NAVIGATION WITH QPHIL: QUANTIZING PLANNER FOR HIERARCHICAL IMPLICIT Q-LEARNING

Alexi Canesse[*,†,1], **Mathieu Petitbois**[*,2], **Ludovic Denoyer**[3], **Sylvain Lamprier**[4], **Rémy Portelas**[2]
[1]ENS Lyon [2]Ubisoft La Forge [3]H Company [4]University of Angers

ABSTRACT

Offline Reinforcement Learning (RL) has emerged as a powerful alternative to imitation learning for behavior modeling in various domains, particularly in complex long range navigation tasks. An existing challenge with Offline RL is the signal-to-noise ratio, i.e. how to mitigate incorrect policy updates due to errors in value estimates. Towards this, multiple works have demonstrated the advantage of hierarchical offline RL methods, which decouples high-level path planning from low-level path following. In this work, we present a novel hierarchical transformer-based approach leveraging a learned quantizer of the state space to tackle long horizon navigation tasks. This quantization enables the training of a simpler zone-conditioned low-level policy and simplifies planning, which is reduced to discrete autoregressive prediction. Among other benefits, zone-level reasoning in planning enables explicit trajectory stitching rather than implicit stitching based on noisy value function estimates. By combining this transformer-based planner with recent advancements in offline RL, our proposed approach achieves state-of-the-art results in complex long-distance navigation environments. Project page: `https://mathieu-petitbois.github.io/projects/qphil/`

1 INTRODUCTION

Navigation and locomotion in complex, long-horizon embodied environments is a long-standing challenge within Machine Learning (Kaelbling et al., 1996; Sutton & Barto, 2018). Operating non-trivial agents in such environments is critical in a wide range of real-world applications, such as in robotics (Eysenbach et al., 2019) or in the video game industry (Alonso et al., 2020). A core difficulty of navigation lies in solving long-horizon tasks that require intricate path planning (Hoang et al., 2021; Park et al., 2024). In the Reinforcement Learning (RL) setting (Sutton & Barto, 2018), traditional online Goal-Conditioned deep Reinforcement Learning (GCRL) methods often struggle with such long-horizon tasks because of the sparse nature of the reward signal, leading to hard exploration problems. Offline GCRL circumvents this exploration problem by leveraging large amounts of unlabeled and diverse demonstration data to learn policies through passive learning (Prudencio et al., 2023). A core advantage of offline RL compared to other forms of behavior extraction from datasets, for instance imitation learning, is the ability to improve over suboptimal datasets, e.g. by learning state value functions to bias the learned policy towards rewarding actions (Kostrikov et al., 2021). However, offline RL is not trivial to apply for long-horizon goal-reaching tasks, which often provide sparse reward signals, leading to a noisy value function which consequently hinders the performance of the policy. Part of this issue comes from low "signal-to-noise" ratio to learn the value function (Park et al., 2024). Because a suboptimal action can be corrected quickly in subsequent steps of a trajectory, its impact on the real value for faraway goals can be covered by the value prediction noise, which can lead to the learning of suboptimal behaviors for the policy.

Hierarchical architectures have shown considerable advantages in goal-conditioned navigation to solve such issues (Vezhnevets et al., 2017; Pertsch et al., 2021; Park et al., 2024). These approaches effectively decompose the problem into two distinct components: high-level path planning and low-level path following. Among these, Hierarchical Implicit Q-Learning (HIQL) (Park et al., 2024) has recently emerged as the state-of-the-art method. HIQL leverages a hierarchical structure to learn

[*]Denotes equal contribution. Correspondance to `mathieu.petitbois@ubisoft.com`
[†]Work done during an internship at Ubisoft La Forge

both a high-level policy for generating subgoals and a low-level policy for achieving these subgoals, all within an offline reinforcement learning framework.

Although introducing a hierarchical structure enhances performance by improving the signal-to-noise ratio at each level, it only partially alleviates the issue. For long-distance tasks, the signal-to-noise ratio still degrades during subgoal generation, which can result in a noisy high-level policy and, consequently, reduced performance. In this paper, we propose to shift the learning paradigm of the high-policy towards discrete space planning. Towards this, we first train a Vector Quantized Variational Autoencoder (VQ-VAE) (Van Den Oord et al., 2017) on the state-space from which we extract its quantized representation, namely leading to a clustering of the state space into what we propose to refer to as *landmarks*. We ensure the temporal consistency of the obtained landmarks through a contrastive regularization of the VQ-VAE loss. Then, we leverage a transformer architecture (Vaswani et al., 2017) to extract a discrete high-level policy, enabling consistent landmark-level planning in the discrete space representation. Finally, we train a combination of a low-level landmark-conditioned policy and a low-level goal conditioned policy to solve the subgoal and last-goal navigation tasks respectively.

Our main contribution is to introduce **Quantizing Planner for Hierarchical Implicit Learning (QPHIL)**, a novel approach that addresses long-range navigation by combining the strengths of VQ-VAE, transformers, and offline reinforcement learning. We first evaluate QPHIL on the established AntMaze navigation tasks (Fu et al., 2020; Jiang et al., 2022), then introduce a novel, more challenging variant, *AntMaze-Extreme*, along with two associated datasets, better suited to study long-distance navigation. In both settings we demonstrate that QPHIL matches or outperforms prior offline goal-conditioned RL methods, particularly in large-scale settings.

2 Related work

VQ-VAEs for reinforcement learning Vector Quantized Variational Autoencoders (Van Den Oord et al., 2017) have demonstrated their utility in reinforcement learning, particularly for offline setups. TAP (Jiang et al., 2022), SAQ (Luo et al., 2023) and L-MAP Luo et al. (2025) use a VQ-VAE to discretize continuous action spaces, mitigating the curse of dimensionality. In contrast, QPHIL uses a VQ-VAE to quantize states, simplifying waypoint generation in long-horizon tasks. These two approaches have different purposes and are complementary: action quantization focuses on simplifying policy search, while state quantization helps in task decomposition such as in (Hamed et al., 2024) where a VQ-VAE is used to define landmarks as states like in QPHIL but without a contrastive loss and in the online setting. Within goal-conditioned reinforcement learning, several works used VQ-VAE to encode observations into discrete (sub)goals. (Lee et al., 2024a) learn quantized goals to simplify curriculum learning. (Islam et al., 2022) also use quantization to map (sub)goals into discrete and factorized representations to efficiently handle novel goals at test time. Both these works focus on online goal-conditioned reinforcement learning while QPHIL tackles the offline learning setting. (Kujanpää et al., 2023) also uses VQ-VAE to generate discrete subgoals, but while they rely on a finite subgoal set derived from offline learning, QPHIL identifies subgoals directly through the VQ-VAE. To the best of our knowledge, VQ-VAE has never been applied for discrete state planning in offline GCRL settings.

Hierarchical offline learning Leveraging a hierarchical structure to decompose and simplify sequential decision-making problems is a long-standing idea and subject of research within machine learning (Schmidhuber, 1991; Sutton et al., 1999). Recently, multiple successful works renewed the interest of the research community to these models, both for online (Vezhnevets et al., 2017; Pertsch et al., 2021; Kim et al., 2021; Fang et al., 2022) and offline reinforcement learning (Nachum et al., 2018b; Ajay et al., 2020; Rao et al., 2021; Rosete-Beas et al., 2023; Yang et al., 2023; Shin & Kim, 2023). Among these, Hierarchical Implicit Q-Learning (HIQL) from (Park et al., 2024) has emerged as the state-of-the-art method for offline goal-conditioned RL. Most methods differ in how they represent and use subgoals; for instance, (Nachum et al., 2018a) compare different subgoal representations, while (Ajay et al., 2020) focus on detecting and combining primitive behaviors. While all aforementioned works focus on continuous state representations, QPHIL leverages the advantage of discrete state representations to simplify long-distance navigation.

Planning Hierarchical planning methods are a natural complement to RL. For instance, HIPS (Kujanpää et al., 2023) learns to segment trajectories, generating subgoals of varying lengths to facilitate

adaptive planning. HIPS-ε (Jiang et al., 2023; Kujanpää et al., 2024) introduces hybrid hierarchical methods and offers completeness guarantees, but their method is only usable with a discrete state space. (Li et al., 2022) proposes a method combining a high-level planner, based on a VAE, with a low-level offline RL policy. The VAE serves as the planner by leveraging the low-level policy's value function. QPHIL, on the other hand, utilizes a simpler subgoal planning approach directly tied to a discrete state-space via VQ-VAE.

Transformers for (hierarchical) offline RL Transformers have recently found application in hierarchical setups to solve RL problems. In (Correia & Alexandre, 2023), transformers are used for hierarchical subgoal sampling, where a high-level transformer generates subgoals for a low-level transformer responsible for action selection. Similarly, (Badrinath et al., 2024) combine a Decision Transformer (DT) from (Chen et al., 2021) with a waypoint-generation network. (Ma et al., 2024) present a hierarchical transformer-based approach outperforming both aforementioned methods. (Li et al., 2022; Ma et al., 2024) are extensions of DT that allows stitching, a known issue with Decision Transformer (Fujimoto & Gu, 2021; Emmons et al., 2021; Kostrikov et al., 2021; Yamagata et al., 2023; Xiao et al., 2023). Their approach, Goal-prompted Autotuned Decision Transformer (G-ADT) (Ma et al., 2024), is able to efficiently integrate an offline RL learning scheme similar to HIQL. G-ADT uses a low-level transformer and a simple high level subgoal policy, while we focus on the opposite scenario: our transformer is used to generate a plan of subgoal, which are then followed by a fully connected neural network. G-ADT uses an offline reinforcement learning objective to train its high-level policy while we rely on a simpler imitation learning objective.

Sequence generation (Lee et al., 2024b) explore sequence generation using tokens learned via a VQ-VAE, much like QPHIL; but their work does not extend to reinforcement learning. In RL, sequence models have been used for various components (Bakker, 2001; Heess et al., 2015; Chiappa et al., 2017; Parisotto et al., 2020; Kumar et al., 2020). However, none of these approaches integrate planning over a discrete representation, which is central to our method. Sequence generation also plays a key role in offline RL, where it is helping to prevent out-of-distribution actions (Fujimoto et al., 2019; Kumar et al., 2019; Ghasemipour et al., 2021). Trajectory Transformer (Janner et al., 2021) completely treats offline RL as a sequence generation problem. They use a transformer to perform planning using Beam-Search and an IQL-like value function. Contrary to QPHIL, they use a simple dimension-wise uniform or quantile discretization.

