Transfer learning in offline reinforcement learning settings

Bhavesh Gawri
bgawri@cs.stonybrook.edu

Abstract

Transfer learning in reinforcement learning (RL) is an active and evolving field, with recent works exploring techniques such as fine-tuning pre-trained policies, model distillation, and domain adaptation to effectively transfer knowledge from source tasks to improve learning in target tasks. This work explores the effectiveness of fine-tuning pre-trained decision transformer (DT) based policies for knowledge transfer and compares its performance with policies trained from scratch in target environments.

1 Introduction

Offline RL (Levine et al., 2020) is a paradigm that allows agents to learn from pre-collected datasets, avoiding the need for costly online interactions. It has gained attention for its potential to tackle real-world RL problems. However, applying offline RL to complex tasks remains challenging due to distributional shift and the curse of dimensionality.

Transfer learning (Taylor and Stone, 2009; Pan and Yang, 2010; Zhuang et al., 2020), on the other hand, aims to leverage knowledge acquired from related tasks to improve learning for a target task. By transferring learned policies, transfer learning enables agents to generalize from past experiences and accelerate learning in new domains. In RL, transfer learning can be achieved through various methods, such as fine-tuning existing policies (Lee et al., 2022; Xu et al., 2023), policy distillation (Rusu et al., 2015; Tseng et al., 2022), and domain adaptation (Xing et al., 2021; Eysenbach et al., 2020). Fine-tuning existing policies involves initializing the policy with parameters from a source task and then continuing the training process using data from the target task. In policy distillation, a student policy aims to mimic the behavior of the expert teacher policy, thus transferring its knowledge and expertise whereas, in domain adaptation, a policy adapts to the differences between the source and target domains.

This work explores fine-tuning of pre-trained policies for knowledge transfer. Specifically, pre-trained decision transformer based policies. Decision transformer (Chen et al., 2021) is a variant of transformer models (Vaswani et al., 2017) that have demonstrated exceptional capabilities in capturing complex dependencies in sequential data, making them well-suited for tasks involving natural language processing (Vaswani et al., 2017) and computer vision (Dosovitskiy et al., 2020). Extending their application to RL settings to process state-action-reward sequences works effectively in modeling dynamics in online (Zheng et al., 2022) as well as offline (Meng et al., 2021; Furuta et al., 2021) RL to facilitate knowledge transfer (Lee et al., 2022; Xu et al., 2023).

2 Contributions

In this work, my contributions are four-fold:

- Implementation of a DT module with a customizable architecture for training and fine-tuning a given model on environments of varying dimensionality.
- Implementation of a custom dataloader to normalize states and actions, calculate returns-to-go and attention masks, and return randomized batches of data for variable DT context lengths.
- Implementation of an evaluator class to test and evaluate the pre-trained and fine-tuned policies as well as calculate training statistics such as rewards throughout the training process.
- Evaluation and comparison between 36 fine-tuned and 3 trained-from-scratch policies in 3 different target environments.
- (Proposed - novel but not implemented) Parameter efficient fine-tuning for DTs using Adapter(Houlsby et al., 2019) modules. As this work got published very recently (Xu et al., 2023), I deferred my implementation for parameter efficient fine-tuning to future works and instead went ahead with full-model fine-tuning to focus on maximizing rewards while not conditioning it on parameter efficiency.

Code available at: `https://github.com/bhaveshgawri/decision-transformer-transfer-learning`

3 Implementation Details

This work experiments with training the decision transformer based policies in a source environment, fine-tuning the pre-trained policy in a target environment, and comparing the performance of the fine-tuned policy with a policy trained from scratch directly in the target environment. The experiments with pre-training and fine-tuning the policies are conducted in 9 different combinations of MuJoCo (OpenAI, 2023) environments composed of Half-Cheetah, Hopper, and Walker2d. For instance, one of the nine combinations could be a policy pre-trained in Half-Cheetah environment and later fine-tuned for Walker2d environment.

