Transformers are Meta-Reinforcement Learners

Luckeciano C. Melo [1 2]

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. We show that the self-attention computes a consensus representation that minimizes the Bayes Risk at each layer and provides meaningful features to compute the best actions. We conducted experiments in high-dimensional continuous control environments for locomotion and dexterous manipulation. Results show that TrMRL presents comparable or superior asymptotic performance, sample efficiency, and out-of-distribution generalization compared to the baselines 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., 2020b): the attention weights work as context-dependent parameters, inducing better generalization. Furthermore, this architecture parallelizes token processing by design. This property avoids backpropagation through time, making it less prone to vanishing/exploding gradients, a very common problem 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 hypothesize that this challenge is because the instability of early stages of transformer optimization harms initial exploration, which is crucial for environments where the learned behaviors must guide exploration to prevent poor policies. 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).

We conducted a series of experiments to evaluate meta-training, fast adaptation, and out-of-distribution generalization in continuous control environments for locomotion and

[1]Microsoft, USA [2]Center of Excellence in Artificial Intelligence (Deep Learning Brazil), Brazil. Correspondence to: Luckeciano C. Melo <luckeciano@gmail.com>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

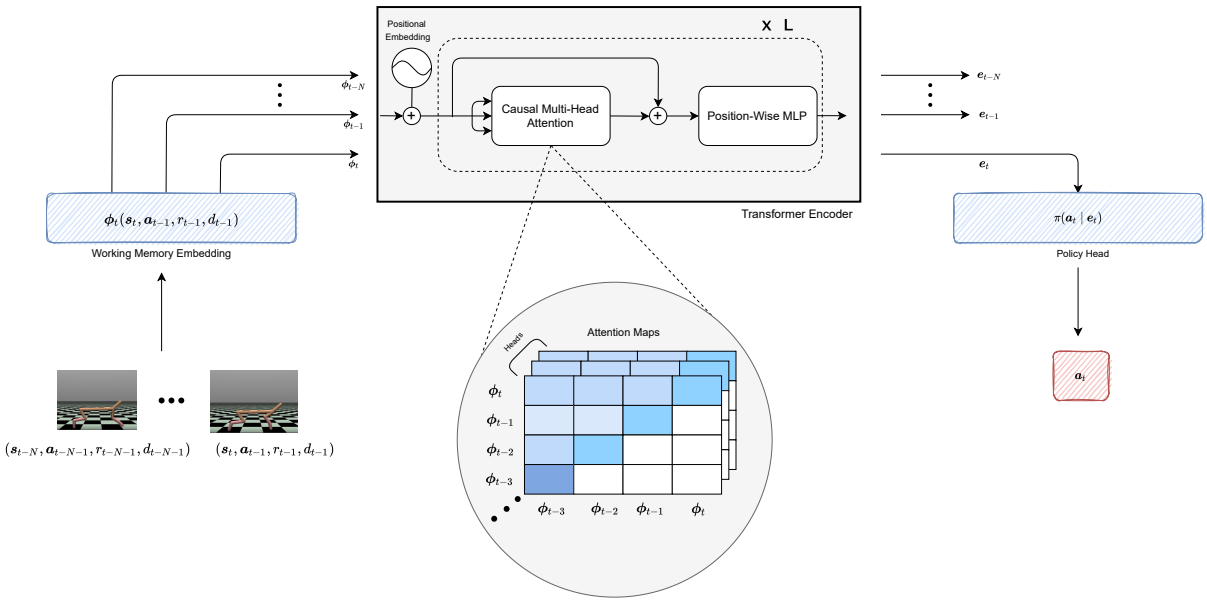

Figure 1. Illustration of the TrMRL agent. At each timestep, it associates the recent past of working memories to build an episodic memory through transformer layers recursively. We argue that the self-attention works as a fast adaptation strategy since it provides context-dependent parameters.

robotic manipulation. Results show that TrMRL presents comparable or superior performance and sample efficiency compared to the meta-RL baselines. We also conducted an experiment to validate the episode memory refinement process. Finally, we 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. This formulation presents more challenges, since the agent needs to explore in the environment and apply RL gradients to maximize rewards, which is acknowledged to be much noisier (Norouzi et al., 2016).

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 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., 2020a; 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 resid-

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

ual 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:

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

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 $0.67N^{-\frac{1}{4}}$.

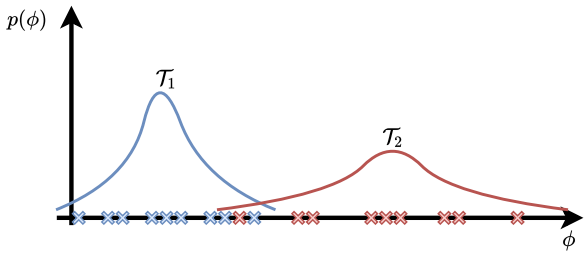

Figure 2. The illustration of two tasks ($\mathcal{T}_1$ and $\mathcal{T}_2$) as distributions over working memories. The intersection of both densities represents the ambiguity between $\mathcal{T}_1$ and $\mathcal{T}_2$.

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:

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

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.

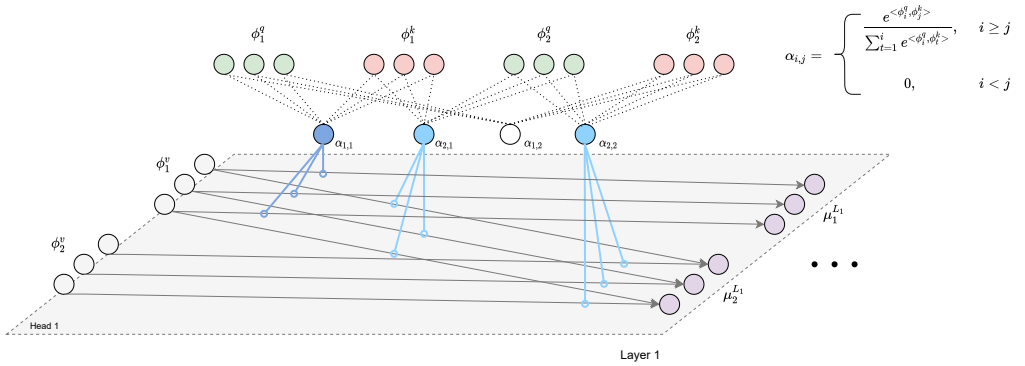

Figure 3. Illustration of causal self-attention as a fast adaptation strategy. In this simplified scenario (2 working memories), the attention weights $\alpha_{i,j}$ drives the association between the current working memory and the past ones to compute a task representation μ_t. Self-attention computes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

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.

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:

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

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:

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

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

[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.

