A Large Recurrent Action Model: xLSTM enables Fast Inference for Robotics Tasks

Thomas Schmied[1] **Thomas Adler**[1] **Vihang Patil**[1] **Maximilian Beck**[1,2] **Korbinian Pöppel**[1,2]
Johannes Brandstetter[1,2] **Günter Klambauer**[1,2] **Razvan Pascanu**[3,4] **Sepp Hochreiter**[1,2]
[1] ELLIS Unit, LIT AI Lab, Institute for Machine Learning, JKU Linz, Austria
[2] NXAI GmbH, Linz, Austria, [3] Google DeepMind, [4] UCL

Abstract

In recent years, there has been a trend in the field of Reinforcement Learning (RL) towards large action models trained offline on large-scale datasets via sequence modeling. Existing models are primarily based on the Transformer architecture, which result in powerful agents. However, due to slow inference times, Transformer-based approaches are impractical for real-time applications, such as robotics. Recently, modern recurrent architectures, such as xLSTM and Mamba, have been proposed that exhibit parallelization benefits during training similar to the Transformer architecture while offering fast inference. In this work, we study the aptitude of these modern recurrent architectures for large action models. Consequently, we propose a Large Recurrent Action Model (LRAM) with an xLSTM at its core that comes with linear-time inference complexity and natural sequence length extrapolation abilities. Experiments on 432 tasks from 6 domains show that LRAM compares favorably to Transformers in terms of performance and speed.

1 Introduction

Reinforcement Learning (RL) has been responsible for impressive success stories such as game-playing [Silver et al., 2016; Vinyals et al., 2019; Berner et al., 2019; Patil et al., 2022], plasma control for fusion [Degrave et al., 2022], or navigation of stratospheric balloons [Bellemare et al., 2020]. While these successes were based on classical RL approaches, in which agents have been trained online with RL objectives, recently there has been a trend towards offline RL settings [Levine et al., 2020; Schweighofer et al., 2022] and sequence models trained via behavior cloning [Chen et al., 2021; Janner et al., 2021]. Such approaches, in which agents are trained on large-scale offline datasets with causal sequence modeling objectives, have been driven by the proliferation of Transformer-based architectures and gave rise to what we refer to as *Large Action Models* (LAMs) to highlight their similarity to large language models (LLMs) [Radford et al., 2018]. LAM approaches can also be used in multi-task settings to develop generalist agents such as Gato [Reed et al., 2022].

Existing LAMs are primarily based on the Transformer [Vaswani et al., 2017] architecture. Because of their powerful predictive performance, robotics has become an emergent application area for large models [Brohan et al., 2023b,a; Octo Model Team et al., 2024; Gu et al., 2023; Wang et al., 2023] and a number of large multi-task datasets were collected [Jia et al., 2024; Embodiment Collaboration et al., 2024; Jiang et al., 2023; Mandlekar et al., 2023]. This development bears the potential to produce robotics agents that learn to master complex tasks in a wide range of environments and even different embodiments. For example, recently it has been demonstrated, albeit in restricted settings, that sequence models trained on multi-episodic contexts can perform in-context learning (ICL) [Laskin et al., 2020; Lee et al., 2023]. One potential application of ICL can be to learn new related tasks in robotics without the need for re-training or fine-tuning.

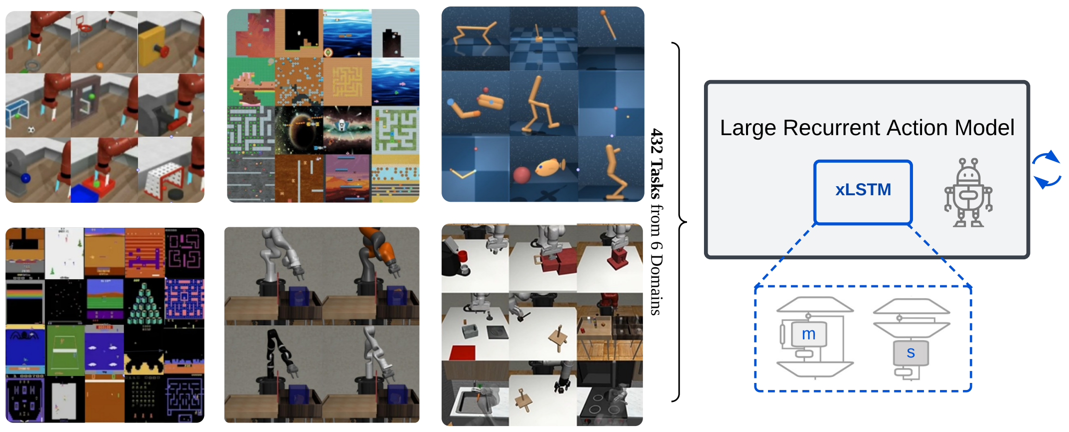


Figure 1: Illustration of our Large Recurrent Action Model (LRAM) with an xLSTM [Beck et al., 2024] at its core.

One of the key reasons for the success of Transformer-based models is their ability to scale to large datasets through their efficient parallelization during training. However, despite numerous success stories in RL, language modeling [Brown et al., 2020] or computer vision [Dosovitskiy et al., 2021; He et al., 2022], a persistent drawback of Transformer-based architectures is their high inference cost in terms of both speed and memory [Kim et al., 2023]. Consequently, deploying Transformer-based models in resource-constrained scenarios, such as on devices with limited hardware capacity and/or real-time constraints, e.g., robots or smartphones, is difficult because of the required fast inference times [Firoozi et al., 2023; Hu et al., 2023]. A basic principle of control theory is that the controller sample rate should be at least in the order of magnitude of the sample rate of the sensors [Franklin et al., 1998, Ch. 11]. To illustrate this, for typical robots such as drones or industrial robot arms, rates of 100Hz-1000Hz are necessary to keep the system stable [Salzmann et al., 2023; El-Hussieny, 2024; Hu et al., 2023; Chignoli et al., 2021]. This implies inference times of less than 10ms. At 1000Hz, a 15-second movement of the agent corresponds to a sequence of 15K steps [El-Hussieny, 2024] resulting in long context lengths even without ICL. While there exists a range of techniques to make large models faster, such as quantization [Frantar et al., 2023], distillation [Hinton et al., 2015], or pruning [LeCun et al., 1989], the quadratic-time complexity of self attention still remains.

Recently, *modern recurrent architectures* have been proposed, which exhibit similar parallelization properties during training as the Transformer architecture while offering linear-time inference complexity. These modern recurrent architectures include xLSTM [Beck et al., 2024] and state-space models (SSMs), such as Mamba [Gu and Dao, 2023; Dao and Gu, 2024] and Griffin/Hawk [De et al., 2024], and have challenged the dominance of the Transformer in language modeling but also in other domains such as computer vision [Alkin et al., 2024; Zhu et al., 2024], and biomedicine [Schmidinger et al., 2024]. More importantly, their linear-time inference makes them suitable for deployment in scenarios with limited compute, large context sizes, and real-time requirements, such as robotics.

In this work, we assess the aptitude of modern recurrent architectures, such as xLSTM and Mamba, as large action models. To this end, we introduce a Large Recurrent Action Model (LRAM) with an xLSTM at its core (see Figure 1). We train our agents on 432 tasks from 6 domains using a supervised learning setting similar to that of the Decision Transformer [Chen et al., 2021, DT]. We use data collected during online-RL training of single-task specialist agents and compile these trajectories alongside other expert demonstrations into a large-scale multi-domain dataset comprising 894M transitions. Due to their parallelization properties, the modern recurrent architectures considered in this work can process this large-scale training set as efficiently as the Transformer while being faster at inference. Experiments across 4 models sizes with our multi-task models indicate that xLSTM compares favorably to Transformers in terms of both performance and speed. In addition, we study the effect of modern recurrent architectures on fine-tuning performance and in-context learning abilities, and find that they exhibit strong performance in both dimensions.

The main purpose of this paper is to test the hypothesis that *modern recurrent model architectures are better suited for building LAMs than Transformers*. Hereby, we make the following **contributions**.

- We propose a Large Recurrent Action Model (LRAM) with an xLSTM at its core that enables efficient inference.
- We assess the aptitude of modern recurrent architectures as backbones for large-action models with respect to their efficiency at inference time and overall performance in multi-task, fine-tuning, and in-context learning settings.
- To foster further research on large action models, we release our data preparation pipeline and generated datasets[1].

2 Related work

Sequence Models in RL. LSTM [Hochreiter and Schmidhuber, 1997] is the dominant backbone architecture for partially observable online RL problems and has been behind achievements such as mastering Starcraft II [Vinyals et al., 2019], Dota 2 [Berner et al., 2019], and Atari [Espeholt et al., 2018; Kapturowski et al., 2019]. After the success of the Transformer in NLP [Devlin et al., 2019; Radford et al., 2019; Brown et al., 2020], computer vision [Dosovitskiy et al., 2021; He et al., 2022; Radford et al., 2021; Fürst et al., 2022] and speech recognition [Radford et al., 2022; Baevski et al., 2020], the architecture has found its way into RL. Chen et al. [2021] proposed the Decision Transformer (DT) a GPT-style model [Radford et al., 2018], that learns to predict actions from offline trajectories via behavior cloning. Trajectory Transformer [Janner et al., 2021] predicts action tokens along with states and rewards, which allows for dynamics modeling. A number of follow-up works build on the DT-architecture [Zheng et al., 2022; Wang et al., 2022; Shang et al., 2022; Meng et al., 2021; Siebenborn et al., 2022; Schmied et al., 2024a]. Furthermore, sequence models trained on offline data were found to exhibit in-context learning if conditioned on previous trajectories [Laskin et al., 2022; Lee et al., 2022; Kirsch et al., 2023], albeit only in limited scenarios.

Large Action Models (LAMs). LAMs, such as the Decision Transformer, are well suited for multi-task settings. Lee et al. [2022] found that a multi-game DT can learn to play 46 Atari games. Reed et al. [2022] introduced a generalist agent trained on over 600 tasks from different domains, ranging from Atari to manipulation of a robot arm. Jiang et al. [2022] a Transformer for robot manipulation based on multi-modal prompts, that allow to steer the model to perform new tasks. Recently, Raad et al. [2024] introduced an agent instructable via language to play a variety of commercial video games. Since then, robotics has become an emergent area for developing LAMs [Brohan et al., 2023b,a; Octo Model Team et al., 2024; Gu et al., 2023; Wang et al., 2023; Di Palo et al., 2023; Kim et al., 2024], also due to the availability of large-scale robotics datasets [Jia et al., 2024; Embodiment Collaboration et al., 2024; Jiang et al., 2023; Mandlekar et al., 2023].

Next-generation Sequence Modeling Architectures. Linear recurrent models, such as state-space models (SSM, Gu et al., 2021, 2022b; Smith et al., 2023; Orvieto et al., 2023) have challenged the dominance of the Transformer [Vaswani et al., 2017] architecture on long-range tasks [Tay et al., 2020]. The key insight of those linear RNNs was to diagonalize the recurrent state matrix and enforce stable training via an exponential parameterization [Gu et al., 2022a; Orvieto et al., 2023]. Since then, there have been efforts to include features such as gating from RNNs [Elman, 1990; Jordan, 1990; Hochreiter and Schmidhuber, 1997; Cho et al., 2014]. Non-linear gates are believed to have higher expressivity, but are harder to train. Griffin [De et al., 2024] mixes gated linear recurrences with local attention to achieve more training data efficiency than Llama-2 [Touvron et al., 2023] and better sequence extrapolation. Mamba [Gu and Dao, 2023] introduces a selection mechanism similar to gating into SSMs, which makes its state and input matrix time dependent. This is similar to the gating mechanism of RNNs but also bears resemblance to approaches like fast weights [Schmidhuber, 1992] and Linear Attention [Katharopoulos et al., 2020]. Mamba-2 [Dao and Gu, 2024] highlight the connection between SSMs with input dependent state and input matrices and (Gated) Linear attention variants. Most recently, the xLSTM [Beck et al., 2024] was proposed as an improvement over the classic LSTM [Hochreiter and Schmidhuber, 1997] that combines gating, linear recurrences and recurrent weights into a single architecture for language modeling. First, xLSTM leverages exponential gating with stabilization to RNNs for stronger emphasis on important inputs. Second, xLSTM is composed of two variants, the mLSTM variant with an emphasis on memory that proves important in language modeling and the sLSTM variant that keeps the non-diagonalized recurrent matrix to enable state-tracking [Merrill et al., 2024]. State tracking is important in logic tasks and

[1]GitHub: `https://github.com/ml-jku/LRAM`

cannot be modeled fundamentally by linearized recurrent or state-space models like Mamba, Griffin or Transformers.

3 Large Recurrent Action Models

3.1 Background

Reinforcement Learning. We assume the standard RL formulation via a Markov Decision Process (MDP) represented by a tuple of $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where $\mathcal{S}$ and $\mathcal{A}$ denote state and action spaces, respectively. At every timestep t the agent observes state $s_t \in \mathcal{S}$, predicts action $a_t \in \mathcal{A}$, and receives a scalar reward r_t. The reward is determined by the reward function $\mathcal{R}(r_t \mid s_t, a_t)$. $\mathcal{P}(s_{t+1} \mid s_t, a_t)$ defines the transition dynamics and constitutes a probability distribution over next states s_{t+1} when executing action a_t in state s_t. The goal of RL is to learn a policy $\pi(a_t \mid s_t)$ that predicts an action a_t in state s_t that maximizes r_t.

Decision Transformer [Chen et al., 2021] casts the RL problem setting as next action prediction task via causal sequence modeling. At training time, DT aims to learn a policy π_θ that maps future rewards to actions, which is often referred to as upside-down RL [Schmidhuber, 2019]. At inference time, the DT is conditioned via a target return to emit high-reward actions. Consequently, we assume access to a dataset $\mathcal{D} = \{\tau_i\}_{i=1}^N$ containing N trajectories τ_i consisting of quadruplets $\tau_i = (s_1, \hat{R}_1, a_1, r_1, \dots, s_T, \hat{R}_T, a_T, r_T)$ of state s_t, return-to-go (RTG) $\hat{R}_t = \sum_{t'=t}^T r_{t'}$, action a_t, and reward r_t. Here, T refers to the length of the trajectory. The DT π_θ is trained to predict the ground-truth action a_t conditioned on sub-trajectories from the dataset:

$$\hat{a}_t \sim \pi_\theta(\hat{a}_t \mid s_{t-C}, \hat{R}_{t-C}, a_{t-C}, r_{t-C}, \dots, s_{t-1}, \hat{R}_{t-1}, a_{t-1}, r_{t-1}, s_t, \hat{R}_t), \tag{1}$$

where $C \leq T$ is the size of the context window. In fact, Equation 1 describes the setting of the multi-game DT [Lee et al., 2022], which also includes rewards in the sequence representation.

3.2 Large Recurrent Action Models (LRAMs)

Our LRAM has a modern recurrent architecture at its core (see Figure 1), which comes with a parallel training and a recurrent inference mode. We instantiate LRAM with three different variants, two different xLSTM configurations and Mamba. Furthermore, we use a training protocol similar to that of Lee et al. [2022] and Reed et al. [2022] with some differences.

