EECS 545: Project Progress Report
Comparison of Various Meta-Learning Paradigms in Few-Shot Preference-Based Reinforcement Learning

Ishan Kapnadak, Sudhanshu Agarwal, Rishikesh Ksheersagar, Shlok Agarwal, Karan Anand
{kapnadak, sudhagar, rishiksh, ashlok, karanand}@umich.edu

June 19, 2024

Abstract

Previous work on preference-based reinforcement learning algorithms with human feedback has provided some promising results in robotics and other domains. However, these methods require an exorbitant number of human queries that are implausible for humans to answer. We explore a few-shot approach to the problem by learning a generalized reward function over multiple tasks and then quickly adapting it to a desired task through the use of a small number of human-annotated queries. We experiment with three different meta-learning paradigms and the inclusion of a prior policy in adaptation.

Introduction

There has been a lot of recent development in the use of reinforcement learning (RL) methods in multiple domains including game-playing and robot control. However, one crucial aspect of RL methods that still faces serious issues is the design of the reward function. While a sparse reward signal does not offer a lot of learning opportunities for the model, designing a dense reward function by hand requires a lot of work, and such a dense reward function is often prone to reward hacking where the agent tries to exploit the specific details of the reward function without actually aligning with the end user goal [Zhu+20]. These issues are further brought to light in multi-task settings with high-dimensional and continuous state and action spaces.

One new paradigm that has emerged to tackle this issue is the use of human preferences. Reward functions that are guided by human preferences are often dense and easy to align with human intent. However, this presents another bottleneck – for the high-dimensional and continuous state and action spaces for robotics tasks, learning from human preferences would require a lot of human data - something that is implausible. Solving this bottleneck can have several positive impacts given the widespread use of RL today. Moreover, if we can provide a method to learn reward functions through the use of a small number of human-annotated examples, we may also be able to further extend the use cases of RL to domains that have suffered from the big data curse. This makes few-shot preference-based RL an interesting and relevant problem to tackle.

Related Work

Early work in improving RL focused on using human-collected data to learn the reward function for a given task. In particular, inverse RL [AN04] gained some popularity in using human demon-

strations to learn rewards. Other strategies that emerged involved humans guiding the agent with natural language instructions [Co-+19], or humans providing rankings [Bro+19]. However, all these methods face challenges - for example, demonstration-based learning is quite expensive and may lead to rewards that do not align with user intentions. Similarly, while natural language instructions and rankings do a better job at aligning the rewards with human intentions, they are still quite hard for the user to provide. The easiest way to involve a human in the loop is via pairwise comparisons, since these sort of comparisons are quite simple for users to provide and still do a good job at aligning rewards with user intentions.

The use of pairwise comparisons gave birth to the domain of preference-based learning. There has been a lot of recent work in preference-based reinforcement learning. For example, [LSA21] proposes PEBBLE which represents one of the state-of-the-art algorithms for human-in-the-loop RL. However, even PEBBLE suffers from the requirement of an unreasonable number of human queries. Our methodology largely follows [HS22] which builds upon PEBBLE and combines preference-based learning with meta-learning [Hos+20]. Meta-learning methods are specifically designed for few-shot scenarios in supervised learning where predictions on new problems are made with only a small amount of data. [HS22] further combines the methodologies of PEBBLE with Model-Agnostic Meta-Learning (MAML, [FAL17]) to propose a solution to the few-shot preference-based reinforcement-learning task. We replicate the methodology proposed in [HS22], and then build on it by experimenting with different meta-learning paradigms and adding a prior policy. These variations are described in depth in the following section.

