Project Report Template

Kefan Song

September 2023

1 Introduction

A key sign of intelligence is the capacity to solve many tasks with relatively good performance. In the field of psychometric, intelligence is measured by the g-factor, which is a single factor that strongly positively correlates to the individual's performance of any arbitrary task presented to the individual. Reinforcement learning is motivated by some as a pathway to Artificial General Intelligence(AGI), because its machine-learning nature allows it to solve tasks from an arbitrary domain. However, this is done through collecting an enormous amount of data from a single task, for example in the case of AlphaGo, essentially "over-fitting" to the one task. When trained on a diversified set of tasks together, current Reinforcement Learning such as PPO suffers in performance. Some techniques are proposed to allow a single policy to train on a set of tasks but still have poor generalization to new tasks because the tasks are learned separately without exploiting the shared representations among tasks.

Observing the human's capability to quickly master a new skill with very few examples, Meta-Learning is proposed to train a model over a distribution of tasks that can quickly adapt to new tasks. Existing Meta-Reinforcement Learning methods are evaluated on tasks that are almost identical to each other, for example, from the same robot in the same environment only with varying running speeds. It is an open problem for meta-learning algorithms to work on a qualitatively different task from a different distribution, such as the ML-45 task from the Meta-World benchmark.

In addition, meta-learning methods such as MAML use training tasks only to pick the best parameters, discarding the representations learned in training tasks without leveraging the transfer learning possible from neural networks between training tasks and test tasks.

Whereas Transfer Learning aims to make use of past data when faced with a new task, which is closer to the idea of few-shot learning. With the success of pre-training large language models, we would like to follow the same idea of pre-training in reinforcement learning.

In this paper, we share the Meta-Learning problem's same objective, but instead of designing our algorithm explicitly as a Meta-Learning algorithm, we would like to pre-train a policy over a set of diversified tasks that shows emergent

meta-learning capabilities. We introduced an expert path planner as a retrieval machanism of working memory to enhance RL performance.

2 Related Work

Clearly describe existing methods most relevant to your work. Discuss the methods in the reading material that could be used for this + the advantages and disadvantages of those methods.

Multi-Task Reinforcement Learning

Gradient Surgery is proposed to eliminate the gradient conflict during training Multi-task RL. This only makes the learning of individual tasks in a single neural network not interfere with each other. This is essentially allowing a single neural network to learn individual tasks, without exploiting generalizing between similar tasks.

Tasks Grouping is proposed that achieves similar performance as gradient surgery, but the similarity between groups is separately learned. Instead of emerging from end-to-end training RL algorithms, it's unclear whether the success of training a single agent by this technique is a result of the algorithm's general problem-solving capacity.

Meta Learning Meta Learning is a machine learning principle to learn inductive bias from a distribution of tasks to produce a data-efficient algorithm.

RL^2 assumes that the policy has access to a set of MDPs from the same distribution, and a faster convergence on a similar task, not concerned with mastering all of them.

Memory-based Meta Reinforcement Learning

Duan et al. (2016) proposed an architecture to use RNN for feature vectors of the states.

Ritter et al. (2021) first collects trajectories in a buffer called episodic memory, and uses the transformer architecture over RNN to compute the feature vectors representing state s as input for the RL policy.

3 Methods

Describe the methods you have used in your work clearly and in detail. It should be so detailed that anybody without knowledge of those methods can understand it. You should not copy and paste any text from the books or papers. The description must be in your own words and with your illustration. Use figures and diagrams to illustrate.

Note: This part is the general description of the methods and should not be mixed up with your formal design.

In this paper, we would like to introduce the working memory mechanism to reinforcement learning aiming to improve its few-shot transfer learning performance on tasks from novel domains.

We emphasize a distinction between short-term memory and working memory. Short-term memory refers to the storage of states or observations from the environment with a capacity of a limited number of items, whereas working memory also refers to storing the retrieved task-relevant knowledge from long-term memory.

Therefore, for our design to take the role of working memory, it has to meet the following criteria:

1. It can leverage the past experiences collected from multiple tasks.
2. Its output can be used by the RL agent to improve decision-making.
3. The number of items from its output is naturally limited.

Based on these criteria, we proposed the method with the following components:

1. An expert path planner that predicts optimal future states given a sequence of states experienced by RL.

The expert path planner is ideally trained on experiences from tasks previously collected. Unlike trajectory transformer from Janner et al. (2021), the proposed expert path planner is only based on sequences of states and does not involve action or reward during training and prediction. This way, the first criterion is met, since previous experiences of tasks which is analogous to knowledge stored in long-term memory are efficiently utilized.

