History Compression via Language Models in Reinforcement Learning

Fabian Paischer [1 2 *]   **Thomas Adler** [1 *]   **Vihang Patil** [1]   **Angela Bitto-Nemling** [1 2 3]   **Markus Holzleitner** [1]
Sebastian Lehner [1 2]   **Hamid Eghbal-zadeh** [1]   **Sepp Hochreiter** [1 2 3]

Abstract

In a partially observable Markov decision process (POMDP), an agent typically uses a representation of the past to approximate the underlying MDP. We propose to utilize a frozen Pretrained Language Transformer (PLT) for history representation and compression to improve sample efficiency. To avoid training of the Transformer, we introduce *FrozenHopfield*, which automatically associates observations with pretrained token embeddings. To form these associations, a modern Hopfield network stores these token embeddings, which are retrieved by queries that are obtained by a random but fixed projection of observations. Our new method, HELM, enables actor-critic network architectures that contain a pretrained language Transformer for history representation as a memory module. Since a representation of the past need not be learned, HELM is much more sample efficient than competitors. On Minigrid and Procgen environments HELM achieves new state-of-the-art results. Our code is available at https://github.com/ml-jku/helm.

1. Introduction

Recently, reinforcement learning (RL) for partially observable Markov decision processes (POMDPs; Åström, 1964; Kaelbling et al., 1998) became popular and attracted a lot of attention through mastering complex games such as DotA (Berner et al., 2019) or Starcraft (Vinyals et al., 2019). In POMDPs, only observations are available but the state of the underlying Markov decision process (MDP) is not known. To approximate the state most methods use the current observation together with a memory mechanism, such as an

LSTM (Hochreiter & Schmidhuber, 1997), that stores past observations. The memory stores only abstractions of the raw observations to allow for generalization and effective processing. Abstractions are also used in human language when describing objects, their mutual relations, events, situations, concepts and ideas. Human language is tailored to represent and compress the past via abstraction to convey past experiences and events from one generation to the next. History compression only stores observations that are relevant, i. e., contain new information (Schmidhuber, 1992). Abstraction is a form of compression that maps segments of constituent information to a single segment of information (Chaitin, 2006; Gell-Mann, 1995). During evolution, human language has been optimized with respect to compression and abstraction for being effective in storing past experiences of one human to pass them on to other humans (Nowak & Krakauer, 1999). Thus, language is inherently well-suited for constructing abstractions and representing the past. Language models are usually scaled up and pretrained on large-scale text corpora. During pretraining, language models learn to compress information to represent abstract concepts prevalent in natural language. This motivates us to leverage pretrained language models for history compression.

The Transformer (Vaswani et al., 2017) proved to be very effective for language modeling (Brown et al., 2020; Dai et al., 2019). Further, it has been successfully applied for sequence modeling in the domain of RL (Janner et al., 2021; Chen et al., 2021). Therefore, it is very well suited to represent and store past observations. The gated TransformerXL (Parisotto et al., 2020) is a Transformer architecture tailored to on-policy reinforcement learning and trained from scratch. However, it tends to overfit on the usually scarce data generated by the current policy. The same problem appears when finetuning a large pretrained Transformer model. To avoid these overfitting problems, we use a Transformer pretrained on a large text corpora without finetuning or weight adjustments. To enable the usage of a fixed transformer, we propose our new *FrozenHopfield* that maps observations and actions to the input space of the Transformer, without any training, by associating them with token embeddings. These associations are obtained by first randomly projecting observations and actions to state patterns (queries) and then using

---
*Equal contribution [1]LIT AI Lab, Institute for Machine Learning, Johannes Kepler University Linz, Austria [2]ELLIS Unit Linz [3]Institute of Advanced Research in Artificial Intelligence (IARAI), Vienna, Austria. Correspondence to: Fabian Paischer <paischer@ml.jku.at>.

them to retrieve token embeddings from a modern Hopfield network (MHN, Ramsauer et al., 2021). The MHN stores the pretrained token embeddings, therefore their retrievals are in the simplex spanned by these token embeddings. We propose HELM (History comprEssion via Language Models) for learning a policy by any RL algorithm like, e.g., Proximal Policy Optimization (PPO; Schulman et al., 2017) in POMDPs. HELM uses novel policy and value network architectures that include a pretrained Transformer for abstract history representation and compression. If the current state can be predicted from the current observation and the history, then HELM even transforms the POMDP into an MDP. Since HELM does not require learning a history representation, it is much more sample efficient than previous methods.

Our contributions are as follows:

- We introduce *FrozenHopfield*, which automatically associates actions and observations with original token embeddings of the Transformer without training or weight adjustments while still being theoretically sound.

- We introduce HELM, a framework for actor-critic network architectures with a pretrained language Transformer for abstract history representation as a memory module.

2. History Compression via Language Models

Our target is to leverage a Pretrained Language Transformer without any finetuning in the highly sensitive setting of on-policy RL for partially observable environments. This raises two problems: (i) the problem of finding a suitable translation from environment observations to the language domain, and (ii) how to effectively utilize the compressed history for policy optimization. To address (i), we introduce *Frozen-Hopfield*, a frozen associative memory based on random projections as queries that map from any modality other than language to a token representation. We feed sequences of projected environment observations into the PLT and retrieve a compressed representation of the current episode history. To tackle (ii), we propose a simple framework in which the representation of the past is concatenated with a learned encoding of the current observation via a convolutional neural network (CNN; Fukushima, 1980; LeCun et al., 1989), and serves as input to the policy optimization algorithm. The actor-critic networks consist of shallow multi-layer perceptrons which are, along with the CNN encoder, the only trainable components in our proposed setting. Thus, we leverage the ability to abstract information learned during pretraining of PLTs for partially observable environments in RL.

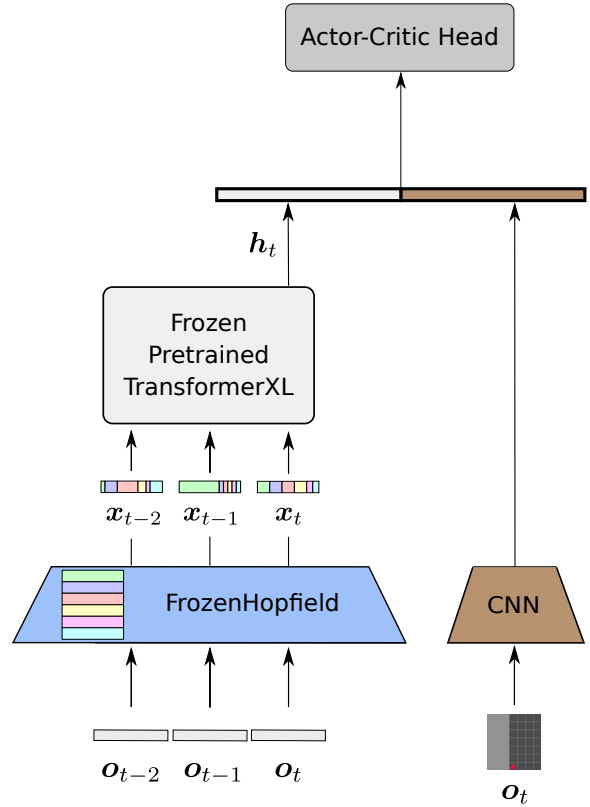Our proposed framework, HELM, consists of three main

Figure 1: Architecture of HELM: The sequence of observations is encoded via the *FrozenHopfield* mechanism and fed into the TransformerXL. The current timestep is encoded with a CNN and concatenated with the output of the language model to form the input for the actor-critic head.

$$x_t^\top = \sigma\left(\beta\ o_t^\top\ P^\top\ E^\top\right)E$$

Figure 2: The *FrozenHopfield* mechanism is realized as a random but fixed retrieval from a MHN. The current observation o_t forms the query and the pretrained token embeddings of the language model E are used as stored patterns. The random projection P is leveraged for projecting the observations to the same dimensionality as the token embeddings. β is the inverse temperature that determines how close the retrievals are to the stored tokens. The retrieval x_t is located in the simplex spanned by the rows of E.

parts: (i) the memory mechanism which utilizes a pretrained language model, (ii) an encoder of the current timestep, and

(iii) the actor-critic networks trained for policy optimization. We instantiate (i) with a TransformerXL (TrXL; Dai et al., 2019). For (ii) we utilize a CNN based on the small Impala architecture (Espeholt et al., 2018) since we mainly focus on image-based environments. Finally, we utilize a MLP with one hidden layer for (iii). Figure 1 shows a graphical illustration of HELM. Pseudocode can be found in Section 2.

We use a TrXL that is pretrained via Causal Language Modeling (CLM) based on next-token prediction. Thus, during pretraining the information of an entire trajectory is compressed in the most recent token, based on which the next token is predicted. TrXL uses relative positional encodings and provides a memory register which stores past activations and allows processing sequences beyond a fixed context length. This property is beneficial in the context of RL where the length of each episode is unknown a-priori during inference. Also leveraging the memory register drastically reduces computational complexity. Each time a new observation is processed, its corresponding activations are shifted into the memory register, which allows feeding a single observation at a time into the model while still preserving the context up to a certain length. We use the huggingface implementation (Wolf et al., 2020) of TrXL consisting of 257M parameters (18 hidden layers, 1024 hidden units per layer, 16 attention heads), which is pretrained on the WikiText 103 dataset (Merity et al., 2017) . We found TrXL to strike a reasonable balance between complexity and performance.

In order to utilize the Transformer for RL we aim to find a suitable input representation for each environment observation in the Transformer embedding space. Finding an efficient mapping from, e. g., images to the language domain is not straightforward and usually involves training. A common approach is to learn an input projection either via a linear layer (Lu et al., 2021) or a convolutional neural network (Tsimpoukelli et al., 2021). However this requires back-propagating gradients through the entire language model which is computationally expensive. To this end, we introduce *FrozenHopfield*, a random but fixed Hopfield retrieval mechanism, which allows mapping inputs from any modality to the language domain by utilization of pretrained token embeddings. The *FrozenHopfield* mechanism consists of two essential steps:

1. Random projections of observations (and possibly actions) to the Transformer token embedding space

2. Retrievals from the MHN that stores the original token embeddings and uses the projections as state patterns or queries

The first step is justified by the Johnson-Lindenstrauss (JL) lemma (Johnson & Lindenstrauss, 1984), which guarantees approximate preservation of the mutual distances of the observations after a random projection with high probability.

The second step ensures that after retrieval each observation is represented within the convex hull of the Transformer embeddings. In combination these two steps associate observations from the environment with embeddings from the language domain of the pretrained Transformer without any training. We found that this mechanism provides a cheap and effective way to map observations to a suitable input representation for the Transformer. It even yields significantly better results in terms of sample efficiency as opposed to learning an input projection (see Appendix D). Figure 2 shows a graphical illustration of the *FrozenHopfield* mechanism, which we describe more formally in the following.

Let $\boldsymbol{E} = (\boldsymbol{e}_1, \ldots, \boldsymbol{e}_k)^\top \in \mathbb{R}^{k \times m}$ be the token embedding matrix from the pretrained Transformer consisting of k embeddings $\boldsymbol{e}_i \in \mathbb{R}^m$. At time t, we obtain inputs $\boldsymbol{x}_t \in \mathbb{R}^m$ for the Transformer from observations $\boldsymbol{o}_t \in \mathbb{R}^n$ via the *FrozenHopfield* mechanism by

$$\boldsymbol{x}_t^\top = \sigma(\beta \boldsymbol{o}_t^\top \boldsymbol{P}^\top \boldsymbol{E}^\top) \boldsymbol{E}, \tag{1}$$

where σ is the softmax function. Further, $\boldsymbol{P} \in \mathbb{R}^{m \times n}$ is a random projection matrix whose entries are independently sampled from a $\mathcal{N}(0, n/m)$ Gaussian distribution.

The variance n/m is a consequence of the JL lemma, which implies the following. Given two observations $\boldsymbol{o}_t, \boldsymbol{o}_t'$, let $\boldsymbol{d} = \boldsymbol{o}_t - \boldsymbol{o}_t'$. For any $0 < \varepsilon < 1$ with probability at least $1 - \delta$ we have

$$(1 - \varepsilon)\|\boldsymbol{d}\|_2^2 \leq \|\boldsymbol{P}\boldsymbol{d}\|_2^2 \leq (1 + \varepsilon)\|\boldsymbol{d}\|_2^2, \tag{2}$$

where $\|\cdot\|_2$ is the Euclidean norm and

$$\delta = 2\exp\left(-\frac{m(\varepsilon^2/2 - \varepsilon^3/3)}{2}\right). \tag{3}$$

This means that the projection $\boldsymbol{P}$ is approximately distance preserving with high probability. Details can be found in Appendix E.

Equation (1) is the Hopfield retrieval (Ramsauer et al., 2021), where $\boldsymbol{P}\boldsymbol{o}_t$ are the state patterns or queries and $\boldsymbol{E}$ are the stored patterns. Equivalently, it can be viewed as attention mechanism with query $\boldsymbol{P}\boldsymbol{o}_t$, and keys and values $\boldsymbol{E}$. See Appendix F for details. Since the inputs $\boldsymbol{x}_t$ are in the simplex spanned by the stored patterns $\boldsymbol{E}$, it is guaranteed that $\boldsymbol{x}_t$ lies in the convex hull of the set $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_k\}$. Thus, we feed the transformer with inputs from the regime it was trained on. The inverse temperature $\beta > 0$ of the MHN is a hyperparameter of our method and controls how close the retrievals are to the original token embeddings. For $\beta \to 0$, the mean $\frac{1}{k} \sum_{i=1}^{k} \boldsymbol{e}_i$ is retrieved, independent of the query. Whereas for $\beta \to \infty$, the original token embedding $\boldsymbol{e}_i$ is retrieved that is closest to the mapped $\boldsymbol{o}_t$, that is, $i = \arg\max_{i \in \{1, \ldots, k\}} \boldsymbol{e}_i^\top \boldsymbol{P}\boldsymbol{o}_t$.

The representation of the current timestep is obtained via the CNN encoder. We specify the output of the encoder

to be of the same dimensionality as the output of TrXL. The representation of the current timestep and the context representation h_t are concatenated and serve as input to the actor-critic head. In order to enhance the reliance of the policy algorithm on the Transformer output, we construct the actor-critic networks as shallow as possible with only a single hidden layer. In principle, any algorithm for policy optimization may be used. In our experiments, we focus on PPO with multistep experience, clipped importance sampling ratio (Schulman et al., 2017), and generalized advantage estimation (Schulman et al., 2016). We use the `stable-baselines3` (Raffin et al., 2019) package for the implementation of our algorithm.

Algorithm 1 HELM

Require: Interaction budget N, Horizon T, Actor-critic head π, Environment `env`, RL algorithm `update`
  $\mathcal{B} \leftarrow \emptyset$ {Initialize buffer}
  for $i = 1$ **to** N **do**
    $o_1 \leftarrow$ `env.reset()`
    for $t = 1$ **to** T **do**
      $x_t \leftarrow$ FrozenHopfield(o_t) {Retrieve associations}
      $h_t \leftarrow$ TrXL(x_t) {Compress history}
      $\tilde{s}_t \leftarrow [h_t; \text{CNN}(o_t)]$ {Concatenate}
      $a_t \sim \pi(\tilde{s}_t)$ {Sample action}
      $o_{t+1}, r_{t+1} \leftarrow$ `env.step`(a_t) {Interact}
      $\mathcal{B} \leftarrow [\mathcal{B}; (o_t, h_t, a_t, r_{t+1})]$ {Extend buffer}
    end for
    `update`$(\text{CNN}, \pi, \mathcal{B})$
  end for

Algorithm 1: History Compression via HELM. Observations are mapped to the language domain and processed by TrXL to obtain a compressed version of the history. Actor-critic networks are trained on the history representations concatenated with an encoding of the current timestep. TrXL can be exchanged with any other pretrained language model, and actor-critic are initialized depending on the RL algorithm. For on-policy RL the buffer $\mathcal{B}$ corresponds to a rollout buffer, while for off-policy RL it is initialized as replay buffer (Lin, 1992). The CNN encoder may be exchanged with any suitable encoder for the current observation.