Methodology

Meta-Learning and Preference-Based Learning

Given access to a dataset of N previous tasks $\{\tau_i\}_{i=1}^N$, our goal is to learn a policy $\pi_{\text{new}}(a \mid s)$ for a new task τ_{new} from human feedback using as few human queries as possible. The method proposed in [HS22] (which we follow) proceeds in two phases. First, we pre-train on the datasets for $\{\tau_i\}_{i=1}^N$ to learn a generalized reward function $\hat{r}_\psi(s, a)$. In the second phase, this reward function is adapted using online human feedback to better suit the task at hand. Instead of trying to learn the reward via regression, we use a preference-based approach. Further, instead of single state-action pairs, we consider partial trajectory segments of the form $\sigma = (s_t, a_t, \ldots, s_{t+k-1}, a_{t+k-1})$ since these are often more informative than single state-action pairs [WFT12]. For two partial trajectory segments σ_1 and σ_2 (of length k where k is chosen appropriately), a preference predictor is defined over these two segments according to the Bradley-Terry model [BT52] as follows:

$$P[\sigma_1 \succ \sigma_2] = \frac{\exp \sum_t \hat{r}_\psi(s_t^1, a_t^1)}{\exp \sum_t \hat{r}_\psi(s_t^1, a_t^1) + \exp \sum_t \hat{r}_\psi(s_t^2, a_t^2)}$$

where $\sigma_1 \succ \sigma_2$ means that σ_1 is preferred over σ_2. Using this, our learning task boils down to a simple classification task of predicting the preferred segment (note that we actually learn the reward function and then use this to classify segment pairs), for which we use the standard binary cross-entropy loss. For a given dataset $\mathcal{D}$ of labelled queries (σ_1, σ_2, y) (where y represents which segment is preferred), the cross-entropy loss being used is

$$\mathcal{L}_{\text{pref}}(\psi, \mathcal{D}) = -\mathbb{E}_{(\sigma_1,\sigma_2,y)\sim\mathcal{D}} \left(\mathbb{I}\{y = 1\} \log P[\sigma_1 \succ \sigma_2] + \mathbb{I}\{y = 2\} \log(1 - P[\sigma_1 \succ \sigma_2])\right)$$

For pre-training, we use labelled datasets $\mathcal{D}_i$ for N tasks where the labels are generated using simulated policies or offline datasets (notice that this means that we can exploit large amounts

of data at this stage without requiring human feedback). The proposed method in [HS22] uses Model-Agnostic Meta-Learning (MAML, [FAL17]) which uses the following update rule for the parameters ψ of the generalized reward function $\hat{r}_\psi(s, a)$:

$$\psi \leftarrow \psi - \beta \nabla_\psi \sum_{i=1}^{N} \mathcal{L}_{\text{pref}} \left(\psi - \alpha \nabla_\psi \mathcal{L}_{\text{pref}}(\psi, \mathcal{D}_i), \mathcal{D}_i \right)$$

where α and β represent the inner and outer learning rates. Once the pre-training is complete, we generate a new dataset $\mathcal{D}_{\text{new}}$ of queries for a new task. This is where the human enters the loop to provide feedback. The advantage of MAML is that it exploits the shared structure between similar tasks so that the generalized reward function can be easily adapted to a new task using just a few human queries. In the adaptation phase, we use a simple gradient descent rule to update

$$\psi \leftarrow \psi - \alpha \nabla_\psi \mathcal{L}_{\text{pref}}(\psi, \mathcal{D}_{\text{new}}).$$

Further, the re-adaptation time can be further reduced by choosing only the most informative queries to adapt the reward function. This is done by using a disagreement metric [Dan+15] that maximizes the deviation of $P[\sigma_1 \succ \sigma_2]$. Once the reward function is re-adapted to the task at hand, we use a Soft Actor-Critic algorithm [Haa+18] to learn the policy π_{new}. The overall learning paradigm is visually depicted below in Figure 1.

