# A NEW PERSPECTIVE ON TRANSFORMERS IN ONLINE RL FOR CONTINUOUS CONTROL

**Nikita Kachaev**[1,3*] **Daniil Zelezetsky**[2*] **Alexey K. Kovalev**[1,2] **Aleksandr I. Panov**[1,2]
[1]AIRI, Moscow, Russia [2]MIPT, Dolgoprudny, Russia [3]HSE University, Moscow, Russia
zelezetskii.dv@phystech.edu, {kachaev,kovalev,panov}@airi.net

## ABSTRACT

Developing transformer-based models in online reinforcement learning (RL) faces a wide range of difficulties such as training instability or suboptimal behavior. In this paper, we find out whether the transformer architecture can be considered as a backbone for RL algorithms. We show that transformers can be trained by classical online RL algorithms without requiring global changes in the training process. Moreover, we explore different transformer architectures and ways to train them. As a result we form a set of recommendations and practical takeaways about how to develop stable approaches of transformer training. We hope that our work will help in understanding the intricacies of configuring transformers for reinforcement learning and will allow to formulate the basic principles of forming a training pipeline for transformer-based architectures.

## 1 INTRODUCTION

The introduction of the transformer model (Vaswani et al., 2017) turned out to be a driving force in a wide range of fields connected with deep learning solutions. The reinforcement learning (RL) domain has also been influenced by transformers, especially in Partially Observable Markov Decision Process (POMDP) environments and memory-demanding tasks (Ni et al., 2023; Pleines et al., 2023; Lampinen et al., 2021; Goyal et al., 2022). Another key advantage of transformers lies in their ability to process multimodal input (Xu et al., 2023). The efficient processing of multimodal data helps to create Vision models (Radford et al., 2021), Vision-Language models (Lu et al., 2024) , Vision-Language-Action models (Kim et al., 2024), multitask agents (Grigsby et al., 2024), or even generalist agents (Team et al., 2024; Jiang et al., 2022; Reed et al., 2022).

All these benefits in the RL domain can be achieved mostly by using offline datasets (Chen et al., 2021; Janner et al., 2021; Zheng et al., 2022) and an offline pretraining stage (Nair et al., 2020; Schwarzer et al., 2021; Sun et al., 2023). On the one hand, the supervised learning approach is well-studied, so we have a set of measures that stabilize transformers and provide a performance gain during training. On the other hand, classical RL works under the assumptions of a fully online training setting, and there may be many obstacles to obtaining offline data, such as the high costs of collecting it or the absence of trained policy which can do it. Various efforts (Zheng et al., 2022; Sun et al., 2023; Elawady et al., 2024) have been made to improve transformer performance, but they still require offline data, which contradicts the fully online setting.

The main question we want to answer in this paper is: **"What limits the potential of transformers in online RL, and how to overcome these limitations?"**. To answer this question we created a list of recommendations and practical takeaways that facilitate working with transformers.

Our key contributions are as follows:

- **Transformer Viability in Online RL:** We demonstrate that transformers can be effectively trained in online RL for continuous control tasks, not requiring explicit memory usage, achieving performance comparable to MLP-based baselines even in MDP settings.
- **Empirical Insights and Training Recommendations:** We conduct a series of experiments to gain a deeper understanding of transformer performance in online control tasks. Our find-

*Equal contribution.

ings provide valuable insights and practical recommendations for stabilizing and effectively training transformers in Online RL.

## 2 TRANSFORMERS IN ONLINE RL

To the best of our knowledge, unlike fully-offline and semi-offline approaches, which are well-developed, online setting faces more difficulties and less covered by the research. Esslinger et al. (2022) utilizes transformer model with Deep Q-Network-like (DQN) training algorithm to train model from scratch in online setting. Initially developed for natural language processing (NLP), Transformer-XL (Dai, 2019) (TrXL) has a recurrent structure of processing input sequences which eliminates the need of capturing all dependent tokens into one sequence. Parisotto et al. (2020) proposed a modification of the TrXL which is developed especially for the RL, enhanced by gating mechanisms and turned out to be a powerful model in memory tasks (Pleines et al., 2023). Gating mechanisms stabilize large multi-layered transformers by giving them the opportunity to bypass the attention and feed-forward network inside the transformers block. Addressing the problems of transformer's quadratic computational comlexity, Pramanik et al. (2023) proposed Recurrent Linear Transformer (ReLiT) and Approximate Recurrent Linear Transformer (AReLiT) with context-independent inference cost and their gated versions.

## 3 BACKGROUND

### 3.1 TRANSFORMERS

Transformers (Vaswani et al., 2017) have become a widely used model in various domains due to their attention module, which detects dependencies between input sequences of tokens. Formally, the attention module can be described by equation 1.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{1}$$

where: $Q = XW^Q$ – query matrix, $K = XW^K$ – key matrix, $V = XW^V$ – value matrix, $X$ – input sequence, $d_k$ – inner dimension, $W^Q, W^K, W^V$ – learnable parameters

### 3.2 REINFORCEMENT LEARNING

Markov Decision Process (MDP) is defined as a tuple: $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ where: $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, $\mathcal{P}(s'|s, a)$ is a state transition probability function, $\mathcal{R}(s, a, s')$ – a reward function, and $\gamma \in [0, 1]$ – a discount factor.

The goal in the RL is to find a policy $\pi^*(a|s)$ that maximizes the expected cumulative reward $\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$ where $r_t$ is the reward received at time $t$.

## 4 EXPERIMENTAL SETUP

### 4.1 ENVIRONMENTS

**MuJoCo** (Todorov et al., 2012) is a set of motion control tasks which trains agent to efficiently control movements. In this paper, we use HalfCheetah-v4, Ant-v4, Hopper-v4, Humanoid-v4, Walker-v4, Pusher-v4, Reacher-v4 tasks. All the environments used in this work are vector-based.

**ManiSkill3** (Tao et al., 2024) is a framework for robotic manipulation powered by SAPIEN (Xiang et al., 2020). It has a wide range of robotic tasks which can be solved by RL algorithms. We choose PushT-v0, PickCube-v0, TriFingerRotateCube-v0, PushCube-v0 and PullCube-v0 tasks to demonstrate transformers ability to solve them and use vector-based environments for both of them.

### 4.2 TRANSFORMER BASELINES

In this study, we employed both on-policy and off-policy RL algorithms: Twin Delayed Deep Deterministic Policy Gradient (TD3), Soft-Actor-Critic (SAC) (Haarnoja et al., 2018), and Proximal

Policy Optimization (PPO) to train transformer-based models. Our goal was to identify general patterns in transformer training within online RL settings, independent of the specific RL algorithms and their parameters. For PPO and TD3 detailed description see subsection A.1 and subsection A.2 respectively.