The actions and rewards are deliberately discarded during the design of the expert path planner because only involving the sequence of states allows the planner to have the potential to be pre-trained on internet video data, where the action and reward are unknown because we do not have direct control to the particular kind of robot in the video. However, the sequence of states as characterized by frames of the video is naturally directly accessible.

Ideally, with enough pre-training on related tasks, the output of this expert path planner will not be any randomly sampled next state, but the next state that would be caused by a near-optimal policy taking an action in the environment.

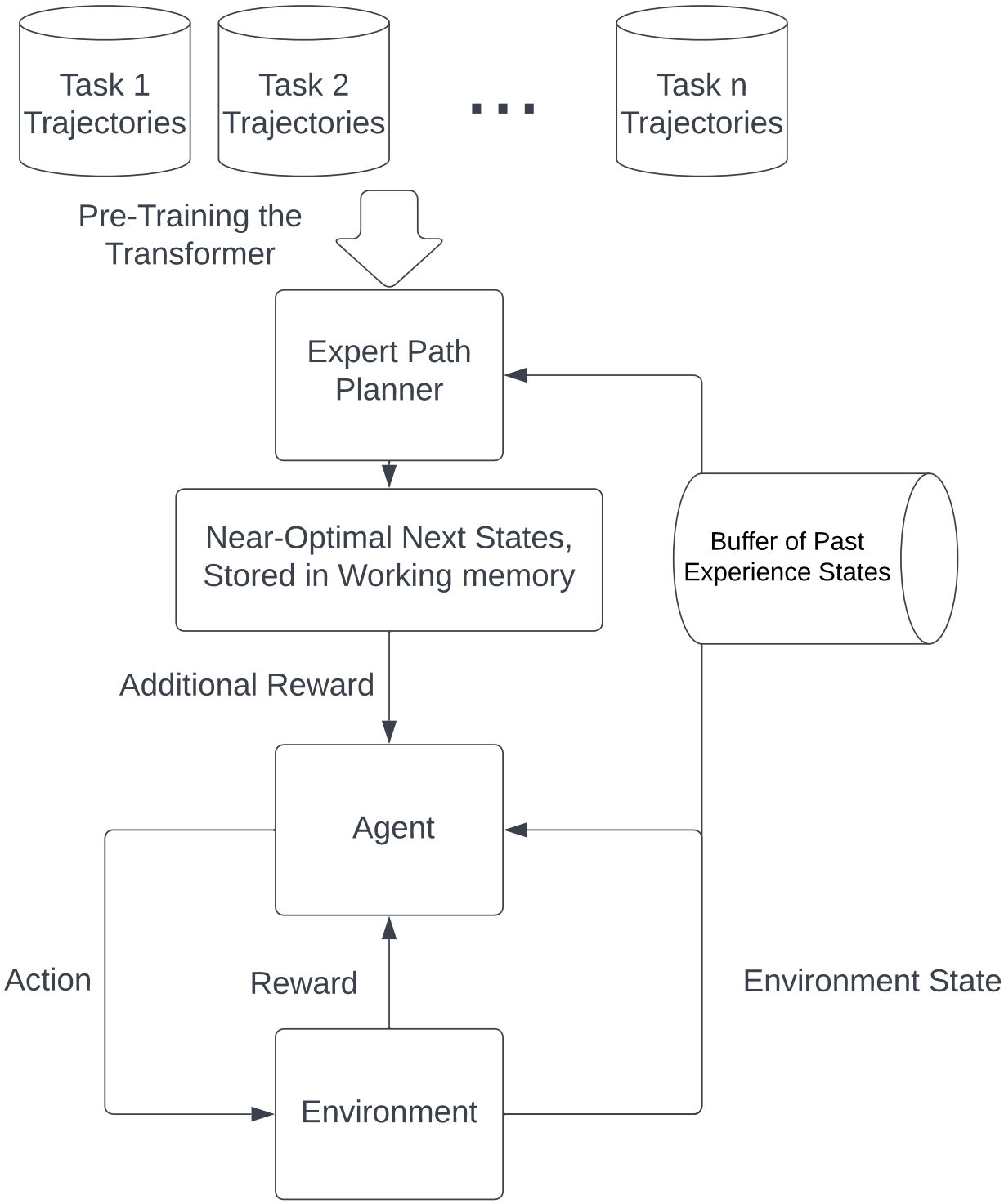
For example, for tasks such as opening the sliding window in the Meta-World Mujoco environment, we can utilize the experiences collected by other agents on other tasks, by pre-training the expert path planner on similar tasks such as opening the drawer and opening the door from the same Meta-World Mujoco environment, or even more ideally, from internet videos of people opening real-world sliding windows. If the model generalizes well, when tested on the sliding window task, the expert planner may predict the path to reach the window and the path to slide the window to the correct position.

