DA7400: Recent Advances in Reinforcement Learning

Endterm Report

Adversarial Offline RL with Bidirectional Imagination

Anirud N **CE21B014**
Navin Sriram **ED21B044**

1 Introduction

Offline RL has seen a lot of progress in recent years in different fields in robotics and other safety-critical applications. One common idea shared within each of these proposed workflows is a form of regularization to avoid distributional shift, .i.e, a way to make sure your model does not go bonkers when it encounters an unseen state transition. Model-free offline RL methods often effectively address distributional shift issues, but their constrained policy search can limit the generalization beyond the offline dataset. In contrast, model-based offline RL first learns a forward dynamics model from the offline dataset with conservatism quantifications and then generates imaginary trajectories/rollouts. These methods use a conservative estimation of the value functions to ensure the confidence of model-based rollouts. However, because samples in offline datasets are limited, conservatism quantifications often suffer from over-generalization, especially in out-of-support regions. We explore **Reverse Model Based Imagination** [1] which outperforms most Offline RL algorithms. We incorporate this notion of generating reverse rollouts enables effective conservative generalization and use this in a robust adversarial learning framework similar to the one suggest in RAMBO-RL [2] using the existing adversarial pipeline for robust performance.

In model-based offline reinforcement learning (RL), generating rollouts from learned models of the environment/transition dynamics is a common approach for data augmentation. However, standard methods often rely solely on forward dynamics models, limiting the diversity and scope of the simulated data. To address this limitation, we propose a novel framework incorporating bidirectional imagination, leveraging forward and reverse dynamics models to generate diverse trajectories. We aim to analyze the effects of using these bidirectional rollouts for dataset augmentation while adversarially optimizing both the policy and the learned state dynamics models.

This method is inspired by two significant papers in model-based offline RL: RAMBO-RL (Robust Adversarial Model-Based Offline RL) and ROMI (Offline RL with Reverse Model Imagination), each highlighting potential gains from adversarial optimization and reverse imagination. Our work builds on their future suggestions, specifically integrating bidirectional imagination into policy optimization.

2 Why Reverse Imagination?

The paper [1] introduces Reverse Offline Model-based Imagination (ROMI). This algorithm uses reverse rollouts sampled from a "generative model policy". This model policy is learned through the "reverse transition dynamics," i.e., similar to imagining the original transition dynamics from backward. [1] claims that reverse imaginations generate possible traces leading to target goal states inside the offline dataset, which provides a conservative way of augmenting the offline dataset. Reverse models maintain the generalization ability to interpolate between given trajectories and potentially better policies and avoid radical model imagination by enabling bidirectional search in conjunction with the existing forward real trajectories in the offline dataset. ROMI trains a reverse dynamics model and generates backward imaginary trajectories by a backtracking rollout policy.

Recent Works like ROMI [1] show the success of such methods, but a key assumption involved is a near-expert dataset to build the reverse dynamics model. ROMI trains this in a single shot, rolls out trajectories and uses model-free offline RL methods after this. However, access to near-expert datasets cannot always be assumed. We aim to build upon this idea to execute it along with adversarial updates to make the training more Robust and better performance even for noisy datasets.

Adversarial Updates We further update the model adversarially with the policy to see if it makes it robust to noisy demonstrations. Additionally, we reinforce RI with Forward Imagination, one of the authors' stated future works.

3 Why Adversarial Updates?

RAMBO-RL [2] shows that Model-based policies efficiently handle the problem of distributional shift using adversarial dynamic model updates when used with Forward Dynamics Models. It also shows the success of adversarial updates in medium and noisy datasets. We can adjust the transition model to enforce conservatism by introducing adversarial dynamics. This ensures that our agent doesn't venture too far into uncertain territory where the learned dynamics may be inaccurate. RAMBO-RL introduced this as a Two-player 0-sum game where the dynamics tries to minimize the value function, and the policy tries to maximize it. RAMBO [2] showed the failure of current model-based approaches in environments such as Antmaze. The model-free algorithms perform significantly better, giving a significantly high positive score. This is because the current model-based algorithms are aggressive and in this scenario more prone to collision with the wall. This calls for a need for better model-based offline algorithms that enhance performance. . This is because the model-based algorithms are not conservative enough, and their policies often lead to a collision with the walls, leading to increased negative rewards. To mitigate the above issue and to enforce more conservatism, we introduce:

A two simultaneous 2 player Zero-Sum Game: We propose to extend this model into a simultaneous 2 two-player zero-sum game, incorporating:

- Policy
- Forward Dynamics Model
- Reverse Dynamics Model

This new structure allows for a more nuanced interaction between the policy and the dynamics models, further enhancing conservatism.

