DA7400
Recent Advances in RL

End Term Report

Srinivasan Kidambi ee21b139@smail.iitm.ac.in

Justin Rangad ee21b062@smail.iitm.ac.in

Code Presentation

Indian Institute of Technology, Madras

November 14, 2024

Contents

1 Introduction **2**

2 Framework **3**
2.1 CVAE . . . 3
2.2 Low Level Agent . . . 4
2.2.1 Reward Relabeling and Noisy Sampling: . . . 4
2.2.2 Low Level Training Algorithm: . . . 5
2.3 High Level Planner . . . 5
2.3.1 Optimization Objectives . . . 5
2.3.2 Optimization Method (CEM) and Planning Sequence . . . 6

3 Implementation Details **7**
3.1 Environments: . . . 7
3.2 Low Level Agents: . . . 7
3.2.1 Why not CQL-SAC . . . 7
3.2.2 IQL . . . 8
3.2.3 TD3+BC . . . 8
3.3 CVAE Network . . . 10
3.4 High Level Planner . . . 11

4 Results **11**

5 Inferences **13**

6 Improvements **13**

7 Conclusion **14**

8 Acknowledgments **15**

List of Tables

1 Performance of IQL and TD3+BC in our environments . . . 8
2 IQL Hyperparameters . . . 9
3 TD3+BC Hyperparameters . . . 9
4 Hyperparameters for HiGOC and distance-HiGOC planners . . . 11
5 Performance of various low level agents and their HiGOC counterparts . . . 11
6 Comparison of Distance HiGOC-IQL and HiGOC-IQL runtimes . . . 14

1 Introduction

This report presents the final project we completed for the course DA7400. **The code and presentation links can be found on the cover page.** It details our approach, methodology, and process in implementing the algorithm described in the paper **Hierarchical Planning Through Goal-Conditioned Reinforcement Learning** (3).

The paper address the problem of applying reinforcement learning to long-horizon, temporally extended tasks. This is especially important in safety critical environments where exploration is risky and costly, this limits the agent's ability to explore freely, which is often necessary for effective RL training.

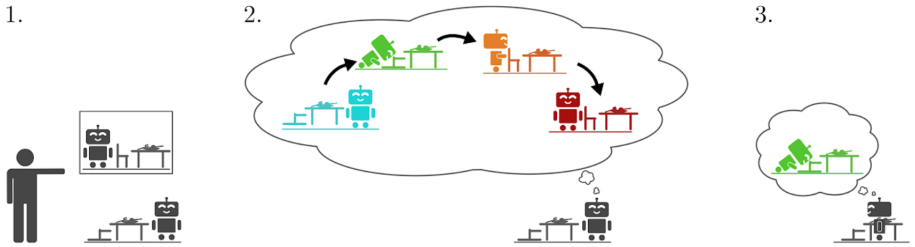

Figure 1: Goal Conditioned RL

The paper also addresses the challenges associated with offline RL, where policies are trained from static datasets rather than live interactions with the environment. While offline RL is safer, it has limitations, particularly related to distributional shift and out-of-distribution (OOD) goals. Distributional shift occurs when the policy encounters situations during deployment that were not adequately represented in the offline training data, which can lead to poor or unpredictable performance. OOD goals, or goals that fall outside the distribution seen in training, pose a major problem because the policy may fail to generalize or may act irrationally when encountering these unfamiliar states.

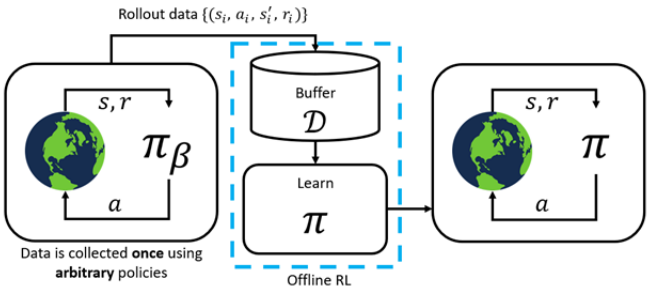

Figure 2: Offline Reinforcement Learning

To tackle these issues, the paper proposes a hierarchical framework that combines a high-level planning module with a low-level goal-conditioned RL policy. This structure enables the agent to manage long-term tasks effectively, while specific techniques like noisy, unreachable goal generation help the high-level planner recognize and avoid OOD goals, thereby enhancing safety and reliability in real-world applications.

