Transformer Adapters for Robot Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract: Large transformer-based architectures are capable of complex robot task planning and low-level control. In the natural language processing (NLP) community, fine-tuning large pretrained models (PTMs) such as GPT-3 and PaLM is the de-facto standard. With the scalability of transformer models and growing availability of large-scale multimodal robot data, we investigate pretraining large backbone models to capture useful behavioral priors that enable efficient few-shot transfer to downstream robot tasks. We explore the setting of modular reinforcement learning (RL) in which each downstream task is encapsulated by an independently learned module. With many downstream tasks, fine-tuning or training separate copies of these large PTMs become computationally and memory intensive. We propose to pretrain a large *transformer* backbone on task-agnostic data and learn small task-specific *adapters* using few-shot imitation learning to quickly adapt to downstream tasks. We evaluate on complex robot manipulation tasks in the Metaworld environment and demonstrate that adapter training is a parameter-efficient approach for modular RL.

Keywords: Parameter-efficient fine-tuning, Modular Reinforcement Learning, Few-shot imitation learning

1 Introduction

As large pretrained models (PTMs) are being adopted in robotics and eventually deployed onto physical robot systems, how can non-industry practitioners adapt these large PTMs to custom tasks beyond just relying on their zero-shot capabilities? We present a scalable approach, inspired by work in the NLP community, for a large pretrained generalist robot model to acquire diverse behavior and quickly adapt to new downstream tasks.

Large pretrained models such as PaLM [1], DALL-E [2], and GPT-3 [3] have demonstrated impressive capabilities across many domains ranging from natural language processing (NLP) to vision [2] and even robotics [4] [5] [6] [7]. Fine-tuning these large PTMs can incur significant training and storage costs. GPT-3, for example, has 175 billion parameters, cost $10 million and a few months of training, and takes up 350 GB of storage [3].

The prevalent paradigm for overcoming large model training costs in learning new NLP tasks is to train small neural modules called *adapters* [8] on a relatively small amounts of data specific to the task. Adapters are embedded inside Transformer blocks of large PTMs. This design facilitates weight sharing and accelerates learning of new NLP tasks by leveraging pretrained linguistic priors. We advocate for a similar framework whereby an autoregressive Transformer-based policy is pretrained on a large, diverse task-agnostic robot dataset. Small task-specific adapter modules are then learned for each new downstream task while sharing a common transformer backbone. We demonstrate on a challenging robot manipulation environment that a pretrained Transformer model can zero-shot perform unseen tasks and task-specific adapters can quickly adapt to new tasks with a few demonstrations using $< 2\%$ of the full model parameters.

