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.

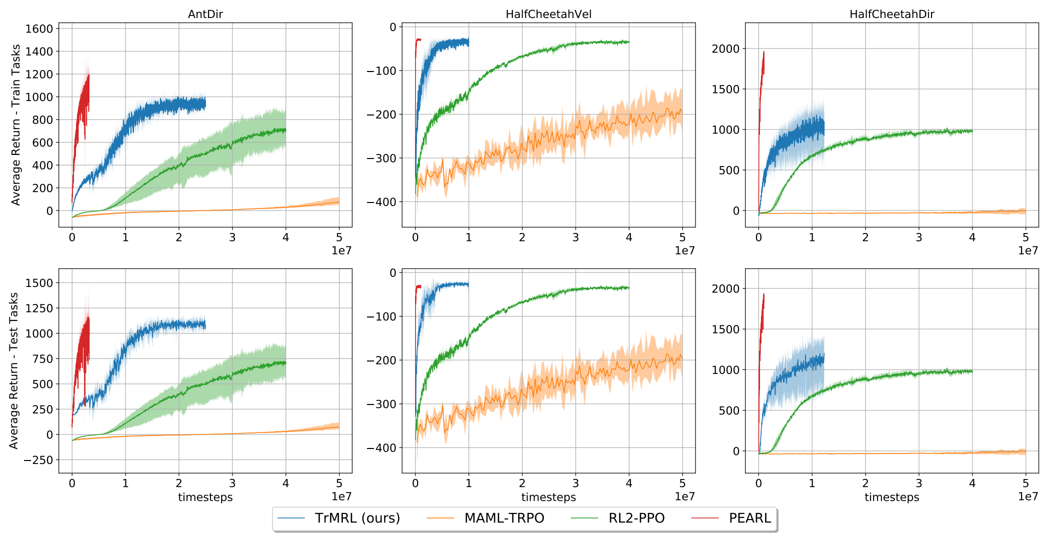

Figure 8: Meta-Training results for MuJoCo locomotion benchmarks. The plots on top represent performance on training tasks, while the plots on bottom represent the test tasks.

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.

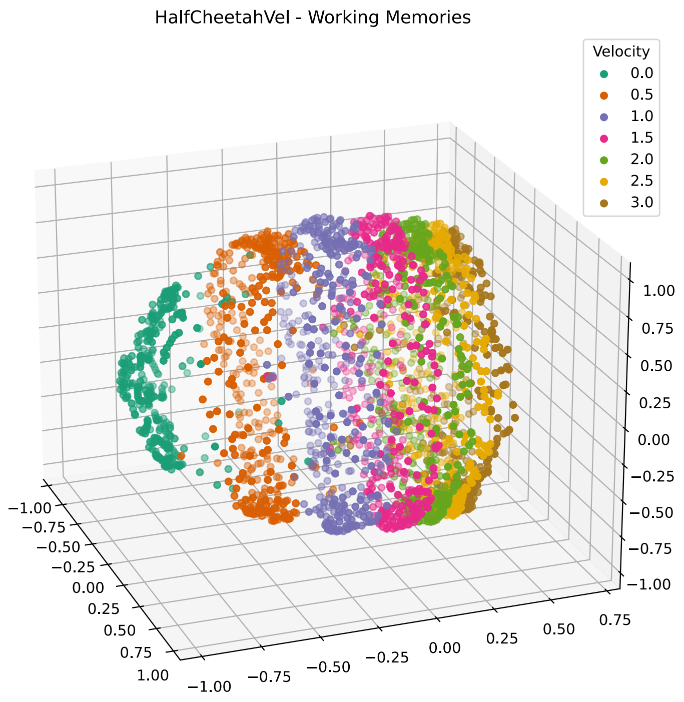

Figure 9: 3-D Latent visualization of the working memories for the HalfCheetahVel environment. We plotted the 3 most relevant components from PCA. TrMRL learns a representation of each MDP as a distribution over the working memories. This representation distinguishes the tasks and approximates similar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

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.