To satisfy the second criterion, we treat these near-optimal next states predicted by the expert path planner as subgoals of the RL to enhance its decision-making. Specifically, we calculate the distance between the near-optimal next state and the actual next state transitioned by our current RL policy and use it to design the reward for RL.

Finally, the number of next states predicted is naturally limited. Keeping a limited window of near future next states may help RL to make better decisions.

Since these next states are treated as subgoals, if we have too many of them, then it will create no meaningful direction for the RL.

4 Architectural Design


Task 1 Trajectories
Task 2 Trajectories
. . .
Task n Trajectories
Pre-Training the Transformer
Expert Path Planner
Near-Optimal Next States, Stored in Working memory
Buffer of Past Experience States
Additional Reward
Agent
Action
Reward
Environment State
Environment

5 Formal Design

A finite-horizon MDP is defined as a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, H, P, r, \mu)$, where $\mathcal{S}$ is the state space, H number of steps in each episode, $r : \mathcal{S} \times \mathcal{A} \leftarrow [0, 1]$ is the reward function. We assume the reward function r is known.

The general methods described in the previous section must be adapted using your notations. Explain what adjustments you have made to the original method and why.

Definition 1: We aim to train an expert path planner that learns a state transition probability given an averaged unknown expert behavior policy as an implicit parameter $\pi_b(a|s)$ defined as $P(s'|s; \pi_b(\cdot|s))$

The more common notation of state transition probability involves action: $P(s'|s, a)$, by directly learning from the sequence of states, we are implicitly taking an expectation over action with respect to the agents' expert behavior policy $\pi_b(\cdot|s))$.

Definition 2: Let $\text{dist}(A, B)$ be any distance measurement of two matrices $A, B \in \mathcal{R}^{m,n}$, and let $\bar{s}_t$ be the expected next state predicted by the expert path planner at timestep $t-1$: $\bar{s}_t = \sum_{s_t} P(s_t|s_{t-1}; \pi_b(\cdot|s_{t-1}))s_t$, the reward is calculated by reward $\text{reward} = r(s_t, a_t) + \text{dist}(s_t, \bar{s}_t)$

Since the MDP could be stochastic and we are eventually calculating a distance between a subgoal state and a current state, we calculate an expected next state from the expert path planner's predictions.

6 Implementation

Describe the steps you have taken to implement the methods. What packages have you used, how data from other components is used, what format the input and output data is in, etc..

For the RL policies, we use the CleanRL implementation Huang et al. (2021). Meta-World is used for the task environment Yu et al. (2019). To identify multiple tasks, a one-hot encoder was concatenated to each state matrix for multi-task experiments, which might not be necessary with a good expert path planner.

To verify if such expert path planner will enhance the RL policy, we would like to ensure the expert path planner is able to predict reasonably good next states. Therefore, we trained the expert path planner using previously collected experience from an expert policy from the same task, and leave predicting the optimal path using other task experiences to the later development of this project.

10, and 400 demonstrations were each sampled to compute the expected next state $\bar{s}_t$.

7 Evaluation

Describe your experimental setup and present your steps for evaluating each method and the overall component. Describe the input data and output. Provide samples of input and output.

