TRANSFORMERS ARE META-REINFORCEMENT LEARNERS

Anonymous authors
Paper under double-blind review

ABSTRACT

The transformer architecture and variants presented a remarkable success across many machine learning tasks in recent years. This success is intrinsically related to the capability of handling long sequences and the presence of context-dependent weights from the attention mechanism. We argue that these capabilities suit the central role of a Meta-Reinforcement Learning algorithm. Indeed, a meta-RL agent needs to infer the task from a sequence of trajectories. Furthermore, it requires a fast adaptation strategy to adapt its policy for a new task - which can be achieved using the self-attention mechanism. In this work, we present TrMRL (**Tr**ansformers for **M**eta-**R**einforcement **L**earning), a meta-RL agent that mimics the memory reinstatement mechanism using the transformer architecture. It associates the recent past of working memories to build an episodic memory recursively through the transformer layers. This memory works as a proxy to the current task, and we condition a policy head on it. We conducted experiments in high-dimensional continuous control environments for locomotion and dexterous manipulation. Results show that TrMRL achieves or surpasses state-of-the-art performance, sample efficiency, and out-of-distribution generalization in these environments.

1 INTRODUCTION

In recent years, the Transformer architecture (Vaswani et al., 2017) achieved exceptional performance on many machine learning applications, especially for text (Devlin et al., 2019; Raffel et al., 2020) and image processing (Dosovitskiy et al., 2021b; Caron et al., 2021; Yuan et al., 2021). This intrinsically relates to its few-shot learning nature Brown et al. (2020a): the attention weights work as context-dependent parameters, inducing better generalization. Furthermore, this architecture parallelizes token processing by design. This capability avoids the vanishing gradients problem, very common for recurrent models. As a result, they can handle longer sequences more efficiently.

This work argues that these two capabilities are essential for a Meta-Reinforcement Learning (meta-RL) agent. We propose TrMRL (**Tr**ansformers for **M**eta-**R**einforcement **L**earning), a memory-based meta-Reinforcement Learner which uses the transformer architecture to formulate the learning process. It works as a memory reinstatement mechanism (Rovee-Collier, 2012) during learning, associating recent working memories to create an episodic memory which is used to contextualize the policy.

Figure 1 illustrates the process. We formulated each task as a distribution over working memories. TrMRL associates these memories using self-attention blocks to create a task representation in each head. These task representations are combined in the position-wise MLP to create an episodic output (which we identify as episodic memory). We recursively apply this procedure through layers to refine the episodic memory. In the end, we select the memory associated with the current timestep and feed it into the policy head.

Nonetheless, transformer optimization is often unstable, especially in the RL setting. Past attempts either fail to stabilize (Mishra et al., 2018) or required architectural additions (Parisotto et al., 2019) or restrictions on the observations space (Loynd et al., 2020). We argue that this challenge can be mitigated through a proper weight initialization scheme. For this matter, we applied T-Fixup initialization (Huang et al., 2020).

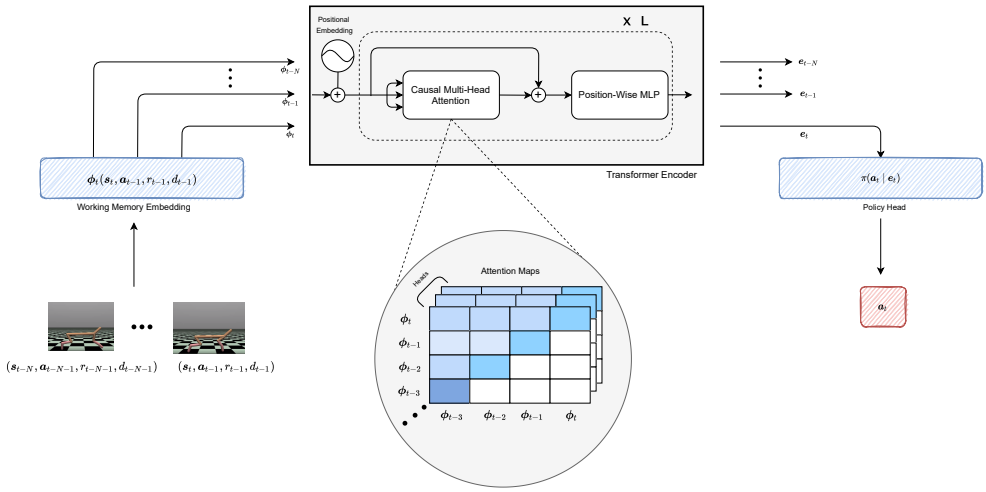

Figure 1: Illustration of the TrMRL agent. At each timestep, it associates the recent past of working memories to build an episodic memory through transformer layers recursively. We argue that the multi-head self-attention mechanism works as a fast adaptation strategy since it provides context-dependent parameters.

We conducted a series of experiments to evaluate meta-training, fast adaptation, and out-of-distribution generalization in continuous control environments for locomotion and robotic manipulation. Results show that TrMRL consistently achieves or surpasses the current state-of-the-art meta-RL agents in performance and sample efficiency. It presents online adaptation, requiring as few as 20 timesteps to identify and achieve the desired performance on test tasks. We also conducted an ablation study to show the effectiveness of the T-Fixup initialization, and the sensibility to network depth, sequence size, and the number of attention heads.

2 Related Work

Meta-Learning is an established Machine Learning (ML) principle to learn inductive biases from the distribution of tasks to produce a data-efficient learning system (Bengio et al., 1991; Schmidhuber et al., 1996; Thrun & Pratt, 1998). This principle spanned in a variety of methods in recent years, learning different components of an ML system, such as the optimizer (Andrychowicz et al., 2016; Li & Malik, 2016; Chen et al., 2017), neural architectures (Hutter et al., 2019; Zoph & Le, 2017), metric spaces (Vinyals et al., 2016), weight initializations (Finn et al., 2017; Nichol et al., 2018; Finn et al., 2018), and conditional distributions (Zintgraf et al., 2019; Melo et al., 2019). Another branch of methods learns the entire system using memory-based architectures (Ortega et al., 2019; Wang et al., 2017; Duan et al., 2016; Ritter et al., 2018a) or generating update rules by discovery (Oh et al., 2020) or evolution (Co-Reyes et al., 2021).

Memory-Based Meta-Learning is the particular class of methods where we focus on in this work. In this context, Wang et al. (2017); Duan et al. (2016) concurrently proposed the RL^2 framework, which formulates the learning process as a Recurrent Neural Network (RNN) where the hidden state works as the memory mechanism. Given the recent rise of attention-based architectures, one natural idea is to use it as a replacement for RNNs. Mishra et al. (2018) proposed an architecture composed of causal convolutions (to aggregate information from past experience) and soft-attention (to pinpoint specific pieces of information). In contrast, our work applies causal, multi-head self-attention by stabilizing the complete transformer architecture with an arbitrarily large context window. Finally, Ritter et al. (2021) also applied multi-head self-attention for rapid task solving for RL environments. However, in a different dynamic: their work applied the attention mechanism iteratively in a pre-defined episodic memory, while ours applies it recursively through transformer layers to build an episodic memory from the association of recent working memories.

