VAE Matters: Latent Compression Choices for DiT Architectures

Artur B. Carneiro, Eric Liu, Sherry Xie
Stanford University
arturbc, eliu2002, ycxie @ stanford.edu

Abstract

Diffusion Transformer (DiT) models have become state-of-the-art architectures for image generation, leveraging tokenizers to compress high-dimensional images into efficient latent representations. Despite various existing tokenizer architectures, from discrete Vector Quantized Variational Autoencoders (VQ-VAEs) to continuous VAEs, the impact of tokenizer choice on DiT performance has not been extensively studied. In this project, we systematically evaluate how different VAE-based tokenizers influence DiT architectures. Specifically, we benchmark `DiT-XL` and `DiT-S` models paired with widely adopted VAEs, establishing a baseline using the default VAEs used in the DiT paper (`stabilityai/sd-vae-ft-ema` and `stabilityai/sd-vae-ft-mse`). We primarily conduct experiments with the smaller `DiT-S` model due to computational constraints, while maintaining `DiT-XL` results as reference points. Our findings provide initial insights into the tokenizer's effect on generated image quality and suggest that careful selection and tuning of tokenizers are critical for maximizing DiT performance. Future directions include comprehensive experimentation across additional VAEs and exploration of how tokenizer-induced latent-space properties correlate with final generative capabilities.

1. Introduction

Diffusion transformer models (DiT) [4] have emerged as the state-of-the-art generative models for images. A crucial component in these systems is the tokenizer, which compresses high-dimensional input data into a latent space for efficient modeling.

In the past, Vector Quantized Variational Autoencoders (VQ-VAEs) are commonly used for this purpose. As time goes on, there are several alternative tokenizers such as continuous VAEs.

However, the impact of tokenizer choice on the downstream performance remains underexplored. Therefore, our research project focuses on this exact issue: namely, how does VAE choice affect DiT model performance.

2. Related Work

In recent years, diffusion models [1] have proven to be effective in image generation tasks. There has also been increasing interest in combining the Transformer model [8], which has emerged as the paradigm for natural language processing, with diffusion models [4]. Diffusion transformers replace existing U-Net backbones with transformers to operate on images in the latent space.

The DiT paper, like Stable Diffusion [5], operates in a compressed latent space defined by a Variational Autoencoder (VAE). It uses the `kl-f8` autoencoder from LDM, which was originally trained on OpenImages.

Additionally, several improvements to this autoencoder have been developed. Notably, two fine-tuned variants, known as `ft-EMA` and `ft-MSE`, were released to enhance facial reconstruction and compatibility with Stable Diffusion training distributions. These were trained on a 1:1 mixture of LAION-Aesthetics and LAION-Humans. The `ft-EMA` model was trained based on the original checkpoint using an L1 + LPIPS loss with EMA weights, while `ft-MSE` adds an MSE term to prioritize pixel-wise fidelity. Both models retain compatibility with existing pipelines by modifying only the decoder.

Beyond these variants, numerous state-of-the-art VAE architectures have emerged. For instance, NVAE [7] and VAE-XL [6] improve expressiveness and stability by pushing most parameters into ultra-deep residual decoders that inject latents hierarchically (coarse→fine), use anti-alias upsample and self-attention for clean long-range detail, and predict sharper logistic-mixture outputs. Vector-quantized approaches like `MOVQ` [2] and `Fused VQ-VAE` [3] offer compelling alternatives by discretizing latent space for better compression and generative performance. These advances continue to shape how VAEs are used in large-scale diffusion systems.

3. Data

We use the CIFAR-10 dataset for this project because it is a small yet widely used dataset in Computer Vision. Due to computation constraints, we further sample 10% from the CIFAR-10 dataset to train on our models.

4. Methods

4.1. Key Methodology

There are two methods to explore the effect of different tokenizers with Diffusion Transformer Models:

- Fine-tune different VAEs to match the existing DiT Model Structure
- Retrain the same DiT model based on VAE variants

We aim to control the experiment to only one variable while exploring the effect of different VAEs. Therefore, the second method best fits the objective. To that end, we trained the DiT models conforming to VAE variants in our experimentation stage.