As the transformer backbone, we employed a transformer architecture without any special modifications for the RL setting. One of the primary transformer backbones used for preprocessing sequences of observations was GPT-2 (Radford et al., 2019), integrated with flash attention (Dao, 2023) to speed up training. The corresponding model variations are referred to as TD3-GPT, SAC-GPT and PPO-GPT (Figure 1). Additionally, we experimented with a TD3 Vanilla Transformer Encoder variant (TD3-TransEncoder), which utilizes a classical transformer encoder (Vaswani et al., 2017) for actor and uses MLP-based critic.

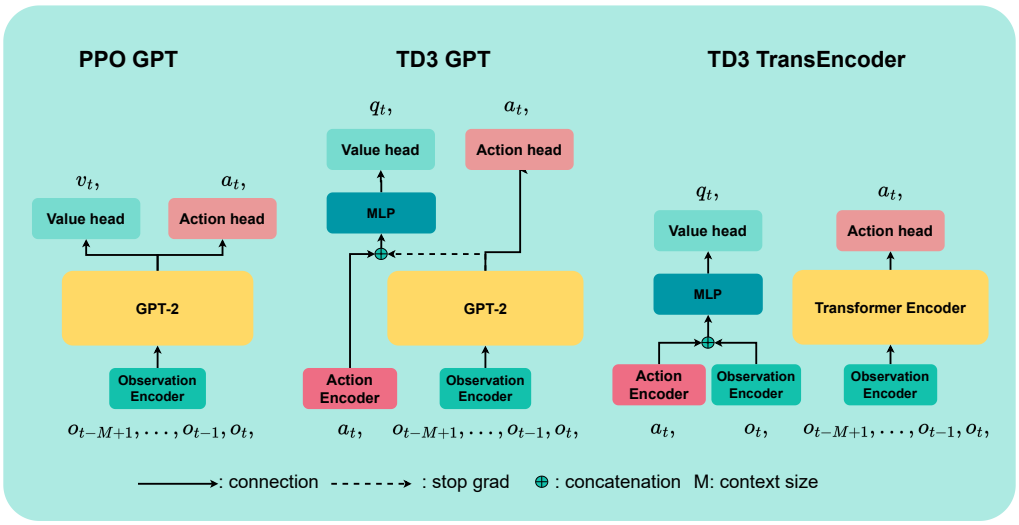


Figure 1: PPO-GPT, TD3-GPT with shared sequence encoder and TD3-TransEncoder with separated MLP critic.

### 4.3 Experimental protocol

For each experiment, we conducted three runs of the agents with different initializations and performed evaluation during training using 100 random seeds ranging from 0 to 99. The results are presented as the mean episodic reward (or success rate) $\pm$ the standard deviation (std). All the parameters of the models used in the experiments are listed in Table 5 and Table 6.

## 5 Experiments

Developing transformers in online RL requires careful handling to ensure stability and high performance. In this section, we conduct a study of the transformer architecture: we compare it with classical baselines, study it's weaknesses and subtleties of the replay buffer design, check transformers resistance to scaling, provide an exploration of the gatings behavior and compare transformer with its sequence-to-sequence alternatives.

To ensure consistency and eliminate potential bias from hyperparameter tuning, we fixed all RL algorithm parameters for TD3, SAC and PPO at the beginning of our experiments. These parameters remained unchanged across all modifications and experimental setups, allowing us to isolate the effects of transformer architectures on learning dynamics.

### 5.1 Transformer Comparison with MLP and LSTM Baselines

**Research Question.** How effectively and stably can transformer-based models be trained in control tasks by online RL algorithms, and how do their performance compare to MLP and LSTM-based models under identical RL parameter settings?

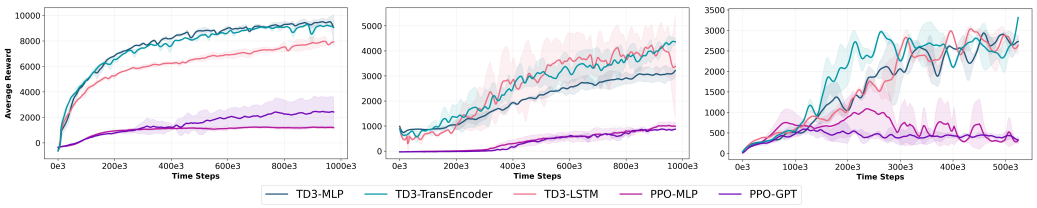


Figure 2: Performance of the baselines on HalfCheetah (left), Ant (center) and Hopper (right).

**Details.** In this experiment, we used TD3, PPO and Soft-Actor-Critic (SAC) algorithms to evaluate our baselines in MuJoCo and ManiSkill environments. All the RL parameters are fixed and listed in Table 3. TD3 and PPO algorithms were taken from the official CleanRL implementation with fixed original RL parameters. SAC implementation was taken from the original ManiSkill3 codebase with its original parameters. We compare TD3-TransEncoder and PPO-GPT with these baselines by training in control environments such as HalfCheetah, Ant, Hopper. Also, we provide experiments with PickCube, PushT and TriFinger- RotateCube environments using SAC GPT. Experiments with TD3 algorithm on PushCube and PullCube environments are available in the Appendix.

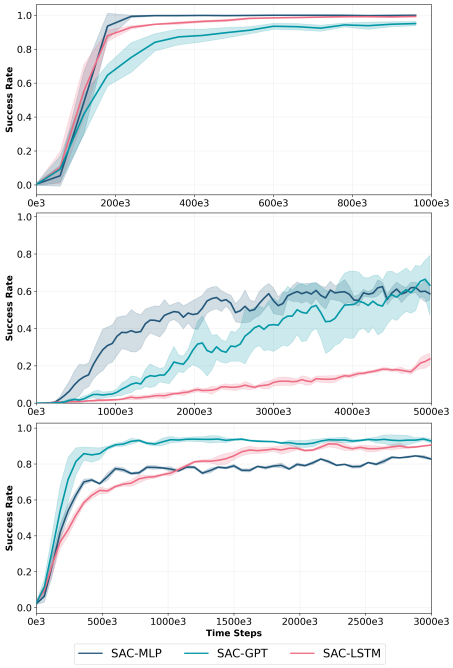


Figure 3: Performance on ManiSkill3 tasks: PickCube (above), PushT (middle), TriFinger-RotateCube (below).