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 now argue that the transformer model implements a memory reinstatement mechanism for episodic memory retrieval. An episodic memory system is a long-lasting memory that allows an agent to recall and re-experience personal events (Tulving, 2002). It complements the working memory system, which is active and relevant for short periods (Baddeley, 2010) and works as a buffer for episodic memory retrieval (Zilli & Hasselmo, 2008). Adopting this memory interaction model, we model an episodic memory as a transformation over a collection of past memories. More concretely, we consider that a transformer layer implements this transformation:

$$e_t^l = f(e_0^{l-1}, \dots, e_t^{l-1}), \tag{6}$$

where e_t^l represents the episodic memory retrieved from the last layer l for the timestamp t and f represents the transformer layer. Equation 6 provides a recursive definition, and e_t^0 (the base case) corresponds to the working memories ϕ_t In this way, the transformer architecture recursively refines the episodic memory interacting memories retrieved from the past layer. We show the pseudocode for this process in Algorithm 1 (Appendix G). This refinement is guaranteed by a crucial property of the self-attention mechanism: it computes a consensus representation across the input memories associated to the sub-trajectory, as stated by Theorem 4.1 (Proof in Appendix F). Here, we define consensus representation as the memory representation that is closest on average to all likely representations (Kumar & Byrne, 2004), i.e., minimizes the Bayes risk considering the set of episodic memories.

Theorem 4.1. *Let $\mathcal{S}^l = (\boldsymbol{e}_0^l, \dots, \boldsymbol{e}_N^l) \sim p(\boldsymbol{e}|\mathcal{S}^l, \boldsymbol{\theta}_l)$ be a set of normalized episodic memory representations sampled from the posterior distribution $p(\boldsymbol{e}|\mathcal{S}^l, \boldsymbol{\theta}_l)$ induced by the transformer layer l, parameterized by $\boldsymbol{\theta_l}$. Let K, Q, V be the Key, Query, and Value vector spaces in the self-attention mechanism. Then, the self-attention in the layer $l+1$ computes a consensus representation $e_N^{l+1} = \frac{\sum_{t=1}^N \boldsymbol{e}_t^{l,V} \cdot \exp \langle \boldsymbol{e}_t^{l,Q}, \boldsymbol{e}_i^{l,K} \rangle}{\sum_{t=1}^N \exp \langle \boldsymbol{e}_t^{l,Q}, \boldsymbol{e}_i^{l,K} \rangle}$ whose associated Bayes risk (in terms of negative cosine similarity) lower bounds the Minimum Bayes Risk (MBR) predicted from the set of candidate samples $\mathcal{S}^l$ projected onto the V space.*

Lastly, we condition the policy head in the episodic memory from the current timestep to sample an 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 Discussion

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 B. For reproducibility (source code and hyperparameters), we refer to the released source code at `https://github.com/luckeciano/transformers-metarl`.

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 E 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 $E = 2$. 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

[3]We highlight that both MuJoCo (Locomotion Tasks) and Meta-World 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.

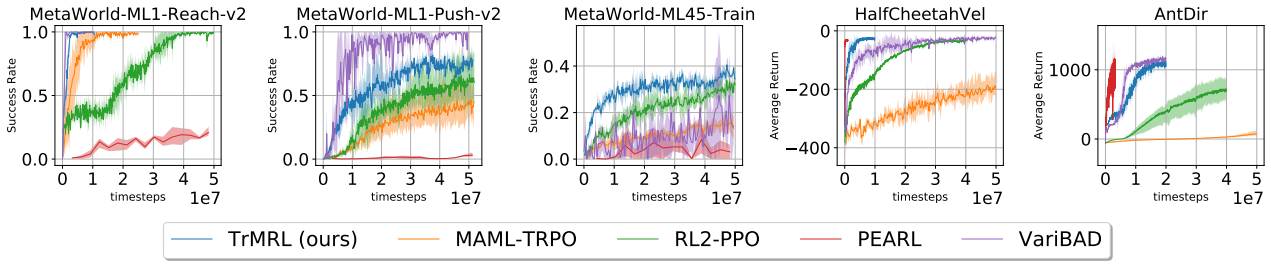

Figure 4. Meta-Training results for MetaWorld (success rate) and MuJoCo (average return) benchmarks. All subplots represent performance on test tasks over the training timesteps.

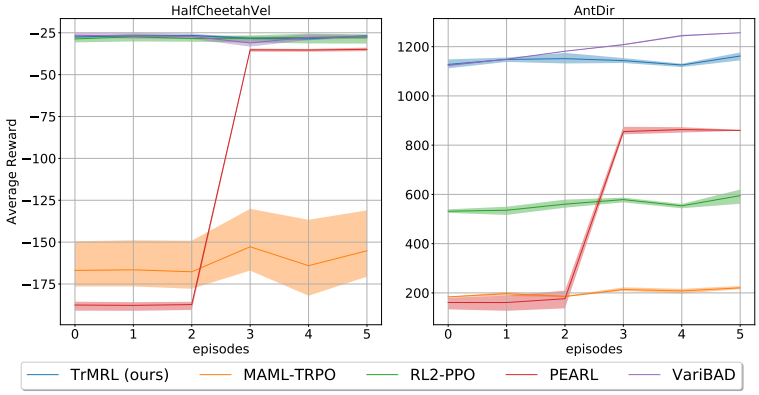

Figure 5. Fast adaptation results on MuJoCo locomotion tasks. Each curve represents the average performance over 20 test tasks. TrMRL presented high performance since the first episode due to the online adaptation nature from attention weights.

most recent one (in the "front").

Comparison Methods. For comparison, we evaluated three different meta-RL baselines: RL^2 (Duan et al., 2016), optimized using PPO (Schulman et al., 2017); PEARL (Rakelly et al., 2019); MAML (Finn et al., 2017), whose outer-loop used TRPO (Schulman et al., 2015); and VariBAD ((Zintgraf et al., 2020).

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 (success rates) and MuJoCo (average returns) environments. All subplots represent performance on test tasks over the training timesteps. TrMRL presented comparable or superior performance to the baseline methods. In scenarios where the task ambiguity is high, VariBAD presented stronger results, especially for Push-v2. In this context, ambiguity relates to the same working memories belonging to multiple different MDPs (as illustrated in Figure 2). These results support the effectiveness of the VariBAD objective that incorporates task uncertainty directly during action selection (Zintgraf et al., 2020). Reach-v2 and AntDir also presented some level of ambiguity, but TrMRL is on par with these scenarios. For scenarios with less ambiguity, such as HalfCheetaVel, TrMRL is way more sample efficient than VariBAD. It is worth mentioning that VariBAD's training objective can be leveraged by other encoding methods (such as TrMRL) (Zintgraf et al., 2020), and we let this as a future line of work.

PEARL failed to explore and learn these robotic manipulation tasks: prior work already pointed out the difficulty of training PEARL's task encoder for dexterous manipulation (Yu et al., 2021). On the other side, it presented better results on locomotion tasks, especially in sample efficiency. This is because of its off-policy nature inherited from the Soft Actor-Critic framework (Haarnoja et al., 2018). Compared with other on-policy methods (such as RL^2 and MAML), TrMRL significantly improved performance or sample efficiency.