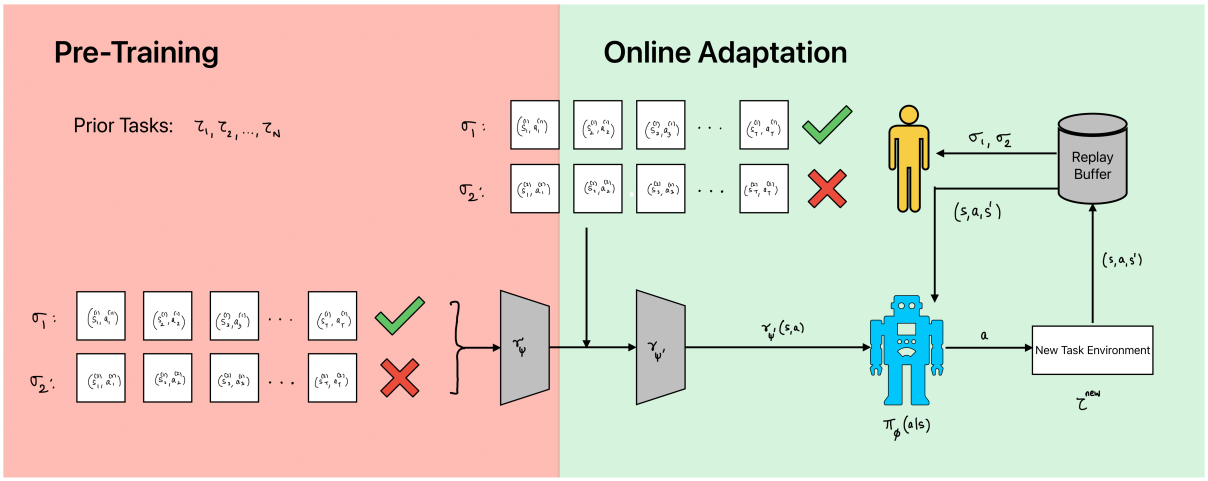


Figure 1: (Adapted from [HS22]) The left half depicts the pre-training phase where labeled queries over different tasks are used to learn a generalized reward function. In the right half, we re-adapt this generalized reward function to a new task using human feedback.

REPTILE and Iterated MAML

We also experiment with two variants of the MAML learning algorithm. Firstly, we experiment with REPTILE [NAS18] which has the following update rule:

$$\psi \leftarrow \psi + \alpha \sum_{i=1}^{N} \left(\tilde{\psi}_i - \psi \right)$$

where $\tilde{\psi}_i$ denotes the updated parameters on task i after running some η iterations of SGD or Adam (where η is a hyperparameter). Secondly, we try an *iterated* version of MAML where the inner step is performed multiple times before performing the outer update. This may be represented as follows

$$\psi \leftarrow \psi - \alpha \nabla_{\psi} \sum_{i=1}^{N} \mathcal{L}_{\text{pref}} \left(\tilde{\psi}_i, \mathcal{D}_i \right)$$

where $\tilde{\psi}_i$ as before denotes the updated parameters on task i after running some η iterations of SGD or Adam. Notice the similarity between the three variants of meta-learning we have seen so far. Our proposed iterated version combines the update form of REPTILE while also adding a second gradient as per MAML.

Inclusion of a Prior Policy

In addition to employing two auxiliary models to learn the reward for a given state-observation pair, we conducted experiments involving the adaptation process by introducing a bias in the form of a prior weighted policy. This adaptation process involves real-time retraining of a pre-trained model to acquire reward functions and policies tailored for accomplishing specific tasks. Unlike the approach outlined in the [HS22] where the model is solely retrained on noisy task data, we opted to generate clean data from a distinct policy within the MetaWorld benchmarks for the same task. We augmented the datasets for both the validation and adaptation phases with this prior expert data.

Our rationale behind incorporating this prior policy data, in conjunction with human feedback, lies in our belief that it offers superior selection for a given set of state-observation pairs. This strategic integration aims to leverage the strengths of both the expert prior data and human input to enhance the adaptation process and ultimately improve task performance.

Data Preparation

For generating data, we use the MetaWorld environment. We consider a total of 10 different tasks within this task, namely – `basketball`, `button-press-topdown`, `door-open`, `drawer-close`, `peg-insert-side`, `pick-place`, `push`, `reach`, `sweep`, `window-open`. For each task, we further consider 25 parametric variations of the ground-truth reward function with some Gaussian noise. For each such variation, we generate 50 episodes where 15 episodes are generated using expert policy, 25 episodes are generated from parametric variations of the same task, and 10 episodes are generated using expert policies from a different task. This gives us a total of 1250 episodes per task. Each episode is made up of a series of entries/rows where each entry/row contains the state (or observation) at that timestep, the action taken, the reward obtained, the discount, and a `done` flag which indicates whether the task has been completed. Our observation space is 39-dimensional and our action space is 4-dimensional. Once the episodes are collected each episode is further divided into segments, each of length l_{seg}. Once the episodes have been segmented, we randomly sample some number of segments and generate comparisons between all these segments. This is done separately for each given task. For a given pair of segments (σ_1, σ_2) we generate the true preference y based on the ground-truth reward. In particular, we compute $\sum_t r(s_t^{(1)}, a_t^{(1)})$ and $\sum_t r(s_t^{(2)}, a_t^{(2)})$ and let y indicate which of the two is greater. Data for the pre-training tasks can be found under the folder `datasets/mw` in our GitHub repository.