3. Experiments

We conduct experiments on partially observable environments to exploit the compressed history abstraction of the PLT. Furthermore, we train on procedurally generated environments which enhance diversity and force the agent to learn generalizable skills by sampling level configurations from a predefined distribution. As toytasks we generate a RandomMaze environment (Zuo, 2018), select a

memory-dependent environment from Minigrid (KeyCorridor, Chevalier-Boisvert et al., 2018), and evaluate our agents on complex environments from the Procgen suite (Cobbe et al., 2020).

HELM is compared to a recurrent baseline trained from scratch and evaluated for sample efficiency. As baseline we take the small Impala architecture (Impala-PPO, Espeholt et al., 2018), which consists of a convolutional backbone and an LSTM, since it reaches state-of-the-art performance on Procgen environments. To measure the dependence on memory we also include the performance of a Markovian baseline (CNN-PPO) which consists of the same architecture as Impala-PPO without the LSTM. In accordance with recent work (Jiang et al., 2021; Goyal et al., 2021; Madan et al., 2021), we use PPO as on-policy algorithm for our Minigrid experiments. For Procgen we show performance for PPO as in (Cobbe et al., 2020). Additionally, we provide results for the recently proposed phasic policy gradient (PPG) algorithm (Cobbe et al., 2021) for all methods on the 6 Procgen environments in Appendix C. We evaluate for sample efficiency by measuring the performance at the end of training and test for statistical significance via a one-sided Wilcoxon rank-sum test (Wilcoxon, 1945) at a confidence level of $\alpha = 0.05\,\%$. The performance is evaluated by measuring the interquartile mean (IQM) and 95 % bootstrapped confidence intervals (CIs), as proposed in Agarwal et al. (2021).

The best hyperparameters for each method are determined via gridsearches, while others were set to fixed values to trade off speed and complexity. For the TrXL in HELM, we choose a memory register length of 512 which includes the current timestep. The input to the *FrozenHopfield* component consists of flattened grayscaled observations. Since there is no need to backpropagate through the Transformer, only its hidden representations for each timestep are stored in the rollout buffer. This alleviates the need to process entire trajectories for computing the gradients. The CNN encoder of HELM, as well as CNN-PPO and Impala-PPO are trained on RGB images. A hyperparameter search is performed for all methods over a small grid (see Appendix B).

3.1. RandomMaze

RandomMaze provides a testbed for evaluating the memory capabilities of our methods. The agent is represented as a tile in a maze of randomly sampled size and receives a 9×9 agent centered excerpt of the state to enforce partial observability. A reward of -1 is elicited when bumping against a wall, which ends the current episode. Furthermore, at each interaction step a small negative reward of -0.01 is given. If the agent reaches the goal tile, the episode ends and a positive reward of 1 is elicited. More details on the RandomMaze environment can be found in Appendix A.1.

Table 1: IQM and 95 % bootstrapped CIs of the return over the last 100 episodes across 30 seeds on RandomMaze after 2M steps. Bold indicates maximum reached return.

HELM	IMPALA-PPO	CNN-PPO
$\mathbf{0.185} \pm 0.04$	0.05 ± 0.06	0.16 ± 0.035

To illustrate the dependence on memory we additionally train a Markovian baseline (CNN-PPO) which features the same CNN encoder as HELM, and the actor-critic head, but no memory mechanism to store past events. Table 1 shows the performance of our models. We train for 2M interaction steps and evaluate for sample efficiency at the end of training. The CNN-PPO agent performs surprisingly well under the partial observability in RandomMaze. HELM achieves the highest score indicating its advantage over CNN-PPO due to its history compression component. We find that Impala-PPO reaches a significantly lower score ($p = 1e-4$) than HELM which highlights the superior sample efficiency of HELM. This experiment also demonstrates that raw observations may be of greater or smaller dimensionality than the hidden dimension of TrXL before being processed with the FrozenHopfield component.

3.2. Minigrid

Minigrid environments provide a variety of tasks which are designed to expose weaknesses of current RL algorithms. A sparse reward of 1 is elicited after the agent successfully completes the given task. For each interaction step a small value is subtracted from the reward, thus, the optimal score is close to 1. Furthermore, Minigrid environments are turned into POMDPs by restricting the field of view of the agent. In the KeyCorridor environment the agent must first navigate to the left to pick up a key, and then navigate back to the right to unlock a door and pick up an object. In order to solve the task the agent must effectively represent various entities in its hidden representation and remember that it has picked up a key prior to an attempt of opening the door.

We limit our budget of interaction steps with the environment to 500K and show performance across 30 different seeds. We observe that HELM significantly outperforms Impala-PPO ($p = 2.4e-7$; see Figure 3). This indicates that it successfully compresses the picked up key in its memory representation, which allows solving the environment within fewer interaction steps. Surprisingly, CNN-PPO is also capable of solving the environment in single runs, however exhibits large variance across multiple runs. We provide results on 5 more partially observable Minigrid environments in the Appendix A.2 and find that most of these environments are actually solvable with a Markovian policy. Still, HELM significantly outperforms CNN-PPO ($p = 0.035$) and Impala-PPO (see Appendix A.2). This highlights the

capabilities of HELM to, provide rich history representations that result in enhanced sample efficiency and improved performance even in environments where no memory component is required.
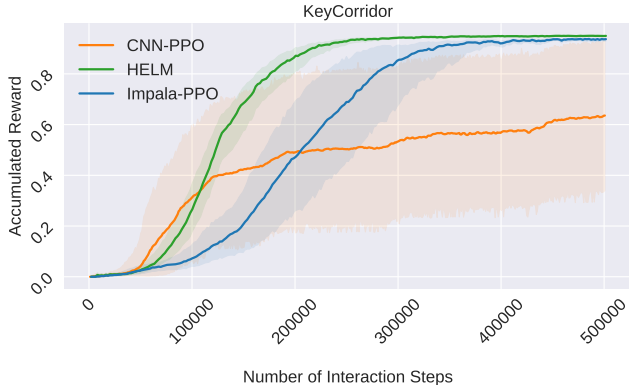

Figure 3: IQM and 95 % bootstrapped CIs over 100 episodes for HELM, Impala-PPO, and CNN-PPO on the KeyCorridor environment. Performance across 30 seeds is shown.

3.3. Procgen

Procgen provides a mode for evaluating the memory mechanism of an agent as well as sample efficiency. Out of a total of 16 environments, six support the *memory* distribution mode, namely caveflyer, dodgeball, heist, jumper, maze, and miner. The problems posed in the *memory* mode are particularly challenging to solve since either observations are cropped and agent centered, or the level generation logic is altered. While Procgen provides dense rewards, it is critical that history representations are invariant to spurious correlations introduced by human-designed backgrounds. The maximum score for each Procgen environment in the *memory* mode can be found in Appendix A.3.

The budget of interaction steps for Procgen is limited to 25M steps and we train on the entire level distribution across 10 seeds to evaluate for sample efficiency. The resulting learning curves are shown in Figure 6. HELM significantly outperforms Impala-PPO in 4 out of 6 environments, namely miner ($p = 2e-4$), caveflyer ($p = 0.03$), jumper ($p = 0.001$), and heist ($p = 2.4e-4$). Although not significant ($p = 0.052$), HELM still shows superior sample efficiency on maze. We can confirm the findings of Cobbe et al. (2020) that training an LSTM on the Procgen environments can lead to instabilities in certain environments (heist for *hard* mode). This effect is exacerbated in the *memory* mode and affects more environments (jumper, heist, caveflyer) since environments are larger and partially observable. To compare HELM to a more stable baseline we add results for the CNN-PPO agent. While CNN-PPO is more stable

in jumper it still exhibits instabilities in heist and caveflyer. HELM however performs more stable in these environments. Remarkably, the CNN-PPO agent significantly outperforms both HELM ($p = 6.5\text{e-}4$) and Impala-PPO ($p = 0.003$) in the dodgeball environment. This is due to the fact that none of the agents reached the stage of the game in which memory is actually required (finding the way back to the goal tile) and only need to perform reactive tasks (shooting enemies that spawn next to it and dodge incoming shots). In most other environments the addition of a memory component results in significant improvements of HELM over CNN-PPO ($p = 5.4\text{e-}3$ for heist, $p = 0.016$ for caveflyer, $p = 3.8\text{e-}4$ for miner, $p = 6.2\text{e-}4$ for maze).

Figure 4 shows the normalized return for each environment at the end of training. The upper bound for the normalization range is determined by the maximum possible score for each environment. The lower bound is determined by training Impala-PPO on fully masked observations to determine what score is trivially reachable. Remarkably on jumper and heist, both Impala-PPO and CNN-PPO even exhibit a negative normalized return. Additionally, CNN-PPO reaches a negative normalized return on caveflyer. A possible reason for that might be the presence of spurious correlations in the backgrounds and the need for longer training to deal with them accordingly. When observations are masked out the agent solely relies on positional information and is not affected by noise in the observations which leads to improved performance over CNN-PPO and Impala-PPO within the 25M interaction steps. Nevertheless, HELM manages to progress in heist and reaches a positive normalized return on jumper which highlights the stability of HELM over the baselines on particularly hard environments.

vironments. Although both methods show high variance due to low scores in particularly hard environments, like jumper, HELM significantly outperforms Impala-PPO ($p = 0.022$) while showing improved sample efficiency compared to CNN-PPO ($p = 0.058$). We include a comparison to the LSTM baseline trained by Cobbe et al. (2020) on distribution mode *hard* (LSTM-PPO). In the *hard* mode the state of the environment is fully observable, thus, one would suspect the performance of a recurrent agent trained on the *hard* mode to significantly exceed the performance of an agent trained on the *memory* mode. Surprisingly, we observed that LSTM-PPO performs on-par with HELM ($p = 0.61$). Furthermore, HELM achieves the highest mean normalized IQM across all Procgen environments on the *memory* mode.

Figure 5: Mean IQM and 95 % bootstrapped CIs of normalized return across the six Procgen environments over 10 seeds. LSTM-PPO refers to the benchmark results from Cobbe et al. (2020) on distribution mode *hard*, which provides fully observable states. HELM significantly outperforms Impala-PPO.

3.4. Ablation Studies

We compare HELM to:

- Training a linear projection as input to the Transformer without *FrozenHopfield* (HELM-TrainInput)

- Training the linear projection within *FrozenHopfield* (HELM-TrainFH)

- The Frozen Pretrained Transformer (FPT) setting analogous to Lu et al. (2021) by training an input projection, and finetuning layer norm parameters (Ba et al., 2016), and the actor-critic networks (HELM-FPT)
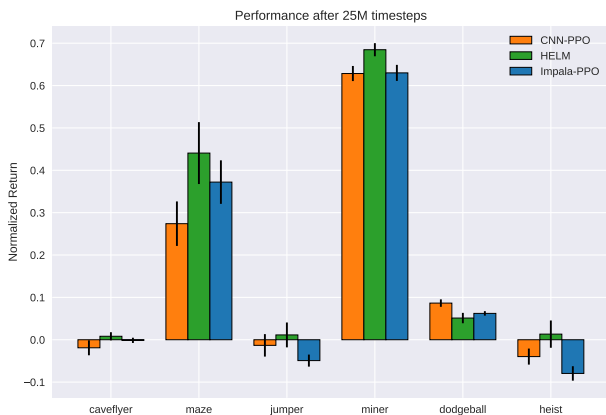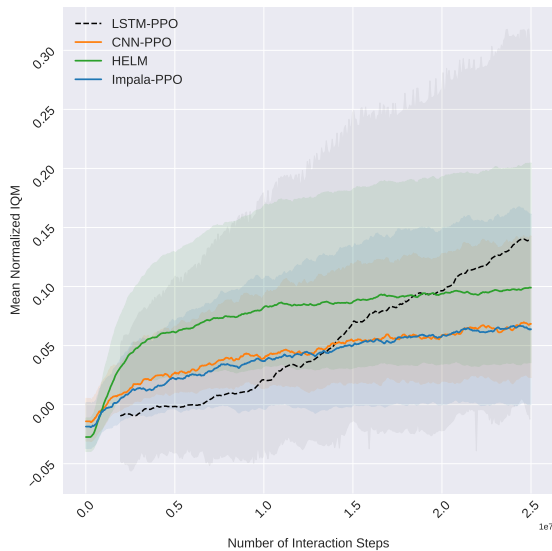

Figure 4: IQM and 95 % bootstrapped CIs of the normalized return across 10 seeds over the last 100 episodes for all Procgen environments. Returns are normalized according to ranges in Appendix A.3.

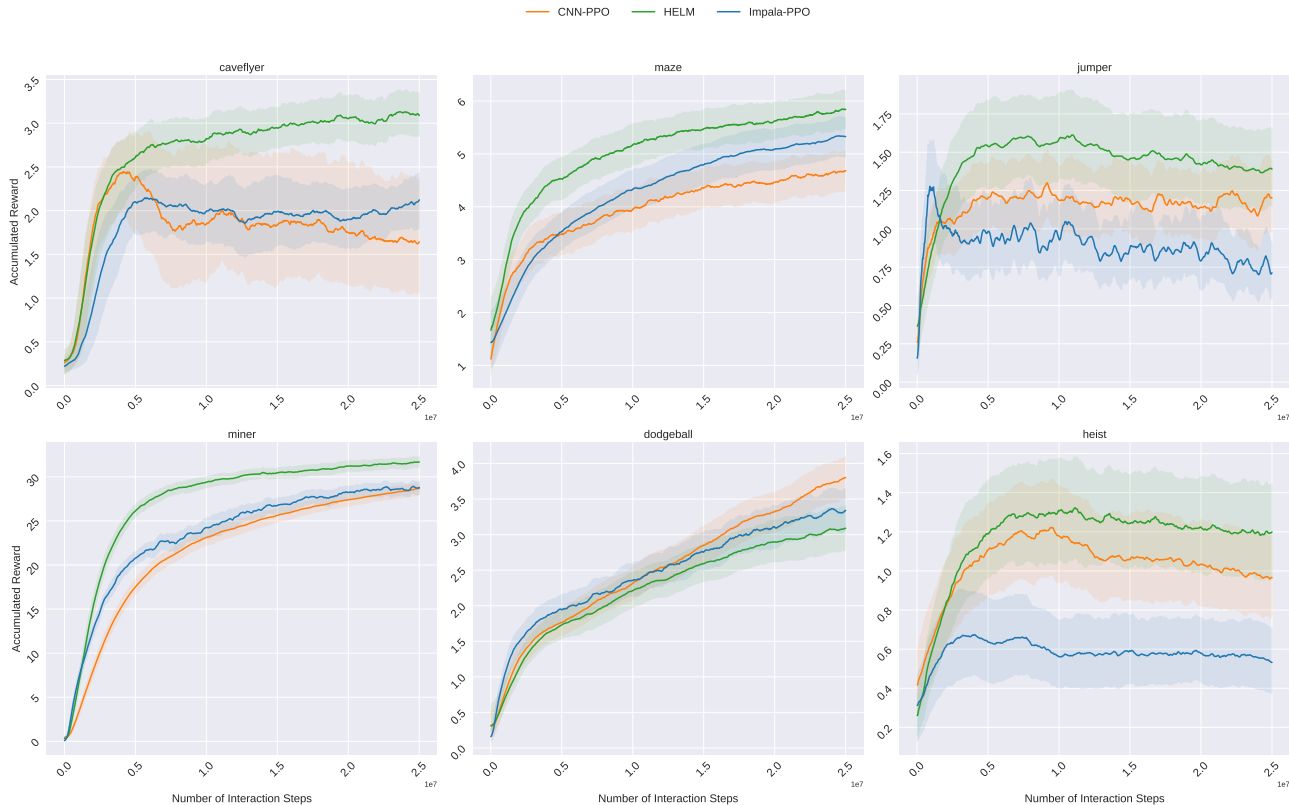We plot the mean IQM (see Figure 5) across all 6 Procgen en-

Figure 6: IQM and 95 % confidence intervals of the return over 100 episodes across 10 seeds for HELM and Impala-PPO on Procgen environments caveflyer, dodgeball, jumper, maze, miner, and heist for distribution mode *memory*.