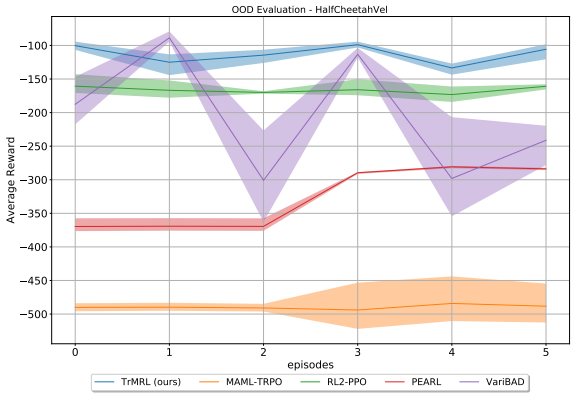

Figure 6. OOD Evaluation in HalfCheetahVel environment. TrMRL surpasses all the baselines methods with a good margin, suggesting that the context-dependent weights learned a robust adaptation strategy, while other methods memorized some aspects of the standard distribution of tasks.

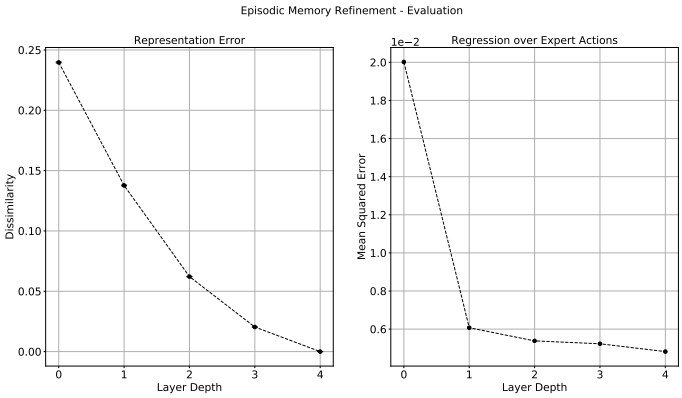

Figure 7. Episodic Memory Refinement Evaluation. We sampled 30 tasks from the HalfCheetahVel and interacted one of the trained policies to collect the episodic memories from each layer for 3 episodes per task. For the representation error, we computed the dissimilarity as 1.0 minus the cosine similarity between the episodic memory and the final representation from the last layer. For the linear regression models, we trained using a set of 20 tasks and evaluated the mean squared error over a test set of 10 tasks.

Finally, for "MetaWorld-ML45-Train" (a simplified version of ML45 benchmark where we evaluate on the training tasks), the more complex environment in this work, TrMRL achieved the best learning performance among the methods, supporting that the proposed method can generalize better across very different tasks in comparison with other methods. Additionally, VariBAD's performance suggests that its proposed objective does not scale well for this scenario, harming the final performance when compared with RL2. We hypothesize that the supervision from the dynamics and rewards in the training objective provides conflicting learning signals to the RNN.

Fast Adaptation Evaluation. A critical skill for meta-RL agents is the capability of adapting to new tasks given a few episodes. We evaluate this by running meta-testing on 20 test tasks over six sequential episodes. Each agent runs its adaptation strategy to identify the task and maximize the reward across episodes. Figure 5 presents the results for the locomotion tasks. TrMRL again presents a comparable or superior performance in comparison to the baselines.

We highlight that TrMRL presented high performance since the first episode. 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 9 in Appendix D). Therefore, it presents a nice property for online adaptation. Other methods, such as PEARL and MAML, do not present such property, as 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 6 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 for locomotion tasks among the methods but performed poorly in this setting. VariBAD also presented unstable performance across the episodes. These results suggest that their task encoding mechanisms do not generate effective latent representations for OOD tasks.

Episodic Memory Refinement. We evaluated Theorem 4.1 empirically in Figure 7. In these two experiments, we interacted with one of the trained policies within the HalfCheetahVel environment and collected the episodic memories computed from each layer throughout a few episodes and across different tasks. Firstly, we computed the dissimilarity between the memories from each layer and the final representation from the last layer. We refer to this metric as the representation error. Next, we computed linear regression models over the "expert" actions sampled from the trained policy. We used the episodic memories from each layer as features and reported the mean squared error on a test set. We aim to evaluate how meaningful the representations

from each layer are by predicting the best action given a very lightweight model.

Figure 7 shows that the representation error gradually decreases throughout the layers, suggesting the refinement of the episodic memory and the convergence to a consensus representation, as supported by Theorem 4.1. It also shows a correlated behavior in the regression error over the expert actions. This evidence suggests that the refinement of an episodic memory induces a consensus representation that is more meaningful to predict the best actions from the expert agent.

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.

TrMRL demonstrated a valuable capability of learning from reward signals. On the other side, recent Language Models presented substantial improvements by designing self-supervised tasks (Devlin et al., 2019; Raffel et al., 2020) or even automating their generation (Shin et al., 2020). As future work, we aim to investigate how to enable these forms of self-supervision to leverage off-policy data collected during the training and further improve sample efficiency in transformers for the meta-RL scenario.

References

Andrychowicz, M., Denil, M., Gómez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and de Freitas, N. Learning to learn by gradient descent by gradient descent. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/fb87582825f9d28a8d42c5e5e5e8b23d-Paper.pdf.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization, 2016.

Baddeley, A. Working memory. *Current Biology*, 20(4):R136–R140, 2010. ISSN 0960-9822. doi: https://doi.org/10.1016/j.cub.2009.12.014. URL https://www.sciencedirect.com/science/article/pii/S0960982209021332.

Bengio, Y., Bengio, S., and Cloutier, J. 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.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020a. URL https://arxiv.org/abs/2005.14165.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners, 2020b.

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. Emerging properties in self-supervised vision transformers, 2021.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling, 2021.

Chen, Y., Hoffman, M. W., Colmenarejo, S. G., Denil, M., Lillicrap, T. P., Botvinick, M., and de Freitas, N. Learning to learn without gradient descent by gradient descent. In Precup, D. and Teh, Y. W. (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.

Co-Reyes, J. D., Miao, Y., Peng, D., Real, E., Levine, S., Le, Q. V., Lee, H., and Faust, A. Evolving reinforcement learning algorithms, 2021.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. 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.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale, 2021b.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. Rl^2: Fast reinforcement learning via slow reinforcement learning, 2016.

Fakoor, R., Chaudhari, P., Soatto, S., and Smola, A. J. Meta-q-learning, 2020.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In Precup, D. and Teh, Y. W. (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.

Finn, C., Xu, K., and Levine, S. 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.

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

Glorot, X. and Bengio, Y. 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.

Gupta, A., Mendonca, R., Liu, Y., Abbeel, P., and Levine, S. 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.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition, 2015.

Hoskin, A. N., Bornstein, A. M., Norman, K. A., and Cohen, J. D. 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.