4.2. Models and Baseline Training Process

We leveraged the machine learning processes taught in class to train DiT models. Specifically, we utilized the original training file of the DiT paper code-base in our training process. Diverging from the published training process and results, we used the `DiT-S` model series due to compute constraints.

We set a learning rate of $3e-5$ with scheduler. Additionally, we trained our model with 10 epochs because we noticed significant plateauing in training loss after a few epochs and believed that 10 epochs adequate for establishing the baseline models. The experimentation results are provided in Section 5.3.

4.3. Evaluation Metrics

To fully compare the performance of our baseline DiT models and those of DiT models with different VAEs, we selected the following five evaluation metrics across two domains.

1. **Latent Space Data**
   Using different VAEs will shift the latent distribution as inputs for DiTs. As a result, latent spcace related metrics will help track and illuminate how different VAEs impact DiTs' performances.

   (a) Latent Norm. This metric gives us the average L2 norm of latent vectors. Below is the latent norm equation we used:

$$\text{Latent Norm} = \frac{1}{N}\sum_{i=1}^{N}\|z_i\|_2 \tag{1}$$

   Where:

   - $\|z_i\|_2$ calculates the L2 norm of vector z_i

   (b) Latent standard deviation. This metric measures the spread of latent vectors. Below is the latent standard deviation equation we used:

$$\text{Latent Std} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\|z_i - \bar{z}\|_2^2} \tag{2}$$

   Where:

   - z_i is the i-th latent vector
   - $\bar{z}$ is the mean latent vector

2. **Image Quality Metrics**
   We include three different metrics to evaluate both the pixel level comparisons and the structural level comparisons of our generated images and original images.

   (a) PSNR (Peak Signal-to-Noise Ratio). It measures the quality of reconstructed images compared to the original on the pixel level. It allows us to quantify the distortion introduced during the reconstruction process. Below is the equation for PSNR we used in evaluation:

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{\text{MAX}^2}{\text{MSE}}\right) \tag{3}$$

   Where:

   - **MAX** is the maximum possible pixel value
   - **MSE** is the mean squared error:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(I_i - \hat{I}_i)^2 \tag{4}$$

   Where:

   - N is the total number of pixels in the image
   - I_i is the value of the i-th pixel in the ground truth image
   - $\hat{I}_i$ is the value of i-th pixel in the generated image

   (b) SSIM (Structural Similarity Index). This metric measures the perceptual similarity between original and reconstructed images rather than the pixel-level differences. Below is the equation we used in calculating SSIM:

$$\text{SSIM}(x,y) = [l(x,y)]^{\alpha} \cdot [c(x,y)]^{\beta} \cdot [s(x,y)]^{\gamma} \tag{5}$$

   Where:

   - $l(x,y)$ compares luminance

- $c(x, y$ compares contrast
- $s(x, y)$ compares structure
- α, β, and γ are weights

(c) LPIPS (Learned Perceptual Image Patch Similarity). This metric uses deep features from pretrained neural networks to measure perceptual similarity between images. Unlike PSNR and SSIM, LPIPS better aligns with human perceptual judgments by comparing images in a learned feature space. Below is the equation we used for calculating LPIPS:

$$\text{LPIPS}(x, \hat{x}) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} \|w_l \odot \delta_l^{hw}\|_2^2 \quad (6)$$

Where:

- $\delta_l^{hw} = \phi_l(x)_{hw} - \phi_l(\hat{x})_{hw}$ is the feature difference
- ϕ_l denotes the features from layer l of a pretrained network
- H_l and W_l are the height and width of the feature maps at layer l
- w_l represents the learned perceptual weights for layer l
- x is the original image and $\hat{x}$ is the generated image.

4.4. Retrain DiT with VAE variants

After establishing our baseline, we retrained the `DiT-S` models using different VAE variants to explore the effect of different VAEs on the same DiT model architecture.