2 Framework

The paper introduces the **HiGOC** framework, a goal-conditioned planning approach designed specifically for offline reinforcement learning applications. While it incorporates several foundational elements from the **LEAP** algorithm (4) on hierarchical reinforcement learning, HiGOC extends this framework to address offline RL. The main architectural elements include:

1. **High Level Planner:** This module searches for **"Sub-Goals"**(Temporally extended tasks) for the Low level Planner. These sub-goals help the agent to plan effectively in temporaly extended enviroment.

2. **Low Level Agent:** This module determines the low level policy of the agent. It takes the high level planner's sub-goals and breaks these sub-goals down to primitive actions that can be performed in the environment

3. **CVAE:** A conditional variational auto-encoder is used to model the latent state space. This is integral to the high level planner as well as the training of the low level agent.

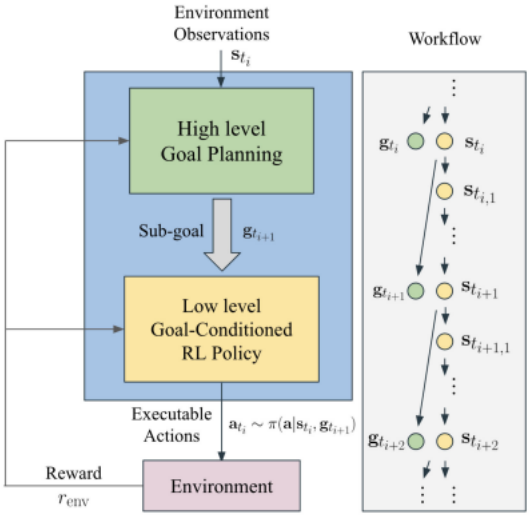

Figure 3: **HiGOC** framework

2.1 CVAE

CVAE (Conditional Variational Autoencoder) is a generative model consisting of an encoder and a decoder as shown in the architecture below:

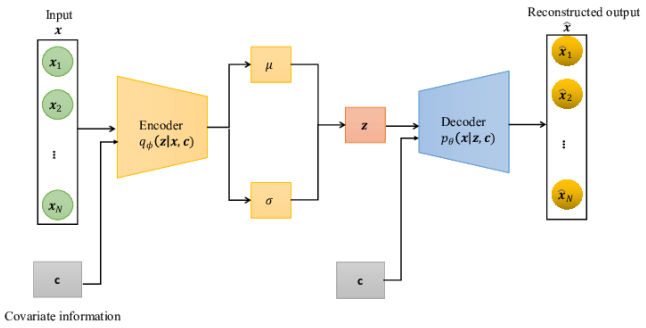

Figure 4: CVAE architecture

The encoder learns a latent space for the input conditioned on the input and covariate information while the decoder learns to reconstruct a vector conditioned on the latent vector and a possibly different covariate. In our use case in the framework, the details of the network are:

- **Encoder Module $\mathcal{E}(z \mid g_t, g_{t+1})$:** Learns the latent state space for subgoals conditioned on a goal and the next subgoal
- **Decoder Module $\mathcal{D}(g_{t+1} \mid g_t, z)$:** Learns to reconstruct the next subgoal conditioned on the latent vector and previous subgoal

The training of the CVAE involves the use of the following loss function:

$$\mathcal{L}_{CVAE} = \underbrace{|\hat{g}_{t+1} - g_{t+1}|}_{\text{Reconstruction Loss}} + \underbrace{D_{KL}(z \parallel \mathcal{N}(0,1))}_{\text{KL Divergence}} \tag{1}$$

The reconstruction loss forces the model to learn accurate state representations while the KL divergence loss allows for convergence to a simple distribution which will aid us greatly in the process of subgoal planning as we shall see later. We also see that the training of this network is independent of the rest of the model and is done **once per environment** in a **semi-supervised manner using the replay buffer**.

2.2 Low Level Agent

The low level agent is the policy which will interact with the environment using primitives. It performs actions to achieve the subgoals which are dictated by the high level planner. The value function used for training the planner is also valuable as it is used in the high level planner.

2.2.1 Reward Relabeling and Noisy Sampling:

As the value function used in the training of the low level agent is also used for planning subgoals, we must:

- Ensure that goals which are not reachable have a low value in the value function, this can be done through reward relabeling:

$$r_{TDM}(s_\tau, g_t, \tau) = -\delta(\tau = t)\, d(s_\tau, g_t) \tag{2}$$

  - $r_{TDM}(s_\tau, g_t, \tau)$: Temporal Difference Model (TDM) reward function at state s_τ, goal g_t, and time τ.
  - s_τ: State at time τ.
  - g_t: Target goal state to be reached by time t.
  - τ: Current time step in the trajectory.
  - t: Target time step to reach g_t.
  - $\delta(\tau = t)$: Indicator function, equals 1 if $\tau = t$ (reward only applies at time t), and 0 otherwise.
  - $d(s_\tau, g_t)$: Distance metric between current state s_τ and goal g_t.