Following this, we also generate data for four additional new tasks that we wish to adapt our reward function to, namely – `drawer-open`, `lever-pull`, `shelf-place`, `sweep-into`. This data is

contained in `datasets/mw_valid`. As described in the previous subsection, we also generate data for these same tasks using a prior policy, which can be found under `datasets/mw_valid_policy_v1`.

Implementation

Disclaimer: Although the authors of our primary reference [HS22] published their code on GitHub, we faced a host of issues when trying to use their code for replicating the MAML portion of our experiments. In particular, some of their dependencies were deprecated and could not be used without a bunch of errors popping up. Since their code for data generation was working, we used this to generate our data. Following this, our implementation of MAML and the other algorithms was done from scratch and thus was an intensive task in its own right.

To format the data to be fed into the model we have created functions to break each of the episodes into segments of fixed size k. These segments ($\sigma_{s_i}^{t_j,E_n}$) are then stacked together in a column vector with y denoting the corresponding column vector of ground-truth rewards as shown below:

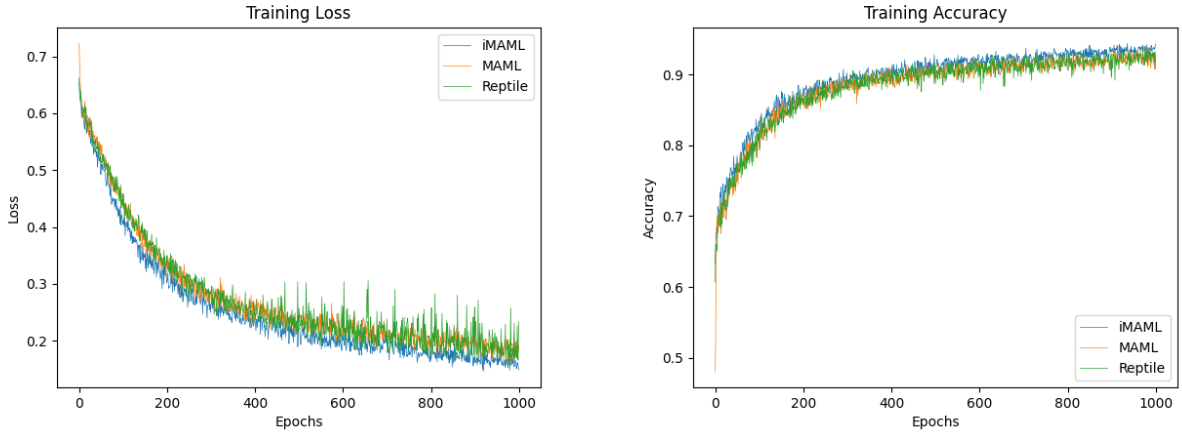
$$X = \begin{bmatrix} \sigma_{s_1}^{t_1,E_1} \\ \sigma_{s_2}^{t_1,E_1} \\ \vdots \\ \sigma_{s_{N_{\text{seg}}}}^{t_1,E_1} \\ \vdots \\ \sigma_{s_{N_{\text{seg}}}}^{t_N,E_{N_{\text{ep}}}} \end{bmatrix} \quad y = \begin{bmatrix} y_{s_1}^{t_1,E_1} \\ y_{s_2}^{t_1,E_1} \\ \vdots \\ y_{s_{N_{\text{seg}}}}^{t_1,E_1} \\ \vdots \\ y_{s_{N_{\text{seg}}}}^{t_N,E_{N_{\text{ep}}}} \end{bmatrix}$$

where $\sigma_{s_i}^{t_j,E_l}$ denotes the i^{th} segment of the l^{th} episode of the j^{th} task (which itself has k rows of state-action pairs) and $y_{s_i}^{t_j,E_l}$ denote corresponding awards. Given the above-formatted data, we randomly select R number of sigma points from each task and feed-forward through a neural network with three linear layers. Due to a time constraint, we were not able to use the disagreement metric [Dan+15] that the original paper [HS22] used for selecting informative queries.