Our work has intersections with Cognitive Neuroscience research on memory for learning systems (Hoskin et al., 2018; Rovee-Collier, 2012; Wang et al., 2018). In this context, Ritter et al. (2018c)

extended the RL^2 framework incorporating a differentiable neural dictionary as the inductive bias for episodic memory recall. In the same line, Ritter et al. (2018b) also extended RL^2 but integrating a different episodic memory system inspired by the reinstatement mechanism. In our work, we also mimic reinstatement to retrieve episodic memories from working memories but using self-attention. Lastly, Fortunato et al. (2019) studied the association between working and episodic memories for RL agents, specifically for memory tasks, proposing separated inductive biases for these memories based on LSTMs and auxiliary unsupervised losses. In contrast, our work studies this association for the Meta-RL problem, using memory as a task proxy implemented by the transformer architecture.

Meta-Reinforcement Learning is a branch of Meta-Learning for RL agents. Some of the algorithms described in past paragraphs extend to the Meta-RL setting by design (Finn et al., 2017; Mishra et al., 2018; Wang et al., 2017; Duan et al., 2016). Others were explicitly designed for RL and often aimed to create a task representation to condition the policy. PEARL (Rakelly et al., 2019) is an off-policy method that learns a latent representation of the task and explores via posterior sampling. MAESN (Gupta et al., 2018) also creates task variables but optimizes them with on-policy gradient descent and explores by sampling from the prior. MQL (Fakoor et al., 2020) is also an off-policy method, but it uses a deterministic context that is not permutation invariant implemented by an RNN. Lastly, VariBAD Zintgraf et al. (2020) formulates the problem as a Bayes-Adaptive MDP and extends the RL^2 framework by incorporating a stochastic latent representation of the task trained with a VAE objective. Our work contrasts all the previous methods in this task representation: we condition the policy in the episodic memory generated by the transformer architecture from the association of past working memories. We show that this episodic memory works as a *proxy* to the task representation.

Transformers for RL. The application of the transformer architecture in the RL setting is still an open challenge. Mishra et al. (2018) tried to apply this architecture for simple bandit tasks and tabular MDPs and reported unstable train and random performance. Parisotto et al. (2019) then proposed some architectural changes in the vanilla transformer, reordering layer normalization modules and replacing residual connections with expressing gating mechanisms, improving state-of-the-art performance for a set of memory environments. Loynd et al. (2020) also studied how transformer-based models can improve the performance of sequential decision-making agents. It stabilized the architecture using factored observations and an intense hyperparameter tuning procedure, resulting in improved sample efficiency. In contrast to these methods, our work stabilizes the transformer model by improving optimization through a better weight initialization. In this way, we could use the vanilla transformer without architectural additions or imposing restrictions on the observations.

Finally, recent work studied how to replace RL algorithms with transformer-based language models (Janner et al., 2021; Chen et al., 2021). Using a supervised prediction loss in the offline RL setting, they modeled the agent as a sequence problem. Our work, on the other hand, considers the standard RL formulation in the meta-RL setting.

3 PRELIMINARIES

We define a Markov decision process (MDP) by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{P}_0, \gamma, H)$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ is an action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, \infty)$ is a transition dynamics, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to [-R_{max}, R_{max}]$ is a bounded reward function, $\mathcal{P}_0 : \mathcal{S} \to [0, \infty)$ is an initial state distribution, $\gamma \in [0, 1]$ is a discount factor, and H is the horizon. The standard RL objective is to maximize the cumulative reward, i.e., $\max \mathbb{E}[\sum_{t=0}^{T} \gamma^t \mathcal{R}(s_t, a_t)]$, with $a_t \sim \pi_{\boldsymbol{\theta}}(a_t \mid s_t)$ and $s_t \sim \mathcal{P}(s_t \mid s_{t-1}, a_{t-1})$, where $\pi_{\boldsymbol{\theta}} : \mathcal{S} \times \mathcal{A} \to [0, \infty)$ is a policy parameterized by $\boldsymbol{\theta}$.

3.1 PROBLEM SETUP: META-REINFORCEMENT LEARNING

In the meta-RL setting, we define $p(\mathcal{M}) : \mathcal{M} \to [0, \infty)$ a distribution over a set of MDPs $\mathcal{M}$. During meta-training, we sample $\mathcal{M}_i \sim p(\mathcal{M})$ from this distribution, where $\mathcal{M}_i = (\mathcal{S}, \mathcal{A}, \mathcal{P}_i, \mathcal{R}_i, \mathcal{P}_{0,i}, \gamma, H)$. Therefore, the tasks[1] share a similar structure in this setting, but reward function and transition dynamics vary. The goal is to learn a policy that, during meta-testing, can adapt to a new task sampled from the same distribution $p(\mathcal{M})$. In this context, adaptation means

[1] We use the terms task and MDP interchangeably.

maximizing the reward under the task in the most efficient way. To achieve this, the meta-RL agent should learn the prior knowledge shared across the distribution of tasks. Simultaneously, it should learn how to differentiate and identify these tasks using only a few episodes.

3.2 Transformer Architecture

The transformer architecture (Vaswani et al., 2017) was first proposed as an encoder-decoder architecture for neural machine translation. Since then, many variants have emerged, proposing simplifications or architectural changes across many ML problems (Dosovitskiy et al., 2021a; Brown et al., 2020b; Parisotto et al., 2019). Here, we describe the encoder architecture as it composes our memory-based meta-learner.

The transformer encoder is a stack of multiple equivalent layers. There are two main components in each layer: a multi-head self-attention block, followed by a position-wise feed-forward network. Each component contains a residual connection (He et al., 2015) around them, followed by layer normalization (Ba et al., 2016). The multi-head self-attention (MHSA) block computes the self-attention operation across many different heads, whose outputs are concatenated to serve as input to a linear projection module, as in Equation 1:

$$\begin{aligned} \text{MHSA}(K, Q, V) &= \text{Concat}(h_1, h_2, \ldots, h_\omega) W_o, \\ h_i &= \text{softmax}(\frac{QK^T}{\sqrt{d}} \cdot M)V, \end{aligned} \tag{1}$$

where K, Q, V are the keys, queries, and values for the sequence input, respectively. Additionally, d represents the dimension size of keys and queries representation and ω the number of attention heads. M represents the attention masking operation. W_o represents a linear projection operation.

The position-wise feed-forward block is a 2-layer dense network with a ReLU activation between these layers. All positions in the sequence input share the parameters of this network, equivalently to a 1×1 temporal convolution over every step in the sequence. Finally, we describe the positional encoding. It injects the relative position information among the elements in the sequence input since the transformer architecture fully parallelizes the input processing. The standard positional encoding is a sinusoidal function added to the sequence input (Vaswani et al., 2017).

3.3 T-Fixup Initialization

The training of transformer models is notoriously difficult, especially in the RL setting (Parisotto et al., 2019). Indeed, gradient optimization with attention layers often requires complex learning rate warmup schedules to prevent divergence (Huang et al., 2020). Recent work suggests two main reasons for this requirement. First, the Adam optimizer (Kingma & Ba, 2017) presents high variance in the inverse second moment for initial updates, proportional to a divergent integral (Liu et al., 2020). It leads to problematic updates and significantly affects optimization. Second, the backpropagation through layer normalization can also destabilize optimization because the associated error depends on the magnitude of the input (Xiong et al., 2020).

Given these challenges, Huang et al. (2020) proposed a weight initialization scheme (T-Fixup) to eliminate the need for learning rate warmup and layer normalization. This is particularly important to the RL setting once current RL algorithms are very sensitive to the learning rate for learning and exploration.