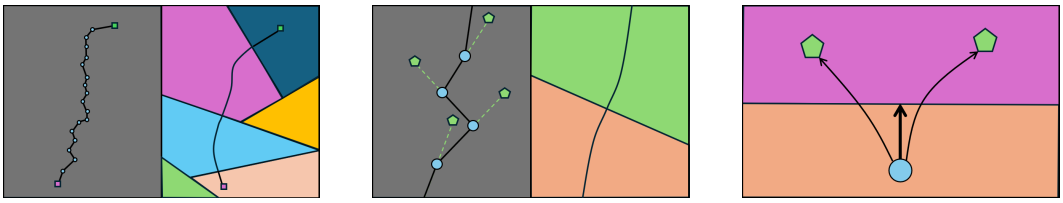
Spatial zone learning and skill discovery Zone-based spatial navigation and skill discovery are additional techniques that overlap with hierarchical RL and QPHIL. For instance in the context of online RL, (Kamienny et al., 2022) proposes to discover a set of "easy-to-learn" short policies, each corresponding to a given skill defined as an area of the state space, which can be eventually composed for the task at hand. Using a VQ-VAE, (Mazzaglia et al., 2022) define skills that are used to influence exploration in online policies. In a closer context with a known state space, (Gao et al., 2023) segments space into zones and uses these segments to facilitate goal-reaching tasks, but their method is restricted to vision environments. Similarly, (Hausman et al., 2017) uses a multi-modal imitation learning approach to discover and segment skills from unstructured demonstrations, which is close to the discrete landmark generation in QPHIL. In skill decision transformer (Sudhakaran & Risi, 2023), a transformer is used to predict actions from latent variables discovered by a VQ-VAE. While their approach is close to QPHIL, it differs by not using contrastive loss and not planning subgoals.

3 PRELIMINARIES

3.1 OFFLINE GOAL CONDITIONED REINFORCEMENT LEARNING

We frame our work in the context of offline goal-conditioned Reinforcement Learning (offline GCRL), which involves training an agent to interact with an environment to reach specific goals, without getting access to the environment itself at train time. It is typically modeled as a Markov Decision Process (MDP) defined by a tuple $(\mathcal{S}, \mathcal{A}, p, \mu, r)$, a dataset of demonstration trajectories $\mathcal{D}$ and a given goal space $\mathcal{G}$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, $p(s_{t+1}|s_t, a_t) \in \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ the transition dynamics, $\mu \in \mathcal{P}(\mathcal{S})$ the initial state distribution and $r(s_t, g) \in \mathcal{S} \times \mathcal{G} \to \mathbb{R}$ the goal-conditioned reward function given the goal space $\mathcal{G}$. In GCRL, the reward is sparse as $r(s, g) = 1$ only when s reaches g and 0 otherwise. Hence, our objective is to leverage the dataset $\mathcal{D}$ composed of reward-free pre-recorded trajectoires $\tau = (s_0, a_0, ..., s_{T-1}, a_{T-1}, s_T)$ to learn the pol-

icy $\pi(a_t|s_t, g)$ to reach given goals $g \in \mathcal{G}$, by maximizing the expected cumulative reward, or return, $J(\pi)$, which can be expressed as: $J(\pi) = \mathbb{E}_{\substack{g\sim d_g \\ \tau\sim d_\pi}}\left[\sum_{t=0}^{T}\gamma^t r(s_t, g)\right]$, where γ is a discount factor, d_g represents the goal distribution and d_π is the trajectory distribution defined by: $d_\pi(\tau, g) = \mu(s_0)\prod_{t=0}^{T}\pi(a_t|s_t, g)p(s_{t+1}|s_t, a_t)$ and represents the trajectory distribution when the policy is used. To study long range navigation, we consider in the following positional environments, where the goal space $\mathcal{G}$ can be computed as the set of positions coordinates $\mathcal{S}^p$.



(a) Simpler high level planning (b) Smoother low level targets (c) Easier target conditioning

Figure 1: **Motivations behind QPHIL** (a) QPHIL aims to simplify the planning of the subgoals by leveraging discrete tokens. (b) By doing so, QPHIL avoids the noisy high-frequency target subgoal updates by updating the subgoal of the low-level policy after each landmark traversal only. (c) The subgoal reaching tasks are less demanding in conditioning for the low policy as it corresponds to the reaching of an entier subzone instead of a precise subgoal.

3.2 Hierarchical Implicit Q-Learning (HIQL)

Because of the sparsity of the offline GCRL's reward signals, bad actions can be corrected by subsequent good actions, and good actions can be undermined by future bad actions in a trajectory. Hence, offline RL methods risk mislabeling bad actions as good ones, and vice versa. This leads to the learning of a noisy goal-conditioned value function: $\hat{V}(s_t, g) = V^*(s_t, g) + N(s_t, g)$ where V^* corresponds to the optimal value function and N corresponds to a noise. As the "signal-to-noise" ratio worsens for longer term goals, offline RL methods such as IQL (Kostrikov et al., 2021) struggle when the scale of the GCRL problem increases, leading to noisy advantages estimates for which the noise overtakes the signal. To alleviate this issue, HIQL (Park et al., 2024) first proposes to learn a noisy goal conditioned action-free value function, inspired from IQL (Kostrikov et al., 2021):

$$\mathcal{L}_V(\theta_V) = \mathbb{E}_{(s_t, s_{t+1}, g)\sim\mathcal{D}}\left[L_2^x(r(s_t, g) + \gamma V_{\theta_V}(s_{t+1}, g) - V_{\theta_V}(s_t, g))\right] \tag{1}$$

using an expectile loss $L_2^x(u) = |x - \mathbb{1}(u < 0)|u^2, x \in [0.5, 1)$ on the temporal difference of the value function, which aims at anticipating in-distribution sampling without querying the environment. This loss is then used to learn two policies with advantage weighted regression (AWR): $\pi^h(s_{t+k}|s_t, g)$ to generate subgoals on the path towards goal g and $\pi^l(a_t|s_t, g')$, with $g' \in \mathcal{G}$ a given subgoal, to generate actions to reach the subgoals, in a hierarchical manner. With this division, each policy profits from higher signal-to-noise ratios, as the low-policy only queries the value function for nearby subgoals $V(s_{t+1}, s_{t+k})$ and the high policy queries the value function for more distant goal $V(s_{t+k}, g)$.

3.3 Quantizing Planner for Hierarchial Implicit Learning (QPHIL)

If HIQL shows significant improvements in offline GCRL problems compared to previous flat-policy methods, its performance depends on the right choice of the subgoal step k. A high k would improve the high policy's signal-to-noise ratio by querying more diverse subgoals but at the cost of decreasing the signal-to-noise ratio of the low policy. Conversely, a low k would improve the low policy's signal-to-noise ratio by querying values for nearby goals but at the cost of the diversity of the high subgoals. Hence, HIQL might struggle for longer term goal reaching tasks as the low level performance imposes the choice of a sufficiently low k, which leads to high frequency noisy high subgoal targets for the low policy to follow. QPHIL proposes to mitigate this issue by shifting the learning paradigm of the high policy into planning in a discretized learned space representation (see Fig. 1).

4 QUANTIZING PLANNER FOR HIERARCHICAL IMPLICIT LEARNING

In this paper, we propose a new hierarchical goal conditionned offline RL algorithm: Quantizing Planner for Hierarchical Implicit Q-Learning (QPHIL). While current hierarchical methods rely heavily on continuous waypoint predictions, we propose to consider discrete subgoals, allowing to simplify the planning process. Instead of relying on precise coordinates, QPHIL identifies key landmarks to guide trajectory planning, much like how a road trip is described by cities or highways rather than specific geographic points. The algorithm detects these landmarks, creates landmark-based sequences, and formulates policies to navigate between them, ultimately reaching the target destination.

4.1 OVERALL DESIGN

QPHIL operates through four components: (1) a state quantizer q_{θ_q}, which divides the state space positions into a finite set of landmarks, (2) a plan generator $\pi^p_{\theta_p}$, which acts as a high-level policy to generate a sequence of landmarks to be reached given the final goal, and two low-level policy modules: (3) $\pi^{lm}_{\theta_{lm}}$, which targets state areas defined as landmarks and (4) $\pi^g_{\theta_g}$, which targets a specific goal state. At a timestep t, given a state history $(s_0, ..., s_t) \in \mathcal{S}_H$ (S_H being the set of state histories) and a target goal $g \in \mathcal{G}$:

(1) The quantizer $q_{\theta_q} : \mathcal{S}^p \to \Omega$ (with $\Omega = \{1, \dots, k\}$) is used to map the current state s_t positions into a set of k landmark indexes (or tokens). It serve as the building blocks for our planning strategy, while allowing for maintaining a certain landmark history $\bar{\tau}^{\omega}_{\leq n}$ by appending each new landmark to $\bar{\tau}^{\omega}_{\leq n-1}$, with n the current number of landmarks in the history.

(2) The planner $\pi^p_{\theta_p} : \Omega_H \times \Omega \to \Omega_P$ (with Ω_H the set of landmark histories and Ω_P the set of landmark plans), if needed, processes these discrete landmark histories to generate a coherent plan of landmarks that outlines the overall trajectory to be followed in the environment to reach the requested goal landmark $q_{\theta_q}(g)$. Given any history of tokens $\bar{\tau}^{\omega}_{\leq n} = (\omega_0, \cdots, \omega_n)_{n>0} \in \Omega_H$ and a targeted goal $g \in \mathcal{G}$, the plan generator produces a feasible sequence of tokens $(\omega_{n+1}, \cdots, \omega_{n+l})_{n>0,l>0} \in \Omega_P$ auto-regressively until $\omega_{n+l} = q_{\theta_q}(g)$ following:

$$\forall l > 0, \pi^p_{\theta_p}\left((\omega_{n+1}, \cdots, \omega_{n+l}) \,|\bar{\tau}^{\omega}_{\leq n}, q_{\theta_q}(g)\right) = \prod_{t=1}^{l} \pi^p_{\theta_p}\left(\omega_{n+t} \,|\bar{\tau}^{\omega}_{\leq n+t-1}, q_{\theta_q}(g)\right) \tag{2}$$

(3) The landmark policy $\pi^{lm}_{\theta_{lm}} : \mathcal{A} \times \mathcal{S} \times \Omega \to [0, 1]$ is then called if the goal landmark $q_{\theta_q}(g)$ is not yet reached to generate an action through $\pi^{lm}_{\theta_{lm}}(a_t|s_t, \omega_{n+1})$ to reach the next landmark ω_{n+1} in the plan given the current state s_t.