**Interpretation.** Results on Figures 2, 3 show that transformer-encoder modification achieves performance comparable to MLP model on TD3 algorithm while GPT-based architecture trained on SAC achieves results comparable to SAC MLP, despite the conventional expectation that transformers require more data and computational resources for effective training. This indicates that even small transformers can be adequately optimized in online RL tasks without major modifications to the training pipeline.

Additionally, according to Figures 2, 3 the training stability of transformers remained at an acceptable level compared to MLP models, regardless of their higher complexity. This suggests that transformers can effectively leverage sequential representations without significant failures in learning.

> **Practical Takeaway:** Even without extensive hyperparameter tuning, transformer-based models can achieve performance comparable to well-established MLP-based baselines in continuous control MDP tasks across both on-policy and off-policy RL algorithms. This indicates that transformers can be effectively used in online RL without requiring major modifications to the algorithm.

### 5.2 Comparison of Scaling

**Research Question.** How well do transformers scale compared to MLPs in terms of performance, and should we expect a loss of stability from larger models?

**Details.** To evaluate the change in performance of transformers and MLPs as the number of model parameters increases, we tested two variations: PPO-MLP and PPO-GPT. We fixed the RL parameters across both models and increased: the number of transformer layers from 1 to 6 and the hidden dimension from 128 to 256. Similarly, for MLP: the number of layers from 2 to 5 and the hidden dimension from 64 to 512 (see Table 6). As a result, both models had a comparable number of parameters (about 1.5M). We then tested these models on three standard Mujoco environments.

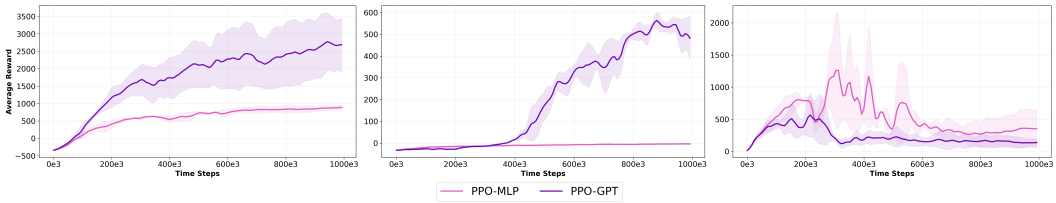


Figure 5: Performance of 1.5M PPO-GPT and PPO-MLP on HalfCheetah (left), Ant (center) and Hopper (right).

**Interpretation.** Our results showed that increasing the number of parameters in MLP-based models caused them to fail to learn properly under the original training settings. As seen in Figure 5, learning quality significantly degraded. In contrast, transformers were less affected by the parameter increase, showing higher performance.

> **Practical Takeaway:** Transformers are more robust to parameter scaling rather than MLPs: MLP-based models does not scale well without careful tuning.

### 5.3 Shared/Separate Transformer Encoder

**Research Questions.** How does the separation or sharing of the transformer encoder between the actor and the critic affect the stability and efficiency of learning in actor-critic RL algorithms?

**Details.** In this experiment, we employed TD3-GPT – an actor-critic algorithm, integrating a transformer backbone to encode the sequence of observations. We tested different methods of using the transformer encoder:

1. Shared Transformer – the actor and critic share a common transformer, which both can update via their gradients. 2. Shared Transformer with Freezing – the actor and critic share a common transformer, but during backpropagation from the critic's loss, the transformer backbone is frozen. Only the actor can update the transformer. 3. Separate Transformer – both the actor and the critic have their own transformer for processing observation sequences.

To isolate the impact of other training effects associated with transformer-based actor-critic models, as discussed in Section 5.5, we conducted this experiment in environments without terminations, such as HalfCheetah, Pusher, and Reacher.

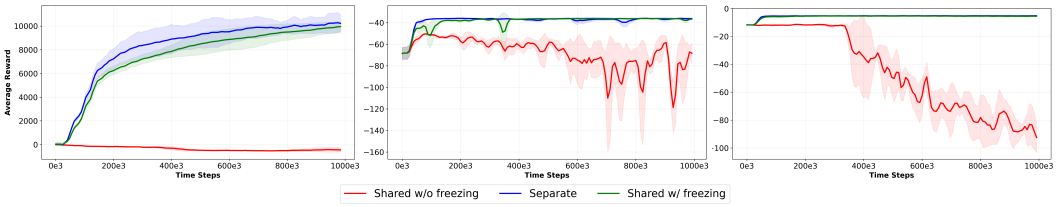


Figure 6: Comparison of different TD3-GPT architectures on HalfCheetah (left), Pusher (center) and Reacher (right).

**Interpretation.** Experiments (Figure 6) revealed that when both the actor and the critic update the same transformer, gradient conflicts arise: the actor attempts to optimize the policy by maximizing

rewards, on the other hand, the critic, learns to predict state values by minimizing TD error. These two objectives are distinct and can even partially contradict each other. As a result, the transformer receives conflicting gradient updates, leading to unstable learning or even complete suppression due to gradient conflict.

**Practical Takeaway:** Sharing a transformer backbone between actor and critic in off-policy algorithms leads to gradient conflicts and instability. A separate transformer ensures stability but increases computational cost. Freezing the shared transformer for the critic offers a balanced trade-off between efficiency and stability.

### 5.4 Shared/Separate MLP observation Encoder

**Research Question.** Previous research has reported similar issues with sharing in RNN-based off-policy architectures (Ni et al., 2023). Is this a general issue across all sequential models, or does it extend beyond them? Do such problems also arise with MLP-based encoders?

**Details.** To conduct this experiment, similarly to the previous one but with MLP, we added an additional linear layer to the TD3-MLP architecture for preprocessing observations. We then tested two settings: 1. Shared MLP Encoder Layer – The actor and critic share an MLP encoder layer, which both can update using their gradients. 2. Separate MLP Encoder Layer – The actor and critic each have their own MLP encoder layer for processing observation sequences.

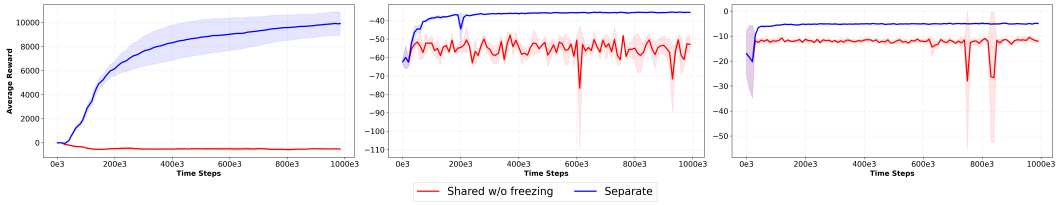


Figure 7: Comparison of different TD3-MLP architectures on HalfCheetah (left), Pusher (center) and Reacher (right).