T-Fixup appropriately bounds the original Adam update to make variance finite and reduce instability, regardless of model depth. We refer to Huang et al. (2020) for the mathematical derivation. We apply the T-Fixup for the transformer encoder as follows:

- Apply Xavier initialization (Glorot & Bengio, 2010) for all parameters excluding input embeddings. Use Gaussian initialization $\mathcal{N}(0, d^{-\frac{1}{2}})$, for input embeddings, where d is the embedding dimension;
- Scale the linear projection matrices in each encoder attention block and position-wise feed-forward block by $0.67N^{-\frac{1}{4}}$.

4 TRANSFORMERS ARE META-REINFORCEMENT LEARNERS

In this work, we argue that two critical capabilities of transformers compose the central role of a Meta-Reinforcement Learner. First, transformers can handle long sequences and reason over long-term dependencies, which is essential to the meta-RL agent to identify the MDP from a sequence of trajectories. Second, transformers present context-dependent weights from self-attention. This mechanism serves as a fast adaptation strategy and provides necessary adaptability to the meta-RL agent for new tasks.

4.1 TASK REPRESENTATION

We represent a working memory at the timestep t as a parameterized function $\phi_t(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \eta_t)$, where $\boldsymbol{s}_t$ is the MDP state, $\boldsymbol{a}_t \sim \pi(\boldsymbol{a}_t \mid \boldsymbol{s}_t)$ is an action, $r_t \sim \mathcal{R}(\boldsymbol{s}_t, \boldsymbol{a}_t)$ is the reward, and η_t is a boolean flag to identify whether this is a terminal state. Our first hypothesis is that we can define a task $\mathcal{T}$ as a distribution over working memories, as in Equation 2:

$$\mathcal{T}(\phi) : \Phi \to [0, \infty), \tag{2}$$

where Φ is the working memory embedding space. In this context, one goal of a meta-RL agent is to learn ϕ to make a distinction among the tasks in the embedding space Φ. Furthermore, the learned embedding space should also approximate the distributions of similar tasks so that they can share knowledge. Figure 2 illustrates this concept for a one-dimensional representation.

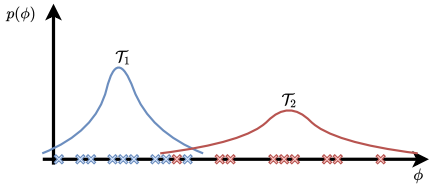

Figure 2: The illustration of two tasks ($\mathcal{T}_1$ and $\mathcal{T}_2$) as distributions over working memories. The intersection of both densities represents the ambiguity between $\mathcal{T}_1$ and $\mathcal{T}_2$.

We aim to find a representation for the task given its distribution to contextualize our policy. Intuitively, we can represent each task as a linear combination of working memories sampled by the policy interacting with it:

$$\mu_{\mathcal{T}} = \sum_{t=0}^{N} \alpha_t \cdot \mathcal{W}(\phi_t(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \eta_t)),$$
$$\text{with} \sum_{t=0}^{N} \alpha_t = 1 \tag{3}$$

where N represents the length of a segment of sampled trajectories during the policy and task interaction. $\mathcal{W}$ represents an arbitrary linear transformation. Furthermore, α_t is a coefficient to compute how relevant a particular working memory t is to the task representation, given the set of sampled working memories. Next, we show how the self-attention computes these coefficients, which we use to output an episodic memory from the transformer architecture.

4.2 SELF-ATTENTION AS A FAST ADAPTATION STRATEGY

In this work, our central argument is that self-attention works as a fast adaptation strategy. The context-dependent weights dynamically compute the working memories coefficients to implement Equation 3. We now derive *how* we compute these coefficients. Figure 3 illustrates this mechanism.

Let us define ϕ_t^k, ϕ_t^q, and ϕ_t^v as a representation[2] of the working memory at timestep t in the keys, queries, and values spaces, respectively. The dimension of the queries and keys spaces is d. We aim to compute the attention operation in Equation 1 for a sequence of T timesteps, resulting in Equation 4:

[2]we slightly abuse the notation by omitting the function arguments – $\boldsymbol{s}_t$, $\boldsymbol{a}_t$, r_t, and η_t – for the sake of conciseness.

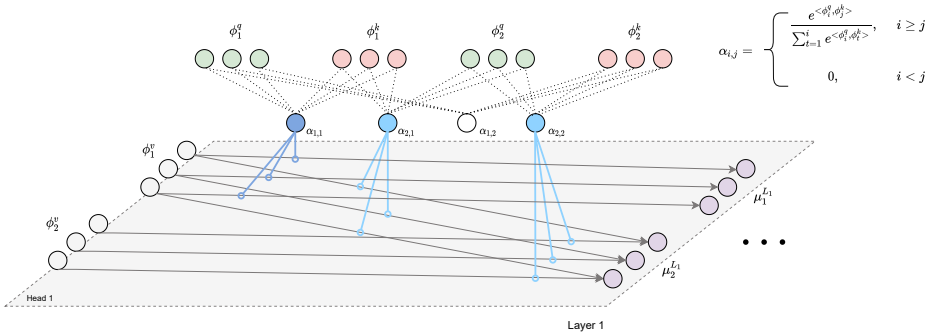

Figure 3: Illustration of causal self-attention as a fast adaptation strategy. In this simplified scenario (2 working memories), we observe that the attention weights $\alpha_{i,j}$ drives the association between the current working memory and the past ones to compute a task representation μ_t. Self-attention computes this association by relative similarity.

$$\text{softmax}(\frac{QK^T}{\sqrt{d}} \cdot M)V = \frac{1}{\sqrt{d}} \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \cdots \\ \vdots & \ddots & \\ \alpha_{T,1} & & \alpha_{T,T} \end{bmatrix} \begin{bmatrix} \phi_1^v \\ \vdots \\ \phi_T^v \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_T \end{bmatrix},$$
$$\text{where} \begin{cases} \alpha_{i,j} = \frac{\exp \langle \phi_i^q, \phi_j^k \rangle}{\sum_{n=1}^{i} \exp \langle \phi_1^q, \phi_n^k \rangle} & \text{if } i \leq j \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

where $\langle a_i, b_j \rangle = \sum_{n=0}^{d} a_{i,n} \cdot b_{j,n}$ is the dot product between the working memories a_i and b_j. Therefore, for a particular timestep t, the self-attention output is:

$$\begin{aligned} \mu_t &= \frac{1}{\sqrt{d}} \cdot \frac{\phi_1^v \cdot \exp \langle \phi_t^q, \phi_1^k \rangle + \cdots + \phi_t^v \cdot \exp \langle \phi_t^q, \phi_t^k \rangle}{\sum_{n=1}^{i} \exp \langle \phi_1^q, \phi_n^k \rangle} \\ &= \frac{1}{\sqrt{d}} \sum_{n=0}^{t} \alpha_{t,n} W_v(\phi_t). \end{aligned} \tag{5}$$

Equation 5 shows that the self-attention mechanism implements the task representation in Equation 3 by associating past working memories *given that* the current one is ϕ_t. It computes this association with *relative similarity* through the dot product normalized by the softmax operation. This inductive bias helps the working memory representation learning to approximate the density of the task distribution $\mathcal{T}(\phi)$.

