# Noise2Art with Latent Diffusion Models

Rami Matar
*Dept. of Computer Science*
*Columbia University*
New York City, USA
rhm2142@columbia.edu

Mustafa Eyceoz
*Dept. of Computer Science*
*Columbia University*
New York City, USA
me2680@columbia.edu

Justin Lee
*Dept. of Computer Science*
*Columbia University*
New York City, USA
jjl2245@columbia.edu

GitHub Repo Link
github.com/Maxusmusti/
latent-diffusion-exp

## I. Introduction

### A. Generative Modeling

Generative modeling is the task of learning to generate data that could have risen in the true underlying data distribution. Neural network methods completely revolutionized generative tasks, with architectures like the variational autoencoder (VAE) and generative adversarial networks (GANs) [4] [8]. Those networks have shown impressive results and generate incredibly realistic data in many tasks.

In general, generative neural networks learn a mapping $f : z \to x$ from a simple distribution $z$, to the underlying data distribution $x$. Typically, $z$ is chosen to be a standard gaussian distribution for its compactness and form. The VAE [8] is trained using a combination of a reconstruction loss and a regularization term. The reconstruction loss ensures that the generated data is similar to the input data, while the regularization term encourages the latent space to be structured and follow the prior distribution (typically a standard Gaussian). Specifically, the VAE learns an encoder $q_\phi(z|x)$, which approximates the true posterior $p(z|x)$, and a decoder $p_\theta(x|z)$, which generates data samples given a latent code $z$. The objective function, called the Evidence Lower Bound (ELBO) optimizes the Kullback-Leibler divergence, a measure of the dissimilarity of two distributions. The optimization of this objective ensures that the learned representations are both expressive and compact.

On the other hand, GANs [4] are trained using a min-max adversarial game between a generator $G$ and a discriminator $D$. The generator learns to generate realistic data samples, while the discriminator learns to distinguish between real and generated samples. This adversarial training procedure has been shown to produce high-quality samples, but it also has several well-known issues. One such problem is mode collapse, where the generator learns to generate only a limited variety of samples, ignoring many modes in the true data distribution [16]. This occurs when the generator finds a way to consistently fool the discriminator, causing it to focus on producing similar samples rather than exploring the diversity of the data distribution.

Another challenge in training GANs is the unstable training dynamics. The adversarial game between the generator and the discriminator can lead to oscillations and non-convergence. This is because the generator and discriminator are essentially competing against each other, with each one trying to outperform the other. As a result, their updates can destabilize the learning process.

Despite the remarkable achievements of VAEs and GANs in generative modeling, each method has its own set of limitations. VAEs tend to produce blurry reconstructions due to the use of an explicit reconstruction loss, while GANs face challenges such as mode collapse and unstable training dynamics, stemming from the adversarial nature of their training procedure [16] [4] [8].

### B. Diffusion Models

In response to these challenges, diffusion models have recently emerged as a promising area of research, gaining significant traction in the field of generative modeling [15] [5] [11] [1]. The key difference in the training of diffusion models is that they rely on a denoising process, where a series of noise-corrupted samples are progressively denoised to recover the original data. This approach has several advantages over traditional VAE and GAN techniques, such as stable training and the ability to capture diverse modes in the data distribution.

Diffusion models can be formally described as a Markov process, where the data is progressively corrupted by noise over a sequence of time steps. In the forward pass, the data $x_0$ is transformed into a sequence of increasingly noisy versions, $x_1, x_2, \ldots, x_T$, following a noise schedule. At each time step $t$, the process can be defined by:

$$x_t = \sqrt{1 - \beta_t} x_{t-1} + \sqrt{\beta_t} \epsilon_t, \tag{1}$$

where $\epsilon_t \sim \mathcal{N}(0, I)$ is Gaussian noise, and $\beta_t$ is a noise schedule parameter that controls the amount of noise introduced at each step. The noise schedule is chosen such that the final corrupted sample $x_T$ is almost entirely noise.

The reverse process, also referred to as the denoising process, aims to recover the original data $x_0$ from the noisy version $x_T$. In order to learn this reverse process, a neural network, typically a conditional denoising score-matching estimator $\epsilon_\theta(x_t, t)$, is trained to predict the noise at step $t \leq T$.

By learning this reverse process, the model essentially learns to generate data by reversing the "destruction" of the data through the noise corruption. This approach is intuitive, as it allows the model to capture the data distribution by

learning how to recover the data from increasingly challenging corruptions.