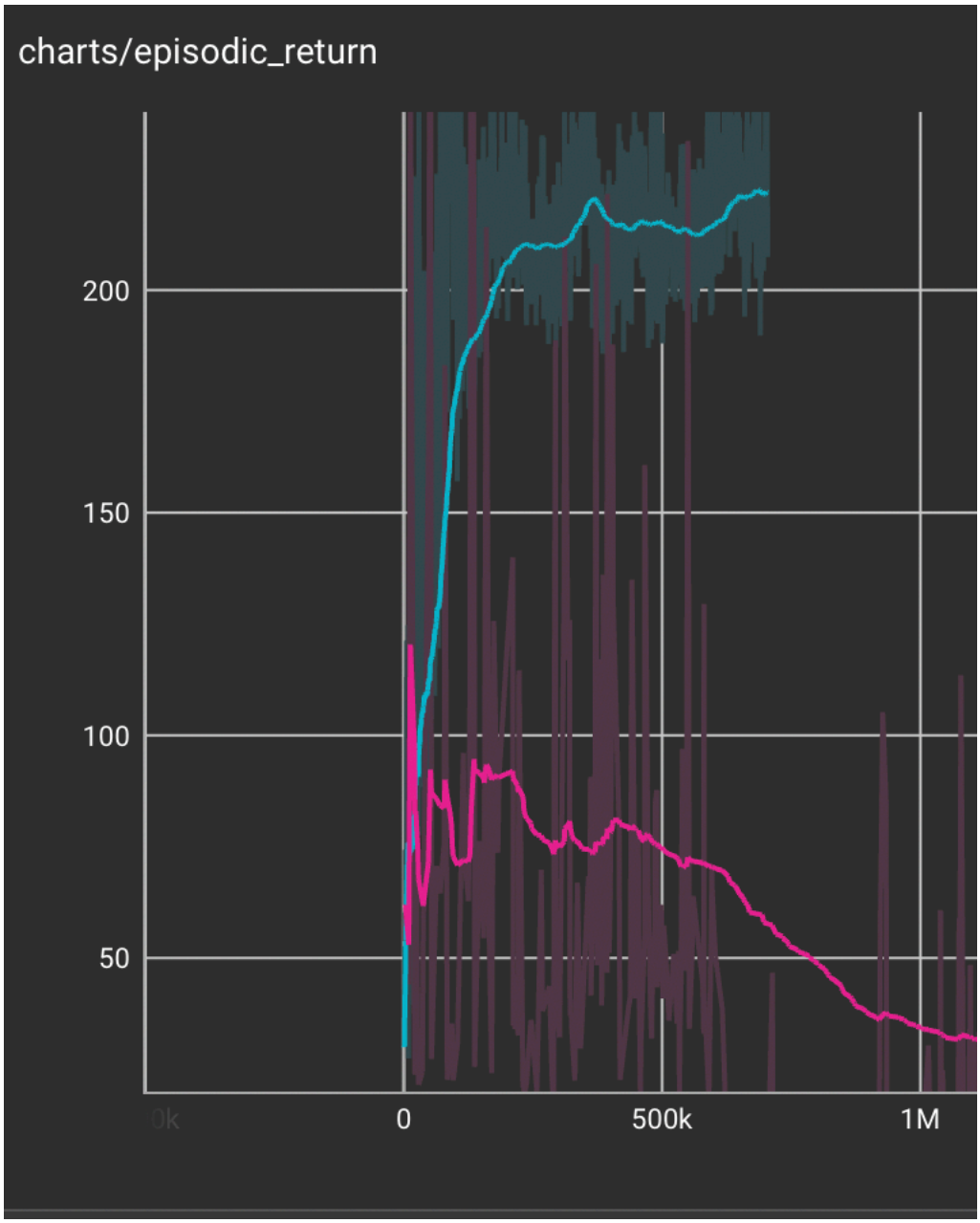
The experiment is evaluated by the speed of performance improvement measured by the expected return over timesteps.

Two distance measurement, cosine similarity and mean squared error was implemented by first flattening the matrix as a vector.

