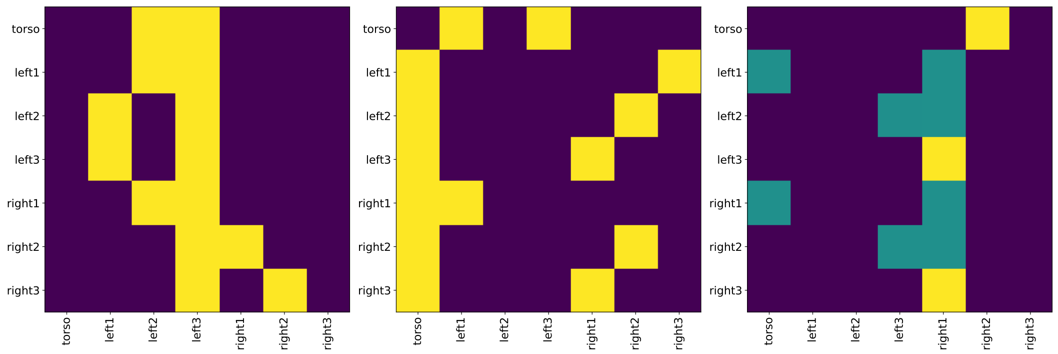


Figure 12: `Walker++` masks for the 3 attention layers on `Walker-7` at the end of training.

C.2 Attention masks cumulative change

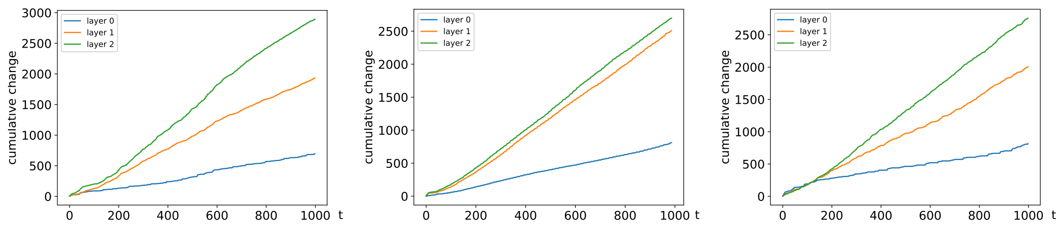


Figure 13: Absolute cumulative change in the attention masks for three different models on `Walker-7`.

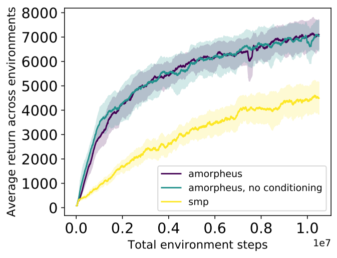
D Generalisation results

Table 3: Initial results on generalisation. The numbers show the average performance of three seeds evaluated on 100 rollouts and standard error of the mean. While the average values are higher for Amorpheus on 5 out of 7 benchmarks, high variance of both methods might be indicative of instabilities in generalisation behaviour due to large differences between the training and testing tasks.

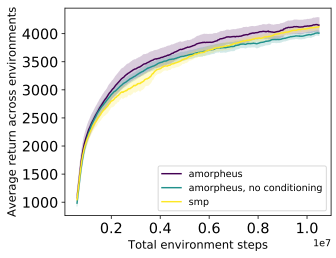
	Amorpheus	SMP
`walker-3-main`	**666.24** (133.66)	175.65 (157.38)
`walker-6-main`	**1171.35** (832.91)	729.26 (135.60)
`humanoid-2d-7-left-leg`	**2821.22** (1340.29)	2158.29 (785.33)
`humanoid-2d-8-right-knee`	**2717.21** (624.80)	327.93 (125.75)
`cheetah-3-balanced`	**474.82** (74.05)	156.16 (33.00)
`cheetah-5-back`	3417.72 (306.84)	**3820.77** (301.95)
`cheetah-6-front`	5081.71 (391.08)	**6019.07** (506.55)

E Residual Connection Ablation

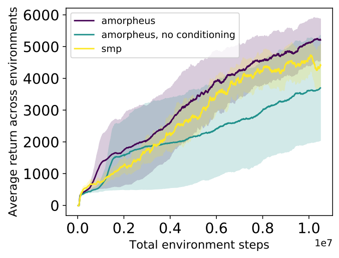
We use the residual connection in Amorpheus as a safety mechanim to prevent nodes from forgetting their own observations. To check that Amorpheus's improvements do not come from the residual connection alone, we performed the ablation. As one can see on Figure 14, we cannot attribute the success of our method to this improvement alone. High variance on `Humanoid++` is related to the fact that one seed started to improve much later, and the average performance suffered as the result.



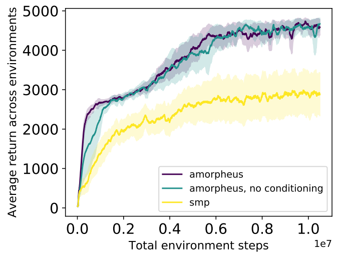
(a) Walker++



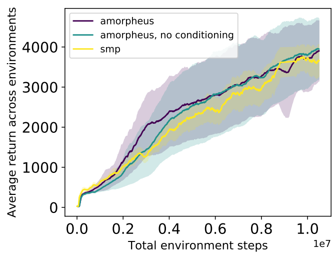
(b) Cheetah++



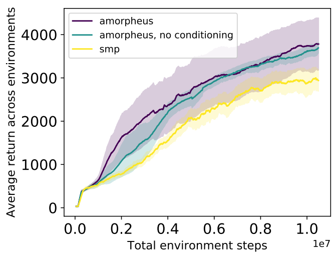
(c) Humanoid++



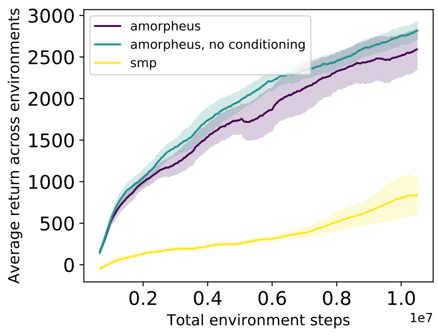
(d) Hopper++



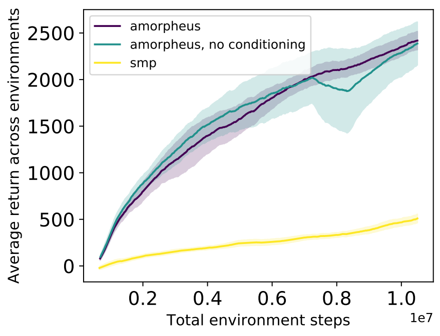
(e) Walker-Humanoid++



(f) Walker-Humanoid-Hopper++



(g) Cheetah-Walker-Humanoid++



(h) Cheetah-Walker-Humanoid-Hopper++

Figure 14: Residual connection ablation experiment.