- Training an input projection and fully finetuning TrXL (HELM-Finetuned)

Since Rothermel et al. (2021) found the learning rate to be critical for finetuning PLTs for cross-modal transfer, we run all aforementioned experiments with three different values for the learning rate used to finetune the memory mechanism (see Appendix D). The learning rate for the remaining trainable parameters is set to the best value found by the hyperparameter search for HELM (see Appendix B). Furthermore, we include a comparison to the gated TransformerXL (GTrXL, Parisotto & Salakhutdinov, 2021). For a fair comparison we initialize GTrXL to approximately match the number of parameters and the hidden dimensionality of TrXL and perform a hyperparameter search (Appendix B). The best learning rate for each setting is chosen and compared to HELM on the KeyCorridor environment (Figure 7). Additionally, we show the performance of all aforementioned variants on the Procgen environment maze for different learning rates (see Appendix D).

Remarkably, HELM significantly ($p = 5.1\text{e-}4$) outperforms all aforementioned methods in terms of sample efficiency after training for 500K timesteps. GTrXL fails to make

any significant progress in the environment. This might be explained by the difficulty of training or finetuning in the low-data regime of on-policy RL since a distribution shift occurs every time the policy is updated. Previous work (Zhang et al., 2021; Mosbach et al., 2021) highlighted problems with finetuning PLTs in low-data regimes and proposed longer training times as remedy. The same applies to GTrXL, since our budget of interaction steps with the environment is much smaller than the one in Parisotto & Salakhutdinov (2021). Furthermore, we observe that fully finetuning the PLT (HELM-Finetuned) is more sample efficient than retraining an input projection (HELM-TrainInput), or minimal finetuning (HELM-FPT), which is in line with findings of (Rothermel et al., 2021). However, we observe that a frozen PLT already provides compact history representations which enables solving the KeyCorridor environment with fewer interaction steps.

We conduct additional ablation studies on exchanging TrXL with LSTM models pretrained on language modeling and dynamics modeling in the KeyCorridor environment. We find that HELM even yields performance on-par with an agent that uses a dynamics model pretrained in the KeyCorridor environment. Further we provide compelling evidence
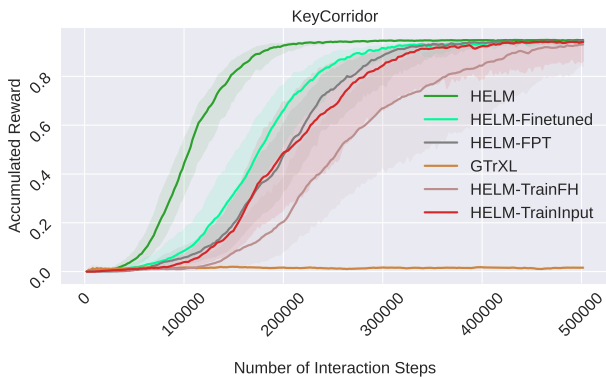
Figure 7: Comparison of HELM to various finetuning techniques and a GTrXL trained from scratch. HELM significantly outperforms all competitors. IQM and 95 % bootstrapped confidence interval of the average return over 100 episodes across 30 seeds are shown.

on the importance of transferring from the language domain. The results for our ablation studies can be found in Appendix D.

3.5. Analysis of representations

We conduct a qualitative analysis on the abstraction obtained by the *FrozenHopfield mechanism* and the history representation obtained by the PLT. First, we verify the distance preserving property of the *FrozenHopfield* component by comparing the distances of observations across different environments before and after being processed. In this regard we sample observations with a random policy from two Minigrid environments, namely KeyCorridor and DynamicObstacles and measure their distances in observation space and in the token embedding space at the output of the *FrozenHopfield* component for different values of β (Figure 21). We find that the parameter β is crucial to maintain the distance preserving property since it controls the amount of dispersion of the embeddings in the token space. Even when distances are not entirely preserved after the linear projection, they can still be enhanced by setting β to larger values.

Furthermore, the attention maps of the PLT show that in certain attention heads the Transformer model attends to observations that correspond to important events for solving the task at hand. Figure 8 shows the attention maps of TrXL for an episode sampled with a trained policy consisting of 18 interaction steps with the environment. The full episode can be observed in Figure 22. We observe that the PLT reacts to certain changes in its input which correspond to key events in the KeyCorridor environment. For example, at timestep 2 the agent observes the key for the first time,

or at timestep 11 the agent observes the ball for the first time. Moreover, when considering the token annotations in Figure 22, which corresponds to the closest token (i.e. $\beta \to \infty$), we find that observations that share similarity in the input space map to the same tokens, e.g., "recollection" when the agent faces a closed door. Thus, the parameter β controls the amount of clustering in the token embedding space to form abstractions of previously encountered events.
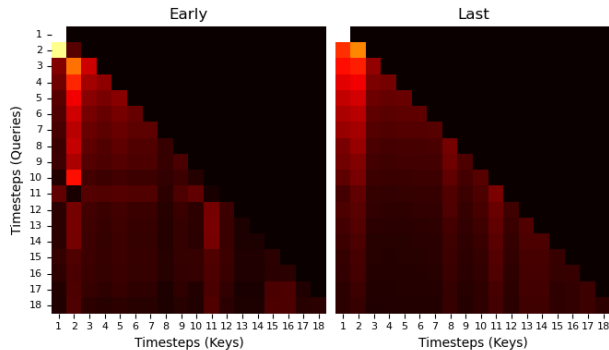

Figure 8: Attention maps of an attention head in an early and one of the last layers of TrXL for an episode sampled with a trained policy. The language model attends to changes in the observations that correspond to important events, e.g., observing the key for the first time (Timestep 2), or observing the ball for the first time (Timestep 11).

4. Related Work

Credit assignment is one of the central problems of RL (Puterman, 1994; Sutton & Barto, 2018). Assigning credit is difficult in the presence of delayed rewards (Sutton, 1984) and partial observability (Cassandra et al., 1994). History compression has been used to address credit assignment (Arjona-Medina et al., 2019; Patil et al., 2020; Holzleitner et al., 2020; Widrich et al., 2021) as well as partial observability (Hausknecht & Stone, 2015; Vinyals et al., 2019). The question of what information to store in a stream of data has been well studied before. Schmidhuber (1992) ignores expected values and only stores unexpected inputs. Continual learning methods like Ruvolo & Eaton (2013) and Schwarz et al. (2018) store model components over a history of tasks. Methods addressing catastrophic forgetting (Kirkpatrick et al., 2016; Zenke et al., 2017) implicitly compress knowledge from previous tasks.

The Transformer architecture (Vaswani et al., 2017) has dominated the field of natural language processing over the past few years (Devlin et al., 2019; Radford et al., 2019; Brown et al., 2020). Recent work adjusted the Transformer architecture for RL (Parisotto et al., 2020; Sukhbaatar et al., 2021; Parisotto & Salakhutdinov, 2021). Pretraining on

large-scale text corpora enables such complex models to abstract relevant knowledge for solving math-word problems (Griffith & Kalita, 2019), symbolic math problems (Noorbakhsh et al., 2021), and even university-level math exercises (Drori et al., 2021). As shown by Petroni et al. (2019); Talmor et al. (2020); Kassner et al. (2020), PLTs can learn abstract symbolic rules and perform reasoning.

Recently, it was shown that PLTs even transfer between languages or across modalities. Frozen PLTs have been commonly used to extract relevant knowledge about a downstream task (Li & Liang, 2021; Lester et al., 2021). Tsimpoukelli et al. (2021) learn prompts out of images to extract general knowledge obtained during pretraining in the context of cross-domain few-shot learning (Adler et al., 2020). Artetxe et al. (2020) and Minixhofer et al. (2021) transfer a monolingual model to different target languages via finetuning. Orhan (2021) shows that pretraining on foreign source language corpora can improve the performance on a target language. Lu et al. (2021) and Rothermel et al. (2021) found that pretraining on natural language is beneficial for certain cross-domain downstream tasks.

Pretrained language models have also been transferred to RL. In text-based games, PLTs have been used for enhancing exploration (Yao et al., 2020). Li et al. (2022) use a PLT as initialization to learn a policy given a collection of expert data. Further, PLTs have been used for planning via prompting (Huang et al., 2022), creating abstractions of human language instructions (Hill et al., 2021), using language instructions (Mu et al., 2022) or pretrained multimodal models (Tam et al., 2022) for abstraction to enhance exploration, and in the offline RL setting for sequence modeling (Reid et al., 2022). In contrast to these works, we perform cross-modal transfer of a PLT to on-policy RL with scarce data and sparse rewards for history compression.

FrozenHopfield is an instance of a MHN (Ramsauer et al., 2021; Fürst et al., 2021; Schäfl et al., 2022) that can associate image observations (or any other input) with pretrained token embeddings (Mikolov et al., 2013; Wu et al., 2016; Sennrich et al., 2016). MHNs have already been successfully applied to immune repertoire classification (Widrich et al., 2020), and chemical reaction prediction (Seidl et al., 2022). The JL lemma (Johnson & Lindenstrauss, 1984) has been used in the context of masked language modeling (Wang et al., 2020) to reduce the number of tokens within a sequence. In contrast, *FrozenHopfield* uses a random projection to map observations into the Transformer embedding space.

Our work also draws connections to the reservoir computing theory (Jaeger, 2001; Maass et al., 2002a;b). A reservoir, e.g., a recurrent neural network, is usually initialized at random and held fixed while only a shallow readout layer is optimized for the task at hand.

5. Conclusion

Partial observability in RL requires an agent to store past information in order to approximate the true underlying state of the environment. Language is inherently well suited to construct abstractions and efficiently convey information. We propose HELM, a novel framework that leverages pretrained language models to compress history information for actor-critic network architectures. To translate observations to the language domain we introduced *FrozenHopfield*, a mechanism that acts as an associative memory and transforms a given observation to input tokens for the language model. On memory-dependent toytasks HELM outperforms an LSTM agent trained from scratch. Further, we observe consistent improvements when applying HELM to more complex environments of the Procgen benchmark suite for which we set a new state of the art. Remarkably, HELM outperforms training a complex Transformer model from scratch, as well as finetuning different components of the memory mechanism. HELM also yields comparable performance to an agent that leverages a pretrained dynamics model for history compression. Further, HELM provides performance gains when applied to fully observable MDPs, highlighting the benefits of history compression in the language domain. The generality of HELM allows processing of any input modality using any language model as well as any RL algorithm for optimization.

Acknowledgments

References

Abbott, L. F. and Arian, Y. Storage capacity of generalized networks. *Phys. Rev. A*, 36(10):5091–5094, November 1987. doi: 10.1103/PhysRevA.36.5091. Publisher: American Physical Society.

Adler, T., Brandstetter, J., Widrich, M., Mayr, A., Kreil, D. P., Kopp, M., Klambauer, G., and Hochreiter, S. Cross-Domain Few-Shot Learning by Representation Fusion. *CoRR*, abs/2010.06498, 2020. arXiv: 2010.06498.

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., and Bellemare, M. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 34, 2021.

Amari, S.-i. Learning Patterns and Pattern Sequences by Self-Organizing Nets of Threshold Elements. *IEEE Trans. Computers*, 21(11):1197–1206, 1972. doi: 10.1109/T-C.1972.223477.

Arjona-Medina, J. A., Gillhofer, M., Widrich, M., Unterthiner, T., Brandstetter, J., and Hochreiter, S. RUDDER: Return Decomposition for Delayed Rewards. In Wallach, H., Larochelle, H., Beygelzimer, A., Alché-Buc, F. d., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Artetxe, M., Ruder, S., and Yogatama, D. On the Cross-lingual Transferability of Monolingual Representations. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J. R. (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 4623–4637. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.421.

Ba, L. J., Kiros, J. R., and Hinton, G. E. Layer Normalization. *CoRR*, abs/1607.06450, 2016. arXiv: 1607.06450.

Baldi, P. and Venkatesh, S. S. Number of stable points for spin-glasses and neural networks of higher orders. *Phys. Rev. Lett.*, 58(9):913–916, March 1987. doi: 10.1103/PhysRevLett.58.913. Publisher: American Physical Society.

Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., Józefowicz, R., Gray, S., Olsson, C., Pachocki, J., Petrov, M., Pinto, H. P. d. O., Raiman, J., Salimans, T., Schlatter, J., Schneider, J., Sidor, S., Sutskever, I., Tang, J., Wolski, F., and Zhang, S. Dota 2 with Large Scale Deep Reinforcement Learning. *CoRR*, abs/1912.06680, 2019. arXiv: 1912.06680.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., 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., 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. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020.

Caputo, B. and Niemann, H. Storage Capacity of Kernel Associative Memories. In Dorronsoro, J. R. (ed.), *Artificial Neural Networks - ICANN 2002, International Conference, Madrid, Spain, August 28-30, 2002, Proceedings*, volume 2415 of *Lecture Notes in Computer Science*, pp. 51–56. Springer, 2002. doi: 10.1007/3-540-46084-5_9.

Cassandra, A. R., Kaelbling, L. P., and Littman, M. L. Acting Optimally in Partially Observable Stochastic Domains. In Hayes-Roth, B. and Korf, R. E. (eds.), *Proceedings of the 12th National Conference on Artificial Intelligence, Seattle, WA, USA, July 31 - August 4, 1994, Volume 2*, pp. 1023–1028. AAAI Press / The MIT Press, 1994.

Chaitin, G. The limits of reason. *Sci Am*, 294(3):74–81, 2006.

Chen, H. H., Lee, Y. C., Sun, G. Z., Lee, H. Y., Maxwell, T., and Giles, C. L. High Order Correlation Model for Associative Memory. In *AIP Conference Proceedings 151 on Neural Networks for Computing*, pp. 86–99, USA, 1987. American Institute of Physics Inc. event-place: Snowbird, Utah, USA.

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. *CoRR*, abs/2106.01345, 2021. arXiv: 2106.01345.

Chevalier-Boisvert, M., Willems, L., and Pal, S. Minimalistic Gridworld Environment for OpenAI Gym, 2018. Publication Title: GitHub repository.

Cobbe, K., Hesse, C., Hilton, J., and Schulman, J. Leveraging Procedural Generation to Benchmark Reinforcement Learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2048–2056. PMLR, 2020.

Cobbe, K., Hilton, J., Klimov, O., and Schulman, J. Phasic Policy Gradient. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual*

Event, volume 139 of *Proceedings of Machine Learning Research*, pp. 2020–2027. PMLR, 2021.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., and Salakhutdinov, R. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2978–2988, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1285.

Dasgupta, S. and Gupta, A. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Struct. Algorithms*, 22(1):60–65, 2003. doi: 10.1002/rsa.10073.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423.