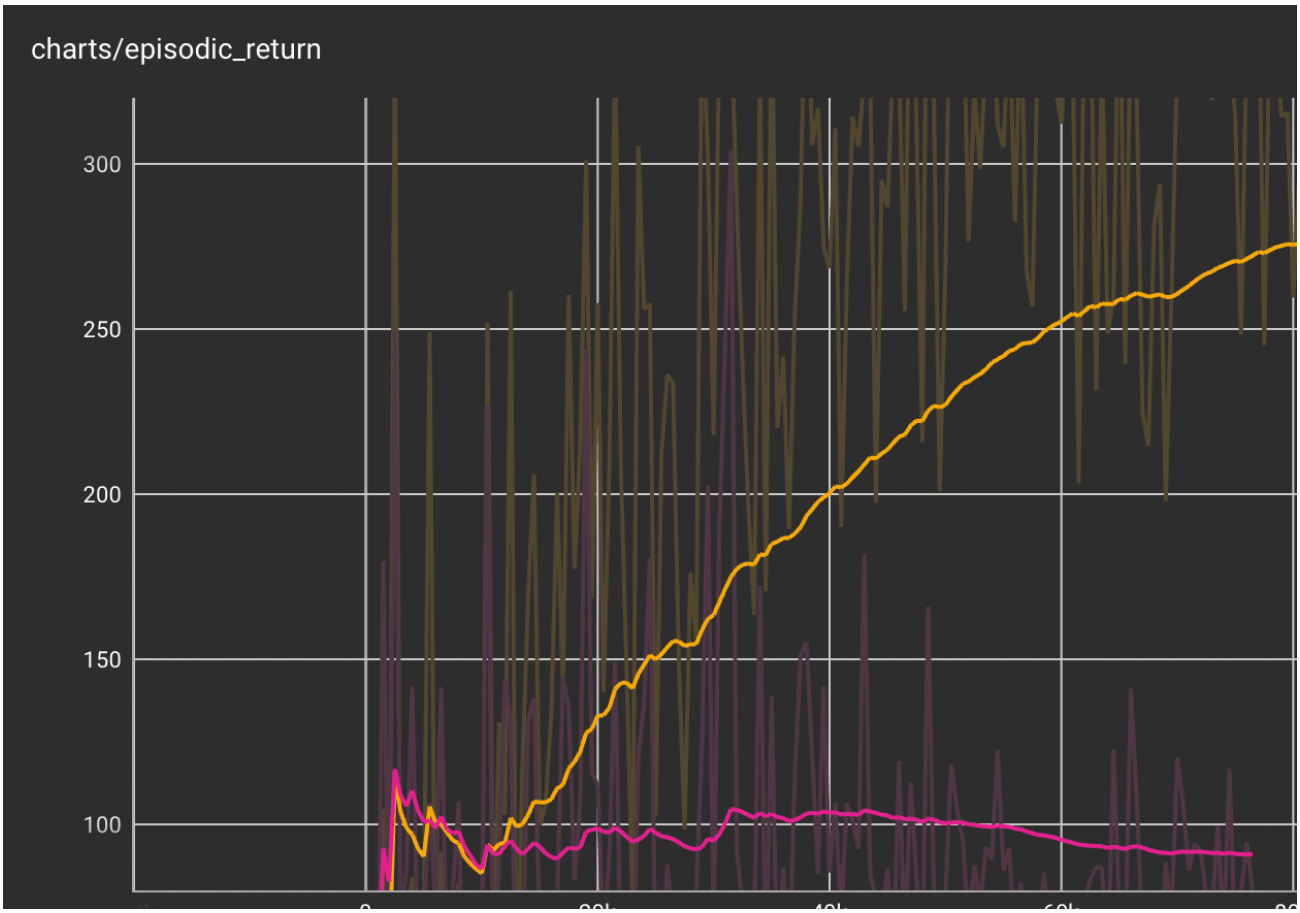
8 Results

Present the results of the evaluation. Use plots and figures for illustration. Describe the performance of your methods. Use high-resolution figures in pdf format.

