Reasoning with Latent Diffusion in Offline Reinforcement Learning

Siddarth Venkatraman[*]
Carnegie Mellon University,
Mila, Université de Montréal

Shivesh Khaitan[*]
Carnegie Mellon University

Ravi Tej Akella[*]
Carnegie Mellon University

John Dolan
Carnegie Mellon University

Jeff Schneider
Carnegie Mellon University

Glen Berseth
Mila, Université de Montréal

Abstract

Offline reinforcement learning (RL) holds promise as a means to learn high-reward policies from large static datasets, without need for further environment interactions. This is especially critical for robotics, where online learning can be prohibitively expensive. However, a key challenge in offline RL lies in effectively stitching portions of suboptimal trajectories from the static dataset while avoiding extrapolation errors arising due to a lack of support in the dataset. In this work, we propose a novel approach that leverages the expressiveness of latent diffusion to model in-support trajectory sequences as compressed latent skills. This facilitates learning a Q-function while avoiding extrapolation error via batch-constraining. The latent space is also expressive and gracefully copes with multi-modal data. We show that the learned temporally-abstract latent space encodes richer task-specific information for offline RL tasks as compared to raw state-actions. This improves credit assignment and facilitates faster reward propagation during Q-learning. Our method demonstrates state-of-the-art performance on the D4RL benchmarks, particularly excelling in long-horizon, sparse-reward tasks.

1 Introduction

Framing offline RL as a generative modeling problem has gained significant traction (Chen et al. [2021], Janner et al. [2021]); with the performance dependent on the expressive power of the generative models used. Recently diffusion models (Sohl-Dickstein et al. [2015], Song and Ermon [2019]), have emerged as state-of-the-art generative models for conditional image-generation (Ramesh et al. [2022], Saharia et al. [2022]). **We model the behavioral policy with diffusion and use this to avoid extrapolation error through batch-constraining,** a method that is highly scalable to large offline datasets prevalent in robotics. Previous diffusion-based sequence modelling methods in offline RL diffused over the raw state-action space. However, the low-level trajectory space tends to be poorly suited for reasoning. Some prior works (Pertsch et al. [2021], Ajay et al. [b]) have proposed to instead reason in more well-conditioned spaces composed of higher-level behavioral primitives (skills). Such temporal abstraction has been shown to result in faster and more reliable credit assignment (Machado et al. [2023], Mann and Mannor [2014]), particularly in long-horizon sparse-reward tasks. **We harness the expressivity of powerful diffusion generative models to reason with temporal abstraction and improve credit assignment.**

[*]Equal contribution

NeurIPS 2023 Workshop on Robot Learning: Pretraining, Fine-Tuning, and Generalization with Large Scale Models, New Orleans, United States of America

Inspired by the recent successes of Latent Diffusion Models (LDMs) (Rombach et al. [2022], Jun and Nichol [2023]), we propose learning similar latent trajectory representations for offline RL tasks by encoding rich high-level behaviors and learning a policy decoder to roll out low-level action sequences conditioned on these behaviors. The idea is to diffuse over semantically rich latent representations while relying on powerful decoders for high-frequency details. We perform state-conditioned diffusion on the learnt latent space and then learn a Q-function over states and corresponding latents. During Q-learning, we batch-constrain the candidate latents for the target Q-function using our expressive diffusion prior, thus mitigating extrapolation error. Our final policy samples latent skills from the LDM, scores the latents using the Q-function and executes the best behavior with the policy decoder. We refer to our method as **L**atent **D**iffusion-**C**onstrained **Q**-learning (**LDCQ**). The proposed latent diffusion skill learning method offers several advantages:

Flexible decoders for high-fidelity actions. The latent diffusion framework allows us to use powerful decoders for our low-level policy π_θ. Previous diffusion works for offline RL (Janner et al. [2022], Ajay et al. [a]) directly diffused over the raw state-action space, and architectural considerations for effective diffusion models limited the networks to be simple U-Nets (Ronneberger et al. [2015]). The separation of the diffusion model from the low-level policy allows us to model π_θ using an expressive autoregressive decoder.

Temporal Abstraction with information dense latent space. Prior works (Pertsch et al. [2021], Ajay et al. [b]) have learned latent space representations of skills using VAEs. Their use of weaker Gaussian priors forces them to use higher values of the KL penalty multiplier β to ensure the latents are well regularized. However, doing so restricts the information capacity of the latent, which limits the variation in behaviors captured by the latents. Our method allows modeling the dense latent space with diffusion.

Faster training and inference. Inference with LDMs is significantly faster than having to reconstruct the entire trajectory every forward pass with a raw trajectory diffusion model.

2 Latent Diffusion Reinforcement Learning

2.1 Two-Stage LDM training

Latent Representation and Low-Level Policy. The first stage in training the latent diffusion model is comprised of learning a latent trajectory representation. This means, given a dataset $\mathcal{D}$ of H-length trajectories $\boldsymbol{\tau}_H$ represented as sequences of states and actions, $\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \cdots \mathbf{s}_{H-1}, \mathbf{a}_{H-1}$, we want to learn a low-level policy $\pi_\theta(\mathbf{a}|\mathbf{s}, \mathbf{z})$ such that $\mathbf{z}$ represents high-level behaviors in the trajectory. This is done using a β-Variational Autoencoder (VAE) (Kingma and Welling, Higgins et al. [2016]). Specifically, we maximize the evidence lower bound (ELBO):

$$\mathcal{L}(\theta, \phi, \omega) = \mathbb{E}_{\boldsymbol{\tau}_H \sim D}[\mathbb{E}_{q_\phi(\mathbf{z}|\boldsymbol{\tau}_H)}[\sum_{t=0}^{H-1} \log \pi_\theta(\mathbf{a}_t \,|\, \mathbf{s}_t, \mathbf{z})] - \beta D_{KL}(q_\phi(\mathbf{z} \,|\, \boldsymbol{\tau}_H) \,||\, p_\omega(\mathbf{z} \,|\, \mathbf{s}_0))] \quad (1)$$