As shown in the DDPM paper, we can train diffusion models to predict noise directly using a simplified version of the variational lower bound. The simplified loss function, $L(\theta)$, is defined as:

$$L(\theta) := \mathbb{E}t, x_0, \epsilon \left[ \frac{(\epsilon - \epsilon_\theta(\sqrt{\alpha_t}x_0 + \sqrt{1-\alpha_t}\epsilon, t))^2}{2} \right]. \quad (2)$$

Here, $t$ is uniform between 1 and $T$. This simplified objective discards the weighting in the standard variational bound, resulting in a weighted variational bound that emphasizes different aspects of reconstruction. In particular, the diffusion process setup causes the simplified objective to down-weight loss terms corresponding to small $t$. These terms train the network to denoise data with very small amounts of noise, so it is beneficial to down-weight them, allowing the network to focus on more difficult denoising tasks at larger $t$ terms. The experiments in the DDPM paper demonstrate that this reweighting leads to better sample quality.

By learning this reverse process, the model essentially learns to generate data by reversing the "destruction" of the data through the noise corruption. This approach is intuitive, as it allows the model to capture the data distribution by learning how to recover the data from increasingly challenging corruptions.

Despite the advantages of diffusion models in terms of stability and diversity in the generated data, the computational cost associated with them can be prohibitive. This is due to the large number of denoising steps required to generate high-quality samples, which has motivated the exploration of more efficient techniques, such as latent diffusion models.

### C. Latent Diffusion Models

Latent diffusion models address the computational challenges of diffusion models by introducing a latent space representation to guide the denoising process. In these models, a latent code $z_t$ is extracted from the noisy data $x_t$, which is then used to condition the denoising process. By leveraging this latent space representation, latent diffusion models can generate samples more efficiently, while retaining the benefits of diffusion models in terms of stability and diverse generation. This makes latent diffusion models a promising solution to the cost of training diffusion models.

Transitioning the diffusion problem into the latent space enables the generation of high-quality data using diffusion models without the prohibitive costs typically associated with them [12]. In the latent space, the denoising process can operate on much smaller spatially reduced images and still generate high-resolution images through the decoder. In our project, we focus on learning a diffusion model that operates directly on the latent space of a vector-quantized variational autoencoder (VQ-VAE), which provides a compact representation of our images in its latent space.

To achieve this, we adapt the denoising process described in the previous section to work within the context of latent diffusion models. We first learn a VQ-VAE that maps the input data to a discrete latent space. The latent codes, $z_t$, corresponding to the noisy data $x_t$, are obtained by encoding the noisy images through the VQ-VAE's encoder. The diffusion process is then performed in the latent space, operating on the latent codes $z_t$.

The forward and reverse processes are adapted to work with the latent codes as follows:

Forward process:

$$z_t = \sqrt{1-\beta_t}z_{t-1} + \sqrt{\beta_t}\epsilon_t, \quad (3)$$

where $\epsilon_t \sim \mathcal{N}(0, I)$ is Gaussian noise, and $\beta_t$ is a noise schedule parameter that controls the amount of noise introduced at each step in the latent space.

Reverse process:

In order to learn the reverse process in the latent space, a neural network, typically a conditional denoising score-matching estimator $\epsilon_\theta(z_t, t)$, is trained to predict the noise at step $t \leq T$ in the latent space. The simplified loss function, $L(\theta)$, is defined as:

$$L(\theta) := \mathbb{E}_{t, z_0, \epsilon} \left[ \frac{(\epsilon - \epsilon_\theta(\sqrt{\alpha_t}z_0 + \sqrt{1-\alpha_t}\epsilon, t))^2}{2} \right] \quad (4)$$

By learning the reverse process in the latent space, the model can generate high-quality data through a more efficient denoising process, while still benefiting from the advantages of diffusion models in terms of stability and diverse generation. This combination of latent space representation and diffusion models represents a powerful approach to generative modeling, particularly for generating complex and high-resolution data such as artistic images.

## II. Prior Work and State of the Art

Diffusion models have experienced significant progress over the past few years. One of the earliest works on diffusion models was introduced by Sohl-Dickstein et al. in their 2016 paper titled "Deep Unsupervised Learning using Nonequilibrium Thermodynamics" [15]. This work laid the foundation for the denoising score matching framework and demonstrated the potential of diffusion models in generative modeling.

The Denoising Diffusion Probabilistic Models (DDPM) paper by Ho et al. [5] brought diffusion models to the forefront of generative modeling. DDPM introduced a more efficient and stable training procedure by leveraging a conditional denoising score-matching estimator, which significantly improved the quality of generated samples.