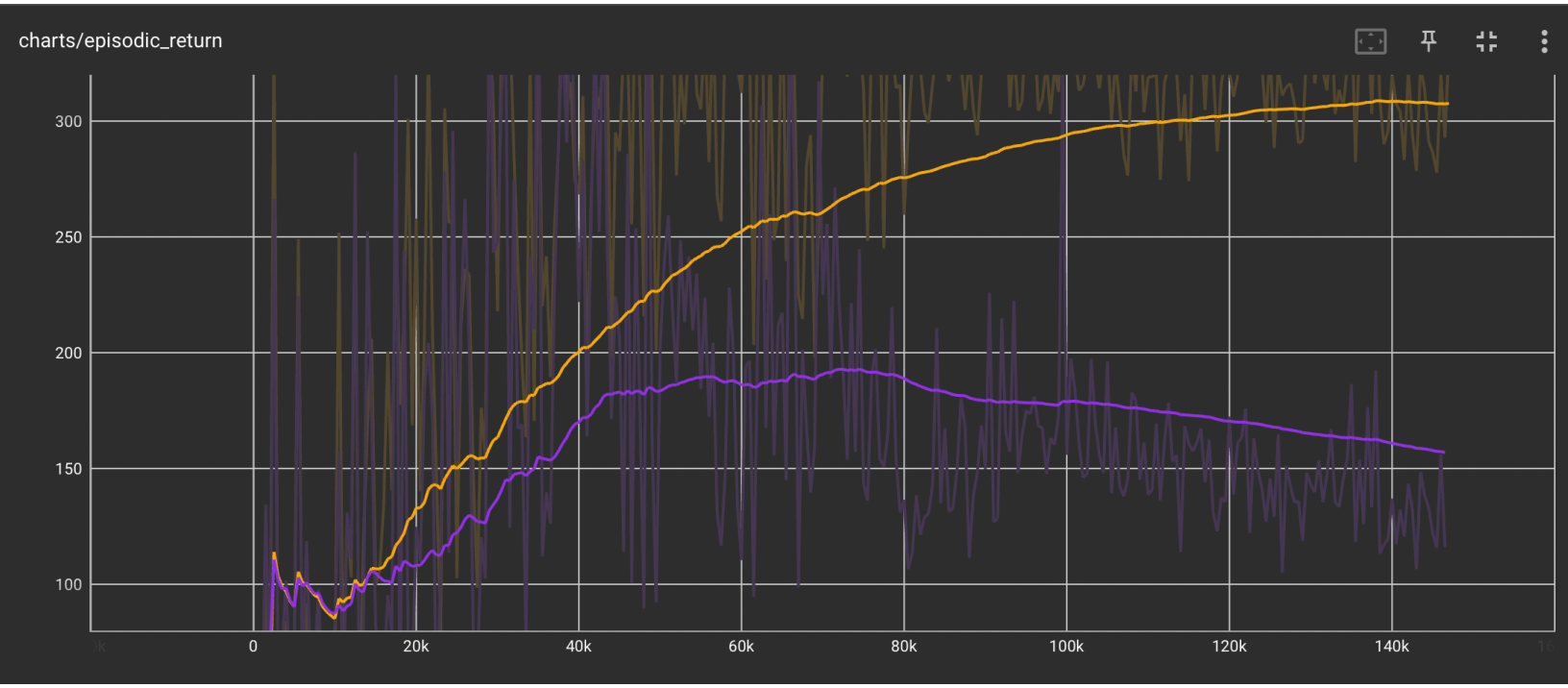
The below diagram shows the performance comparison of using MSE (in puple color) and cosine similarity (in blue color) as the distance measurement. MSE can hardly gain meaningful episodic return, whereas cosine similarity performs reasonably well. We let MSE run extra timesteps, but it ended up decreasing performance even further.