4 Methodology

Algorithm 1 Forward and Reverse Rollout with Adversarial Updates

Require: Normalized dataset $\mathcal{D}$
1: $F_{\phi_1} \leftarrow$ Forward dynamics model - MLE
2: $R_{\phi_2} \leftarrow$ Reverse dynamics model - MLE
3: $\pi_f \leftarrow$ Forward Rollout Policy
4: $\pi_r \leftarrow$ Reverse Rollout Policy
5: **for** $i = 1, 2, \ldots, n_{\text{iter}}$ **do**
6: Generate k-step forward rollouts using π_f Add transition data to $\mathcal{D}_{F_{\phi_1}}$
7: Generate k-step reverse rollouts using π_r. Add transition data to $\mathcal{D}_{R_{\phi_2}}$
8: **Agent update:** Update π_f and π_r using samples from $\mathcal{D} \cup \mathcal{D}_{F_{\phi_1}} \cup \mathcal{D}_{R_{\phi_2}}$ using a SAC Framework
9: **Adversarial updates:** Adversially update the forward and reverse transition dynamics models F_{ϕ_1} and R_{ϕ_2}
10: **end for**

The algorithm involves the initialization of the dynamics models and rollouts for both forward and reverse imaginations. Then, we generate k-step rollouts using the current policies of forward and reverse and add this synthetic dataset to the original dataset. Using this total dataset, we use the Soft Actor-Critic framework to update the policy. Then, we incorporate a competing framework, where the forward and reverse dynamics models compete with each other - whichever is higher in estimating

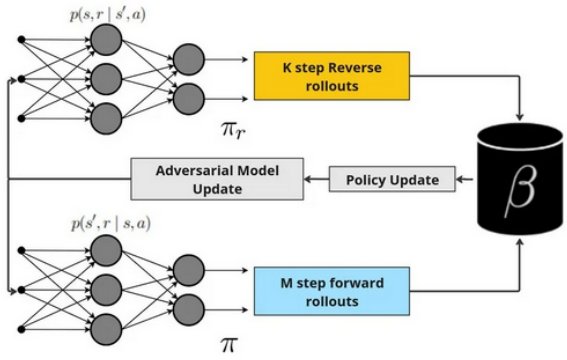


Figure 1: The algorithm

the value function - would be adversely updated. This is the core of our 3-player 0-sum game - the forward and reverse models compete among themselves, and then they compete with the agent by inhibiting its learning.

Algorithm specifics	RAMBO	ROMI	OUR approach
Adversarial Updates	Yes	No	Yes
Reverse model imagination	No	Yes	Yes
Multi-agent game	2	-	2 - 2

Table 1: Comparison of RAMBO, ROMI, and our approach

4.1 Initialising the Transition dynamics models and rollout Policies

We start with the offline dataset D. We train the Forward and the Reverse dynamics model using **MLE - maximum likelihood estimator loss**. We modify the dataset $D = \{(s_t, a_t, r_t, s_{t+1})\}$, where s_t represents the current state, a_t the action taken, r_t the reward obtained after the action, and s_{t+1} the next state.

Forward Dynamics Model with Rewards The forward dynamics model F_{ϕ_1} now predicts both the next state s_{t+1} and the reward r_t given the current state s_t and action a_t:

$$F_{\phi_1}(s_t, a_t) \approx (s_{t+1}, r_t)$$

The Maximum Likelihood Estimation (MLE) loss for the forward dynamics model is given by:

$$\mathcal{L}_{\text{forward}}(\phi_1) = - \sum_{(s_t, a_t, r_t, s_{t+1}) \in D} \log p(s_{t+1}, r_t | s_t, a_t; \phi_1)$$