4.3 Transformers and Memory Reinstatement

We presented that the self-attention mechanism works as a fast adaptation strategy by retrieving a task representation from the past working memories. We now highlight two main concerns of solely using a single self-attention block to build the meta-RL agent. First, A single head might not be expressive enough to provide a good task representation. Second, The self-attention coefficients are biased towards the current working memory. Indeed, we compute the coefficients in Equation 5 in the perspective of ϕ_t. If ϕ_t is not representative of the task, the final task representation will also fail to represent the task distribution $\mathcal{T}$.

We argue that the transformer architecture mitigates both concerns. For the first one, it implements self-attention in multi-headed blocks. Multiple heads diversify the keys, queries, and values spaces, creating different projections of working memories and, therefore, improving the expressivity of the network. Indeed, each head will create a new task representation, and they will be combined in the position-wise feed-forward network later on.

For the second concern, we consider that a transformer layer outputs an episodic memory as a function of the task representations for each timestep in the trajectory:

$$e_t = f(\mu_t^0, \cdots, \mu_t^\omega), \tag{6}$$

where f represents the position-wise feed-forward transformation and μ_t^h describes the task representation for the timestep t computed using the attention head h. In this way, the transformer architecture recursively refines the episodic memory interacting output memories from the past layer. As we compute each output memory in the perspective of a different timestep t, deeper layers reach a consensus across the whole sub-trajectory. As future work, we aim to investigate how the trajectories of sampled working memories approximate a posterior distribution and how this consensus mechanism relates to minimum Bayes risk decoding (Kumar & Byrne, 2004).

As summarization, the transformer associates a sub-trajectory of working memories to retrieve an episodic memory which proxies the task representation computed by MHSA blocks. As the last step, we condition the policy head in the episodic memory from the current timestep to sample the appropriated action. This complete process resembles a memory reinstatement operation: a reminder procedure that reintroduces past elements in advance of a long-term retention test (Rovee-Collier, 2012). In our context, this "long-term retention test" identifies the task and acts accordingly to maximize rewards.

5 Experiments and Analysis

In this section, we present an empirical validation of our method, comparing it with the current state-of-the-art methods. We considered high-dimensional, continuous control tasks for locomotion (MuJoCo[3]) and dexterous manipulation (MetaWorld). We describe them in Appendix A. For reproducibility (source code and hyperparameters), we refer to the released source code[4].

5.1 Experimental Setup

Meta-Training. During meta-training, we repeatedly sampled a batch of tasks to collect experience with the goal of learning to learn. For each task, we ran a sequence of N episodes. During the interaction, the agent conducted exploration with a gaussian policy. During optimization, we concatenate these episodes to form a single trajectory and we maximize the discounted cumulative reward of this trajectory. This is equivalent to the training setup for other on-policy meta-RL algorithms (Duan et al., 2016; Zintgraf et al., 2020). For these experiments, we considered $N = 2$. We performed this training via Proximal Policy Optimization (PPO) (Schulman et al., 2017), and the data batches mixed different tasks. Therefore, we present here an on-policy version of the TrMRL algorithm. To stabilize transformer training, we used the T-Fixup as a weight initialization scheme.

Meta-Testing. During meta-testing, we sampled new tasks. These are different from the tasks in meta-training, but they come from the same distribution, except during Out-of-Distribution (OOD) evaluation. For TrMRL, in this stage, we froze all network parameters. For each task, we ran few episodes, performing the adaptation strategy. The goal is to identify the current MDP and maximize the cumulative reward across the episodes.

Memory Write Logic. At each timestep, we fed the network with the sequence of working memories. This process works as follows: at the beginning of an episode (when the memory sequence is empty), we start writing the first positions of the sequence until we fill all the slots. Then, for each new memory, we removed the oldest memory in the sequence (in the "back" of this "queue") and added the most recent one (in the "front").

Comparison Methods. For comparison, we evaluated three different state-of-the-art meta-RL algorithms: RL^2 Duan et al. (2016), optimized using PPO Schulman et al. (2017); PEARL (Rakelly et al., 2019); and MAML (Finn et al., 2017), whose outer-loop used TRPO (Schulman et al., 2015).

[3]We highlight that both MuJoCo (Locomotion Tasks) and MetaWorld are built on the MuJoCo physics engine. We identify the set of locomotion tasks as solely for "MuJoCo" to ensure simpler and concise writing during analysis of the results.

[4]Omitted due to double-blind review. See supplementary material.

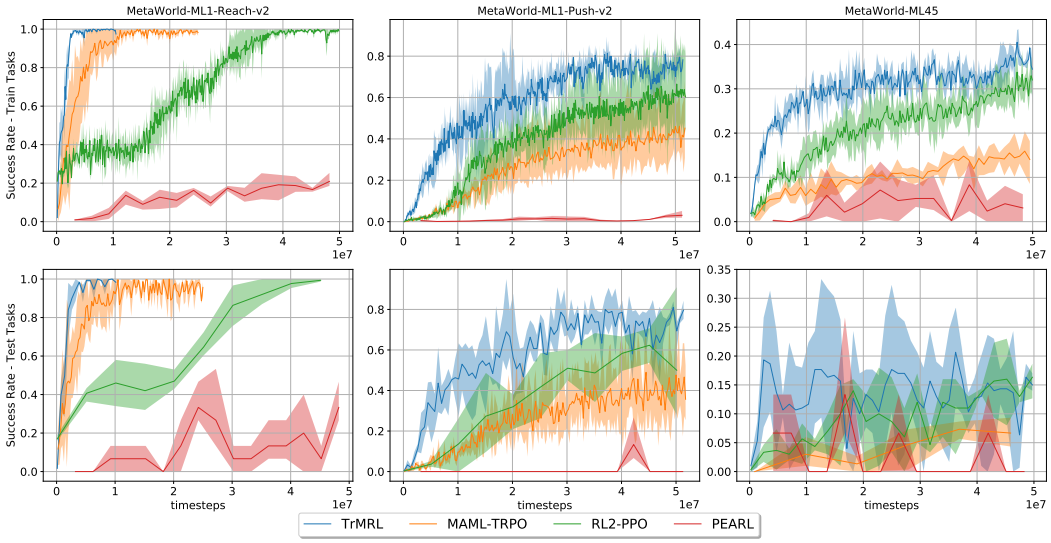

Figure 4: Meta-Training results for MetaWorld benchmarks. The plots on top represents performance on training tasks, while the plots on bottom represents in the test tasks.

5.2 Results and Analysis

We compared TrMRL with baseline methods in terms of meta-training, episode adaptation, and OOD performance. We also present the latent visualization for TrMRL working memories and ablation studies. All the curves presented are averaged across three random seeds, with 95% bootstrapped confidence intervals.

Meta-Training Evaluation. Figure 4 shows the meta-training results for all the methods in the MetaWorld environments. All subplots show the task success rate over the training timesteps. The plots on top represent performance on training tasks, while the plots on the bottom represent on the test tasks. TrMRL outperformed all baseline methods. In the "Reach-v2", TrMRL, RL^2, and MAML reached the perfect success rate, but TrMRL was more sample efficient. For more complex scenarios, such as "Push-v2" and ML45, TrMRL still performs consistently better. Nevertheless, we also present that all the presented methods performed poorly on ML45 for test tasks, highlighting a big improvement room. We hypothesize that this is due to the lack of a meta-exploration strategy and an inductive bias to improve knowledge share among different tasks. For MuJoCo locomotion tasks, we refer to Appendix B.

