# A Conditional Generative Image Model

He Nan Li
lihenan@stanford.edu

Lucas Lu
jiayulu@stanford.edu

## Abstract

*Conditional diffusion models (CDM) have transformed the fields of image and video synthesis, but their complex mechanisms and high computational requirements often make them less accessible. Our project aims to demystify CDMs, offering a clear and concise explanation with intuitive visualizations. We will explore the core architecture of U-Net, breaking down its components, and elucidating the noise estimation process. Furthermore, we will demonstrate the practical implementation of CDMs by training a PyTorch model on the MNIST and CIFAR-10 [3] datasets.*

*Our approach aims to bridge the gap between theoretical understanding and practical application, promoting a deeper comprehension of CDMs and encouraging their wider use in research and development.*

## 1. Introduction

This study proposes the creation of a diffusion model that combines Denoising Diffusion Probabilistic Models (DDPMs) with conditional image generation capabilities. The model will be built using PyTorch and trained on the MNIST and CIFAR-10 datasets. Once trained, the model will allow users to generate new, class-conditional images from pure noise by selecting a desired class from the dataset.

The main objective of this research is to explore the practical trade-offs involved in developing generative models, with a specific focus on:

- The effectiveness of the simpler loss functions proposed by DDPM.
- The impact of architectural variations within the U-Net framework on performance.
- The effect of different noise generation schedules on the forward and reverse diffusion processes.
- The effect of conditioning mechanisms in controlling the semantic content of generated images.
- The influence of dataset complexity on model performance, comparing results between the relatively simple MNIST dataset and the more complex CIFAR-10 dataset.

With unconditional generation, there is no input to the algorithm; the model generates a novel image that shares characteristics with the training data. In contrast, with conditional generation, the input to the algorithm is a valid class choice. For example, in the MNIST dataset, we can instruct the model to generate an image of a digit between 0 and 9, resulting in a novel image from the specified class.

Within the DDPM framework, the U-Net serves as the neural network to predict the noise at each timestep. The input to the U-Net is an image at time $t$, time embedding and contextual embedding. The U-Net outputs $\epsilon$ with the same input image feature dimension as the input image. This $\epsilon$ represents the estimated noise to be subtracted from the current image at $t$ to produce an image at $t-1$, bringing it one step closer to a novel image.

## 2. Related Work

Our project builds on the foundational work of several seminal contributions in the field of generative modeling.

Firstly, the concept of diffusion models was introduced in "Deep Unsupervised Learning using Nonequilibrium Thermodynamics" [8], which utilized iterative forward and reverse processes to transform images into pure noise and then gradually denoise them back into new samples. This transformation is modeled as a Markov chain, where each step depends solely on the previous one.

Secondly, "Denoising Diffusion Probabilistic Models" (DDPMs) [1] on this concept by introducing a more efficient training procedure and demonstrating its effectiveness in generating high-quality images. The authors also proposed a simplified loss function based on mean squared error (MSE) to measure the discrepancy between the true noise added during diffusion and the noise predicted by the neural network.

Building on these advancements, "Improved Denoising Diffusion Probabilistic Models"[5] further refined the DDPM framework by incorporating learnable covariance

and a more flexible noise scheduling mechanism. These modifications resulted in improved sample quality and faster convergence during training.

The neural network architecture used in DDPMs to predict the noise to be removed at each step is typically based on the U-Net architecture, as proposed in "U-Net: Convolutional Networks for Biomedical Image Segmentation" [6]. This architecture features a "U" shape, with down-sampling and up-sampling layers connected by residual connections, allowing it to capture both local and global image features while maintaining the output feature dimensions. The U-Net's ability to preserve spatial information makes it particularly well-suited for image generation tasks in diffusion models.

## 3. Method

Our initial inspiration and starter code are based on [10]. From this source, we incorporated a new noise schedule and a contextual embedding as classifier-free guidance.

### 3.1. Denoising Diffusion Probabilistic Models (DDPM)

DDPM has a forward diffusion process:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I})$$

where, at time $t$, it generates a random noise (unit Gaussian distribution) from the beta schedule and directly adds it to the image at time $t$ to produce the image at $t+1$. This generated random noise is then used as the ground truth value for supervised training with the U-Net model described below

It also has a reverse diffusion process:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

where, at time $t$, it uses the U-Net with learned parameters to estimate a noise (calculated from $\beta$) and subtract it from the image at $t$ to obtain hopefully a less noisy image at time $t-1$. To sample a generated image from pure noise, we need to recursively apply this reverse diffusion process until reaching $t = 0$.

The DDPM component is strictly not a neural network; rather, it includes a neural network, specifically the U-Net.

### 3.2. U-Net

The U-Net (figure 1), named for its characteristic "U" shape in model diagrams, operates by progressively down sampling the feature map resolution of an input image, followed by up sampling to restore the original resolution.

The residual blocks contain convolutional layers that extract image features while maintaining the feature dimensions. The self-attention blocks use the same structure from the Transformer paper [12] to understand spatial context within the feature map. The yellow convolutional layers downsample the feature map to produce a lower resolution, and transposed convolutional layers are used to upsample to a higher resolution. Crucially, skip connections are employed at each resolution level to act as residual connections.

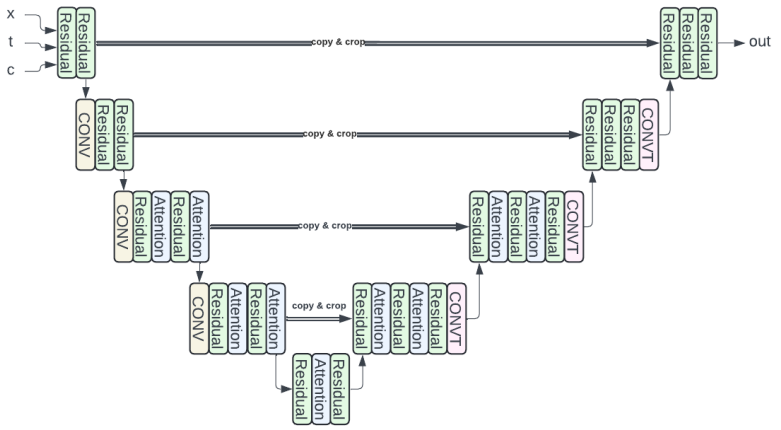


Figure 1. U-Net Architecture

Three inputs: the image ($x$), time schedule embedding ($t$), and contextual embedding ($c$), are fed into the down sampling pathway. However, not all inputs are necessarily required in every block. For instance, the self-attention mechanisms do not utilize the time schedule or the contextual embedding.

We use the derived loss function from the original DDPM paper, which is simplified to the following:

$$L(\theta) = \mathbb{E}\left[\left\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\right\|^2\right]$$

in which $\alpha$ is derived from $\beta$, and $\epsilon$ is the parameter to the unit Gaussian distribution. Essentially we are building a neural network to learn the best $\epsilon$ to denoise at each time step $t$.

#### 3.2.1 Residual Block

In the residual block (figure 2), the feature map, contextual embedding, and time embedding are integrated. To achieve this, the contextual and time embeddings are each processed through an activation layer followed by a linear layer, ensuring their output dimensions align with that of the feature map after a single convolution layer.

Subsequently, the combined output of the three components undergoes another convolution layer incorporating dropout, with the residual connections added on top. The inclusion of dropout mitigates the risk of over fitting on a

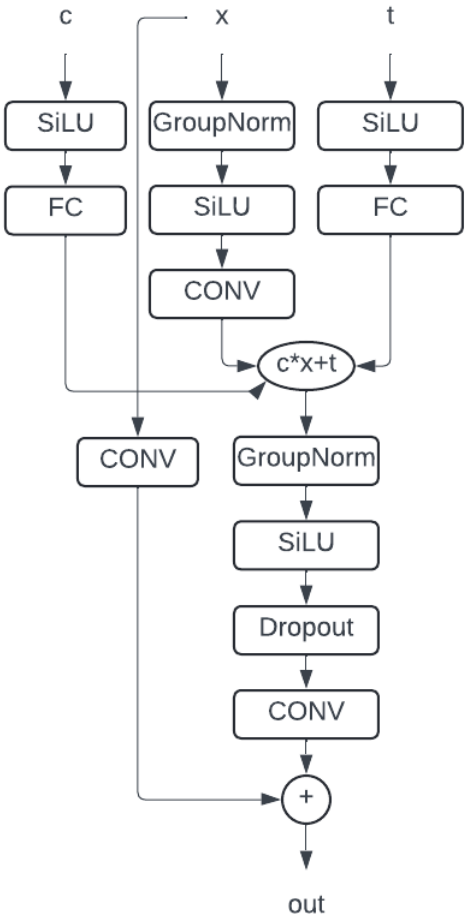


Figure 2. Residual Block Architecture

limited set of neurons or features, while the residual connections enhances flexibility and facilitates deeper neural networks.

It is worth noting that the contextual embedding and feature map are multiplied rather than added, resulting in different outcomes. In the experiments, this multiplication leads to better-generated images.

#### 3.2.2 Attention Block

The attention blocks (figure 3) uses the same structure as described in the Transformer paper [11]. They are applied at lower resolution layers. As discussed in the lecture, the self-attention layer within these blocks is a crucial factor in enhancing image generation quality. The attention blocks allow for communication between distant regions of the image feature map, facilitating the construction of a global context. This integration of global information significantly improves the quality of generated images compared to our project milestone.

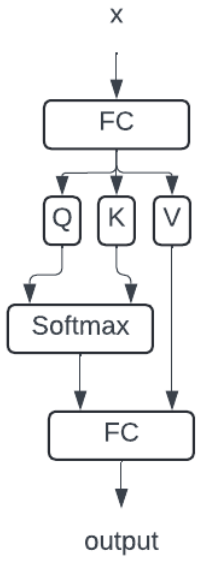


Figure 3. Attention Block Architecture

## 4. Dataset and Features

The MNIST database of handwritten digits (0 to 9) comprises a training set of 60,000 28x28 examples and a test set of 10,000 examples. We primarily used MNIST to facilitate quick iterations development.

The CIFAR-10 dataset consists of 60,000 32x32 color images divided into 10 classes, with 6,000 images per class. It includes 50,000 training images and 10,000 test images.

Since these labeled images are widely available in the public domain, we did not collect our own image set. However, the ”style” of a generated image is highly dependent on the training dataset.

In the U-Net, we followed the original paper and selected 64 as the feature dimension for the first convolutional layer.

## 5. Experiments

### 5.1. Hyperparameters

Table 1 describes the hyper parameters used throughout the experiments. ”Total Timesteps” refers to the terminology used in the original DDPM paper. In this context, a real image from the dataset is considered to be at time 0 and pure noise is considered to be at time 1000. The other parameters are fairly standard starter values. We experimented with AdamW [4] (which includes decay) to extend training duration while still observing marginal improvements.

### 5.2. Unconditional Generation

As a baseline, we tested the unconditional generation of the images, where newly generated images can belong to

| Hyperparameter | Value |
|---|---|
| Batch Size | 64 |
| Total Timesteps | 1000 |
| Learning Rate | $5 \cdot 10^{-5}$ |
| First CONV Output Channels | 64 |
| Optimizer | AdamW |

Table 1. Hyperparameters used throughout the experiments

any class. Figure 4 shows generated images from training on MNIST dataset for a few epoch. Although these images are not recognizable as handwritten digits, some curves are discernible, making them significantly better than pure noise.

Figure 5 shows unconditionally generated images from training on CIFAR10 dataset for 10 epoch. Similarly, while the images are not yet clearly recognizable, they are much better than pure noise.

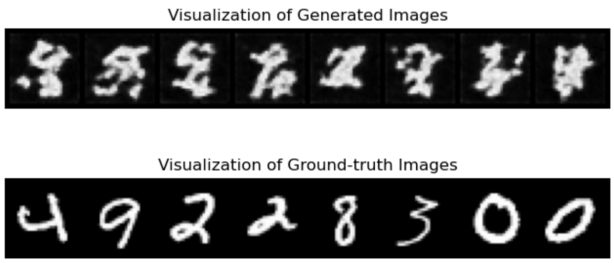


Figure 4. The top row sampled 8 generated images from the model after training for 4 epoch. The bottom row are sampled from the MNIST dataset representing the ground truth.

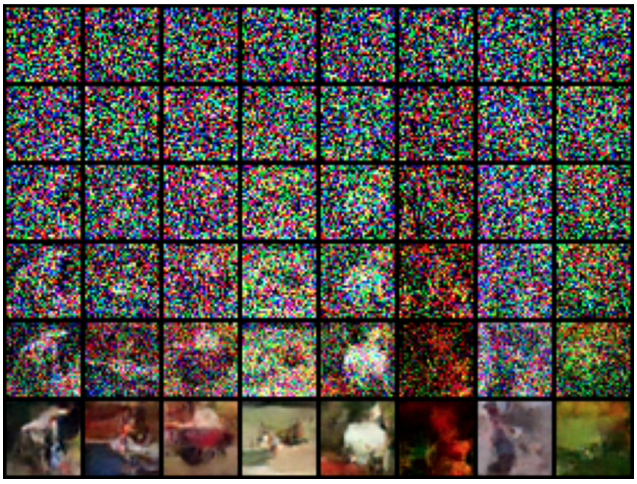

Figure 5. Unconditional generated images from training on CIFAR10 for 10 epoch.

### 5.3. Noise Schedule

We improved upon the original DDPM implementation by using a better noise ($\beta$) schedule. In the original paper, $\beta$ is incremented linearly. As suggested from [5], we can use a cosine schedule to scale up beta which is much slower than linear schedule. For the total time step $T$, at some time step $t$, we generate the next $\beta$ is generated as follows

$$\beta_{t+1} = 1 - \frac{\cos\left(\frac{\pi}{2}\frac{(t+1)/T+0.008}{1.008}\right)^2}{\cos\left(\frac{\pi}{2}\frac{t/T+0.008}{1.008}\right)^2}$$

As a result, shown in figure 6, the cosine noise schedule adds noise much slower than the linear noise schedule shown in figure 7. In the linear schedule, for more than half of the timesteps, the images are close to pure noise and have lost most of their features, making the U-Net learn a lot of "useless" information.

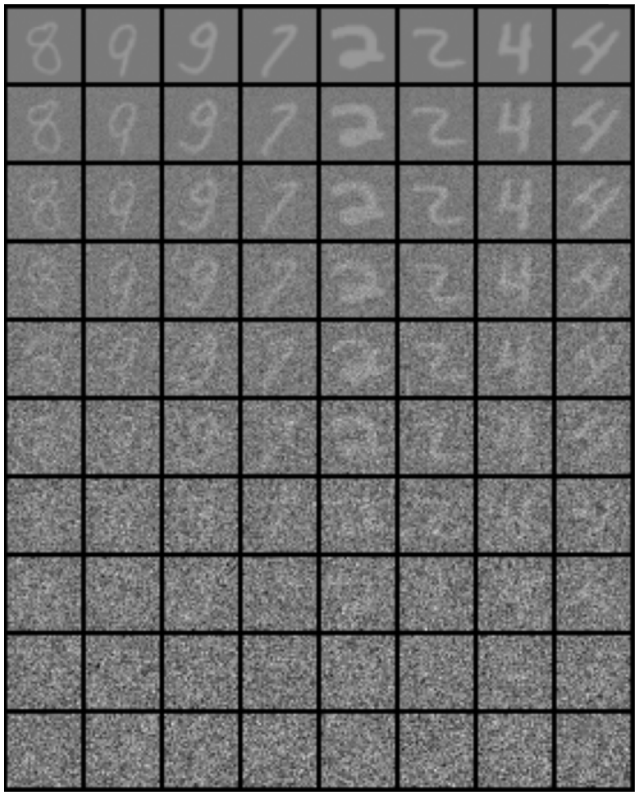

Figure 6. Cosine noise schedule.

The cosine schedule enables the U-Net to learn more meaningful data. As expected, when generating new images using the cosine noise schedule, the generation occurs more gradually (figure 8), whereas with linear noise schedule, most of the generation happens at the last steps (figure 9).

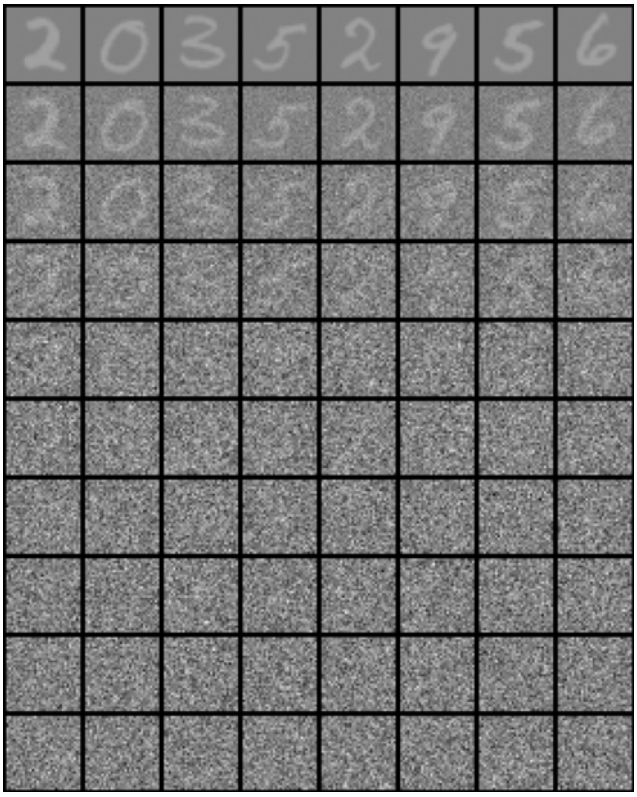

Figure 7. Linear noise schedule.

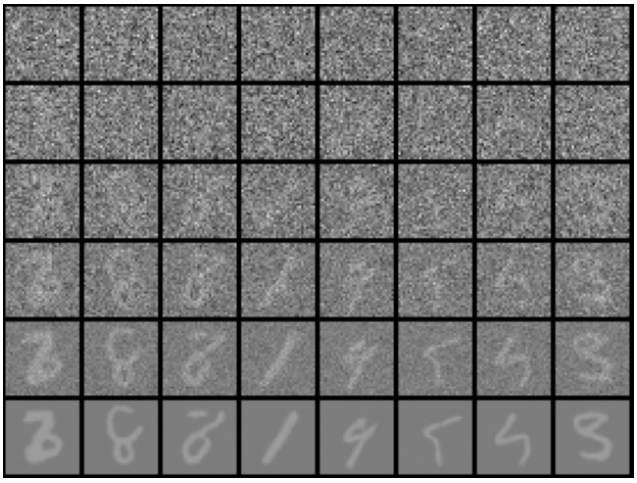

Figure 8. Unconditional generated images based on cosine beta schedule. We can recognize the gradual generation of the image features.

### 5.4. Conditional Generation

To facilitate conditional generation, class information needs to be transformed into contextual embedding and applied to various layers of the neural network. The classes (usually in integers) are first converted into one-hot encoding, then processed through two fully connected linear layers with activation to generate the contextual embedding ($c$). This embedding is then added to each residual block in the U-Net, similar to the time embedding.

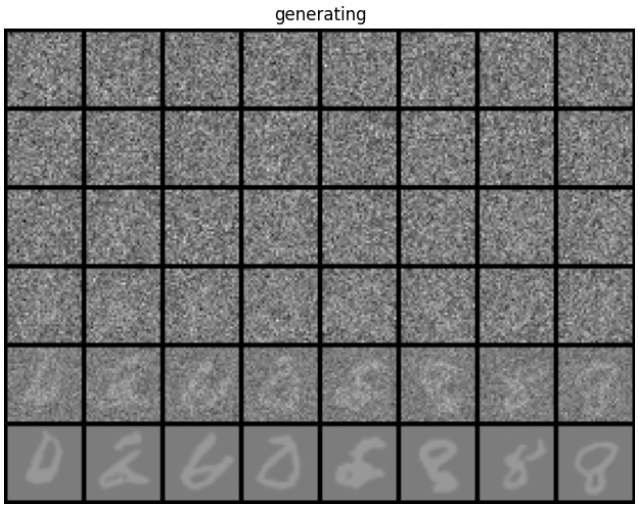


Figure 9. Unconditional generated images based on linear beta schedule. For most of the time steps during the generation, it's mostly noise.

As suggested by [5], for a certain activation layer ($a_L$) in the U-Net, in addition to the time embedding $t$, the contextual and time embeddings are combined as $a_{L+1} = c \cdot a_L + t$ to produce the next layer.

Figure 10 showed examples of conditional generation of digit 0 through 7 after training on MNIST dataset for 10 epoch.

Figure 11 showed examples of conditional generation of first 8 classes (airplane, automobile, bird, cat, deer, dog, frog, horse) after training on CIFAR10 dataset for 60 epochs. Arguably, features resembling the class are recognizable in some examples.

### 5.5. Result

Figure 12 shows the loss curve on the U-Net while training on CIFAR-10 dataset for 20 epoch. During training, the model is not overfitted suggesting the model is not complex enough, or the training data set is limiting.

The Inception Score [7] can be used to assess the quality of generated images. As a baseline, we calculated the Inception Score for both pure noise and the CIFAR-10 dataset. Pure noise represents an untrained model, and its Inception Score serves as the lower bound for performance. We expect any trained model to outperform pure noise. On the other hand, the Inception Score of CIFAR-10 serves as an upper bound for a generative model's performance.

