# Examining Latent Space According to VAE Configurations

Yunho Kim  Mentor: Jaehyun Park, Sundong Kim

GIST Data Science Lab

## Abstract

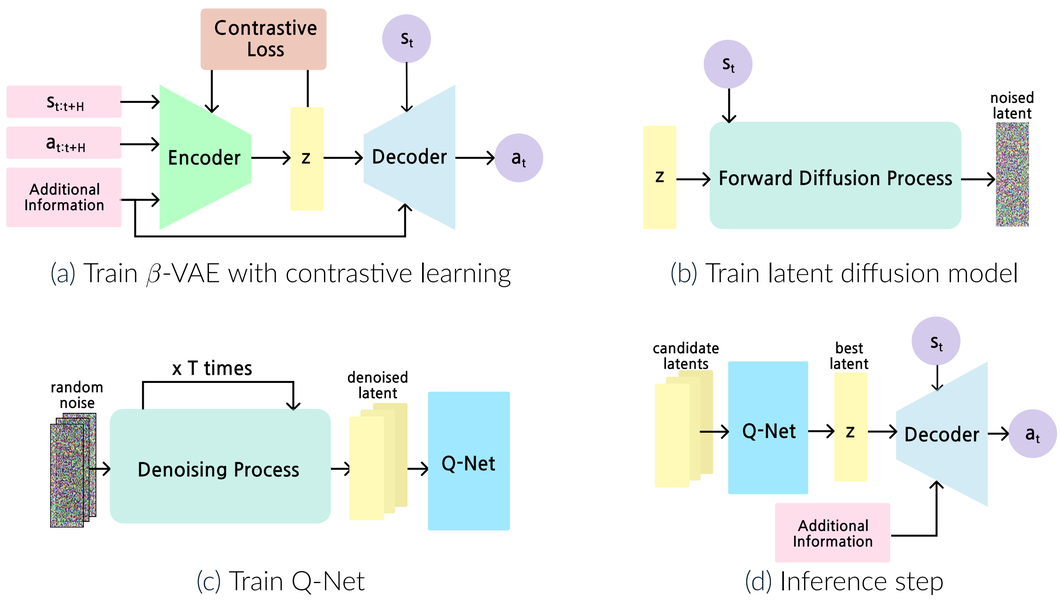
This research investigates Latent Diffusion Constrained Q-learning (LDCQ), a novel offline reinforcement learning(RL) method that combines skill-based RL with generative models to enhance planning in long-horizon, sparse reward environments. LDCQ integrates skill encoding through a $\beta$-Variational Autoencoder (VAE), diffusion model training, and Q-function learning. It shows superior performance in challenging long-horizon, sparse reward tasks such as Maze2d and FrankaKitchen. Numerous experiments highlighted that effective learning of the latent space through the $\beta$-VAE is crucial, as it directly impacts the following diffusion model training and Q-function learning. To further enhance this latent space representation, I tried combining InfoNCE contrastive learning and Wasserstein Distance concept to train $\beta$-VAE. However, these approaches unexpectedly led to performance declines. This research analyzes the reasons behind the inefficiency of these approaches and discusses their impact on the overall effectiveness of the LDCQ method.

## Research objectives

- **Objective 1:** Applying contrastive learning and Wasserstein concept to train the β-VAE in LDCQ.
- **Objective 2:** Analyzing the impact of various options in the β-VAE.

## Latent Diffusion Constrained Q-learnnig (LDCQ)

There are three steps in training with LDCQ: 1) training the $\beta$-Variational Autoencoder (VAE) to learn latent representations of $H$-length trajectories, 2) training the diffusion model using these latent vectors, and 3) training the Q-Net with latents sampled from the diffusion model to learn the Q-function.




(a) Train $\beta$-VAE with contrastive learning

(b) Train latent diffusion model

(c) Train Q-Net

(d) Inference step

Figure 1. (a)∼(c) 3 step of training LDCQ, (d) Inference step

## $\beta$-VAE with KL divergence VS Wasserstein distance

$\beta$-VAE in LDCQ is trained by optimizing the following loss function. It consists of a reconstruction loss and a KL divergence term between the approximate posterior $q_\phi(\mathbf{z}|\tau_H)$ and the conditional prior $p_\omega(\mathbf{z}|s_0)$.

$$\mathcal{L}_{VAE} = -\mathbb{E}_{\tau_H \sim D}\left[\mathbb{E}_{q_\phi(\mathbf{z}|\tau_H)}\left[\sum_{t=0}^{H-1}\log \pi_\theta(a_t|s_t,\mathbf{z})\right] - \beta D_{KL}(q_\phi(\mathbf{z}|\tau_H)||p_\omega(\mathbf{z}|s_0))\right] \tag{1}$$

Wasserstein distance is a metric used to measure the distance between two probability distributions by calculating the minimum cost required to transform one distribution into the other. This approach allows the VAE to utilize a more complex and flexible latent space. I refer to the VAE with the Wasserstein distance concept as WVAE. The quality of the latent space learned by WVAE is high because the sample distribution in the latent space is trained to closely match the prior distribution.