(4) The goal policy $\pi^g_{\theta_g} : \mathcal{A} \times \mathcal{S} \times \mathcal{G} \to [0, 1]$ is finally called once the goal landmark $q_{\theta_q}(g)$ as been attainted to reach the actual goal g given the current state s_t by generating actions through $\pi^g_{\theta_g}(a_t|s_t, g)$.

4.2 QUANTIZER

The first component to be trained is the quantizer q_{θ_q} based on the VQ-VAE (Van Den Oord et al., 2017) architecture. It is composed of an encoder $f^e_{\theta_e} : \mathcal{S}^p \to \mathbb{R}^d$ and a learned code-book $z = \mathbb{R}^{k \times d}$, which associates each landmark to a continuous embedding such that $\theta_q = [\theta_e, z]$. It is trained alongside a decoder $f^d_{\theta_d} : \mathbb{R}^d \to \mathcal{S}^p$. From these, q_{θ_q} is defined as:

$$\forall s^p \in \mathcal{S}^p, q_{\theta_q}(s^p) = \text{argmin}_{\omega \in \Omega} ||f^e_{\theta_e}(s^p) - z_\omega||^2_2 \tag{3}$$

Three losses are considered to train these components. First, the VQ-VAE learns a meaningful representation by considering a reconstruction loss, which ensures that states can be decoded from tokens they are projected into:

$$\mathcal{L}_{\text{recon}}(\theta_q, \theta_d) = \mathbb{E}_{s^p \sim \mathcal{D}}\left[||f^d_{\theta_d}(z_{q_{\theta_q}(s^p)}) - s^p||^2_2\right] \tag{4}$$

where $\mathcal{D}$ is the dataset. As the gradient flow stops in the previous loss due to the discrete projection performed, we need to consider an additional commitment loss to learn the encoder parameters:

$$\mathcal{L}_{\text{commit}}(\theta_q) = \mathbb{E}_{s^p \sim \mathcal{D}} \left[||f^e_{\theta_e}(s^p) - \text{sg}[z_{q_{\theta_q}(s^p))}]||_2^2 + \beta ||\text{sg}[f^e_{\theta_e}(s^p)] - z_{q_{\theta_q}(s^p))}||_2^2 \right] \tag{5}$$

where sg is the stop-gradient operator. The first term of this loss aims at attracting encoder outputs close to the code-book's codes. As explained by authors of the VQ-VAE architecture (Van Den Oord et al., 2017), the embedding space grows arbitrarily if the token embedding does not train as fast as the encoder parameters. The second term, weighted by a hyperparameter β, aims to prevent this issue by forcing the encodings to commit to a code-book's embedding. To introduce dynamics of the environment in the representation learned, we consider a third contrastive loss that incentivize temporally close states to be assigned the same tokens, while temporally distant states receive different tokens. To achieve this, we use the triplet margin loss (Balntas et al., 2016) applied to our setting:

$$\mathcal{L}_{\text{contrastive}}(\theta_e) = \mathbb{E}_{\substack{s^p_t \sim \mathcal{D} \\ k \sim \{-\delta, \dots, \delta\} \\ k' \sim \mathbb{Z} \setminus \{-\delta, \dots, \delta\}}} \left[\max \left\{ ||f^e_{\theta_e}(s^p_t), f^e_{\theta_e}(s^p_{t+k})||_2^2 - ||f^e_{\theta_e}(s^p_t), f^e_{\theta_e}(s^p_{t+k'})||_2^2, 0 \right\} \right] \tag{6}$$

where δ is the time window used to specify temporal closeness of states in demonstration trajectories. This loss is of crucial importance in navigation, for instance in settings with thin walls, where two states can be close in the input space while corresponding to very different situations. The tokenizer loss, used in training is then a linear combination of the three previous losses:

$$\mathcal{L}_{\text{quantizer}}(\theta_q, \theta_d) = \alpha_{\text{recon}} \mathcal{L}_{\text{recon}}(\theta_q, \theta_d) + \alpha_{\text{commit}} \mathcal{L}_{\text{commit}}(\theta_q) + \alpha_{\text{contrastive}} \mathcal{L}_{\text{contrastive}}(\theta_e) \tag{7}$$

4.3 PLANNER

Once the quantizer has been trained, each state from dataset trajectories $\tau = (s_0, ..., s_T)$ can be discretized leading to sequences of landmark tokens $\tau^\omega = (q_{\theta_q}(s^p_0), ..., q_{\theta_q}(s^p_T))$. Temporal consistency induces sub-sequences of repeated tokens corresponding to positions within a given region. By applying a simple post-processing step noted that removes consecutive repetitions of tokens from τ^ω, we obtain more concise sequences $\bar{\tau}^\omega$ that succinctly represent key zones to traverse in the correct order. For instance, a tokenized sequence such as "1 1 1 2 2 3 3 3 4 4" is simplified to "1 2 3 4", reflecting the core structure of the trajectory. Then, the planner $\pi^p_{\theta_p}$ is trained following a teacher forcing approach on compressed sequences, considering any future token of $\bar{\tau}^\omega$ as the associated training goal:

$$\mathcal{L}_{\text{planner}}(\theta_p) = -\mathbb{E}_{\tau \sim \mathcal{D}} \left[\mathbb{E}_{n < |\bar{\tau}^\omega| - 1} \left[\mathbb{E}_{\omega_g \in \bar{\tau}^\omega_{>n}} \left[\log \pi^p_{\theta_p}(\bar{\tau}^\omega_{n+1} \,|\bar{\tau}^\omega_{\leq n}, \omega_g)) \right] \right] \right] \tag{8}$$

with $|\bar{\tau}^\omega|$ the number of compressed tokens and $\bar{\tau}^\omega_{\leq n}$ (resp. $\bar{\tau}^\omega_{>n}$) the history (resp. future) of $\bar{\tau}^\omega$ at step n. We implement $\pi^p_{\theta_p}$ using a transformer architecture (Vaswani et al., 2017), where $\pi^p_{\theta_p}(\omega \,|\bar{\tau}^\omega_{\leq n}, \omega_g)$ is defined for any $\omega \in \Omega$ using a softmax on the outputs of the transformer. While an alternative would be to consider a markov assumption stating that $\pi^p_{\theta_p}(\cdot|\bar{\tau}^\omega_{\leq n}, \omega_g) = \pi^p_{\theta_p}(\cdot|\bar{\tau}^\omega_n, \omega_g)$, we claim that dependency on the full history of the sequence is useful to anticipate the next token to be reached, as it can be leveraged to deduce the precise location of the agent in large landmark areas (which is unknown during plan generation). Also, we perform a data augmentation process which relies on the opportunity of accurate trajectory stitching that our quantized space offers. As such, we augment every token sequence of $\mathcal{D}$ by stitching all other sub-sequences from the dataset that contains its last token, allowing for a higher state coverage.

4.4 LANDMARK POLICY

Independently from the planner, we train the landmark policy $\pi^{lm}_{\theta_{lm}}$ by first training a landmark conditioned value function $V^{lm}_{\phi_{lm}}$ learned with IVL (Park et al., 2024), a value function only IQL (Kostrikov et al., 2021), by minimizing the following loss:

$$\mathcal{L}_{V^{lm}}(\phi_{lm}) = \mathbb{E}_{(s_t, s_{t+1}, \omega)} \left[L_2^{x_{lm}}(r^\omega(s_t, \omega) + \gamma_{lm} V^{lm}_{\bar{\phi}_{lm}}(s_{t+1}, \omega) - V^{lm}_{\phi_{lm}}(s_t, \omega)) \right] \tag{9}$$

To get the training sample (s_t, s_{t+1}, ω), we sample two consecutive states (s_t, s_{t+1}) and with a probability $p_{current}$ we set $\omega = q_{\theta_q}(s_t^p)$, otherwise we sample ω uniformly among the next tokens in the trajectory if they exist: $\omega \sim \mathcal{U}(\{\omega \in \tau_{>t}^{\omega}, \omega \neq q_{\theta_q}(s_t^p)\})$. We set $r^{\omega}(s, \omega) = 0$ if $q_{\theta_q}(s^p) = \omega$ and -1 otherwise. After that, we can train $\pi_{\theta_{lm}}^{lm}$ through AWR by maximizing:

$$J_{\pi^{lm}}(\theta_{lm}) = \mathbb{E}_{(s_t, s_{t+1}, \omega)} \left[\exp(\beta_{lm} \cdot (V_{\phi_{lm}}^{lm}(s_{t+1}, \omega) - V_{\phi_{lm}}^{lm}(s_t, \omega)) \log \pi_{\theta_{lm}}^{lm}(a_t | s_t, \omega)) \right] \quad (10)$$

To get the training samples, we sample two consecutive states (s_t, s_{t+1}) and we set the target landmark ω as the first future token of the compressed sequence if it exists, otherwise the current token.

4.5 GOAL POLICY

Independently from the planner and the landmark policy, we train the goal policy $\pi_{\theta_g}^{g}$ using the GC-IVL (Park et al., 2024) algorithm. We train first a value function to minimize the following loss:

$$\mathcal{L}_{V^g}(\phi_g) = \mathbb{E}_{(s_t, s_{t+1}, g)} \left[L_2^{x_g}(r^g(s_t, g) + \gamma_g V_{\phi_g}^{g}(s_{t+1}, g) - V_{\phi_g}^{g}(s_t, g)) \right] \quad (11)$$

We get the training samples by sampling two consecutive states (s_t, s_{t+1}) and set g as the current state with probability $p_{current}$ or sample it uniformly among the future trajectory states with probability p_{future} and otherwise we set the goal by sampling a random state from the dataset. We set $r^{\omega}(s, g) = 0$ if $s = g$ and -1 otherwise. As for the policy, we perform again an AWR training by optimizing the following objective:

$$J_{\pi^g}(\theta_g) = \mathbb{E}_{(s_t, s_{t+1}, g)} \left[\exp(\beta_g \cdot (V_{\phi_g}^{g}(s_{t+1}, g) - V_{\phi_g}^{g}(s_t, g)) \log \pi_{\theta_g}^{g}(a_t | s_t, g)) \right] \quad (12)$$

The training samples are obtained by sampling two consecutive states (s_t, s_{t+1}) and we sample the goal g uniformly from the future states of the trajectory.