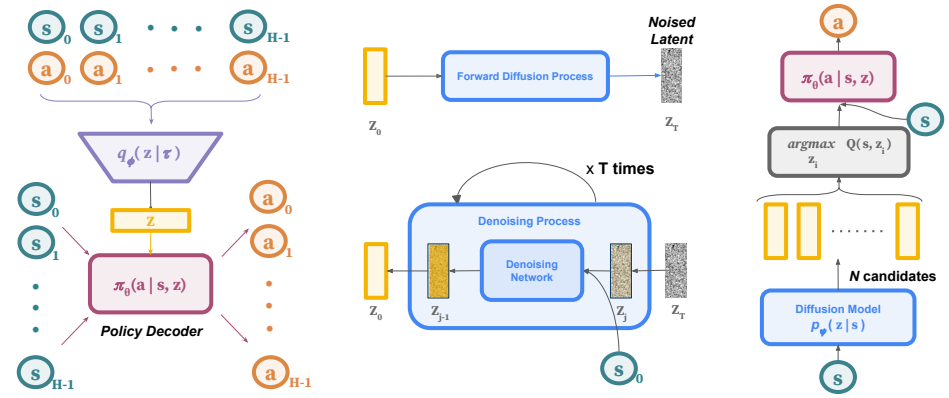
where q_ϕ represents our approximate posterior over $\mathbf{z}$ given $\boldsymbol{\tau}_H$, and p_ω represents our conditional Gaussian prior over $\mathbf{z}$, given $\mathbf{s}_0$. Note that unlike BCQ, which uses a VAE's conditional Gaussian prior as the state-conditioned generative model, our latent diffusion model only uses the β-VAE to learn a latent space to diffuse over. As such, the prior p_ω is simply a loose regularization of this latent space, and not a strong constraint. This is facilitated by the ability of latent diffusion models to later sample from such complex latent distributions. Prior works (Pertsch et al. [2021], Ajay et al. [b]) have learned latent space representations of skills using VAEs. Their use of weaker Gaussian priors forces them to use higher values of the KL penalty multiplier β to ensure the latents are well regularized. However, doing so restricts the information capacity of the latent, which limits the variation in behaviors captured by the latents. The low-level policy π_θ is represented as an autoregressive model which can capture the fine details across the action dimensions, and is similar to the decoders used by Ghasemipour et al. [2021] and Ajay et al. [b].

Latent Diffusion Prior. For training the diffusion model, we collect a dataset of state-latent pairs ($\mathbf{s}_0$, $\mathbf{z}$) such that $\boldsymbol{\tau}_H \sim \mathcal{D}$ is a H-length trajectory snippet, $\mathbf{z} \sim q_\phi(\mathbf{z} \mid \boldsymbol{\tau}_H)$ where q_ϕ is the VAE encoder trained earlier, and $\mathbf{s}_0$ is the first state in $\boldsymbol{\tau}_H$. We want to model the prior $p(\mathbf{z} \mid \mathbf{s}_0)$, which is the distribution of the learnt latent space $\mathbf{z}$ conditioned on a state $\mathbf{s}_0$. This effectively represents the different behaviors possible from the state $\mathbf{s}_0$ as supported by the behavioral policy that collected the dataset. To this end, we learn a conditional latent diffusion model $p_\psi(\mathbf{z} \mid \mathbf{s}_0)$ by learning the

time-dependent denoising function $\mu_\psi(\mathbf{z}_t, \mathbf{s}_0, t)$, which takes as input the current diffusion latent estimate $\mathbf{z}_t$ and the diffusion timestep t to predict the original latent $\mathbf{z}_0$:

$$\mathcal{L}(\psi) = \mathbb{E}_{t\sim[1,T],\boldsymbol{\tau}_H\sim\mathcal{D},\mathbf{z}_0\sim q_\phi(\mathbf{z}|\boldsymbol{\tau}_H),\mathbf{z}_t\sim q(\mathbf{z}_t|\mathbf{z}_0)}[\min\{\text{SNR}(t),\gamma\}(||\ \mathbf{z}_0 - \mu_\psi(\mathbf{z}_t, \mathbf{s}_0, t)\ ||^2)] \quad (2)$$

We use DDPM (Ho et al. [2020]) to sample from the diffusion prior in this work due to its simple implementation. As proposed in Ho and Salimans, we use classifier-free guidance for diffusion. This modifies the original training setup to learn both a conditional $\mu_\psi(\mathbf{z}_t, \mathbf{s}_0, t)$ and an unconditional model. The unconditional version, is represented as $\mu_\psi(\mathbf{z}_t, \emptyset, t)$ where a dummy token $\emptyset$ takes the place of $\mathbf{s}_0$. The following update is then used to generate samples: $\mu_\psi(\mathbf{z}_t, \emptyset, t) + w(\mu_\psi(\mathbf{z}_t, \mathbf{s}_0, t) - \mu_\psi(\mathbf{z}_t, \emptyset, t))$, where w is a tunable hyper-parameter. Increasing w results in reduced sample diversity, in favor of samples with high conditional density.



a) Latent Space Representation Training b) Latent Diffusion Model Training c) Policy Execution

Figure 1: **Latent Diffusion Reinforcement Learning Overview** a) We first learn the latent space and low-level policy decoder by training a β-VAE over H-length sequences from the demonstrator dataset. b) We train a latent diffusion prior conditioned on $\mathbf{s}_0$ to predict latents generated by the VAE encoder. c) After learning the high level Q function using BCQ, we score latents sampled by the prior with this Q function and execute the low-level policy π_θ conditioned on the argmax latent.