Fast Adaptation Evaluation. Another important skill for meta-RL agents is the capability of adapting to the tasks given a few episodes. We evaluate this by running the meta-test procedure on 20 test tasks over 6 sequential episodes. Each agent will run its adaptation strategy to identify the task and maximize the reward across episodes. Figure 6 presents the results for the locomotion tasks. For AntDir and HalfCheetaVel, TrMRL outperformed all methods. For HalfCheetahDir, TrMRL started with better performance, but PEARL outperformed after running its adaptation mechanism.

We highlight that TrMRL presented high performance since the first episode. In fact, it only requires a few timesteps to achieve high performance in test tasks. In the HalfCheetahVel, for example, it only requires around 20 timesteps to achieve the best performance (Figure 5). Therefore, it presents a nice

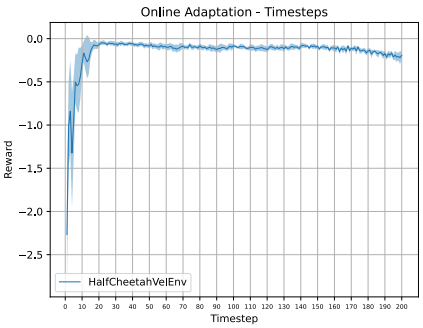

Figure 5: TrMRL's adaptation for HalfCheetahVel environment.

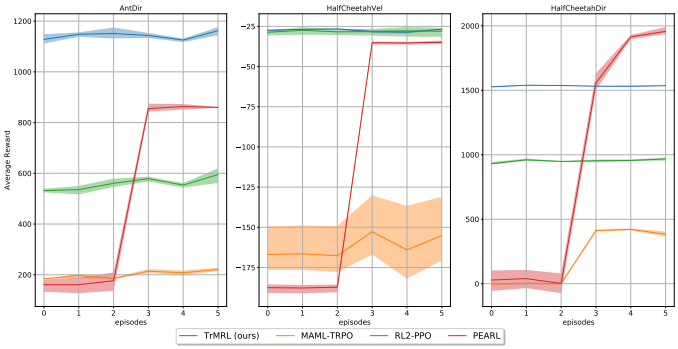

Figure 6: Fast adaptation results on MuJoCo locomotion tasks. Each curve represents the average performance over 20 test tasks. TrMRL presented high performance since the first episode due to the online adaptation nature from attention weights.

property for online adaptation. This is because the self-attention mechanism is lightweight and only requires a few working memories to achieve good performance. Hence, we can run it efficiently at each timestep. Other methods, such as PEARL and MAML, do not present such property, and they need a few episodes before executing adaptation efficiently.

OOD Evaluation. Another critical scenario is how the fast adaptation strategies perform for out-of-distribution tasks. For this case, we change the HalfCheetahVel environment to sample OOD tasks during the meta-test. In the standard setting, both training and testing target velocities are sampled from a uniform distribution in the interval $[0.0, 3.0]$. In the OOD setting, we sampled 20 tasks in the interval $[3.0, 4.0]$ and assessed adaptation throughout the episodes. Figure 7 presents the results. TrMRL surpasses all the baselines methods with a good margin, suggesting that the context-dependent weights learned a robust adaptation strategy, while other methods memorized some aspects of the standard distribution of tasks. We especially highlight PEARL, which achieved the best performance among the methods but performed poorly in this setting, suggesting that it does not generate useful latent representations for OOD tasks.

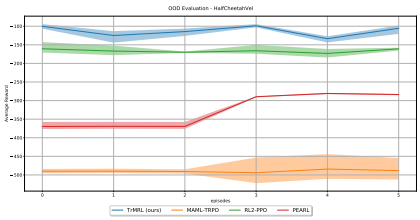

Figure 7: OOD Evaluation in HalfCheetahVel environment.

6 Conclusion and Future Work

In this work, we presented TrMRL, a memory-based meta-RL algorithm built upon a transformer, where the multi-head self-attention mechanism works as a fast adaptation strategy. We designed this network to resemble a memory reinstatement mechanism, associating past working memories to dynamically represent a task and recursively build an episode memory through layers.

As future work, we plan to work on the current limitations of the proposed method. First, TrMRL does not present any mechanism for meta-exploration and relies solely on the gaussian policy exploration; we hypothesize that meta-exploration plays a key role for more challenging environments, such as the MetaWorld ML45. Second, we plan to deepen the theoretical understanding of how the working memory sequence approximates the true posterior distribution and how the proposed reinstatement mechanism relates to the Bayesian framework.

References

Marcin Andrychowicz, Misha Denil, Sergio Gómez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Gar-

nett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/fb87582825f9d28a8d42c5e5e5e8b23d-Paper.pdf.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

Y. Bengio, S. Bengio, and J. Cloutier. Learning a synaptic learning rule. In *IJCNN-91-Seattle International Joint Conference on Neural Networks*, volume ii, pp. 969 vol.2–, 1991. doi: 10.1109/IJCNN.1991.155621.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020a.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020b. URL https://arxiv.org/abs/2005.14165.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers, 2021.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling, 2021.

Yutian Chen, Matthew W. Hoffman, Sergio Gómez Colmenarejo, Misha Denil, Timothy P. Lillicrap, Matt Botvinick, and Nando de Freitas. Learning to learn without gradient descent by gradient descent. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 748–756. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/chen17e.html.

John D. Co-Reyes, Yingjie Miao, Daiyi Peng, Esteban Real, Sergey Levine, Quoc V. Le, Honglak Lee, and Aleksandra Faust. Evolving reinforcement learning algorithms, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=YicbFdNTTy.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021b.

Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning, 2016.

Rasool Fakoor, Pratik Chaudhari, Stefano Soatto, and Alexander J. Smola. Meta-q-learning, 2020.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/finn17a.html.

Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 9537–9548, Red Hook, NY, USA, 2018. Curran Associates Inc.

Meire Fortunato, Melissa Tan, Ryan Faulkner, Steven Hansen, Adrià Puigdomènech Badia, Gavin Buttimore, Charles Deck, Joel Z Leibo, and Charles Blundell. Generalization of reinforcement learners with working and episodic memory. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/02ed812220b0705fabb868ddbf17ea20-Paper.pdf.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics*, 2010.

Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 5307–5316, Red Hook, NY, USA, 2018. Curran Associates Inc.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

Abigail N. Hoskin, Aaron M. Bornstein, Kenneth A. Norman, and Jonathan D. Cohen. Refresh my memory: Episodic memory reinstatements intrude on working memory maintenance. *bioRxiv*, 2018. doi: 10.1101/170720. URL https://www.biorxiv.org/content/early/2018/05/30/170720.

Xiao Shi Huang, Felipe Perez, Jimmy Ba, and Maksims Volkovs. Improving transformer optimization through better initialization. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 4475–4483. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/huang20f.html.

Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated Machine Learning: Methods, Systems, Challenges*. Springer Publishing Company, Incorporated, 1st edition, 2019. ISBN 3030053172.

Michael Janner, Qiyang Li, and Sergey Levine. Reinforcement learning as one big sequence modeling problem, 2021.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

Shankar Kumar and William Byrne. Minimum Bayes-risk decoding for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pp. 169–176, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics. URL https://aclanthology.org/N04-1022.