- Make sure the agent visits these OOD subgoals so that the value function can reflect that they give low values, this is done by Noisy sampling:

$$g_{t_i+1} = D_\nu \left(E_\mu(s^k_\tau) + \epsilon, s^k_{t_i-N} \right) \tag{3}$$

  - g_{t_i+1}: Goal generated for the next time step $t_i + 1$.
  - D_ν: Decoder function, parameterized by ν.
  - E_μ: Encoder function, parameterized by μ.
  - s^k_τ: State at time τ in the k-th trajectory.
  - ϵ: Noise added to the encoded state to encourage exploration.
  - $s^k_{t_i-N}$: Previous subgoal (at time $t_i - N$) used as context.

2.2.2 Low Level Training Algorithm:

Algorithm 1 Training Procedure of HiGOC Low Level Agent and CVAE

Initialize: A Q-network Q_θ parametrized by θ, a target network $\bar{Q}_\theta = Q_\theta$ parametrized by $\bar{\theta}$, a policy network π_φ parametrized by φ, an encoder $\mathcal{E}$ and a decoder $\mathcal{D}$ for the CVAE, a training dataset replay buffer $\mathcal{R}$

for *step* $c \in range(0, C)$ **do**
  Sample a batch of b states s from the replay buffer $\mathcal{R}$
  Train $\mathcal{D}$ and $\mathcal{E}$ as per CVAE objective (1)
end

for *step* $m \in range(0, M)$ **do**
  Sample a pair $\left(s^k_{t_i,j}, a^k_{t_i,j}, s^k_{t_i,j+1}, r^k_{env,t_i,j} \right)$
  Sample a future state s^k_τ within the k-th trajectory and add noise with a probability η to obtain the goal as per equation (3)
  Relabel the reward following equation (2)
  Update θ, φ as per low level agent
  if m mod *target_update* $== 0$ **then**
    Soft update the target network: $\bar{\theta}_m \leftarrow (1-\tau)\bar{\theta}_{m-1} + \tau\theta_{m-1}$
  end
end

2.3 High Level Planner

The high level planner plans the subgoals that are followed by the low level agent. The **planning is carried out in the latent state space** by solving an optimization problem to avoid the curse of dimensionality. We break the discussion into two parts - the optimization objective and planning method.

2.3.1 Optimization Objectives

The original objective for optimization from the paper is:

$$\mathcal{L}_{HiGOC} = V^\pi \left(s_0, g_{t_1} \right) + \sum_{i=2}^{H-1} V^\pi \left(g_{t_{i-1}}, g_{t_i} \right) + V^\pi \left(g_{t_{H-1}}, g \right) - \lambda \log \left(p(z) \right) \tag{4}$$

where,

- $V^{\pi}\left(g_{t_{i-1}}, g_{t_i}\right)$ is the value of state $g_{t_{i-1}}$ conditioned on subgoal g_{t_i} under policy π
- $\lambda \log\left(p(\mathbf{z})\right)$ is the weighted log likelihood of the latent vectors used to form the sequence of subgoals
- s_0 is the starting state for the planning sequence
- g is the final goal for the agent

While this objective does lead to effective learning for the model, we propose an alternative objective for planning. We refer the framework using this objective as **distance-HiGOC** for reasons that will become evident. We withhold the motivation for this proposal until after presenting the results for the original framework for better understanding:

$$\mathcal{L}^{j}_{distance-HiGOC} = V_d^{\pi}\left(s_0, g_{t_1}\right) + \sum_{i=2}^{H_j - 1} V_d^{\pi}\left(g_{t_{i-1}}, g_{t_i}\right) + V_d^{\pi}\left(g_{t_{H_j-1}}, g\right) - \lambda \log\left(p(z)\right) \quad (5)$$

$$V_d^{\pi}\left(g_{t_{i-1}}, g_{t_i}\right) = V_d^{\pi}\left(g_{t_{i-1}}, g_{t_i}\right) + \alpha\left(|g_{t_i} - g|\right) \quad (6)$$

where,

- $\alpha\left(|g_{t_i} - g|\right)$ is the weighted distance between a subgoal and the final goal
- The index j indicates the number of times the planning has occurred prior to this time, $H_j = H - j$