Multi-modal sequence representation. To encode input from different environments with varying state and action spaces, we use separate encoders per modality that are shared across tasks and domains. For encoding images we use a CNN similar to Espeholt et al. [2018], whereas for low-dimensional inputs we use a fully connected network. We refrain from patchifying images and tokenizing continuous states to avoid unnecessarily long sequences. Similarly, we use linear layers to encode rewards and RTGs. We omit actions in our sequence formulation, as we found that this can be detrimental to performance, in particular for continuous control tasks (see Section 4.3). Consequently, our trajectories have the form $\tau_i = (s_1, \hat{R}_1, r_1, \dots, s_T, \hat{R}_T, r_T)$ and we train our policy π_ρ to predict the ground-truth action a_t as:

$$\hat{a}_t \sim \pi_\rho(\hat{a}_t \mid s_{t-C}, \hat{R}_{t-C}, r_{t-C}, \dots, s_{t-1}, \hat{R}_{t-1}, r_{t-1}, s_t, \hat{R}_t). \tag{2}$$

Shared action head. Action spaces in RL typically vary across environments. For example, in the environments we consider, there are 18 discrete actions and a maximum of 8 continuous dimensions for continuous control environments. Therefore, we employ discretization of continuous action dimensions into 256 uniformly-spaced bins, similar to Reed et al. [2022] and Brohan et al. [2023b]. Unlike prior work, we leverage a shared action head to predict all discrete actions or continuous action dimensions jointly. We found this setup significantly reduces inference time compared to using autoregressive action prediction of continuous actions.

Recurrent inference mode. At inference time, we leverage the recurrent backbone and maintain the hidden states of the last timestep. This enables fast inference with linear-time complexity along the sequence length. In addition, the recurrent-style inference is well suited for online fine-tuning via RL objectives, similar to LSTM-based policies in online RL. To further speed-up inference, we leverage custom kernels for the xLSTM backbone (see Appendix 22).

Table 1: Dataset statistics for all 432 training tasks.

Dataset	Tasks	Trajectories	Mean Trj. Length	Total Transitions	Repetitions
Atari	41	136K	2733	205M	1.03×
Composuite	240	480K	500	240M	0.87×
DMControl	11	110K	1000	110M	1.92×
Meta-World	45	450K	200	90M	2.34×
Mimicgen	83	83K	300	25M	8.5×
Procgen	12	2185K	144	224M	0.94×
Total	432	3.4M	-	894M	-

Our unified discrete action representation enables consistent training of our agents via the cross-entropy loss as training objective across all tasks and domains, similar to Reed et al. [2022]. We use separate reward scales per domain and target returns per task. Furthermore, we do not make use of timestep encodings as used by Chen et al. [2021], which are detrimental when episode lengths vary. We provide additional implementation details in Appendix B.

4 Experiments

We study the aptitude of modern recurrent architectures as LAMs on 432 tasks from 6 domains: Atari [Bellemare et al., 2013], Composuite [Mendez et al., 2022], DMControl [Tassa et al., 2018], Meta-World [Yu et al., 2020b], Mimicgen [Mandlekar et al., 2023], and Procgen [Cobbe et al., 2020a]. To this end, we compile a large-scale dataset containing 894 million transitions (see Section 4.1).

Across all experiments, we compare four backbone variants: xLSTM [7:1], xLSTM [1:0] [Beck et al., 2024], Mamba [Gu and Dao, 2023], and the GPT-2 style Transformer employed in the DT [Chen et al., 2021]. Following [Beck et al., 2024], we use the bracket notation for xLSTM, which indicates the ratio of mLSTM to sLSTM blocks. For example, xLSTM [1:0] contains only mLSTM blocks.

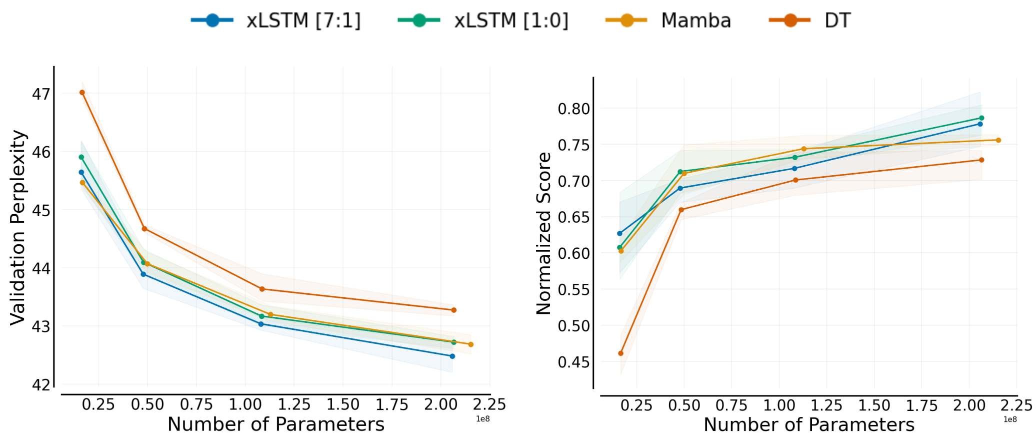
In Section 4.2, we conduct a scaling comparison for four model sizes ranging from 16M to 208M parameters that shows that modern recurrent architectures achieve performance comparable or favorable to the Transformer baseline across different model sizes. In Section 4.3, we study the impact of the recurrent backbones on fine-tuning performance and ICL abilities, and further analyze our trained recurrent backbones. Finally, in Section 4.4, we empirically examine the differences at inference time in terms of latency and throughput between xLSTM-based and Transformer-based agents, which indicate a clear advantage for the recurrent backbone.

4.1 Datasets & Environments

Datasets. We compile a large-scale dataset comprising 432 tasks from six domains. We leverage datasets from prior works. For Atari, we extract 5M transitions per task from the DQN-Replay dataset released by Agarwal et al. [2020]. For Composuite, we leverage the datasets released by [Hussing et al., 2023]. For Meta-World, we use 2M transitions per task released by [Schmied et al., 2024a]. For DMControl, we generate 10M transitions per task using task-specific RL agents. For Mimicgen, we use the datasets for the 21 tasks released by [Mandlekar et al., 2023] and generate trajectories for the remaining 62 tasks. Finally, for Procgen, we extract 20M transitions from the datasets released by [Schmied et al., 2024b]. Our final dataset contains 3.4M trajectories and in total 894M transitions (see Table 4.1). We reserve an additional 37 tasks from the same domains for zero-shot evaluation. To foster future research, we release our data-preparation pipeline and generated data.

Environments. Atari and Procgen come with image observations and discrete actions. In contrast, the remaining four domains exhibit state-based observations and continuous actions. Consequently, our experiments involve a mixture of state and action spaces as well as varying episode lengths (see Table 4.1). Periodically evaluating the trained agents on all 432 tasks sequentially is time-consuming and we, therefore, distributed the evaluation across GPUs and parallel processes (see Appendix B).

Additional details on our datasets, environments are available in Appendix A.



(a) Sequence prediction (b) Environment interaction

Figure 2: Scaling comparison. We compare xLSTM, Mamba, DT in four model sizes: 16M, 48M, 110M, and 206M parameters. We show the **(a)** validation perplexity on the hold-out datasets, and **(b)** normalized scores obtained from evaluating in the training task environments, averaged over all 6 domains.

4.2 Scaling comparison

To conduct our main comparisons, we train our four backbone variants on the full training task mixture of 432 tasks. For each architecture backbone, we report performance scores for four model sizes: 16M, 48M, 108M, and 206M parameters. We train all models for 200K updates with a batch size of 128 and context length of 50 timesteps. All domains are represented with approximately equal proportion, resulting in 33K updates per domain. Additional implementation details and hyperparameters for every backbone variant and model size are available in Appendix B.

Sequence prediction performance. In Figure 2a, we report the validation set perplexity for all backbones and model sizes averaged over the individual scores from all domains. To achieve this, we maintain a hold-out set of trajectories for each training task (2.5%) and compute the perplexities after every 50K steps. Both recurrent backbones outperform the Transformer baseline considerably, especially as the model sizes increase. We provide the perplexities on the training set in Figure 13.

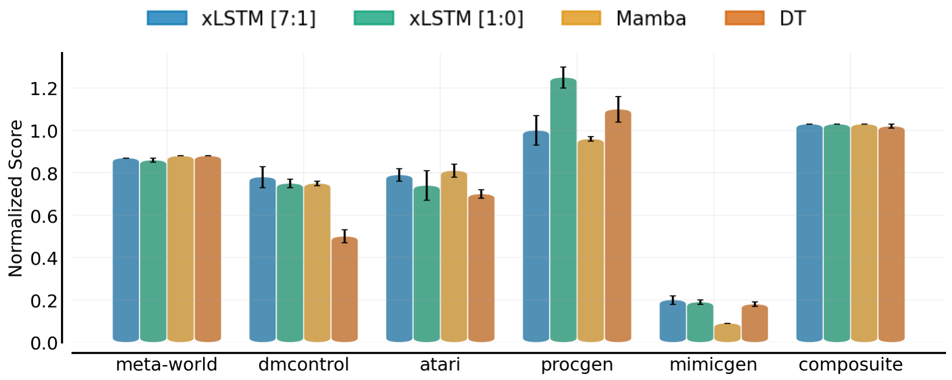


Figure 3: Normalized scores per domain for model size 206M. For Meta-World, DMControl, Mimicgen, Composuite and Procgen we report data-normalized scores, for Atari we report human-normalized scores.

Evaluation performance. During training, we evaluate our agents after every 50K step in all 432 training environments. In Figure 2b, we report the resulting normalized performances averaged across all six domains. The recurrent backbones outperform the Transformer one across model sizes. While xLSTM and Mamba performs similarly at smaller scales, xLSTM tends to outperform Mamba at larger scales (206M). This is an important advantage of xLSTM, as LRAM agents can strongly benefit from more data and consequently larger models. Note, that Mamba has a significantly higher number of parameters than competitors. For the zero-shot evaluation performances on the 37 hold-out tasks, we refer to Figure 15 in Appendix C.2.

Performance per domain. In Figure 3, we report the normalized scores for the 206M parameter models attained on all six domains. For Meta-World, DMControl, Mimicgen, Composuite, and Procgen we use data-normalized scores, as suggested by [Levine et al., 2020]. For Atari, we report human-normalized scores. Overall, we observe that the xLSTM backbone outperforms competitors on three of the six domains, while all methods perform similarly on the remaining 3 domains.

These experiments suggest that modern recurrent backbones can be attractive alternatives to the Transformer architecture for building LAMs in terms of final performance.

4.3 Analyses & Ablations

Fine-tuning. To assess the effect of the recurrent backbones on fine-tuning performance, we fine-tune our models on 37 held-out environments from all 6 domains. We evaluate the fine-tuning performance of the xLSTM architecture for both the 16M parameter pretrained models and compared it against an xLSTM trained from scratch. The pretrained LRAM outperforms the randomly initialized xLSTM model in most domains. For detailed results, see Appendix C.3. This suggests that fine-tuning performance is not negatively affected by switching the backbone.

In-context Learning. Next, we study the ICL abilities of our recurrent backbones on the Dark-Room environment considered in prior work on in-context RL [Laskin et al., 2022; Lee et al., 2023; Schmied et al., 2024b]. To study ICL in isolation, we train models from scratch with a multi-episodic context, which results in a large context length (we refer to Appendix C.4 for details on the experiment setup). In particular, we adopt the Algorithm Distillation (AD, Laskin et al., 2022) framework and exchange the Transformer backbone architecture with modern recurrent architectures. In Figure 17, we report the ICL performance on (a) 80 train and (b) 20 hold-out tasks. We find that xLSTM [7:1] attains the highest overall scores both on training and hold-out tasks, which we attribute to the state-tracking abilities [Merrill et al., 2024] of sLSTM blocks.

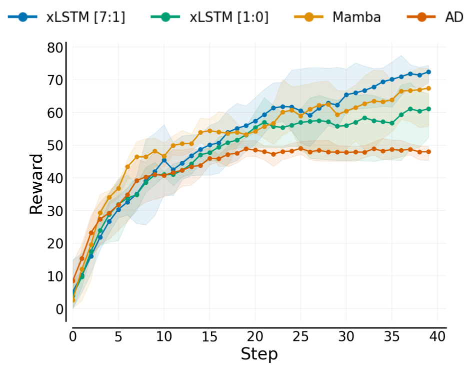
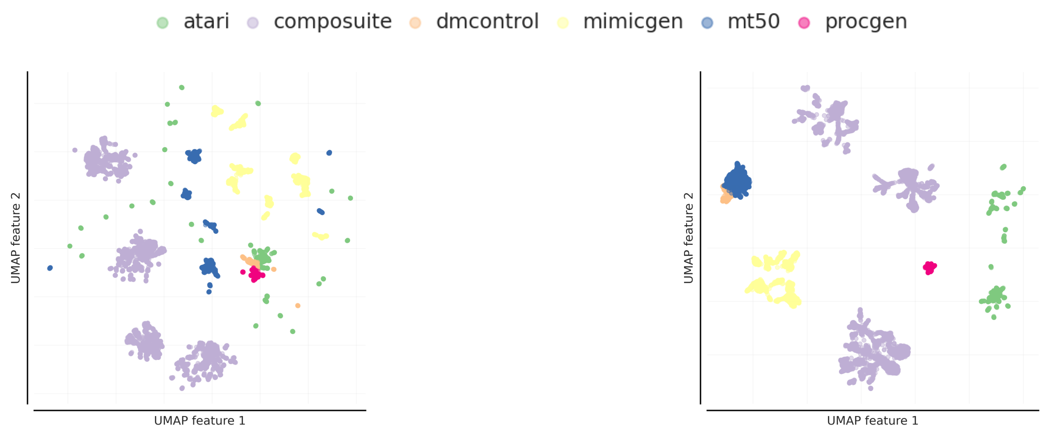


Figure 4: ICL with modern recurrent architectures on Dark-Room 10×10.

Embedding space analysis. In Figure 5, we analyze the representations learned by our model. To this end, we sample 32 sub-trajectories from every task, extract the sequence representation at the last layer, cluster them using UMAP [McInnes et al., 2018], and color every point by its domain. Appendix E describes the setup in greater detail. We find that tasks from the same domain cluster together. Furthermore, xLSTM exhibits a more refined domain separation compared to DT, which may contribute to the better down-stream performance.

Removing Actions & Effect of Context Length. We found that removing actions from the context results in better performance across backbones. While context lengths beyond 1 hurt performance on Meta-World when training with actions, the reverse is true when training without actions (see Figure 23). This is in contrast to recent works, which did not benefit from longer contexts [Octo Model Team et al., 2024]. While removing actions improves performance on the robotics domains, it does not affect performance on discrete control. For robotics, we observed that the models become overly confident (high action logits), which is problematic if poor initial actions are produced. We assume this is because in robotics actions change smoothly and by observing previous actions the agent learns shortcuts. Removing actions from the input prevents the agent from using shortcuts. Importantly, the evaluation performance improves as the sequence length increases, which indicates that the history helps to predict the next action (e.g., by observing mistakes made in the recent past).