The decision transformer used in pre-training and fine-tuning consists of linear embedding layers for input states, actions, rewards-to-go (rtg), and time-steps. The inputs states and actions to DT consists of positional and action values of different body parts as defined respectively by the observation and action space of the environment. Rewards-to-go are cumulative returns at a state and are calculated as:

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1} R_T$$

The results of the embeddings layer is passed to a three layer encoder whose output connects to the final prediction layer for output states, actions, and rewards. As a result, for each of the nine combinations of the environments, experiments are conducted with 4 fine-tuning strategies. For the first strategy, all the three layers of the encoder are kept frozen during the fine-tuning phase. In the remaining three strategies, one encoder layer is sequentially unfrozen starting from the last encoder layer. However, for the training phase, the entire DT is set as trainable. In addition, during both the training and fine-tuning phases, the state and reward values are normalized as suggested by (Chen et al., 2021), i.e. the state values are mean centered and rewards are scaled down by a scaling factor for a smoother training.

In order to make fair comparisons between the performance of fine-tuned models and models trained from scratch, the architecture of DT is set to a fixed number of layers with a fixed dimensionality across all experiments. The DT architecture is only changed in the case when state and action dimensions of the pre-training environment differ from the dimensions of the target environment. In this scenario, only the dimensions of the embeddings layer and final prediction layer are changed whereas the encoder dimensions stay the same. In addition, for training and fine-tuning across the experiments all of the training hyper-parameters including learning rate, DT context length, and training epochs are kept constant.

4 Results and Evaluation

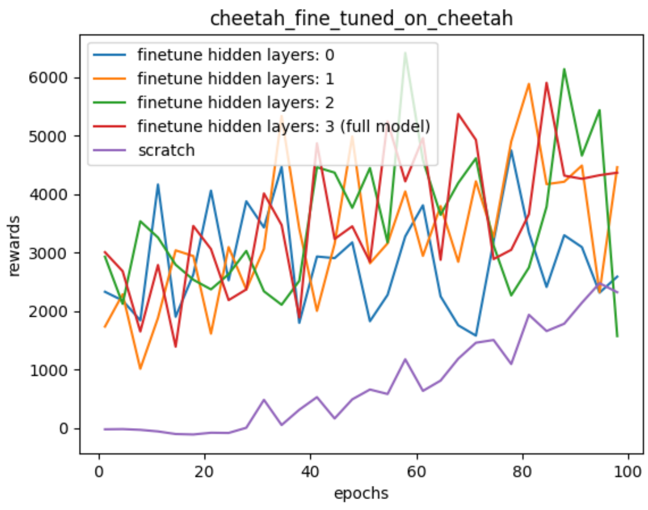
The main purpose of this evaluation is to compare the performance of the fine-tuned models with the models trained from scratch in the target environment. The evaluation metric chosen for this comparison is average rewards after a given number of epochs. During the training and fine-tuning phase, rewards are averaged over five episodes after every five epochs of training based on the hyper-parameters, i.e. after every five passes through the entire training dataset (of size 1000 expert trajectories), the current model is evaluated to calculate rewards over five episodes starting with randomly initialized states. These rewards are then averaged and tracked as a part of evaluation. The graphs in figure 1, 2, and 3 display these rewards against 100 epochs of training and fine-tuning.

Each of the figures below contain three sub-figures that represent the nine combinations of MuJoCo environments: Half-Cheetah, Hopper, and Walker2d. Figure 1 contains results for source environment Half-Cheetah on which the model is pre-trained and the sub-figures 1.a, 1.b, and 1.c represent the target or fine-tuning environment which include Half-Cheetah for (1.a), Hopper for (1.b), and Walker2d for (1.c). The ordering of fine-tuning environments in sub-figures stays the same across all the three figures 1, 2, and 3 but the source environment changes, i.e. Half-Cheetah for figure 1, Hopper for figure 2, and Walker2d for figure 3. Further, for each sub-figure, there are four fine-tuning strategies defined in the legend and explained in section 3 and a baseline metric for the model trained from scratch in the target environment.