2.2 Q-Learning with Diffusion Skill Prior

We collect a replay buffer $\mathcal{B}$ for the dataset $\mathcal{D}$ of H-length trajectories and store transition tuples $(\mathbf{s}_t, \mathbf{z}, r_{t:t+H}, \mathbf{s}_{t+H})$ from $\boldsymbol{\tau}_H \sim \mathcal{D}$, where $\mathbf{s}_t$ is the first state in $\boldsymbol{\tau}_H$, $\mathbf{z} \sim q_\phi(\mathbf{z} \mid \boldsymbol{\tau}_H)$ is the latent sampled from the VAE approximate posterior, $r_{t:t+H}$ represents the γ-discounted sum of rewards accumulated over the H time-steps in $\boldsymbol{\tau}_H$, and $\mathbf{s}_{t+H}$ represents the state at the end of H-length trajectory snippet. The Q-function is learned with temporal-difference updates (Sutton and Barto [2018]), where we sample a batch of latents for the target argmax using the diffusion prior $p_\psi(\mathbf{z} \mid \mathbf{s}_{t+H})$. This should only sample latents which are under the support of the behavioral policy, and hence with the right temporal abstraction, allows for stitching skills to learn an optimal policy grounded on the data support. This is a form of BCQ as proposed by Fujimoto et al. [2019]. The resulting Q update can be summarized as:

$$Q(\mathbf{s}_t, \mathbf{z}) \leftarrow (r_{t:t+H} + \gamma^H Q(\mathbf{s}_{t+H}, \underset{\mathbf{z}_i\sim p_\psi(\mathbf{z}|\mathbf{s}_{t+H})}{\text{argmax}} (Q(\mathbf{s}_{t+H}, \mathbf{z}_i)))) \quad (3)$$

2.3 Policy Execution

The final policy for LDCQ comprises generating candidate latents $\mathbf{z}$ for a particular state $\mathbf{s}$ using the latent diffusion prior $\mathbf{z} \sim p_\psi(\mathbf{z} \mid \mathbf{s})$. These latents are then scored using the learnt Q-function and the best latent $\mathbf{z}_{max}$ is decoded using the VAE autoregressive decoder $\mathbf{a} \sim \pi_\theta(\mathbf{a} \mid \mathbf{s}, \mathbf{z}_{max})$ to obtain H-length action sequences which are executed sequentially. Note that the latent diffusion model is used both during training the Q-function and during the final evaluation phase, ensuring that the sampled latents do not go out-of-support.

3 Offline RL Benchmarks

We investigate the effectiveness of our Latent Diffusion Reinforcement Learning methods on the D4RL offline RL benchmark suite (Fu et al.). We compare with Behavior Cloning and several *state-of-the-art* offline RL methods. Diffuser (Janner et al. [2022]) and Decision Diffuser (Ajay et al. [a]) are prior raw trajectory diffusion methods. In maze2d and AntMaze tasks we use $H = 30$, in kitchen tasks we use $H = 20$ and in locomotion and adroit tasks we use $H = 10$. The other hyperparameters which are constant across tasks are provided in the supplemental material. In Table 1, we compare performance across tasks in the D4RL suite. We would like to highlight our results in the sparse reward AntMaze, FrankaKitchen and Adroit simulated robotics control tasks which are known to be quite challenging in D4RL, with most algorithms performing poorly since they require long horizon reasoning. In the maze navigation tasks, we also evaluate the performance of our goal-conditioned (LDGC) variant (described in supplemental material).

Table 1: Performance comparison on D4RL tasks. LDGC only evaluated in goal-directed maze tasks.

Dataset	**BC**	**BCQ**	**CQL**	**IQL**	**DT**	**Diffuser**	**DD**	**LDCQ (Ours)**	**LDGC (Ours)**
maze2d-large-v1	5.0	6.2	12.5	58.6	18.1	123.0	-	**150.1** $\pm$ 2.9	**206.8** $\pm$ 3.1
antmaze-medium-diverse-v2	0.0	0.0	53.7	**70.0**	0.0	45.5	24.6	**68.9** $\pm$ 0.7	**75.6** $\pm$ 0.9
antmaze-large-diverse-v2	0.0	2.2	14.9	47.5	0.0	22.0	7.5	**57.7** $\pm$ 1.8	**73.6** $\pm$ 1.3
kitchen-partial-v0	38.0	31.7	50.1	46.3	42.0	-	57.0	**67.8** $\pm$ 0.8	-
kitchen-mixed-v0	51.5	34.5	52.4	51.0	50.7	-	**65.0**	**62.3** $\pm$ 0.5	-
halfcheetah-medium-expert-v2	55.2	64.7	**91.6**	86.7	86.8	88.9	**90.6**	**90.2** $\pm$ 0.9	-
walker2d-medium-expert-v2	107.5	57.5	**108.8**	**109.6**	108.1	106.9	**108.8**	**109.3** $\pm$ 0.4	-
hopper-medium-expert-v2	52.5	**110.9**	105.4	91.5	107.6	103.3	**111.8**	**111.3** $\pm$ 0.2	-
halfcheetah-medium-v2	42.6	40.7	44.0	47.4	42.6	42.8	**49.1**	42.8 $\pm$ 0.7	-
walker2d-medium-v2	75.3	53.1	72.5	78.3	74.0	79.6	**82.5**	69.4 $\pm$ 3.5	-
hopper-medium-v2	52.9	54.5	58.5	66.3	67.6	74.3	**79.3**	66.2 $\pm$ 1.7	-
halfcheetah-medium-replay-v2	36.6	38.2	**45.5**	**44.2**	36.6	37.7	39.3	41.8 $\pm$ 0.4	-
walker2d-medium-replay-v2	26.0	15.0	**77.2**	73.9	66.6	70.6	75.0	68.5 $\pm$ 4.3	-
hopper-medium-replay-v2	18.1	33.1	95.0	94.7	82.7	93.6	**100.0**	86.2 $\pm$ 2.5	-
pen-human	34.4	68.9	37.5	71.5	-	-	-	**74.1**	-
hammer-human	1.2	0.3	**4.4**	1.4	-	-	-	1.5	-
door-human	0.5	0.0	9.9	4.3	-	-	-	**11.8**	-
relocate-human	0.0	-0.1	0.2	0.1	-	-	-	**0.3**	-
pen-cloned	37.0	44.0	39.2	37.3	-	-	-	**47.7**	-
hammer-cloned	0.6	0.4	2.1	2.1	-	-	-	**2.8**	-
door-cloned	0.0	0.0	0.4	**1.6**	-	-	-	1.1	-
relocate-cloned	-0.3	-0.3	-0.1	-0.2	-	-	-	-0.2	-
carla-lane-v0	17.2	-0.1	20.9	18.6	-	-	-	**24.7**	-