We present additional ablations on the effect of reducing the number of layers in xLSTM and disabling Dropout on DT in Appendix D.3 and D.2, respectively.

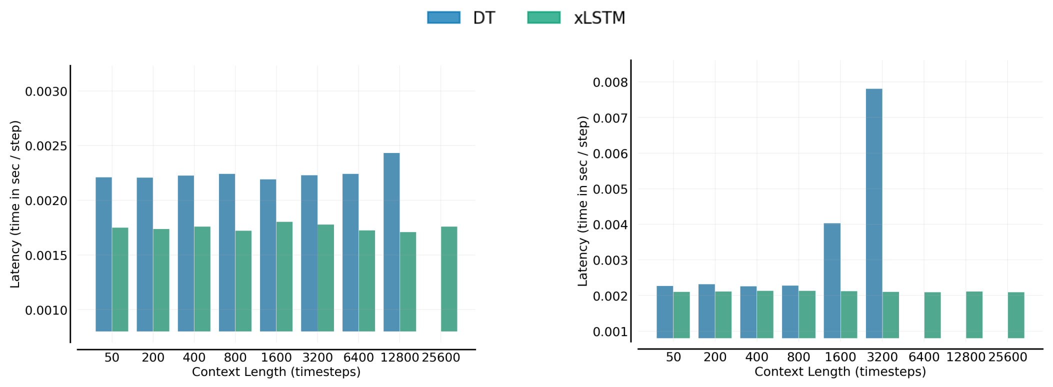


(a) DT (b) xLSTM

Figure 5: Embedding space comparison. UMAP clustering of hidden states for all tasks for 16M models, colored by domain. xLSTM exhibits a better domain separation than DT.

4.4 Inference Time Comparison

Finally, we empirically examine the difference between xLSTM-based and Transformer-based agents at inference time. Similar to De et al. [2024], we report both latency and throughput. We focus our analysis on latency, as it is the more important dimension for real-time applications.



(a) $B = 1$ (b) $B = 16$

Figure 6: Latency comparison on A100. We report latency for varying context lengths (in timesteps) with fixed batch sizes B of 1 and 16. We compare DT to xLSTM with the same number of layer blocks and parameters on Atari `Freeway`. Missing bars for DT indicate out-of-memory (OOM).

Setup. We conduct all inference time tests on A100 GPUs with 40GB of RAM using 206M parameter models. For the Transformer, we use KV-caching and FlashAttention [Dao, 2023] as supported by PyTorch [Paszke et al., 2019]. For xLSTM, we use recurrent-style inference using custom kernels to accelerate the computations (see Figure 22 for the impact of kernel acceleration). For both backbones, we use `torch.compile`. The Transformer with KV-caching has a linear time complexity per step and quadratic in the sequence length. In contrast, the xLSTM has a constant time complexity per step and linear in the sequence length. Therefore, we expect speed-ups especially for longer sequences and larger batch sizes, as observed by De et al. [2024]. To ensure a fair comparison, we compare DT and xLSTM with the same number of layer blocks and increase the hidden size of xLSTM to match the number of parameters of DT (see Appendix D.3 for evaluation performance of these models). We provide further details on our inference time tests in Appendix C.5.

Environment. We conduct all inference time tests on the environment that exhibited the longest average episode lengths in our experiments, the Atari game `Freeway`. Every episode in `Freeway` lasts for 8192 steps, which is equivalent to 24576 tokens (s/rtg/r). We evaluate all models for 5 episodes and preserve the KV-cache/hidden state across episode boundaries. The reported latencies and throughputs are averaged across all evaluation episodes, except for the first episode, which we discard to exclude compilation times and prefilling. We opted for measuring the inference times during environment interaction, i.e., including simulator latency, rather than mere token generation.

Latency. Similar to De et al. [2024], we measure latency by the average time (in seconds) taken to perform a single inference step with a fixed batch size B (lower is better). In Figure, 6, we report the latencies for varying context lengths, $C \in [50, 25600]$ and two batch sizes $B \in \{1, 16\}$. Note that C is in time steps and every time step contains 3 tokens (state, reward-to-go, reward). Hence, the effective sequence length for the largest C is 76800. As expected, we find that the recurrent backbone attains lower inference latencies than the Transformer one. As the sequence length increases, DT runs out of memory due to the increasing size of the KV cache (see Figure 7). In contrast, the inference speeds for xLSTM are independent of the context length, and therefore enable significantly longer context lengths. This property is particularly interesting for in-context RL, which requires keeping multiple episodes in the context [Laskin et al., 2022]. Nevertheless, our experiments highlight that the materialization of the complexity advantage (quadratic vs. linear) depends on the device, model size, batch size and the context length, which is similar to findings by De et al. [2024].

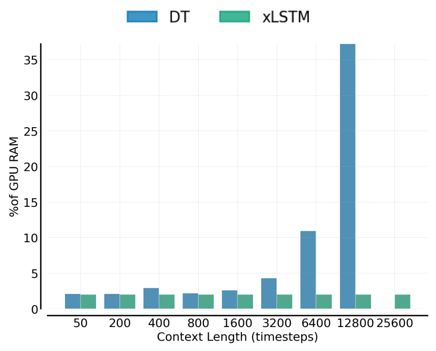


Figure 7: Memory consumption during **Latency** comparison on A100 (% of GPU memory) for varying context lengths and $B = 1$.

Throughput. Throughput is measured by the total amount of inference steps performed per second for a model with a fixed context length. In Figure, 8, we report the throughputs for varying batch sizes, $B \in [1, 128]$ for a fixed context length of $C = 1600$. Here, the batch size can be interpreted as the number of parallel environments the agent interacts with. As expected, we find that xLSTM attains considerably higher throughputs than the DT. The benefit of xLSTM increases with larger batch sizes. While the DT with quadratic complexity in the sequence length goes OOM for batch sizes above 64, the xLSTM with linear complexity can easily handle larger batch sizes. In both experiments, the recurrent xLSTM performs favorably over the Transformer backbone.

5 Conclusion

In this work, we study the aptitude of modern recurrent architectures as alternatives to Transformers for building LAMs. We found that our LRAM with an xLSTM or Mamba at its core compare favorably to the Transformer in terms of evaluation performance across different model scales. Moreover, we demonstrated that xLSTM-based LRAMs exhibit higher inference speeds, especially at large context sizes. Thus, the empirical evidence suggests, that recurrent backbones such as the xLSTM can be attractive alternatives for LAMs. Notably, the linear-time inference complexity of xLSTM may enable applications that require long context lengths, such as in-context RL, and facilitate the application of large-scale agents for real-time applications, such as robotics.

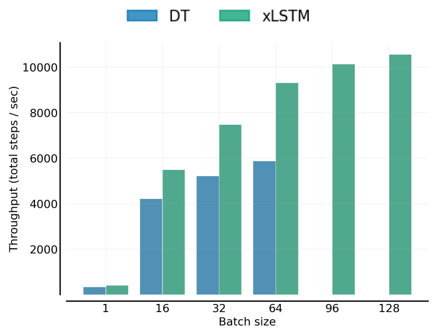


Figure 8: Throughput comparison on A100 for varying batch sizes with $C = 1600$ timesteps on the Atari `Freeway` environment. Missing bars for DT indicate OOM.

Limitations. The primary target application of LAMs is robotics. While the majority of our experiments involve robotic simulations, we do not yet provide empirical evidence for real robots. We do, however, believe that our findings translate to real-world scenarios and aim to provide further evidence in future work. Moreover, the fine-tuning experiments in this work are limited to offline RL. We envision that an agent pre-trained by behavioral cloning on large-scale offline RL datasets may be successfully fine-tuned in an online RL setting to explore new strategies that do not appear in the training data. Modern recurrent architectures offer both parallel and recurrent training mode, which might be the key to success for such applications. While we provide initial evidence of improved ICL abilities of modern recurrent architectures, we only consider a limited grid-world setting. Consequently, we aim to further investigate the in-context RL abilities of recurrent backbones on more complex environments in future work.

6 Ethics Statement

While we conduct all our experiments in simulated environments, the primary target application of our method is robotics. We believe that our work can positively impact applications in the near future, which require efficient inference, on-device processing, or have real-time constraints. However, robotics applications in the real world are not without risks. In particular, in areas where humans are involved, such as factory settings, special care is required. LAMs are trained via next-action prediction similar to LLMs. Consequently, LAMs may also suffer from hallucinations in unknown scenarios. We therefore strongly discourage users from blindly following the predictions made by real-world LAMs without appropriate safeguards regarding safety and robustness. It is essential to ensure responsible deployment of such future technologies, and we believe that more research on the robustness of LAMs is necessary.

7 Reproducibility

Our code-base used for our experiments and the datasets we generated are publicly available at: `https://github.com/ml-jku/LRAM`. We describe the environments we use for our experiments and provide dataset statistics in Appendix A. Furthermore, in Appendix B, we provide implementation details for all methods and a list of hyperparameters used for our experiments. In Appendix C, we present additional figures that accompany our results in the main text (e.g., all model sizes). Finally, in Appendices D and E, we provide further details on the conducted ablation studies and the embedding space analysis, respectively.

Acknowledgments and Disclosure of Funding

We acknowledge EuroHPC Joint Undertaking for awarding us access to Karolina at IT4Innovations, Czech Republic, MeluXina at LuxProvide, Luxembourg, and Leonardo at CINECA, Italy. The ELLIS Unit Linz, the LIT AI Lab, the Institute for Machine Learning, are supported by the Federal State Upper Austria. We thank the projects Medical Cognitive Computing Center (MC3), INCONTROL-RL (FFG-881064), PRIMAL (FFG-873979), S3AI (FFG-872172), DL for GranularFlow (FFG-871302), EPILEPSIA (FFG-892171), AIRI FG 9-N (FWF-36284, FWF-36235), AI4GreenHeatingGrids (FFG- 899943), INTEGRATE (FFG-892418), ELISE (H2020-ICT-2019-3 ID: 951847), Stars4Waters (HORIZON-CL6-2021-CLIMATE-01-01). We thank NXAI GmbH, Audi.JKU Deep Learning Center, TGW LOGISTICS GROUP GMBH, Silicon Austria Labs (SAL), FILL Gesellschaft mbH, Anyline GmbH, Google, ZF Friedrichshafen AG, Robert Bosch GmbH, UCB Biopharma SRL, Merck Healthcare KGaA, Verbund AG, GLS (Univ. Waterloo), Software Competence Center Hagenberg GmbH, Borealis AG, TÜV Austria, Frauscher Sensonic, TRUMPF and the NVIDIA Corporation.

References

Agarwal, R., Schuurmans, D., and Norouzi, M. (2020). An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pages 104–114. PMLR.

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., and Bellemare, M. (2021). Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320.

Alkin, B., Beck, M., Pöppel, K., Hochreiter, S., and Brandstetter, J. (2024). Vision-lstm: xlstm as generic vision backbone. *CoRR*, abs/2406.04303.

Baevski, A., Zhou, Y., Mohamed, A., and Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460.

Beck, M., Pöppel, K., Spanring, M., Auer, A., Prudnikova, O., Kopp, M., Klambauer, G., Brandstetter, J., and Hochreiter, S. (2024). xlstm: Extended long short-term memory. *CoRR*, abs/2405.04517.

Bellemare, M. G., Candido, S., Castro, P. S., Gong, J., Machado, M. C., Moitra, S., Ponda, S. S., and Wang, Z. (2020). Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588(7836):77–82.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The Arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279.

Berner, C., Brockman, G., Chan, B., Cheung, V., Dkebiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al. (2019). Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*.

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., Ding, T., Driess, D., Dubey, A., Finn, C., et al. (2023a). Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*.

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Dabis, J., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jackson, T., Jesmonth, S., Joshi, N. J., Julian, R., Kalashnikov, D., Kuang, Y., Leal, I., Lee, K., Levine, S., Lu, Y., Malla, U., Manjunath, D., Mordatch, I., Nachum, O., Parada, C., Peralta, J., Perez, E., Pertsch, K., Quiambao, J., Rao, K., Ryoo, M. S., Salazar, G., Sanketi, P. R., Sayed, K., Singh, J., Sontakke, S., Stone, A., Tan, C., Tran, H. T., Vanhoucke, V., Vega, S., Vuong, Q., Xia, F., Xiao, T., Xu, P., Xu, S., Yu, T., and Zitkovich, B. (2023b). RT-1: robotics transformer for real-world control at scale. In Bekris, K. E., Hauser, K., Herbert, S. L., and Yu, J., editors, *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., 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. (2020). Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. (2021). Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097.

Chignoli, M., Kim, D., Stanger-Jones, E., and Kim, S. (2021). The mit humanoid robot: Design, motion planning, and control for acrobatic behaviors. In *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, pages 1–8. IEEE.

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Moschitti, A., Pang, B., and Daelemans, W., editors, *Proceedings of the 2014*

Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL, pages 1724–1734. ACL.

Cobbe, K., Hesse, C., Hilton, J., and Schulman, J. (2020a). 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*, pages 2048–2056. PMLR.

Cobbe, K., Hesse, C., Hilton, J., and Schulman, J. (2020b). Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pages 2048–2056. PMLR.

Dao, T. (2023). Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Dao, T. and Gu, A. (2024). Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*.

De, S., Smith, S. L., Fernando, A., Botev, A., Cristian-Muraru, G., Gu, A., Haroun, R., Berrada, L., Chen, Y., Srinivasan, S., et al. (2024). Griffin: Mixing gated linear recurrences with local attention for efficient language models. *arXiv preprint arXiv:2402.19427*.

Degrave, J., Felici, F., Buchli, J., Neunert, M., Tracey, B., Carpanese, F., Ewalds, T., Hafner, R., Abdolmaleki, A., de Las Casas, D., et al. (2022). Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., and Solorio, T., editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Di Palo, N., Byravan, A., Hasenclever, L., Wulfmeier, M., Heess, N., and Riedmiller, M. (2023). Towards a unified agent with foundation models. *arXiv preprint arXiv:2307.09668*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

El-Hussieny, H. (2024). Real-time deep learning-based model predictive control of a 3-dof biped robot leg. *Scientific Reports*, 14(1):16243.

Elman, J. L. (1990). Finding structure in time. *Cogn. Sci.*, 14(2):179–211.