2.3.2 Optimization Method (CEM) and Planning Sequence

In the framework explored here, we exploit the CEM (Cross Entropy Maximization) optimization method. This is a gradient free optimization technique which is used to plan in the latent space to get a new subgoal per the following pseudocode:

Algorithm 2 CEM Optimization and Planning Sequence

$\mu \leftarrow 0$, $\Sigma \leftarrow I$, samples $\leftarrow$ [[ ]], objectives $\leftarrow$ [ ]
Input: H, ρ, K, $\mathcal{L}$, g_{t-1}
for $n = 1$ *to* N **do**
  for $k = 1$ *to* K **do**
    for $h = 1$ *to* H **do**
      samples[k][h] $\leftarrow \mathcal{N}(\mu, \Sigma)$
    end
    objectives[k] $\leftarrow \mathcal{L}(\mathcal{D}(samples[k]))$
  end
  Select top ρK samples with highest objective
  Update μ and Σ using top ρK samples
end
$z \leftarrow \mathcal{N}(\mu, \Sigma)$, $g_t \leftarrow \mathcal{D}(z, g_{t-1})$
return g_t

where,

- H is constant in the original objective and decreases with each planning in our proposal
- $\mathcal{L}$ is the appropriate objective depending on whether HiGOC or distance-HiGOC is used
- $\mathcal{D}$ is the decoder of the CVAE

3 Implementation Details

In this section we go through the various details of our implementation of the **HiGOC** framework.

3.1 Environments:

To test the algorithm we used the **Antmaze** environments, the Antmaze environment presents a navigation domain with a complex 8-DoF Ant quadruped robot. For training we used the datasets that are available in **d4rl**. In particular we used two environments:

1. **Umaze:** In this environment the ant has a fixed starting and finish points.
2. **Umaze-diverse:** In this environment the ant has random start and finishing points.

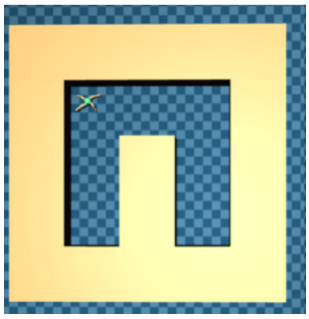

Figure 5: antmaze U-maze

In both environments the agent gets a sparse reward only on reaching the goal state. The completion of this is judged by the distance of the agent from the goal state being within an acceptable threshold. A maximum of 700 timesteps are allowed for the completion.

3.2 Low Level Agents:

In our project, we abandon the original CQL-SAC agent in favor of others. We describe in detail why we choose this, how we choose to evaluate our model given this change and how we narrowed the final choices for our agents.

3.2.1 Why not CQL-SAC

We chose not to proceed with the original CQL-SAC agent since:

- Original CQL implementation is not reproducible on Antmaze tasks with the authors themselves reccomending an alternate implementation
- The author recommended implementation used tricks to achieve convergence, but we were unsure of how that would affect the HiGOC framework

Therefore, we choose to instead use alternate agents since the effect of the high level planner can be judged by an **improvement in performance over the base agent**.

3.2.2 IQL

We choose the Intrinsic Q-Learning (IQL) agent (2) for the low level agent for three main reasons:

- Fast convergence which enables us to easily train the model with limited resources
- Better performance than CQL-SAC on most tasks
- Verified and reproducible performance on benchmarks with hyperparameter choices

The pseudocode for IQL is as follows:

Algorithm 3 Implicit Q-learning

Initialize parameters ψ, θ, $\hat{\theta}$, ϕ

TD learning:
for *each gradient step* **do**
$\psi \leftarrow \psi - \lambda_V \nabla_\psi L_V(\psi)$
$\theta \leftarrow \theta - \lambda_Q \nabla_\theta L_Q(\theta)$
$\hat{\theta} \leftarrow (1-\alpha)\hat{\theta} + \alpha\theta$
end

Policy extraction:
for *each gradient step* **do**
$\phi \leftarrow \phi - \lambda_\pi \nabla_\phi L_\pi(\phi)$
end

where,

- ψ: Parameters for the value network.
- θ: Parameters for the Q-network.
- $\hat{\theta}$: Target parameters for the Q-network.
- ϕ: Parameters for the policy network.