Following DDPM, OpenAI published two influential papers that pushed the boundaries of diffusion models. The first paper, "Improved Denoising Diffusion Probabilistic Models" [11], enhanced the training stability and sample quality by proposing novel architecture choices and improved loss functions. The second paper, "Diffusion Models Beat GANs on Image

Synthesis" [1], demonstrated that diffusion models could outperform state-of-the-art GANs in terms of image synthesis, establishing diffusion models as a competitive alternative to GAN-based approaches.

A recent work by Rombach et al., titled "High-resolution Image Synthesis with Latent Diffusion Models" [12], presents an important development in the field of diffusion models. This paper proposes a latent diffusion model that combines the benefits of diffusion models with the efficiency of latent space representations. By leveraging a latent code to guide the denoising process, the authors were able to generate high-resolution images more efficiently without sacrificing sample quality. The work by Rombach et al. [12] serves as the inspiration for our project, and we have largely based our approach on their paper and model.

The latent diffusion models follow a two-stage training process. In the first stage, an autoencoder is trained to learn a compact latent representation of the data. The autoencoder consists of an encoder $q_\phi(z|x)$, which maps the data $x$ to a latent code $z$, and a decoder $p_\theta(x|z)$, which reconstructs the data from the latent code. The autoencoder is trained by minimizing the reconstruction loss, typically using a combination of a pixel-wise loss and a perceptual loss:

$$\begin{aligned} L_{AE}(\theta, \phi; x) =& L_{pixel}(x, p_\theta(q_\phi(x))) \\ &+ \lambda L_{perceptual}(x, p_\theta(q_\phi(x))) \end{aligned} \tag{5}$$

where $\lambda$ is a weighting factor that balances the contribution of the two loss terms.

Once the autoencoder is pretrained, the second stage involves learning the diffusion process while utilizing the pretrained autoencoder. The forward diffusion process is performed in the latent space, with the latent codes $z_t$ being perturbed by noise:

$$z_t = \sqrt{1 - \beta_t} z_{t-1} + \sqrt{\beta_t} \epsilon_t, \tag{6}$$

where $\epsilon_t \sim \mathcal{N}(0, I)$ is Gaussian noise, and $\beta_t$ is a noise schedule parameter that controls the amount of noise introduced at each step.

The denoising process is conditioned on the noisy latent codes $z_t$ instead of the noisy data $x_t$. The denoising model $epsilon_\theta(z_t, t)$ is trained to predict the noise $\epsilon_t$ that was added at each step in the latent space. During the denoising process, the model estimates the noise added at each step and subtracts it from the noisy latent codes, progressively recovering the original latent code. Once the original latent code is recovered, the decoder can reconstruct the original data from it.

By using the pretrained autoencoder to guide the denoising process in the latent space, the latent diffusion model can generate high-quality samples more efficiently compared to traditional diffusion models, offering a promising direction for further research and development in generative modeling.

The forward pass can be represented in a compact way at an arbitrary timestep $t$ in closed form. This representation is based on the property that the sum of gaussians can be modeled as a gaussian, and it immensely speeds up the forward process calculation, improving the efficiency of diffusion models.

To achieve this representation, we define $\alpha_t = 1 - \beta_t$, and $\bar{\alpha_t} := \prod_{s=1}^{t} \alpha_s$. Then, the forward pass can be expressed as

$$x_t = \sqrt{\bar{\alpha_t}} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$$

## III. Approach and Architecture

Our architecture comprises two main components, trained in two separate stages. First, a VQGAN is employed to obtain a latent representation by transforming full-sized images into compressed images with compact latent representations. The architecture processes an input image $x_t \in \mathbb{R}^{3 \times H \times W}$ and employs a VQGAN encoder to transform it into a latent space representation $z_t \in \mathbb{R}^{n_z \times h \times w}$.

In the second stage, we learn the diffusion process on this $z_t$ representation using a time-conditioned U-Net model, which serves as the denoising diffusion probabilistic model (DDPM). This U-Net model is designed to learn the conditional distribution $p(z_{t-1}|z_t)$, which guides the denoising process in the latent space. By conditioning the U-Net model on time, we enable it to adapt to the noise schedule, effectively capturing the underlying data distribution in the latent space at different denoising stages. Once the diffusion model is trained, it can be used to generate new valid latent representations, which capture the diversity and structure of the original data distribution. These generated latent representations can then be decoded using the VQGAN decoder to obtain synthetic images that resemble samples from the true data distribution.