Embodiment Collaboration, O'Neill, A., Rehman, A., Maddukuri, A., Gupta, A., Padalkar, A., Lee, A., Pooley, A., Gupta, A., Mandlekar, A., Jain, A., Tung, A., Bewley, A., Herzog, A., Irpan, A., Khazatsky, A., Rai, A., Gupta, A., Wang, A., Singh, A., Garg, A., Kembhavi, A., Xie, A., Brohan, A., Raffin, A., Sharma, A., Yavary, A., Jain, A., Balakrishna, A., Wahid, A., Burgess-Limerick, B., Kim, B., Schölkopf, B., Wulfe, B., Ichter, B., Lu, C., Xu, C., Le, C., Finn, C., Wang, C., Xu, C., Chi, C., Huang, C., Chan, C., Agia, C., Pan, C., Fu, C., Devin, C., Xu, D., Morton, D., Driess, D., Chen, D., Pathak, D., Shah, D., Büchler, D., Jayaraman, D., Kalashnikov, D., Sadigh, D., Johns, E., Foster, E., Liu, F., Ceola, F., Xia, F., Zhao, F., Stulp, F., Zhou, G., Sukhatme, G. S., Salhotra, G., Yan, G., Feng, G., Schiavi, G., Berseth, G., Kahn, G., Wang, G., Su, H., Fang, H., Shi, H., Bao, H., Amor, H. B., Christensen, H. I., Furuta, H., Walke, H., Fang, H., Ha, H., Mordatch, I., Radosavovic, I., Leal, I., Liang, J., Abou-Chakra, J., Kim, J., Drake, J., Peters, J., Schneider, J., Hsu, J., Bohg, J., Bingham, J., Wu, J., Gao, J., Hu, J., Wu, J., Wu, J., Tan, J., Oh, J., Wu, J., Lu, J., Yang, J., Salvador, J., Lim, J. J., Han, J., Wang, K., Rao, K., Pertsch, K., Hausman, K., Go, K., Gopalakrishnan, K., Goldberg, K., Byrne, K., Kawaharazuka, K., Black, K., Lin, K., Zhang, K., Ehsani, K., Lekkala, K., Ellis, K., Rana, K., Fang, K., Singh, K., Zeng, K., Hatch, K., Hsu, K., Itti, L., Chen, L. Y., Pinto, L., Fei-Fei, L., Tan, L., Fan, L., Ott, L., Lee, L., Weihs, L., Chen, M., Lepert, M., Memmel, M., Tomizuka, M., Itkina, M., Castro, M. G., Spero, M., Du, M., Ahn, M., Yip, M. C., Zhang, M., Ding, M., Heo, M., Srirama, M. K., Sharma, M., Kim, M. J., Kanazawa, M., Hansen, N., Heess, N., Joshi, N. J., Suenderhauf, N., Liu, N., Palo, N. D., Shafiullah, N., Mees, O., Kroemer, O., Bastani, O., Sanketi, P. R., Miller, P., Yin, P., Wohlhart, P., Xu, P., Fagan, P., Mitrano, P., Sermanet, P., Abbeel, P., Sundaresan, P., Chen, Q., Vuong, Q., Rafailov, R., Tian, R., Doshi, R., Martín-Martín, R., Baijal, R., Scalise, R., Hendrix, R., Lin, R., Qian, R., Zhang, R., Mendonca, R., Shah, R., Hoque, R., Julian, R., Bustamante, S., Kirmani, S., Levine, S., Lin, S., Moore, S., Bahl, S., Dass, S., Sonawani, S., Song, S., Xu, S., Haldar, S., Karamcheti, S., Adebola, S., Guist, S., Nasiriany, S., Schaal, S., Welker, S., Tian, S., Ramamoorthy, S., Dasari, S., Belkhale, S., Park, S., Nair, S., Mirchandani, S., Osa, T., Gupta, T., Harada, T., Matsushima, T., Xiao, T., Kollar, T., Yu, T., Ding, T., Davchev, T., Zhao, T. Z., Armstrong, T., Darrell, T., Chung, T., Jain, V., Vanhoucke, V., Zhan, W., Zhou, W., Burgard, W., Chen, X., Wang, X., Zhu, X., Geng, X., Liu, X., Liangwei, X., Li, X., Lu, Y., Ma, Y., Kim, Y., Chebotar, Y., Zhou, Y., Zhu, Y., Wu, Y., Xu, Y., Wang, Y., Bisk, Y., Cho, Y., Lee, Y., Cui, Y., Cao, Y., Wu, Y., Tang, Y., Zhu, Y., Zhang, Y., Jiang, Y., Li, Y., Li, Y., Iwasawa, Y., Matsuo, Y., Ma, Z., Xu, Z., Cui, Z., Zhang, Z., Fu, Z., and Lin, Z. (2024). Open x-embodiment: Robotic learning datasets and rt-x models.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al. (2018). Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pages 1407–1416. PMLR.

Firoozi, R., Tucker, J., Tian, S., Majumdar, A., Sun, J., Liu, W., Zhu, Y., Song, S., Kapoor, A., Hausman, K., et al. (2023). Foundation models in robotics: Applications, challenges, and the future. *The International Journal of Robotics Research*, page 02783649241281508.

Franklin, G. F., Powell, J. D., Workman, M. L., et al. (1998). *Digital control of dynamic systems*, volume 3. Addison-wesley Menlo Park.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. (2023). OPTQ: accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Fürst, A., Rumetshofer, E., Lehner, J., Tran, V., Tang, F., Ramsauer, H., Kreil, D., Kopp, M., Klambauer, G., Bitto-Nemling, A., and Hochreiter, S. (2022). Cloob: Modern hopfield networks with infoloob outperform clip.

Gu, A. and Dao, T. (2023). Mamba: Linear-time sequence modeling with selective state spaces. *CoRR*, abs/2312.00752.

Gu, A., Goel, K., Gupta, A., and Ré, C. (2022a). On the parameterization and initialization of diagonal state space models. *Advances in Neural Information Processing Systems*, 35:35971–35983.

Gu, A., Goel, K., and Ré, C. (2022b). Efficiently modeling long sequences with structured state spaces. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Gu, A., Johnson, I., Goel, K., Saab, K., Dao, T., Rudra, A., and Ré, C. (2021). Combining recurrent, convolutional, and continuous-time models with linear state space layers. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 572–585.

Gu, J., Kirmani, S., Wohlhart, P., Lu, Y., Arenas, M. G., Rao, K., Yu, W., Fu, C., Gopalakrishnan, K., Xu, Z., Sundaresan, P., Xu, P., Su, H., Hausman, K., Finn, C., Vuong, Q., and Xiao, T. (2023). Rt-trajectory: Robotic task generalization via hindsight trajectory sketches.

Gu, X., Wang, Y.-J., and Chen, J. (2024). Humanoid-gym: Reinforcement learning for humanoid robot with zero-shot sim2real transfer.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Dy, J. G. and Krause, A., editors, *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*, pages 1856–1865. PMLR.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. (2019). Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. B. (2022). Masked autoencoders are scalable vision learners. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 15979–15988. IEEE.

Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M. G., and Silver, D. (2017). Rainbow: Combining improvements in deep reinforcement learning. *ArXiv*.

Hinton, G. E., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2022). Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Hu, Y., Xie, Q., Jain, V., Francis, J., Patrikar, J., Keetha, N., Kim, S., Xie, Y., Zhang, T., Zhao, Z., et al. (2023). Toward general-purpose robots via foundation models: A survey and meta-analysis. *arXiv preprint arXiv:2312.08782*.

Huang, S., Hu, J., Chen, H., Sun, L., and Yang, B. (2024). In-context decision transformer: Reinforcement learning via hierarchical chain-of-thought. *arXiv preprint arXiv:2405.20692*.

Hussing, M., Mendez, J. A., Singrodia, A., Kent, C., and Eaton, E. (2023). Robotic manipulation datasets for offline compositional reinforcement learning. *arXiv preprint arXiv:2307.07091*.

Janner, M., Li, Q., and Levine, S. (2021). Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286.

Jia, X., Blessing, D., Jiang, X., Reuss, M., Donat, A., Lioutikov, R., and Neumann, G. (2024). Towards diverse behaviors: A benchmark for imitation learning with human demonstrations. In *The Twelfth International Conference on Learning Representations*.

Jiang, Y., Gupta, A., Zhang, Z., Wang, G., Dou, Y., Chen, Y., Fei-Fei, L., Anandkumar, A., Zhu, Y., and Fan, L. (2022). Vima: General robot manipulation with multimodal prompts. *arXiv preprint arXiv:2210.03094*.

Jiang, Y., Gupta, A., Zhang, Z., Wang, G., Dou, Y., Chen, Y., Fei-Fei, L., Anandkumar, A., Zhu, Y., and Fan, L. (2023). Vima: General robot manipulation with multimodal prompts.

Jordan, M. I. (1990). *Attractor dynamics and parallelism in a connectionist sequential machine*, page 112–127. IEEE Press.

Kapturowski, S., Ostrovski, G., Dabney, W., Quan, J., and Munos, R. (2019). Recurrent experience replay in distributed reinforcement learning. In *International Conference on Learning Representations*.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. (2020). Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR.

Kim, M. J., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., Rafailov, R., Foster, E., Lam, G., Sanketi, P., et al. (2024). Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*.

Kim, S., Hooper, C., Wattanawong, T., Kang, M., Yan, R., Genc, H., Dinh, G., Huang, Q., Keutzer, K., Mahoney, M. W., et al. (2023). Full stack optimization of transformer inference: a survey. *arXiv preprint arXiv:2302.14017*.

Kirsch, L., Harrison, J., Freeman, C., Sohl-Dickstein, J., and Schmidhuber, J. (2023). Towards general-purpose in-context learning agents. In *NeurIPS 2023 Workshop on Generalization in Planning*.

Laskin, M., Lee, K., Stooke, A., Pinto, L., Abbeel, P., and Srinivas, A. (2020). Reinforcement learning with augmented data. *ArXiv*, 2004.14990.

Laskin, M., Wang, L., Oh, J., Parisotto, E., Spencer, S., Steigerwald, R., Strouse, D., Hansen, S., Filos, A., Brooks, E., et al. (2022). In-context reinforcement learning with algorithm distillation. *arXiv preprint arXiv:2210.14215*.

LeCun, Y., Denker, J. S., and Solla, S. A. (1989). Optimal brain damage. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]*, pages 598–605. Morgan Kaufmann.

Lee, J. N., Xie, A., Pacchiano, A., Chandak, Y., Finn, C., Nachum, O., and Brunskill, E. (2023). Supervised pretraining can learn in-context reinforcement learning. *arXiv preprint arXiv:2306.14892*.

Lee, K.-H., Nachum, O., Yang, M., Lee, L., Freeman, D., Xu, W., Guadarrama, S., Fischer, I., Jang, E., Michalewski, H., et al. (2022). Multi-game decision transformers. *arXiv preprint arXiv:2205.15241*.

Levine, S., Kumar, A., Tucker, G., and Fu, J. (2020). Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*.

Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M., and Raffel, C. (2022). Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *arXiv preprint arXiv:2205.05638*.

Loshchilov, I. and Hutter, F. (2018). Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Mandlekar, A., Nasiriany, S., Wen, B., Akinola, I., Narang, Y., Fan, L., Zhu, Y., and Fox, D. (2023). Mimicgen: A data generation system for scalable robot learning using human demonstrations.

McInnes, L., Healy, J., and Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Mendez, J. A., Hussing, M., Gummadi, M., and Eaton, E. (2022). Composuite: A compositional reinforcement learning benchmark. In Chandar, S., Pascanu, R., and Precup, D., editors, *Conference on Lifelong Learning Agents, CoLLAs 2022, 22-24 August 2022, McGill University, Montréal, Québec, Canada*, volume 199 of *Proceedings of Machine Learning Research*, pages 982–1003. PMLR.

Meng, L., Wen, M., Yang, Y., Le, C., Li, X., Zhang, W., Wen, Y., Zhang, H., Wang, J., and Xu, B. (2021). Offline pre-trained multi-agent decision transformer: One big sequence model conquers all starcraftii tasks. *arXiv preprint arXiv:2112.02845*.

Merrill, W., Petty, J., and Sabharwal, A. (2024). The illusion of state in state-space models. *CoRR*, abs/2404.08819.

Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., et al. (2017). Mixed precision training. *arXiv preprint arXiv:1710.03740*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., , and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Nikulin, A., Kurenkov, V., Zisman, I., Agarkov, A., Sinii, V., and Kolesnikov, S. (2023). Xland-minigrid: Scalable meta-reinforcement learning environments in jax. *arXiv preprint arXiv:2312.12044*.

Nikulin, A., Zisman, I., Zemtsov, A., Sinii, V., Kurenkov, V., and Kolesnikov, S. (2024). Xland-100b: A large-scale multi-task dataset for in-context reinforcement learning. *arXiv preprint arXiv:2406.08973*.

Octo Model Team, Ghosh, D., Walke, H., Pertsch, K., Black, K., Mees, O., Dasari, S., Hejna, J., Kreiman, T., Xu, C., Luo, J., Tan, Y. L., Sanketi, P., Vuong, Q., Xiao, T., Sadigh, D., Finn, C., and Levine, S. (2024). Octo: An open-source generalist robot policy.

Orvieto, A., Smith, S. L., Gu, A., Fernando, A., Gülçehre, Ç., Pascanu, R., and De, S. (2023). Resurrecting recurrent neural networks for long sequences. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J., editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 26670–26698. PMLR.

Paischer, F., Hauzenberger, L., Schmied, T., Alkin, B., Deisenroth, M. P., and Hochreiter, S. (2024). One initialization to rule them all: Fine-tuning via explained variance adaptation. *arXiv preprint arXiv:2410.07170*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Patil, V., Hofmarcher, M., Dinu, M., Dorfer, M., Blies, P. M., Brandstetter, J., Arjona-Medina, J. A., and Hochreiter, S. (2022). Align-rudder: Learning from few demonstrations by reward redistribution. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvári, C., Niu, G., and Sabato, S., editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 17531–17572. PMLR.

Raad, M. A., Ahuja, A., Barros, C., Besse, F., Bolt, A., Bolton, A., Brownfield, B., Buttimore, G., Cant, M., Chakera, S., et al. (2024). Scaling instructable agents across many simulated worlds. *arXiv preprint arXiv:2404.10179*.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. (2021). Learning transferable visual models from natural language supervision. In Meila, M. and Zhang, T., editors, *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*, pages 8748–8763. PMLR.

Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., and Sutskever, I. (2022). Robust speech recognition via large-scale weak supervision. *arXiv preprint arXiv:2212.04356*.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training.

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

Raparthy, S. C., Hambro, E., Kirk, R., Henaff, M., and Raileanu, R. (2023). Generalization to new sequential decision making tasks with in-context learning.

Reed, S. E., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J. T., Eccles, T., Bruce, J., Razavi, A., Edwards, A., Heess, N., Chen, Y., Hadsell, R., Vinyals, O., Bordbar, M., and de Freitas, N. (2022). A generalist agent. *CoRR*, abs/2205.06175.

Salzmann, T., Kaufmann, E., Arrizabalaga, J., Pavone, M., Scaramuzza, D., and Ryll, M. (2023). Real-time neural mpc: Deep learning model predictive control for quadrotors and agile robotic platforms. *IEEE Robotics and Automation Letters*, 8(4):2397–2404.

Schmidhuber, J. (1992). Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Comput.*, 4(1):131–139.