Huang, X. S., Perez, F., Ba, J., and Volkovs, M. Improving transformer optimization through better initialization. In III, H. D. and Singh, A. (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.

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

Janner, M., Li, Q., and Levine, S. Reinforcement learning as one big sequence modeling problem, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2017.

Kumar, S. and Byrne, W. 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.

Li, K. and Malik, J. Learning to optimize, 2016.

Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., and Han, J. On the variance of the adaptive learning rate and beyond, 2020.

Loynd, R., Fernandez, R., Celikyilmaz, A., Swaminathan, A., and Hausknecht, M. Working memory graphs, 2020.

Melo, L. C., Maximo, M. R. O. A., and da Cunha, A. M. Bottom-up meta-policy search, 2019.

Mishra, N., Rohaninejad, M., Chen, X., and Abbeel, P. A simple neural attentive meta-learner, 2018.

Nichol, A., Achiam, J., and Schulman, J. On first-order meta-learning algorithms, 2018.

Norouzi, M., Bengio, S., Chen, z., Jaitly, N., Schuster, M., Wu, Y., and Schuurmans, D. Reward augmented maximum likelihood for neural structured prediction. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/2f885d0fbe2e131bfc9d98363e55d1d4-Paper.pdf.

Oh, J., Hessel, M., Czarnecki, W. M., Xu, Z., van Hasselt, H. P., Singh, S., and Silver, D. Discovering reinforcement learning algorithms. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (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.

Ortega, P. A., Wang, J. X., Rowland, M., Genewein, T., Kurth-Nelson, Z., Pascanu, R., Heess, N., Veness, J., Pritzel, A., Sprechmann, P., Jayakumar, S. M., McGrath, T., Miller, K., Azar, M., Osband, I., Rabinowitz, N., György, A., Chiappa, S., Osindero, S., Teh, Y. W., van Hasselt, H., de Freitas, N., Botvinick, M., and Legg, S. Meta-learning of sequential strategies, 2019.

Parisotto, E., Song, H. F., Rae, J. W., Pascanu, R., Gulcehre, C., Jayakumar, S. M., Jaderberg, M., Kaufman, R. L., Clark, A., Noury, S., Botvinick, M. M., Heess, N., and Hadsell, R. Stabilizing transformers for reinforcement learning, 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer, 2020.

Rakelly, K., Zhou, A., Quillen, D., Finn, C., and Levine, S. Efficient off-policy meta-reinforcement learning via probabilistic context variables, 2019.

Ritter, S., Wang, J., Kurth-Nelson, Z., Jayakumar, S., Blundell, C., Pascanu, R., and Botvinick, M. Been there, done that: Meta-learning with episodic recall. In Dy, J. and Krause, A. (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.

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

Ritter, S., Wang, J. X., Kurth-Nelson, Z., Jayakumar, S. M., Blundell, C., Pascanu, R., and Botvinick, M. Been there, done that: Meta-learning with episodic recall, 2018c.

Ritter, S., Faulkner, R., Sartran, L., Santoro, A., Botvinick, M., and Raposo, D. Rapid task-solving in novel environments, 2021.

Rothfuss, J., Lee, D., Clavera, I., Asfour, T., and Abbeel, P. Promp: Proximal meta-policy search, 2018.

Rovee-Collier, C. *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.

Schmidhuber, J., Zhao, J., and Wiering, M. Simple principles of metalearning. Technical report, 1996.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In Bach, F. and Blei, D. (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.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms, 2017.

Shin, T., Razeghi, Y., IV, R. L. L., Wallace, E., and Singh, S. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *CoRR*, abs/2010.15980, 2020. URL https://arxiv.org/abs/2010.15980.

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

Todorov, E., Erez, T., and Tassa, Y. 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.

Tulving, E. Episodic memory: From mind to brain. *Annual review of psychology*, 53:1–25, 02 2002. doi: 10.1146/annurev.psych.53.100901.135114.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. 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.

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

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M. Learning to reinforcement learn, 2017.

Wang, J. X., Kurth-Nelson, Z., Kumaran, D., Tirumala, D., Soyer, H., Leibo, J. Z., Hassabis, D., and Botvinick, M. 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.

Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., and Liu, T.-Y. On layer normalization in the transformer architecture, 2020.

Yu, T., Quillen, D., He, Z., Julian, R., Narayan, A., Shively, H., Bellathur, A., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning, 2021.

Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Jiang, Z., Tay, F. E., Feng, J., and Yan, S. Tokens-to-token vit: Training vision transformers from scratch on imagenet, 2021.

Zilli, E. A. and Hasselmo, M. E. Modeling the role of working memory and episodic memory in behavioral tasks. *Hippocampus*, 18(2):193–209, 2008.

Zintgraf, L., Shiarlis, K., Igl, M., Schulze, S., Gal, Y., Hofmann, K., and Whiteson, S. Varibad: A very good method for bayes-adaptive deep rl via meta-learning, 2020.

Zintgraf, L. M., Shiarlis, K., Kurin, V., Hofmann, K., and Whiteson, S. Fast context adaptation via meta-learning, 2019.

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

A. Reproducibility Statement

Code Release. To ensure the reproducibility of our research, we released all the source code associated with our models and experimental pipeline. We refer to the supplementary material of this submission. It also includes the hyperparameters and the scripts to execute all the scenarios presented in this paper.

Baselines Reproducibility. We strictly reproduced the results from prior work implementations for baselines, and we provide their open-source repositories for reference.

Proof of Theoretical Results and Pseudocode. We provide a detailed proof of Theorem 4.1 in Appendix F, containing all assumptions considered. We also provide pseudocode from TrMRL's agent to improve clarity on the proposed method.

Availability of all simulation environments. All environments used in this work are freely available and open-source.

B. Metal-RL Environments Description

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

B.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 2 different environments.

- **AntDir**: This environment has an ant 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.

B.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 ML1 mode 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. Given the difficulty of such benchmark, in this work, we adopted a simplified version, "ML45-Train", where we also evaluate the performance of the methods in the 45 training tasks.

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

C. Working Memories Latent Visualization

Figure 8 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.