**Interpretation.** Results in Figure 7 show that the issues associated with encoder sharing are not exclusive to sequential models; they also arise in MLP architectures. Our experiment demonstrated that similar learning instabilities occur when an MLP agent shares an encoder without freezing it.

The experiment confirmed that the issues previously observed in RNNs and transformers with shared encoders also manifest themselves in MLPs. When the actor and critic share an MLP encoder and update it with their gradients, conflicts in learning arise, leading to convergence degradation.

**Practical Takeaway:** The sharing issue is not unique to sequential models (RNNs, Transformers) but is also present in MLPs. This indicates a fundamental difficulty in jointly training the actor and critic in off-policy alorithms with the same feature extraction function in state based enviroments.

### 5.5 Comparison of different batch formation methods

**Research Question.** How does batch formation impact off-policy transformer-based models training in online RL on continuous control tasks?

**Details.** In this experiment, we trained TD3-GPT in environments with fixed and variable episode lengths using different batch formation methods: Method 1 - rolling window of context length within episodes, Method 2 - rolling window of context length across all episodes. According to the Figure 8, the key difference between Method 2 and Method 1 is the ability to create sequences that include observations from multiple episodes.

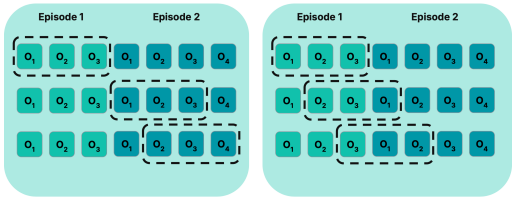


Figure 8: Visualization of different batch formation: Method 1 (left), Method 2 (right).

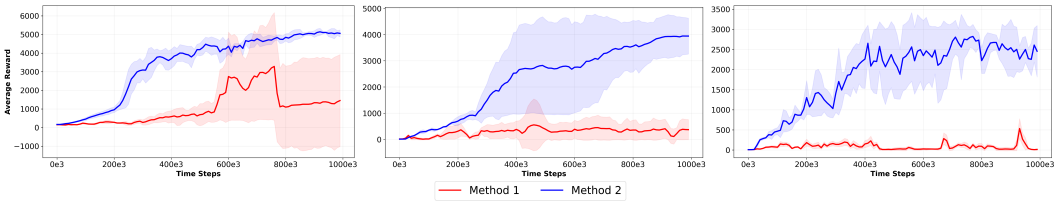


Figure 9: Comparison of different batch formation methods for TD3-GPT on Humanoid (left), Walker (center) and Hopper (right).

**Interpretation.** Results in Figure 9 show that in environments with a fixed episode length, such as HalfCheetah, Pusher, and Reacher, both methods performed well. However, in environments where early episode termination is possible, such as Humanoid, Walker and Hopper, training with Method 1 failed to work effectively. In these environments, the agent needs to take correct actions from the very first steps. The sequential nature of transformer models causes the data in training batches to become highly correlated, especially at the initial stages of training in environments that allow early termination. This high correlation makes it more difficult for the learning signal associated with successful actions to propagate. Method 1 constructs batches only within a single episode, causing the training to overlook cases where the agent quickly terminates an episode due to poor actions.

In these environments, Method 2 (batching across episodes) is preferable because it allows the inclusion of data from different episodes. This provides the model with a clearer learning signal and insights into which actions lead to early termination failures and which lead to successful episode continuation, thereby making the training more effective.

**Practical Takeaway:** If early episode termination is possible, the model must see enough successful cases to learn effective strategies. For environments that allow early episode termination, batching across episodes is preferable because it improves adaptation from the start of an episode and provides the model with a clearer learning signal.

### 5.6 Gatings Impact on the Learning Process

**Research Question.** How attention mechanism adjust to MDP environments during training of the transformer-based models and how gating mechanisms influence the adaptation of transformer-based models?

**Details.** In this experiment, we incorporated **gating mechanisms** into the transformer architecture. Specifically, we employed **GRU-based gating**, as proposed by Parisotto et al. (2020), to control the information flow through the attention mechanism. Instead of using a standard skip connections, we replaced it with a two GRU gatings: the first one is applied after the multihead attention, and the second one is applied after the FFN module.

$$r = \sigma(W_r^{(l)} y + U_r^{(l)} x), \quad z = \sigma(W_z^{(l)} y + U_z^{(l)} x - b_g^{(l)}),$$

$$\hat{h} = \tanh(W_g^{(l)} y + U_g^{(l)} (r \odot x)), \quad g^{(l)}(x, y) = (1 - z) \odot x + z \odot \hat{h}.$$

To quantify the effect of this modification, we logged mean values of $1 - z$ during training. This vector determines how much of the information bypasses the attention or FFN module versus how much is updated based on the attention outputs. A high value of $1 - z$ for attention gate implies that a significant portion of the input information bypasses the attention mechanism, because the gate itself allows the transformer to completely ignore the information coming from the attention block, so potentially the transformer can be reduced to an MLP-based model if $(1 - z) \to I$.

We conducted experiments using the TD3-TransEncoder architecture in two robotic manipulation environments from ManiSkill3.

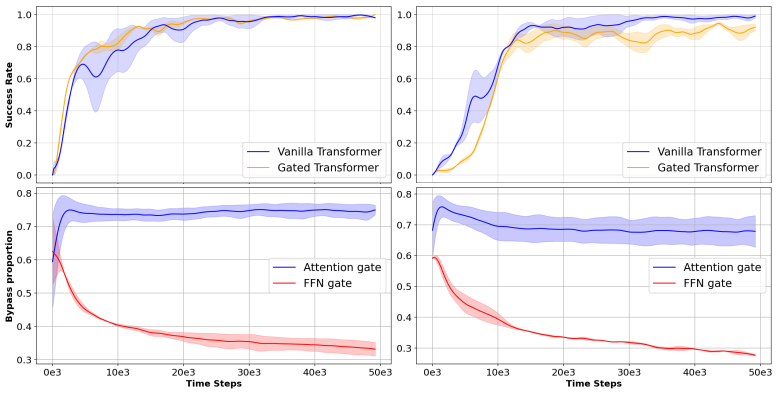


Figure 10: Success rate (top) and gate bypass proportion (bottom) on PushCube (left) and PullCube (right) tasks of TD3-TransEncoder.