$$W_p(P,Q) = \left(\inf_{\gamma\in\Gamma(P,Q)}\mathbb{E}_{(x,y)\sim\gamma}\left[d(x,y)^p\right]\right)^{\frac{1}{p}} \tag{2}$$

$$\mathcal{L}_{WVAE} = -\mathbb{E}_{\tau_H \sim D}\left[\mathbb{E}_{q_\phi(\mathbf{z}|\tau_H)}\left[\sum_{t=0}^{H-1}\log \pi_\theta(a_t|s_t,\mathbf{z})\right] - \beta W(q_\phi(\mathbf{z}|\tau_H)||p_\omega(\mathbf{z}|s_0))\right] \tag{3}$$

## infoNCE

I use InfoNCE method for contrastive learning, which is based on the similarity between latents. Eq. **??** shows InfoNCE loss, where $z_i$ is the target latent, and $\mathbf{z}_i^+$ denotes the positive pair of $\mathbf{z}_i$. The InfoNCE method involves multiple negative pairs and one positive pair. To ensure that each latent representation captures a distinct meaning, each trajectory is treated as a positive pair with itself, while all others are considered negative pairs.

$$\mathcal{L}_{InfoNCE} = -\log\frac{\exp(sim(\mathbf{z}_i,\mathbf{z}_i^+))}{\sum_{j=0}^{N}\exp(sim(\mathbf{z}_i,\mathbf{z}_j))} \tag{4}$$

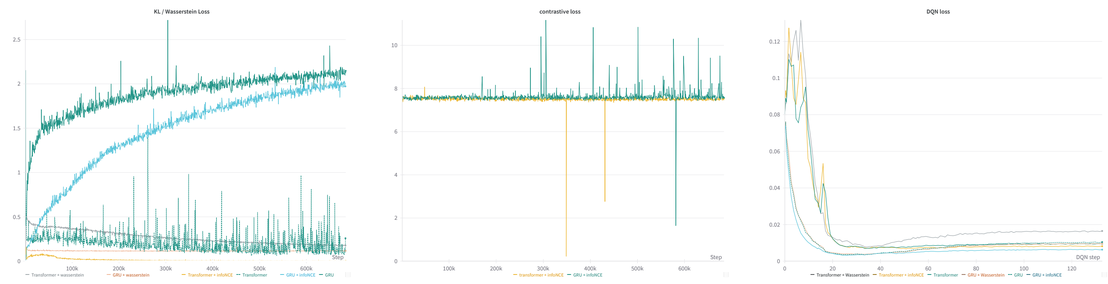
## Q-function learning

LDCQ uses $\mathbf{z}$ sampled from diffusion model to train the Q-function, allowing the evaluation of expected values across various $\mathbf{z}$. By training Q-function with this , the method can consider multiple possible futures from a given state and select the best $\mathbf{z}$.

$$Q(s_t,\mathbf{z}) \leftarrow r_{t:t+H} + \gamma^H Q(s_{t+H}, \underset{\mathbf{z}_i\sim p_\psi(\mathbf{z}|s_{t+H})}{\arg\max} Q(s_{t+H},\mathbf{z}_i)) \tag{5}$$

## Experiment & Results

As shown in Figure **??**, the contrastive loss did not decrease as expected. Even after adjusting factors like scale, there was still no significant change in the loss.