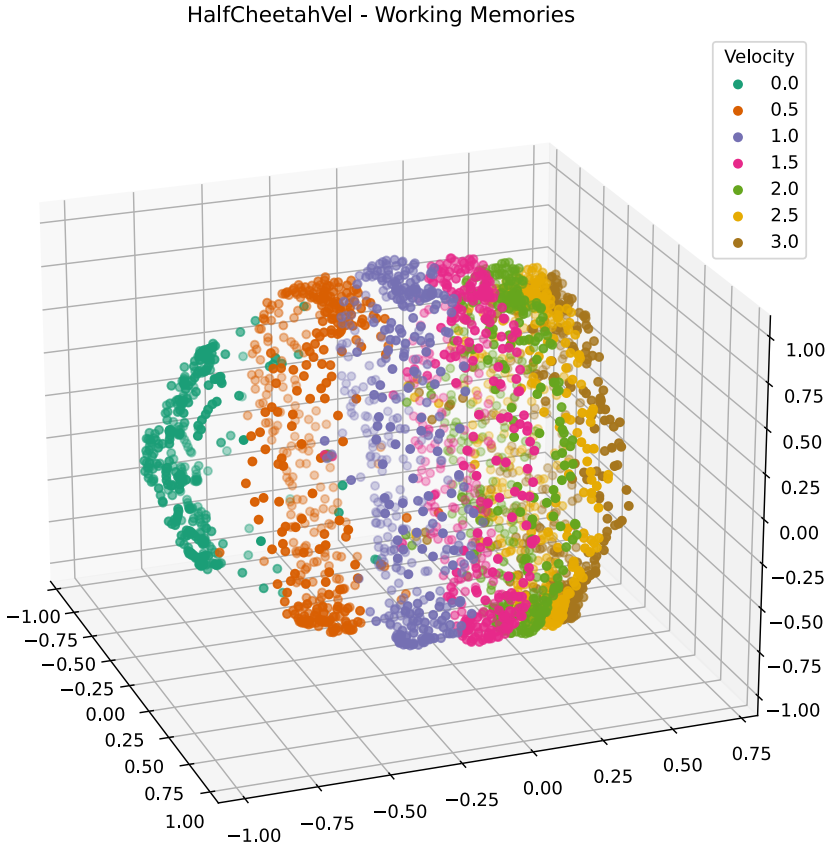

Figure 8. 3-D Latent visualization of the working memories for the HalfCheetahVel environment. We plotted the 3 most relevant components from PCA. TrMRL learns a representation of each MDP as a distribution over the working memories. This representation distinguishes the tasks and approximates similar tasks, which helps knowledge sharing among them.

D. Online Adaptation

In this Appendix, 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. Figure 9 shows that it only requires around 20 timesteps to achieve the best performance in the HalfCheetahVel environment. Therefore, it presents a nice 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.

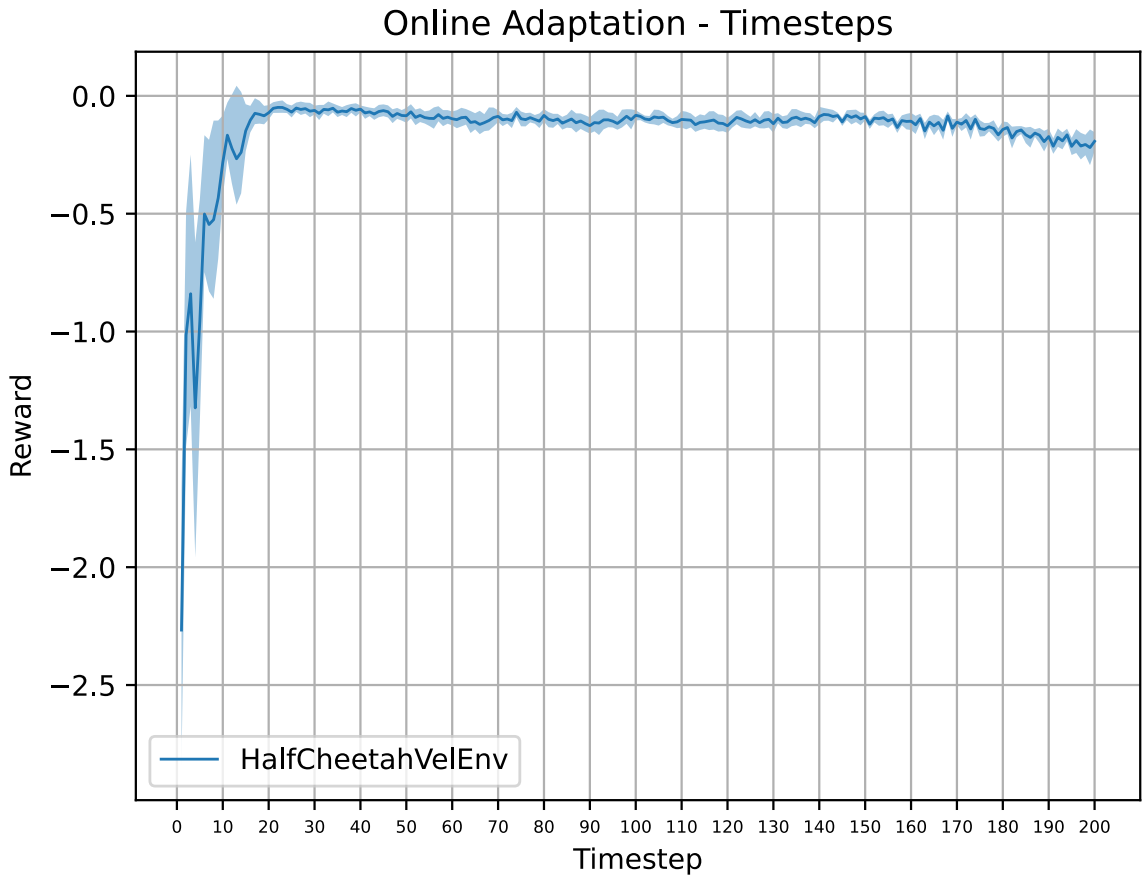

Figure 9. TrMRL's adaptation for HalfCheetahVel environment.

E. 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.

E.1. T-Fixup

In this work, we employed T-Fixup to address the instability from the early stages of transformer training, given the reasons described in Section 3.3. In RL, the early stages of training are also the moment when the learning policies are more exploratory to cover the state and action spaces better and discover rewards, preventing the convergence to sub-optimal policies. Hence, it is crucial for RL that the transformer policy learns appropriately since the beginning to drive exploration.

This section evaluated how T-Fixup becomes essential for environments where the learned behaviors must guide exploration to prevent poor policies. For this, we present T-Fixup ablation (Figure 10) for two settings: MetaWorld-ML1-Reach-v2 and HalfCheetahVel. For the reach environment, we compute the reward distribution using the distance between the gripper and the target location. Hence, it is always a dense and informative signal: even a random policy can easily explore the environment, and T-Fixup does not interfere with the learning curve. On the other side, HalfCheetahVel requires a functional locomotion gate to drive exploration; otherwise, it can get stuck with low rewards (e.g., cheetah is exploring while fallen). In this scenario, T-Fixup becomes crucial to prevent unstable learning updates that could collapse the learning policy to poor behaviors.