Schmidhuber, J. (2019). Reinforcement learning upside down: Don't predict rewards–just map them to actions. *arXiv preprint arXiv:1912.02875*.

Schmidinger, N., Schneckenreiter, L., Seidl, P., Schimunek, J., Luukkonen, S., Hoedt, P.-J., Brandstetter, J., Mayr, A., Hochreiter, S., and Klambauer, G. (2024). Bio-xlstm: Generative modeling, representation and in-context learning of biological and chemical sequences. *Under reveiw*.

Schmidt, D. and Schmied, T. (2021). Fast and data-efficient training of rainbow: an experimental study on atari. *arXiv preprint arXiv:2111.10247*.

Schmied, T., Hofmarcher, M., Paischer, F., Pascanu, R., and Hochreiter, S. (2024a). Learning to modulate pre-trained models in rl. *Advances in Neural Information Processing Systems*, 36.

Schmied, T., Paischer, F., Patil, V., Hofmarcher, M., Pascanu, R., and Hochreiter, S. (2024b). Retrieval-augmented decision transformer: External memory for in-context rl. *arXiv preprint arXiv:2410.07071*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2018). Proximal policy optimization algorithms. *ArXiv*.

Schwarzer, M., Ceron, J. S. O., Courville, A., Bellemare, M. G., Agarwal, R., and Castro, P. S. (2023). Bigger, better, faster: Human-level atari with human-level efficiency. In *International Conference on Machine Learning*, pages 30365–30380. PMLR.

Schweighofer, K., Dinu, M.-c., Radler, A., Hofmarcher, M., Patil, V. P., Bitto-Nemling, A., Eghbal-zadeh, H., and Hochreiter, S. (2022). A dataset perspective on offline reinforcement learning. In *Conference on Lifelong Learning Agents*, pages 470–517. PMLR.

Shang, J., Kahatapitiya, K., Li, X., and Ryoo, M. S. (2022). Starformer: Transformer with state-action-reward representations for visual reinforcement learning. In *European Conference on Computer Vision*, pages 462–479. Springer.

Siebenborn, M., Belousov, B., Huang, J., and Peters, J. (2022). How crucial is transformer in decision transformer? *arXiv preprint arXiv:2211.14655*.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T. P., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Sinii, V., Nikulin, A., Kurenkov, V., Zisman, I., and Kolesnikov, S. (2023). In-context reinforcement learning for variable action spaces. *arXiv preprint arXiv:2312.13327*.

Smith, J. T. H., Warrington, A., and Linderman, S. W. (2023). Simplified state space layers for sequence modeling. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., de Las Casas, D., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T. P., and Riedmiller, M. A. (2018). Deepmind control suite. *CoRR*, abs/1801.00690.

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., and Metzler, D. (2020). Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*.

Todorov, E., Erez, T., and Tassa, Y. (2012a). MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033.

Todorov, E., Erez, T., and Tassa, Y. (2012b). Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Canton-Ferrer, C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. (2023). Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, l., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

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, Ç., 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. (2019). Grandmaster level in starcraft II using multi-agent reinforcement learning. *Nat.*, 575(7782):350–354.

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. (2023). Voyager: An open-ended embodied agent with large language models.

Wang, K., Zhao, H., Luo, X., Ren, K., Zhang, W., and Li, D. (2022). Bootstrapped transformer for offline reinforcement learning. *arXiv preprint arXiv:2206.08569*.

Wolczyk, M., Zajkac, M., Pascanu, R., Kuciński, L., and Miloś, P. (2021). Continual world: A robotic benchmark for continual reinforcement learning. *Advances in Neural Information Processing Systems*, 34:28496–28510.

Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., and Finn, C. (2020a). Gradient surgery for multi-task learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. (2020b). Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR.

Zheng, Q., Zhang, A., and Grover, A. (2022). Online decision transformer. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvári, C., Niu, G., and Sabato, S., editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 27042–27059. PMLR.

Zhu, G., Lin, Z., Yang, G., and Zhang, C. (2020). Episodic reinforcement learning with associative memory. In *International Conference on Learning Representations*.

Zhu, L., Liao, B., Zhang, Q., Wang, X., Liu, W., and Wang, X. (2024). Vision mamba: Efficient visual representation learning with bidirectional state space model. *CoRR*, abs/2401.09417.

Appendix

Contents

A Environments & Datasets

A.1 General

We compile a large-scale dataset comprising 432 tasks from six domains, 3.4M trajectories, and 894M transitions in total (see Table 4.1). To enable fast and targeted data-loading, every trajectory is stored in a separate `hdf5` file. We trade off some data-loading speed for disk space efficiency, by compressing trajectories that contain image-based observations.

A.2 Atari

The Arcade Learning Environment (ALE) [Bellemare et al., 2013] is the standard benchmark for evaluating RL agents and consists of 57 Atari games. Input observations in Atari are RGB images, but as is standard practice we gray-scale and crop frames ($|\mathcal{S}| = 1 \times 64 \times 64$). There are 18 discrete action across all 57 Atari games ($|\mathcal{A}| = 18$), but individual games may use only use a subset of these actions. Furthermore, we adopt the standard Atari recipe as used in prior works, including a frame skip of 4, maximum number of no-ops of 30, resetting on life loss, and reward clipping to $[-1, 1]$ [Mnih et al., 2015; Hessel et al., 2017].

Tasks. Similar to Lee et al. [2022], we assign 41 games to the training set, and 5 additional tasks to the hold-out set. The 41 training tasks include:

`amidar`, `assault`, `asterix`, `atlantis`, `bank-heist`, `battle-zone`, `beam-rider`, `boxing`, `breakout`, `carnival`, `centipede`, `chopper-command`, `crazy-climber`, `demon-attack`, `double-dunk`, `enduro`, `fishing-derby`, `freeway`, `frostbite`, `gopher`, `gravitar`, `hero`, `ice-hockey`, `jamesbond`, `kangaroo`, `krull`, `kung-fu-master`, `name-this-game`, `phoenix`, `pooyan`, `qbert`, `riverraid`, `road-runner`, `robotank`, `seaquest`, `time-pilot`, `up-n-down`, `video-pinball`, `wizard-of-wor`, `yars-revenge`, `zaxxon`

The 5 hold-out tasks include: `alien`, `pong`, `ms-pacman`, `space-invaders`, `star-gunner`

Table 2: Atari Dataset Statistics.

Task	**# of Trajectories**	**Mean Length**	**Mean Return**
`amidar`	1813	2753	145
`pooyan`	2773	1800	176
`frostbite`	5218	766	18
`video-pinball`	1023	3902	266
`wizard-of-wor`	3059	1314	15
`chopper-command`	5452	738	18
`breakout`	3780	1300	39
`phoenix`	3307	1509	49
`asterix`	5250	951	55
`enduro`	571	8720	636
`kung-fu-master`	1775	2812	131
`hero`	3022	1345	168
`assault`	3782	1170	77
`demon-attack`	1649	2431	116
`qbert`	3939	1138	155
`jamesbond`	2841	1758	11
`bank-heist`	4146	1204	62
`up-n-down`	3246	1538	99
`centipede`	6879	582	81
`boxing`	4796	1041	63
`battle-zone`	1933	2134	15
`name-this-game`	988	5049	389
`zaxxon`	2561	1950	12
`beam-rider`	1232	3248	77
`time-pilot`	3886	1029	11
`ice-hockey`	1465	3407	-6
`riverraid`	2645	1512	143
`krull`	3032	1319	528
`gopher`	1817	2338	185
`freeway`	2438	2048	33
`seaquest`	2807	1779	150
`double-dunk`	1774	2815	0
`road-runner`	3308	1217	135
`atlantis`	186	26349	1394
`gravitar`	6187	646	1
`yars-revenge`	4094	1036	96
`crazy-climber`	1105	3954	572
`kangaroo`	1787	2792	50
`fishing-derby`	2737	1825	0
`carnival`	21131	194	37
`robotank`	747	6652	56
Average	3321	2734	153

Dataset. For Atari, we leverage the DQN-Replay dataset released by Agarwal et al. [2020]. The dataset contains the trajectories seen over the entire training of the DQN agent (50M frames), We extract a subset of the last 5M transitions for every task, amounting to 205M transitions in total for the 41 training tasks. The number of episodes, the episodes lengths and total achieved rewards vary across tasks, as shown in Table 2.

A.3 Meta-World

The Meta-World benchmark [Yu et al., 2020a] consists of 50 manipulations tasks using a Sawyer robotic arm, ranging from opening or closing windows, to pressing buttons. Meta-World is based on the MuJoCo physics engine [Todorov et al., 2012a]. Observations in Meta-World are 39-dimensional continuous vectors ($|\mathcal{S}| = 1 \times 64 \times 39$), and actions are represented by 6 continuous dimensions ($|\mathcal{A}| = 18$) in range $[-1, 1]$. All tasks share a common action and state space. Following Wolczyk et al. [2021] and Schmied et al. [2024a], we limit the episode lengths to 200 interactions.

Tasks. We follow Yu et al. [2020a] and split the 50 Meta-World tasks into 45 training tasks (MT45) and 5 evaluation tasks (MT5).

The 45 training tasks are:

`reach`, `push`, `pick-place`, `door-open`, `drawer-open`, `drawer-close`, `button-press-topdown`, `peg-insert-side`, `window-open`, `window-close`, `door-close`, `reach-wall`, `pick-place-wall`, `push-wall`, `button-press`, `button-press-topdown-wall`, `button-press-wall`, `peg-unplug-side`, `disassemble`, `hammer`, `plate-slide`, `plate-slide-side`, `plate-slide-back`, `plate-slide-back-side`, `handle-press`, `handle-pull`, `handle-press-side`, `handle-pull-side`, `stick-push`, `stick-pull`, `basketball,soccer`, `faucet-open`, `faucet-close`, `coffee-push`, `coffee-pull`, `coffee-button`, `sweep`, `sweep-into`, `pick-out-of-hole`, `assembly`, `shelf-place`, `push-back`, `lever-pull`, `dial-turn`

The 5 evaluation tasks are: `bin-picking`, `box-close`, `door-lock`, `door-unlock`, `hand-insert`

Dataset. For Meta-World, we use the datasets released by [Schmied et al., 2024a], which contain 2M transitions per tasks and consequently 90M transitions in total for the training set. All episodes last for 200 environment interaction steps, and consequently there are 10K episodes for every task. For detailed dataset statistics per task, we refer to their publication.

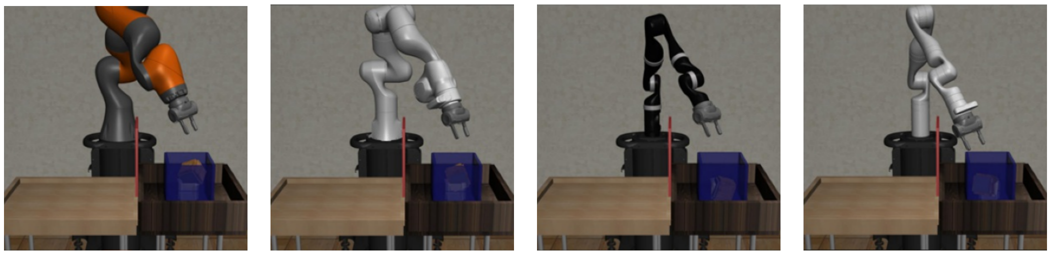

(a) IIWA (b) Panda (c) Jaco (d) Gen3

Figure 9: Illustration of the four supported robot arms in **Composuite** [Mendez et al., 2022].

A.4 DMControl

The DMControl benchmark [Tassa et al., 2018] consists of 30 different robotic tasks. Unlike Meta-World, the benchmark contains robots with different morphologies instead of a single common Sawyer arm. Due to the different robot morphologies, the state, and action spaces vary across tasks ($3 \leq |\mathcal{S}| \leq 24$, $1 \leq |\mathcal{A}| \leq 6$), with all actions in range $[-1, 1]$.

Tasks. We do not use all 30 tasks contained in the DMControl benchmark, but select 16 of the 30 tasks that have been used in prior works [Hafner et al., 2019; Schmied et al., 2024a,b], which we refer to as DMC11 and DMC5 respectively.

The 11 training tasks are:

`finger-turn_easy`, `fish-upright`, `hopper-stand`, `point_mass-easy`, `walker-stand`, `walker-run`, `ball_in_cup-catch`, `cartpole-swingup`, `cheetah-run`, `finger-spin`, `reacher-easy`

The 5 evaluation tasks are:

`cartpole-balance`, `finger-turn_hard`, `pendulum-swingup`, `reacher-hard`, `walker-walk`

Dataset. For DMControl, we generate 10M transitions per task by training task-specific SAC [Haarnoja et al., 2018] agents, using the same setup as Schmied et al. [2024a]. Episodes in all DMControl tasks last for 1000 environment steps and per time-step a maximum reward of +1 can be achieved, which results in a maximum reward of 1000 per episode. Consequently, our training set contains 10K episodes per tasks, amounting to 110K episodes and 110M transitions in total across all tasks. We list the dataset statistics for all 11 tasks in Table 3.

A.5 Composuite

The Composuite benchmark [Mendez et al., 2022], is a robotics benchmark for grasping and object manipulation. The benchmark is implemented on top of `robotsuite` [Zhu et al., 2020], which in turn leverages the MuJoCo

Table 3: DMControl Data statistics.

Task	# of Trajectories	Mean Length	Mean Return
`point_mass_easy`	10K	1K	851
`cheetah_run`	10K	1K	385
`walker_run`	10K	1K	230
`ball_in_cup_catch`	10K	1K	969
`hopper_stand`	10K	1K	460
`walker_stand`	10K	1K	939
`finger_turn_easy`	10K	1K	954
`reacher_easy`	10K	1K	938
`cartpole_swingup`	10K	1K	817
`fish_upright`	10K	1K	815
`finger_spin`	10K	1K	966
Average	19628	152	8.2

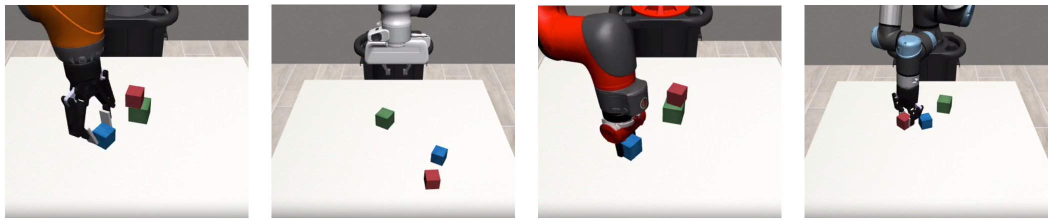

(a) IIWA (b) Panda (c) Sawyer (d) UR5e

Figure 10: Illustration of the four supported robot arms in **Mimicgen** [Mandlekar et al., 2023] solving the `stack-three` task.

simulator under the hood [Todorov et al., 2012b]. Composuite contains a mix of 4 simulated robot arms: `IIWA`, `Jaco`, `Gen3`, and `Panda` (see Figure 9). All arms share a common state and action space containing 93 continuous state dimensions and 8 continuous action dimensions, respectively ($|\mathcal{S}| = 93$, $|\mathcal{A}| = 8$).