We will provide a detailed explanation of the VQGAN and U-Net architectures in Sections 3.1 and 3.2, respectively. By combining the VQGAN's latent space representation with the diffusion process and leveraging the time-conditioned U-Net model, our architecture efficiently generates high-quality images while preserving the stability and diversity of diffusion models.

### A. VQGAN

The VQGAN (Fig. 1) is designed to leverage the highly expressive transformer architecture for image synthesis by expressing the constituents of an image in the form of a sequence. Instead of using individual pixels, a discrete codebook of learned representations is used, such that any image $x \in \mathbb{R}^{H \times W \times 3}$ can be represented by a spatial collection of codebook entries $z_q \in \mathbb{R}^{h \times w \times n_z}$, where $n_z$ is the dimensionality of codes.

To effectively learn a discrete spatial codebook, the VQGAN uses a convolutional model consisting of an encoder $E$ and a decoder $G$. They learn to represent images with codes from a learned, discrete codebook $Z = \{z_k\}_{k=1}^{K} \subset \mathbb{R}^{n_z}$. The model approximates a given image $x$ by $\hat{x} = G(z_q)$. The quantized representation $z_q$ is obtained using the encoding $\tilde{z} = E(x) \in \mathbb{R}^{h \times w \times n_z}$ and a subsequent element-wise quantization $q(\cdot)$ of each spatial code $\tilde{z}ij \in \mathbb{R}^{n_z}$ onto its closest codebook entry $z_k$:

$$z_q = q(\tilde{z}) := \arg\min_{z_k \in Z} |\tilde{z}_{ij} - z_k| \in \mathbb{R}^{h \times w \times n_z}. \tag{7}$$

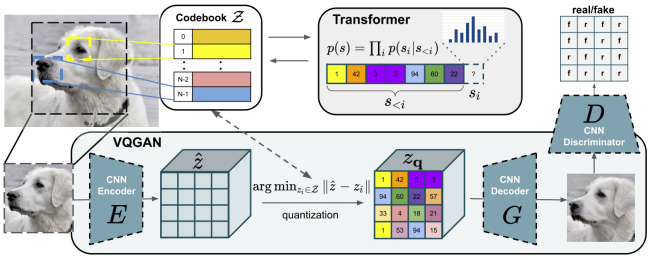


Fig. 1. The VQGAN architecture illustration from the Taming Transformers paper showcasing the three components, an encoder, a codebook, and a decoder. Our latent representation is that of the codebook, although we use a continuous version rather than a quantized one. [2]

The reconstruction $\hat{x} \approx x$ is then given by

$$\hat{x} = G(z_q) = G(q(E(x))). \tag{8}$$

The VQGAN is trained end-to-end with a loss function $L_{VQ}(E, G, Z)$, which includes three components: a reconstruction loss $L_{rec}$ (using a perceptual loss instead of the L2 loss), a commitment loss, and a discriminator loss. The non-differentiable quantization operation in the model is addressed by using a straight-through gradient estimator.

To learn a rich codebook and maintain good perceptual quality at increased compression rates, the VQGAN introduces an adversarial training procedure with a patch-based discriminator $D$. The adversarial loss $L_{GAN}(E, G, Z, D)$ aims to differentiate between real and reconstructed images:

$$L_{GAN}(E, G, Z, D) = \log D(x) + \log(1 - D(\hat{x})). \tag{9}$$

The complete objective for finding the optimal compression model $Q' = E', G', Z'$ is given by:

$$Q' = \arg \min_{E,G,Z} \max_{D} \mathbb{E}_{x \sim p(x)} [L_{VQ}(E, G, Z) + \lambda L_{GAN}(E, G, Z, D)] \tag{10}$$

where $\lambda$ is an adaptive weight computed according to the gradients of the perceptual reconstruction loss $L_{rec}$ and the adversarial loss $L_{GAN}$.

When designing the model, we consider the compression ratio we want considering a tradeoff in quality and computation performance. We typically try to choose a small value for $n_z$ to achieve a high degree of compression, but the most important factor is the spatial compression. The paper experimented and found the best values to be $4, 8, 16$ for $c = \frac{H}{h} = \frac{W}{w}$. We chose the ratio to be 8 for a significant improvement in the forward pass of the UNet. To quantify the significance of the compression, this means the latent space is $\frac{1}{64}$ as spatially expensive as the pixel space for our diffusion model. The number of channels also is chosen to be low by allowing a large number of codebook vectors.