**Interpretation.** Our results on the Figure 10 reveal a clear trend: throughout training, in both PushCube and PullCube, the GRU gating module bypasses up to about 75% of the information from the attention module, allowing it to pass unchanged through the network. This suggests that the transformer learns to rely less on attention-based updates and instead prioritizes direct propagation of previous information. This insight can be interpreted from the perspective that the gating mechanism helps transformer ”understand” the MDP properties of the environment and bypass information directly to the FFN module. This theory is partially supported by the fact that the mean value of the FFN decreases to about 30% during training, indicating that the transformer uses its skip connection for only 30% of the information.

At the same time, the gated transformer exhibits approximately the same performance during training. Despite the significant impact of gatings in memory tasks which require multi-layered transformers, small models in MDP environments do not benefit from them. Other comparisons can be found in the Figure 17.

**Practical Takeaway:** GRU-based gating mechanisms allow transformers to regulate their reliance on the attention mechanism by dynamically adjusting how much information bypasses attention blocks. In MDP environments, the transformer naturally shifts toward bypassing attention, indicating that the model can efficiently extract task-relevant information from state observations alone.

### 5.7 Attention Alternatives

**Research Question.** How do models based on alternative attention mechanisms compare to the standard self-attention mechanism in continuous control MDP tasks? Can replacing traditional attention-based transformers with alternative architectures improve performance?

**Details.** In this experiment, we investigated whether modifying the attention mechanism could enhance model performance in online reinforcement learning. Specifically, we evaluated TD3-TransEncoder (”Vanilla Transformer” on the Figure 11) and its two modifications: Differential

Attention Transformer and Mamba-based model. These models were tested on multiple continuous control environments to assess their effectiveness compared to the standard transformer encoder.

Differential Attention Transformer (Ye et al., 2024) introduces an attention modification that partitions the query and key vectors into two groups, producing separate softmax attention maps that are then subtracted. This aims to improve stability by denoising the attention mechanism.

Mamba (Gu & Dao, 2023), on the other hand, replaces the attention mechanism with a structured state-space model (SSM). Unlike transformers, which rely on self-attention with quadratic complexity, Mamba processes sequences using a recurrent structure with linear-time complexity, making it a potential alternative for long-range dependencies.

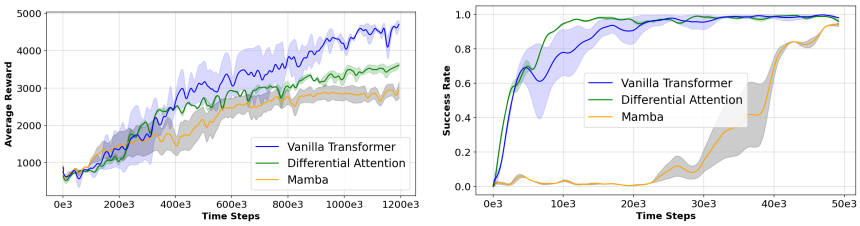


Figure 11: Averaged reward on Ant (left) and success rate on PushCube (right) environments.

**Interpretation.** According to Figure 11, the differential attention does not impact the training process and failed to achieve superior overall results. In some cases, it reached peak performance faster (e.g., in the PushCube task), but in others, it underperformed compared to the standard transformer. Other comparisons can be found in the Figure 18. Also the recurrent properties of Mamba do not contribute to the stabilization or improvement of the learning process. On the contrary, in almost all experiments, Mamba performed worse, requiring more time to achieve efficiency comparable to the transformer, or not achieving it at all.

> **Practical Takeaway:** A simple replacement of the attention mechanism with Differential Attention or Mamba does not provide clear advantages in MDP continuous control tasks.

## 6 Discussion

| | **TD3-GPT(ours)** | **TD3-MLP(ours)** | TD3-MLP* | SAC-MLP* | PPO-GRU* | A2C-GRU* | SAC-Transformer* | VRM* | MF-RNN* | GPIDE-ESS* | RESeL* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| AntBLT-V-v0 | 922 ± 82 | 549 ± 221 | 476 ± 114 | 651 ± 65 | 690 ± 158 | 264 ± 60 | 692 ± 89 | 291 ± 23 | 1137 ± 178 | 1017 ± 80 | 1971 ± 60 |
| HalfCheetahBLT-V-v0 | 1324 ± 150 | 343 ± 96 | 177 ± 115 | 513 ± 77 | 1072 ± 195 | -412 ± 191 | -449 ± 72 | -1443 ± 220 | 2073 ± 69 | 1886 ± 165 | 2678 ± 176 |
| HopperBLT-V-v0 | 759 ± 80 | 98 ± 58 | 223 ± 28 | 243 ± 4 | 438 ± 126 | 301 ± 155 | 240 ± 79 | 476 ± 28 | 1003 ± 426 | 2537 ± 167 | 2480 ± 91 |
| AntBLT-P-v0 | 1984 ± 105 | 846 ± 34 | 897 ± 83 | 1147 ± 49 | 2103 ± 80 | 916 ± 60 | 894 ± 36 | 323 ± 37 | 352 ± 88 | 2597 ± 76 | 2829 ± 56 |
| HalfCheetahBLT-P-v0 | 1788 ± 319 | 824 ± 137 | 906 ± 19 | 970 ± 47 | 1460 ± 143 | 353 ± 74 | 1400 ± 655 | -1317 ± 217 | 2802 ± 88 | 2466 ± 129 | 2900 ± 179 |
| HopperBLT-P-v0 | 1971 ± 243 | 525 ± 224 | 490 ± 140 | 310 ± 35 | 1592 ± 60 | 467 ± 78 | 1763 ± 493 | 557 ± 85 | 2234 ± 102 | 2373 ± 568 | 2769 ± 85 |
| WalkerBLT-P-v0 | 1516 ± 190 | 451 ± 109 | 505 ± 32 | 483 ± 86 | 651 ± 156 | 200 ± 104 | 1150 ± 352 | 372 ± 96 | 940 ± 272 | 1502 ± 521 | 2505 ± 96 |

Table 1: Average Return on the Mujoco tasks with masked velocities/positions at 1.5M steps ± std over 6 seeds. * – results from Luo et al. (2024).

Through a series of experiments, we have demonstrated that transformer-based models perform competitively in MDP continuous control tasks. Their ability to achieve results comparable to well-established baselines suggests that transformers can be a viable alternative to traditional architectures in online reinforcement learning for continuous control. Furthermore, our experiments in POMDP (Table 1) environments indicate promising results when compared to other baseline models. These findings highlight the potential of transformers to effectively capture temporal dependencies and leverage sequential representations even in settings where full observability is not guaranteed.