Tasks. CompoSuite is designed as a compositional multi-task benchmark for RL, in which a particular *robot* manipulates a particular *object* given an *objective*, while avoiding *obstacles*. Overall, there are 4 robots arms, 4 objects, 4 obstacles, and 4 task objectives. This results in 256 possible robot/object/objective/obstacles combinations. For our experiments, we assign 240 tasks to the training set and use the remaining 16 tasks as hold-out set (`Panda` and `Object_Wall`) combinations. For a list of all 256 tasks, we refer to Mendez et al. [2022].

Dataset. For Composuite, we leverage the datasets released by Hussing et al. [2023]. For every task, we select 2000 episodes, which last on average for 500 steps. This amounts to 1M transitions per task, and 240M transitions across all 240 training tasks. For dataset statistics, we refer to Hussing et al. [2023].

A.6 Mimicgen

Similar to Composuite, Mimicgen [Mandlekar et al., 2023] is based on `robosuite` and the MuJoCo simulator. Mimicgen is designed for automatically synthesizing large-scale datasets from only a handful of human demonstrations. Observations in Mimicgen can be represented as images (from multiple cameras) or low dimensional continuous states. For our experiments, we opt for the low-dimensional state representation to simplify learning. Therefore, observations and actions are represented by 37-dimensional and 7-dimensional continuous vectors, respectively ($|\mathcal{S}| = 37$, $|\mathcal{A}| = 7$). Similar to Composuite, Mimicgen supports 4 different robot arms: `Panda`, `IIWA`, `Sawyer`, and `UR5e` (see Figure 10).

Tasks. Mimicgen consists of 24 diverse tasks, including stacking blocks, re-assembling objects, and even long-horizon tasks like coffee preparation. These 24 tasks can be performed with the four supported robot arms, amounting to 96 tasks in total.

Dataset. Mandlekar et al. [2023] released dataset for the 24 tasks using the default robot arm `Panda`. To increase the dataset diversity, we additionally generated data for the remaining 3 robot arms. However, not all data generation runs produce successful trajectories, and we discard with too few successful trajectories. Our final dataset for Mimicgen contains 83 training and 2 evaluation tasks. For each task, we collect 1000 successful demonstrations (we do not include unsuccessful trajectories). Episode lengths vary across tasks, ranging from 260 to 850 environment steps.

A.7 Procgen

Procgen benchmark consists of 16 procedurally-generated video games [Cobbe et al., 2020b]. Observations in Procgen are RGB images of dimension $3 \times 64 \times 64$. However, for training efficiency, we apply gray-scaling to image observations ($|\mathcal{S}| = 1 \times 64 \times 64$). All 16 environments share a common action space of 15 discrete actions ($|\mathcal{A}| = 16$). Procgen is designed to test the generalization abilities of RL agents. Consequently, procedural generation is employed to randomize background and colors, while retaining the game dynamics.

Tasks. Following prior works [Raparthy et al., 2023; Schmied et al., 2024b], we assign 12 and 4 tasks to training and hold-out set, respectively. The 12 training tasks are:

`bigfish`, `bossfight`, `caveflyer`, `chaser`, `coinrun`, `dodgeball`,
`fruitbot`, `heist`, `leaper`, `maze`, `miner`, `starpilot`

The 4 hold-out tasks are: `climber`, `ninja`, `plunder`, `jumper`

Dataset. We leverage the datasets released by Schmied et al. [2024b], which contain 20M transitions per task. The datasets were generated by recording all transitions observed by training RL agents for 25M steps, followed by uniform subsampling to 20M transitions. Consequently, the dataset contains mixed quality trajectories ranging from random (beginning of training) to expert (end of training). We list the dataset statistics for all 16 tasks in Table 4.

Table 4: Procgen Data statistics.

Task	# of Trajectories	Mean Length	Mean Return
`bigfish`	82835	230	6.251
`bossfight`	112459	141	1.946
`caveflyer`	151694	105	7.745
`chaser`	93612	212	3.248
`coinrun`	261117	51	9.473
`dodgeball`	144364	137	2.884
`fruitbot`	73653	270	16.094
`heist`	101361	196	8.405
`leaper`	296084	67	4.446
`maze`	482245	41	9.432
`miner`	288818	68	11.8
`starpilot`	96468	206	17.3
Average	182059	144	8.3

B Experimental & Implementation Details

B.1 Training & Evaluation.

In our experiments, we compare two variants of xLSTM, Mamba and DT. For our main experiments in Section 4.2, we train all models for 200K updates, and evaluate after every 50K update steps. We report the mean and 95% confidence intervals over three seeds in our experiments, as suggested by Agarwal et al. [2021]. For every evaluation tasks, we take the average of 3 evaluation seeds.

We train our agents with a batch size of 128 and gradient accumulation across the 6 domains, such that every domain is represented with the same proportion. Consequently, the effective batch size is 768. We use a learning rate of $1e^{-4}$, 4000 linear warm-up steps followed by a cosine decay to $1e^{-6}$, and train using the AdamW optimizer [Loshchilov and Hutter, 2018]. In addition, we employ gradient clipping of 0.25, weight decay of 0.01 for all models. We do not employ Dropout, as is standard practice in DTs, as we found that it negatively affects performance (see Section 4.3). We use separate reward scales of 200, 100 and 20 for Meta-World, DMControl and Atari, respectively. Furthermore, for all domains, we set the target return to the maximum return achieved for a particular task in the training datasets. This is particularly useful for domains, where the maximum returns differ heavily across tasks (e.g., Atari). We list all hyperparameters in Table 5.

Table 5: Hyperparameters for RA-DT.

Parameter	**Value**
Gradient steps	200K
Evaluation frequency	50K
Evaluation episodes	5
Optimizer	AdamW
Batch size	128
Gradient accumulation	6
Lr schedule	Linear warm-up + Cosine
Warm-up steps	4000
Learning rate	1e-4 $\rightarrow$ 1e-6
Weight decay	0.01
Gradient clipping	0.25
Dropout	0.2
Context len (timesteps)	50
Reward scale	per-domain
Target return	per-task

B.2 Context Lengths.

By default, we train all models with a context length $C = 50$ timesteps. For every timestep there are three tokens (s/rt/r) and consequently, the effective context length is 150. We found that performance improves for longer context lengths (see Section D.1), but limit our experiments to $C = 50$ to reduce the computational cost.

B.3 Model Architectures.

We train models across 4 models sizes: 16M, 48M, 110M, and 206M. We follow Lee et al. [2022] in selecting the number of layers and hidden dimensions. For xLSTM and Mamba, we use twice the number of layers blocks to match the number of parameters of the Transformer [Beck et al., 2024; Gu et al., 2024] (see Table 6) For our xLSTM [7:1] variant, which contains sLSTM blocks, we strive to maintain the same ratio as proposed by Beck et al. [2024]. Not all our model sizes are divisible by 8 and only the 16M and 110M models exhibit the exact 7:1 ratio of mLSTM to sLSTM blocks. For consistency, however, we maintain the same notation as Beck et al. [2024]. We place sLSTM blocks at positions [1], [1, 3], [1, 3], and [1, 3, 5] for the 16M, 48M, 110M, 206M, respectively.

Across backbones, we use linear layers to encode continuous states, reward returns-to-go, similar to Chen et al. [2021]. The maximal state-dimension across continuous control environments is 204 in our experiments. To use a shared linear embedding layer for continuous states, we pad states that have lower number of dimensions to 204 dimensions using zeros. To encode image inputs on visual domains, we use the IMPALA-CNN proposed by Espeholt et al. [2018] and adopted by previous works on Procgen [Cobbe et al., 2020b] and Atari [Schmidt and Schmied, 2021; Schwarzer et al., 2023]. Consequently, we do not make use of discretization of continuous states or patchification of images. This design choice significantly reduces the sequence length to only three tokens per time-step (see Appendix B.2) and consequently results in faster inference.

For continuous actions, we make use of discretization and discretize of every action dimension into 256 uniformly-spaced bins, similar to Reed et al. [2022] and Brohan et al. [2023b]. We experimented with lower/higher number of bins, but did not observe a benefit beyond 256 bins. Consequently, this resolution is sufficient for the environments we consider. We use a shared action head to predict the action bins of all continuous dimensions jointly. The maximum number of continuous action dimensions is 8 in our experiments and consequently the number of discrete action classes is 2048. In addition, there are 18 discrete actions originating from Atari and Procgen. Therefore, our action head learns to predict the correct action among the 2066 discrete classes. We opt for the shared action head representation, as this further speeds up inference and does not require autoregressive action prediction.

For the Transformer baseline, we use global positional embeddings similar to Chen et al. [2021]. For the recurrent backbones, we do not make use of positional encodings.

B.4 Hardware & Training Times.

We train all our models on a server equipped with 4 A100 GPUs. We use distributed data parallel to distribute the workload, as supported in PyTorch [Paszke et al., 2019]. Training times range from 5 hours for the smallest

Table 6: Model Sizes.

Model	Layers	Hidden Dim	Heads	Parameters
Transformer	4	512	8	16M
Transformer	6	768	12	48M
Transformer	8	1024	16	110M
Transformer	10	1280	20	206M
Mamba	8	512	-	16M
Mamba	12	768	-	48M
Mamba	16	1024	-	110M
Mamba	20	1280	-	206M
xLSTM	8	512	4	16M
xLSTM	12	768	4	48M
xLSTM	16	1024	4	110M
xLSTM	20	1280	4	206M

DT model to 30 hours for the largest Mamba model. Throughout all our experiments, we use mixed precision training [Micikevicius et al., 2017] as supported in PyTorch to speed up training time.

We evaluate our models after every 50K steps. However, periodically evaluating the trained agents on all 432 tasks sequentially is time-consuming. Therefore, we perform parallel evaluation with 4 processes at a time. For multi-GPU setups, we distribute the evaluation workload among the available GPUs. For example, with 4 available GPUs and 4 evaluation processes per GPU, 16 environments are evaluated simultaneously. Consequently, the total evaluation time for all 432 tasks, ranges from 18 minutes for the smallest DT model to roughly 2 hours for the largest Mamba model.

C Additional Results

C.1 Training Tasks

In Figures 11 and 12, we report the normalized scores obtained per domain and the average learning curves across tasks for all four model sizes.

In Figure 13, we report the training perplexity on the 432 training tasks over 200K updates. Here, we observe that the training perplexity behaves similar to the validation perplexity. This is expected, as our models see most transitions only a single time (see Table 4.1 for the number of repetitions per domain).

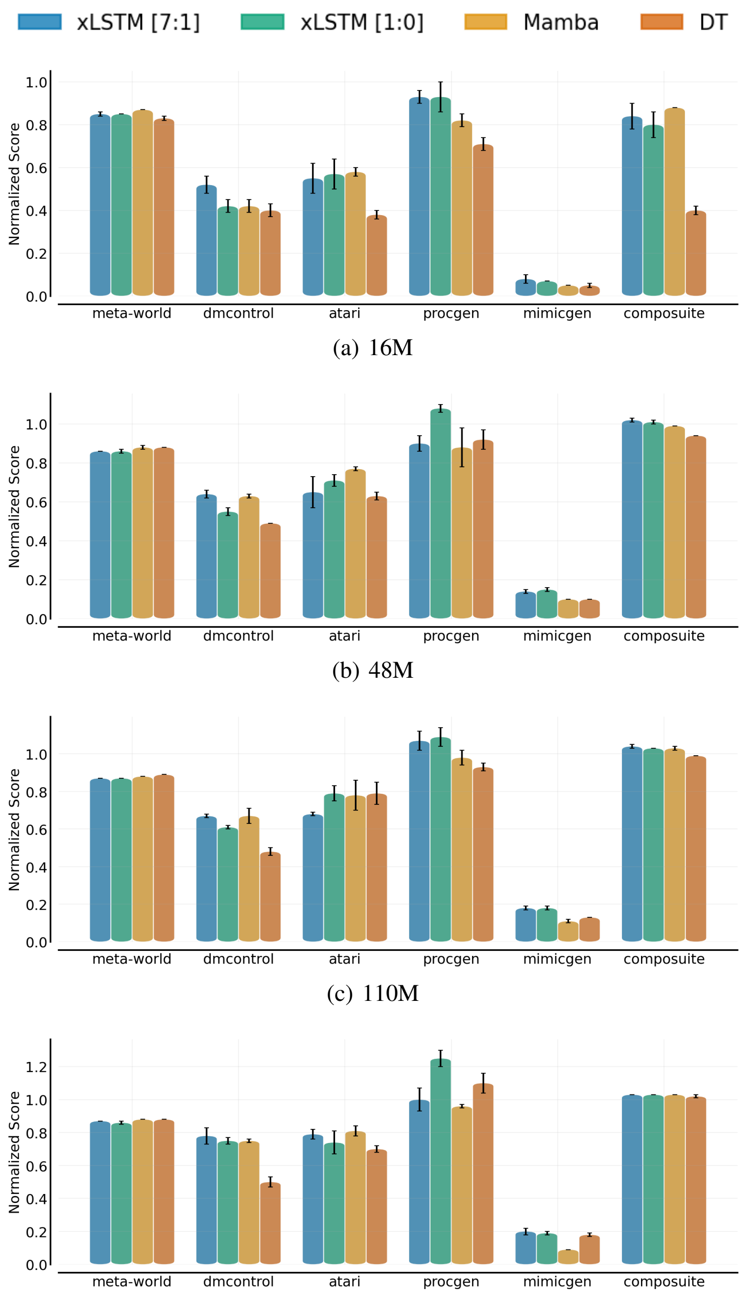
Furthermore, we report the scaling curves with an additional model size of 408M parameters in Figure 14. Due to the high computational cost of the 408M models, we were currently only able to conduct a single run for this size. However, we aim to provide further empirical evidence for this model sizes in future work.

C.2 Hold-out Tasks

In Figure 15, we show the zero-shot evaluation performance on the hold-out tasks 15. We want to highlight, that the performance declines for all methods and model sizes compared to performance on training tasks. This is because, hold-out tasks exhibit severe shifts in state-spaces, action-spaces and reward functions.

C.3 Fine-Tuning

In Figure 16, we present the fine-tuning evaluation performance on the held-out tasks. We compare an xLSTM trained from scratched against an xLSTM initialized with the pre-trained weights. We use full fine-tuning for our experiments, but highlight that parameter-efficient approaches are attractive alternatives [Hu et al., 2022; Liu et al., 2022; Paischer et al., 2024]. We do observe consistent improvement of the pre-trained models over the models trained from scratch. This indicates that fine-tuning performance is not negatively impacted by the recurrent backbone. While we train on a substantial number of environments, the total amount of data used is, however, still only a fraction of that employed in training other large-scale models, such as LLMs. Consequently, we do not observe comparable few-shot generalization. We anticipate that few-shot generalization capabilities will emerge as we increase both data volume and model size.