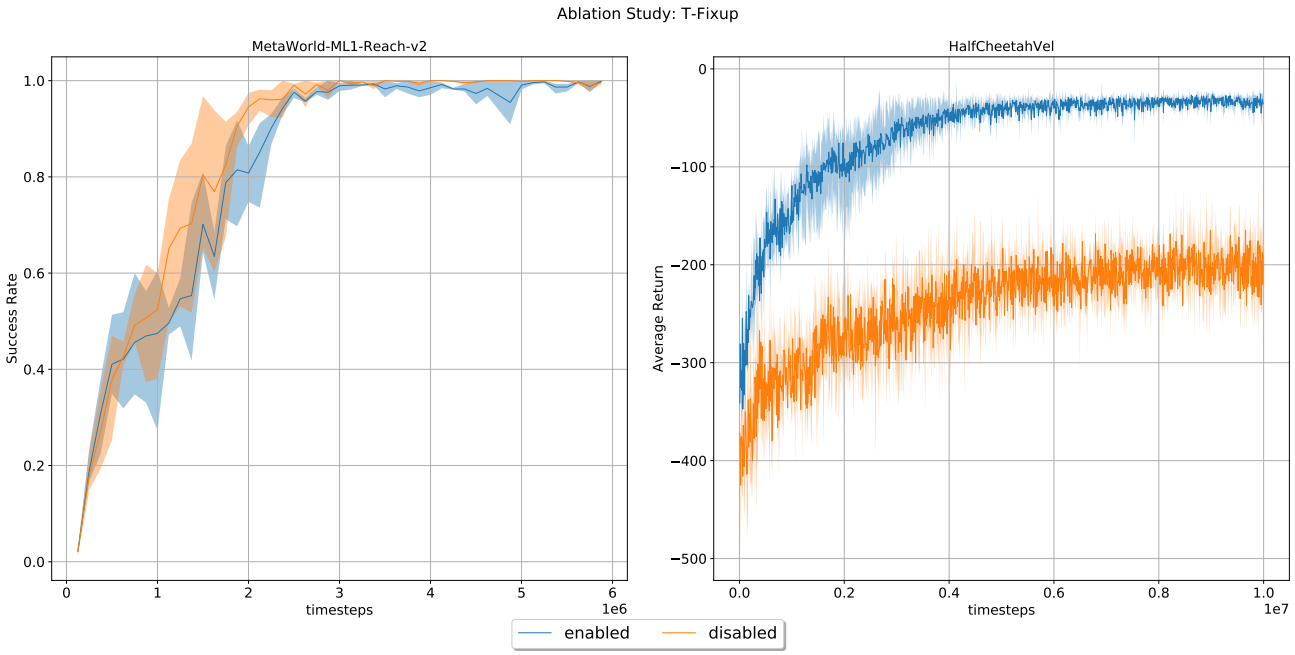

Figure 10. Ablation results for the T-Fixup component.

E.2. Working Memory Sequence Length

A meta-RL agent requires a sequence of interactions to identify the running task and act accordingly. The length of this sequence N should be large enough to address the ambiguity associated with the set of tasks, but not too long to make the transformer optimization harder and less sample efficient. In this ablation, we study two environments that present different levels of ambiguity and show that they also require different lengths to achieve optimal sample efficiency.

We first analyze MetaWorld-ML1-Reach-v2. The environment defines each target location in the 3D space as a task. The

associated reward is the distance between the gripper and this target. Hence, at each timestep, the reward is ambiguous for all the tasks located on the sphere's surface with the center in the gripper position. This suggests that the agent will benefit from long sequences. Figure 11 (left) confirms this hypothesis, as the sample efficiency improves until sequences with several timesteps (N = 50).

The HalfCheetahVel environment defines each forward velocity as a different task. The associated reward depends on the difference between the current cheetah velocity and this target. Hence, at each timestep, the emitted reward is ambiguous only for two possible tasks. To identify the current task, the agent needs to estimate its velocity (which requires a few timesteps) and then disambiguate between these two tasks. This suggests that the agent will not benefit from very long sequences. Figure 11 (right) confirms this hypothesis: there is an improvement in sample efficiency from N = 1 to N = 5, but it decreases for longer sequences as the training becomes harder (it is worth mentioning that all evaluated policies achieved the best asymptotic performance).

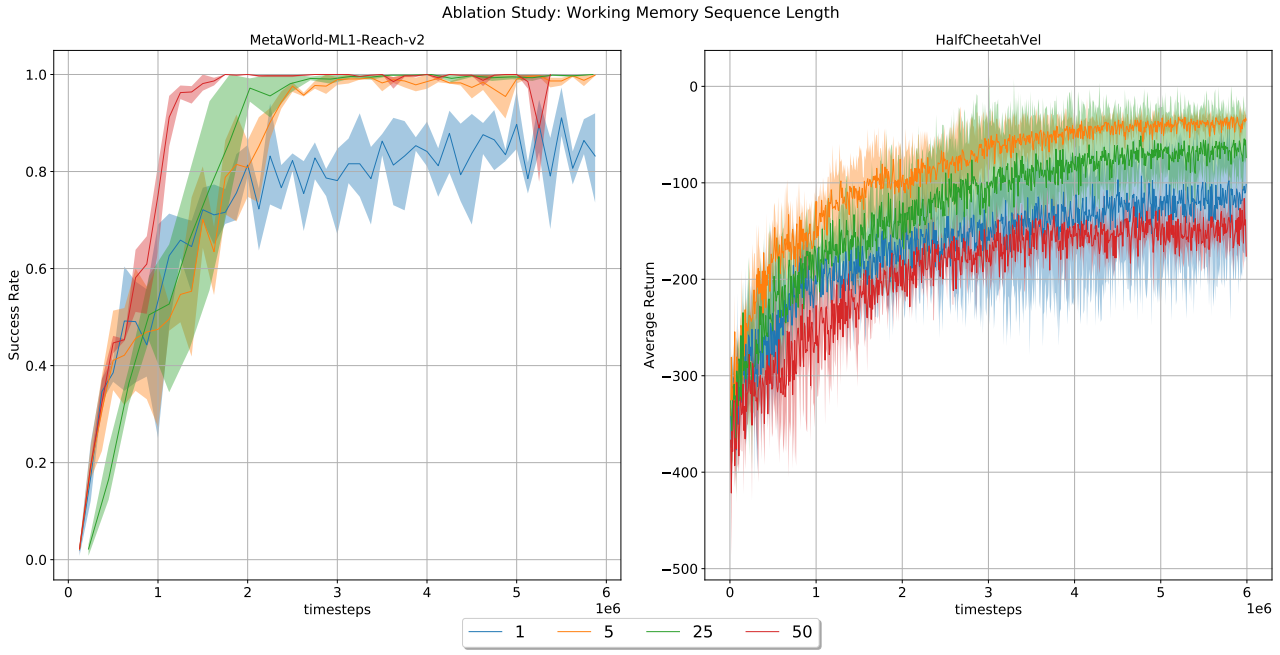

Figure 11. Ablation results for the working memory sequence length.

E.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 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.