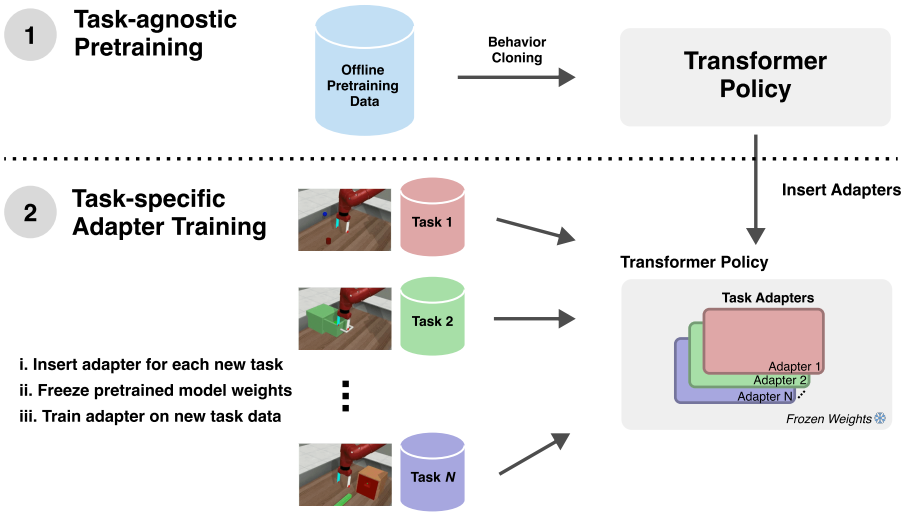


Figure 1: We pretrain a Transformer policy on a large corpus of offline task-agnostic data, which captures useful manipulation priors such as picking up objects. We can use this shared pretrained backbone to accelerate the learning of new tasks in both offline and online settings. We propose to train task-specific adapters, small neural modules that are injected in between each transformer layer to capture task relevant information without damaging the weights of the pretrained model.

2 Approach

We train an imitation learning policy for pretraining transformers using task-agnostic experience dataset. Thereafter, we learn adapters with frozen transformer backbone for downstream task adaptation using few-shot task demonstrations.

2.1 Preliminaries

Transformer Policy. Decision Transformers (DT) [9] demonstrate that self-attention based models can be applied to solve RL problems due to its efficiency and scalability in modeling long sequential data. Rather than learning explicit value functions, DT treats policy learning as a causal sequence modeling problem. DT uses a Transformer backbone, specifically GPT-2 [10], trained on a causal modeling objective of predicting the next action token given the context history. Trajectories are sequences of states s_t, actions a_t, and reward-to-go $\hat{r}_t$ tokens. Unlike DT, which learns a reward-conditioned policy, we pretrain our model on valid trajectories only $\tau = \{s_t, a_t\}_{t=1}^{T}$ (Figure 2), followed by goal-conditioned few-shot imitation learning.

Adapters introduce a small set of new parameters between the layers of a large pretrained model. During downstream training, the pretrained model weights are fixed while the adapter weights learn to encode task-specific representations. Adapters efficiently share a majority of the weights with the pretrained model, and this facilitates information sharing that improves downstream learning. In practice, adapters are very lightweight, typically using only 0.5-8% of the full backbone model parameters. Houlsby et al. [8] empirically show that a two-layer feed-forward bottleneck architecture works the best for the adapter design. Adapters are defined as:

$$A^l(\mathbf{x}) = \mathbf{x} + W^l_{up}(\text{GeLU}(W^l_{down}(\mathbf{x}))), \tag{1}$$

where x is the input hidden state, W^l_{up}, W^l_{down} are the weights for the feed-forward up and down-projection for layer l respectively. Adapters learn to encapsulate task-specific information that is non-destructive to the original model [8]. Adapters are modular by design.

2.2 Problem Formulation

We assume access to a task-agnostic dataset $D^{PT} = \{\tau_i\}_{i=1}^{N}$ for pretraining the backbone model. Each trajectory τ_i is collected by an agent interacting with a Markov Decision process (MDP) de-

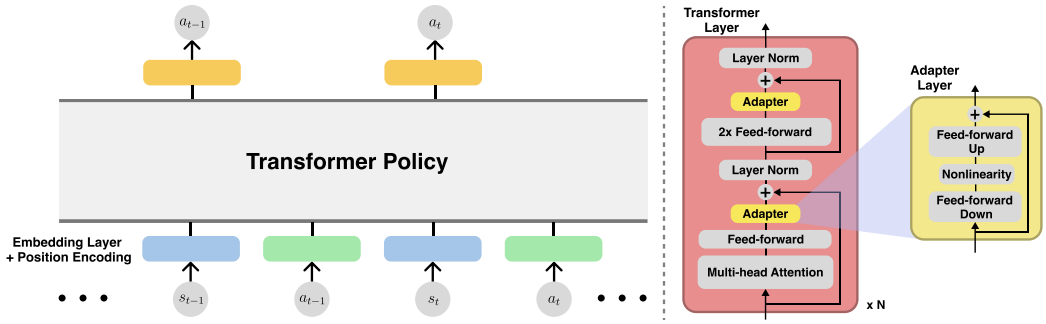


Figure 2: We embed state and action sequence inputs with modality-specific layers and added to learned positional timestep encodings. A GPT-2 [10] model autoregressively predicts actions corresponding to each state. Two adapter layers are added to every Transformer block.