5 EXPERIMENTS

Our experiments aim to address the following questions:

1. Does QPHIL's architecture enable efficient long-term navigation?
2. What is the impact of the contrastive loss used for landmark learning?
3. What is the impact of the different losses when training the quantizer ?

5.1 EXPERIMENTAL SETUP

Scope QPHIL focuses on advancing hierarchical offline reinforcement learning for long-range navigation, a domain where state quantization and hierarchical planning address noise in long-horizon value estimates, a critical challenge in offline goal-conditioned RL. We thus evaluate QPHIL on the AntMaze suite, a well-suited benchmark for this scope, deferring orthogonal tasks like dexterous manipulation (e.g., Adroit) or locomotion (e.g., HalfCheetah) to future work, as these prioritize control dynamics over spatial path-planning.

Antmaze We measure QPHIL's performance through a set of the AntMaze environments of increasing sizes. The Antmaze suite is an established and challenging benchmask which calls for long-term planning benchmark as well as solving hard locomotion tasks which illustrates well QPHIL's scope. In AntMaze the agent controls an 8-DoF ant-shaped robot. Observations consist in a 29-dimension state vector (e.g. positions, torso coordinates and velocity, angles between leg parts). For original AntMaze environments we use datasets provided in the D4RL library (Fu et al., 2020) as well as additional datasets for even larger mazes, namely Antmaze-Ultra provided by (Jiang et al., 2022) and Antmaze-Extreme which we created. For each maze type (medium, large, ultra and extreme), we train on two types of datasets: "play" and "diverse", each containing 1000 (500 for extreme) trajectories of 1000 steps (2000 for extreme). The "play" variant has been generated using hand-picked locations for the goal and starting positions while the "diverse" variant has been generated using random goal and starting positions.

Baselines We compare QPHIL to 5 previous methods ranging from model-free behavior cloning and offline RL methods. For the behavior cloning methods, we include in our baselines the flat Goal-Conditioned Behavior Cloning (GCBC) (Ghosh et al., 2019) and the Hierarchical Goal-Conditioned Behavior Cloning (HGCBC) (Gupta et al., 2019). For offline RL, we include a goal-conditioned variant of Implicit Q-Learning (GCIQL) (Kostrikov et al., 2021), Hierarchical Implicit Q-Learning (HIQL) (Park et al., 2024) as well as a flatten version of it called GCIVL which relies only the the value function to compute the advantages (Park et al., 2024). We performed our experiments on QPHIL on 8 seeded runs, results are provided with the mean $\pm$ std format in Table 1. As the transformer-based methods such as DT (Chen et al., 2021), TAP (Jiang et al., 2022) or V-ADT (Ma et al., 2024) as been shown to fall short in long-range settings (see Park et al. (2024); Ma et al. (2024)) compared to hierarchical and/or value-based methods such as HIQL, we decided to focus our comparisons on the latter methods.

5.2 DOES QPHIL ARCHITECTURE ENABLE EFFICIENT LONG-TERM NAVIGATION ?

Table 1: **Evaluating QPHIL on AntMaze environments.** We see that QPHIL scales well with the length of the navigation range, competing with SOTA performance on the smaller mazes and improving significantly on the SOTA on the larger settings.

env_name	GCBC	GCIQL	GCIVL	HGCBC w/o repr	HGCBC w/repr	HIQL w/o repr	HIQL w/repr	QPHIL
antmaze-medium-diverse-v2	65 ± 11	80 ± 8	82 ± 8	46 ± 11	13 ± 9	92 ± 4	92 ± 4	92 ± 4
antmaze-medium-play-v2	60 ± 11	80 ± 9	84 ± 9	48 ± 9	10 ± 8	90 ± 5	92 ± 4	91 ± 2
antmaze-large-diverse-v2	10 ± 5	21 ± 10	59 ± 13	78 ± 7	18 ± 8	88 ± 4	85 ± 8	82 ± 6
antmaze-large-play-v2	9 ± 5	25 ± 12	50 ± 9	79 ± 7	14 ± 13	87 ± 7	84 ± 7	80 ± 3
antmaze-ultra-diverse-v0	16 ± 9	20 ± 8	6 ± 5	71 ± 12	32 ± 24	71 ± 7	71 ± 12	62 ± 7
antmaze-ultra-play-v0	15 ± 10	20 ± 10	10 ± 6	64 ± 12	13 ± 15	63 ± 20	74 ± 23	62 ± 4
antmaze-extreme-diverse-v0	9 ± 6	12 ± 7	5 ± 10	6 ± 14	0 ± 4	14 ± 17	20 ± 20	40 ± 13
antmaze-extreme-play-v0	8 ± 7	16 ± 7	0 ± 0	11 ± 10	4 ± 8	12 ± 16	28 ± 27	50 ± 7

We first analyze the performance in success rate of the different baselines on the state-based AntMaze-{Medium,Large,Ultra} settings. As usual in D4RL Antmaze benchmarks (Park et al., 2024), starting state and target goals are sampled near two reference points, forcing the agent to walk across the entire maze. Figure 2. showcases examples of tokenizations obtained by our VQ-VAE model. One can observe a tendency for learned clusters to decrease in size in the middle of all mazes, which can be explained by the higher number of demonstration data in those areas. Table 1 shows the results of our experiment on the set of AntMaze map. For HIQL, "w/ repr." means the use of representations for subgoals and goals, in opposite of "w/o repr." that takes raw values (see (Park et al., 2024)). We see that our method is near the state-of-the art on smaller and average scale maps. To test QPHIL's scaling ability in more difficult settings, we created a larger AntMaze environment called AntMaze-Extreme (Figure 3), along with two datasets variants "diverse" and "play". As shown in Table 1, in AntMaze-Extreme QPHIL attains up to 50% success rate, which is significantly above baseline results, e.g. HIQL scores in diverse and play datasets. While QPHIL remains competitive in short-term settings, our approach is especially well suited for long-distance goal reaching navigation.

5.3 WHAT IS THE IMPACT OF THE CONTRASTIVE LOSS USED FOR LANDMARK LEARNING ?

The use of a contrastive loss is essential in the context of high dimensional data, where the VQ-VAE reconstruction loss is not enough to learn temporally consistent latent encodings (meaning that temporally nearby states share spatially nearby encodings). To assess the impact of the contrastive loss on the learned landmarks, we compute, for each token, the minimum and maximum distances between states position and their corresponding codebook's decoded position. This is represented as histograms in Figure 4. While the unconstrained VQ-VAE tends to allocate higher token density in areas of higher data density, we observe that the contrastive loss results in a smoother repartition of the tokens, which in consequence increases the performance of our model. This allows to stabilize conditioning in areas of high data density.

Figure 2: **Tokenization example.** Each background point's color corresponds to its associated token. Tokens align with walls thanks to the contrastive loss.

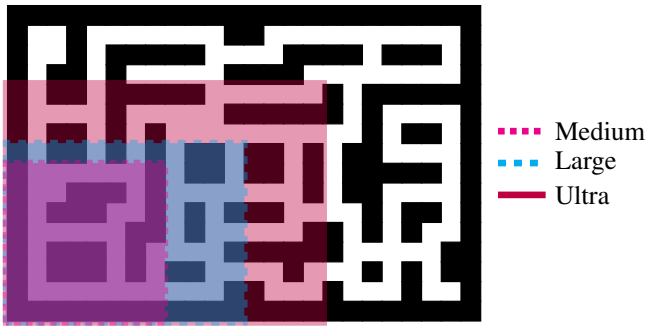
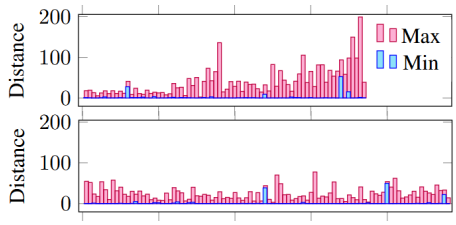


Figure 3: **Evaluating QPHIL on AntMaze-Extreme.** A top-down view of the maze with size comparisons.



(a) Intertoken histograms

(b) Performances on Antmaze-Extreme.

Method	Diverse	Play
QPHIL (non-cont)	17 ± 1	26 ± 2
QPHIL	$\mathbf{40 \pm 13}$	$\mathbf{50 \pm 7}$

Figure 4: **Non-contrastive (top) and contrastive (bottom) intertoken distance histograms.** The non-contrastive tokenization results in higher extreme sized tokens, while the contrastive tokenization yields a smoother distribution, improving performance.

5.4 What is the impact of the different losses when training the quantizer ?

In QPHIL, the VQ-VAE quantizer is learned using a linear composition of three losses: the commit loss, the contrastive loss, and the reconstruction loss, which all have an associated coefficient hyperparameter to be tuned. In our experiments, we used the same set of coefficients for each maze shape, which we found through a preliminary hyperparameter search to be robust across the diversity of maze sizes we considered. In the following, we analyze the impact of the different coefficients on the quality of the discretization.

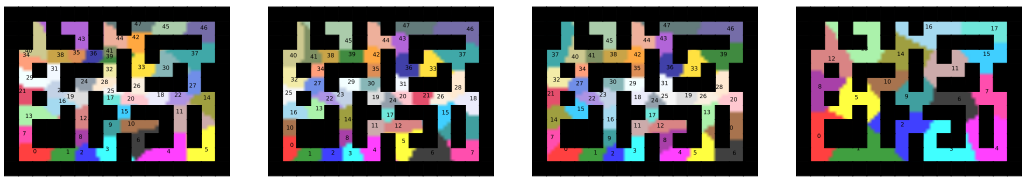


Figure 5: **Commit coefficient impact.** From left to right $\alpha_{\text{commit}} \in \{0, 1e1, 1e3, 1e6\}$, $\alpha_{\text{contrastive}} = 2e1$, $\alpha_{\text{recon}} = 1e5$. We see that a low commit coefficient leads to varying sized landmarks, while high commit loss diminishes the number of landmarks.

Commit loss The commit loss serves the purpose of maintaining a vicinity between the continuous encodings and the elements of the codebook in the latent space. A zero or low commit loss creates a poor repartition of the continuous representations with regard to the codebook. This creates varying

sized areas, where some are too small and some too big (e.g. Figure 5, leftmost map). On the other hand, a high commit loss will force the encoder to match the codebook vectors too rigidly. This leads to information loss as the encoder might struggle to represent subtle variations in the input. Also, this increases the amount of dead codebook vectors which consequently reduces the amount of used tokens for discretization (e.g. Figure 5, rightmost map).

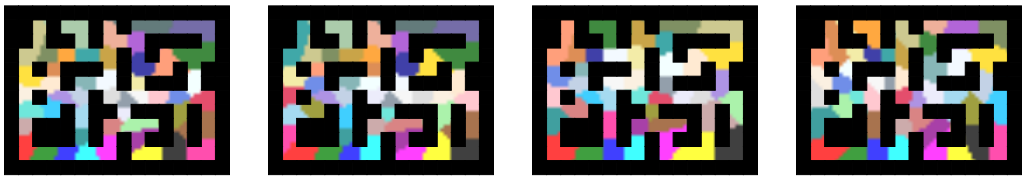


Figure 6: **Contrastive coefficient impact with reconstruction loss** With $\alpha_{\text{commit}} = 1e3, \alpha_{\text{contrastive}} \in \{0, 2, 2e1, 2e5\}, \alpha_{\text{recon}} = 1e5$. We see that a low contrastive coefficient leads to multiple piece landmarks, while high values of contrastive seem to fix this issue.

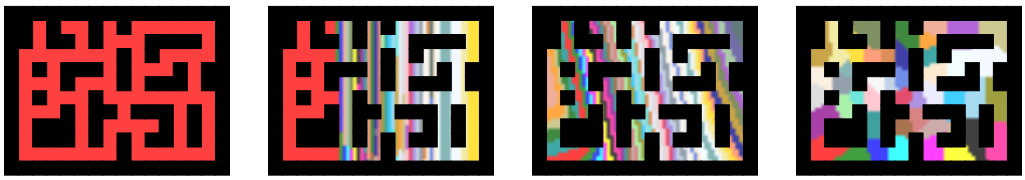


Figure 7: **Contrastive coefficient impact without reconstruction loss** With $\alpha_{\text{commit}} = 1e3, \alpha_{\text{contrastive}} \in \{0, 2, 2e1, 2e5\}, \alpha_{\text{recon}} = 0$. We see that the reconstruction loss is not needed to generate good tokens, however, it makes the tokenization more stable to hyperparamters.

Contrastive loss The contrastive loss serves the purpose of organizing temporally the latent space, which is paramount to create navigable landmarks. When used jointly with the reconstruction loss (see Figure 6), it ensures that the landmarks do not span through obstacles, consequently ensuring that landmarks are of single piece and navigable. If the contrastive coefficient is too low, multiple piece landmarks can appear by spanning across obstacles. If it is too high, it might overshadow other components, leading to the aforementioned failures. Used without the reconstruction loss (see Figure 7), the contrastive can manage a good tokenization but the reconstruction loss helps by showing better tokenization for a higher number of contrastive coefficient values.

6 Conclusion

Takeaways We proposed QPHIL, a hierarchical offline goal-conditioned reinforcement learning method that leverages pre-recorded demonstration to learn a discrete and temporally consistent representation of the state space. QPHIL utilizes those discrete state representations to plan subgoals in a discretized space and guide a low policy towards its final goal in a human-inspired manner. QPHIL reaches top performance in challenging long-term navigation benchmarks, showing promising next steps for discretization and planning in continous offline RL settings.

Limitations and future work QPHIL is a method aimed at longterm navigation as it aims to solve tasks where a low amount of landmarks is sufficient. QPHIL demonstrated the interest of space quantization for long range navigation settings in offline GCRL. As a future work, studying quantization techniques applicable to higher dimensional data (images, ...) and in more intricate planning settings (multi-token discretization and combinatorial representations), such as multi-task robotics scenarios could be promising. Finally, as discretized planning brings interpretability and editability, one could study how users can edit learned quantization to further improve performances or cheaply "update" the agent to environmental changes.

7 ACKNOWLEDGEMENTS

This work was granted access to the HPC resources of IDRIS under the allocation 2023-AD011014679 made by GENCI.

8 REPRODUCIBILITY STATEMENT

To ensure the reproductibility of our experiments, we provided all needed implementation details and hyperparameter values in the appendix. All our training have been seeded with the same 8 seeds: 0, 1, 2, 3, 4, 6 and 7. Also, we provide QPHIL's codebase along with the pretrained models ready to be evaluated on our test environments.

REFERENCES

Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. Opal: Offline primitive discovery for accelerating offline reinforcement learning. *arXiv preprint arXiv:2010.13611*, 2020.

Eloi Alonso, Maxim Peter, David Goumard, and Joshua Romoff. Deep reinforcement learning for navigation in aaa video games, 2020. URL https://arxiv.org/abs/2011.04764.

Anirudhan Badrinath, Yannis Flet-Berliac, Allen Nie, and Emma Brunskill. Waypoint transformer: Reinforcement learning via supervised learning with intermediate targets. *Advances in Neural Information Processing Systems*, 36, 2024.

Bram Bakker. Reinforcement learning with long short-term memory. *Advances in neural information processing systems*, 14, 2001.

Vassileios Balntas, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. In *Bmvc*, volume 1, pp. 3, 2016.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Silvia Chiappa, Sébastien Racaniere, Daan Wierstra, and Shakir Mohamed. Recurrent environment simulators. *arXiv preprint arXiv:1704.02254*, 2017.

André Correia and Luís A. Alexandre. Hierarchical decision transformer. In *IROS*, pp. 1661–1666, 2023. doi: 10.1109/IROS55552.2023.10342230. URL https://doi.org/10.1109/IROS55552.2023.10342230.

Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. Rvs: What is essential for offline rl via supervised learning? *arXiv preprint arXiv:2112.10751*, 2021.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning, 2019. URL https://arxiv.org/abs/1906.05253.

Kuan Fang, Patrick Yin, Ashvin Nair, and Sergey Levine. Planning to practice: Efficient online fine-tuning by composing goals in latent space. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4076–4083. IEEE, 2022.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems*, 34:20132–20145, 2021.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*, pp. 2052–2062. PMLR, 2019.

Chen Gao, Xingyu Peng, Mi Yan, He Wang, Lirong Yang, Haibing Ren, Hongsheng Li, and Si Liu. Adaptive zone-aware hierarchical planner for vision-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14911–14920, 2023.

Seyed Kamyar Seyed Ghasemipour, Dale Schuurmans, and Shixiang Shane Gu. Emaq: Expected-max q-learning operator for simple yet effective offline and online rl. In *International Conference on Machine Learning*, pp. 3682–3691. PMLR, 2021.

Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. *arXiv preprint arXiv:1912.06088*, 2019.

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.

Hany Hamed, Subin Kim, Dongyeong Kim, Jaesik Yoon, and Sungjin Ahn. Dr. strategy: Model-based generalist agents with strategic dreaming. *arXiv preprint arXiv:2402.18866*, 2024.

Karol Hausman, Yevgen Chebotar, Stefan Schaal, Gaurav Sukhatme, and Joseph J Lim. Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets. *Advances in neural information processing systems*, 30, 2017.

Nicolas Heess, Jonathan J Hunt, Timothy P Lillicrap, and David Silver. Memory-based control with recurrent neural networks. *arXiv preprint arXiv:1512.04455*, 2015.

Christopher Hoang, Sungryull Sohn, Jongwook Choi, Wilka Carvalho, and Honglak Lee. Successor feature landmarks for long-horizon goal-conditioned reinforcement learning. *Advances in neural information processing systems*, 34:26963–26975, 2021.

Riashat Islam, Hongyu Zang, Anirudh Goyal, Alex Lamb, Kenji Kawaguchi, Xin Li, Romain Laroche, Yoshua Bengio, and Remi Tachet Des Combes. Discrete factorial representations as an abstraction for goal conditioned reinforcement learning. *arXiv preprint arXiv:2211.00247*, 2022.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

Zhengyao Jiang, Tianjun Zhang, Michael Janner, Yueying Li, Tim Rocktäschel, Edward Grefenstette, and Yuandong Tian. Efficient planning in a compact latent action space. *arXiv preprint arXiv:2208.10291*, 2022.

Zhengyao Jiang, Yingchen Xu, Nolan Wagener, Yicheng Luo, Michael Janner, Edward Grefenstette, Tim Rocktäschel, and Yuandong Tian. H-gap: Humanoid control with a generalist planner. *arXiv preprint arXiv:2312.02682*, 2023.

Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

Pierre-Alexandre Kamienny, Jean Tarbouriech, Sylvain Lamprier, Alessandro Lazaric, and Ludovic Denoyer. Direct then diffuse: Incremental unsupervised skill discovery for state covering and goal reaching. In *ICLR 2022*, 2022.

Junsu Kim, Younggyo Seo, and Jinwoo Shin. Landmark-guided subgoal generation in hierarchical reinforcement learning. *Advances in neural information processing systems*, 34:28336–28349, 2021.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Kalle Kujanpää, Joni Pajarinen, and Alexander Ilin. Hierarchical imitation learning with vector quantized models. In *International Conference on Machine Learning*, pp. 17896–17919. PMLR, 2023.

Kalle Kujanpää, Joni Pajarinen, and Alexander Ilin. Hierarchical imitation learning with vector quantized models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 17896–17919. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/kujanpaa23a.html.

Kalle Kujanpää, Joni Pajarinen, and Alexander Ilin. Hybrid search for efficient planning with completeness guarantees. *Advances in Neural Information Processing Systems*, 36, 2024.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *Advances in neural information processing systems*, 32, 2019.

Shakti Kumar, Jerrod Parker, and Panteha Naderian. Adaptive transformers in rl. *arXiv preprint arXiv:2004.03761*, 2020.

Seungjae Lee, Daesol Cho, Jonghae Park, and H Jin Kim. Cqm: curriculum reinforcement learning with a quantized world model. *Advances in Neural Information Processing Systems*, 36, 2024a.

Yoonhyung Lee, Younhyung Chae, and Kyomin Jung. Leveraging vq-vae tokenization for autoregressive modeling of medical time series. *Artificial Intelligence in Medicine*, pp. 102925, 2024b.

Jinning Li, Chen Tang, Masayoshi Tomizuka, and Wei Zhan. Hierarchical planning through goal-conditioned offline reinforcement learning. *IEEE Robotics and Automation Letters*, 7(4):10216–10223, 2022. doi: 10.1109/LRA.2022.3190100.

Baiting Luo, Ava Pettet, Aron Laszka, Abhishek Dubey, and Ayan Mukhopadhyay. Scalable decision-making in stochastic environments through learned temporal abstraction. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=pQsllTesiE.

Jianlan Luo, Perry Dong, Jeffrey Wu, Aviral Kumar, Xinyang Geng, and Sergey Levine. Action-quantized offline reinforcement learning for robotic skill learning. In *Conference on Robot Learning*, pp. 1348–1361. PMLR, 2023.

Yi Ma, Jianye HAO, Hebin Liang, and Chenjun Xiao. Rethinking decision transformer via hierarchical reinforcement learning. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=WsM4TVsZpJ.

Pietro Mazzaglia, Tim Verbelen, Bart Dhoedt, Alexandre Lacoste, and Sai Rajeswar. Choreographer: Learning and adapting skills in imagination. *arXiv preprint arXiv:2211.13350*, 2022.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-optimal representation learning for hierarchical reinforcement learning. *arXiv preprint arXiv:1810.01257*, 2018a.

Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018b.

Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing transformers for reinforcement learning. In *International conference on machine learning*, pp. 7487–7498. PMLR, 2020.

Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. Hiql: Offline goal-conditioned rl with latent states as actions. *Advances in Neural Information Processing Systems*, 36, 2024.

Karl Pertsch, Youngwoon Lee, Yue Wu, and Joseph J Lim. Guided reinforcement learning with learned skills. *arXiv preprint arXiv:2107.10253*, 2021.

Rafael Figueiredo Prudencio, Marcos ROA Maximo, and Esther Luna Colombini. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

Dushyant Rao, Fereshteh Sadeghi, Leonard Hasenclever, Markus Wulfmeier, Martina Zambelli, Giulia Vezzani, Dhruva Tirumala, Yusuf Aytar, Josh Merel, Nicolas Heess, et al. Learning transferable motor skills with hierarchical latent mixture policies. *arXiv preprint arXiv:2112.05062*, 2021.

Erick Rosete-Beas, Oier Mees, Gabriel Kalweit, Joschka Boedecker, and Wolfram Burgard. Latent plans for task-agnostic offline reinforcement learning. In *Conference on Robot Learning*, pp. 1838–1849. PMLR, 2023.

Jürgen Schmidhuber. Learning to generate sub-goals for action sequences. In *Artificial neural networks*, pp. 967–972, 1991.

Wonchul Shin and Yusung Kim. Guide to control: Offline hierarchical reinforcement learning using subgoal generation for long-horizon and sparse-reward tasks. In *IJCAI*, pp. 4217–4225, 2023.

Shyam Sudhakaran and Sebastian Risi. Skill decision transformer. *arXiv preprint arXiv:2301.13573*, 2023.

Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT Press, 2 edition, 2018.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International conference on machine learning*, pp. 3540–3549. PMLR, 2017.

Chenjun Xiao, Han Wang, Yangchen Pan, Adam White, and Martha White. The in-sample softmax for offline reinforcement learning. *arXiv preprint arXiv:2302.14372*, 2023.

Taku Yamagata, Ahmed Khalil, and Raul Santos-Rodriguez. Q-learning decision transformer: Leveraging dynamic programming for conditional sequence modelling in offline rl. In *International Conference on Machine Learning*, pp. 38989–39007. PMLR, 2023.

Yiqin Yang, Hao Hu, Wenzhe Li, Siyuan Li, Jun Yang, Qianchuan Zhao, and Chongjie Zhang. Flow to control: Offline reinforcement learning with lossless primitive discovery. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 10843–10851, 2023.

A DETAILS ABOUT BASELINES

Implicit Q-Learning (IQL) The main issue of offline RL is the overestimation of the value of out-of-distribution actions when minimizing the temporal difference error, leading the policy to favor overestimated actions:

$$\mathcal{L}_{TD} = \mathbb{E}_{(s_t,a_t,s_{t+1})\sim\mathcal{D}}\left[\left(r(s_t,a_t) + \gamma \max_{a_{t+1}} Q_{\hat{\theta}_Q}(s_{t+1},a_{t+1}) - Q_{\theta_Q}(s_t,a_t)\right)^2\right] \tag{13}$$

with $r(s_t, a_t)$ the task reward, $\hat{\theta}_Q$ the parameters of a target network. Without further interactions with the environment, those over-estimations cannot be corrected. While other work propose to regularize the loss function to avoid sampling out-of-distribution actions, Kostrikov et al. (2021) proposes IQL which estimates the in-distribution $\max Q$ operator with expectile regression. To do so, it learns a state value function $V_{\theta_V}(s_t)$ along a state-action value function $Q_{\theta_Q}(s_t, a_t)$:

$$\mathcal{L}_V(\theta_V) = \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}\left[L_2^\tau(Q_{\hat{\theta}_Q}(s_t,a_t) - V_{\theta_V}(s_t))\right] \tag{14}$$

$$\mathcal{L}_Q(\theta_Q) = \mathbb{E}_{(s_t,a_t,s_{t+1})\sim\mathcal{D}}[r(s_t,a_t) + \gamma\, V_{\theta_V}(s_{t+1}) - Q_{\theta_Q}(s_t,a_t))^2] \tag{15}$$

with $L_2^x(u) = |x - \mathbb{1}(u < 0)|u^2, x \in [0.5, 1)$ the expectile loss, which corresponds to an asymmetric square loss which penalises positive values more than negative ones the more x tends to 1, consequently leading $V_{\theta_V}(s_t)$ to lean towards $\max_{\,a_{t+1}\in\mathcal{A},\text{ st. }\pi_\beta(a_t|s_t)>0}\; Q_{\hat{\theta}_Q}(s_t,a_t)$ with π_β the datasets behavior policy. The use of two different networks is justified to train the value function only on the dataset action distribution without incorporating environment dynamics in the TD-error loss, which avoids the overestimation induced by lucky transitions. Then, the trained $V_{\theta_V}(s_t)$ and $Q_{\theta_Q}(s_t, a_t)$ are used to compute advantages to extract a policy π_{θ_π} with advantage weighted regression (AWR):

$$\mathcal{L}_\pi(\theta_\pi) = -\mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}\left[\exp\left(\beta(Q_{\hat{\theta}_Q}(s_t,a_t) - V_{\theta_V}(s_t)\right)\log\pi_{\theta_\pi}(a_t|s_t)\right], \tag{16}$$

with $\beta \in (0, +\infty]$ an inverse temperature. This corresponds to the cloning of the demonstrations with a bias towards actions that present a higher Q-value.

Hierarchical Implicit Q-Learning (HIQL) In the offline GCRL setting, the rewards are sparse, only giving signals for states where the goal is reached. Hence, as bad actions can be corrected by good actions and good actions can be polluted by bad actions in the future of the trajectory, offline RL methods are at risk of wrongly label bad actions as good one, and conversely good actions as bad ones. This leads to the learning of a noisy goal-conditioned value function: $\hat{V}(s_t, g) = V^*(s_t, g) + N(s_t, g)$ where V^* corresponds to the optimal value function and N corresponds to a noise. As the 'signal-to-noise' ratio worsens for longer term goals, offline RL methods such as IQL struggle when the scale of the GCRL problem increases, leading to noisy advantages in IQL's AWR weights for which the noise overtakes the signal. To alleviate this issue, Park et al. (2024) propose HIQL which leverages a learned noisy goal conditioned action-free value function inspired by IQL:

$$\mathcal{L}_V(\theta_V) = \mathbb{E}_{(s_t,s_{t+1})\sim\mathcal{D},g\sim p(g|\tau)}\left[L_2^x(r(s_t,g) + \gamma V_{\hat{\theta}_V}(s_{t+1},g) - V_{\theta_V}(s_t,g))\right] \tag{17}$$

by using it to train two policies, $\pi^h(s_{t+k}|s_t, g)$ to generate subgoals from the goal and $\pi^l(a_t|s_t, g)$ to generate actions to reach the subgoals:

$$\mathcal{L}_{\pi^h}(\theta_h) = \mathbb{E}_{(s_t,s_{t+k},g)}[\exp(\beta\cdot(V_{\theta_V}(s_{t+k},g) - V_{\theta_V}(s_t,g)))\log\pi^h_{\theta_h}(s_{t+k}|s_t,g)] \tag{18}$$

$$\mathcal{L}_{\pi^l}(\theta_l) = \mathbb{E}_{(s_t,a_t,s_{t+1},s_{t+k})}[\exp(\beta\cdot(V_{\theta_V}(s_{t+1},s_{t+k}) - V_{\theta_V}(s_t,s_{t+k}))\log\pi^l_{\theta_l}(a_t|s_t,s_{t+k})] \tag{19}$$

With this division, each policy benefits from higher signal-to-noise ratios as the low-policy only queries the value function for nearby subgoals $V(s_{t+1}, s_{t+k})$ and the high policy queries the value function for more diverse states leading to dissimilar values $V(s_{t+k}, g)$.For high-dimensional states like images, HIQL proposes to learn states and goal representations $\phi(s)$ to reduce the dimension, allowing consequently easier subgoal generation. With representations, the writing of the policies become $\pi^h_{\theta_h}(\phi(s_{t+k})|s_t, g)$ and $\pi^l_{\theta_l}(a_t|s_t, \phi(s_{t+k}))$.