### *B. UNet with Self-Attention*

The architecture of our latent diffusion model is based on the UNet, a well-known and widely used architecture for image segmentation and other tasks requiring dense prediction

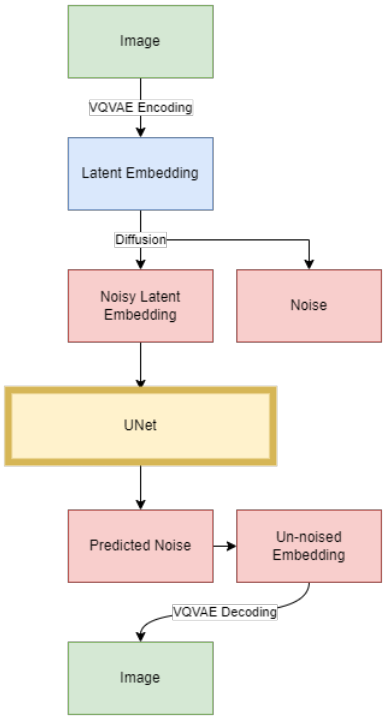


Fig. 2. The architecture of the UNet, showing our training process starting from receiving the input image, then obtaining a latent representation from the VQGAN encoder; then, we perform the forward pass of the diffusion model using our noise schedule. Lastly, the UNet is tasked with predicting the noise, which we can use to obtain a clean latent representation of the image, which is transformed into the image space using the VQGAN decoder.

[13]. The UNet consists of an encoder-decoder structure, with skip connections between corresponding layers of the encoder and decoder. This allows the model to preserve spatial information and better reconstruct the input image. To improve the performance of the UNet, we incorporate self-attention mechanisms [17] into the architecture (Fig. 2).

The self-attention mechanism enables a model to weigh the importance of different elements in a sequence, based on their relevance to the current element. In the context of our image generation task, this allows the model to attend to different spatial locations in the input image based on their significance for the output. The self-attention mechanism can be formally defined as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V, \tag{11}$$

where $Q$, $K$, and $V$ are the query, key, and value matrices, respectively, and $d_k$ is the dimension of the key vectors.

In our latent diffusion model, we use the enhanced UNet to predict the input image in the latent space given a noised latent code at step $t \leq T$. The model learns to approximate the joint distribution of the original image $x$ and the noised latent code $z_t$ at each time step. The notation commonly used in diffusion models includes $p(z_t|x)$ and $q(x|z_t)$, representing the probability of the noised latent code given the image and

the probability of the image given the noised latent code, respectively.

To condition the UNet on the noisy input and the time step, we employ a sine and cosine positional embedding, as described in the "Attention is All You Need" paper [17]. Specifically, the positional encoding for each time step $t$ is computed as:

$$\begin{aligned} PE_{(t,2i)} &= \sin(\frac{t}{10000^{2i/d}}) \\ PE_{(t,2i+1)} &= \cos(\frac{t}{10000^{2i/d}}), \end{aligned} \tag{12}$$

where $i$ ranges from 0 to $d/2-1$, and $d$ is the dimension of the embedding space. This positional encoding is then used to modulate the UNet's features, allowing the model to adapt its behavior based on the current time step.

The time step conditioning and the predetermined noise schedule are crucial for the performance of denoising diffusion probabilistic models (DDPMs) [5]. The noise schedule is a sequence of noise levels $\sigma_t$ for each time step $t$, determining the amount of noise added at each step. The model is trained to denoise the images while maintaining the fidelity of the original image, with the noise schedule and time step conditioning guiding the denoising process.

In summary, our approach involves using an enhanced UNet with self-attention to predict the input image in the latent space, given a noised latent code at a specific time step. The UNet is conditioned on the noisy input and the time step through the use of a timestep embedding. This conditioning, combined with the predetermined noise schedule, is crucial for the success of denoising diffusion probabilistic models.

### C. Conditional Generation

In our approach, we enhance the model by conditioning the generated images on a pretrained domain-specific encoder's encoding. We extend the UNet with the encoding and the time step. We project the encoding to the dimension of the specific layer in the UNet and concatenate it with the time step embedding. To train this model, we need to introduce pair datasets such as image-caption datasets.

To train the model without relying on gradients from an image classifier, we employ Classifier-Free Guidance (CFG) as introduced in the original paper [6]. With CFG, we modify the loss function to incorporate the pretrained domain-specific experts. The domain-specific experts are pretrained and fixed, and we do not train them during the training process.