fined by a tuple $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}\}$ of states, actions, rewards, and transition probability. Data can be collected from an expert planner, play interactions, or even a random policy. We focus on an imitation learning setting in which the MDP does not have an explicitly defined reward function, i.e $\mathcal{M} \setminus \mathcal{R}$. The objective is to learn an *imitation policy* $\pi(a|s)$ such that the agent behaves like the expert provided a set of expert demonstrations for a task. During downstream task adaptation, we assume that there is a dataset D^{test} containing a few demonstrations for each target task $\mathcal{T}_i \in \mathcal{T}^{test}$.

2.3 Transformer Adapters

Our approach has two stages: pretraining and downstream task learning. During pretraining, the model learns to capture useful behavioral priors from the task-agnostic data, such as how to manipulate different objects. By leveraging the capacity of large models, we capture diverse prior knowledge that can accelerate learning of downstream tasks. Following [9], our pretraining objective is to minimize the mean-squared error between the predicted action and ground truth actions:

$$L_{PT}(\theta) = \mathbb{E}_{\tau_i = \{s_t, a_t\}_{i,t=1}^{T} \sim \mathcal{D}^{PT}} \Big[-\log p_\theta(a_t | s_{t-H}, a_{t-H}, ..., a_{t-1}, s_t) \Big], \tag{2}$$

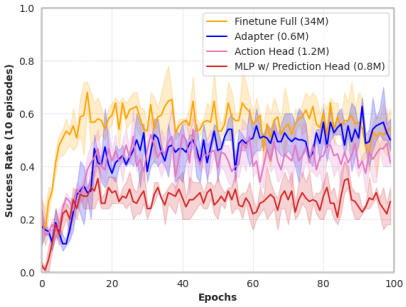
where H is the context length, θ are the policy parameters. During the fine-tuning stage, we use the pretrained model to bootstrap the learning of new downstream tasks via few-shot imitation learning. The agent is provided with a small set of expert demonstrations $\{\tau_i\}_{i=1}^{K}$ for each target task. We insert *adapter* modules into the layers of the pretrained transformer (see Figure 2), and only train the weights of the adapter with the pretraining objective using the expert demonstrations.

3 Experiments and Results

We aim to answer two main questions: 1) Does a transformer policy pretrained on large, diverse offline data provide a strong backbone for transfer to downstream unseen tasks? 2) Are adapters a parameter-efficient and scalable approach for few-shot imitation learning of new robot tasks?

Environment and Data We evaluate our method in the MetaWorld robot manipulation benchmark [11]. MetaWorld contains 50 different robot control and manipulation tasks with a Sawyer arm in the Mujoco environment. We evaluate on the Meta-Learning 45 (`ML45`) setting, where we pretrain our policy on 45 tasks and evaluate downstream adaptation to 5 held-out tasks. The task splits are hand-selected in [11] such that the training tasks are structurally similar to the test tasks. Object and goal positions are randomized at the start of every episode. The action space $A \in \mathbb{R}^4$, is the $\Delta(xyz)$ of the end-effector, and a continuous scalar value for gripper torque. The state $S \in \mathbb{R}^{39}$, contain the 3D positions of the end-effector, first object, second object and the goal.

The goal position is masked during pretraining and provided to the model during downstream adaptation. We use a scripted policy to collect 10 trajectories for each of the 50 tasks. We pretrain on 450 trajectories from the training split and few-shot imitation learned on 10 demonstrations per test task. We report averaged task success across all 5 test tasks over 10 evaluation rollouts and 3 seeds.