5. Experiments

5.1. Exploration

We began by exploring different DiT models provided in the paper [4]. We chose the XL model as it is widely available on Huggingface and directly downloadable. We chose the VAE `stabilityai/sd-vae-ft-ema` cited in the paper. We ran inference and generated images using the specified model and VAE to explore its capabilities. Our sample images from this model are shown in Figure 1.

Since the model was trained on the default VAE, we were able to run the inference smoothly. In further exploration, we experimented with the running of the same model using another VAE `stabilityai/sd-vae-ft-mse`. We noticed similar results to the baseline we established above.

5.2. Model Selection and Data Sampling

Due to our compute capabilities, we are not able to retrain models with differnt VAEs using the `DiT-XL` models.

Figure 1: Baseline Sample Image generated by `DiT-XL-2-256x256` model

.

Instead, we chose to conduct all our experiments using the `DiT-S` models. However, since the weights for the `DiT-S` models are not released in the paper, we also needed to train the `DiT-S` model with the VAEs the paper provided as a baseline.

To further improve the efficiency of our training while experimenting process, we sampled the training data by conducting training on only 10% of the dataset, as previously explained in Section 3.

5.3. Baseline Experiments

We trained the `DiT-S` models against 2 VAEs provided by the original paper: `stabilityai/sd-vae-ft-ema` and `stabilityai/sd-vae-ft-mse`. Using the hyperparameters previously discussed in Section 4.2, we trained our `DiT-S` models.

A summary of the performances of our baseline models against the four evaluation metrics mentioned in section 4.3 is included in the following tables:

VAE	Latent Norm	Latent STD
ft-EMA	1.33782	0.64034
ft-MSE	1.33439	0.63853

Table 1: `DiT-S` model performances with Latent Norm and Latent Standard Deviation metrics.

VAE	PSNR	SSIM	LPIPS
ft-EMA	9.81	0.6383	0.0530
ft-MSE	10.62303	0.66282	0.0579

Table 2: `DiT-S` model performances with PSNR and SSIN metrics

We include the final training loss of these two baseline models in Table 3 for further comparison.

VAE	Final Loss
ft-EMA	0.224701
ft-MSE	0.208439

Table 3: Final training loss of `DiT-S` models

5.4. VAE Variants

To explore the effect of different tokenizers with Diffusion Transformer Models, we selected a variety of different VAEs to run experiments against. We chose the following two VAE variants:

1. `ostris/vae-kl-f8-d16`: We selected this VAE because it is lighter weight, faster, and trained on a variety of images. Exploring this lighter weight option may be useful for future DiT training when there is compute constraint.

2. `stabilityai/sdxl-vae`: We selected this VAE because this is the official SDXL encoder which will give us a rich latent space.

Similar to our baseline experiments, we also include the performances of the DiT models trained on these variants against the four evaluation metrics mentioned in section 4.3 in the following tables:

VAE	Latent Norm	Latent STD
ostris/vae-kl-f8-d16	1.39753	0.6516
sdxl-vae	1.33439	0.63853

Table 4: `DiT-S` model performances on VAE variants with Latent Norm and Latent Standard Deviation metrics.

VAE	PSNR	SSIM	LPIPS
ostris/vae-kl-f8-d16	12.32	0.7155	0.0205
sdxl-vae	9.29641	0.59034	0.1356

Table 5: `DiT-S` model performances on VAE variants with PSNR and SSIN metrics

VAE	Final Loss
ostris/vae-kl-f8-d16	0.1272484944968284
sdxl-vae	0.2357452244886869

Table 6: Final loss of `DiT-S` models trained on VAE variants

6. Analysis

6.1. Training Loss Visualization for All VAEs

In order to fully analyze the performances of these different models, we first include the following figure for all the training losses.

Figure 2: Training Losses for all the DiT models

.

We notice that the overall shape and trend of the training process are very similar, with the initial training loss starting from around 0.95 and the final training loss ending at around 0.20 except for the DiT model trained on the `ostris/vae-kl-f8-d16` VAE, which ends at around 0.12.

This helps us get a general sense that all these VAEs are suitable VAEs for the DiT model architecture and shows that there will be minor, fine-grained differences in the effect of different VAEs on DiTs rather than bigger and more dramatic differences.

6.2. Baseline Model Analysis

We analyze the baseline model performances by looking at the PSNR, SSIM and LPIPS metrics generated in Table 5. Since the PSNR is based on the pixel-wise error and we have the MSE term in the denominator of the equation for PSNR, a higher PSNR means that the generated image is closer to the original image. SSIM measures the structural similarity between the generated image and original image. SSIM value of 1 would represent the generated image is the exact reconstruction of the original image. Finally, LPIPS measures how far the generated image is from the original image, so a lower LPIPS value is better. Looking at Table 5, we can see that the performances of both VAEs used here are pretty similar, with `stabilityai/sd-vae-ft-mse` performing marginally better in the PSNR and SSIM metrics while `stabilityai/sd-vae-ft-ema` is performing marginally better in the LPIPS metric.

6.3. VAE Variant Analysis

When looking at the PSNR, SSIM, and LPIPS metrics for the two VAE variants, we see that the DiT with the `ostris/vae-kl-f8-d16` VAE is performing significantly better than the `stabilityai/sdxl-vae` VAE in all three metrics. Notably, the LPIPS value generated by the DiT trained on `ostris/vae-kl-f8-d16` is over 6 times lower than that of the DiT trained on `stabilityai/sdxl-vae`. To visualize this difference we see, we also ran inference on both DiT models trained on these VAEs. Specifically, we ask the model to reconstruct some original images in the dataset. The reconstructed images are included in the figures below with the first row being the original images and the second row being the images reconstructed by DiT models trained on different VAEs.

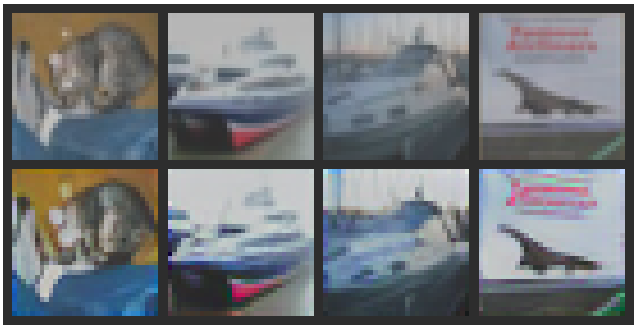

Figure 3: Original and reconstructed images generated by DiT trained on `ostris/vae-kl-f8-d16`

.

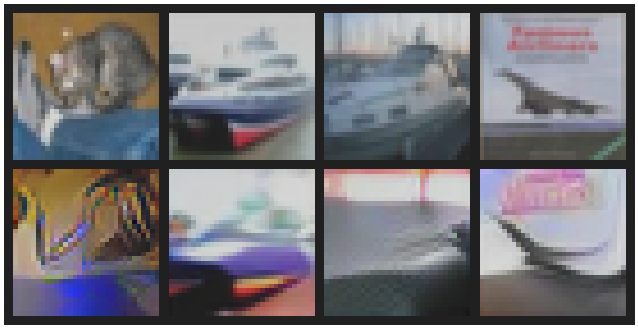

Figure 4: Original and reconstructed images generated by DiT trained on `stabilityai/sdxl-vae`

.

As we observed in Figure 3 and Figure 4, the DiT trained on `ostris/vae-kl-f8-d16` reconstructed images more clearly both on a structural and on a pixel level, with more visibly similar colors and structurally consistent shapes. This is aligned with the numerical results we observed earlier.