(a) 16M

(b) 48M

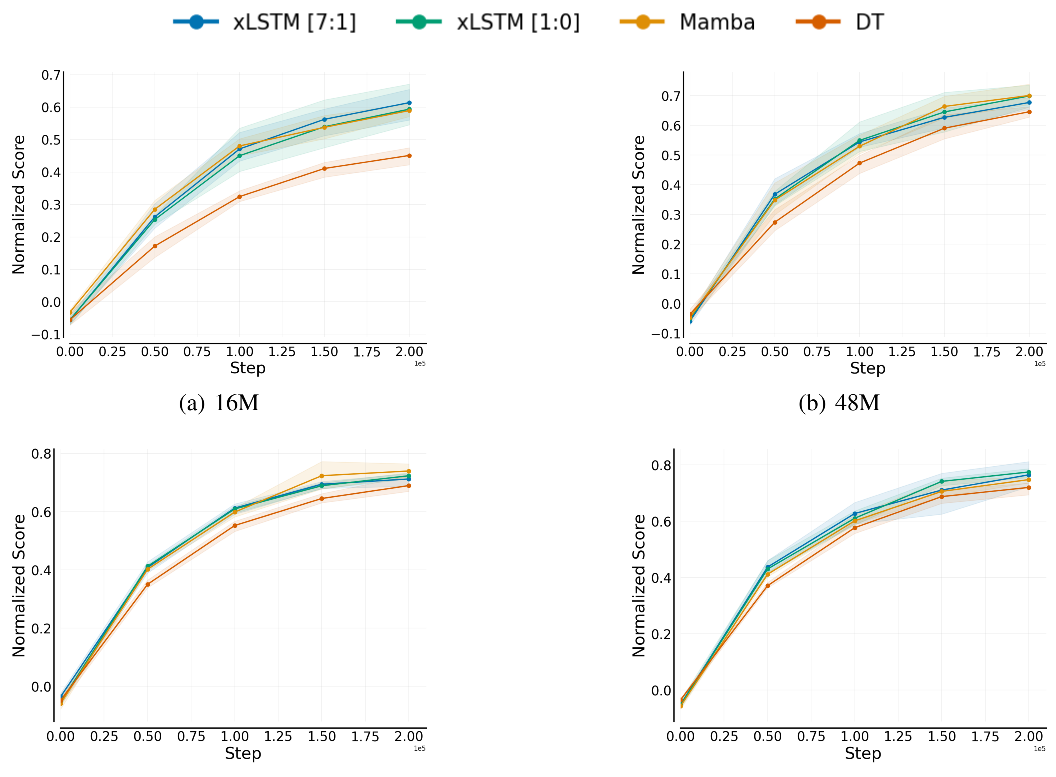
(c) 110M

(d) 206M

Figure 11: Normalized scores per-domain all four model sizes: 16M, 48M, 110M, and 206M. For Meta-World, DMControl, Mimicgen, Composuite, and Procgen we report data-normalized scores, for Atari we report human-normalized scores.

C.4 In-context Learning

We assess the ICL abilities of modern recurrent architectures on the Dark-Room environment considered in prior works on in-context RL [Laskin et al., 2022; Lee et al., 2023; Kirsch et al., 2023; Sinii et al., 2023; Huang et al., 2024; Schmied et al., 2024b]. In Dark-Room, the agent is located in a dark room. The task is to navigate to an invisible goal location in that dark room. The state is partially observable, as the agent only observes its own x-y position on the grid ($|\mathcal{S}| = 2$). The action space consists of 5 discrete actions: move up, move down, move left, move right, stay ($|\mathcal{A}| = 5$). Upon reaching the goal location, the agent receives a reward of +1 for every step



(a) 16M

(b) 48M

(c) 110M

(d) 206M

Figure 12: Learning curves for all four model sizes, 16M, 48M, 110M, and 206M, on the training tasks.

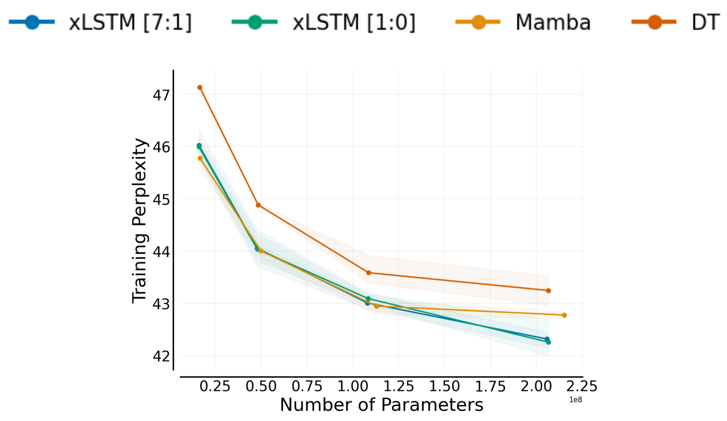
in the episode it resides on the goal location. Consequently, the agent first has to explore the room to find the goal. Once the goal location is found (as indicated by the positive reward), the agent can exploit this knowledge. Given a multi-episodic context, the agent should be able to exploit information contains in the previous trials (e.g., exploiting one path vs. avoiding another).

In our experiments, the Dark-Room is a 10×10 grid and episodes last for 100 steps, starting in the top left corner of the grid. We adopt the same experiment setup as Schmied et al. [2024b] and leverage their datasets. We train 16M parameter agents on datasets from 80 randomly selected goal locations in the grid. The datasets contain 100K transitions per task and are obtained by training task-specific PPO [Schulman et al., 2018] agents. Then, we evaluate the in-context abilities of our agents on 20 hold-out goal locations. During evaluation, the agent is given 40 episodes to interact with the environment, which we refer to as ICL-trials. Furthermore, we adopt the AD [Laskin et al., 2022] framework for training our agents with a multi-episodic context. We use the same sequence representation as used in our main experiments, consisting of states, returns-to-go (target return set to 80 during evaluation), and rewards. Note that this differs from the sequence representation used by Laskin et al. [2022]. We set the context length for all agents to the equivalent of two episodes, which amounts to 200 timesteps in total.

In Figure 17, we report the ICL performance over the 40 ICL trials on (a) 80 training locations and (b) 20 hold-out locations for the 4 different backbones considered in this work. We observe that the recurrent backbones attain considerably higher scores than the Transformer backbone. Furthermore, we find that xLSTM [7:1] attains the highest overall scores, which we attribute to the state-tracking abilities [Merrill et al., 2024] of sLSTM blocks. We aim to explore the ICL abilities of modern recurrent backbones on more complex benchmarks such as XLand-minigrid [Nikulin et al., 2023, 2024] in future work.

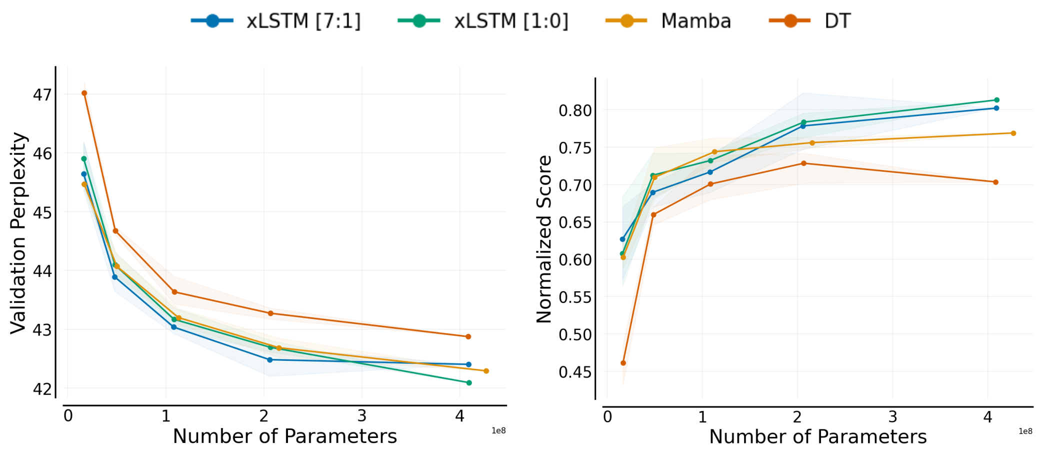
C.5 Inference Time Comparisons

We empirically examine the difference in inference speed between of our models. Similar to De et al. [2024], we report both latency and throughput. For real-time applications, latency is the more important dimension, and therefore we focus our analysis on latency.



(a) Training Perplexity

Figure 13: Scaling comparison. We compare xLSTM, Mamba, DT in four model sizes: 16M, 48M, 110M, and 206M parameters. We show the **training perplexity** on the training dataset to evaluate the sequence prediction performance.



(a) Sequence prediction (b) Environment interaction

Figure 14: Scaling comparison with additional 408M parameter models. We show the **(a)** validation perplexity on the hold-out datasets, and **(b)** normalized scores obtained from evaluating in the training task environments, averaged over all 6 domains.

C.5.1 Latency

In Figures 18 and 19, we report the latencies for DT and xLSTM with the same number of layer blocks as DT, and twice the number of layers blocks as DT, respectively. We conduct our comparison for two different batch sizes and across varying sequence lengths.

C.5.2 Throughput

In Figures 20 and 21, we similarly report the attained throughput for DT and xLSTM with the same number of layer blocks as DT, and twice the number of layers blocks as DT, respectively. We conduct our comparison for two fixed context lengths and varying batch sizes.

C.5.3 xLSTM Kernel Comparisons

We leverage custom kernels for xLSTM to conduct our inference-speed comparisons. In particular, we compare 4 variants: recurrent-style inference with and without kernel acceleration, and chunkwise inference with and without kernel acceleration. In our experiments, every timestep contains 3 individual tokens. Consequently,

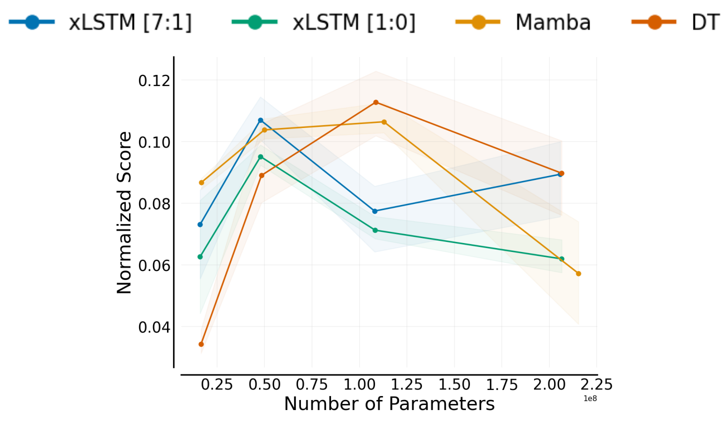


Figure 15: Scaling comparison. **Zero-shot performance on hold-out tasks** at four models sizes, 16M, 48M, 110M, and 206M. Note that performance declines for all methods and model sizes compared to performance on training tasks. This is because, hold-out tasks exhibit severe shifts in state-spaces, action-spaces and reward functions.

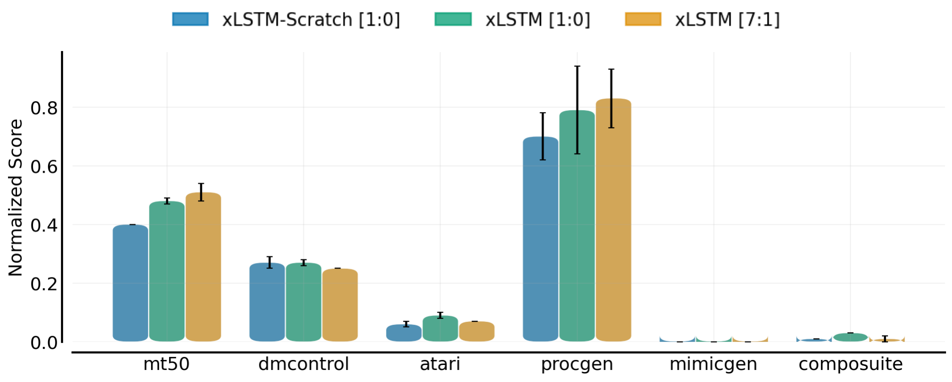
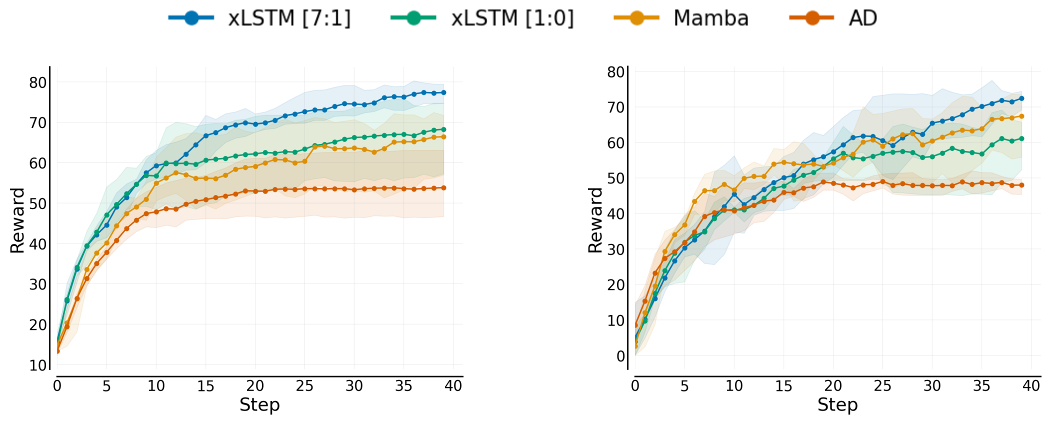


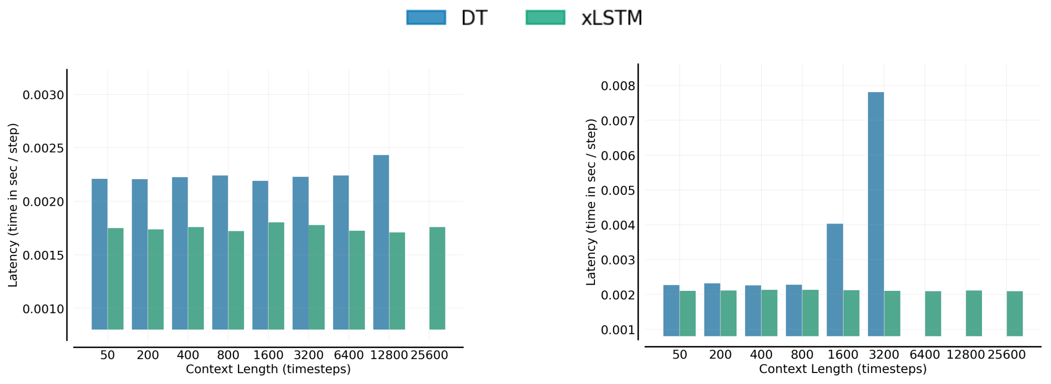
Figure 16: Fine-tune performance on hold-out tasks. We compare the performance of a pretrained xLSTM against an xLSTM trained from scratch, both with 16 million parameters. We select the top 5% percent of trajectories from our held-out tasks based on performance and used this subset to fine-tune the models. We perform 25K update steps during fine-tuning and show the normalized scores, averaged across held-out tasks from each domain.

regular recurrent-style inference requires iterating over the token sequence of length 3 in a loop given the hidden state of the previous timestep. This requires 3 forward passes. In contrast, the chunkwise implementation operates on chunks of timesteps given a hidden state. Consequently, this only requires a single forward pass. In Figure 22, we illustrate the impact of kernel acceleration. We find that our chunkwise kernels result in considerably lower latencies. Interestingly, we find that for $B = 1$, our chunkwise implementation without kernel acceleration is faster than the recurrent-style inference with kernel acceleration. However, as the batch size increases, this trend reverses. This highlights the importance of kernel acceleration for efficient inference.