Drori, I., Tran, S., Wang, R., Cheng, N., Liu, K., Tang, L., Ke, E., Singh, N., Patti, T. L., Lynch, J., Shporer, A., Verma, N., Wu, E., and Strang, G. A Neural Network Solves and Generates Mathematics Problems by Program Synthesis: Calculus, Differential Equations, Linear Algebra, and More. *CoRR*, abs/2112.15594, 2021. arXiv: 2112.15594.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., and Kavukcuoglu, K. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1406–1415. PMLR, 2018.

Fukushima, K. Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological Cybernetics*, 36:193–202, 1980.

Fürst, A., Rumetshofer, E., Tran, V., Ramsauer, H., Tang, F., Lehner, J., Kreil, D. P., Kopp, M., Klambauer, G., Bitto-Nemling, A., and Hochreiter, S. CLOOB: Modern Hopfield Networks with InfoLOOB Outperform CLIP. *CoRR*, abs/2110.11316, 2021. arXiv: 2110.11316.

Gardner, E. Multiconnected neural network models. *Journal of Physics A: Mathematical and General*, 20(11):3453–3464, August 1987. doi: 10.1088/0305-4470/20/11/046. Publisher: IOP Publishing.

Gell-Mann, M. What is complexity? Remarks on simplicity and complexity by the Nobel Prize-winning author of The Quark and the Jaguar. *Complex.*, 1(1):16–19, 1995.

Goyal, A., Lamb, A., Hoffmann, J., Sodhani, S., Levine, S., Bengio, Y., and Schölkopf, B. Recurrent Independent Mechanisms. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Griffith, K. and Kalita, J. Solving Arithmetic Word Problems Automatically Using Transformer and Unambiguous Representations. *CoRR*, abs/1912.00871, 2019. arXiv: 1912.00871.

Hausknecht, M. J. and Stone, P. Deep Recurrent Q-Learning for Partially Observable MDPs. In *2015 AAAI Fall Symposia, Arlington, Virginia, USA, November 12-14, 2015*, pp. 29–37. AAAI Press, 2015.

Hill, F., Tieleman, O., Glehn, T. v., Wong, N., Merzic, H., and Clark, S. Grounded Language Learning Fast and Slow. In *International Conference on Learning Representations*, 2021.

Hochreiter, S. and Schmidhuber, J. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735.

Holzleitner, M., Gruber, L., Arjona-Medina, J. A., Brandstetter, J., and Hochreiter, S. Convergence Proof for Actor-Critic Methods Applied to PPO and RUDDER. *CoRR*, abs/2012.01399, 2020. arXiv: 2012.01399.

Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982. ISSN 0027-8424. doi: 10.1073/pnas.79.8.2554. Publisher: National Academy of Sciences _eprint: https://www.pnas.org/content/79/8/2554.full.pdf.

Hopfield, J. J. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences*, 81(10):3088–3092, 1984. ISSN 0027-8424. doi: 10.1073/pnas.81.10.3088. Publisher: National Academy of Sciences _eprint: https://www.pnas.org/content/81/10/3088.full.pdf.

Horn, D. and Usher, M. Capacities of multiconnected memory models. *Journal de Physique*, 49(3):389–395, 1988. Publisher: Société Française de Physique.

Howard, J. and Ruder, S. Universal Language Model Fine-tuning for Text Classification. In Gurevych, I. and Miyao, Y. (eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018,*

Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers, pp. 328–339. Association for Computational Linguistics, 2018. doi: 10.18653/v1/P18-1031.

Huang, W., Abbeel, P., Pathak, D., and Mordatch, I. Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents. *CoRR*, abs/2201.07207, 2022. arXiv: 2201.07207.

Jaeger, H. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148(34): 13, 2001. Publisher: Bonn.

Janner, M., Li, Q., and Levine, S. Reinforcement Learning as One Big Sequence Modeling Problem. *CoRR*, abs/2106.02039, 2021. arXiv: 2106.02039.

Jiang, M., Grefenstette, E., and Rocktäschel, T. Prioritized Level Replay. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4940–4950. PMLR, 2021.

Johnson, W. B. and Lindenstrauss, J. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics*, 26, 1984.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, 101:99–134, 1998.

Kassner, N., Krojer, B., and Schütze, H. Are Pretrained Language Models Symbolic Reasoners over Knowledge? In Fernández, R. and Linzen, T. (eds.), *Proceedings of the 24th Conference on Computational Natural Language Learning, CoNLL 2020, Online, November 19-20, 2020*, pp. 552–564. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.conll-1.45.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N. C., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796, 2016. arXiv: 1612.00796.

Krotov, D. and Hopfield, J. J. Dense Associative Memory for Pattern Recognition. In Lee, D. D., Sugiyama, M., Luxburg, U. v., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 1172–1180, 2016.

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., and Jackel, L. D. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Comput.*, 1(4):541–551, 1989. doi: 10.1162/neco.1989.1.4.541.

Lester, B., Al-Rfou, R., and Constant, N. The Power of Scale for Parameter-Efficient Prompt Tuning. *CoRR*, abs/2104.08691, 2021. arXiv: 2104.08691.

Li, S., Puig, X., Paxton, C., Du, Y., Wang, C., Fan, L., Chen, T., Huang, D.-A., Akyürek, E., Anandkumar, A., Andreas, J., Mordatch, I., Torralba, A., and Zhu, Y. Pre-Trained Language Models for Interactive Decision-Making. *CoRR*, abs/2202.01771, 2022. arXiv: 2202.01771.

Li, X. L. and Liang, P. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In Zong, C., Xia, F., Li, W., and Navigli, R. (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pp. 4582–4597. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.acl-long.353.

Lin, L. J. Self-Improving Reactive Agents Based On Reinforcement Learning, Planning and Teaching. *Mach. Learn.*, 8:293–321, 1992. doi: 10.1007/BF00992699.

Loshchilov, I. and Hutter, F. Decoupled Weight Decay Regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Lu, K., Grover, A., Abbeel, P., and Mordatch, I. Pretrained Transformers as Universal Computation Engines. *CoRR*, abs/2103.05247, 2021. arXiv: 2103.05247.

Maass, W., Natschläger, T., and Markram, H. A Model for Real-Time Computation in Generic Neural Microcircuits. In Becker, S., Thrun, S., and Obermayer, K. (eds.), *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*, pp. 213–220. MIT Press, 2002a.

Maass, W., Natschläger, T., and Markram, H. Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560, November 2002b. ISSN 0899-7667. doi: 10.1162/089976602760407955.

Madan, K., Ke, N. R., Goyal, A., Schölkopf, B., and Bengio, Y. Fast And Slow Learning Of Recurrent Independent

Mechanisms. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer Sentinel Mixture Models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Merity, S., Keskar, N. S., and Socher, R. Regularizing and Optimizing LSTM Language Models. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Mikolov, T., Yih, W.-t., and Zweig, G. Linguistic Regularities in Continuous Space Word Representations. In Vanderwende, L., III, H. D., and Kirchhoff, K. (eds.), *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pp. 746–751. The Association for Computational Linguistics, 2013.

Minixhofer, B., Paischer, F., and Rekabsaz, N. WECHSEL: Effective initialization of subword embeddings for cross-lingual transfer of monolingual language models. *CoRR*, abs/2112.06598, 2021. arXiv: 2112.06598.

Mosbach, M., Andriushchenko, M., and Klakow, D. On the Stability of Fine-tuning BERT: Misconceptions, Explanations, and Strong Baselines. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Mu, J., Zhong, V., Raileanu, R., Jiang, M., Goodman, N. D., Rocktäschel, T., and Grefenstette, E. Improving Intrinsic Exploration with Language Abstractions. *CoRR*, abs/2202.08938, 2022. arXiv: 2202.08938.

Noorbakhsh, K., Sulaiman, M., Sharifi, M., Roy, K., and Jamshidi, P. Pretrained Language Models are Symbolic Mathematics Solvers too! *CoRR*, abs/2110.03501, 2021. arXiv: 2110.03501.

Nowak, M. A. and Krakauer, D. C. The evolution of language. *Proceedings of the National Academy of Sciences*, 96(14):8028–8033, 1999. doi: 10.1073/pnas.96.14.8028. _eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.96.14.8028.

Olver, F. W., Lozier, D. W., Boisvert, R. F., and Clark, C. W. *NIST Handbook of Mathematical Functions*. Cambridge University Press, USA, 1st edition, 2010.

Orhan, A. E. Compositional generalization in semantic parsing with pretrained transformers. *CoRR*, abs/2109.15101, 2021. arXiv: 2109.15101.

Parisotto, E. and Salakhutdinov, R. R. Efficient Transformers in Reinforcement Learning using Actor-Learner Distillation. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Parisotto, E., Song, F., Rae, J., Pascanu, R., Gulcehre, C., Jayakumar, S., Jaderberg, M., Kaufman, R. L., Clark, A., Noury, S., Botvinick, M., Heess, N., and Hadsell, R. Stabilizing Transformers for Reinforcement Learning. In *International Conference on Machine Learning*, pp. 7487–7498. PMLR, November 2020. ISSN: 2640-3498.

Patil, V. P., Hofmarcher, M., Dinu, M.-C., Dorfer, M., Blies, P. M., Brandstetter, J., Arjona-Medina, J. A., and Hochreiter, S. Align-RUDDER: Learning From Few Demonstrations by Reward Redistribution. *CoRR*, abs/2009.14108, 2020. arXiv: 2009.14108.

Petroni, F., Rocktäschel, T., Riedel, S., Lewis, P. S. H., Bakhtin, A., Wu, Y., and Miller, A. H. Language Models as Knowledge Bases? In Inui, K., Jiang, J., Ng, V., and Wan, X. (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pp. 2463–2473. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1250.

Psaltis, D. and Park, C. H. Nonlinear discriminant functions and associative memories. In *AIP conference Proceedings*, volume 151, pp. 370–375. American Institute of Physics, 1986. Issue: 1.

Puterman, M. L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994. doi: 10.1002/9780470316887.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Raffin, A., Hill, A., Ernestus, M., Gleave, A., Kanervisto, A., and Dormann, N. Stable Baselines3, 2019. Publication Title: GitHub repository.

Raileanu, R. and Rocktäschel, T. RIDE: Rewarding Impact-Driven Exploration for Procedurally-Generated Environments. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Ramsauer, H., Schäfl, B., Lehner, J., Seidl, P., Widrich, M., Gruber, L., Holzleitner, M., Adler, T., Kreil, D., Kopp, M. K., Klambauer, G., Brandstetter, J., and Hochreiter, S. Hopfield Networks is All You Need. In *International Conference on Learning Representations*, 2021.

Reid, M., Yamada, Y., and Gu, S. S. Can Wikipedia Help Offline Reinforcement Learning? *CoRR*, abs/2201.12122, 2022. arXiv: 2201.12122.

Rothermel, D., Li, M., Rocktäschel, T., and Foerster, J. N. Don't Sweep your Learning Rate under the Rug: A Closer Look at Cross-modal Transfer of Pretrained Transformers. *CoRR*, abs/2107.12460, 2021. arXiv: 2107.12460.

Ruvolo, P. and Eaton, E. ELLA: An Efficient Lifelong Learning Algorithm. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pp. 507–515. JMLR.org, 2013.

Schmidhuber, J. Learning Complex, Extended Sequences Using the Principle of History Compression. *Neural Comput.*, 4(2):234–242, 1992. doi: 10.1162/neco.1992.4.2.234.

Schulman, J., Moritz, P., Levine, S., Jordan, M. I., and Abbeel, P. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In Bengio, Y. and Le-Cun, Y. (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347, 2017. arXiv: 1707.06347.

Schwarz, J., Czarnecki, W., Luketina, J., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R., and Hadsell, R. Progress & Compress: A scalable framework for continual learning. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4535–4544. PMLR, 2018.

Schäfl, B., Gruber, L., Bitto-Nemling, A., and Hochreiter, S. Hopular: Modern Hopfield Networks for Tabular Data. *CoRR*, abs/2206.00664, 2022. doi: 10.48550/arXiv.2206.00664. arXiv: 2206.00664.

Seidl, P., Renz, P., Dyubankova, N., Neves, P., Verhoeven, J., Wegner, J. K., Segler, M., Hochreiter, S., and Klambauer, G. Improving Few-and Zero-Shot Reaction Template Prediction Using Modern Hopfield Networks. *Journal of Chemical Information and Modeling*, 2022. Publisher: ACS Publications.

Sennrich, R., Haddow, B., and Birch, A. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016. doi: 10.18653/v1/p16-1162.

Sukhbaatar, S., Ju, D., Poff, S., Roller, S., Szlam, A., Weston, J., and Fan, A. Not All Memories are Created Equal: Learning to Forget by Expiring. In *International Conference on Machine Learning*, pp. 9902–9912. PMLR, July 2021. ISSN: 2640-3498.

Sutton, R. S. *Temporal Credit Assignment in Reinforcement Learning*. PhD Thesis, University of Massachusetts, Dept. of Comp. and Inf. Sci., 1984.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

Talmor, A., Elazar, Y., Goldberg, Y., and Berant, J. oLMpics - On what Language Model Pre-training Captures. *Trans. Assoc. Comput. Linguistics*, 8:743–758, 2020.

Tam, A. C., Rabinowitz, N. C., Lampinen, A. K., Roy, N. A., Chan, S. C. Y., Strouse, D. J., Wang, J. X., Banino, A., and Hill, F. Semantic Exploration from Language Abstractions and Pretrained Representations. *CoRR*, abs/2204.05080, 2022. doi: 10.48550/arXiv.2204.05080. arXiv: 2204.05080.

Tsimpoukelli, M., Menick, J., Cabi, S., Eslami, S. M. A., Vinyals, O., and Hill, F. Multimodal Few-Shot Learning with Frozen Language Models. *CoRR*, abs/2106.13884, 2021. arXiv: 2106.13884.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Vershynin, R. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2018. doi: 10.1017/9781108231596.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J. P., Jaderberg, M., Vezhnevets, A. S., Leblond, R., Pohlen, T., Dalibard, V., Budden, D., Sulsky, Y., Molloy, J., Paine, T. L., Gülçehre, C., Wang, Z., Pfaff, T., Wu, Y., Ring, R., Yogatama,

D., Wünsch, D., McKinney, K., Smith, O., Schaul, T., Lillicrap, T. P., Kavukcuoglu, K., Hassabis, D., Apps, C., and Silver, D. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nat.*, 575(7782):350–354, 2019. doi: 10.1038/s41586-019-1724-z.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-Attention with Linear Complexity. *CoRR*, abs/2006.04768, 2020. arXiv: 2006.04768.

Widrich, M., Schäfl, B., Pavlovic, M., Ramsauer, H., Gruber, L., Holzleitner, M., Brandstetter, J., Sandve, G. K., Greiff, V., Hochreiter, S., and Klambauer, G. Modern Hopfield Networks and Attention for Immune Repertoire Classification. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.-F., and Lin, H.-T. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Widrich, M., Hofmarcher, M., Patil, V. P., Bitto-Nemling, A., and Hochreiter, S. Modern Hopfield Networks for Return Decomposition for Delayed Rewards. In *Deep RL Workshop NeurIPS 2021*, 2021.

Wilcoxon, F. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6):80–83, 1945. ISSN 00994987. Publisher: [International Biometric Society, Wiley].

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., Platen, P. v., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. Transformers: State-of-the-Art Natural Language Processing. In Liu, Q. and Schlangen, D. (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, pp. 38–45. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-demos.6.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR*, abs/1609.08144, 2016. arXiv: 1609.08144.

Yao, S., Rao, R., Hausknecht, M. J., and Narasimhan, K. Keep CALM and Explore: Language Models for Action Generation in Text-based Games. In Webber, B., Cohn, T.,

He, Y., and Liu, Y. (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pp. 8736–8754. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.704.

Zenke, F., Poole, B., and Ganguli, S. Continual Learning Through Synaptic Intelligence. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3987–3995. PMLR, 2017.