Each state-action pair in our query is fed through a three-layer linear neural network. Each of the hidden layers is appended with a leaky ReLU activation. The activation at the output layer is a tanh activation. We have an input size of 43 (since the state space is 39-dimensional and the action space is 4-dimensional). We further use a hidden size of 256 for each hidden layer.

Experimental Results

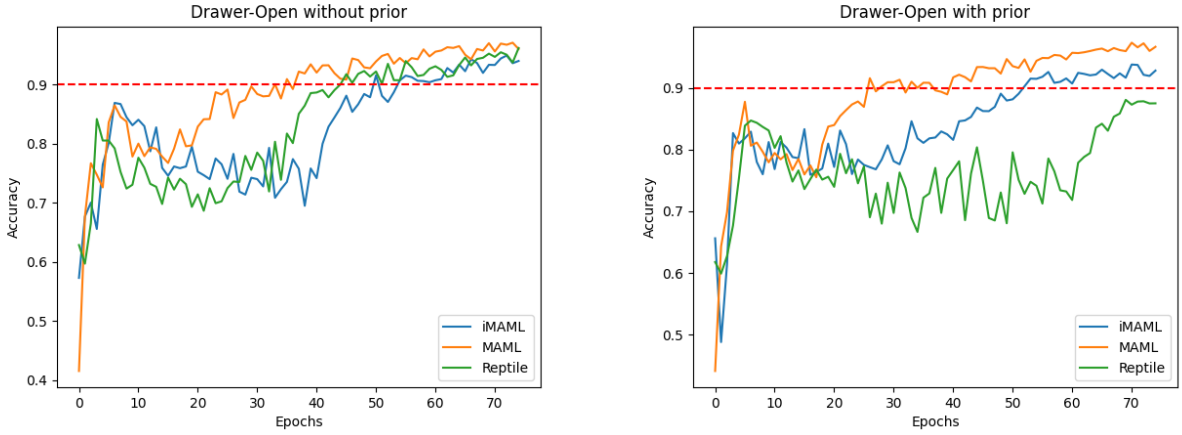
We were able to successfully able to pre-train our reward network using all three learning algorithms. The plots for pre-training loss and accuracy are shown below in Figure 2. As seen, all three algorithms achieve similar performance on both loss and accuracy after 1000 epochs. Once the network was pre-trained, it was saved. Our saved models can be found under the `models` directory. Once the model was pre-trained, we adapted this model to each of the four validation tasks. For each validation task, we had two datasets – one with prior policy incorporated, and one without. Since our main goal was to be able to adapt to a new task in a few-shot scenario, the adaptation phase was run only 75 epochs instead of the 1000 epochs required for pre-training. A comparison of the three algorithms for two of the four tasks has been shown below in Figures 3 and 4.
We see that in all cases, the algorithms were able to achieve good performance on both tasks with as few as 75 epochs. Further, our proposed variant, iMAML, achieves performance comparable to both MAML and REPTILE. However, we were not able to glean a lot of information from the