3.2.3 TD3+BC

We also chose to experiment with TD3+BC (1) as a low level agent. The main reason for this was the under-performance of IQL on Umaze-Diverse. This makes it tough to see the effectiveness of high level planning on this environment with IQL. Using this agent has the added benefit of observing how well the planner can make up for the simplicity of the low level agent. The scores for the two agents are shown in table 1 below:

Method	**IQL**	**TD3+BC**
Umaze	87.5	78.6
Umaze-Diverse	62.2	71.4

Table 1: Performance of IQL and TD3+BC in our environments

Hyperparameter	Value	Description
eval_freq	100	Frequency (in time steps) at which the evaluation is performed
n_episodes	1	Number of episodes to run during each evaluation
max_timesteps	40,000	Maximum number of time steps to run the environment
buffer_size	2,000,000	Size of the replay buffer
batch_size	2048	Batch size used for all networks
discount	0.99	Discount factor for future rewards
tau	0.005	Target network update rate
beta	3.0	Inverse temperature; controls trade-off between behavior cloning (small beta) and Q-maximization (large beta)
iql_tau	0.9	Coefficient for asymmetric loss
iql_deterministic	False	Flag indicating whether to use a deterministic actor
normalize	False	Flag indicating whether to normalize states
normalize_reward	False	Flag indicating whether to normalize rewards
vf_lr	0.0003	Learning rate for the value function
qf_lr	0.0003	Learning rate for the critic
actor_lr	0.0003	Learning rate for the actor

Table 2: IQL Hyperparameters

Hyperparameter	Value	Description
eval_freq	100	Frequency (in time steps) at which the evaluation is performed
n_episodes	1	Number of episodes to run during each evaluation
max_timesteps	40,000	Maximum number of time steps to run the environment
TD3-specific Hyperparameters		
buffer_size	2,000,000	Size of the replay buffer
batch_size	256	Batch size used for all networks
discount	0.99	Discount factor for future rewards
expl_noise	0.1	Std of Gaussian exploration noise
tau	0.005	Target network update rate
policy_noise	0.2	Noise added to target actor during critic update
noise_clip	0.5	Range to clip target actor noise
policy_freq	2	Frequency of delayed actor updates
TD3+BC-specific Hyperparameters		
alpha	2.5	Coefficient for Q function in actor loss
critic_lr	0.0003	Learning rate for the critic
actor_lr	0.0003	Learning rate for the actor
normalize	False	Flag indicating whether to normalize states
normalize_reward	False	Flag indicating whether to normalize rewards

Table 3: TD3+BC Hyperparameters

3.3 CVAE Network

The detailed outline of the CVAE is shown below in Figure 6. The training of the network is done using Adam optimizer with default β_1 and β_2, along with cosine annealing scheduler.