Zhang, T., Wu, F., Katiyar, A., Weinberger, K. Q., and Artzi, Y. Revisiting Few-sample BERT Fine-tuning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Zuo, X. mazelab: A customizable framework to create maze and gridworld environments., 2018. Publication Title: GitHub repository.

Åström, K. J. Optimal control of Markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10:174–205, 1964.

A. Environments

A.1. RandomMaze

The RandomMaze environment is implemented as a procedurally generated POMDP. Each time the environment is reset the size of the grid is sampled randomly from $\{5, \ldots, 25\} \subset \mathbb{N}$. The agent spawns at random positions on the grid, and the goal tile is always located in the lower right corner. Partial observability is enforced by cropping the observation to a 9×9 agent centered grid. Thus, the agent must remember if it has seen the goal before moving into different directions. Four possible actions can be issued resembling the four directions up, down, left, and right. If the agent bumps into a wall, it receives a reward of -1, for each interaction step it receives a reward of -0.01, and for reaching the goal tile it receives a reward of $+1$. Figure 9 depicts an example of a randomly sampled maze and its partially observable counterpart.

A.2. Minigrid

We select a set of six diverse partially observable Minigrid (Chevalier-Boisvert et al. (2018)) environments:

- **MiniGrid-DoorKey-5x5-v0 (DoorKey5x5):** The agent must pick up a key and unlock a door, before navigating to a goal tile in a 5×5 grid.

- **MiniGrid-DoorKey-6x6-v0 (DoorKey6x6):** Same task as above, but in a 6×6 grid, which requires more exploration.

- **MiniGrid-KeyCorridorS3R1-v0 (KeyCorridor):** The agent must find a key behind a closed door, unlock another door, and pick up an item behind the unlocked door.

- **MiniGrid-Dynamic-Obstacles-Random-5x5-v0 (DynamicObstacles):** The agent spawns at random positions in a 5×5 grid and must navigate towards a goal tile while avoiding dynamically moving obstacles. If the agent collides with an obstacle it receives a negative reward and the episode ends. This environment provides a strong local optimum by simply avoiding the obstacles which leads to a final return of 0.

- **MiniGrid-Unlock-v0 (Unlock):** The agent has to open a locked door. This environment is a subtask of the DoorKey-environments, but in a bigger grid.

- **MiniGrid-RedBlueDoors-6x6-v0 (RedBlueDoors):** The agent spawns at random positions in a 6×6 grid in which a red and a blue door is placed in opposite directions. The aim is to first open the red door and then open the blue door in that order.

Figure 10 depicts the six different toytasks. In the partial observability setting the agent receives a 7×7 agent-centered excerpt of the grid depending on which direction the agent is currently facing. We compare HELM to a Markov policy (CNN-PPO) to elaborate whether the agent truly requires memory to solve the corresponding tasks. Further, we compare HELM to the Impala-PPO agent. Figure 11 shows the performance for each method at the end of training after 500K interaction steps. Surprisingly, we find that 5 out of 6 environments are solvable without access to memory, indicating that even though partial observability is enforced, no essential information is neglected. The only environment to be truly dependent on memory is the KeyCorridor environment, for which the Markov policy stagnates at a low return. Remarkably, HELM yields performance gains over CNN-PPO on 2 environments which are solvable without a memory mechanism while performing on par with CNN-PPO on DoorKey6x6. This highlights HELM's ability of compact history compression can even be beneficial in environments where no memory is required. Impala-PPO is consistently outperformed by competitors.

A.3. Procgen

The Procgen benchmark Cobbe et al. (2020) allows benchmarking of RL algorithms for generalization and sample efficiency. Generalization is evaluated by training on a subset of procedurally generated environments and evaluating on a held-out set of levels. Sample efficiency is evaluated by training on the entire level distribution for a limited amount of interaction steps with the environment. Further, different distribution methods are provided to modify the training distribution, e. g., *easy*, *hard*, or *memory*. The *hard* mode enhances difficulty by increasing the world size and spawning more objects as in the *easy* mode. The *memory* uses even larger world sizes and is only available for a subset of the environments provided by Procgen:

- **caveflyer:** The agent must traverse a network of caves and reach the exit. Additional reward may be collected by shooting enemies along the way.
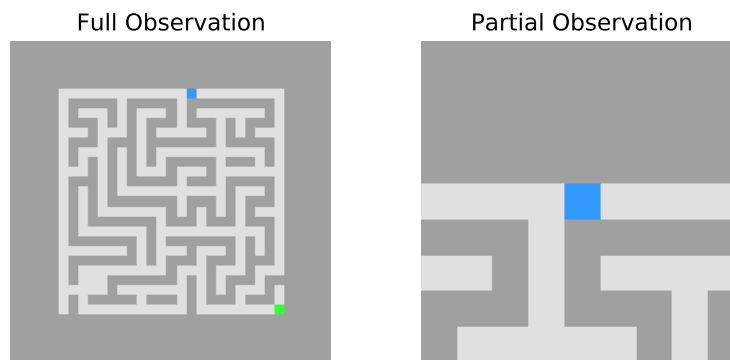
Figure 9: Example level for RandomMaze environment fully observable (left), and partially observable (right).
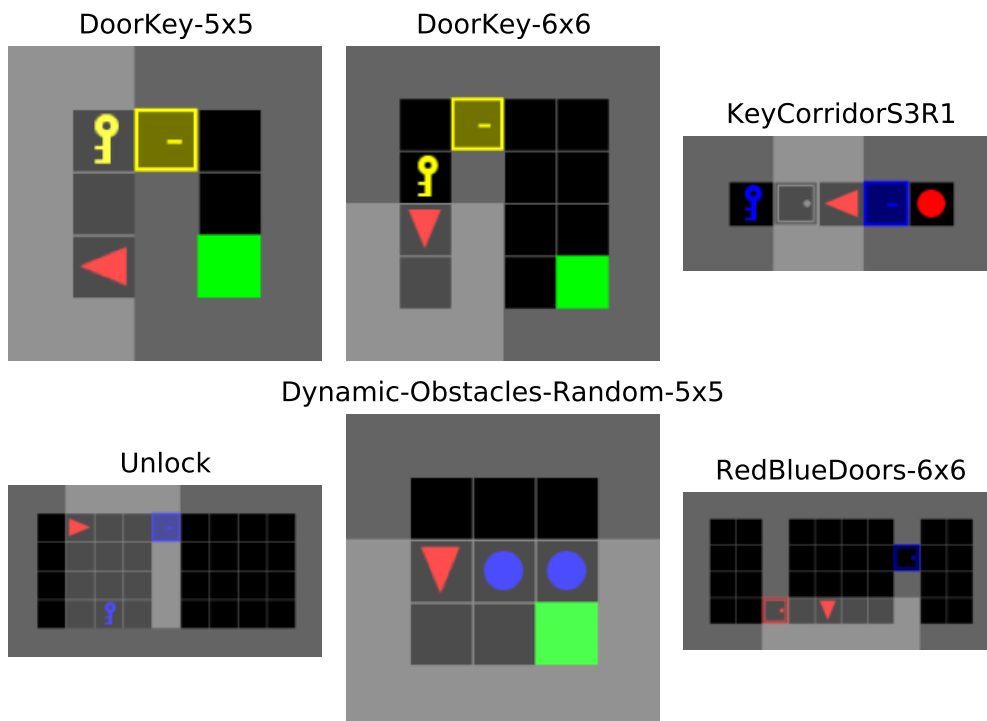

Figure 10: Example levels for Minigrid environments.

- **dodgeball:** The agent spawns in a room with a certain amount of enemies and must shoot those before unlocking a platform to which it must navigate to.

- **heist:** The agent spawns in a maze and must collect a gem behind a network of locked doors. The keys for different doors are in certain colors and distributed across the environment.

- **jumper:** The agent must find a carrot in an open world environment which contains spike obstacles.

- **maze:** The agent spawns in a maze and must find a piece of cheese.

- **miner:** The agent must dig through dirt and collect gems before ending the episode by moving to the exit.

Additonally, partial observability is enforced by cropping observations (miner, maze, dodgeball, heist), or altering the level generation logic (caveflyer, jumper). We train on the *memory* distribution mode since the agent must exploit representations of the pretrained language models to effectively solve the environments. A sample observation for each of the 6 environments is shown in Figure 12.

The lower bound for the computation of the mean normalized return measure R_{min} is obtained by training the Impala-PPO agent on fully masked observations across 10 seeds and averaging over performance at the end of training. For miner, we compute the optimal upper bound R_{max}. For maze, jumper, and heist the R_{max} value is fixed, and for caveflyer and dodgeball we estimate R_{max} by sampling 100 different level configurations and averaging over their maximum reachable returns. Table 2 depicts all values for R_{min} and R_{max} for their respective environments. For LSTM-PPO, we use the originally proposed normalization ranges of the *hard* mode (Cobbe et al., 2020).

Table 2: R_{min} and R_{max} values used for computation of mean normalized score across Procgen environments for distribution mode *memory*.

ENVIRONMENT	R_{min}	R_{max}
CAVEFLYER	2.5	43.1
DODGEBALL	1.5	32.1
MINER	1.3	46
MAZE	2.7	10
HEIST	1.2	10
JUMPER	1.2	10

B. Hyperparameter Search

We perform a parameter search over learning rate in $\{3e\text{-}4, 1e\text{-}5, 5e\text{-}5, 1e\text{-}4\}$, entropy coefficient in $\{0.05, 0.01, 0.005, 0.001\}$, rollout length in $\{64, 128, 256\}$, and the softmax scaling factor $\beta \in \{0.5, 1, 10, 50, 100\}$ for HELM for the Minigrid environments and the RandomMaze environment. For Procgen environments, we reduce the possible values for the entropy coefficient to $[0.01, 0.005, 0.001]$ and for β to $\{1, 10, 100\}$. Cobbe et al. (2020) found that scaling up the number of channels of the convolutional encoder led to significant improvements. Thus, by default we scale the number of channels of the CNN encoder by $\lambda_c = 4$. For all experiments with HELM, we set the number of minibatches to 8. For the Impala-PPO baseline we search over batch sizes of $\{4, 8, 16\}$ sequences per batch, learning rates $\{1e\text{-}5, 5e\text{-}5, 1e\text{-}4\}$, and entropy coefficients $\{0.05, 0.01, 0.005, 0.001\}$ for Minigrid and RandomMaze environments. Further, we also search over channel scaling factors $\lambda_c \in \{1, 2, 4\}$. For Procgen, we set $\lambda_c = 4$ and run a reduced search over parameter values for the entropy coefficient in $\{0.01, 0.005, 0.001\}$ and batch sizes of $\{8, 16\}$ sequences per batch. The values for γ and λ_{GAE} were found experimentally for Minigrid environments, while for Procgen they were chosen according to prior work (Cobbe et al., 2020). The number of PPO epochs is set to 3 for all experiments. We found that learning rate decay did not lead to improvements except for the RedBlueDoors environment. We also experimented with decaying the clipping range without observing any benefits though. Thus, we set the clipping range to 0.2 for all experiments. We choose AdamW (Loshchilov & Hutter, 2019) as optimizer for all environments with default values for weight decay. Moreover, we clip the norm of the gradients to 0.5 for all of our experiments. For CNN-PPO, we run a grid comprising the same parameters as for HELM, except for the parameter β, which is only available for the FrozenHopfield component. We set $\lambda_c = 4$ by default for all environments. The number of environments for RandomMaze and Minigrid are set to
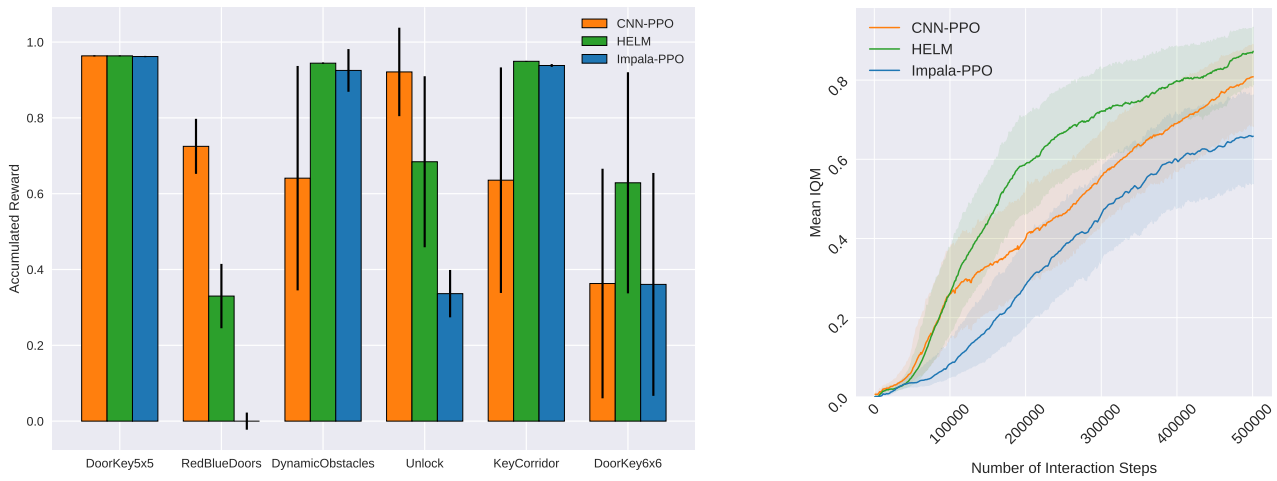
Figure 11: Comparison of return over last 100 episodes of HELM to Impala-PPO and CNN-PPO (left). IQM and 95 % bootstrapped CIs are shown across 30 seeds. Mean IQM and 95 % CIs across all environments for all methods (right).
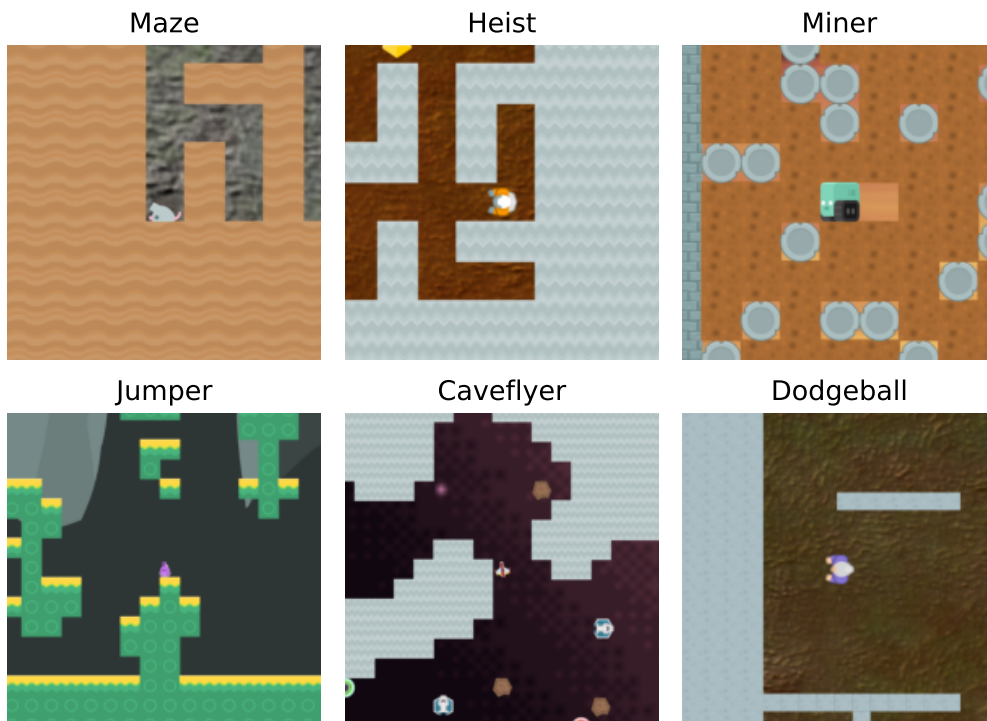

Figure 12: Procgen environments in distribution mode *memory*. Observations of heist, maze, miner, and dodgeball are cropped and centered on the agent. Jumper and caveflyer avoid pruning of dead ends during game generation.