The training algorithm for conditional generation with CFG is as follows:

1) Train an unconditional denoising diffusion model $p_\theta(z)$ parameterized through a score estimator $\epsilon_\theta(z_\lambda)$ together with the conditional model $p_\theta(z|c)$ parameterized through $\epsilon_\theta(z_\lambda, c)$.
2) Use a single neural network to parameterize both models. For the unconditional model, simply input a null token $\emptyset$ for the class identifier $c$ when predicting the score, i.e., $\epsilon_\theta(z_\lambda) = \epsilon_\theta(z_\lambda, c = \emptyset)$.
3) Jointly train the unconditional and conditional models by randomly setting $c$ to the unconditional class identifier $\emptyset$ with some probability $p_{\text{uncond}}$, set as a hyperparameter.
4) During sampling, perform the following linear combination of the conditional and unconditional score estimates: $\tilde{\epsilon_\theta}(z\lambda, c) = (1+w)\epsilon_\theta(z_\lambda, c) - w\epsilon_\theta(z_\lambda)$, where $w$ is the guidance strength.

In summary, we extend the UNet with a domain-specific encoder and a sine and cosine positional embedding to condition the generation process on both the noisy input and the time step. By using pair datasets and Classifier-Free Guidance, we can train the model to generate images that are consistent with both the input image and an additional condition such as a caption. This makes our approach flexible and easy to train while maintaining the benefits of denoising diffusion probabilistic models.

## IV. Dataset and Features

### A. Initial Dataset

We used the WikiArt dataset [3] from Hugging Face which contains a total of approximately 100,000 art images of various sizes retrieved from WikiArt.org. The art in the dataset comes from various artists, including Vincent Van Gogh, Rembrandt, and Renoir. The dataset contains the images as well as the artist data, genre data (landscape, portrait, still-life, etc.), and style data (Realism, Romanticism, Baroque, etc.).

We randomly split the WikiArt dataset into a train set and a validation set, taking 95% as training data and 5% as validation data. Then, the WikiArt images were preprocessed by scaling the pixel values to a range from -1 to 1. Finally, the images were resized and cropped at the center, resulting in images of size 256x256 regardless of the original image dimensions.

We used the WikiArt dataset for the training of our latent diffusion model for both general and class-conditional training. Using CFG, we can generate conditional input based on specific metadata such as genre or artist. When training without conditional input, we simply use the images from the WikiArt dataset without any labels.

### B. Captioned Dataset

We also used the LAION-art dataset [9] from Hugging Face which is an 8-million image subset of the LAION-5B dataset [10]. LAION-5B is a 5-billion image dataset popular for training large models like diffusion models and contains the images as well as captions, watermark probability which ranges from 0 to 1, unsafe probability which ranges from 0 to 1, and aesthetic scores which ranges from 1 to 10. The LAION-art dataset was created by filtering the LAION-5B dataset down to only the images with an aesthetic rating higher than 8, watermark probability less than 0.8, and unsafe probability less than 0.5. Most importantly, the LAION-art dataset retains the captions of each image.

We randomly split the LAION-art dataset into a train set and a validation set, taking 95% as training data and 5% as validation data. Then, the LAION-art images were

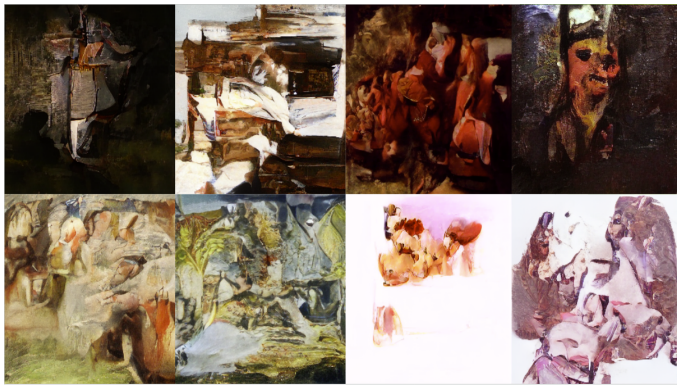

Fig. 3. At more stable points in training, the model was able to sample high-quality artistic images. The model was not able to learn to produce realistic outputs, perhaps due to the limited training time or the variety of the dataset given its small size.

Fig. 4. The EMA model results were on average significantly higher quality and were able to start generating more complex features such as faces (although clearly not well yet).

preprocessed by scaling the pixel values to a range from -1 to 1. Finally, the images were resized and cropped at the center, resulting in images of size 256x256 regardless of the original image dimensions.