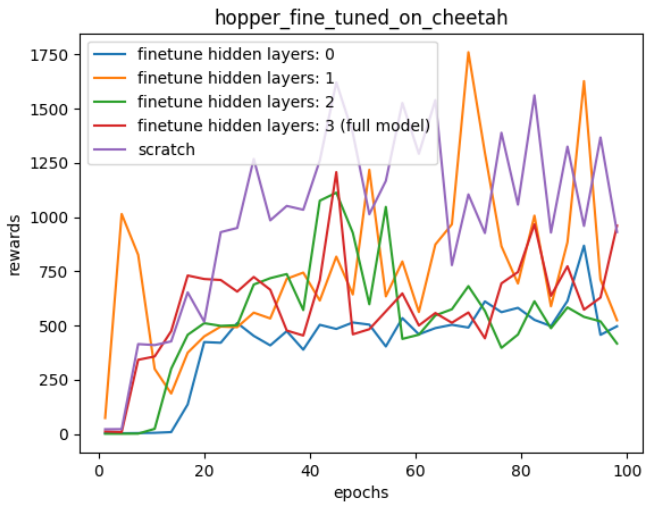
From the figures 1, 2, and 3, one can observe that fine-tuning helps improve training performance for a given number of epochs in some scenarios whereas for certain cases it doesn't help much. For instance, when the source environment is Walker2d as in figure 3, it can be observed that both the target environments Half-Cheetah, and Hopper performed better than the baseline trained-from-scratch models. However, the opposite is true with Hopper as source environment in figure 2 where baseline model performed better than all the fine-tuned models for both the target environments Half-Cheetah, and Walker2d. Subsequently, for Half-Cheetah in figure 1 as the source or pre-training environment, the results are inconclusive as fine-tuned model performance is very close to the models trained from scratch in the target environment. From this evaluation, one can conclude that for a limited amount of training data and compute, one cannot reliably depend on fine-tuning for transfer learning between any combination of source and target environments. If the source and target are carefully chosen, one can start to observe slight improvements in the returns for fine-tuning over training-from-scratch as the process progresses.

5 Conclusion and Future works

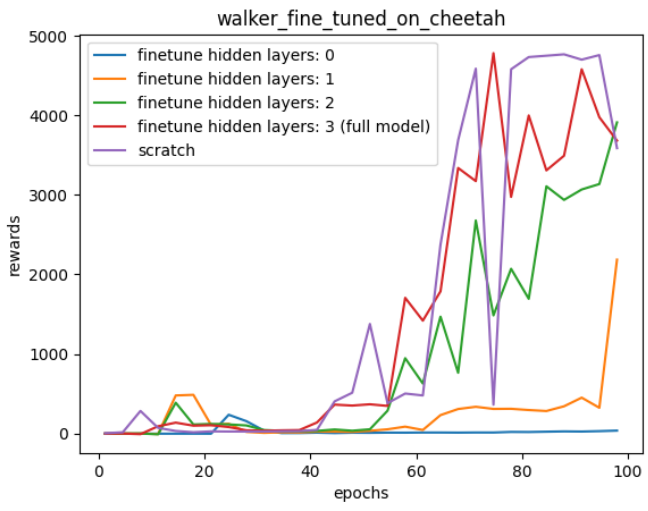
In this project, I experimented with various fine-tuned policies in three environments to investigate whether fine-tuning would impart an edge over the policies that are trained directly from scratch. The results indicate that this is possible for a certain combinations of source



(a)

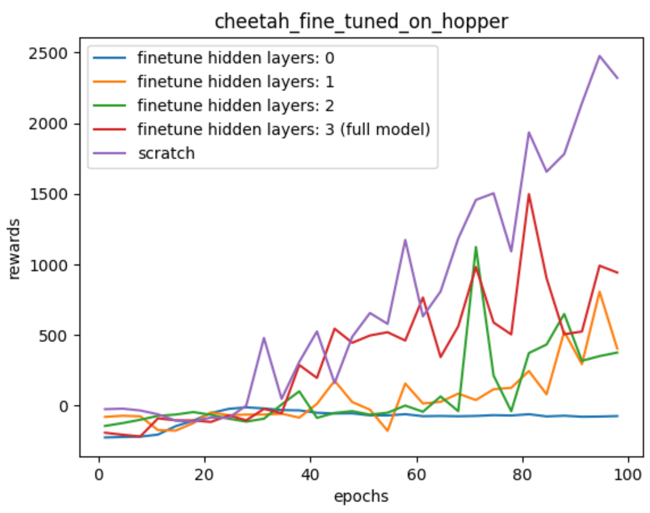


(b)