Ke Li and Jitendra Malik. Learning to optimize, 2016.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond, 2020.

Ricky Loynd, Roland Fernandez, Asli Celikyilmaz, Adith Swaminathan, and Matthew Hausknecht. Working memory graphs, 2020.

Luckeciano C. Melo, Marcos R. O. A. Maximo, and Adilson Marques da Cunha. Bottom-up meta-policy search, 2019.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner, 2018.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms, 2018.

Junhyuk Oh, Matteo Hessel, Wojciech M. Czarnecki, Zhongwen Xu, Hado P van Hasselt, Satinder Singh, and David Silver. Discovering reinforcement learning algorithms. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1060–1070. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/0b96d81f0494fde5428c7aea243c9157-Paper.pdf.

Pedro A. Ortega, Jane X. Wang, Mark Rowland, Tim Genewein, Zeb Kurth-Nelson, Razvan Pascanu, Nicolas Heess, Joel Veness, Alex Pritzel, Pablo Sprechmann, Siddhant M. Jayakumar, Tom McGrath, Kevin Miller, Mohammad Azar, Ian Osband, Neil Rabinowitz, András György, Silvia Chiappa, Simon Osindero, Yee Whye Teh, Hado van Hasselt, Nando de Freitas, Matthew Botvinick, and Shane Legg. Meta-learning of sequential strategies, 2019.

Emilio Parisotto, H. Francis Song, Jack W. Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant M. Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, Matthew M. Botvinick, Nicolas Heess, and Raia Hadsell. Stabilizing transformers for reinforcement learning, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2020.

Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables, 2019.

Sam Ritter, Ryan Faulkner, Laurent Sartran, Adam Santoro, Matt Botvinick, and David Raposo. Rapid task-solving in novel environments, 2021.

Samuel Ritter, Jane Wang, Zeb Kurth-Nelson, Siddhant Jayakumar, Charles Blundell, Razvan Pascanu, and Matthew Botvinick. Been there, done that: Meta-learning with episodic recall. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4354–4363. PMLR, 10–15 Jul 2018a. URL https://proceedings.mlr.press/v80/ritter18a.html.

Samuel Ritter, Jane X. Wang, Zeb Kurth-Nelson, and Matthew Botvinick. Episodic control through meta-reinforcement learning. In *CogSci*, 2018b. URL https://mindmodeling.org/cogsci2018/papers/0190/index.html.

Samuel Ritter, Jane X. Wang, Zeb Kurth-Nelson, Siddhant M. Jayakumar, Charles Blundell, Razvan Pascanu, and Matthew Botvinick. Been there, done that: Meta-learning with episodic recall, 2018c.

Jonas Rothfuss, Dennis Lee, Ignasi Clavera, Tamim Asfour, and Pieter Abbeel. Promp: Proximal meta-policy search, 2018.

Carolyn Rovee-Collier. *Reinstatement of Learning*, pp. 2803–2805. Springer US, Boston, MA, 2012. ISBN 978-1-4419-1428-6. doi: 10.1007/978-1-4419-1428-6_346. URL https://doi.org/10.1007/978-1-4419-1428-6_346.

Juergen Schmidhuber, Jieyu Zhao, and Marco Wiering. Simple principles of metalearning. Technical report, 1996.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1889–1897, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/schulman15.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

Sebastian Thrun and Lorien Pratt. *Learning to Learn: Introduction and Overview*, pp. 3–17. Kluwer Academic Publishers, USA, 1998. ISBN 0792380479.

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*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/90e1357833654983612fb05e3ec9148c-Paper.pdf.

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn, 2017.

Jane X. Wang, Zeb Kurth-Nelson, Dharshan Kumaran, Dhruva Tirumala, Hubert Soyer, Joel Z. Leibo, Demis Hassabis, and Matthew Botvinick. Prefrontal cortex as a meta-reinforcement learning system. *bioRxiv*, 2018. doi: 10.1101/295964. URL https://www.biorxiv.org/content/early/2018/04/13/295964.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture, 2020.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Avnish Narayan, Hayden Shively, Adithya Bellathur, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning, 2021.

Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zihang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet, 2021.

Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning, 2020.

Luisa M Zintgraf, Kyriacos Shiarlis, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning, 2019.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. 2017. URL https://arxiv.org/abs/1611.01578.

A METAL-RL ENVIRONMENTS DESCRIPTION

In this Appendix, we detail the environments considered in this work.

A.1 MUJOCO – LOCOMOTION TASKS

This benchmark is a set of locomotion tasks on the MuJoCo (Todorov et al., 2012) environment. It comprises different bodies, and each environment provides different tasks with different learning goals. These locomotion tasks are previously introduced by Finn et al. (2017) and Rothfuss et al. (2018). We considered 3 different environments.

- **AntDir**: This environment has an ant body, and the goal is to move forward or backward. Hence, it presents these 2 tasks.
- **HalfCheetahDir**: This environment has a half cheetah body, and the goal is to move forward or backward. Hence, it presents these 2 tasks.
- **HalfCheetalVel**: This environment also has a half cheetah body, and the goal is to achieve a target velocity running forward. This target velocity comes from a continuous uniform distribution.

These locomotion task families require adaptation across reward functions.

A.2 METAWORLD

The MetaWorld (Yu et al., 2021) benchmark contains a diverse set of manipulation tasks designed for multi-task RL and meta-RL settings. MetaWorld presents a variety of evaluation modes. Here, we describe the two modes used in this work. For more detailed description of the benchmark, we refer to Yu et al. (2021).

- **ML1**: This scenario considers a single robotic manipulation task but varies the goal. The meta-training "tasks" corresponds to 50 random initial object and goal positions, and meta-testing on 50 heldout positions.
- **ML45**: With the objective of testing generalization to new manipulation tasks, the benchmark provides 45 training tasks and holds out 5 meta-testing tasks.

These robotic manipulation task families require adaptation across reward functions and dynamics.

B MuJoCo: Meta-Training Evaluation

In this Appendix, we supplement the meta-training evaluation with the results on MuJoCo locomotion tasks. Figure 8 shows the average return over train tasks (on top) and test tasks (on bottom) for AntDir, HalfCheetahVel, and HalfCheetahDir, respectively. While PEARL failed to explore and learn in robotic manipulation tasks, it presented better results on locomotion tasks, especially in sample efficiency. This is because of its off-policy nature: it efficiently reuses trajectories sampled from previous policy versions, reducing the number of training steps needed. TrMRL achieved the same test task performance in AntDir and HalfCheetahVel as PEARL, also keeping the metric stable over the training. When comparing with other on-policy methods, TrMRL significantly improved performance and sample efficiency.