Given these insights, we see potential for transformers to become a universal architecture for online RL, much like their success in supervised learning. However, realizing this vision requires a deeper understanding of how transformers behave across different settings and task distributions in online RL. Stability, training efficiency, and adaptability remain key areas for investigation. Future research

should focus on evaluating transformers' ability to generalize, especially in multi-task RL and meta-learning settings. Additionally, further studies are needed to explore their role in memory retention, credit assignment, and long-horizon planning for continuous control tasks. By addressing these challenges, we can unlock the full potential of transformers in online RL and establish more robust and scalable training pipelines.

## 7 Conclusion

In this work, we investigated the potential of transformer-based models in online RL for continuous control tasks. Our primary objective was to assess whether transformers can be effectively trained in fully online settings without requiring substantial modifications to the training pipeline. Through a series of experiments, we demonstrated that transformers can achieve performance comparable to classical MLP-based architectures when trained using standard online RL algorithms such as TD3 and PPO. Additionally, we provided insights into the challenges and design choices necessary for stabilizing transformer training in online RL.

Our results highlight the capability of transformers to function as a viable alternative to traditional architectures, offering flexibility in handling sequential dependencies. However, our study primarily focused on MDP environments, and the behavior of transformers in POMDP settings remains an open question. Exploring their robustness in partially observable environments and extending their application to vision-based online RL tasks are promising directions for future research.

We believe that our work provides valuable insights into the training dynamics of transformers in online RL and serves as a stepping stone for further advancements in this field.

## References

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Zihang Dai. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning, 2023. URL `https://arxiv.org/abs/2307.08691`.

Ahmad Elawady, Gunjan Chhablani, Ram Ramrakhya, Karmesh Yadav, Dhruv Batra, Zsolt Kira, and Andrew Szot. Relic: A recipe for 64k steps of in-context reinforcement learning for embodied ai. *arXiv preprint arXiv:2410.02751*, 2024.

Kevin Esslinger, Robert Platt, and Christopher Amato. Deep transformer q-networks for partially observable reinforcement learning. *arXiv preprint arXiv:2206.01078*, 2022.

Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

Anirudh Goyal, Abram Friesen, Andrea Banino, Theophane Weber, Nan Rosemary Ke, Adria Puigdomenech Badia, Arthur Guez, Mehdi Mirza, Peter C Humphreys, Ksenia Konyushova, et al. Retrieval-augmented reinforcement learning. In *International Conference on Machine Learning*, pp. 7740–7765. PMLR, 2022.

Jake Grigsby, Justin Sasek, Samyak Parajuli, Daniel Adebi, Amy Zhang, and Yuke Zhu. Amago-2: Breaking the multi-task barrier in meta-reinforcement learning with transformers. *arXiv preprint arXiv:2411.11188*, 2024.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts. *arXiv preprint arXiv:2210.03094*, 2(3):6, 2022.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Andrew Lampinen, Stephanie Chan, Andrea Banino, and Felix Hill. Towards mental time travel: a hierarchical memory for reinforcement learning agents. *Advances in Neural Information Processing Systems*, 34:28182–28195, 2021.

TP Lillicrap. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Haoyu Lu, Wen Liu, Bo Zhang, Bingxuan Wang, Kai Dong, Bo Liu, Jingxiang Sun, Tongzheng Ren, Zhuoshu Li, Hao Yang, et al. Deepseek-vl: towards real-world vision-language understanding. *arXiv preprint arXiv:2403.05525*, 2024.

Fan-Ming Luo, Zuolin Tu, Zefang Huang, and Yang Yu. Efficient recurrent off-policy rl requires a context-encoder-specific learning rate, 2024. URL `https://arxiv.org/abs/2405.15384`.

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Tianwei Ni, Michel Ma, Benjamin Eysenbach, and Pierre-Luc Bacon. When do transformers shine in rl? decoupling memory from credit assignment. *Advances in Neural Information Processing Systems*, 36:50429–50452, 2023.

Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. In *International conference on machine learning*, pp. 7487–7498. PMLR, 2020.

Marco Pleines, Matthias Pallasch, Frank Zimmer, and Mike Preuss. Memory gym: Partially observable challenges to memory-based agents in endless episodes. *arXiv preprint arXiv:2309.17207*, 2023.

Subhojeet Pramanik, Esraa Elelimy, Marlos C Machado, and Adam White. Recurrent linear transformers. *arXiv preprint arXiv:2310.15719*, 2023.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI*, 2019. URL `https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf`. Accessed: 2024-11-15.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/radford21a.html`.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL `https://arxiv.org/abs/1707.06347`.

Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, R Devon Hjelm, Philip Bachman, and Aaron C Courville. Pretraining representations for data-efficient reinforcement learning. *Advances in Neural Information Processing Systems*, 34:12686–12699, 2021.

Yanchao Sun, Shuang Ma, Ratnesh Madaan, Rogerio Bonatti, Furong Huang, and Ashish Kapoor. Smart: Self-supervised multi-task pretraining with control transformers. *arXiv preprint arXiv:2301.09816*, 2023.

Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse-kai Chan, et al. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *arXiv preprint arXiv:2410.00425*, 2024.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. *Proceedings of the ... IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 10 2012. doi: 10.1109/IROS.2012.6386109.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11097–11107, 2020.

Peng Xu, Xiatian Zhu, and David A Clifton. Multimodal learning with transformers: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(10):12113–12132, 2023.

Tianzhu Ye, Li Dong, Yuqing Xia, Yutao Sun, Yi Zhu, Gao Huang, and Furu Wei. Differential transformer. *arXiv preprint arXiv:2410.05258*, 2024.

Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. In *international conference on machine learning*, pp. 27042–27059. PMLR, 2022.

# A APPENDIX

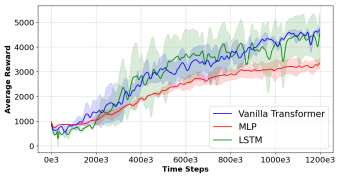


Figure 12: TD3-MLP, TD3-LSTM and TD3-Trans Encoder comparison on Ant environment.

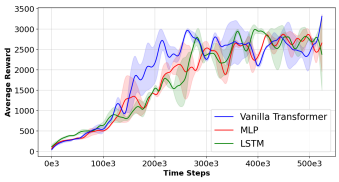


Figure 13: TD3-MLP, TD3-LSTM and TD3-Trans Encoder comparison on Hopper environment.