Table 3: Hyperparameters for all six MiniGrid environnments, DK is short for DoorKey, and RBD is short for RedBlueDoors.

PARAMETER	DK5x5	DK6x6	UNLOCK	KEYCORRIDOR	DYNAMICOBSTACLES	RBD
LEARNING RATE	1E-4	3E-4	3E-4	1E-4	1E-4	5E-5
ROLLOUT LENGTH	64	64	64	128	64	256
ENTROPY COEFF.	1E-2	1E-2	1E-2	5E-2	5E-3	1E-2
γ	0.99	0.99	0.99	0.99	0.99	0.99
λ	0.99	0.99	0.99	0.99	0.99	0.99
β	1	10	50	100	100	1
λ_c	4	4	4	4	4	4

Table 4: Hyperparameters for Procgen environnments.

PARAMETER	CAVEFLYER	DODGEBALL	HEIST	JUMPER	MAZE	MINER
LEARNING RATE	1E-5	1E-4	1E-5	5E-5	5E-5	1E-4
ROLLOUT LENGTH	128	256	256	256	128	128
ENTROPY COEFF.	5E-3	1E-3	1E-2	5E-3	5E-3	1E-3
γ	0.999	0.999	0.999	0.999	0.999	0.999
λ	0.95	0.95	0.95	0.95	0.95	0.95
β	1	1	100	1	1	1

16, while it is set to 64 for Procgen. The parameter search is run for 60K, 500K, and 2.5M interaction steps for Minigrid, RandomMaze, and Procgen, respectively. We select the parameter combination of the setting which yields the highest IQM at training end over 5 seeds. The best hyperparameters for HELM for all Minigrid environments are shown in Table 3 while the best hyperparameters for the Procgen environments are shown in Table 4. For GTrXL, we perform a parameter search over the same grid as for HELM, excluding the parameter β. We use the same CNN encoder as for all other experiments for encoding the environment observations before feeding them into GTrXL. The finetuning experiments use the same parameter values as found in the parameter search for HELM, but we additionally search over different values for the learning rate {1e-4, 1e-5, 1e-6} used for finetuning the Transformer and the input mapping. During training and finetuning we chunk sequences into a length of 20 timesteps and feed them into the Transformer after applying the input encoder.

C. PPO vs PPG

The PPG algorithm (Cobbe et al., 2021) was proposed for improving sample efficiency over PPO on the Procgen benchmark suite. It separates the updates of the policy and the value function in different phases while adding an additional phase in which the features of the value function are distilled into the policy network. This resulted in substantial improvements in terms of sample efficiency on the Procgen environments in the *hard* mode. Since HELM may be optimized with any RL algorithm, we show results of our method when trained with PPG (HELM-PPG). As in Section 3 we compare to a recurrent agent based on the small Impala architecture (Impala-PPG) and a Markovian baseline (CNN-PPO). Additionally, we include the performance of HELM trained with PPO as a point of comparison. We find that HELM-PPG significantly improves upon the baselines in 3 out of the 6 environments, namely caveflyer ($p = 1.1e-3$), maze ($p = 1.56e-3$), and miner ($p = 6.57e-4$) (see Figure 16). On the remaining environments the performance of HELM-PPG is on-par with the competitors. As compared to the performance using PPO we find that Impala-PPG and CNN-PPG are more stable and perform on-par with HELM-PPG on heist and jumper. However, on caveflyer and maze both Impala-PPG and CNN-PPG stagnate within the 25M interaction steps. Furthermore, the scores reached with the PPG variants are in general lower than for PPO. A possible reason is that PPG only overcomes PPO after sufficient amount of training, i.e., after useful features have been learned that are beneficial for the policy. Further, it might be that the advantages of PPG are mostly observed in other Procgen environments that are not dependent on memory. We verify this assumption by visualizing the performance of a Markovian policy trained with PPO (CNN-PPO*) and PPG (CNN-PPG*) for the 6 Procgen environments in the *hard* mode across 3 seeds, which are provided by the authors in the respective codebases (Figure 17). Indeed, we find that the advantages of PPG over PPO can only be observed after a substantial amount of training, which exceeds the limited budget in our work.
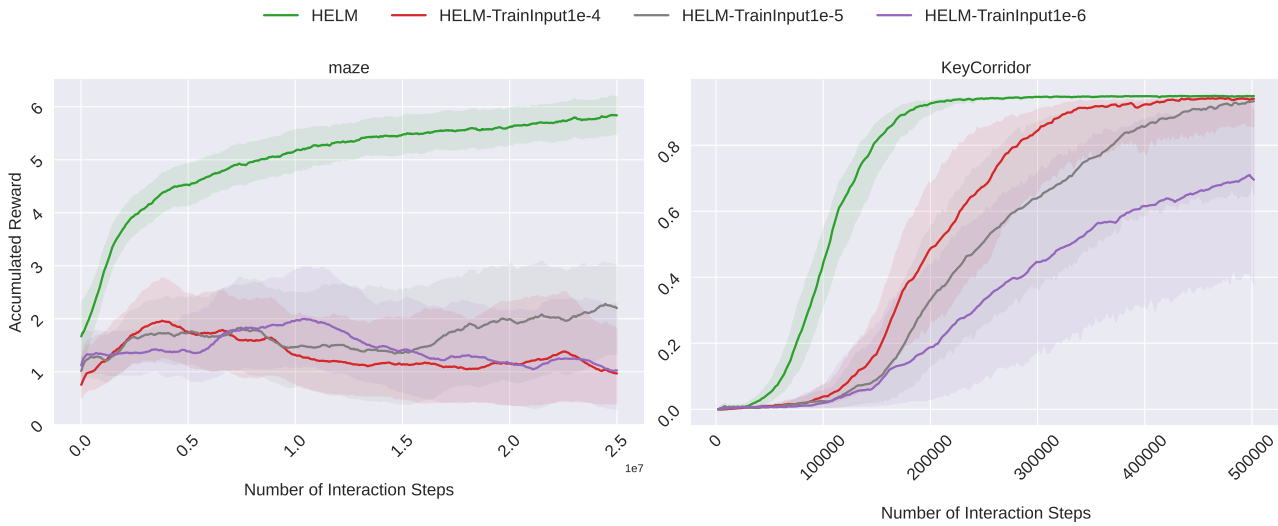
Figure 13: Comparison of HELM to training an input mapping to the Transformer model for three different learning rates. IQM and 95% bootstrapped confidence interval of return over last 100 episodes across 30 seeds are shown.
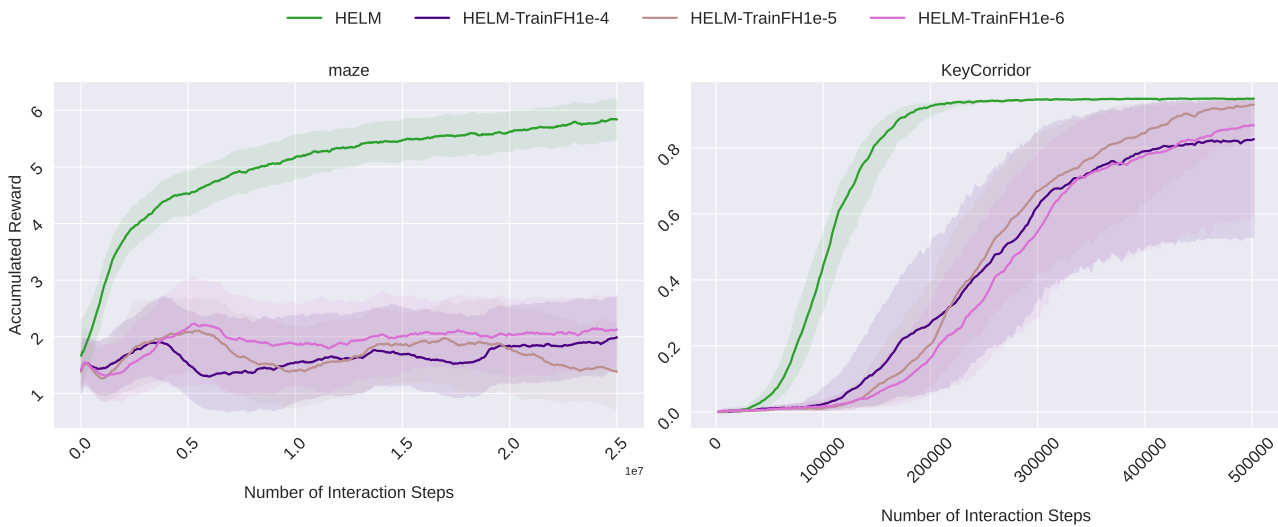

Figure 14: Comparison of HELM to training the linear projection in the *FrozenHopfield* component for three different learning rates. IQM and 95% bootstrapped confidence interval of return over last 100 episodes across 30 seeds are shown.
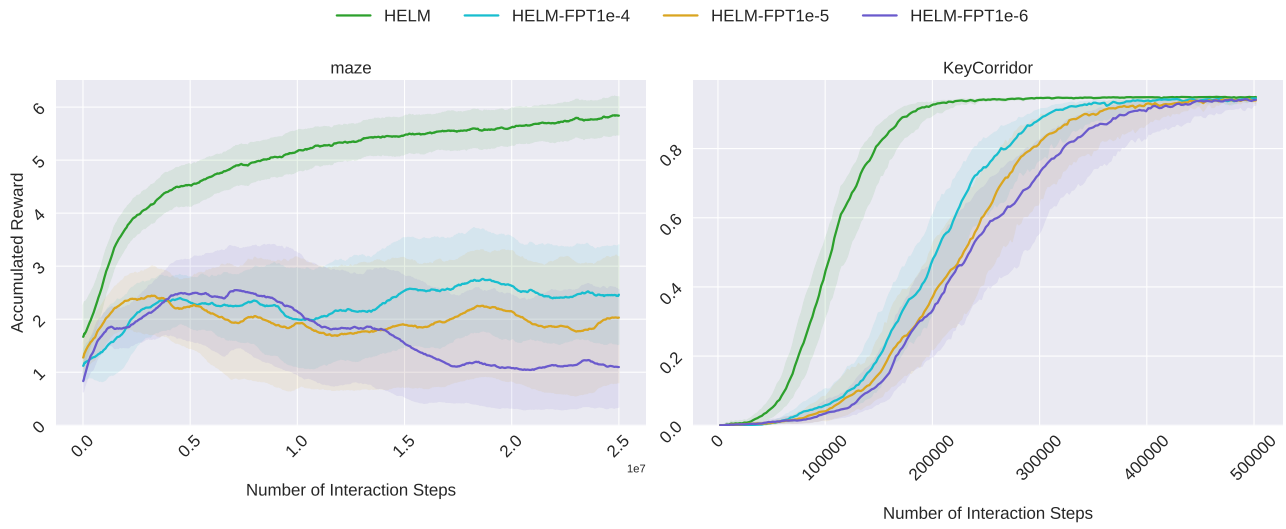
Figure 15: Comparison of HELM to the FPT setting for three different learning rates. IQM and 95% bootstrapped confidence interval of return over last 100 episodes across 30 seeds are shown.
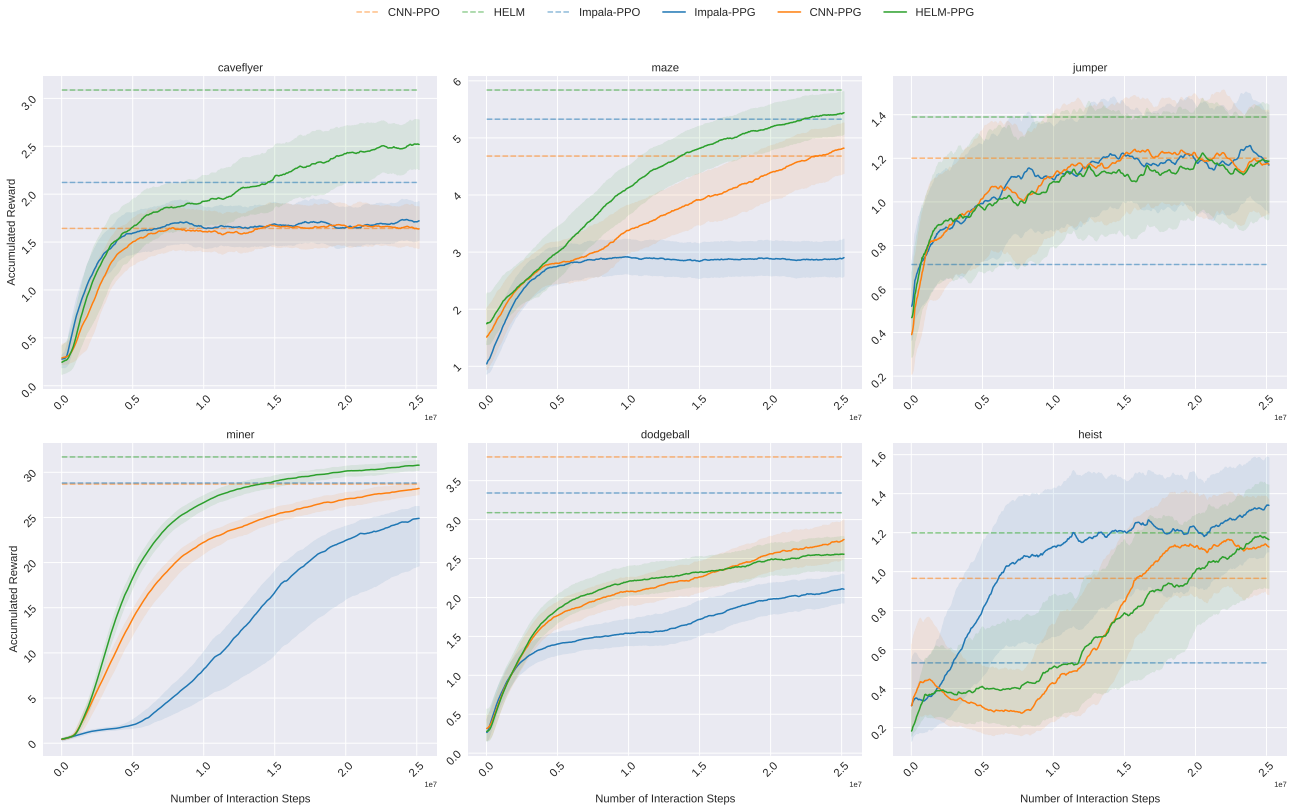

Figure 16: Performance of HELM-PPG compared to Impala-PPG and CNN-PPG after training for 25M interaction steps. IQM and 95 % bootstrapped confidence interval of return over last 100 episodes across 30 seeds are shown.

D. Ablation Studies

We perform ablation studies on the *FrozenHopfield* mapping by exchanging it with a linear layer (HELM-TrainInputX, where X is replaced by the learning rate used for training the linear mapping). As mentioned in Appendix B, we conduct experiments with three different learning rates for training the encoder, while keeping the learning rate for the actor-critic head and the CNN encoder of the current timestep fixed to the learning rate found by our parameter search. We show results on the Minigrid KeyCorridor and Progen maze environments in Figure 13. On both environments HELM significantly outperforms the second-best HELM-TrainInputX variant ($p = 8.64e-7$ for KeyCorridor, $p = 9.08e-5$ for maze). On KeyCorridor the performance decreases with a lower learning rate indicating that a higher learning rate is more suited for re-learning the input projection. Still, there is a clear performance drop compared to HELM. However, training an input projection results in severe instabilities in maze, where a lower learning rate appears to perform better.