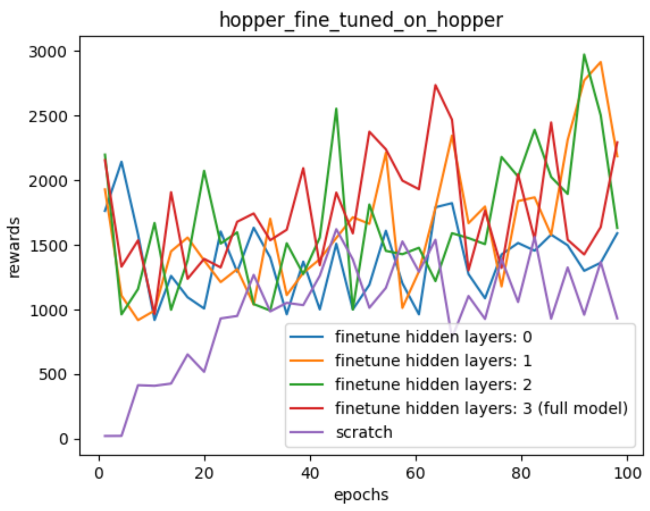


(c)

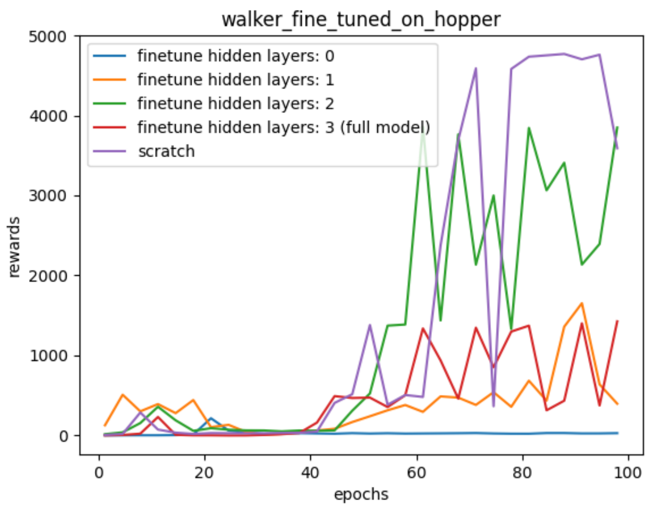
Figure 1: Target env: (a) Half-Cheetah, (b) Hopper, and (c) Walker2d all fine-tuned on source env: Half-Cheetah



(a)

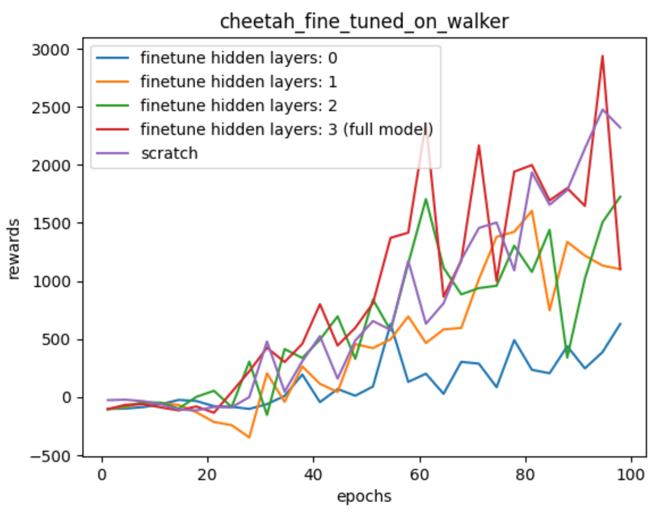


(b)

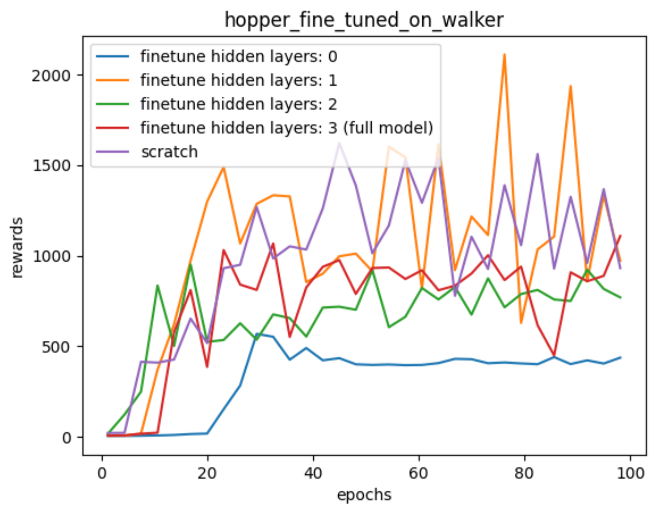


(c)