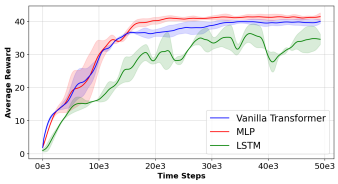


Figure 14: TD3-MLP, TD3-LSTM and TD3-Trans Encoder comparison on PullCube environment (Reward).

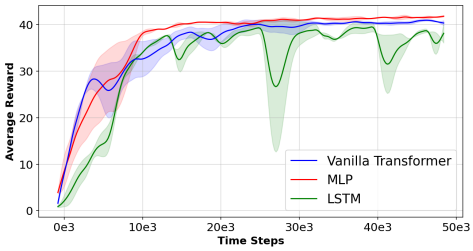


Figure 15: TD3-MLP, TD3-LSTM and TD3-Trans Encoder comparison on PushCube environment (Reward).

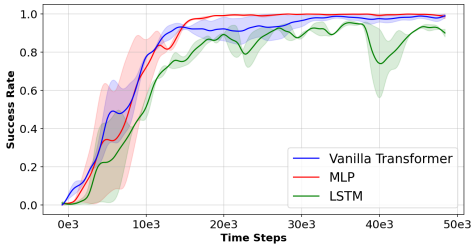


Figure 16: TD3-MLP, TD3-LSTM and TD3-Trans Encoder comparison on PullCube environment (Success Rate).

Table 2: Correlations of training parameters and average episodic reward in MuJoCo environments.

| PARAMETER | CORRELATION |
|---|---|
| $\gamma$-DISCOUNT | +0.76 |
| $\tau$-SOFT UPDATE | +0.14 |
| POLICY NOISE | +0.03 |
| NOISE CLIP | +0.01 |
| EXPLORATION NOISE | -0.14 |
| BATCH SIZE | -0.23 |

Table 3: Hyperparameters for TD3-based baselines.

| | TD3 Transformer Encoder | | TD3-GPT | TD3-MLP |
|---|---|---|---|---|
| Parameter | ManiSkill | MuJoCo | MuJoCo | MuJoCo |
| $\gamma$-discount | 0.8 | 0.99 | 0.99 | 0.99 |
| $\tau$-soft update | 0.01 | 0.007 | 0.005 | 0.005 |
| Policy noise | 0.2 | 0.2 | 0.2 | 0.2 |
| Noise clip | 0.5 | 0.2 | 0.5 | 0.5 |
| Exploration noise | 0.1 | 0.1 | 0.1 | 0.1 |
| Batch size | 500 | 256 | 256 | 256 |
| Learning Rate | $3 \times 10^{-4}$ | $3 \times 10^{-4}$ | $3 \times 10^{-4}$ | $3 \times 10^{-4}$ |
| Buffer size | $0.05 \times 10^{6}$ | $0.5 \times 10^{6}$ | $1.5 \times 10^{6}$ | $1.5 \times 10^{6}$ |
| Learning Starts | 600 | 25000 | 25000 | 25000 |
| Seeds | 1,2,3,4 | 1,2,3,4 | 1,2,3,4 | 1,2,3,4 |
| Num layers | 1 | 1 | 1 | 2 |
| Num heads | 2 | 2 | 4 | - |
| Dim model | 256 | 256 | 128 | - |
| Dim feedforward | 512 | 512 | 256 | 256 |
| Dropout | 0.05 | 0.05 | 0.0 | - |
| Context len | 3 | 3 | 10 | - |

Table 4: Hyperparameters for SAC-based baselines.

| Parameter | SAC-GPT | SAC-LSTM | SAC-MLP |
|---|---|---|---|
| $\gamma$ (discount) | 0.8 | 0.8 | 0.8 |
| $\tau$ (tau) | 0.01 | 0.01 | 0.01 |
| $\alpha$ (alpha) | 0.2 | 0.2 | 0.2 |
| Update epochs | 10 | 10 | 10 |
| Batch size | 1024 | 1024 | 1024 |
| Learning starts | 4000 | 4000 | 4000 |
| Learning rate | $3 \times 10^{-4}$ | $3 \times 10^{-4}$ | $3 \times 10^{-4}$ |
| Seeds | 1,2,3,4 | 1,2,3,4 | 1,2,3,4 |
| Num. layers | 1 | 1 | 3 |
| Num. heads | 2 | - | - |
| Dim. model | 256 | 256 | 256 |
| Dim. feedforward | 512 | 512 | 512 |
| Context length | 10 | 10 | - |

Table 5: Hyperparameters for PPO-based baselines.

| PARAMETER | PPO-GPT | PPO-MLP |
|---|---|---|
| $\gamma$-DISCOUNT | 0.99 | 0.99 |
| GAE $\lambda$ | 0.95 | 0.95 |
| MAX GRAD NORM | 0.5 | 0.5 |
| CLIP COEF | 0.2 | 0.2 |
| VF COEF | 0.5 | 0.5 |
| UPDATE EPOCH | 10 | 10 |
| NUM MINIBATCHES | 32 | 32 |
| BATCH SIZE | 2048 | 2048 |
| LEARNING RATE | $3 \times 10^{-4}$ | $3 \times 10^{-4}$ |
| SEEDS | 1,2,3,4 | 1,2,3,4 |
| NUM LAYERS | 1 | 3 |
| NUM HEADS | 4 | - |
| DIM MODEL | 128 | - |
| DIM FEEDFORWARD | 128 | 64 |
| CONTEXT LEN | 10 | - |

Table 6: Hyperparameters for the scaling experiment.

| PARAMETER | PPO-GPT | PPO-MLP |
|---|---|---|
| $\gamma$-DISCOUNT | 0.99 | 0.99 |
| GAE $\lambda$ | 0.95 | 0.95 |
| MAX GRAD NORM | 0.5 | 0.5 |
| CLIP COEF | 0.2 | 0.2 |
| VF COEF | 0.5 | 0.5 |
| UPDATE EPOCH | 10 | 10 |
| NUM MINIBATCHES | 32 | 32 |
| BATCH SIZE | 2048 | 2048 |
| LEARNING RATE | $3 \times 10^{-4}$ | $3 \times 10^{-4}$ |
| SEEDS | 1,2,3,4 | 1,2,3,4 |
| NUM LAYERS | 6 | 5 |
| NUM HEADS | 4 | - |
| DIM MODEL | 256 | - |
| DIM FEEDFORWARD | 128 | 512 |
| CONTEXT LEN | 10 | - |