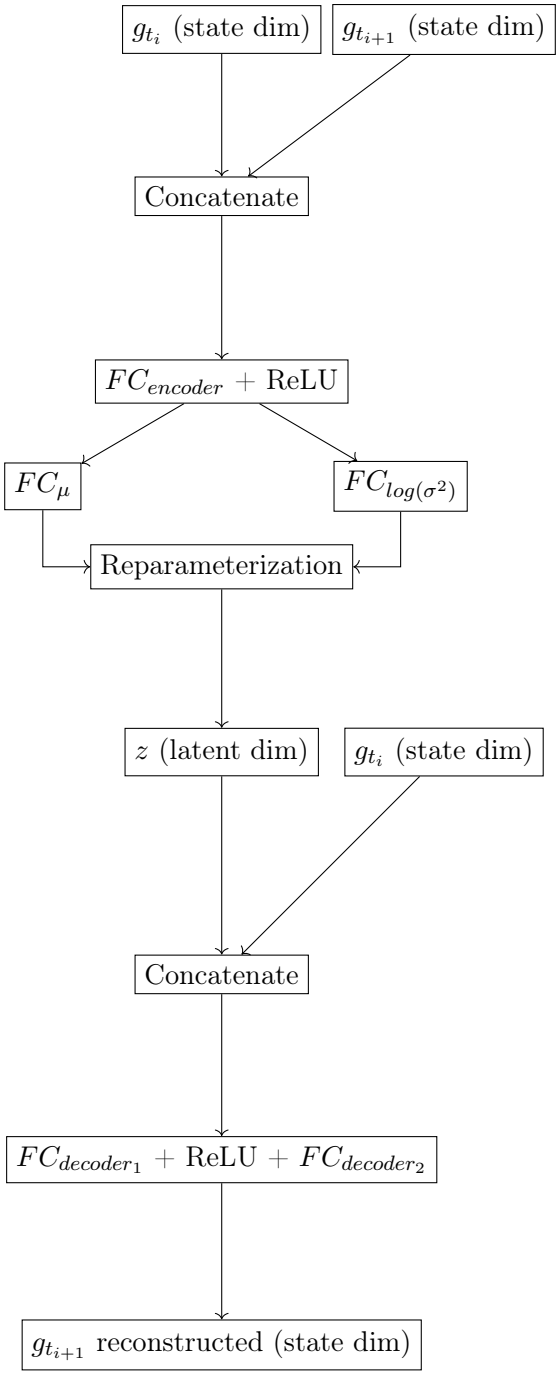

Figure 6: Architecture of the CVAE with Encoder, Reparameterization, and Decoder

We use the following shapes for the layers:

Layer	Size (number of neurons)	Parameter Name in Code
$FC_{encoder}$	64	hidden_size
FC_{μ}, $FC_{\log(\sigma^2)}$	8	latent_dim
$FC_{decoder_1}$	64	hidden_size
$FC_{decoder_2}$	environment specific	state_dim

The other hyperparameters used in the training of the CVAE are:

Hyperparameter	value	Parameter Name in Code
learning rate	0.01	lr
number of epochs	30	epochs
batch size	2048	batch_size

3.4 High Level Planner

The parameters of importance in the high level planning are shown in table 4 below. There is one extra parameter in distance-HiGOC which has been shown in the same table as well.

Hyperparameter	Value (Both Planners)	distance-HiGOC	Param Name in Code
Planning samples	500	-	n_samples
Percentage elite samples	0.25	-	elite_percentage
Number of subgoals	10	-	num_subgoals
Number of CEM iterations	10	-	num_iters
Log probs weight λ (4), (5)	0.1	-	-
Distance weight α (6)	-	0.1	-

Table 4: Hyperparameters for HiGOC and distance-HiGOC planners

4 Results

We performed experiments on the two environments described above across the two aforementioned versions of the low level agent. For this section, all results use the **HiGOC planner**. In this section we showcase the results from these experiments.

We present the training curves and final scores across the two environments with the aforementioned hyperparameters. The scores are normalized across 5 evaluation episodes on 5 random seeds common for the two agents.

Environment	IQL	HiGOC-IQL	TD3+BC	HiGOC-TD3+BC
Umaze-v2	87.5	88	78.6	88
Umaze-Diverse-v0	62.2	72	71.4	80

Table 5: Performance of various low level agents and their HiGOC counterparts

Next, we present the reward curves during the training process:

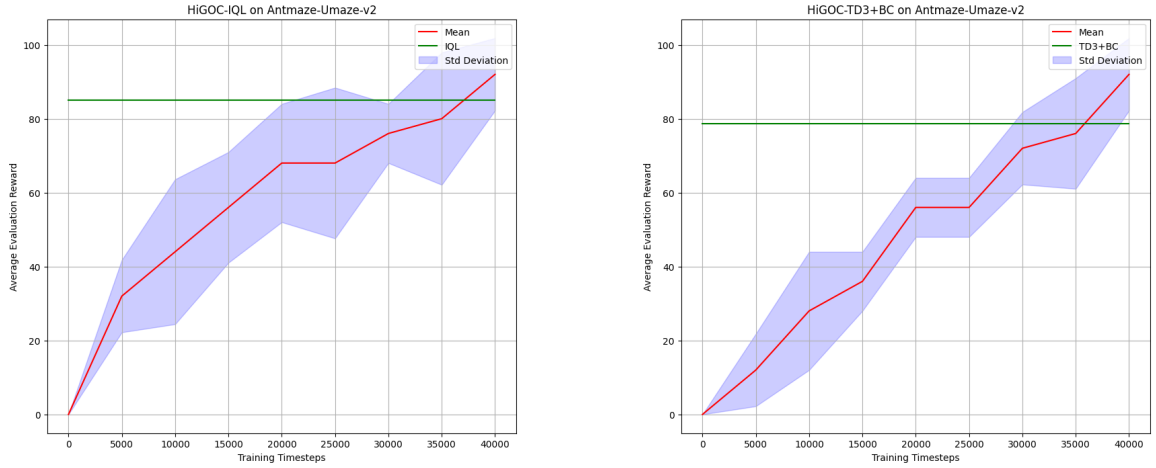

Figure 7: Comparison of low level agents on Umaze-v2

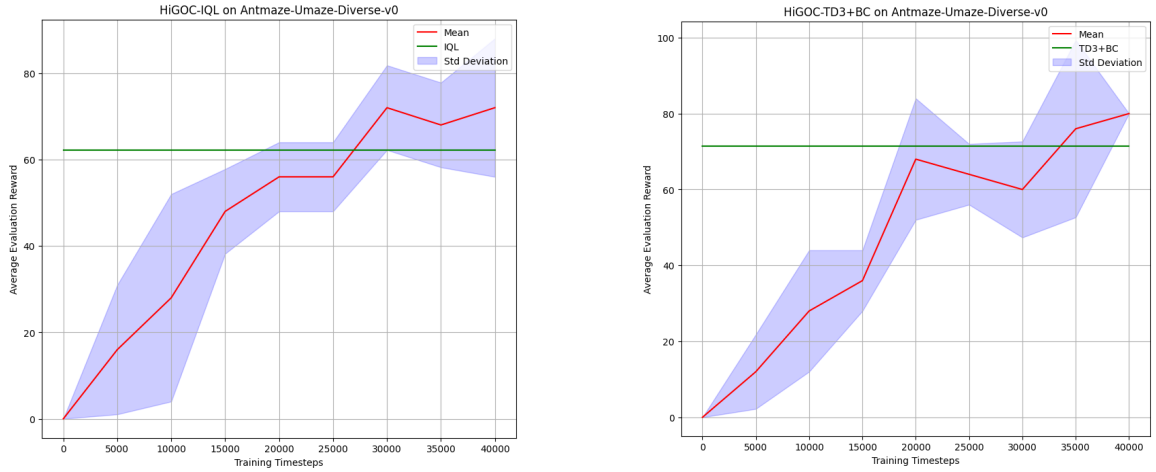

Figure 8: Comparison of low level agents on Umaze-Diverse-v0

Finally, we have the trajectory plots, which show the path taken by the agent and the planned subgoals.

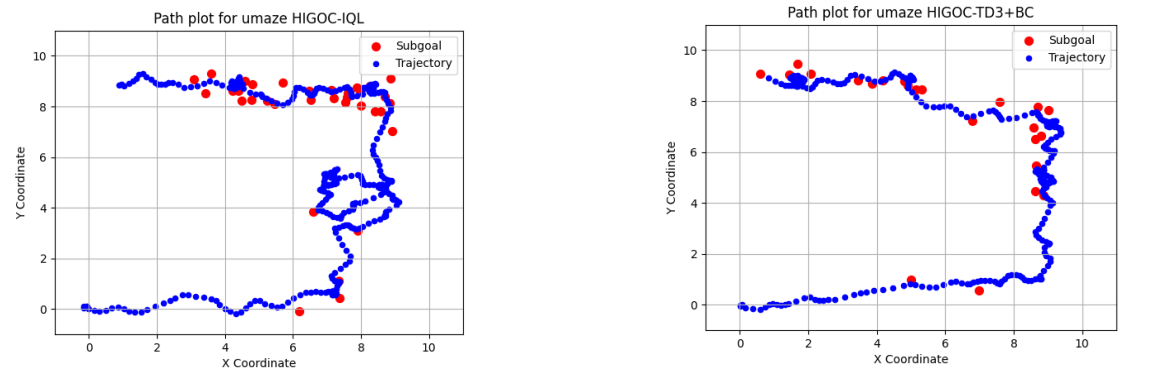

Figure 9: The path plots on Umaze-v2

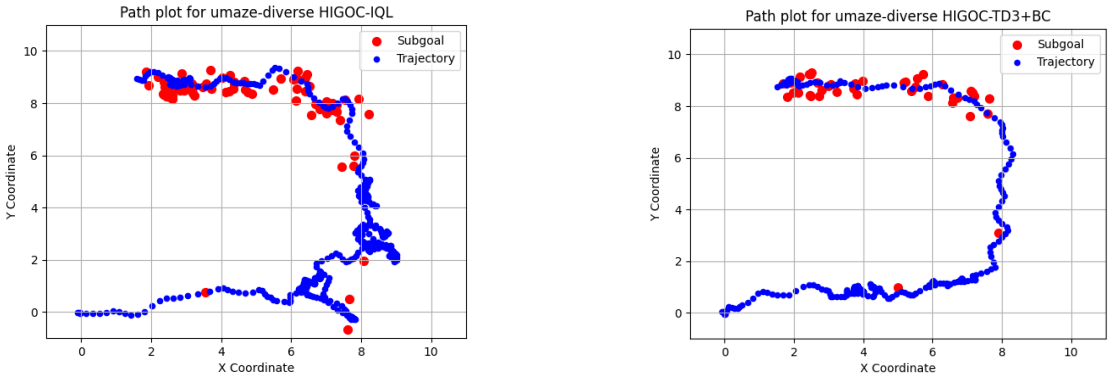

Figure 10: The path plots on Umaze-diverse-v0

5 Inferences

We make the following major observations from the reward curves and the path plots of the two kinds of agents:

1. **Better Performance than Base Agent:** The **HiGOC** high level planner lends to better performance over just the low-level algorithms on their own. Both IQL and TD3 significantly outperform the low level agents on their own.