Reverse Dynamics Model with Rewards The reverse dynamics model R_{ϕ_2} predicts the previous state s_t and the reward r_t given the next state s_{t+1} and the action a_t:

$$R_{\phi_2}(s_{t+1}, a_t) \approx (s_t, r_t)$$

The MLE loss for the reverse dynamics model is given by:

$$\mathcal{L}_{\text{reverse}}(\phi_2) = - \sum_{(s_t, a_t, r_t, s_{t+1}) \in D} \log p(s_t, r_t | s_{t+1}, a_t; \phi_2)$$

Implementation of model pretraining The dynamics pre-training loss function is defined as a maximum likelihood estimation (MLE) objective. First, the model predicts the mean μ and the log variance $\log \sigma^2$.
The loss function $\mathcal{L}$ is given by:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} \left((y_{ij} - \mu_{ij})^2 \cdot \sigma_{ij}^{-2} \right) + \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} \log \sigma_{ij}^2 + D$$

where the first term is the weighted squared error between the target values y_{ij} and the predicted means μ_{ij}, scaled by σ_{ij}^{-2}. The second term penalizes high variances by including the log of the variance, $\log \sigma_{ij}^2$. D represents an additional constant. To regularize the loss, an additional term is added:

$$\mathcal{L} = \mathcal{L} + \lambda \sum \text{max\_logvar} - \lambda \sum \text{min\_logvar}$$

Here, λ is a regularization coefficient, and the terms max_logvar and min_logvar control the spread of log variance values. This regularization helps stabilize training by constraining the range of predicted variances.

This formulation enables the model to optimize both the mean and variance predictions, improving robustness and capturing the uncertainty in the dynamics.

Behavior Cloning for Forward Policy with Rewards The forward policy $\pi_f(a_t|s_t, r_t)$ is trained to maximize the likelihood of the actions a_t given the states s_t and rewards r_t. The objective for training the forward policy is using MSE.

Behavior Cloning for Reverse Policy with Rewards The reverse policy $\pi_r(a_t|s_{t+1}, r_t)$ is trained to predict the actions a_t that lead to the previous state s_t and the corresponding reward r_t. The objective for training the reverse policy is using MSE.

4.2 K synthetic rollouts

Rollouts are essentially used to generate synthetic data to better represent the environment for more effective training. But the paper: Model based Policy Optimisation discusses the issues of having very large synthetic data steps. Fig 2 shows that the variation increases as the number of steps increases,

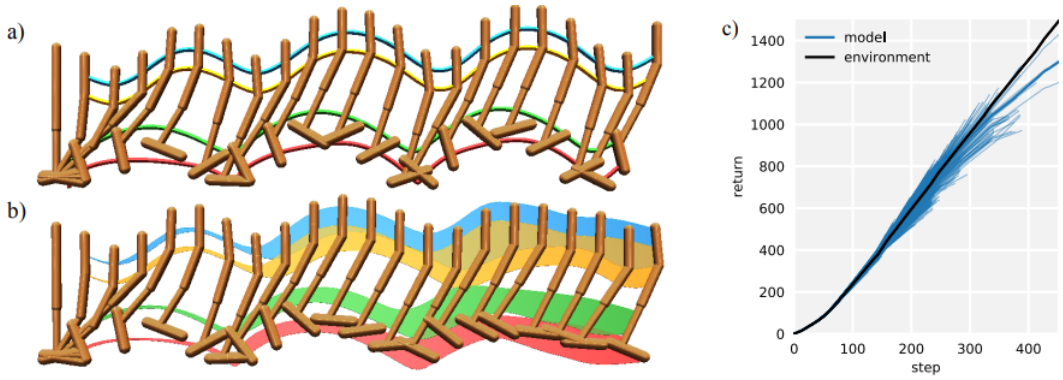


Figure 4: a) A 450-step hopping sequence performed in the real environment, with the trajectory of the body's joints traced through space. b) The same action sequence rolled out under the model 1000 times, with shaded regions corresponding to one standard deviation away from the mean prediction. The growing uncertainty and deterioration of a recognizable sinusoidal motion underscore accumulation of model errors. c) Cumulative returns of the same policy under the model and actual environment dynamics reveal that the policy is not exploiting the learned model. Thin blue lines reflect individual model rollouts and the thick blue line is their mean.

Figure 2: k step

implying that the errors increase and get added progressively as the number of steps increases. So, it is necessary to set a threshold k for the number of steps while doing rollouts for the synthetic dataset. k is set as a hyperparameter.