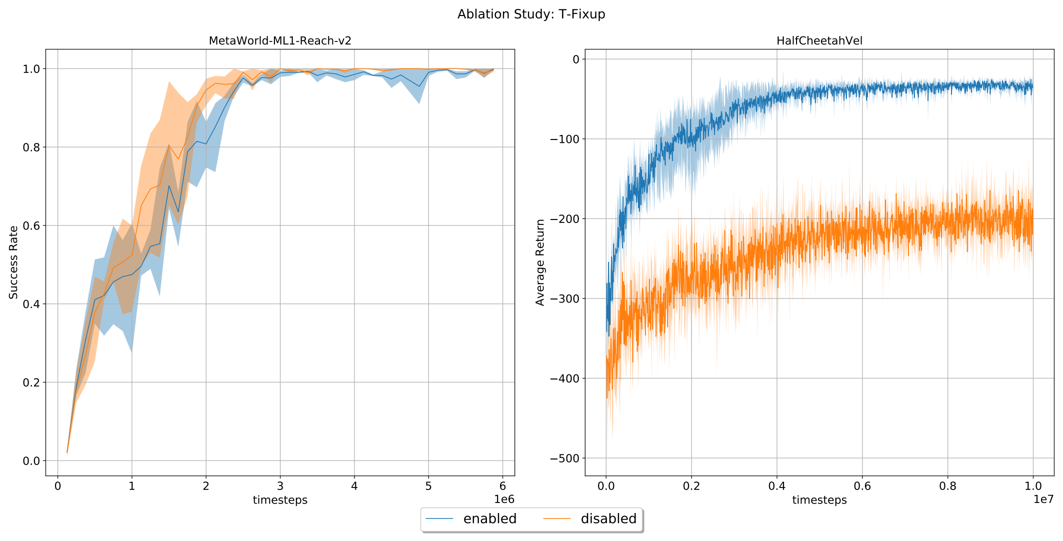

Figure 10: Ablation results for the T-Fixup component.

D.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 from N = 1 to N = 5, but the performance decreases for longer sequences as the training becomes harder.

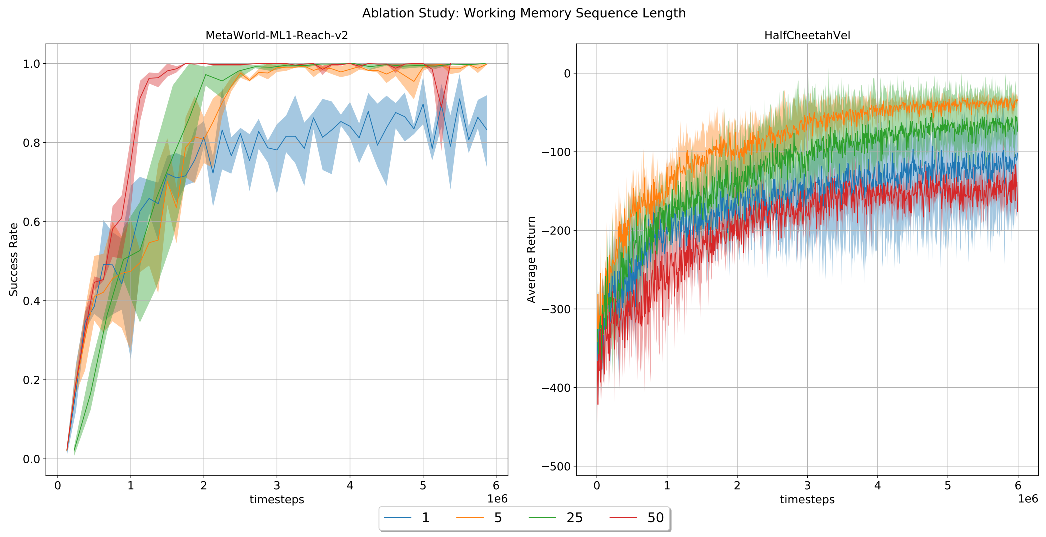

Figure 11: Ablation results for the working memory sequence length.

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 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.

D.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.