2. **Subgoal Clustering:** The subgoals tend to cluster towards the end of the trajectory this is due to the fact that the environment has sparse rewards and hence the reward is concentrated at the end goal resulting in a higher value for the value function towards the end, this in turn makes the high-level planner choose subgoals closer to the end.

3. **OOD querying:** The subgoals suggested by the high level planner are generally within the maze,this shows that the the r_{TDM} reward relabeling and noisy subgoal sampling help in reducing out of distribution querying.

4. **Effect of Low Level Agent:** While the results indicate that the performance of the model is directly influenced by the low level agent, it is possible to greatly improve upon the base agent with planning as evidenced by table 5.

6 Improvements

Sub-goal clustering is something we want to avoid as this leads to excessive planning towards the end which slows down the agent and could lead to inaccurate subgoals. To mitigate this we introduce **Distance HiGOC**.

- **Distance function:** A distance function is added to value given by the value function during planning. What this does is that it gives states far away from the endgoal a higher effective value thus leading to more distributed subgoal planning:

$$d(s, g) = \sqrt{(s_x - g_x)^2 + (s_y - g_y)^2} \tag{7}$$

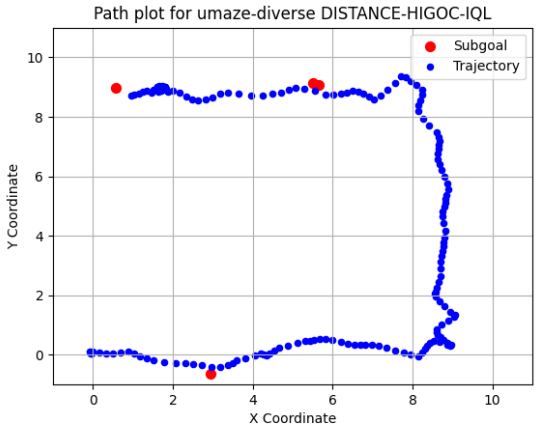

Figure 11: Path plot with improved High level planner

- **Improvements in inference time:** Due to lesser subgoal planning the episodic runtime of the **Distance HiGOC IQL** agent is much faster while having better performance.

Metric	Distance HiGOC-IQL	HiGOC-IQL
Score	73.6	72
Runtime per episode	15 seconds	180 seconds

Table 6: Comparison of Distance HiGOC-IQL and HiGOC-IQL runtimes

7 Conclusion

Through this project we were able to learn much about Goal conditioned RL and offline RL, we were able to implement the entire algorithm of the paper (3) while also adding useful improvements, in particular we were able to implement:

- **Codebase for HiGOC**:
  In this project, we have fully coded from scratch the HiGOC planner from the paper. This was a paper that does not have a codebase and we hope this will aid future work in goal-conditioned RL.
- **Effect of Low Level Agent**:
  We investigated the effect that the low-level agent has on the performance of the entire model. We showed that the more effective the agent is at learning the value and action space, the easier it is to get the model to converge.
- **Faster Planner**:
  We also propose a modified high-level planner at evaluation time, which runs significantly (more than 10 times) faster than the original from the paper and which also spreads out subgoals in sparse reward environments, helping the agent reach the goal faster.

8 Acknowledgments

We would like to express our gratitude to Professor B.Ravindran sir and the teaching assistants for their guidance throughout the project and the course. We would like to extend special thanks to Returaj sir for helping us better understand the framework and getting a clearer picture of how to proceed with the implementation.

References

[1] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[2] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. 2021.

[3] Jinning Li, Chen Tang, Masayoshi Tomizuka, and Wei Zhan. Hierarchical planning through goal-conditioned offline reinforcement learning. *IEEE Robotics and Automation Letters*, October 2022.

[4] Soroush Nasiriany, Vitchyr H. Pong, Steven Lin, and Sergey Levine. Planning with goal-conditioned policies. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS)*, November 2019.