Flat-policies "signal-to-noise" issues The usual policy learning strategy in offline RL and as such goal-conditioned offline RL is to learn the policy by weighting its updates by a function of is goal-conditioned advantage. In the case of HIQL, low-level and high-level policy updates are performed using a learned action-free advantage function: $\hat{A}(s_t, s_{t+1}, g) = \hat{V}(s_{t+1}, g) - \hat{V}(s_t, g)$, where $\hat{V}$ itself is a neural network learned through action-free IQL updates. As high values corresponds to an expectation on the discounted cumulative sum of rewards, given a state and a goal, they indicate a notion of temporal proximity in the goal-conditioned sparse rewards case. The advantage gives consequently an indication of an approach of the goal g allowing the policy to learn directions. However, as the goal moves away from the current state, the learned value function may provide noisy estimates. We can write the learned value function in the from: $\hat{V}(s, g) = V^*(s, g) + N(s, g)$ where $V^*(s, g)$ corresponds to the minimal (in-distribution) distance between s and g and $N(s, g)$ a random noise. Borrowing from Park et al. (2024) paper, we could assume $N(s, g) = \sigma z_{s,g} V^*(s, g)$, where σ is the standard deviation and $z_{s,g}$ is the a random variable that follows a standard normal distribution. This corresponds to a gaussian noise proportional to the temporal distance between s and g. If we rewrite the advantage:

$$\begin{aligned}
\hat{A}(s_t, s_{t+1}, g) &= \hat{V}(s_{t+1}, g) - \hat{V}(s_t, g) && (20)\\
&= V^*(s_{t+1}, g) + \sigma z_{s_{t+1},g} V^*(s_{t+1}, g) - V^*(s_t, g) + \sigma z_{s_t,g} V^*(s_t, g) && (21)\\
&= V^*(s_{t+1}, g) - V^*(s_t, g) + \sigma z_{s_{t+1},g} V^*(s_{t+1}, g) - \sigma z_{s_t,g} V^*(s_t, g) && (22)\\
&\overset{d}{=} \underbrace{A^*(s_{t+1}, s_t, g)}_{signal} + \underbrace{z\sqrt{\sigma_1^2 V^*(s_{t+1}, g)^2 + \sigma_2^2 V^*(s_t, g)^2}}_{noise} && (23)
\end{aligned}$$

Hence, for faraway goals, the "signal-to-noise" ratio (SNR) defined in this case by:

$$SNR(s_{t+1}, s_t, g) = \frac{A^*(s_t, s_{t+1}, g)}{N(s_t, s_{t+1}, g)} = \frac{A^*(s_t, s_{t+1}, g)}{z\sqrt{\sigma_1^2 V^*(s_{t+1}, g)^2 + \sigma_2^2 V^*(s_t, g)^2}}$$

can be underwhelming, because the difference in advantage is too small compared to the noise induced by the estimation. HIQL proposes to learn two different policies though the advantage estimates of a single value function. A high policy $\pi^h(s_{t+k}|s_t, g)$ is trained to generate find subgoals that maximizes $\hat{V}(s_{t+k}, g)$ and a low policy $\pi^l(a_t|s_t, g)$ is trained to maximize $\hat{V}(s_{t+1}, g)$. Hence, we can write the SNR for each level:

$$SNR^h(s_t, s_{t+k}, g) = \frac{A^*(s_t, s_{t+k}, g)}{N(s_t, s_{t+k}, g)} = \frac{A^*(s_t, s_{t+k}, g)}{z\sqrt{\sigma_1^2 V^*(s_{t+k}, g)^2 + \sigma_2^2 V^*(s_t, g)^2}}$$

$$SNR^l(s_t, s_{t+1}, s_{t+k}) = \frac{A^*(s_t, s_{t+1}, s_{t+k})}{N(s_t, s_{t+1}, s_{t+k})} = \frac{A^*(s_t, s_{t+1}, s_{t+k})}{z\sqrt{\sigma_1^2 V^*(s_{t+1}, s_{t+k})^2 + \sigma_2^2 V^*(s_t, s_{t+k})^2}}$$

The division in two policies allows to increase the high-policy SNR by comparing values from more distance states (higher signal) while also increasing the low-policy SNR by decreasing the distance between state and goal (lower noise). However, as the low-policy has to generate an instant action information and as the high-policy has to be conditioned on the real goal, HIQL seeks to find the optimal step k that balanced both SNR^h and SNR^l, which might still pose scaling issues for longer-range settings. Consequently, in very long-range scenarios, the high policy $\pi^h(s_{t+k}|s_t, g)$ may lead to noisy subgoal estimates, varying at each timestep and as such difficult for the low policy to follow.

B IMPLEMENTATION DETAILS

Here is the pseudo-codes for QPHIL's training and inference pipelines:

B.1 TRAINING PSEUDO-CODE

Algorithm 1 Quantizing Planner for Hierarchical Implicit Q-Learning (QPHIL)

Input: offline dataset $\mathcal{D}$
Initialize $q_{\theta_q}, \pi^p_{\theta_p}, V^{lm}_{\phi_{lm}}, \pi^{lm}_{\theta_{lm}}, V^g_{\phi_g}, \pi^g_{\theta_g}$

1. Train the quantizer q_{θ_q}
while not converged **do**
$(\theta_q, \theta_d) \leftarrow (\theta_q, \theta_d) - \lambda_{q,d} \nabla_{(\theta_q, \theta_q)} \mathcal{L}_{\text{quantizer}}(\theta_q, \theta_d)$ with Equation 7

2. Train the planner $\pi^p_{\theta_p}$
while not converged **do**
$\phi_p \leftarrow \theta_p - \lambda_p \nabla_{\theta_p} \mathcal{L}_{\text{planner}}(\theta_p)$ with Equation 8

3. Train the landmark value $V^{lm}_{\phi_{lm}}$ **and policy** $\pi^{lm}_{\theta_{lm}}$
while not converged **do**
$\phi_{lm} \leftarrow \phi_{lm} - \lambda_{V^{lm}} \nabla_{\phi_{lm}} \mathcal{L}_{V^{lm}}(\phi_{lm})$ with Equation 9
while not converged **do**
$\theta_{lm} \leftarrow \theta_{lm} + \lambda_{\pi^{lm}} \nabla_{\theta_{lm}} J_{\pi^{lm}}(\theta_{lm})$ with Equation 10

4. Train the goal value $V^g_{\phi_g}$ **and policy** $\pi^g_{\theta_g}$
while not converged **do**
$\phi_g \leftarrow \phi_g - \lambda_{V^g} \nabla_{\phi_g} \mathcal{L}_{V^g}(\phi_g)$ with Equation 11
while not converged **do**
$\theta_g \leftarrow \theta_g + \lambda_{\pi^g} \nabla_{\theta_g} J_{\pi^g}(\theta_g)$ with Equation 12

B.2 INFERENCE PSEUDO-CODE

Algorithm 2 Navigation with QPHIL

Input: start state s_0, goal state g, quantizer q_{θ_q}, planner $\pi^p_{\theta_p}$, landmark policy $\pi^{lm}_{\theta_{lm}}$, and goal policy $\pi^g_{\theta_g}$.
Initialize the history $\bar{\tau}^{\omega}_{\leq} \leftarrow (q_{\theta_q}(s^p_0))$, the plan $\bar{\tau}^{\omega}_{>} \leftarrow ()$, the current state $s \leftarrow s_0$ and the step $t \leftarrow 0$
while step t doesn't exceed a max number of steps **do**
  if $t = 0$ or re-planning is required **then** ▷ Generate future landmarks
    $\bar{\tau}^{\omega}_{>} \leftarrow (\omega^p_0, \omega^p_1, \ldots, \omega_g) \sim \pi^p_{\theta_p}(\cdot | \bar{\tau}^{\omega}_{\leq}, q_{\theta_q}(g))$
  if $\omega^p_0 \neq \omega_g$ **then** sample an action $a \sim \pi^{lm}_{\theta_{lm}}(\cdot | s, \omega^p_0)$
  else sample an action $a \sim \pi^g_{\theta_g}(\cdot \mid s, g)$
  Emit a in the environment and observe new state s'
  Set new current state $s \leftarrow s'$ and increment step $t \leftarrow t + 1$
  if $\omega^p_0 \neq \omega_g$ **then**
    if $q_{\theta_q}(s) = \omega^p_0$ **then** ▷ Subgoal reached, go to the next one
      $\bar{\tau}^{\omega}_{\leq}$.append($\bar{\tau}^{\omega}_{>}$.pop(0));
  else
    if $s \approx g$ **then** ▷ Final goal reached
      return;

B.3 HYPERPARAMETERS

For the VQ-VAE, we based our implementation on the `https://github.com/lucidrains/vector-quantize-pytorch.git` repository. We utilize for all AntMaze variants as the encoder $f^e_{\theta_e}$ and the decoder $f^d_{\theta_d}$ a simple 2-layer MLP with ReLU activations and hidden size of 16 and latent dimension d of 8. We add a gaussian noise to the input positions and we normalize the positions before feeding them to the encoder. Also, we vary the number of encodings in the codebook as the map grows.

For the transformer, we used as described an encoder-decoder architecture inspired by the paper (Vaswani et al., 2017) provided by the **torch.nn library**. We use a max sequence length of size 128, an embedding dimension of 128, a feed-forward dimension of 128, 4 layers of 4 heads and a dropout of 0.2 and we train our model with 250 epoch. We perform validation computation with a 0.95 dataset split and perform sampling with a temperature of 0.9. We optimize our model using the Adam optimizer with a learning rate of 1e-5.

For the low landmark policy and its value function, we adapt the policy proposed by HIQL. For the value function, we use a MLP policy of 3 layers with hidden size of 512 and GeLU activations. We apply layer normalization for each layers and initialize the weights with a variance scaling initialization of scale 1. No dropout is used for the value function. For the policy, we use a two layer MLP with hidden size of 256 and ReLU activations. We don't apply layer norm and initialize the parameters though a variance scaling initialization of scale 0.01. Our policy outputs the mean and standard deviation of an independent normal distribution. We clamp the log std of the output between -5 and 2. We train both the value function and the policy at the same time, performing 1e6 gradient steps with a batch size of 1024. For the IQL parameters, $\beta_{lm} = 3.0$, the expectile $x_{lm} = 0.9$, the polyak coefficient is 0.005 and the discount factor $\gamma_{lm} = 0.995$. We clip the AWR weights to 100. Also, for the value function, we sample the next token with a probability of 0.8 and the current token with probability 0.2. The targets updates are performed at each gradient step.

For the low goal policy, we trained our GC-IQL implementation with GC-IQL's hyper-parameters taken the HIQL paper: $(\beta_g, x_g, \gamma_g) = (0.99, 3, 0.9)$ for the smaller mazes (AntMaze-{Medium-Large}) and $(\beta_g, x_g, \gamma_g) = (0.995, 1, 0.7)$ for the larger mazes (AntMaze-{Ultra,Extreme}).