4.3 Agent update

Both the forward and the reverse policies are updated using a SAC framework. The replay buffer mainly consists of 3 parts:

- Real replay buffer - the offline dataset that we have
- Fake forward replay buffer - the replay buffer that we generate using forward rollouts on the forward transition dynamics model
- Fake reverse replay buffer - the replay buffer that we generate using the reverse rollout on the reverse transition dynamics model

The fake forward and the fake reverse replay buffer is the **Synthetic dataset** that we build by performing rollouts on the transition dynamics models. The SAC samples with a probability of f from the real replay buffer and a probability of 1-f from the synthetic buffer. f is set to be 0.5

4.4 Model Gradient

The concept of model gradient is very similar to the policy gradient approach. As a result, even the equation is very similar. Let ϕ denote the parameters of a parametric MDP model $\hat{T}_\phi$, and let V_ϕ^π denote the value function for policy π in $\hat{T}_\phi$. Then:

$$\nabla_\phi V_\phi^\pi = \mathbb{E}_{s\sim d_\phi^\pi, a\sim\pi_\phi, (s',r)\sim\hat{T}_\phi}\left[(r+\gamma V_\phi^\pi(s'))\cdot\nabla_\phi \log \hat{T}_\phi(s',r|s,a)\right] \tag{1}$$

$$\nabla_\phi V_\phi^\pi = \mathbb{E}_{s\sim d_\phi^\pi, a\sim\pi_\phi, (s',r)\sim\hat{T}_\phi}\left[(r+\gamma V_\phi^\pi(s') - Q_\phi^\pi(s,a))\cdot\nabla_\phi \log \hat{T}_\phi(s',r|s,a)\right] \tag{2}$$

Eq(6) is also derived from a similar approach, showing that adding a baseline that is independent of the transition dynamics, i.e., s' and r, helps in learning and it reduces variance. To estimate V_ϕ^π and Q_ϕ^π, we use the critic learnt by the actor-critic algorithm used for policy optimisation. Q_ϕ^π is the expected Value of taking a state s and following the policy thereafter. The advantage term here is normalised to avoid any issues in the model training specifically due to large variations in gradient.

4.5 Adversarial Model Training

This section talks about how do we use model gradient to solve the adversarial training problem. This can be formulated as :

$$\min_{\hat{T}_\phi} V_\phi^\pi, \quad \text{s.t.} \quad \mathbb{E}_\mathcal{D}\left[\mathrm{TV}(\hat{T}_{\mathrm{MLE}}(\cdot|s,a), \hat{T}_\phi(\cdot|s,a))^2\right] \le \xi. \tag{3}$$

We convert this into an unconstrained optimization problem using Lagrange multipliers.

$$\max_{\lambda\ge 0}\min_{\hat{T}_\phi} L(\hat{T}_\phi,\lambda) := V_\phi^\pi + \lambda\left(\mathbb{E}_\mathcal{D}\left[\mathrm{TV}(\hat{T}_{\mathrm{MLE}}(\cdot|s,a), \hat{T}_\phi(\cdot|s,a))^2\right] - \xi\right) \tag{4}$$

To facilitate for the easier tuning of the learning rate and for a simplified objective function, this was made into;

$$\mathcal{L}_\phi = \lambda V_\phi^\pi - \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\log \hat{T}_\phi(s',r|s,a)\right]. \tag{5}$$

Lesser the lamda - more importance is given to the $\hat{T}$ being close to $\hat{T_{MLE}}$ so that the model fits the dataset accurately and less weight to the adversarial part. The aim of the adversarial training is to minimise $\mathcal{L}_\phi$

The gradient for this loss function is computed using simple mini-batch gradient descent, and the gradient for the value term is calculated using Model Gradient:

$$\nabla_\phi V_\phi^\pi = \mathbb{E}_{s\sim d_\phi^\pi, a\sim\pi_\phi, (s',r)\sim\hat{T}_\phi}\left[(r + \gamma V_\phi^\pi(s') - Q_\phi^\pi(s,a)) \cdot \nabla_\phi \log \hat{T}_\phi(s', r|s, a)\right] \tag{6}$$