charts/episodic_return
200
150
100
50
0k
0
500k
1M

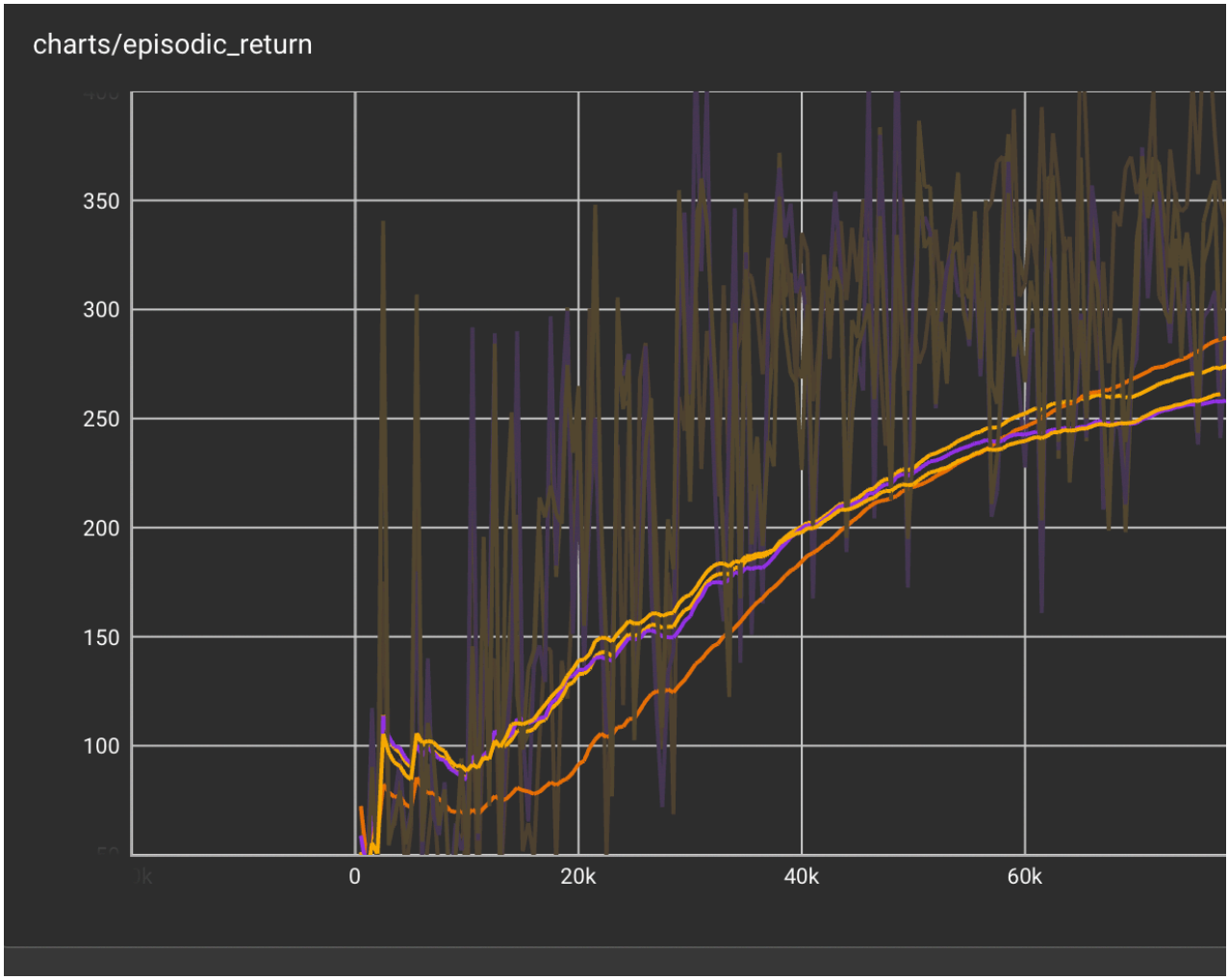
The below diagram compares sampling 10 expert path demonstrations (in color purple) to 400 expert path demonstrations (in orange color), both using cosine similarity as the distance metric.


charts/episodic_return
300
250
200
150
100

We also implemented one subgoal using one next states, and a weighted subgoals using next ten states (in color purple). It shows that a brute force aggregation of ten subgoals in the working memory worsen the performance.


charts/episodic_return
300
250
200
150
100
0
20k
40k
60k
80k
100k
120k
140k

Finally, we compare the vanilla PPO with the expert path planner enhanced RL (in other colors).


charts/episodic_return
350
300
250
200
150
100
0
20k
40k
60k

9 Discussion

It shows that in the first 50k timesteps, the expert path planner consistantly outperformed vanilla PPO, demonstrating a faster convergence. However, as time steps accumulates, PPO performs better. This is phenomenon could be explained as the expert path planner were only giving an expected path, were as RL kept exploring. But expert path planner were able to achieve such return without access to the actual reward function.

The current naive method to calculate the expected state also requires many demonstrations to perform well. Finding a way to compute the optimal stereotypical path instead computing the average may improve sample efficiency. Using Transformer to model the state transition probability may also yields better sample efficiency, as shown by the decision transformer which requires 100 expert demonstrations to do well.

10 Conclusion

Summarize your work, results, and insights, and add any other concluding remarks. Upto this point, we've shown that given a reasonably good expert path planner, we were able to improve the convergence speed of RL training. While keeping looking for ways to improve the sample efficiency such as using an inverse dynamics model, the next step is to train the expert path planner based on other related task trajectories.

References

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. (2016). Rl2: Fast reinforcement learning via slow reinforcement learning. arXiv preprint arXiv:1611.02779.

Huang, S., Dossa, R. F. J., Ye, C., and Braga, J. (2021). Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms.

Janner, M., Li, Q., and Levine, S. (2021). Offline reinforcement learning as one big sequence modeling problem.

Ritter, S., Faulkner, R., Sartran, L., Santoro, A., Botvinick, M., and Raposo, D. (2021). Rapid task-solving in novel environments. *Journal Name*, Volume Number(Issue Number):Page Range.

Yu, T., Quillen, D., He, Z., Julian, R., Narayan, A., Shively, H., Bellathur, A., Hausman, K., Finn, C., and Levine, S. (2019). Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning.