IMAGE GENERATION USING Denoising Diffusion Probability Model With learning rate warm-up and classifier-free guidance

Anonymous author

Abstract

This paper proposes using a Denoising Diffusion Probability Model (DDPM) with learning rate warm-up and classifier-free guidance for image generation on CIFAR-10 and STL-10. The results demonstrate realistic and high-quality output, although hardware limitations made producing high-resolution images on the STL-10 dataset challenging. And the Future work could involve implementing the improved Denoising Diffusion Implicit Model (DDIM) to reduce the processing time and investing in a more advanced and stable hardware environment.

1 Methodology

The project employs the Denoising Diffusion Probabilistic Model (DDPM), a parameterized Markov chain trained by variational inference to generate samples that match the data after a limited time. This chain's transitions are trained to reverse a diffusion process that adds noise to the data in the opposite direction of sampling until the signal is destroyed [2]. The fundamental components of DDPM are a Gaussian diffusion process, a Gaussian diffusion sampler, and a U-Net, and the process has been shown in Figure 1 [2]. The DDPM generates images using a forward diffusion process, which employs a sequence of Gaussian noise levels added to the original image over a discrete sequence of time steps until the noise level is fully saturated, resulting in a final distorted image [2]. The DDPM then applies a learned reverse denoising diffusion process, represented by p_θ, to the distorted images using the Gaussian diffusion sampler to produce denoised images. The denoised images are fed into a U-Net, a neural network architecture that models the probability distribution of the images and generates new image samples [2].

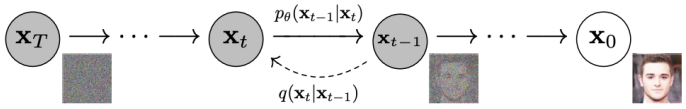


Figure 1: DDPM process [2]

1.1 Gaussian diffusion process

The objective function L_t in Equation 1 is used to optimize the neural network $\varepsilon_\theta(x_t, t)$, which serves as the diffusion sampler by taking the current image x_t and the current time step t as inputs and produces a sample of Gaussian noise that is added to the image at that time step [2]. The sampler is optimized using the Adam optimizer with mean square error (MSE) as the loss function between the true and predicted Gaussian noise. In the objective function, x_0 denotes the initial image, and ε represents the pure noise at time step t. The term $\sqrt{\bar{\alpha}_t}x_0$ corresponds to the contribution of the original image, while $\sqrt{1-\bar{\alpha}_t}\varepsilon$ corresponds to the contribution of the noise. $\bar{\alpha}_t$ is a scalar value that determines the strength of the diffusion process at time step t, with $\bar{\alpha_0} = 0$ and $\bar{\alpha_T} = 1$, where T is the total number

of diffusion steps. Overall, this diffusion process allows the model to gradually add noise to the original image over time, with the noise level increasing as $\bar{\alpha}_t$ approaches 1.

$$L_t = \|\varepsilon - \varepsilon_\theta(x_t, t)\|^2 = \left\|\varepsilon - \varepsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon, t)\right\|^2 \tag{1}$$

1.2 Sampling process

During the sampling process in DDPM, the goal is to reverse the diffusion process used to create the noisy image [2]. To do this, we sample from the conditional probability density function $p_\theta(x_{t-1}|x_t)$, which represents the probability density of x_{t-1} given x_t [2]. This conditional density is a Gaussian distribution with mean $\mu_\theta(x_t, t)$ and standard deviation $\epsilon_\theta(x_t, t)$, both of which are functions of the neural network parameters θ [2]. To sample $x_{t-1} \sim p_\theta(x_{t-1}|x_t)$, we use a reverse function that takes the noisy image x_t and produces the actual image x_{t-1} [2]. The reverse function is given by $\frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right)$, as shown in Equation 3 in the [2]. This function can be simplified to $\bar{\mu}_t = \mu_\theta(x_t, t)$, which allows us to train the mean function approximator μ_θ to predict $\bar{\mu}_t$ directly, rather than computing the reverse function explicitly [2].

$$p_\theta(x_{t-1}|x_t) = N(x_{t-1}; \mu_\theta(x_t, t), \epsilon_\theta(x_t, t)) \tag{2}$$

$$\mu_\theta(x_t, t) = \tilde{\mu_t}\left(x_t, \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(x_t))\right) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right) \tag{3}$$

$$L_{t-1} = \mathbb{E}_{x_0,\epsilon}\left[\frac{\beta_t^2}{2\sigma_t^2(1-\bar{\alpha}_t)}\left\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\right\|^2\right] \tag{4}$$

1.3 U-Net

The U-Net architecture is used in DDPM to model the probability distribution of images and generate new image samples [2]. The U-Net was introduced by [5], and the DDPM implementation includes several modifications to this base architecture. One essential modification is using position embeddings, which incorporate information about each element's position or time step in a sequence or image [4]. Position embeddings encode the spatial position of each pixel in an image, improving the model's ability to generate realistic images [4]. Another modification is using ResNet blocks and an attention module, which allows the model to learn complex representations of the input data and preserve essential features [4]. These blocks take input data, perform operations, and then add the modified version to the original input [4]. This approach enhances the input representation and facilitates the generation of high-quality images [4].

$$PE_{t,i} = sin(\frac{t}{10000^{(\frac{i}{d-1})}}) \tag{5}$$

$$lr = base\_lr \cdot \left((multiplier - 1) \cdot \frac{epoch}{warm\_epoch} + 1\right) \tag{6}$$

1.4 Learning rate warming up & classifier-free guidance

The learning rate warm-up method, introduced by [1], is utilized during training to facilitate faster network convergence and enhance the overall training performance. Instead of a fixed learning rate, a small value is employed initially and gradually increased to enable the optimizer to navigate the optimization landscape more effectively and locate a starting point for the learning process [1]. This approach allows for flexibility in selecting an appropriate learning rate and improves the module's performance [1]. The learning rate is defined by Equation 6, where the multiplier is the factor to multiply the base learning rate after the

Figure 2: non-cherry-picked samples from CIFAR-10

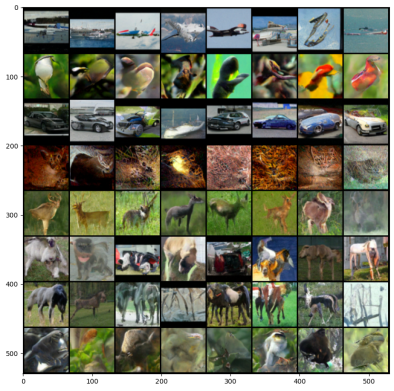


Figure 3: non-cherry-picked samples from STL-10 resized to 64x64

Figure 4: Linearly interpolated samples from CIFAR-10

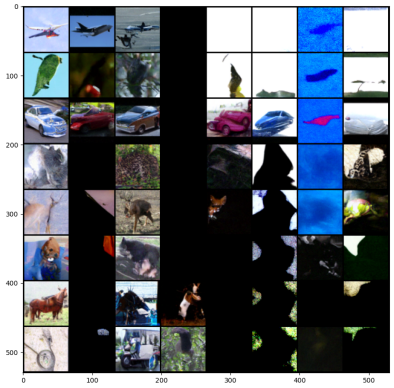


Figure 5: Linearly interpolated samples from STL-10 resized to 64x64

warm-up period [1]. The project also involved classifier-free guidance techniques, which were introduced by [3], with the aim of classifier-free guidance to complicate the training pipeline by training an unconditional denoising diffusion model $p_\theta(z)$ to remove noise [3].

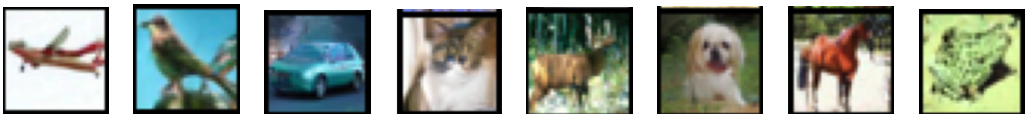
2 Results

The results are pretty positive when considering the performance of CIFAR-10 images and STL-10, as demonstrated in Figure 2 and 3, respectively. Most images are easy to identify, even with a similar colour to the background. The interpolations between the sample pairs depicted in Figures 4 and 5 lack visually convincing and seamless transitions. CIFAR-10 has better performance than STL-10 in general.

Figure 6 and 7 show some cherry-picked samples that show the best outputs the model has generated from STL-10 and CIFAR-10.

Airplane Bird Car Cat Deer Dog Horse Monkey

Figure 6: cherry-picked examples from STL-10 resized to 64x64



Airplane Bird Car Cat Deer Dog Horse Frog

Figure 7: cherry-picked examples from CIFAR-10

3 Limitations

Both the CIFAR-10 and STL-10 results look realistic and are easy to identify; however, some of the images in STL-10, such as the cat, are more difficult to recognize and did not demonstrate the same level of quality as those in CIFAR-10. Also the linear interpolated samples are not perform as expected. Some of the limitations were due to hardware constraints and NCC crashes, which made training a time-consuming and challenging process; In the future, it would be worthwhile to implement the improved version of the diffusion model known as the Denoising Diffusion Implicit Model (DDIM), which has been shown to generate higher-quality samples using fewer steps, and thus could improve the efficiency of the process [6]. Furthermore, using a more advanced and stable hardware environment to run the training process could significantly improve the performance of the model.

Bonuses

This submission has a total bonus of +4 marks (a bonus), as it is trained on STL-10 resized to 64x64 pixels.

References

[1] Priya Goyal et al. "Accurate, large minibatch sgd: Training imagenet in 1 hour". In: *arXiv preprint arXiv:1706.02677* (2017).

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

[3] Jonathan Ho and Tim Salimans. "Classifier-free diffusion guidance". In: *arXiv preprint arXiv:2207.12598* (2022).

[4] Nielsr Rogge and Kashif Rasul. *The Annotated Diffusion Model*. June 2022. URL: `https://huggingface.co/blog/annotated-diffusion`.

[5] 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*. Springer. 2015, pp. 234–241.

[6] Jiaming Song, Chenlin Meng, and Stefano Ermon. "Denoising diffusion implicit models". In: *arXiv preprint arXiv:2010.02502* (2020).