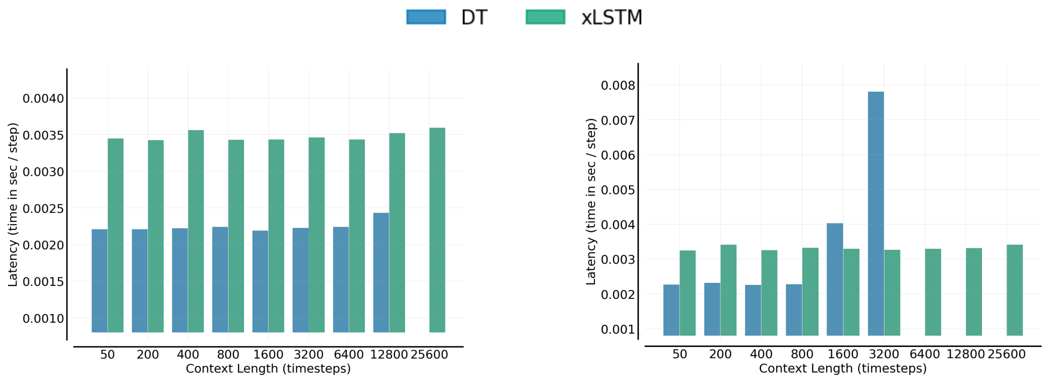
(a) 80 training tasks (b) 20 hold-out tasks

Figure 17: In-context Learning on Dark-Room 10×10.



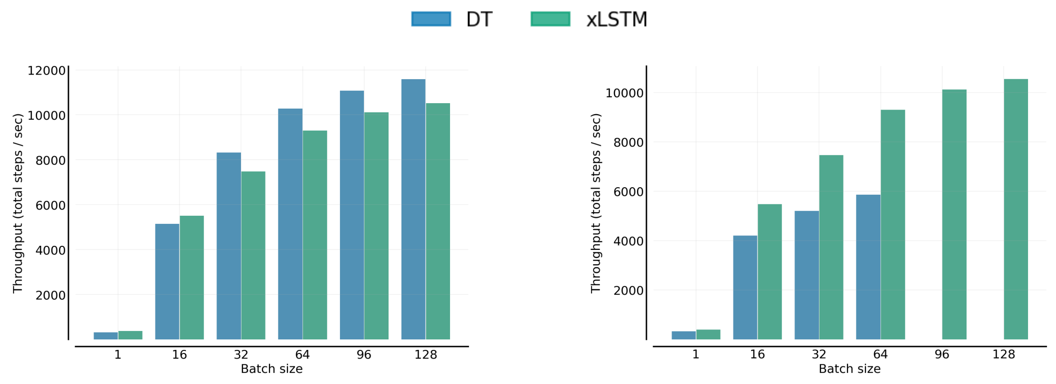
(a) batch_size = 1 (b) batch_size = 16

Figure 18: Latency. We report latency with (a) batch size of 1 and (b) batch size of 16 for DT and xLSTM with 206M parameters. For xLSTM we use the same number of layer blocks as DT and a higher hidden dimension to match parameters.



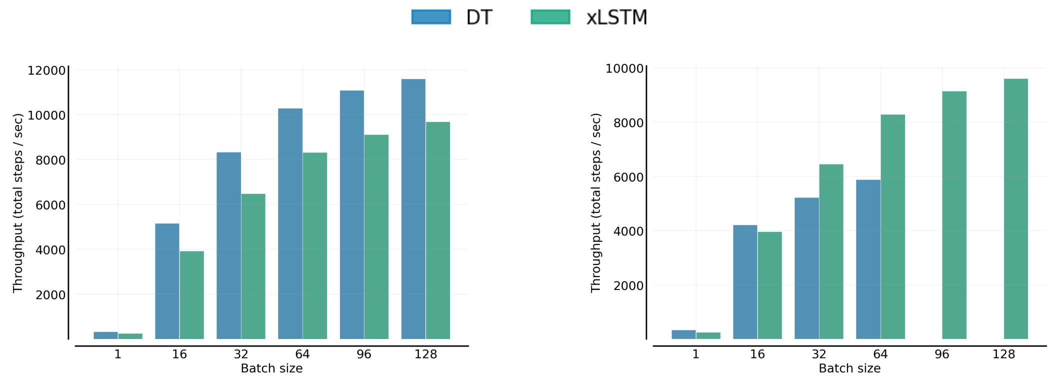
(a) batch_size = 1 (b) batch_size = 16

Figure 19: Latency. We report latency with (a) batch size of 1 and (b) batch size of 16 for DT and xLSTM with 206M parameters. For xLSTM, we use twice the number of layer blocks and the same hidden dimension as the Transformer.



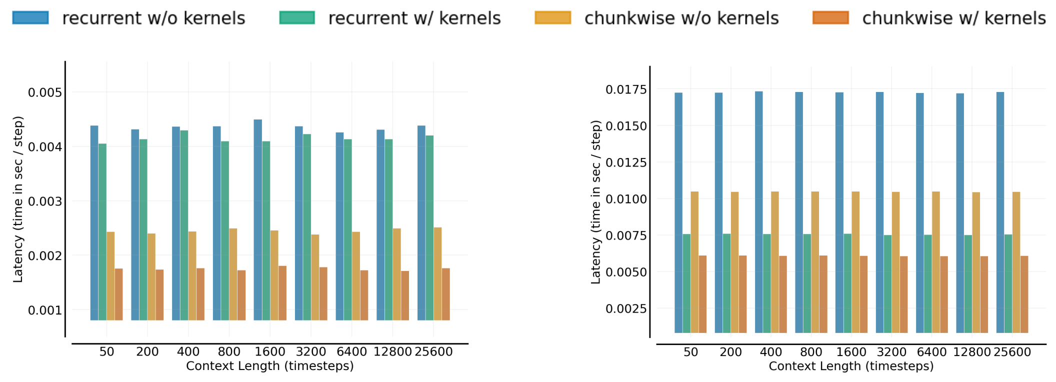
(a) context_length = 800 (b) context_length = 1600

Figure 20: Throughput. We report throughput with (a) context size of 800, and (b) context size of 1600 timesteps for DT and xLSTM with 206M parameters. For xLSTM we use the same number of layer blocks as DT and a higher hidden dimension to match parameters.



(a) context_length = 800 (b) context_length = 1600

Figure 21: Throughput. We report throughput with (a) context size of 800, and (b) context size of 1600 timesteps for DT and xLSTM with 206M parameters. For xLSTM, we use twice the number of layer blocks and the same hidden dimension as the Transformer.



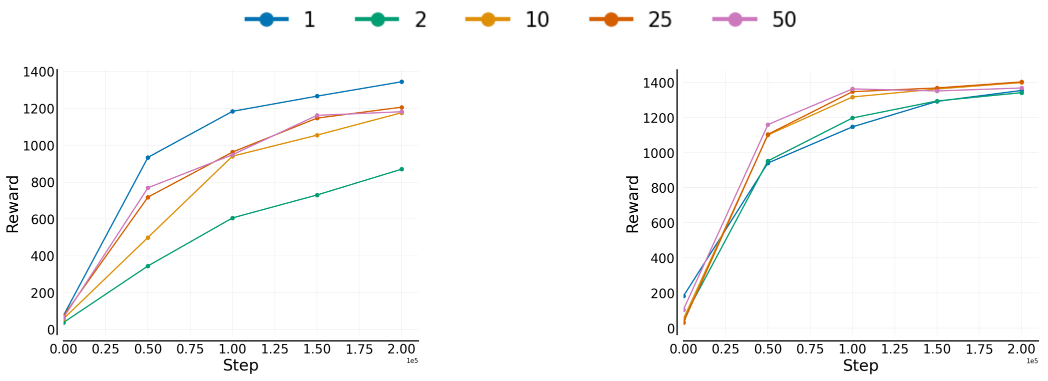
(a) batch_size = 1 (b) batch_size = 16

Figure 22: Impact of kernel acceleration. We report latency with (a) batch size of 1 and (b) batch size of 32 for DT and xLSTM with 206M parameters. For xLSTM we use the same number of layer blocks as DT and a higher hidden dimension to match parameters.

D Ablations

D.1 Removing action condition

We found that removing actions from the context results in better performance across backbones. In Figure 23, we report the learning curves over 200K updates for DT with varying context lengths, both with and without actions in the context. While context lengths beyond 1 hurt performance on meta-world when training with actions, the reverse is true when training without actions. This is in contrast to recent works, which did not benefit from longer contexts [Octo Model Team et al., 2024]. However, while removing actions improves performance on the robotics domains, it does not affect performance on discrete control. For robotics, we observed that the models become overly confident (high action logits), which is problematic if poor initial actions are produced. We assume this is because in robotics actions change smoothly and by observing previous actions the agent learns shortcuts. Thus, removing actions from the input prevents the agent from using shortcuts.



(a) w/ actions (b) w/o actions

Figure 23: Ablation on removing the **action condition** for varying context lengths C. Performance of DT **(a)** with, and **(b)** without action condition on Meta-World. With action in the context, $C > 1$ harms performance due to overconfidence in action predictions. Without actions in the context, the performance of DT improves with increasing C.

D.2 Effect of Dropout in DT

DTs use by default a Dropout [Srivastava et al., 2014] rate of 0.1. However, during our experiments, we found that Dropout has detrimental effects on the evaluation performance, particularly on continuous control domains like Composuite. In Figure 24, we show the validation perplexities and evaluation performance for a DT trained with and without Dropout. Consequently, we remove Dropout from our DT variant.

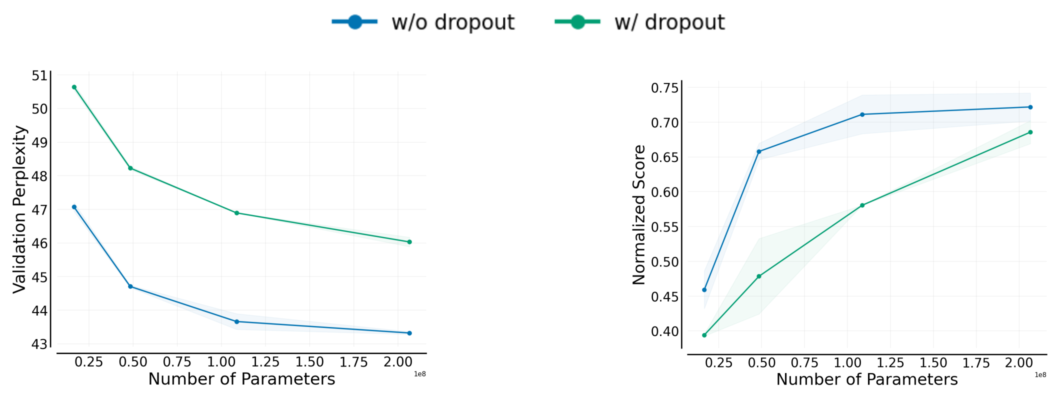
D.3 Effect of reducing number of layers in xLSTM

In prior works, xLSTM and Mamba use twice the number of layers blocks as the Transformer baseline, while maintaining the same hidden dimension [Gu and Dao, 2023; Beck et al., 2024]. For our inference-time comparisons, we therefore reduce the number of layer blocks in xLSTM by half. To ensure a fair comparison, we consequently adjust the hidden size of xLSTM to match the number of parameters of the Transformer baseline. In this section, we investigate the effect of these modifications of the xLSTM architecture on the model performance.

In Figure 25, report the validation perplexities and evaluation performance for the *regular* xLSTM with twice the number of layer blocks as DT, and an xLSTM with *half* the number of blocks. Reducing the number of layer blocks results in slight decrease in performance on both metrics. However, xLSTM still outperforms the Transformer baseline (see Figure 2).

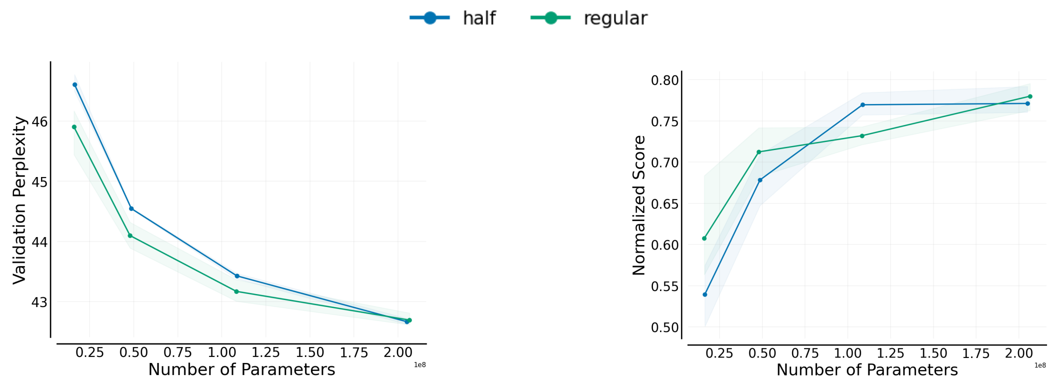
E Embedding Space Analysis

In Figure 5, we analyze the representations learned by our models using UMAP [McInnes et al., 2018]. Here, we explain the clustering procedure in more detail. For every task, we sample 32 sub-trajectories containing 50 timesteps (150 tokens) and encode them using our sequence models. Then, we extract the hidden states at the last layer of our model and aggregate them via mean pooling. We cluster all vectors using default hyperparameters



(a) Sequence Prediction Performance (b) Evaluation Performance

Figure 24: Ablation on the **effect of dropout on DT** performance. We show the **(a)** validation perplexity and **(b)** evaluation performance on the training tasks. DT performance drops considerably if training with dropout.



(a) Sequence Prediction Performance (b) Evaluation Performance

Figure 25: Ablation on the effect of reducing the number of layer blocks in xLSTM. We show the **(a)** validation perplexity and **(b)** evaluation performance on the training tasks for the layer regular and layer-matched matched xLSTM models. Reducing the number of layer blocks in xLSTM results in a slight performance decrease.

of UMAP into a two-dimensional space. Finally, we color the resulting points by their domain. Generally, we find that tasks from the same domain cluster together.