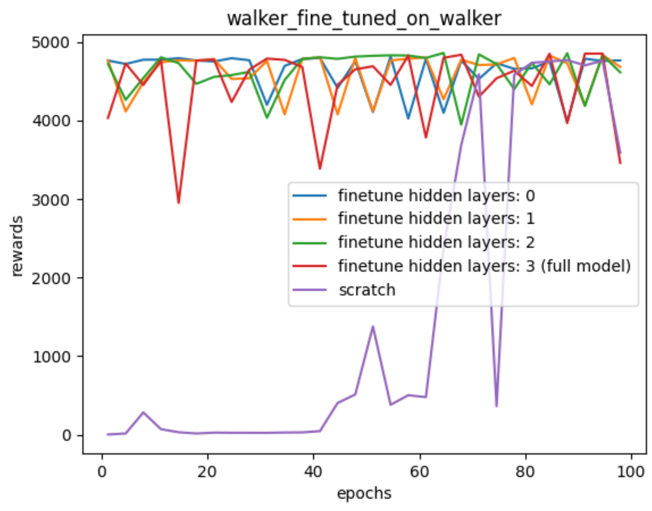
Figure 2: Target env: (a) Half-Cheetah, (b) Hopper, and (c) Walker2d all fine-tuned on source env: Hopper



(a)



(b)



(c)

Figure 3: Target env: (a) Half-Cheetah, (b) Hopper, and (c) Walker2d all fine-tuned on source env: Walker2d

and target environments. However, in the current experiments, the pre-trained policy was only trained on one source environment. In future, I would like to experiment with the scenarios where the pre-trained policy is trained on multiple environments so as to learn from various tasks. In such scenarios, fine-tuning over a new environment should provide more robust returns. Additionally, I would also like to experiment with parameter-efficient ways for fine-tuning to investigate the minimum number of parameters that one might need to achieve the required returns.

6 References

References

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.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Eysenbach, B., Asawa, S., Chaudhari, S., Levine, S., and Salakhutdinov, R. (2020). Off-dynamics reinforcement learning: Training for transfer with domain classifiers. *arXiv preprint arXiv:2006.13916*.

Furuta, H., Matsuo, Y., and Gu, S. S. (2021). Generalized decision transformer for offline hindsight information matching. *arXiv preprint arXiv:2111.10364*.

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

Lee, K.-H., Nachum, O., Yang, M. S., Lee, L., Freeman, D., Guadarrama, S., Fischer, I., Xu, W., Jang, E., Michalewski, H., et al. (2022). Multi-game decision transformers. *Advances in Neural Information Processing Systems*, 35:27921–27936.

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*.

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 tackles all smac tasks. *arXiv e-prints*, pages arXiv–2112.

OpenAI (2023). Mujoco. Accessed: Apr 1, 2023.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Rusu, A. A., Colmenarejo, S. G., Gulcehre, C., Desjardins, G., Kirkpatrick, J., Pascanu, R., Mnih, V., Kavukcuoglu, K., and Hadsell, R. (2015). Policy distillation. *arXiv preprint arXiv:1511.06295*.

Taylor, M. E. and Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7).

Tseng, W.-C., Wang, T.-H. J., Lin, Y.-C., and Isola, P. (2022). Offline multi-agent reinforcement learning with knowledge distillation. *Advances in Neural Information Processing Systems*, 35:226–237.

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

Xing, J., Nagata, T., Chen, K., Zou, X., Neftci, E., and Krichmar, J. L. (2021). Domain adaptation in reinforcement learning via latent unified state representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10452–10459.

Xu, M., Lu, Y., Shen, Y., Zhang, S., Zhao, D., and Gan, C. (2023). Hyper-decision transformer for efficient online policy adaptation. *arXiv preprint arXiv:2304.08487*.

Zheng, Q., Zhang, A., and Grover, A. (2022). Online decision transformer. In *International Conference on Machine Learning*, pages 27042–27059. PMLR.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. (2020). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76.