(a) KL / Wasserstein loss

(b) Contrastive loss

(c) DQN loss

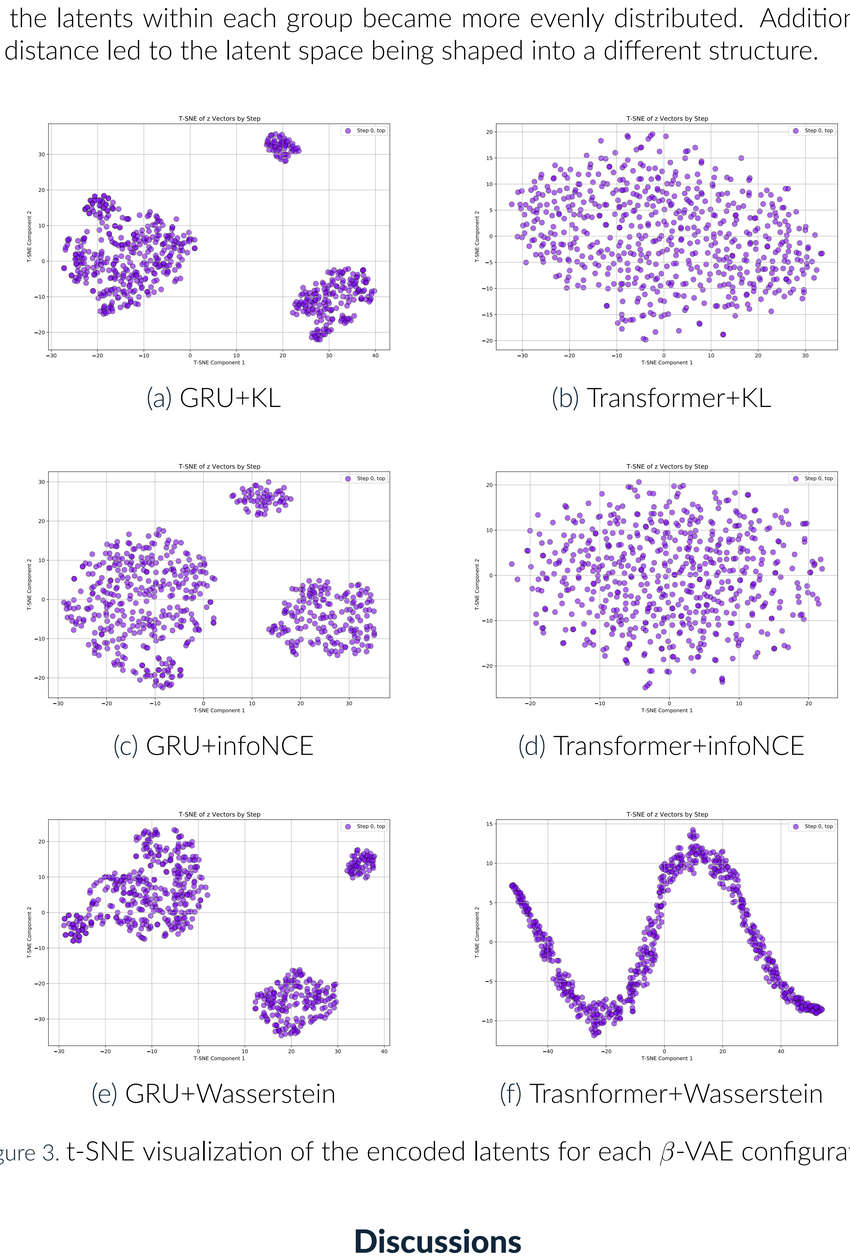
Figure 2. loss graph during train with LDCQ. (a) KL Divergence increase, while wasserstein distance decreases. (b) Contrastive loss doesn't decrease. (c) DQN loss (TD error) decrease at first, and increases after some steps.

Table **??** shows the scores of various models in Kitchen-mixed-v0. Overall, the GRU encoder shows higher performance compared to the Transformer encoder. When the InfoNCE loss was used, there was a general decline in performance, even after training the Q-Net. Similarly, the use of Wasserstein distance resulted in lower performance compared to KL divergence.

Table 1. Performance comparison on the Kitchen-mixed-v0 environment

| $\beta$-VAE configuration | Without Q | DQN step=30 | DQNstep=100 |
|---|---|---|---|
| GRU + KL (LDCQ) | $45.8 \pm 0.5$ | $54.8 \pm 2.2$ | $52.9 \pm 2.5$ |
| GRU + infoNCE | $36.7 \pm 2.2$ | $36.8 \pm 2.6$ | $37.7 \pm 3.6$ |
| GRU + Wasserstein | $27.8 \pm 3.4$ | $37.1 \pm 1.1$ | $35.8 \pm 2.8$ |
| Transformer + KL | $34.7 \pm 1.4$ | $36.3 \pm 1.8$ | $35.9 \pm 1.1$ |
| Transformer + infoNCE | $27.5 \pm 0.9$ | $28.3 \pm 0.3$ | $27.8 \pm 1.5$ |
| Transformer + Wasserstein | $37.1 \pm 3.9$ | $35.1 \pm 3.8$ | $36.2 \pm 1.4$ |

For a more detailed analysis, I visualize the latents encoded by the $\beta$-VAE using t-SNE. Figure **??** shows the results of t-SNE visualization of latent representation of the first trajectory of each episodes. On the multi-task Kitchen dataset, the Transformer encoder learned a uniformly distributed latent space, whereas the GRU encoder exhibited a tendency to cluster into several distinct groups. When InfoNCE was applied, the latents within each group became more evenly distributed. Additionally, using the Wasserstein distance led to the latent space being shaped into a different structure.




(a) GRU+KL

(b) Transformer+KL

(c) GRU+infoNCE

(d) Transformer+infoNCE

(e) GRU+Wasserstein

(f) Trasnformer+Wasserstein

Figure 3. t-SNE visualization of the encoded latents for each $\beta$-VAE configuration.

## Discussions

- Why did the $\beta$-VAE encoder with contrastive learning methods and Wasserstein distance not perform well?
  - Currently, t-SNE only considers the beginning of the episodes. When taking into account latents across all trajectories, starting from each state, the latent space may not be efficiently structured. Specifically, the latents at each step might not be well-distributed overall.
  - To apply contrastive learning effectively, it might be necessary to define positive pairs more rigorously. If only the data point itself is considered as the positive pair, the quality of the representations in the latent space may actually decline.

## Conclusions

- GRU encoder learns latents that better capture the meaning of long-horizon trajectories compared to Transformer encoder.
- Since the encoder has already sufficiently learned each trajectory, distinguishing the latents through contrastive learning seems to diminish the meaningfulness of the latent space.
- Applying the Wasserstein distance allows the Transformer encoder to use the latent space in a nonlinear way. However, the standard VAE with a GRU encoder and KL divergence outperforms this approach, showing better performance and more effective learning of the Q function for unseen latents.