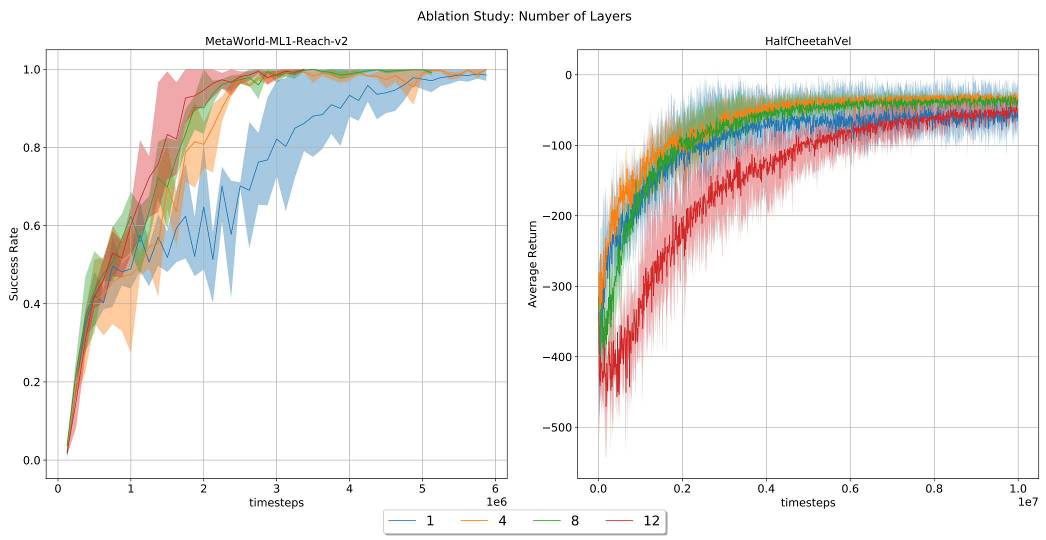

Figure 12: Ablation study for the number of transformer layers.

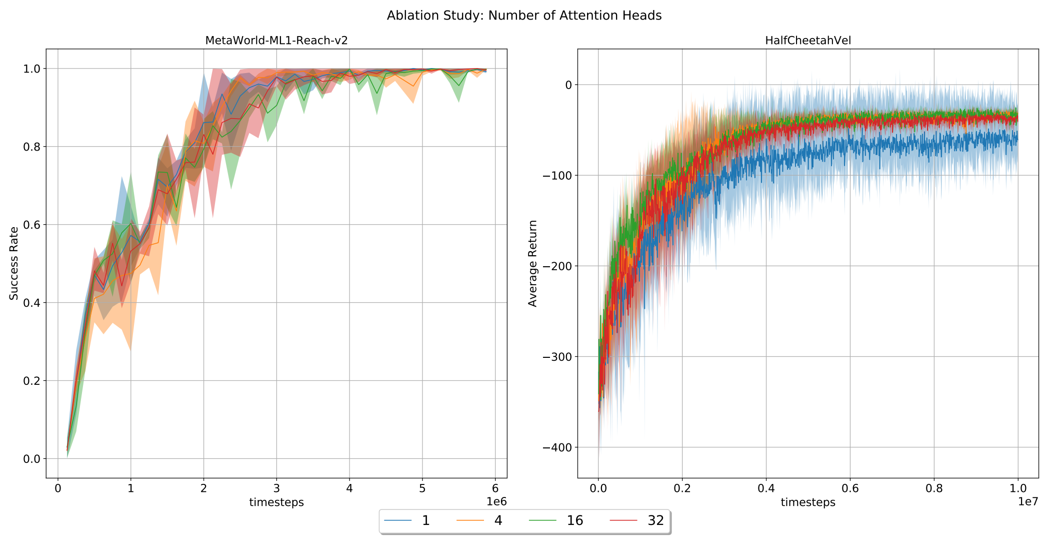

Figure 13: Ablation study for the number of attention heads.

E PROOF OF THEOREM 1

Theorem 1. *Let $\mathcal{S}^l = (\boldsymbol{e}_0^l, \ldots, \boldsymbol{e}_N^l) \sim p(\boldsymbol{e}|\mathcal{S}^l, \boldsymbol{\theta}_l)$ 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: $\mathcal{S}_V^l = (\boldsymbol{e}_0^{l,V}, \ldots, \boldsymbol{e}_N^{l,V})$, 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:

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

□

F 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 $\pi$
  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
```