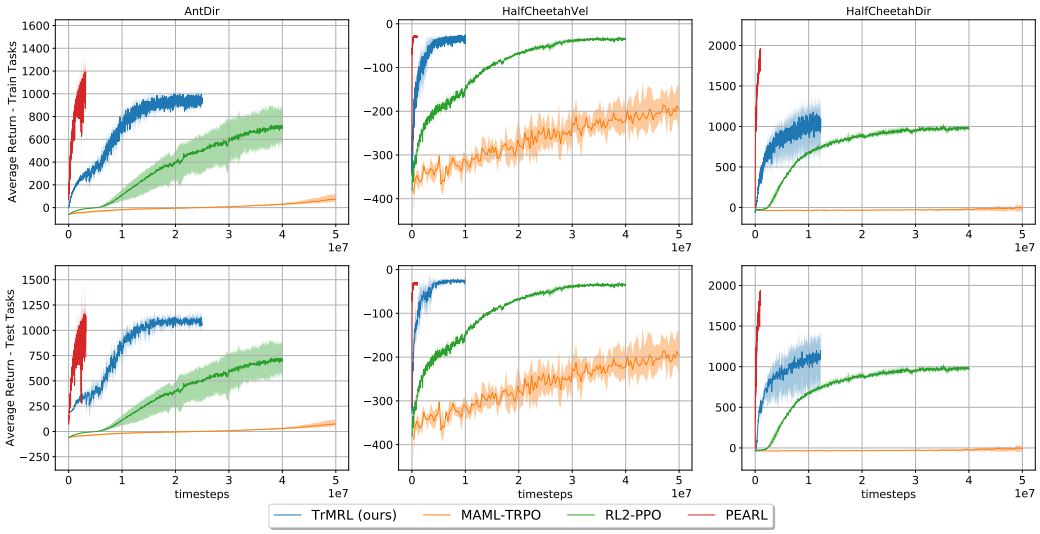

Figure 8: Meta-Training results for MuJoCo locomotion benchmarks. The plots on top represent performance on training tasks, while the plots on bottom represent the test tasks.

C WORKING MEMORIES LATENT VISUALIZATION

Figure 9 presents a 3-D view of the working memories from the HalfCheetahVel environment. We sampled some tasks (target velocities) and collected working memories during the meta-test setting. We observe that this embedding space learns a representation of each MDP as a distribution over the working memories, as suggested in Section 4. In this visualization, we can draw planes that approximately distinguish these tasks. Working memories that cross this boundary represent the ambiguity between two tasks. Furthermore, this representation also learns the similarity of tasks: for example, the cluster of working memories for target velocity $v = 1.0$ is between the clusters for $v = 0.5$ and $v = 1.5$. This property induces knowledge sharing among all the tasks, which suggests the sample efficiency behind TrMRL meta-training.

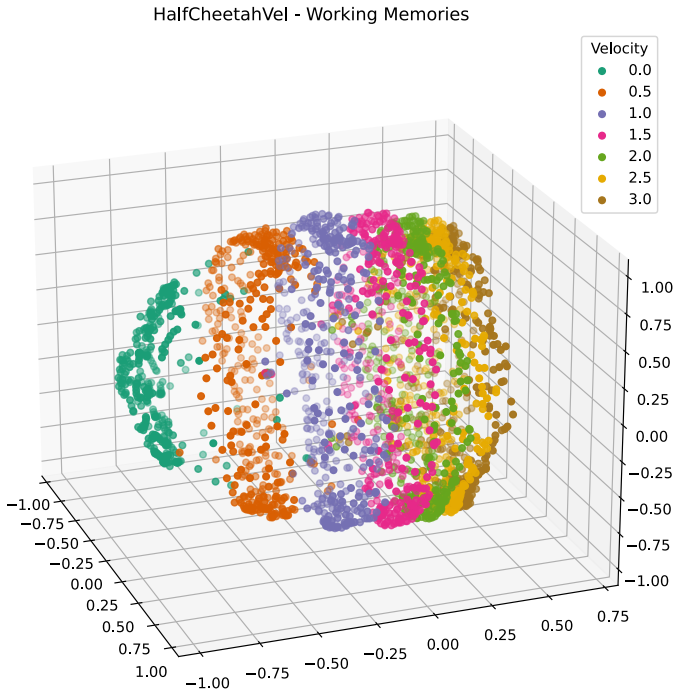

Figure 9: 3-D Latent visualization of the working memories for the HalfCheetahVel environment. We plotted the 3 most relevant components from PCA. TrMRL learns a representation of each MDP as a distribution over the working memories. This representation distinguishes the tasks and approximates similar tasks, which helps knowledge sharing among them.

D ABLATION STUDY

In this section, we present an ablation study regarding the main components of TrMRL to identify how they affect the performance of the learned agents. For all the scenarios, we considered one environment for each benchmark to represent both locomotion (HalfCheetahVel) and dexterous manipulation (MetaWorld ML1 Reach-v2). We evaluated the meta-training phase so that we could analyze both sample efficiency and asymptotic performance.

D.1 T-FIXUP

We aim to evaluate how the T-Fixup is essential to stabilize the training and improve the agent. We present results in Figure 10. For Reach-v2, T-Fixup does not significantly change the performance. We hypothesize that this happens because Reach-V2 is a simple environment for exploration since the reward signal is always dense. Nevertheless, T-Fixup played a crucial role in the HalfCheetahVel environment (where rewards intrinsically depend on the current locomotion gait). This result suggests that stabilizing learning since the beginning is very critical in the RL setting. This is often the moment with higher exploration rates that prevent sub-optimal policies.

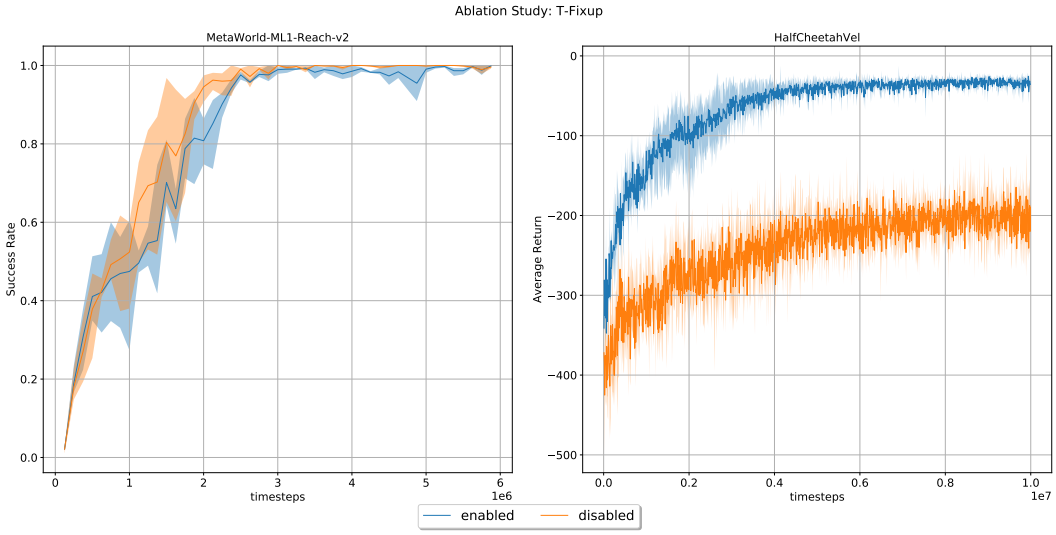

Figure 10: Ablation results for the T-Fixup component.

D.2 WORKING MEMORY SEQUENCE LENGTH

Here, our goal is to evaluate how the history length L of working memories affects the final performance of the TrMRL agent. Figure 11 present the results. For the Reach-v2 environment, there is a clear improvement by adding more working memories to the input. This is because the tasks in this environment (goal variations) are more ambiguous than in HalfCheetahVel: since the main component of the reward is the distance to the goal, many potential tasks will have equivalent working memories (more specifically, all the tasks which goal are in a circumference with the center in the current gripper position). In this way, more steps will mitigate this ambiguity, helping the task identification.