**Supplementary materials for gating exploration :**

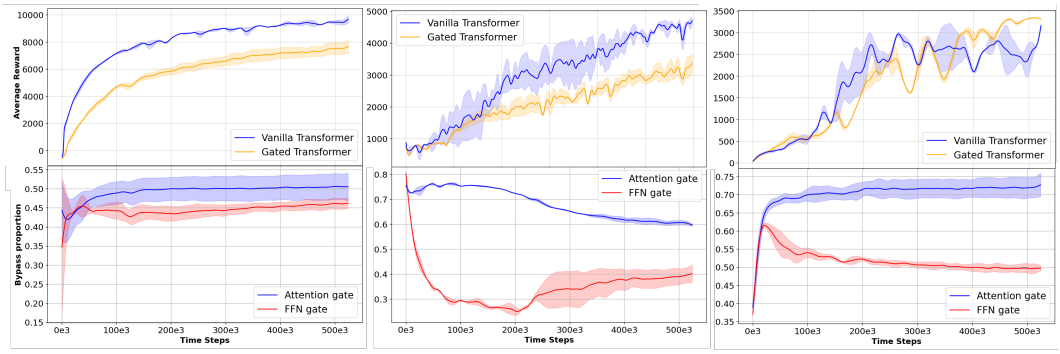


Figure 17: HalfCheetah(left), Ant(center), Hopper(right) rewards with their corresponding bypass proportions.

**Supplementary materials for attention alternatives exploration :**

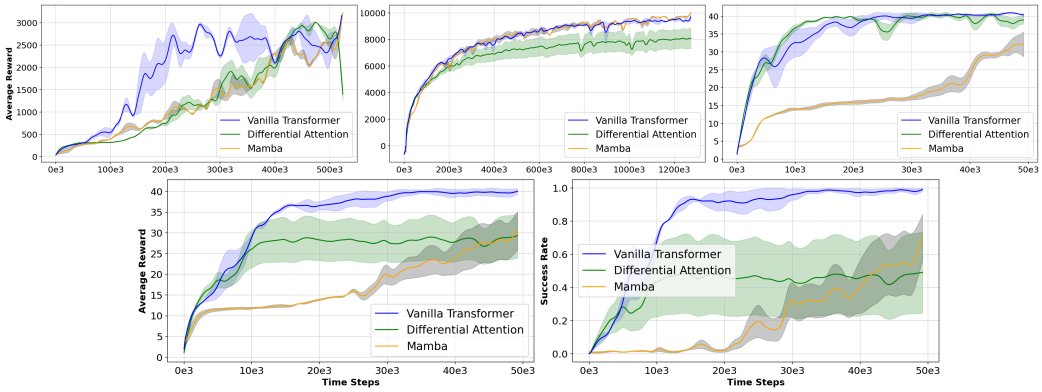


Figure 18: **top line:** averaged rewards on Hopper(left), HalfCheetah(middle), PushCube(right) **bottom line:** averaged reward(left) and success rate(right) on PullCube.

### A.1 PPO ALGORITHM

Proximal Policy Optimization (PPO) Schulman et al. (2017) is an on-policy algorithm that improves stability and efficiency over traditional policy gradient methods. It achieves this by constraining the policy update to prevent overly large changes, which can destabilize training.

**PPO optimizes a surrogate objective function** with a clipped probability ratio:

$$J(\theta) = \mathbb{E}_t \left[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)\right] \tag{2}$$

where: $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio of the new and old policies. $A_t$ is the advantage function, estimating the relative benefit of taking action $a_t$ in state $s_t$. $\epsilon$ is a small clipping threshold (e.g., $0.2$), limiting how much $r_t(\theta)$ can deviate from $1$.

PPO also employs a value function loss for training the critic network:

$$L_V(\theta) = \mathbb{E}_t \left[(V_\theta(s_t) - R_t)^2\right] \tag{3}$$

where $V_\theta(s_t)$ is the estimated state value and $R_t$ is the empirical return.

### A.2 TD3 ALGORITHM

Twin Delayed Deep Deterministic Policy Gradient Fujimoto et al. (2018) (TD3) is an off-policy algorithm that utilizes a replay buffer to learn an optimal policy. It uses an actor-critic pipeline for training: the actor $\pi_\phi(s)$ improves its performance by updating its weights in the direction of the critic's gradient ascent. At the same time, the critic $Q_\theta(s, a)$ is trained to approximate the Q-value function by minimizing the temporal difference error. In order to avoid overly optimal estimations of the critic like in Deep Deterministic Policy Gradient Lillicrap (2015), TD3 has two critics that approximate Q-function separately.

**Critic Loss:** The loss for each critic is computed as the mean squared error between the predicted and target Q-value:

$$\mathcal{L}_{\text{critic}} = \frac{1}{N}\sum_{i=1}^{N}\left(Q_{\theta_j}(s_i, a_i) - y_i\right)^2, \quad j \in \{1, 2\} \tag{4}$$

where $y_i = r_i + \gamma \min_{j=1,2} \hat{Q}_{\theta_j}(s_{i+1}, \hat{\pi}_\phi(s_{i+1}) + \epsilon)$ is a target value, $r_i$ is a reward, $\hat{\pi}_\phi, \hat{Q}_{\theta_j}$ is a target actor, $j$-th target critic, $\epsilon \sim \text{clip}(\mathcal{N}(0, \sigma^2), -c, c)$ is clipped Gaussian noise and $N$ is a batch size.

**Actor Loss:** The actor loss is computed to maximize the expected Q-value under the current policy:

$$\mathcal{L}_{\text{actor}} = -\frac{1}{N}\sum_{i=1}^{N} Q_{\theta_1}(s_i, \pi_\phi(s_i)) \tag{5}$$

### A.3 GRU-GATE DESCRIPTION

$$\begin{aligned} r &= \sigma(W_r^{(l)} y + U_r^{(l)} x), \\ z &= \sigma(W_z^{(l)} y + U_z^{(l)} x - b_g^{(l)}), \\ \hat{h} &= \tanh(W_g^{(l)} y + U_g^{(l)}(r \odot x)), \\ g^{(l)}(x, y) &= (1 - z) \odot x + z \odot \hat{h} \end{aligned} \tag{6}$$

Where $x, y$ is input data flow, $(1 - z)$ represents the proportion of information received from the skip connection and $W_r^{(l)}, U_r^{(l)}, W_z^{(l)}, U_z^{(l)}, W_g^{(l)}, U_g^{(l)}$ are learnable parameters.