4 Conclusion

In this work, we showed that offline RL datasets comprised of suboptimal demonstrations have expressive multi-modal latent spaces which can be captured with temporal abstraction and is well suited for learning high-reward policies. With a powerful conditional generative model to capture the richness of this latent space, we demonstrated that the simple batch-constrained Q-learning framework can be directly used to obtain strong performance. Our biggest improvements come from long-horizon sparse reward tasks, which most prior offline RL methods struggled with, even previous raw trajectory diffusion methods. Our approach also required no task-specific tuning, except for the sequence horizon H. We believe that latent diffusion has enormous potential in offline RL and our work has barely scratched the surface of possibilities.

References

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B Tenenbaum, Tommi S Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? In *NeurIPS 2022 Foundation Models for Decision Making Workshop*, a.

Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. Opal: Offline primitive discovery for accelerating offline reinforcement learning. In *International Conference on Learning Representations*, b.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, P. Abbeel, A. Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Neural Information Processing Systems*, 2021.

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

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

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*, pages 3682–3691. PMLR, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2016.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

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.

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pages 9902–9915. PMLR, 2022.

Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023.

Diederik P Kingma and Max Welling. Auto-encoding variational {Bayes}. In *Int. Conf. on Learning Representations*.

Marlos C Machado, André Barreto, Doina Precup, and Michael Bowling. Temporal abstraction in reinforcement learning with the successor representation. *Journal of Machine Learning Research*, 24(80):1–69, 2023.

Timothy Mann and Shie Mannor. Scaling up approximate value iteration with options: Better policies with fewer iterations. In *International conference on machine learning*, pages 127–135. PMLR, 2014.

Tim Pearce, Tabish Rashid, Anssi Kanervisto, Dave Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, et al. Imitating human behaviour with diffusion models. *arXiv preprint arXiv:2301.10677*, 2023.

Karl Pertsch, Youngwoon Lee, and Joseph Lim. Accelerating reinforcement learning with learned skill priors. In Jens Kober, Fabio Ramos, and Claire Tomlin, editors, *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 188–204. PMLR, 16–18 Nov 2021. URL `https://proceedings.mlr.press/v155/pertsch21a.html`.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. volume 9351, pages 234–241, 10 2015. ISBN 978-3-319-24573-7. doi: 10.1007/978-3-319-24574-4_28.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France, 07–09 Jul 2015. PMLR. URL `https://proceedings.mlr.press/v37/sohl-dickstein15.html`.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

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

A Training Details

A.1 Source Code

The source code is available at: `https://github.com/ldcq/ldcq`.

A.2 Network Architecture

A.2.1 Variational Autoencoder

Encoder. For learning the latent trajectory representation, our VAE uses an architecture similar to Ajay et al. [b]. The encoder consists of two stacked bidirectional GRU layers, followed by mean and standard deviation heads which are each a 2 layer MLP with RELU activation for the hidden layers. The mean output head is a linear layer. The standard deviation output head is followed by a SoftPlus activation function to ensure it is always positive. The hidden layer dimension is fixed to 256.

Decoder. For the low-level policy decoder, we use an auto-regressive policy network similar to that described in EMAQ (Ghasemipour et al. [2021]), in which each element of the action vector has its own MLP network, taking as input the current state, latent representation, and all previously-sampled action elements. The complete action vector is sampled element-by-element, with the most recently sampled element becoming an input to the network for the next element. These MLP networks consists of 2 layers followed by 2 layers of mean and standard deviation heads similar to the encoder network. The mean output head is a linear layer and the standard deviation output head is followed by a SoftPlus activation. Again, ReLU activation is used after all hidden layer and the hidden dimension is fixed to 256.

A.2.2 Diffusion Prior

The diffusion prior is a deep ResNet (He et al. [2016]) architecture consisting of 8 residual blocks. It takes as input a vector representing a latent trajectory $\mathbf{z}$ and outputs a denoised version of the latent. The hidden blocks are of dimensions: [128, 32, 16, 8, 16, 32, 128]. Similar to a U-Net (Ronneberger et al. [2015]), the initial blocks are connected by residual connections to the later blocks having the same hidden dimension. The diffusion timestep t is encoded with a 256-dimensional sinusoidal embedding and then further encoded with a 2-layer MLP. The conditioning state $\mathbf{s}$ is also encoded by a 2 layer MLP. In each residual block, the state and time encodings are concatenated with the current layer activation for conditioning. When training the unconditional diffusion model for classifier-free guidance, the state input is given as a vector of zeros to represent a null vector.