(a) Loss v/s Epochs

(b) Accuracy v/s Epochs

Figure 2: Performance of all three algorithms in the pre-training phase

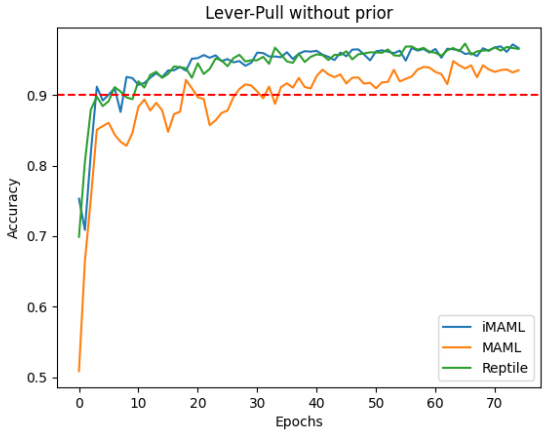


(a) Accuracy on Drawer-Open without prior

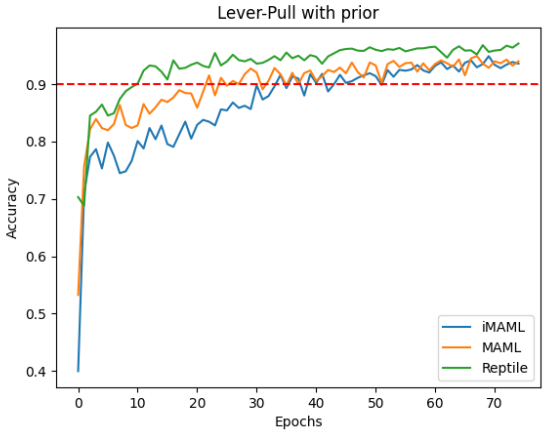
(b) Accuracy on Drawer-Open with prior

Figure 3: Performance of all algorithms on the Drawer-Open task with and without prior

attached plots about how the inclusion of the prior policy affects the performance of our models. We then went on to plot a direct comparison between both cases for all three models. Figures 5 and 6 show these plots. We see that the inclusion of the prior doesn't improve the performance as we had hoped for. Further, in the case of REPTILE for `drawer-open` and iMAML for `lever-pull`, the performance actually significantly deteriorates. Following this, we decided to *reweight* the prior policy in our dataset in order to add some bias towards the existing prior policy. This was achieved by scaling up the rewards for the prior policy dataset by a fixed constant (we chose 1.5 for our experiments here). As we see in Figures 7 and 8, a reweighting of the prior greatly helps our case and improves the performance of our models.

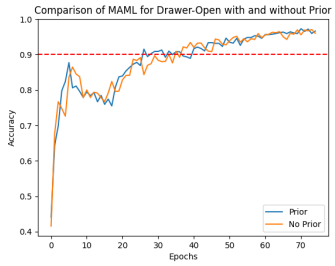


(a) Accuracy on Lever-Pull without prior

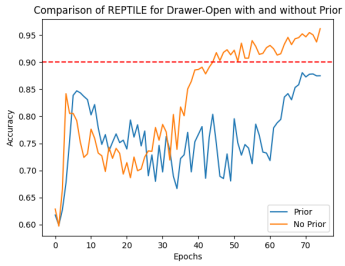


(b) Accuracy on Lever-Pull with prior

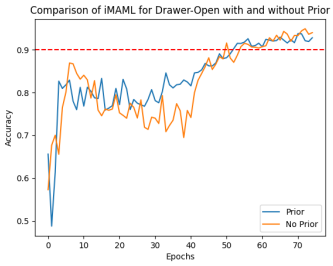
Figure 4: Performance of all algorithms on the Lever-Pull task with and without prior



(a) MAML

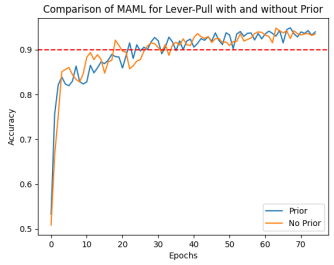


(b) REPTILE

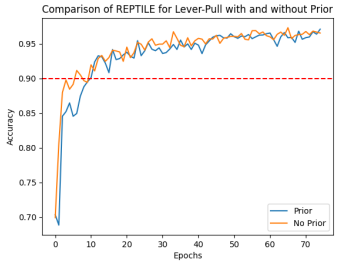


(c) iMAML

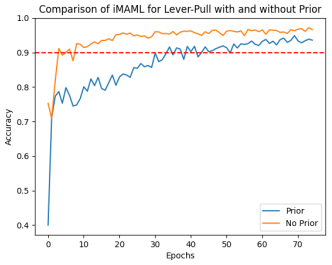
Figure 5: Comparison of prior and no prior for all three algorithms on Drawer-Open