Method	Box Close	Door Lock	Door Unlock	Bin Picking	Hand Insert	Average
FT	**.40**±.08	.40±.14	**1.0**±.00	**.50**±.24	**.43**±.09	**.55**±.07
Adapter	.23±.05	**.43**±.09	.97±.05	.43±.17	**.43**±.24	.50±.06
AH	.20±.14	.30±.16	.90±.08	**.50**±.08	.17±.05	.41±.02
MLP	.00±.00	.27±.05	.67±.09	.20±.08	.20±.14	.27±.06

Figure 3: (*Left:*) Evaluation success rates over 100 epochs. Shaded areas represent standard deviation across three seeds. (*Right:*) Final success rates for each individual evaluation task and the average across all. Highest in **bold** and second highest performance in blue. Methods utilizing pretrained Transformer backbones exhibit positive zero-shot performance on unseen tasks, unlike pretrained MLP. Training separate adapters per task matches the performance of fine-tuning the full model using less than 2% of the model parameters. The pretrained backbone zero-shot solves some of the downstream tasks with a positive success rate compared to the MLP backbone.

Baselines We evaluate Transformer `Adapters` against several baselines:

- **MLP w/ prediction head (`MLP`).** Fine-tune a copy of the action prediction head for each downstream task. This comparison measures the advantage of using self-attention based Transformer models for capturing strong behavioral priors.
- **Transformer w/ prediction head (`ActHead`).** Fine-tune a copy of the action prediction head for each new task. This measures the benefit of training intermediate representations that are dynamically stitched into a pretrained model versus training the last few layers.
- **Fine-tuning entire Transformer (`FT Full`).** Fine-tune the weights of the full model. This is an upper bound for downstream performance if we do not limit the model capacity.

Results and Discussion Learning task-specific adapter modules on a pretrained transformer backbone (`Adapter`) can almost match the performance of fine-tuning the entire model (`FT Full`) while using less than 2% of the pretrained model parameters (Figure 3). Compared to the full transformer backbone with 200 Mb of storage, each individual task adapter requires less than 2 Mb of storage. This reduced storage overhead becomes more substantial when working with large foundation models and deployment on physical robot systems. `Adapter` converges slower than `FT Full` as adapter weights are initialized to zero and trained from scratch. The `MLP` model, which has slightly more trainable weights than the `Adapter` model, fails to converge to good average task performance.

Methods using a frozen pretrained transformer backbone (`Adapter`, `ActHead`, `FT Full`) exhibit positive zero-shot performance on several of the evaluation tasks, indicating that the pretrained model may have captured some useful behavioral priors that can generalize without additional fine-tuning. In comparison, the `MLP` backbone pretrained on the same data is unable to generalize zero-shot to unseen tasks. By pretraining on even more diverse task-agnostic data, we hypothesize that a transformer-based backbone models can provide even stronger zero-shot performance gains. Additionally, zero-shot performance can accelerate online learning because the learned priors enable more guided exploration for the agent to quickly discover reward states.

4 Conclusions and Future Work

We present a two-stage scalable framework for robot learning: task-agnostic transformer policy pretraining followed by parameter-efficient few-shot downstream task adaption using adapters. Task-specific adapters outperform task-specific action heads with the same transformer backbone, while being comparable to fine-tuning the full model. In the future, we hope to extend our setup to other multi-task environments and conduct experiments with visual perception-based states. There is a great breadth of work in the NLP community on parameter-efficient fine-tuning that can be adapted for robot learning. We can explore orthogonal approaches such as LoRA [12] and Compacter [13] that are even more optimized for training efficiency and reducing storage cost. Moreover, we can extend our method to handle long-horizon, compositional tasks using AdapterFusion [14].

References

[1] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[2] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[4] A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *6th Annual Conference on Robot Learning*, 2022.

[5] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg. Progprompt: Generating situated robot task plans using large language models. *arXiv preprint arXiv:2209.11302*, 2022.

[6] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. *arXiv preprint arXiv:2209.05451*, 2022.

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

[8] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

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

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

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

[12] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[13] R. Karimi Mahabadi, J. Henderson, and S. Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. *Advances in Neural Information Processing Systems*, 34:1022–1035, 2021.