Notably, the DiT with the `ostris/vae-kl-f8-d16` VAE is performing even better than the default VAEs we selected both on the pixel level and on the structural level. Figure 5 shows the reconstructed images conducted by the DiT trained on `stabilityai/sd-vae-ft-mse` and Figure 6 shows the reconstructed images conducted by the DiT trained on `stabilityai/sd-vae-ft-ema`. We can see that on both the structural and pixel level, the default VAEs are generating worse results than the DiT trained on the `ostris/vae-kl-f8-d16` VAE. Specifically, Figure 3 offers reconstructed images that are sharper, clearer, and have a more similar overall structure to the original images on the first row. This reflects and supports the numerical values we saw with all three metrics.

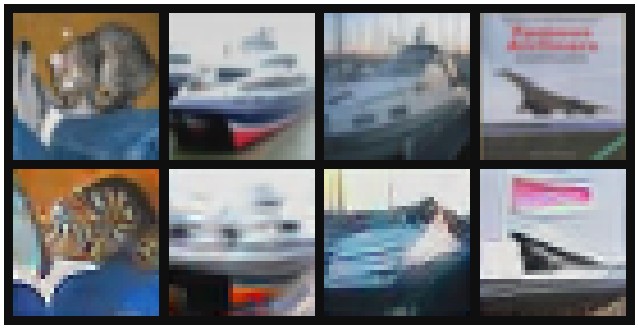

Figure 5: Original and reconstructed images generated by DiT trained on `stabilityai/sd-vae-ft-mse`

.

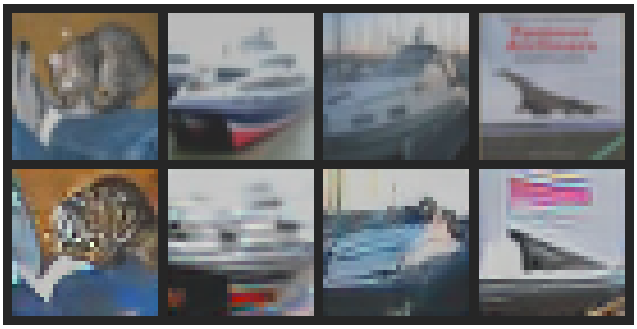

Figure 6: Original and reconstructed images generated by DiT trained on `stabilityai/sd-vae-ft-ema`
.

Even though this VAE is actually lighter weight, the fact that it is trained on a balance of photos, text, cartoons, and vector images contributes to the diversity of this VAE and may have been the reason behind its stellar performance.

6.4. Overall Analysis

Looking at the DiT models trained on both the default VAEs and the VAE variants, we can see that there is a variety of performances from different VAEs when applied to the same DiT model architecture. This reflects that the selection of VAEs does play a role in the effectiveness of the DiT model performance.

7. Conclusion

We trained `DiT-S` models using both default VAEs provided by the original DiT paper [4] and other related VAEs to explore the effect of VAEs on the performance of DiT models. We noticed that the specific VAE we select does have a strong influence on both the pixel-level and structural level performance of generated images. Specifically, one especially light-weight yet diverse VAE performed better at both metrics than the default VAEs given to us by the DiT paper. Moving forward, developing light-weight yet useful VAEs could be a promising direction, exploring whether we can improve DiT performance even in an extremely resource-limited setting.

References

[1] P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis, 2021. 1

[2] P. Esser, R. Rombach, A. Blattmann, and B. Ommer. Image compression with hierarchical latent vector quantization, 2023. 1

[3] J. H. Lee, Y. J. Ko, and Y. Gwon. Fused vq: Efficient image representation learning for compression and generation, 2022. 1

[4] W. Peebles and S. Xie. Scalable diffusion models with transformers, 2023. 1, 3, 6

[5] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022. 1

[6] K. Sohn, D. Cai, C.-L. Li, T. Chen, H. Zhang, J. Lee, S. Levine, and M. Norouzi. Vae-xl: Scaling up latent space models with product quantization, 2023. 1

[7] A. Vahdat and J. Kautz. Nvae: A deep hierarchical variational autoencoder. *Advances in Neural Information Processing Systems*, 33:19667–19679, 2020. 1

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023. 1