(a) MAML



(b) REPTILE

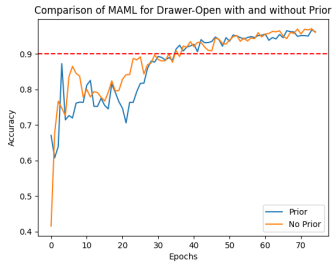


(c) iMAML

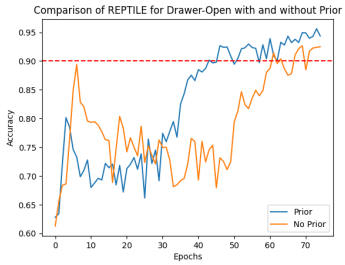
Figure 6: Comparison of prior and no prior for all three algorithms on Lever-Pull

Future Work & Conclusion

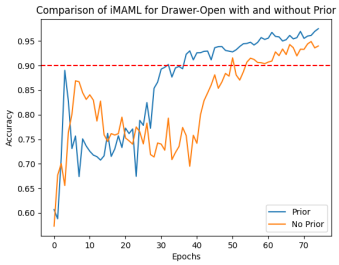
We were able to successfully replicate the algorithm presented in [HS22] and demonstrate its ability to learn a generalized reward function capable of adapting to new tasks. Further, we were also able to implement both REPTILE and iMAML, and both these algorithms achieved performance comparable to MAML. Further, our results show that the network is able to adapt to a new task in as few as 75 epochs (compared to the 1000 epochs required for pre-training), which corroborates the



(a) MAML

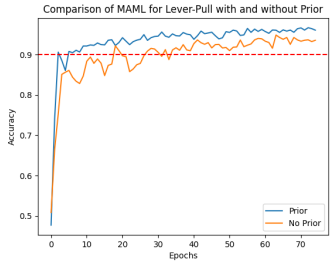


(b) REPTILE

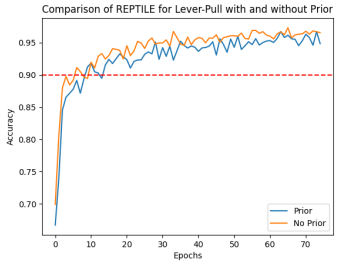


(c) iMAML

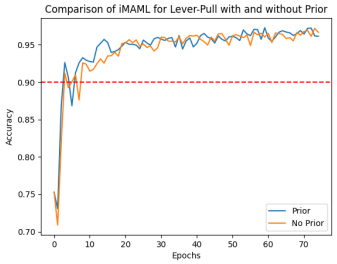
Figure 7: Comparison of prior and no prior for all three algorithms on Drawer-Open with prior reweighted



(a) MAML



(b) REPTILE



(c) iMAML

Figure 8: Comparison of prior and no prior for all three algorithms on Lever-Pull with prior reweighted

few-shot approach we had initially sought. Further, although including the prior policy deteriorated the performance, we saw that a simple reweighting of the prior policy did help in improving the performance of our models in some cases.

So far, we have only learned the reward function and compared performance based on picking preferred segments, however, this doesn't give us a lot of information about the agent's performance in the environment itself. A more systematic approach would be to use the reward function to learn the corresponding policy using a Soft Actor-Critic, and then using this policy simulate the agent on the new tasks. We would then be able to directly look at the success rate of the agent in the environment and be able to better benchmark our methodology. We plan to do this in our future work on this project.

Further, we have as of now reweighted the prior policy by just scaling up the rewards. However, we don't know yet if this affects the quality of the policy learnt since the rewards for the two datasets are on a different scale now. Instead, in the future, we plan to reweight the prior policy by altering the sampling strategy to give more weight to samples generated with a prior policy while sampling random queries.

Author Contributions

Below is a detailed account of each team member's contributions to the project, presented in a table format for clarity. Each category has been renamed to reflect more technical aspects related to our Few-Shot Preference-Based Reinforcement Learning project.