Another experiment investigates the effect of training the linear projection in the *FrozenHopfield* mechanism. Again, we train the linear projection with three different learning rates (HELM-TrainFHX) and show the performances on maze and KeyCorridor in Figure 14. On KeyCorridor the performance is comparable for different learning rates and significantly worse than for HELM ($p = 6.19e-7$). On maze we observe the same trend, with HELM significantly outperforming all competitors ($p = 9.13e-5$).

Next, we compare HELM to minimally finetuning parts of the Transformer language model. Lu et al. (2021) introduced *Frozen Pretrained Transformer* (FPT) which finetunes the input projection, the absolute positional embeddings, the layernorm parameters, and the readout for cross-modal transfer after language pretraining. Analogously, we train an input projection to the Transformer as a linear layer and finetune the layernorm parameters of TrXL (HELM-FPTX) with three different learning rates {1e-4, 1e-5, 1e-6} (see Figure 15). HELM significantly outperforms all other methods on KeyCorridor ($p = 5.15e-4$) and maze ($p = 1.65e-4$).

Moreover, we compare HELM to fully finetuning the Transformer (HELM-FinetunedX). To this end we exchange the FrozenHopfield component with a linear layer and finetune all weights of the TrXL for three different learning rates ($\{1e-4, 1e-5, 1e-6\}$). Figure 18 shows the resulting learning curves on the KeyCorridor and the maze environments. HELM outperforms the second-best method on KeyCorridor and maze with a significance level of $p = 7.15e-6$, and $p = 1.61e-4$, respectively.

While Transformer models excel when sufficient data is available it appears for low-resource settings like on-policy RL plenty of interaction steps with the environment are required for convergence. In contrast, HELM is more sample efficient, stable, and exhibits lower variance than finetuning various components on the downstream task. In line with findings of Rothermel et al. (2021), we observe that fully finetuning on the downstream task appears to be more stable than minimal finetuning of certain components.

Additionally, we conduct an ablation on exchanging TrXL with other models for history compression in RL. Particularly, we exchange TrXL with a pretrained LSTM language model and an LSTM pretrained on dynamics modeling on the KeyCorridor environment.

We show performance of the following settings:

- **HELM**: Our proposed setting using *FrozenHopfield* and TrXL.

- **LSTM**: An LSTM language-model encoder Merity et al. (2018) commonly used for transfer (e. g., Howard & Ruder (2018)), pretrained on WikiText-103 Merity et al. (2017). We still use the FrozenHopfield mechanism for mapping observations to the space of pretrained LSTM token embeddings.

- **RIDE**: An LSTM encoder trained on an intrinsic reward for exploration in procedurally generated environments proposed by Raileanu & Rocktäschel (2020).

Results are shown in Figure 19. Remarkably, HELM performs on-par with RIDE although RIDE is biased towards the dynamics of the KeyCorridor environment. Interestingly, the LSTM model pretrained on natural language also reaches performance on-par with HELM and RIDE. These results provide evidence that the pretraining objective of language modeling is very well suited for history compression.

To highlight the importance of transfer from the language domain we conduct another set of experiments. We train the following variants on the KeyCorridor environment:
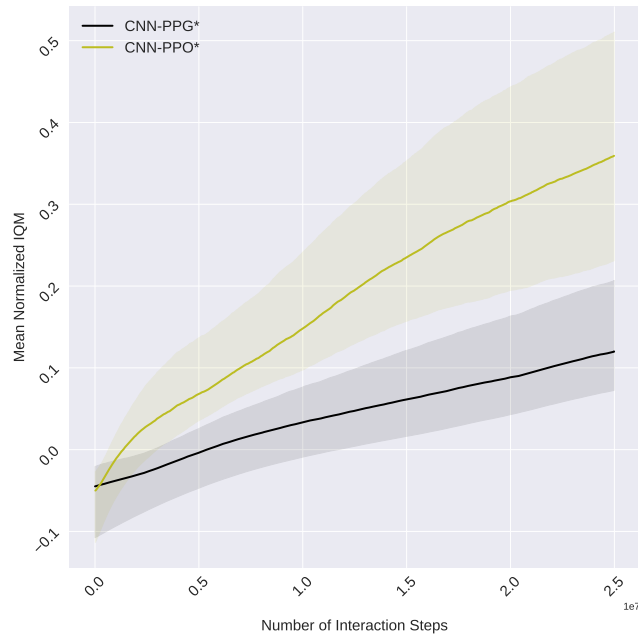
Figure 17: Mean IQM and 95 % bootstrapped confidence interval of the normalized return over 25M interaction steps across 3 seeds provided by the authors of PPG and PPO in their respective codebases.
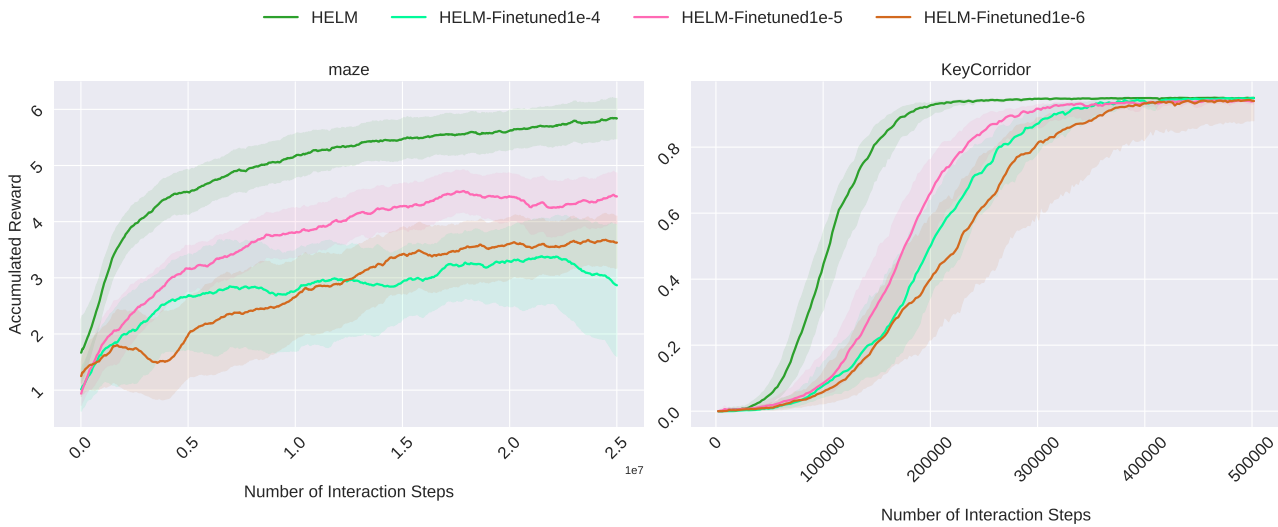

Figure 18: Comparison of HELM to fully finetuning for three different learning rates. IQM and 95 % bootstrapped confidence interval of return over last 100 episodes across 30 seeds are shown.

- **CNN-PPO+N**: Exchanging the output of TrXL with a random noise vector sampled from $\mathcal{N}(0,1)$ to evaluate whether the output of TrXL carries more information than randomness.

- **HELM-R**: Randomly initializing TrXL, since the effectiveness of random functions has been established in prior work (Maass et al., 2002a; Jaeger, 2001).

- **HELM+N**: Corrupting the output of TrXL with a random noise vector sampled from $\mathcal{N}(0,1)$.

- **CNN-PPO+Pos**: Exchanging the output of TrXL with sinusoidal positional embeddings as used in Devlin et al. (2019), to determine whether the only additional information to effectively solve the task is the current position.

We perform gridsearches as mentioned in Appendix B for all aforementioned variants. Figure 20 right shows the comparison of HELM to CNN-PPO, CNN-PPO+N, CNN-PPO+Pos. We observe that exchanging the TrXL output with noise acts as regularization during the hyperparameter search of CNN-PPO, thus, to improved performance over CNN-PPO. Augmenting CNN-PPO with positional information yields a further boost in performance. Still, both variants are less sample efficient than HELM, which indicates that the pretrained TrXL provides more useful information than positional information. Figure 20 left shows the comparison of HELM to HELM-R, and HELM+N. Initializing TrXL at random results in a performance drop and increased variance, which illustrates the importance of the transfer from the language domain. Furthermore, corrupting the output of TrXL with noise results diminishes performance.

E. Notes on the Johnson-Lindenstrauss Lemma

The Johnson-Lindenstrauss lemma (Johnson & Lindenstrauss, 1984) states that the distances between points of a finite set are approximately preserved by a random projection with high probability. One can obtain this result by showing a similar property for a single point and then taking the union bound over all pairwise distances in the set. Since we want to apply the lemma to the observations in an environment, i.e., a possibly infinite set, we are interested in this more elementary version of the lemma. It can be found, e.g., in Vershynin (2018) lemma 5.3.2.

In Section 2 we consider a random projection matrix $\boldsymbol{P} \in \mathbb{R}^{m \times n}$ with P_{ij} sampled i.i.d. from an appropriately scaled Gaussian distribution and an arbitrary but fixed vector $\boldsymbol{d} \in \mathbb{R}^n$. We are interested in how much the random projection $\boldsymbol{Pd}$ distorts the length of $\boldsymbol{d}$, i.e., we want to compare $\|\boldsymbol{Pd}\|$ to $\|\boldsymbol{d}\|$.

First, note that we can interchange the roles of $\boldsymbol{P}$ and $\boldsymbol{d}$, i.e., we can choose $\boldsymbol{d}$ at random and leave $\boldsymbol{P}$ fixed. To see this, let $\boldsymbol{R} \in \mathbb{R}^{n \times n}$ be the rotation matrix that aligns $\boldsymbol{d}$ with the first axis and consider $\boldsymbol{PR}^\top \boldsymbol{Rd}$. Due to rotation invariance of the Gaussian distribution, the distribution of $\boldsymbol{PR}^\top$ is the same as that of $\boldsymbol{P}$. The multiplication with $\boldsymbol{Rd}$ corresponds to selecting the first column of $\boldsymbol{PR}^\top$ and scaling it by $\|\boldsymbol{Rd}\| = \|\boldsymbol{d}\|$. This is the same as choosing $\boldsymbol{d}$ at random with appropriate scaling and mapping it onto its first m coordinates. Further, we may normalize $\boldsymbol{d}$ to unit length and scale the projection accordingly.

Let $d_i \sim \mathcal{N}(0,1)$ and let $\boldsymbol{z} = \boldsymbol{d}/\|\boldsymbol{d}\|$, i.e., we choose $\boldsymbol{z}$ from the unit sphere uniformly at random. Further, let $\boldsymbol{Qz}$ be the projection of $\boldsymbol{z}$ onto its first m coordinates. We have $1 = \mathbb{E}[\|\boldsymbol{z}\|_2^2] = \mathbb{E}[\sum_{i=1}^n z_i^2] = n\mathbb{E}[z_1^2]$ and, consequently,

$$\mathbb{E}\left[\|\boldsymbol{Qz}\|_2^2\right] = \mathbb{E}\left[\sum_{i=1}^m z_i^2\right] = m\mathbb{E}[z_1^2] = \frac{m}{n}. \tag{E.1}$$

From lemma 2.2 (Dasgupta & Gupta, 2003) we conclude that for any $0 < \varepsilon < 1$

$$\mathbb{P}\left[\|\boldsymbol{Qz}\|_2^2 \le (1-\varepsilon)\frac{m}{n} \cup \|\boldsymbol{Qz}\|_2^2 \ge (1+\varepsilon)\frac{m}{n}\right] \le 2\exp\left(-\frac{m(\varepsilon^2/2 - \varepsilon^3/3)}{2}\right). \tag{E.2}$$

Therefore, the complementary event can bounded by

$$\mathbb{P}\left[(1-\varepsilon) \le \left\|\sqrt{\frac{n}{m}}\boldsymbol{Qz}\right\|_2^2 \le (1+\varepsilon)\right] \ge 1 - 2\exp\left(-\frac{m(\varepsilon^2/2 - \varepsilon^3/3)}{2}\right). \tag{E.3}$$

Equation (2) follows by multiplication with $\|\boldsymbol{d}\|_2^2$ inside the probability and letting $\boldsymbol{P} = \sqrt{n/m}\boldsymbol{Q}$.
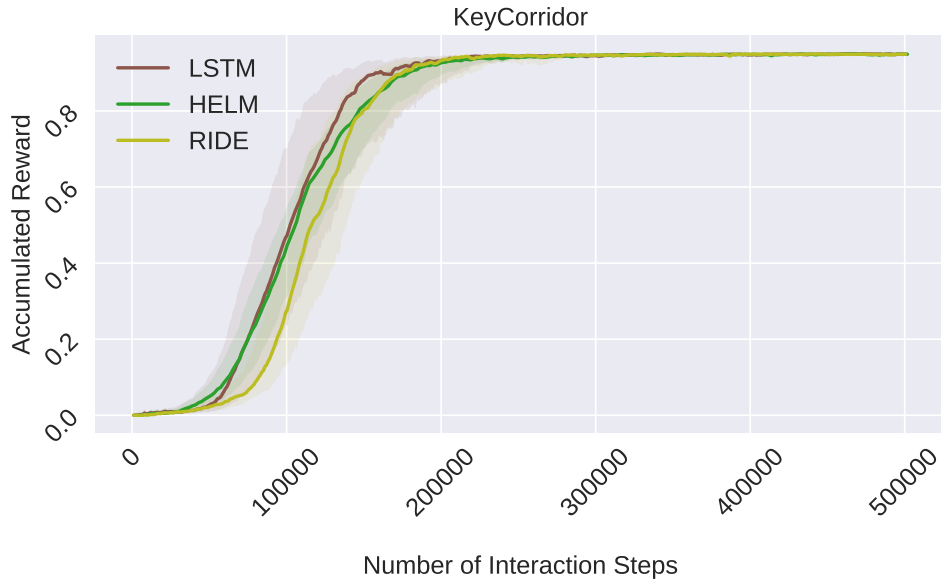
Figure 19: Comparison of HELM to other pretrained recurrent models used for history compression. Mean and 95 % confidence interval of return over last 100 episodes across 30 seeds are shown.
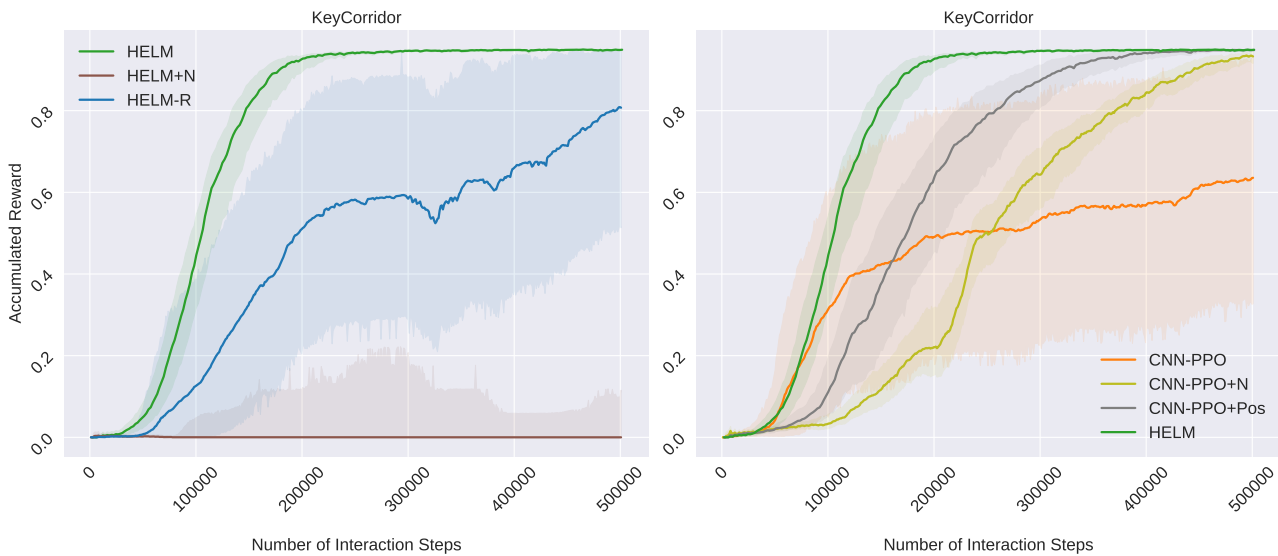

Figure 20: Noise injection experiments to highlight the transfer from the language domain. IQM and 95 % bootstrapped confidence interval of return over last 100 episodes across 30 seeds are shown.