A.2.3 Q-networks

The Q-networks take as input the state $\mathbf{s}$, latent $\mathbf{z}$ and consist of a 5 layer MLP with 256 hidden units in the first 3 layers, 32 hidden units in the third layer, and finally a linear output layer. We use GELU activation function between hidden layers. LayerNorm (Ba et al. [2016]) is applied before each activation.

A.3 Hyperparameters

The hyperparameters which are constant across tasks for the different stages of our proposed method are given in Tables 2, 3 and 4.

Table 2: β-VAE hyperparameters

Parameter	**Value**
Learning rate	5e-5
Batch size	128
Epochs	100
Latent dimension ($\mathbf{z}$)	16
β	0.05
Hidden layer dimension	256

Table 3: Diffusion training hyperparameters

Parameter	Value
Learning rate	1e-4
Batch size	32
Epochs	300
Diffusion steps (T)	500
Drop probability (For unconditional prior)	0.1
Variance schedule	linear
Sampling algorithm	DDPM
γ (For Min-SNR-γ weighing)	5

Table 4: Q-Learning hyperparameters

Parameter	Value
Learning rate	5e-4
Batch size	128
Discount factor (γ)	0.995
Target net update rate (ρ)	0.995
PER buffer α	0.7
PER buffer β	Linearly increased from 0.3 to 1, Grows by 0.03 every 3000 steps
Diffusion samples for batch argmax	500

A.4 Hardware

The models were trained on NVIDIA RTX A6000. Since different tasks have different dataset sizes, the model training times changes across tasks. Depending on the task, training the β-VAE took between 3-7 hours, the diffusion prior took between 4-12 hours and the Q-Learning took between 3-5 hours.

B Latent Diffusion Goal Conditioning (LDGC)

Diffuser (Janner et al. [2022]) proposed framing certain navigation problems as a sequence inpainting task, where the last state of the diffused trajectory is set to be the goal during sampling. We can similarly condition our diffusion prior on the goal to sample from feasible latents that lead to the goal. This prior is of the form $p_\psi(\mathbf{z} \mid \mathbf{s}_0, \mathbf{s}_g)$, where $\mathbf{s}_g$ is the target goal state. Since with latent diffusion, the training of the low-level policy alongside the VAE is done separately from the diffusion prior training, we can reuse the same VAE posterior to train different diffusion models, such as this goal-conditioned variant. At test time, we perform classifer-free guidance to further push the sampling towards high-density goal-conditioned latents. For tasks which are suited to goal conditioning, this can be simpler to implement and achieves better performance than Q-learning. Also, unlike Diffuser, our method does not need to have the goal within the planning horizon of the trajectory. This allows our method to be used for arbitrarily long-horizon tasks.

C Latent Diffusion-Constrained Planning (LDCP)

In this section, we explore another method to derive a policy for offline RL with latent diffusion other than our proposed methods *Latent Diffusion-Constrained Q-Learning (LDCQ)* and *Latent Diffusion Goal Conditioning (LDGC)*. This is a model-based method which learns a temporally abstract world model of the environment from offline data. Specifically, we learn a temporally abstract world model $p_\eta(\mathbf{s}_{t+H} \mid \mathbf{s}_t, \mathbf{z})$ that predicts the state outcome of executing a particular latent behavior after H steps. That is, given the current state $\mathbf{s}_t$ and a latent behavior $\mathbf{z}$ the model predicts the distribution of the state $\mathbf{s}_{t+H}$. This is trained in a supervised manner by sampling transition tuples $(\mathbf{s}_t, \mathbf{z}, \mathbf{s}_{t+H})$ from $\boldsymbol{\tau}_H \sim \mathcal{D}$ and minimizing the objective:

$$\mathcal{L}(\eta) = \mathbb{E}_{\boldsymbol{\tau}_H \sim \mathcal{D}} \mid\mid p_\eta(\mathbf{s}_{t+H} \mid \mathbf{s}_t, \mathbf{z}) - \mathbf{s}_{t+H} \mid\mid^2 \tag{4}$$

where η are the parameters of the temporally abstract world model p_η.

In goal-reaching environments, we leverage this model to do planning using the diffusion prior. We sample n latents $\mathbf{z}^i$ $(1 \leq i \leq n)$ using the diffusion prior for the current state $\mathbf{s}_t$, and use the learnt dynamics model to compute predicted future state $\mathbf{s}^i_{t+H}$ for each latent $\mathbf{z}^i$. These final states are then scored using a cost-function $\mathcal{J}$ and the latent corresponding to the best final state is chosen for execution. Note that sampling latents from the diffusion prior ensures that the world model is not queried on out-of-support data. We refer to this method as *Latent Diffusion-Constrained Planning (LDCP)*.

The cost-function which we use for the goal-reaching environments is the Euclidean distance to the goal. We can also extend this planning to horizons greater than H by further sampling latents for each future state $\mathbf{s}^i_{t+H}$ $(1 \leq i \leq n)$. This means, after sampling n latents for $\mathbf{s}_t$ with the diffusion prior, we further sample n more latents for each of the future states $\mathbf{s}^i_{t+H}$. This increases the 'planning depth' d. The final states at the last level of planning are then scored using the cost-function and the latent at the first level which led to that final state is chosen for execution. This procedure complexity grows exponentially and thus the planning depth has to be restricted. For a planning depth of d, there are n^d model calls. We found a planning-depth of $d = 2$ to be sufficient for all navigation environments achieving state-of-the-art results. Thus, with a latent sequence horizon of $H = 30$, our total planning horizon is 60. The results are tabulated in Table 5.

Table 5: Performance comparison on D4RL navigation tasks with LDCP.