5 Implementation Details

Dynamics Model A Bayesian Neural Network (BNN) to model uncertainties in its predictions. We use an ensemble of 7 networks and run inference on the top 5 performers on the holdout set. We take the MLE loss for the forward and the reverse models

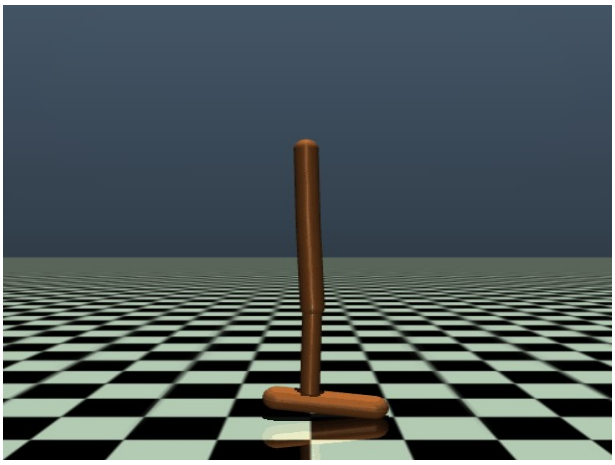
Reverse Rollout Policy that learns using the real + fake(fake forward + reverse) replay buffers, where sampling is done with a probability of f from the real buffer and 1-f from the fake buffer.

5.1 Key Tunable Hyperparameters

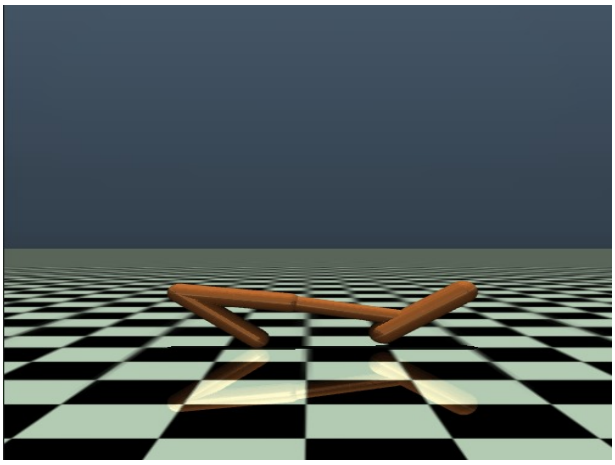
Adversarial loss weight = 3e-4
rollout length = 5
real ratio = 0.5 (ratio of number of transitions sampled from real to synthetic dataset)
Dynamics update frequency = 10000
Actor learning rate = 1e-4
Critic learning rate = 3e-4

5.2 Visualisation of the trained reverse rollout policy

Link to visualizations - here.



(a) Start state - hopper



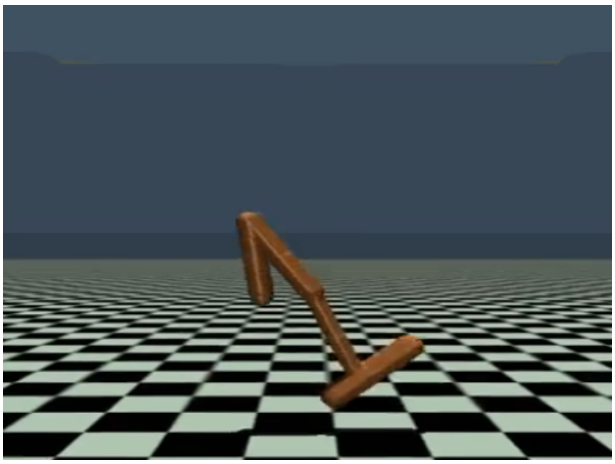
(b) End state - hopper

Figure 3: Reverse rollout visualization

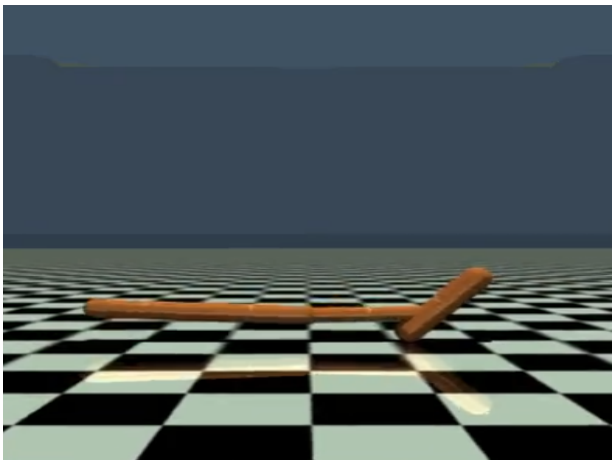
6 Results So Far

NOTE: X-axis is num epochs; for our algorithms, it's around 1000, and baselines are trained for 2000. Reward is averaged for the last 10 epochs over a single seed.