E.4. Number of Attention Heads

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

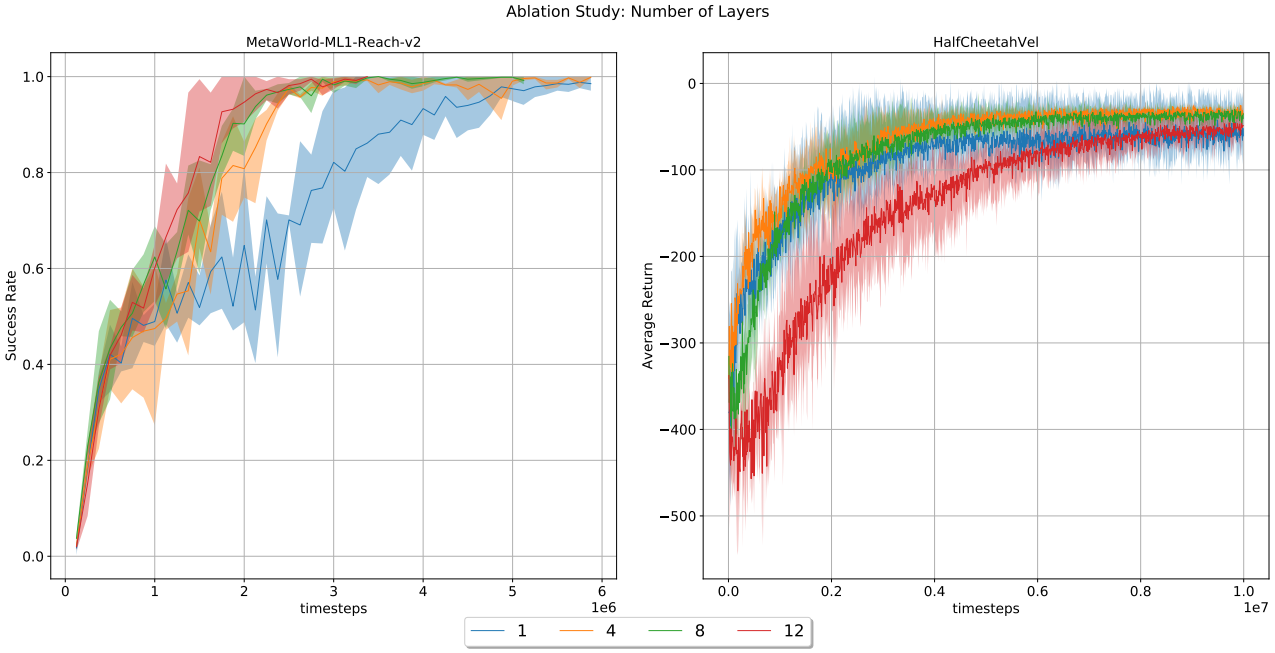

Figure 12. Ablation study for the number of transformer layers.

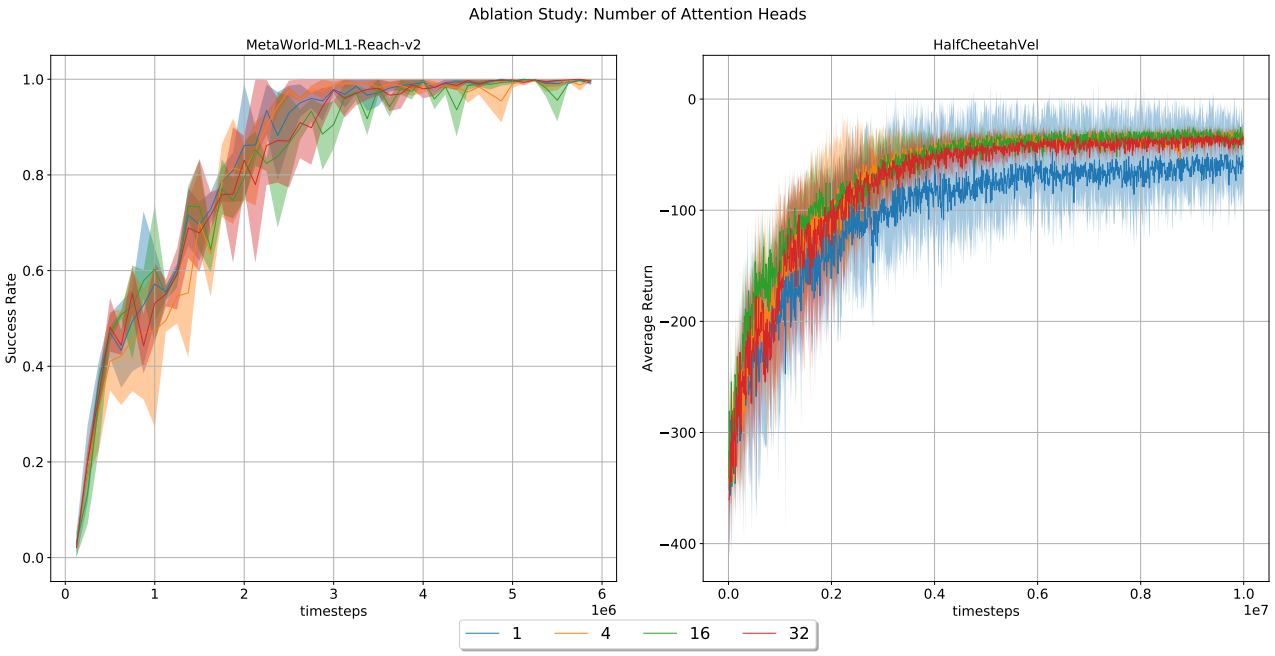

Figure 13. Ablation study for the number of attention heads.

F. Proof of Theorem 4.1

Theorem 4.1. *Let $\mathcal{S}^l = (\boldsymbol{e}_0^l, \dots, \boldsymbol{e}_N^l) \sim p(\boldsymbol{e}|\mathcal{S}^l, \boldsymbol{\theta}_l)$ be a set of normalized episodic memory representations sampled from the posterior distribution $p(\boldsymbol{e}|\mathcal{S}^l, \boldsymbol{\theta}_l)$ induced by the transformer layer l, parameterized by $\boldsymbol{\theta_l}$. Let K, Q, V be the Key, Query, and Value vector spaces in the self-attention mechanism. Then, the self-attention in the layer $l+1$ computes a consensus representation $e_N^{l+1} = \frac{\sum_{t=1}^N \boldsymbol{e}_t^{l,V} \cdot \exp\langle \boldsymbol{e}_t^{l,Q}, \boldsymbol{e}_i^{l,K} \rangle}{\sum_{t=1}^N \exp\langle \boldsymbol{e}_t^{l,Q}, \boldsymbol{e}_i^{l,K} \rangle}$ whose associated Bayes risk (in terms of negative cosine similarity) lower bounds the Minimum Bayes Risk (MBR) predicted from the set of candidate samples $\mathcal{S}^l$ projected onto the V space.*