Dataset	BC	BCQ	CQL	IQL	DT	Diffuser	DD	LDCQ (Ours)	LDGC (Ours)	LDCP (Ours)
maze2d-large-v1	5.0	6.2	12.5	58.6	18.1	123.0	-	**150.1** $\pm$ 2.9	**206.8** $\pm$ 3.1	**184.3** $\pm$ 3.8
antmaze-medium-diverse-v2	0.0	0.0	53.7	70.0	0.0	45.5	24.6	**68.9** $\pm$ 0.7	**75.6** $\pm$ 0.9	**77.0** $\pm$ 1.1
antmaze-large-diverse-v2	0.0	2.2	14.9	47.5	0.0	22.0	7.5	**57.7** $\pm$ 1.8	**73.6** $\pm$ 1.3	**59.7** $\pm$ 1.3

C.1 Visualizing Model Predictions

Learning a world model also allows us to visualize the effect of executing any given latent behavior. This means, even when the model is not used for planning, like in LDCQ, it can be used to compute the final state that will be reached for every latent behavior from a particular state. This information can be used to understand if the model is learning reasonable behavior modes.

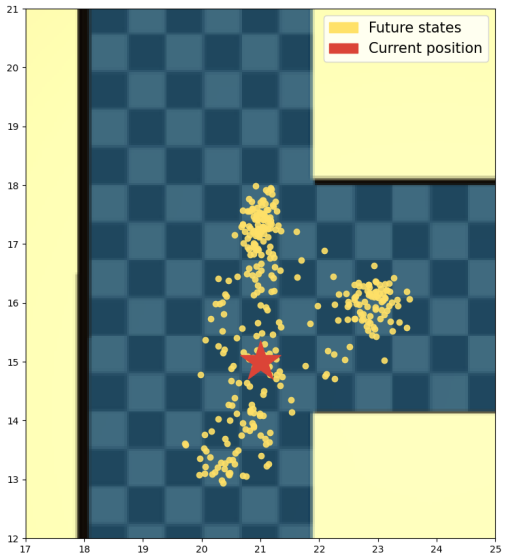


Figure 2: **Visualizing model predictions:** Visualization of future states with latents sampled from the diffusion prior at a T-intersection in antmaze-large-diverse-v2 D4RL task. We can see multimodal future state predidctions corresponding to 3 possible directions at the T-intersection.

We plot the xy-coordinates of our abstract world model $p_\eta(\mathbf{s}_{t+H} \,|\, \mathbf{s}_t, \mathbf{z})$ predictions at a T-intersection of AntMaze large environment for latents sampled from our diffusion prior $\mathbf{z} \sim p_\psi(\mathbf{z} \,|\, \mathbf{s}_t)$ in Figure 2

to demonstrate this. The plot shows that the diffusion prior sampled latents which go in all the three directions at the T-intersection.

D Temporal abstraction induces multi-modality in latent space

In this section, we study how the horizon length H affects the latent space and provide empirical justification for learning long-horizon latent space representations. For our experiment, we consider the *kitchen-mixed-v0* task from the D4RL benchmark suite (Fu et al.). The goal in this task is to control a 9-DoF robotic arm to manipulate multiple objects (microwave, kettle, burner and a switch) sequentially, in a single episode to reach a desired configuration, with only sparse 0-1 completion reward for every object that attains the target configuration. As Fu et al. states, there is a high degree of multi-modality in this task arising from the demonstration trajectories because different trajectories in the dataset complete the tasks in a random order. Thus, before starting to solve any task, the policy implicitly needs to *choose* which task to solve and then generate the actions to solve the task. Given a state, the dataset can consist of multiple behavior modes, and averaging over these modes leads to suboptimal action sequences. Hence, being able to differentiate between these tasks is desirable.

We hypothesize that as we increase our sequence horizon H, we should see better separation between the modes. In 3, we plot a 2D (PCA) projection of the VAE encoder latents of the starting state-action sequences in the kitchen-mixed dataset. With a lower horizon, these modes are difficult to isolate and the latents appear to be drawn from a Normal distribution Figure 3). However, as we increase temporal abstraction from $H = 1$ to $H = 20$, we can see *three* distinct modes emerge, which when cross-referenced with the dataset correspond to the three common tasks executed from the starting state by the behavioral policy (microwave, kettle, and burner). These modes better capture the underlying variation in an action sequence, and having picked one we can run our low-level policy to execute it. As demonstrated in our experiments, such temporal abstraction facilitates easier Q-stitching, with better asymptotic performance. However, in order to train these abstract Q functions, it is necessary to sample from the complex multi-modal distribution and the VAE conditional Gaussian prior is no longer adequate for this purpose, as shown in section E.

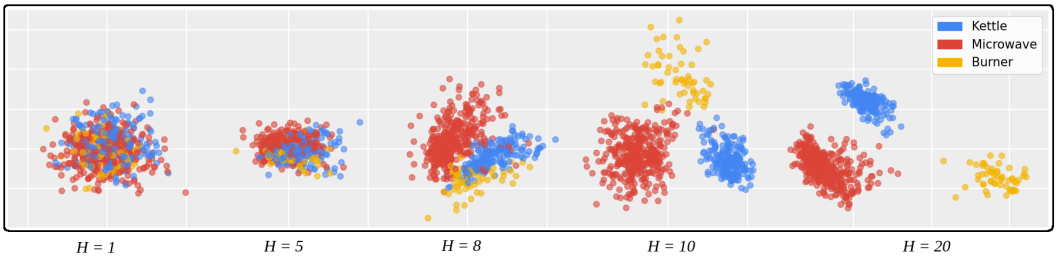
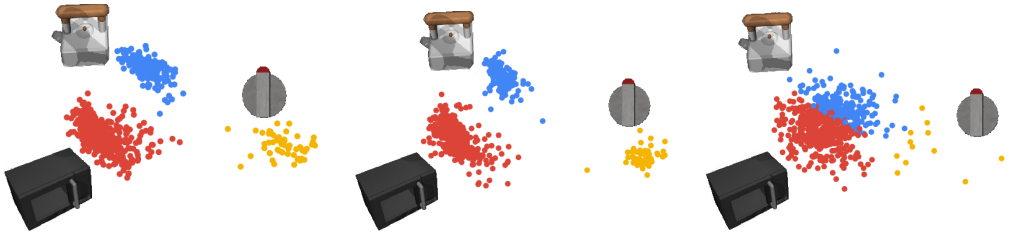


Figure 3: **Projection of latents across horizon**. Latent projections of trajectory snippets with different horizon lengths H. From the initial state there are 3 tasks (Kettle, Microwave, Burner) which are randomly selected at the start of each episode. These 3 primary modes emerge as we increase H, with the distribution turning multi-modal.