We used the LAION-art dataset for caption-based conditional image generation by making a conditional diffusion model trained on this dataset.

### C. Caption Text Embeddings

When building a model for conditional image generation using captions, we needed to transform sentences into text embeddings. For this, we used the SentenceTransformers [14] Python framework for state-of-the-art sentence and text embeddings which was built on SentenceBERT Architecture. In particular, we used the all-MiniLM-L6-v2 pretrained model which is a very fast model that can achieve over 14-thousand sentences per second inference while still achieving comparable performance to DistilRoBERTa and MPNet Base V2. The model maps sentences to a 384-dimension dense vector space.

## V. Results

Our results training the model without were initially very unstable, and the network struggled with converging at all. The loss was extremely unstable because of the stochasticity of the denoising process, and it was initially unclear how well our model can learn. The model is able to generate sufficiently good results and was clearly learning the task (Fig. 3), however, we also show the unstability in training in Fig. 5.

### A. Exponential Moving Average

The motivation for using EMA lies in the fact that the last iterate in the training process tends to be noisy due to stochastic approximation. By employing EMA, we obtain more stable predictions that are less sensitive to sudden changes, making it better suited for generating images. On the other hand, the non-EMA model gives equal weight to all examples, which provides an accurate estimate of the model's performance on new, unseen data, making it more appropriate for training purposes.

Formally, the EMA of the model weights is updated as follows:

$$\bar{\theta}t = \beta_2 \cdot \bar{\theta}t - 1 + (1 - \beta_2)\theta_t, \tag{13}$$

where $\bar{\theta}_t$ is the EMA of the model weights at time step $t$, $\beta_2$ is the EMA decay factor, and $\theta_t$ represents the real model's weights at time step $t$. The initial EMA model weights are set to $\bar{\theta}_0 = \theta$ at some time step $t \geq 0$, and then updated using the equation above after every optimizer step. Typically, a value of $\beta$ is chosen to be very close to 1 for increased stability.

The choice between EMA and non-EMA depends on the specific task and the desired tradeoff between stability and accuracy. By using EMA during training, as described in the Adam optimizer paper [7], we enhance the robustness of our model, making it more resistant to noise and outliers in the training data, while still maintaining an accurate estimate of the model's performance on new data.

Our experiments demonstrated that using the Exponential Moving Average (EMA) during training led to significant improvements in the quality of generated images. By evaluating each model with random noise input samples for 1000 time steps after each epoch, we observed that the EMA model produced consistent high-quality outputs even at early stages of training. In contrast, the non-EMA model took longer to produce acceptable results and was more prone to generating worse results after certain epochs. The stability of the EMA model was consistently superior, showcasing the benefits of employing EMA during the training process.

### B. Linear and Cosine Noise Scheduling

In this subsection, we explore two noise scheduling approaches: linear and cosine noise scheduling (Fig. 6, Fig. 7), used in the diffusion process. The choice of noise schedule can have a significant impact on the model's ability to learn effectively.

Recall that in the diffusion process, the noise schedule is defined by the sequence of $\beta$ values at each time step. Linear


Fig. 5. The EMA model was able to produce reasonable images much faster than the non-EMA model, which produced mostly unstable results early in the training. While it stabilized more as training went on, it still produced lower-quality results and often completely unstable outputs.

Fig. 6. Top row shows the linear noising schedule, whereas the bottom row is the cosine noising schedule. The cosine schedule makes for a more varied representation of the levels of noise and corruption that the model can learn.

Noise Scheduling is a scheme where the variance of the noise at each step increases linearly. On the other hand, Cosine Noise Scheduling, as introduced in the Improved DDPM paper [11], takes a different approach, with the variance of the noise increasing slowly at the beginning, fastest in the middle, and slowly at the end. This prevents abrupt changes in the noise levels at the beginning and end, resulting in a more favorable noise distribution for the model to learn from.