Proof. Let us define $\mathcal{S}_V^l$ as the set containing the projection of the elements in $\mathcal{S}^l$ onto the V space: $\mathcal{S}_V^l = (\boldsymbol{e}_0^{l,V}, \dots, \boldsymbol{e}_N^{l,V})$, where $\boldsymbol{e}^{l,V} = W_V \cdot \boldsymbol{e}^l$ (W_V is the projection matrix). The Bayes risk of selecting $\hat{\boldsymbol{e}}^{l,V}$ as representation, $BR(\hat{\boldsymbol{e}}^{l,V})$, under a loss function $\mathcal{L}$, is defined by:

$$BR(\hat{\boldsymbol{e}}^{l,V}) = \mathbb{E}_{p(\boldsymbol{e}|\mathcal{S}_V^l, \boldsymbol{\theta}_l)}[\mathcal{L}(\boldsymbol{e}, \hat{\boldsymbol{e}})] \tag{7}$$

The MBR predictor selects the episodic memory $\hat{\boldsymbol{e}}^{l,V} \in \mathcal{S}_V^l$ that minimizes the Bayes Risk among the set of candidates: $\boldsymbol{e}_{MBR}^{l,V} = \arg\min_{\hat{\boldsymbol{e}} \in \mathcal{S}_V^l} BR(\hat{\boldsymbol{e}})$. Employing negative cosine similarity as loss function, we can represent MBR prediction as:

$$\boldsymbol{e}_{MBR}^{l,V} = \arg\max_{\hat{\boldsymbol{e}} \in \mathcal{S}_V^l} \mathbb{E}_{p(\boldsymbol{e}|\mathcal{S}_V^l, \boldsymbol{\theta}_l)}[\langle \boldsymbol{e}, \hat{\boldsymbol{e}} \rangle] \tag{8}$$

The memory representation outputted from a self-attention operation in layer $l+1$ is given by:

$$e_N^{l+1} = \frac{\sum_{t=1}^N \boldsymbol{e}_t^{l,V} \cdot \exp\langle \boldsymbol{e}_t^{l,Q}, \boldsymbol{e}_i^{l,K} \rangle}{\sum_{t=1}^N \exp\langle \boldsymbol{e}_t^{l,Q}, \boldsymbol{e}_i^{l,K} \rangle} = \sum_{t=1}^N \alpha_{N,t} \cdot \boldsymbol{e}_t^{l,V} \tag{9}$$

The attention weights $\alpha_{N,t}$ define a probability distribution over the samples in $\mathcal{S}_V^l$, which approximates the posterior distribution $p(\boldsymbol{e}|\mathcal{S}_V^l, \boldsymbol{\theta}_l)$. Hence, we can represent Equation 9 as an expectation: $e_N^{l+1} = \mathbb{E}_{p(\boldsymbol{e}|\mathcal{S}_V^l, \boldsymbol{\theta}_l)}[\boldsymbol{e}]$. Finally, we compute the Bayer risk for it:

$$\begin{aligned}
BR(e_N^{l+1}) &= \mathbb{E}_{p(\boldsymbol{e}|\mathcal{S}_V^l, \boldsymbol{\theta}_l)}[\langle \boldsymbol{e}, \hat{\boldsymbol{e}}_N^{l+1} \rangle] \\
&= \mathbb{E}_{p(\boldsymbol{e}|\mathcal{S}_V^l, \boldsymbol{\theta}_l)}[\langle \boldsymbol{e}, \mathbb{E}_{p(\boldsymbol{e}|\mathcal{S}_V^l, \boldsymbol{\theta}_l)}[\boldsymbol{e}] \rangle] \\
&= \left\langle \mathbb{E}_{p(\boldsymbol{e}|\mathcal{S}_V^l, \boldsymbol{\theta}_l)}[\boldsymbol{e}], \mathbb{E}_{p(\boldsymbol{e}|\mathcal{S}_V^l, \boldsymbol{\theta}_l)}[\boldsymbol{e}] \right\rangle \\
&\geq \left\langle \mathbb{E}_{p(\boldsymbol{e}|\mathcal{S}_V^l, \boldsymbol{\theta}_l)}[\langle \boldsymbol{e}, \hat{\boldsymbol{e}} \rangle], \hat{\boldsymbol{e}} \right\rangle \\
&= \mathbb{E}_{p(\boldsymbol{e}|\mathcal{S}_V^l, \boldsymbol{\theta}_l)}[\langle \boldsymbol{e}, \hat{\boldsymbol{e}} \rangle], \forall \hat{\boldsymbol{e}} \in \mathcal{S}_V^l.
\end{aligned} \tag{10}$$

□

G. Pseudocode

In this section, we present a pseudocode for TrMRL's agent during its interaction with an arbitrary MDP.

Algorithm 1 TrMRL – Forward Pass

Require: MDP $\mathcal{M} \sim p(\mathcal{M})$
Require: Working Memory Sequence Length N
Require: Parameterized function $\phi(\boldsymbol{s}, \boldsymbol{a}, r, \eta)$
Require: Transformer network with L layers $\{f_1, \ldots, f_L\}$
Require: Policy Head π
Initialize Buffer with $N-1$ PAD transitions: $\mathcal{B} = \{(\boldsymbol{s}_{PAD}, \boldsymbol{a}_{PAD}, r_{PAD}, \eta_{PAD})_i\}, i \in \{1, \ldots, N-1\}$
$t \leftarrow 0$
$\boldsymbol{s}_{next} \leftarrow \boldsymbol{s}_0$
while episode not done **do**
  Retrieve the $N-1$ most recent transitions $(\boldsymbol{s}, \boldsymbol{a}, r, \eta)$ from $\mathcal{B}$ to create the ordered subset $\mathcal{D}$
  $\mathcal{D} \leftarrow \mathcal{D} \bigcup (\boldsymbol{s}_{next}, \boldsymbol{a}_{PAD}, r_{PAD}, \eta_{PAD})$
  Compute working memories:
    $\phi_i = \phi(\boldsymbol{s}_i, \boldsymbol{a}_i, r_i, \eta_i), \forall \{\boldsymbol{s}_i, \boldsymbol{a}_i, r_i, \eta_i\} \in \mathcal{D}$
  Set $e_1^0, \ldots, e_N^0 \leftarrow \phi_1, \ldots, \phi_N$
  for each $l \in 1, \ldots, L$ **do**
    Refine episodic memories:
      $e_1^l, \ldots, e_N^l \leftarrow f_l(e_1^{l-1}, \ldots, e_N^{l-1})$
  end for
  Sample $\boldsymbol{a}_t \sim \pi(\cdot | e_N^L)$
  Collect $(\boldsymbol{s}_{t+1}, r_t, \eta_t)$ interacting with $\mathcal{M}$ applying action $\boldsymbol{a}_t$
  $\boldsymbol{s}_{next} \leftarrow \boldsymbol{s}_{t+1}$
  $\mathcal{B} \leftarrow \mathcal{B} \bigcup (\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \eta_t)$
  $t \leftarrow t + 1$
end while