E LDMs address multi-modality in latent space

In this section, we provide empirical evidence that latent diffusion models are superior in modelling multi-modal distributions as compared to VAEs. For our experiment, we again consider the *kitchen-mixed-v0* task. The goal of the generative model here is to learn the prior distribution $p(\mathbf{z} \mid \mathbf{s})$ and sample from it such that we can get candidate latents corresponding to state $\mathbf{s}$ belonging to the support of the dataset. However, as demonstrated earlier, the multi-modality in the latent spaces increases with the horizon. We visualize the latents from the initial states of all trajectories in the dataset in Figure 4a using PCA with $H = 20$. The three clusters in the figure correspond to the latents of three different tasks namely microwave, kettle and burner. Similarly, we also visualize the latents predicted by the diffusion model Figure 4b) and the VAE conditional prior Figure 4c) for the same initial states by projecting them onto the principal components of the ground truth latents. We can see that the diffusion prior is able to sample effectively all modes from the ground truth latent distribution, while

the VAE prior spreads its mass over the three modes, and thus samples out of distribution in between the three modes. Using latents sampled from the VAE prior to learn the Q-function can thus lead to sampling from out of the support, resulting in extrapolation error.



(a) Ground truth (b) Diffusion prior (c) VAE prior

Figure 4: Visualization of latents projected using PCA for kitchen-mixed with $H = 20$. The diffusion prior models the ground truth much more accurately while the VAE prior generates out-of-distribution samples.

F CARLA Autonomous Driving task

To extend our method for tasks with high-dimensional image input spaces, we propose to compress the image space using an autoencoder such that our method operates on a compressed state space. This essentially means we create a low-dimensional compressed representation using an encoder $\mathcal{E}$ before using the LDCQ framework. Note that this encoder operates on a single image and not on a temporal sequence of images (Figure 5). The downstream framework of LDCQ however operates on the temporal compressed image sequences.

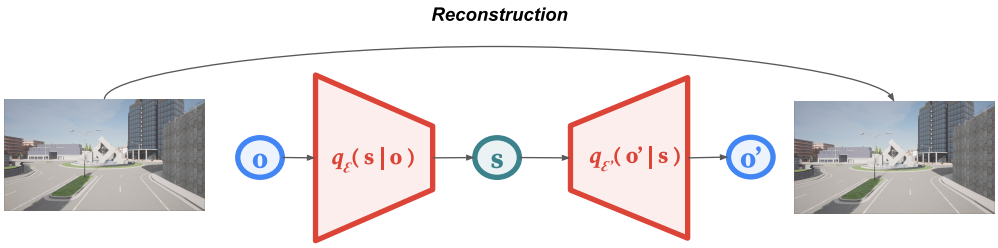


Figure 5: Autoencoder training for image-based task

We evaluate the performance of our method on the CARLA autonomous driving D4RL task. The task consists of an agent which has control to the throttle (gas pedal), the steering, and the break pedal for the car. It receives 48×48 RGB images from the driver's perspective as observations. We use a U-net autoencoder architecture to create a 32-dimensional compressed state for this task. The horizon for LDCQ is set to $H = 30$. The results are tabulated in Table 6.

Table 6: Performance comparison on image-based CARLA task

Dataset	BC	BCQ	CQL	IQL	LDCQ (Ours)
carla-lane-v0	17.2	-0.1	20.9	18.6	**24.7**

G Random walk 1D

In this experiment, we construct a simple toy problem to show how sampling effectively from the multimodal behavioral distribution helps the diffusion prior outperform a Gaussian VAE prior during Q-learning. We construct a simple toy MDP with a one-dimensional state space $\mathcal{S} \in [-10.0, 10.0]$.

The agent starts at the origin (0,0) and receives a reward of 10 if it reaches either the far left (-10.0) or far right (10.0) state, and -1 reward every other step. The environment times out after 500 steps. The action is the distance moved in that timestep with a max distance of length 1, $\mathcal{A} \in [-1.0, 1.0]$. The dataset consists of episodes where the agent randomly selects actions from the uniform distribution $a \sim \mathcal{U}([-1.0, -0.8] \cup [0.8, 1])$. This means the agent has a step size between 0.8 to 1.0 units either left or right every timestep. We train a VAE to try to fit this action distribution, and use BCQ to learn a policy. We also train train a diffusion based policy with LDCQ, using $H = 1$ and compare the results.

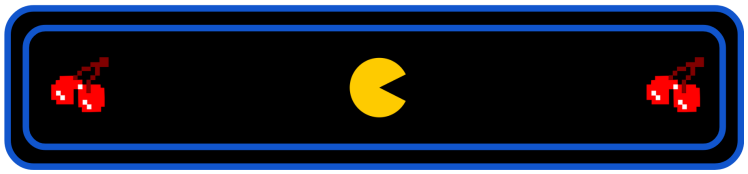


Figure 6: 1D Random walk

We find that the VAE frequently samples actions never present in the dataset. This is because the Gaussian mean to the above action distribution is 0.0, but no actual actions are present between $(-0.8, 0.8)$ where a large proportion of probability mass is assigned by the Gaussian model. Meanwhile, the diffusion prior is able to fit the 2 modes quite well. After 10000 steps of Q-learning, the diffusion constrained policy learns to navigate to either end perfectly and achieves an average reward of **-2.2** while the VAE constrained policy is still almost random, frequently taking actions with small step size and an average reward of **-66**.