F. Review of Modern Hopfield Networks

Hopfield networks are energy-based binary associative memories, which popularized artificial neural networks in the 1980s (Amari, 1972; Hopfield, 1982; 1984). Associative memory networks have been designed to store and retrieve samples. Their storage capacity can be considerably increased by polynomial terms in the energy function (Chen et al., 1987; Psaltis & Park, 1986; Baldi & Venkatesh, 1987; Gardner, 1987; Abbott & Arian, 1987; Horn & Usher, 1988; Caputo & Niemann, 2002; Krotov & Hopfield, 2016).

In contrast to these binary memory networks, continuous MHNs utilize an associative memory with exponential storage capacity. These MHNs for deep learning architectures have an energy function with continuous states and can retrieve samples with only one update (Ramsauer et al., 2021).

We assume a set of patterns (or embeddings) $\{e_1, \ldots, e_k\} \subset \mathbb{R}^m$ that are stacked as row vectors to form the matrix $\boldsymbol{E} = (e_1, \ldots, e_k)^\top$ and a state pattern (query) $\boldsymbol{\xi} \in \mathbb{R}^m$ that represents the current state. In the *FrozenHopfield* mechanism we obtain the state pattern by a random but fixed mapping $\boldsymbol{\xi} = \boldsymbol{P}\boldsymbol{o}_t$. The largest norm of a stored pattern is $M = \max_i \|e_i\|_2$. Continuous MHNs with state $\boldsymbol{\xi}$ have the energy

$$L = -\beta^{-1} \log \left(\sum_{i=1}^m \exp(\beta e_i^\top \boldsymbol{\xi}) \right) + \beta^{-1} \log k + \frac{1}{2} \boldsymbol{\xi}^\top \boldsymbol{\xi} + \frac{1}{2} M^2. \tag{F.1}$$

For energy L and state $\boldsymbol{\xi}$, the update rule

$$\boldsymbol{\xi} \leftarrow f(\boldsymbol{\xi}) = \boldsymbol{E}^\top \sigma(\beta \boldsymbol{E} \boldsymbol{\xi}) \tag{F.2}$$

has been proven to converge globally to stationary points of the energy L, which are almost always local minima (Ramsauer et al., 2021). The update rule (F.2) is also the formula of the well-known Transformer attention mechanism (Vaswani et al., 2017). Therefore, Hopfield retrieval and Transformer attention coincide.

The *separation* Δ_i of a pattern e_i is defined as its minimal dot product difference to any of the other patterns: $\Delta_i = \min_{j,j \neq i}(e_i^\top e_i - e_i^\top e_j)$. A pattern is *well-separated* from the data if $\Delta_i \geq \frac{2}{\beta k} + \frac{1}{\beta} \log \left(2(k-1)k\beta M^2 \right)$. If the patterns e_i are well separated, the update rule (F.2) converges to a fixed point close to a stored pattern. If some patterns are similar to one another and, therefore, not well separated, the update rule (F.2) converges to a fixed point close to the mean of the similar patterns. This fixed point is a *metastable state* of the energy function and averages over similar patterns.

The *FrozenHopfield* mechanism can be viewed as a retrieval with one update. The next theorem states that the update rule (F.2) typically converges after one update if the patterns are well separated. Furthermore, it states that the retrieval error is exponentially small in the separation Δ_i.

Theorem F.1 (Modern Hopfield Networks: Retrieval with One Update). *With query $\boldsymbol{\xi}$, after one update the distance of the new point $f(\boldsymbol{\xi})$ to the fixed point e_i^* is exponentially small in the separation Δ_i. The precise bounds using the Jacobian $J = \frac{\partial f(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}$ and its value $\bar{J}$ in the mean value theorem are:*

$$\|f(\boldsymbol{\xi}) - e_i^*\| \leq \|\bar{J}\|_2 \|\boldsymbol{\xi} - e_i^*\|, \tag{F.3}$$

$$\|\bar{J}\|_2 \leq 2\beta k M^2 (k-1) \exp(-\beta(\Delta_i - 2\max\{\|\boldsymbol{\xi} - e_i\|, \|e_i^* - e_i\|\}M)). \tag{F.4}$$

For given ε and sufficient large Δ_i, we have $\|f(\boldsymbol{\xi}) - e_i^*\| < \varepsilon$, that is, retrieval with one update. The retrieval error $\|f(\boldsymbol{\xi}) - e_i\|$ of pattern e_i is bounded by

$$\|f(\boldsymbol{\xi}) - e_i\| \leq 2(k-1) \exp(-\beta(\Delta_i - 2\max\{\|\boldsymbol{\xi} - e_i\|, \|e_i^* - e_i\|\}M))M. \tag{F.5}$$

For a proof see Ramsauer et al. (2021).

The main requirement of MHNs to be suited for association of different modalities is that they can store and retrieve enough embeddings of the target embedding space. We want to store a potentially large set of embeddings, since TrXL was pretrained with over 260K tokens. We first define what we mean by storing and retrieving patterns from a MHN.

Definition F.2 (Pattern Stored and Retrieved). We assume that around every pattern e_i a sphere S_i is given. We say e_i is *stored* if there is a single fixed point $e_i^* \in S_i$ to which all points $\boldsymbol{\xi} \in S_i$ converge, and $S_i \cap S_j = \emptyset$ for $i \neq j$. We say e_i *is retrieved* for a given ε if update rule (F.2) gives a point $\tilde{x}_i$ that is at least ε-close to the single fixed point $e_i^* \in S_i$. The retrieval error is $\|\tilde{x}_i - e_i\|$.

As with classical Hopfield networks, we consider patterns on the sphere, i.e., patterns with a fixed norm. For randomly chosen patterns, the number of patterns that can be stored is exponential in the dimension m of the space of the patterns ($e_i \in \mathbb{R}^m$).

Theorem F.3 (Modern Hopfield Networks: Exponential Storage Capacity). *We assume a failure probability $0 < p \leq 1$ and randomly chosen patterns on the sphere with radius $M := K\sqrt{m-1}$. We define $a := \frac{2}{m-1}(1 + \ln(2\beta K^2 p(m-1)))$, $b := \frac{2K^2\beta}{5}$, and $c := \frac{b}{W_0(\exp(a+\ln(b)))}$, where W_0 is the upper branch of the Lambert W function ([Olver et al., 2010](), [(4.13)]()), and ensure $c \geq \left(\frac{2}{\sqrt{p}}\right)^{\frac{4}{m-1}}$. Then with probability $1-p$, the number of random patterns that can be stored is*

$$k \geq \sqrt{p}\, c^{\frac{m-1}{4}}\,. \tag{F.6}$$

Therefore it is proven for $c \geq 3.1546$ with $\beta = 1$, $K = 3$, $m = 20$ and $p = 0.001$ ($a + \ln(b) > 1.27$) and proven for $c \geq 1.3718$ with $\beta = 1$, $K = 1$, $m = 75$, and $p = 0.001$ ($a + \ln(b) < -0.94$).

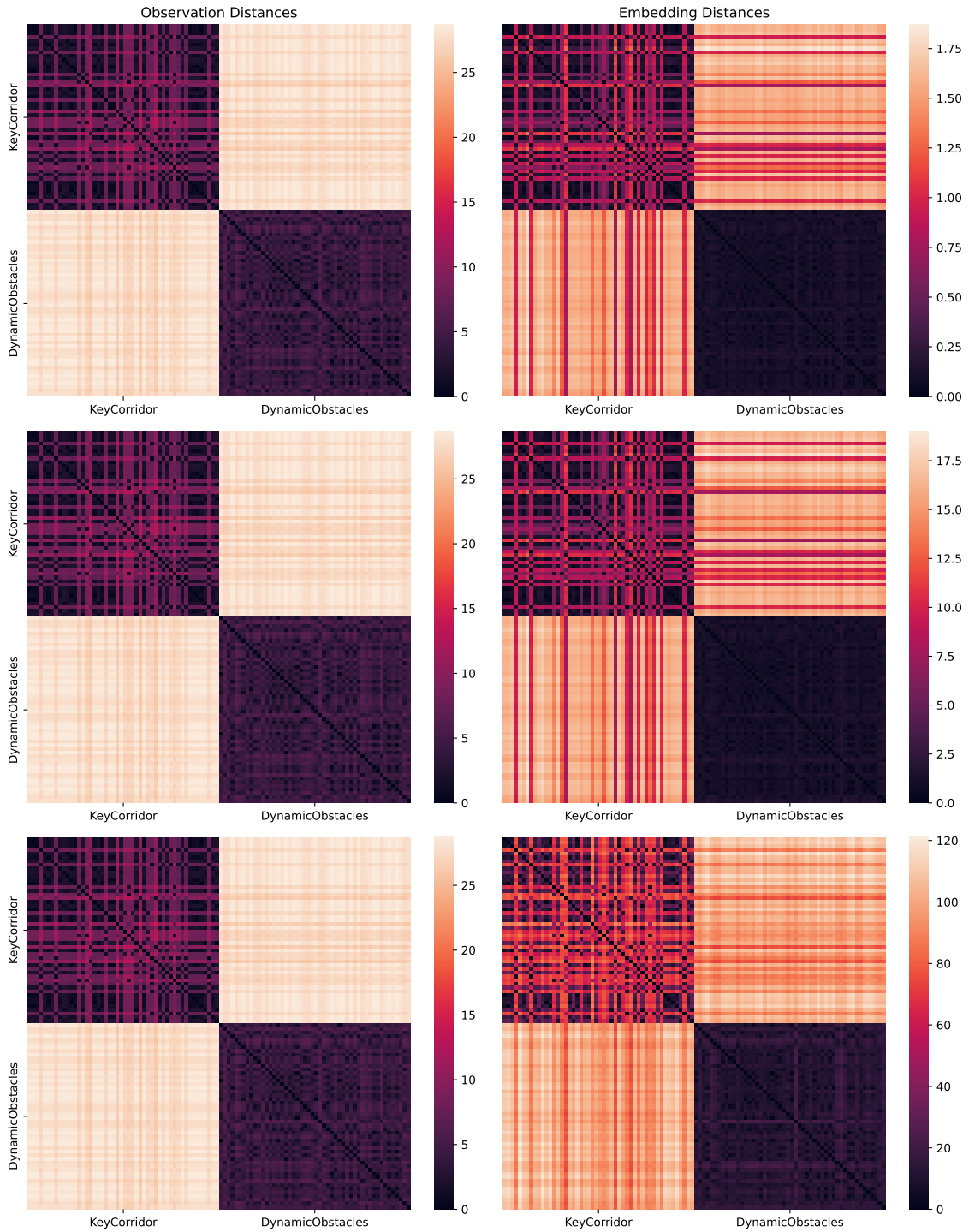For a proof see [Ramsauer et al. (2021)]().

Figure 21: Distance matrices for observations of the KeyCorridor and DynamicObstacles environments for different values of $\beta \in \{1, 10, 100\}$. Image observations are very well separated in pixel space (left column). After mapping the observations to the PLT input space the distances are well separated (right column). Increasing $\beta = 1$ (top) to $\beta = 10$ (middle) the distances in the embedding space are also scaled up by an order of magnitude. Further increasing the temperature parameter to $\beta = 100$ results in enhanced distances across environments, while inter-environment distances are also enhanced. Thus, the parameter β can avoid representation collapse in the token embedding space.
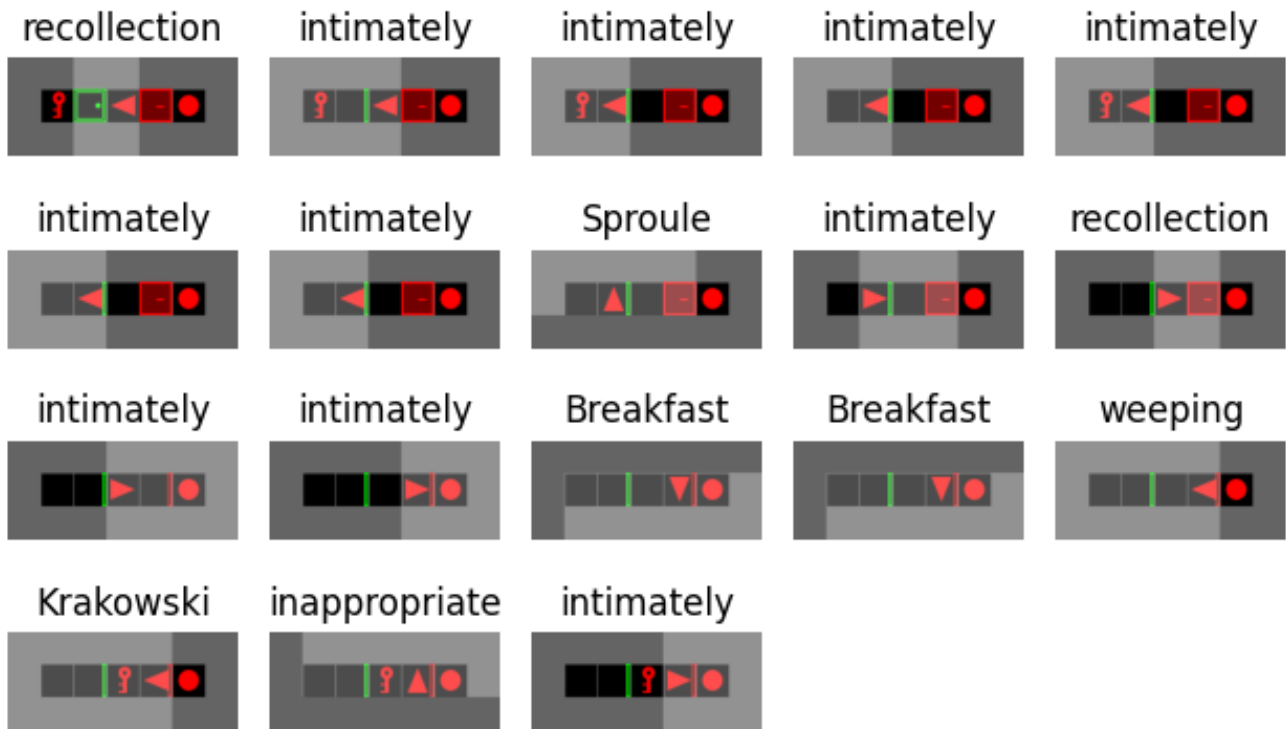
Figure 22: Sample episode of a trained policy with token annotations corresponding to the closest token in the PLT embedding space. The episode starts at left top and ends at right bottom. The token annotations are not meaningful since the mapping to the token space is initialized at random. Still, similar states in the pixel space map to the same token, e.g., "recollection" when the agent faces a door in front of it.