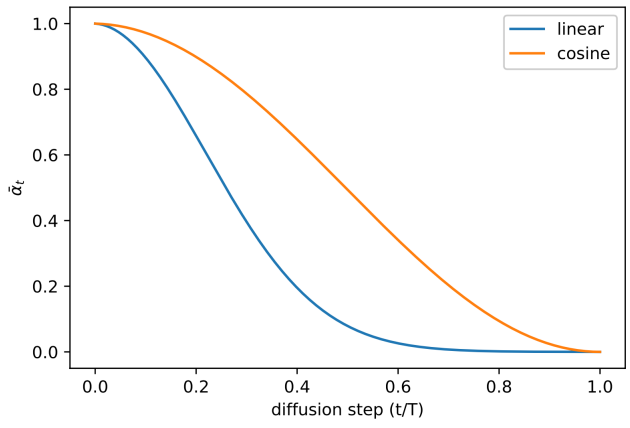


Fig. 7. Comparison of the linear and cosine noising schedules showing that the $alpha_t$ decreases more linearly in the middle. $1 - alpha_t$ represents the variance, which we see having more well-distributed steps for sampling for the cosine schedule. In practice, this does not necessarily mean it is better.

The cosine schedule is constructed using the value of $\alpha$ as follows:

$$\alpha_t = \frac{f(t)}{f(0)}, \quad f(t) = \cos^2\left(\frac{\frac{t}{T} + s}{1+s} \cdot \frac{\pi}{2}\right) \tag{14}$$

To convert this definition to variances $\beta_t$, we use the relation $\beta_t = 1 - \frac{\alpha_t}{\alpha_{t-1}}$. In practice, $\beta_t$ is clipped to be no larger than 0.999 to prevent singularities at the end of the diffusion process near $t = T$. The cosine schedule is designed to have a linear drop-off of $\alpha_t$ in the middle of the process, while changing very little near the extremes of $t = 0$ and $t = T$ to prevent abrupt changes in noise level. The Improved DDPM paper demonstrates that the cosine schedule allows the model to retain information more effectively than the linear schedule from Ho et al. (2020) [5].

In our experiments, we found that on average, the linear noise schedule produced better results. However, given the potential advantages of cosine noise scheduling, we are interested in experimenting with more noise scheduling approaches to further improve the performance of our model. By exploring alternative noise scheduling schemes, we aim to identify the optimal approach that leads to the most effective learning process for our diffusion model.

### C. Conditional Generation from Captions

Conditional generation from captions was a stretch goal of this project; thus, we are currently running and actively experimenting with this step. Though the implementation is complete and training is in progress, we needed some time and work for training to converge and for us to see immediately useful results. In our experiment, we verified our implementation works and the model trains as expected, as we observed loss decreasing.

## VI. Conclusion, Discussion, and Future Work

In our project, we have presented our work on developing latent diffusion models for image generation using an art dataset. Our results demonstrated that our models are capable of learning the task at hand and generating high-quality images, which showcases the potential of latent diffusion models for this domain. The generated images not only exhibit high fidelity but also maintain the artistic essence.

As we look forward to future research directions, we aim to further improve conditional generation on text, allowing users to generate images based on textual descriptions accurately. Additionally, we plan to explore more noise schedules to better understand their impact on model performance and stability. We are also interested in training our models on larger datasets to investigate their ability to provide more coherent and diverse outputs, which is necessary for better conditioned outputs.

It is crucial to acknowledge the potential implications of our work on copyright and creative ownership. This is an academic project and serves as a basis for further exploration and innovation. We encourage ongoing discussions and debates in order to find fair and ethical solutions to the challenges posed by AI-generated art and other forms of media. It

is our hope that this work can ultimately guide people to be more productive and creative, rather than diminish their opportunities for artistic expression. We believe tools like text-conditional image generation, and many downstream tasks that this model implicitly learns and we will later explore, are revolutionary ways to introduce natural language to the digital artistic process.

## REFERENCES


[1] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

[2] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021.

[3] gigant. Wikiart. Hugging Face, https://huggingface.co/datasets/huggan/wikiart.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

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

[6] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[7] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[8] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[9] laion. Laion-art. Hugging Face, https://huggingface.co/datasets/laion/laion-art.

[10] laion. Laion-5b: A new era of open large-scale multi-modal datasets, https://laion.ai/blog/laion-5b/.

[11] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

[12] 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.

[13] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[14] sbert.net. Sentencetransformers documentation, https://www.sbert.net/.

[15] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[16] Hoang Thanh-Tung and Truyen Tran. Catastrophic forgetting and mode collapse in gans. In *2020 international joint conference on neural networks (ijcnn)*, pages 1–10. IEEE, 2020.

[17] 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.


## VII. APPENDIX

### A. Group Member Credits

Most work done as group (on call, pair & group coding). Rami did additional work in VQGAN setup and transfer learning. Mustafa did additional work in UNet training and tuning. Justin did additional work in dataset preparation and conditional generation implementation.