[14] J. Pfeiffer, A. Kamath, A. Rücklé, K. Cho, and I. Gurevych. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247*, 2020.

[15] S. Smith, M. Patwary, B. Norick, P. LeGresley, S. Rajbhandari, J. Casper, Z. Liu, S. Prabhumoye, G. Zerveas, V. Korthikanti, et al. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*, 2022.

[16] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn. Robonet: Large-scale multi-robot learning. *arXiv preprint arXiv:1910.11215*, 2019.

[17] F. Ebert, Y. Yang, K. Schmeckpeper, B. Bucher, G. Georgakis, K. Daniilidis, C. Finn, and S. Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. *arXiv preprint arXiv:2109.13396*, 2021.

[18] A. Rücklé, G. Geigle, M. Glockner, T. Beck, J. Pfeiffer, N. Reimers, and I. Gurevych. Adapterdrop: On the efficiency of adapters in transformers. *arXiv preprint arXiv:2010.11918*, 2020.

A. Related Works

Transformers for Robot Learning. Recently, there has been a surge of research in the robotics community applying large Transformer models to a variety of complex robot learning problems. Decision Transformers (DT) [9] showed that the causal self-attention mechanism of Transformer models can be used to perform credit assignment in sequential decision making. Perceiver-Actor [6] uses a large multi-modal Transformer to learn a multi-task policy for solving robotic manipulation task. SayCan [4] show that large pretrained language models (PTLMs) encapsulate a wealth of commonsense knowledge that can be used for efficient high-level task planning in the real-world. VIMA [7] shows that a Transformer-based model trained on large amounts of expert demonstration can zero-shot generalize to new tasks using multimodal prompts.

Scaling model and data. In the natural language processing (NLP) community, there is a recent trend towards increasingly larger Transformer-style models trained on large, diverse text corpuses. ELMo, one of the earliest works on contextualized word representations, published in 2018 had roughly 94 million parameters. Within the span of three years, OpenAI released GPT-3 [3] (175 billion parameters) and Microsoft and NVIDIA introduced Megatron-Turing [15] (530 billion parameter model). GPT-4 is expected to have about 100 trillion parameters. GPT-3 is reported to be trained on 500 billion tokens of internet text. In contrast, the quantity of robot data available is nowhere near that magnitude hence the slow adoption of the pretraining-finetuning paradigm. There have been several recent attempts to curate and publish larger, more diverse robot datasets to address this gap (e.g. RoboNet [16], Bridge Data [17], VIMA [7], etc).

Parameter-efficient Fine-tuning in NLP. As Transformer models become increasingly large and more resource intensive, it is infeasible to fine-tune the entire model for each new downstream task. Several alternatives have been proposed that update only a small number of extra parameters while keeping the backbone model parameters frozen. For example, Adapter [8] tuning inserts a small set of trainable weights between each layer of the pretrained Transformer. [18] empirically show that Adapters are 60% faster than full-model tuning in terms of computational efficient because of the decrease in overhead in gradient computation and are significantly more storage efficient. More recently, LoRA [12] and Compacter [13] learns low-rank matrices to approximate parameter updates.

B. MetaWorld Downstream Tasks

Figure 4: For each task in the Meta-Learning 45 (ML45) benchmark in the MetaWorld environment, we use a scripted policy to collect a diverse dataset of robot behavior. We use the demonstrations from the 45 training tasks to pretrain our Transformer policy. We then train separate adapters for each test task (shown in the Figure) through imitation learning using the respective demonstrations. Object and goal positions are randomized at the start of every episode during data collection and inference time.

C. Hyperparameters

We show the hyperparameters of Transformer Adapter.

Hyperparameters	Value
context length	50
learning rate	1e-4
learning rate decay	1e-4
number of layers	4
number of attention heads	4
dropout	0.1
embedding dimension	768
batch size	1024
max episode length	500
number of epochs	100