For the HalfCheetahVel environment, there is an improvement from $L = 1$ to $L = 5$. However, the performance decreases after it. We hypothesize that this environment does not need many working memories to uniquely identifies a task and a longer sequence implies a harder optimization.

D.3 NUMBER OF LAYERS

Another important component is the network depth. In Section 4, we hypothesized that more layers would help to recursively build a more meaningful version of the episodic memory since we interact

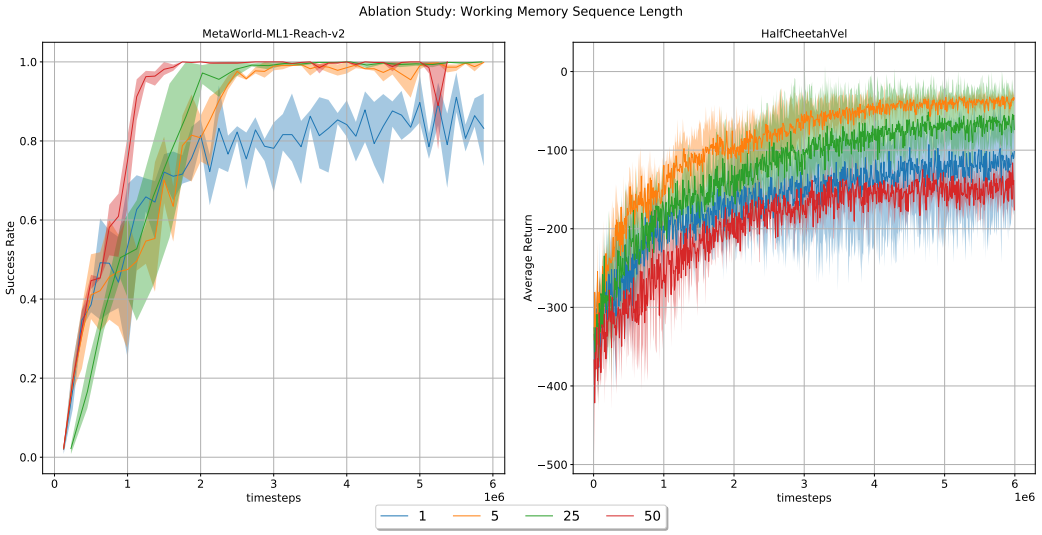

Figure 11: Ablation results for the working memory sequence length.

with output memories from the past layer and mitigates the bias effect from the task representations. Figure 12 shows how TrMRL behaves according to the number of layers. We observe a similar pattern to the previous ablation case. For Reach-v2, more layers improved the performance by reducing the effect of ambiguity and biased task representations. For HalfCheetaVel, we can see an improvement from a single layer to 4 or 8 layers, but for 12 layers, sample efficiency starts to decrease. On the other hand, we highlight that even for a deep network with 12 layers, we have a stable optimization procedure, showing the effectiveness of the T-Fixup initialization.

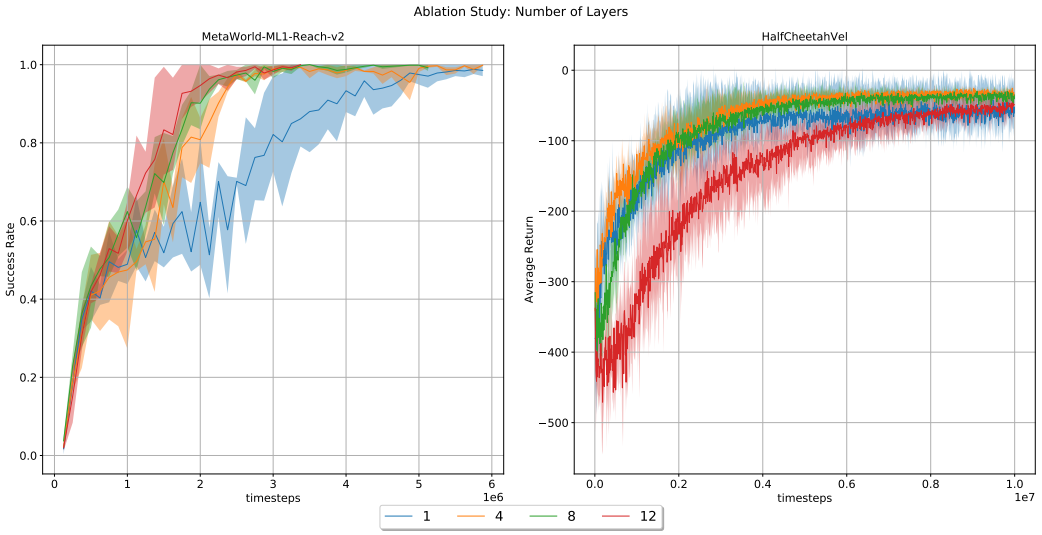

Figure 12: Ablation study for the number of transformer layers.

D.4 Number of Attention Heads

The last ablation case relates to the number of attention heads in each MHSA block. In Section 4, we highlighted that multiples heads would diversify working memory representation and improve network expressivity. In Figure 13, we observe that more heads slightly increased the performance in HalfCheetahVel and did not interfere in Reach-v2 significantly.

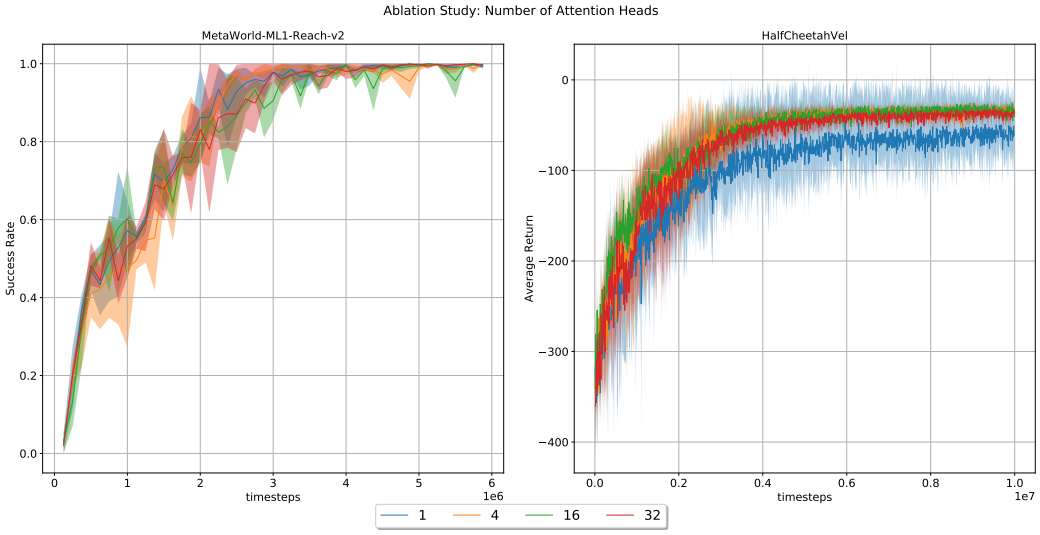

Figure 13: Ablation study for the number of attention heads.