H Increasing diffusion steps improves performance

We study the impact of the number of diffusion steps on the performance for LDCQ. We found that increase in diffusion timesteps T during evaluation generally corresponds to increase in task performance. We plot these results in Figure 7.

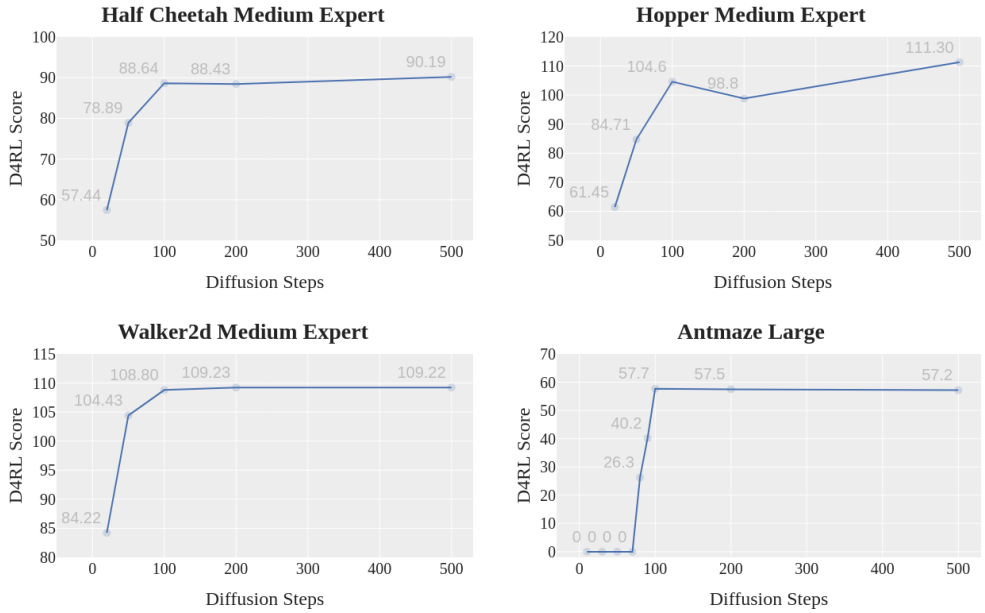


Figure 7: D4RL score for LDCQ with varying diffusion steps T in locomotion tasks.

We also used additional diffusion steps at time $t = 0$ similar to Diffusion-X (Pearce et al. [2023]). This means that after the DDPM sampling of diffusion from time T to 1, we run X additional

diffusion steps to further denoise the sample, assuming time-step $t = 1$. Pearce et al. [2023] proposed that this pushes the samples further towards higher-likelihood regions. We used 10 additional steps across experiments and found this to slightly improve performance.

I Performance improvement with temporal abstraction

We provide empirical evidence for improvement in task performance as we increase temporal abstraction or horizon H for different tasks. In general, we see improvement with increasing temporal abstraction until a certain point, when it drops possibly because of the limited capacity of the policy decoder.

For the locomotion tasks, we did not observe any noticeable difference with increase in temporal abstraction, so we ended up using a moderate sequence length $H = 10$. This could be due to the high frequency periodicity of these tasks that does not require much look-ahead.

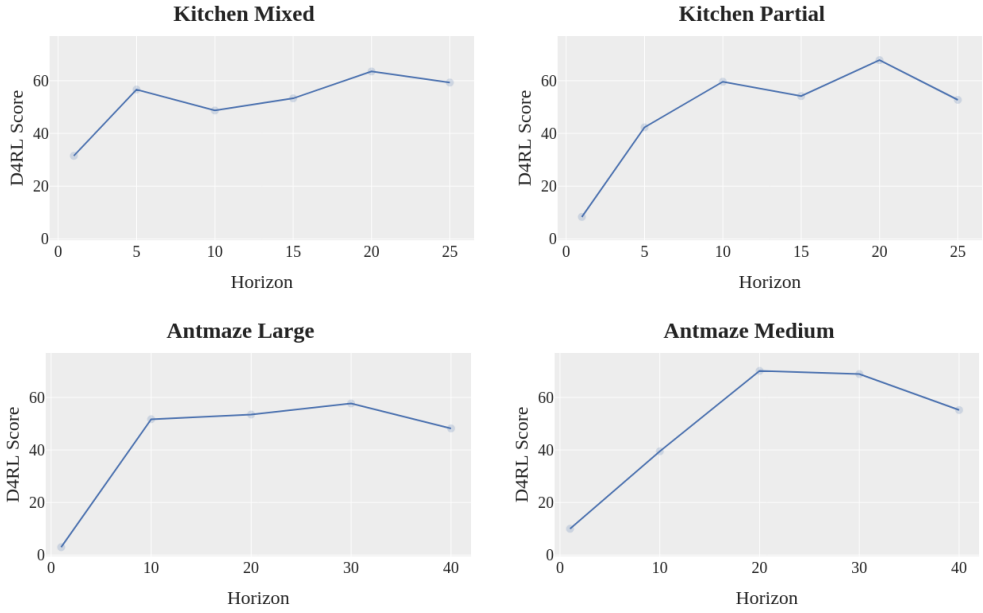


Figure 8: D4RL score for LDCQ with varying sequence horizons H.