	Ishan	Sudhanshu	Rishikesh	Shlok	Karan
State-of-the-Art Review and Gap Identification	✓	✓	✓	✓	✓
Few-Shot Learning Strategy Formulation	✓			✓	
Data Synthesis and Preparation		✓	✓		✓
Algorithm Implementation and Optimization	✓	✓	✓	✓	✓
Inclusion and Reweighting of Prior Policy	✓			✓	
Results Analysis and Interpretation	✓	✓	✓	✓	
Comprehensive Documentation and Reporting	✓			✓	

Table 1: Checklist of contributions of each team member to the project

Ishan Kapnadak worked on all aspects of the project, contributing to the literature review, methodology development with a focus on MAML and iMAML, implementation of the model, inclusion and experimenting with prior policy, generating plots and results, and drafting the final report.
Sudhanshu Agarwal equally contributed to the literature survey regarding MAML and PEBBLE, was involved in data collection and processing, participated in coding the MAML framework into our solution, and helped in writing and revising our documentation.
Rishikesh Ksheersagar was integral in the literature review, methodology design particularly in adapting MAML for our model and implementing REPTILE, handled data segmentation and preparation.
Shlok Agarwal shared responsibilities in reviewing relevant literature, devising the project methodology with emphasis on MAML integration, inclusion of prior policy, played a significant role in implementing the model, and contributed to the analysis and documentation of our results.
Karan Anand engaged in all project phases, with a focus on literature survey, integrating the data framework with our implementation of MAML, and generating and formatting data to be fed into the models.

References

[BT52] Ralph Allan Bradley and Milton E. Terry. "Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons". In: *Biometrika* 39 (1952), p. 324. URL: https://api.semanticscholar.org/CorpusID:125209808.

[AN04] Pieter Abbeel and Andrew Y. Ng. "Apprenticeship learning via inverse reinforcement learning". In: *Proceedings of the Twenty-First International Conference on Machine Learning*. ICML '04. Banff, Alberta, Canada: Association for Computing Machinery, 2004, p. 1. ISBN: 1581138385. DOI: 10.1145/1015330.1015430. URL: https://doi.org/10.1145/1015330.1015430.

[WFT12] Aaron Wilson, Alan Fern, and Prasad Tadepalli. "A Bayesian approach for policy learning from trajectory preference queries". In: *NIPS* 25 (Jan. 2012), pp. 1142–1150.

[Dan+15] Christian Daniel et al. "Active reward learning with a novel acquisition function". In: *Auton. Robots* 39.3 (Oct. 2015), pp. 389–405. ISSN: 0929-5593. DOI: 10.1007/s10514-015-9454-z. URL: https://doi.org/10.1007/s10514-015-9454-z.

[FAL17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 1126–1135. URL: https://proceedings.mlr.press/v70/finn17a.html.

[Haa+18] Tuomas Haarnoja et al. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 1861–1870. URL: https://proceedings.mlr.press/v80/haarnoja18b.html.

[NAS18] Alex Nichol, Joshua Achiam, and John Schulman. "On First-Order Meta-Learning Algorithms". In: *CoRR* abs/1803.02999 (2018). arXiv: 1803.02999. URL: http://arxiv.org/abs/1803.02999.

[Bro+19] Daniel Brown et al. "Extrapolating Beyond Suboptimal Demonstrations via Inverse Reinforcement Learning from Observations". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 783–792. URL: https://proceedings.mlr.press/v97/brown19a.html.

[Co-+19] John D. Co-Reyes et al. *Guiding Policies with Language via Meta-Learning*. 2019. arXiv: 1811.07882 [cs.LG].

[Hos+20] Timothy Hospedales et al. *Meta-Learning in Neural Networks: A Survey*. 2020. arXiv: 2004.05439 [cs.LG].

[Zhu+20] Henry Zhu et al. "The Ingredients of Real-World Robotic Reinforcement Learning". In: *CoRR* abs/2004.12570 (2020). arXiv: 2004.12570. URL: https://arxiv.org/abs/2004.12570.

[LSA21] Kimin Lee, Laura Smith, and Pieter Abbeel. *PEBBLE: Feedback-Efficient Interactive Reinforcement Learning via Relabeling Experience and Unsupervised Pre-training*. 2021. arXiv: 2106.05091 [cs.LG].

[HS22] Joey Hejna and Dorsa Sadigh. *Few-Shot Preference Learning for Human-in-the-Loop RL*. 2022. arXiv: 2212.03363 [cs.RO].