- We were able to train the policy for the above tasks for about 500-1500 epochs.
- Hopper performs better than other SOTA algorithms, underperforming in the medium replay case since Replay datasets are more diverse and include suboptimal data, as they contain the replay buffer of the training process. Standard deviations are very low except for medium replay.



(a) Start state - hopper-medium-v2



(b) End state - hopper-medium-v2

Figure 4: Reverse rollout visualization

	Ours	RAMBO	ROMI	COMBO	MOPO	CQL
Hopper Medium	$\mathbf{105.27 \pm 0.74}$	91.2 ± 16.3	72.3 ± 17.5	94.9	48.0	66.6
Hopper Medium Replay	64.36 ± 29.53	97.6 ± 3.4	$\mathbf{98.1} \pm 2.6$	73.1	97.0	48
Hopper Medium Expert	$94.76 \pm \mathbf{0.85}$	89.5 ± 11.1	$\mathbf{111.4} \pm 5.6$	111.1	106.8	48
Hopper Random	$\mathbf{31.61 \pm 0.63}$	15.5 ± 9.4	30.2 ± 4.4	17.9	4.1	6.7
HalfCheetah Medium	51.6 ± 1.2	71.0 ± 3.0	49.1 ± 0.8	54.2	**69.5**	49.0
HalfCheetah Med. Replay	12.27 ± 14.1	67.0 ± 1.5	47.0 ± 0.7	55.1	68.2	47.1

- The lower standard deviation is comparatively lower than other algorithms, beating COMBO, a SOTA constrained model-based offline RL approach. This could indicate safe and stable learning.
- Cheetah performs worse than partially trained RAMBO but shows a gradual increase in the reward. Since this is a basic implementation where we directly combine hyperparameters from the original RAMBO and ROMI player, we could fine-tune the parameters.
- We provide additional analysis, visualizing and comparing the training process with other algorithms.

7 Short Studies

7.1 What are Reverse Rollouts doing?

- To understand this, we trained the policy with 1. both forward and reverse rollouts, 2. with reverse rollouts and 3. with only forward rollouts. The policy showed best and most stable learning using combined rollouts, but this is just for a single environment (Hopper Medium V2). Policy 2 shows very high instability, indicating that injecting the reverse rollouts adds some indirect constraint, reducing the standard deviation and achieving optimal performance in half the number of epochs.
- A very big surprise was that Policy 3 performed very poorly, with a reward of around 0.75, contrary to the results of ROMI, which used reverse rollouts and performed better. We came up with that this could be due to any of the following reasons
  - The original ROMI paper used a VAE for simulating the reverse rollout policy. This was trained one-shot using vanilla BC. VAE was used to encourage diversity, according to the author. In our approach, we use SAC with a standard MLP and a slightly higher entropy

coefficient for exploration. SAC might be trying to be too greedy compared to the VAE, which is one of the thoughts on our minds.

- The ROMI paper trains the reverse dynamics and rollout policy in a single shot and utilizes them for data augmentation for a model-free offline RL algorithm, Constrained Q Learning in their case. We could check if introducing a model-based RL algorithm is causing any issues, and we could check the literature more thoroughly. This is just a guess, and we are unsure if this is the reason.
- **Even though using reverse rollout for policy learning is bad, its still improving the stability of the learning process for the combined rollout policy, we need to look deeper into the transitions themselves**. The reverse rollout could be just seen as noise when implemented individually but somehow it works while integrating with forward rollouts.