C ENVIRONMENT DETAILS

AntMaze is a very popular benchmark in offline reinforcement learning, and it is part of the D4RL Fu et al. (2020) dataset suite, interfaced through the Gym library Brockman et al. (2016). It uses the Mujoco physics engine Todorov et al. (2012) for its simulation.

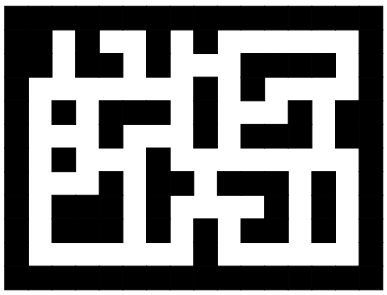


Figure 8: Maze of Antmaze ultra

The antmaze environment consist of a maze-like structure in which a simulated ant agent must navigate from a starting position to a goal position. Contrary to what one might expect, the agent is not controlled by simple directional commands. Instead, the ant is controlled through an 8-dimensional continuous action space. Due to the complex dynamics of the ant and the intricate structure of the maze, this environment poses a significant challenge for both exploration and planning. Each dimension of the action space corresponds to a torque applied to one of the ant's joints. Values in the action space are bounded between -1 and 1 and are in Nm. Hence, $\mathcal{A} = [-1, 1]^9$. The observation space is a 29-dimensional continuous space corresponding to the cartesian product of the x,y ant coordinates and the ant's configuration space S_{ant}. This space is unbounded in all directions: $\mathcal{S}_{ant} = \mathbb{R}^{27}$. The first dimension is the height in meter of the torso. The four following dimensions correspond to respectively the x, y, z and w orientation in radian of the torso. The height next dimensions are the angles between different links, in radian. Then, x, y and z velocities in m.s^{-1} followed by their respective angular velocities and angular velocities of all links, in rad.s^{-1}. The goal space is a subspace of $\mathbb{R}^2$: goals are given in x, y coordinates. The reward is sparse: it is equal to 0 until the ant has reach the goal, where it is equal to 1. Hence, $\mathcal{R} = \{0, 1\}$.

There are three variations of this environment: medium, large and ultra; each one consisting of a different maze structure. The ultra variant is not part of the original dataset and has been introduced in Jiang et al. (2022). Each maze has a different datasets, each one composed of 1000 trajectories of 1000 steps, for a total of 10^6 steps. D4RL provides two variations datasets per: "play" and "diverse". The former is generated using hand-picked locations for the goal and the starting position whereas the latter is generated using a random goal position and starting position for each trajectory.

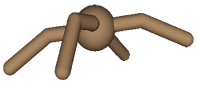


Figure 9: Ant

The extreme maze was designed following the same principles as the original maps. Its surface area is approximately 166% larger than antmaze ultra and three times the size of antmaze large. This new map is a direct extension of the original implementation, and both the implementation and the datasets are provided. A comparison is available in Figure 3. The two provided datasets are "play" and "diverse", both collected using the same methods as for the smaller maps. The maze is structured as a grid, where trajectories are generated on the grid, and a trained policy follows these paths to gather data.

D ADDITIONAL EXPERIMENTS

This sections contains additional experiments assessing the following questions:

1. How does the codebook's size impact the quantization and the overall performance ?
2. How does the coverage of the dataset (in the state space) broadly impact both the learning of the VQ-VAE and the policy ?
3. Does QPHIL still performs in diverse state-target initialization?

D.1 HOW DOES THE CODEBOOK'S SIZE IMPACT THE QUANTIZATION AND THE OVERALL PERFORMANCE ?

Table 2: **Impact of the codebook size on performance** There seem to be a threshold for performance regarding the token use percentage.

Codebook size	Number of used tokens	Performance of policy
1	1 (100 % used)	0
8	8 (100 % used)	6.0
24	24 (100 % used)	52.0
48	48 (100 % used)	86.1
96	96 (100 % used)	72.0
256	57 (22 % used)	88.0
1024	95 (9 % used)	88.0
2048	146 (7 % used)	82.0

We conducted an experiment to assess the impact of the codebook size on the overall performance on antmaze-ultra-diverse-v0, shown in Table 2. The first column corresponds to the number of available codebook vectors of the VQ-VAE quantizer. The second column corresponds to the number of used codebook vectors. We consider that a codebook vector is used if there exists a state that projects its encoding on it and that the set of states whose encodings are projected on the same codebook vector corresponds to a landmark. We observe that after a given threshold on the number of available codebook vectors, their use percentage decreases, i.e. the quantizer do not benefit from additional codebooks to achieve a good quantization. The performance of the method appears to saturate after the threshold of 48 tokens. Regarding the antmaze ultra map, this codebook size corresponds to a good trade-off ensuring an accurate "signal-to-noise" ratio while stabilizing high-level commands.

D.2 HOW DOES THE COVERAGE OF THE DATASET (IN THE STATE SPACE) BROADLY IMPACT BOTH THE LEARNING OF THE VQ-VAE AND THE POLICY ?

We see experimentally that areas with very low to null coverage (specifically walls here) share the same token as one of the nearest in-distribution state, showing a certain amount of generalization of the tokenization. Also, high coverage areas have a tendency to constrain a higher number of smaller landmarks than the low coverage parts of the maze, due to the loss having more weight in those, since there are more samples. Section 5.4 illustrates that this issue is mitigated by the contrastive loss, leading to tokens of more uniform size. For the policy, as we use an offline reinforcement learning (IQL), low coverage areas are to be avoided and IQL seeks to sample actions within training distribution to avoid getting out-of-distribution.

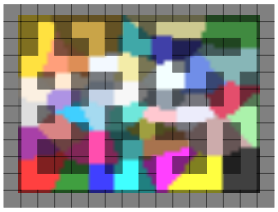


Figure 10: **Landmarks formed within the walls.**

D.3 DOES QPHIL STILL PERFORMS IN DIVERSE STATE-TARGET INITIALIZATION?

Previous sections relied on evaluating performance in AntMaze environments by classically sampling initial states s_0 and goals g from narrow distributions near two fixed points, requiring the agent to cross the entire maze. Regarding QPHIL, given such state and goal distributions do not span across multiple learned landmarks, each sampled navigation scenario leads to the similar landmark-based conditioning for our low-level policy, which is not convenient to conduct a comprehensive performance evaluation. Consequently, we designed Random-AntMaze evaluation environments, which cycles through a diverse set of 50 couples of (s_0, g) allowing a more rigorous test of the generalization capabilities of our model.

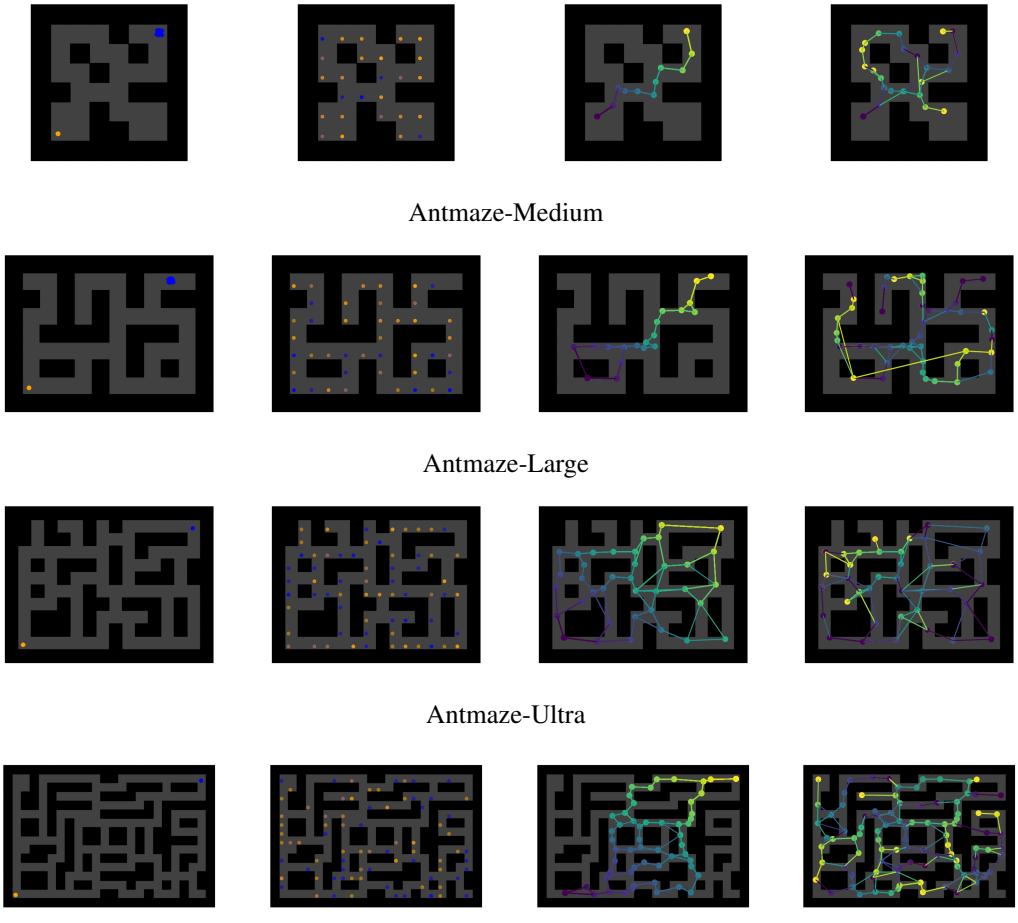


Figure 11: **Initializations comparisions between AntMaze and Random-AntMaze.** (left) Plots of 50 sampled starting positions s_0 (in blue) and target goals g (in orange) for AntMaze and Random-Antmaze. We see that Random-AntMaze has a broader s_0, g distribution and as such is a better fit for a comprehensive evaluation of navigation tasks. (left) Plots of 50 sampled planing paths with a color gradient indicating the order in sequence (yellow to blue). The high policy subgoals exhibit higher diversity in the Random-AntMaze variations.

Table 3: **Performance in long-range Random-AntMaze environments.** QPHIL is robust to random start and goal initializations, maintaining high success rates.

Method	**ultra-diverse**	**ultra-play**	**extreme-diverse**	**extreme-play**
QPHIL	62 ± 7	62 ± 4	40 ± 13	50 ± 7
QPHIL (random)	63 ± 5	63 ± 8	45 ± 5	42 ± 7

Those result the robustness of QPHIL to the diversity of states and goal initializations.