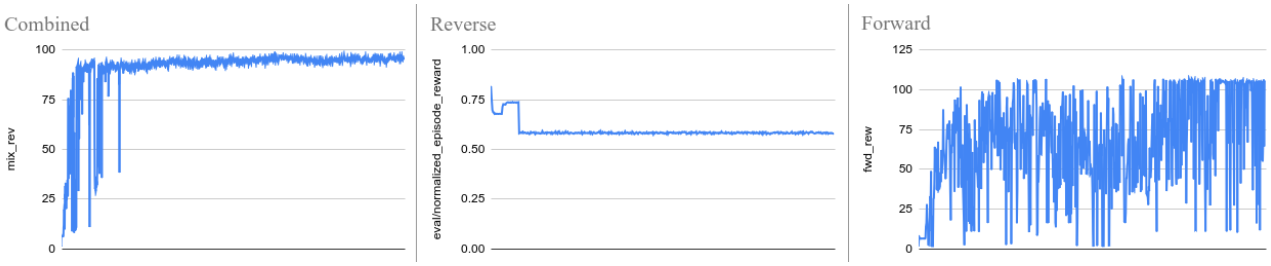


Figure 5: Comparing Unidirectional Imagination results

7.2 Stable Learning?

- In almost all cases, we see a comparatively smoother learning curve with a very, very low standard deviation. Even adding a single transition is causing a significant impact on the standard deviations by order of 100s.
- **This stable learning shows some implicit constraints being enforced due to the data augmentation.** While this field of study has not been looked at thoroughly, we could delve deeply into the impact of data quality and selective modification for enforcing implicit constraints by analysing the patterns within the dataset generated using our approach.

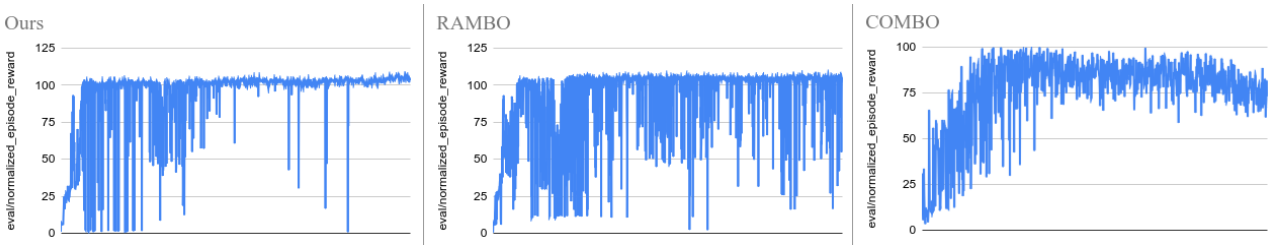


Figure 6: Hopper Medium V2

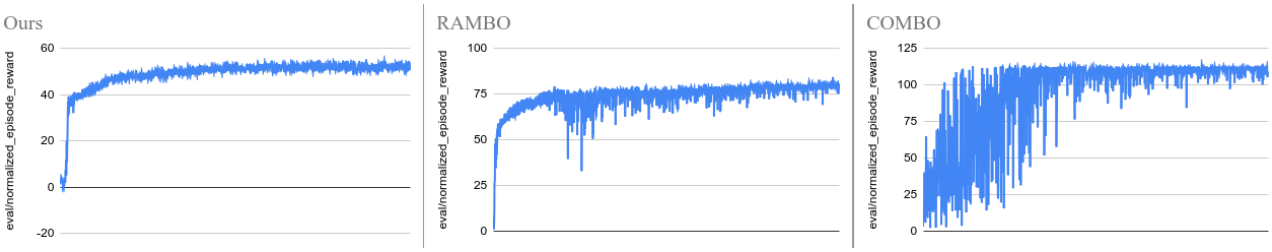


Figure 7: Halfcheetah Medium V2

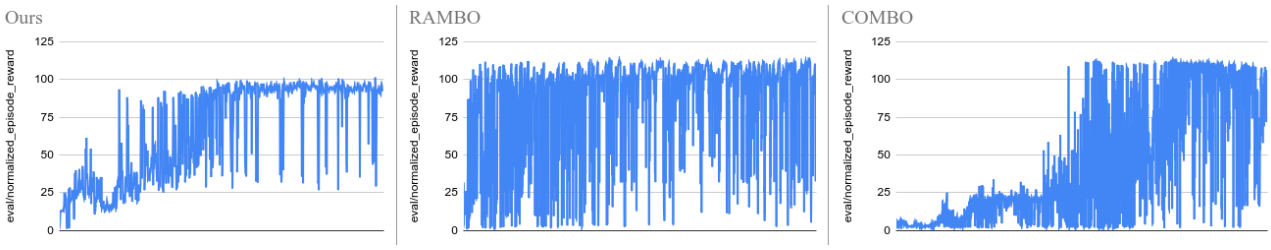


Figure 8: Hopper Medium Expert V2

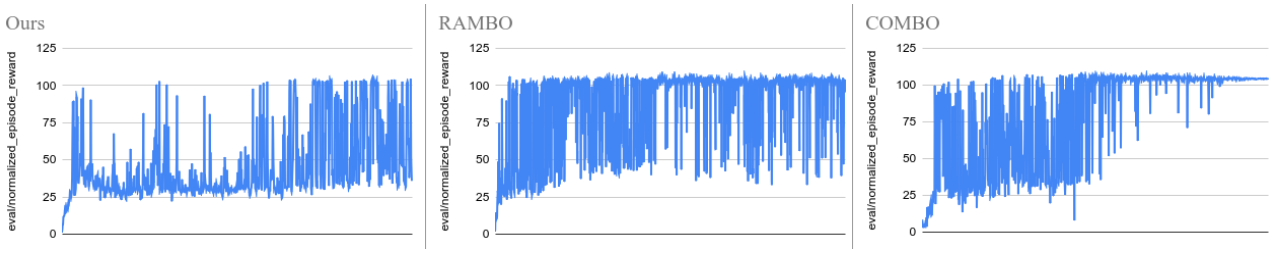


Figure 9: Hopper Medium Replay V2

7.3 Rollout Length

- Increasing the rollout length seems to increase the deviation during training. This is in line with most studies, which show that increasing the rollout size does indeed increase the variance in learning.

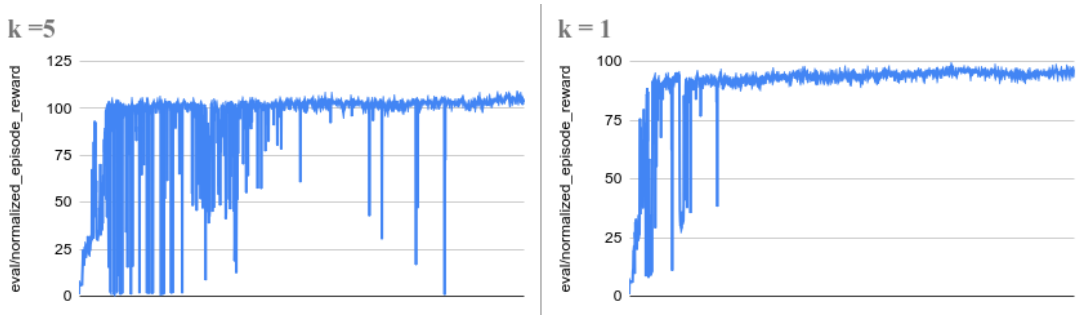


Figure 10: Comparing Effect of rollout length (k)

8 Future work

We would like to discuss on the future work and prospects of the problem statement and the approach presented with Prof. Ravindran or the TAs for seeking guidance.

- Implement a VAE for performing reverse rollouts and analysing the performance
- Perform the studies mentioned previously, particularly on the topic of enforcing implicit constraints through focused data augmentation
- "3 player 0 sum game" - formalizing a competing framework between forward and reverse transition dynamic through adversarial updates
- Performing more thorough experiments on more tasks

This has been a wonderful experience, as we found a huge learning curve. We were exposed to the current real-world problems and subsequent areas of research in RL. We had the opportunity to do hands-on implementation of SOTA architecture, looking at how large codebases are deployed and maintained. We would like to thank the instructor and the TAs for this opportunity.

References

[1] Jianhao Wang, Wenzhe Li, Haozhe Jiang, Guangxiang Zhu, Siyuan Li, and Chongjie Zhang. Offline reinforcement learning with reverse model-based imagination. *Advances in Neural Information Processing Systems*, 34:29420–29432, 2021.

[2] Marc Rigter, Bruno Lacerda, and Nick Hawes. Rambo-rl: Robust adversarial model-based offline reinforcement learning. *Advances in neural information processing systems*, 35:16082–16097, 2022.