We found the Inception Score on pure noise to be close to 1 and the Inception Score on CIFAR-10 is close to 10

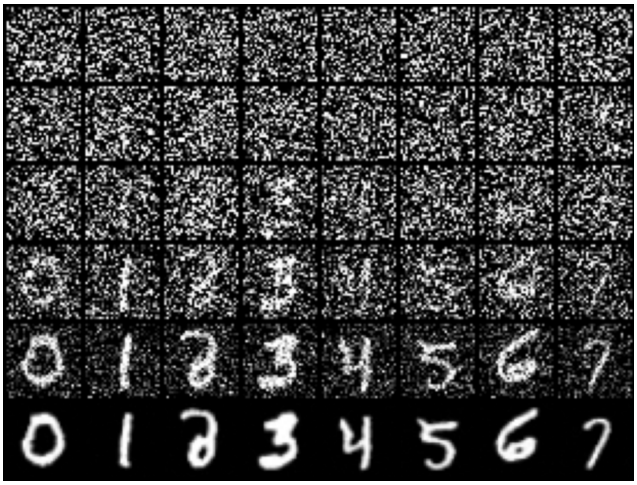

Figure 10. Conditional generated images after training on MNIST dataset.

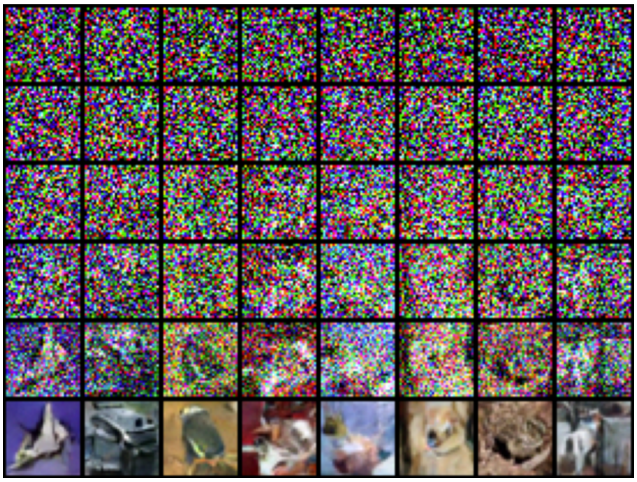

Figure 11. Conditional generated images after training on CIFAR10 dataset.

(See table 2). Our model scored 2.6 on the inception score, which is slightly better than pure noise but not high enough to convincingly resemble new images.

Similarly, as a reference we computed the Fréchet inception distance (FID) by comparing our generated images and real images. The result yield 2.6 FID score for our model on CIFAR-10, which isn't very high compared to the original DDPM paper. One limiting factor may be that we didn't generate enough novel images as they are slow to generate.

# 6. Conclusion and Future Work

## 6.1. Conclusion

The U-Net structure performs well by down sampling and up sampling while maintaining output dimension. The residual connections allowed us to construct deep neural networks without worrying about diminishing gradient problem.

The cosine noise schedule performed better than the linear noise schedule, as the cosine schedule removes noise more gradually while retaining more features.

With contextual embedding, we were able to change the neural network with fairly minimal effort to enable conditional generation.

The overall performance is not on par compared to the original papers with the following possible reasons: insufficient training duration, insufficient data, and the model being too large and complex.


Figure 12. Loss curve training on CIFAR10 dataset for 20 epoch.

| Data Set | IS |
|---|---|
| Pure Noise | $1.2150 \pm 0.0113$ |
| Our Model | $2.6435 \pm 0.1593$ |
| CIFAR-10 | $9.9603 \pm 0.7447$ |

Table 2. Inception score on pure noise, CIFAR-10 data set and our model.

## 6.2. Future Work

To further improve, we may need more data to overfit the neural network, given its many residual and self-attention blocks. The image set we are training on has only 60,000 examples. We can preprocess the dataset to generate more data by cropping, applying jitter, or blocking out some areas.

To sample a new image, we need to iterate 1000 times through the neural network, which is very time consuming. We can improve it by applying Denoising Diffusion Implicit Models [9].

We can experiment with learnable covariance as opposed to fixed covariance schedule. Also, as [2] suggested, a score

based diffusion with varying weight conditional guidance could improve the generation.

## 7. Contributions and Acknowledgements

He Nan spearheaded the development of the U-Net, Attention Block, and Residual Block components. Lucas contributed by implementing classifier-free guidance on top of the existing model and setting up the Google Cloud Platform (GCP) environment with Docker and Jupyter server. Collaboratively, they refined the model, progressing from basic convolution layers to a sophisticated diffusion model.

Our starter code is inspired from [10], from which we implemented a full U-Net, changed noise scheduling, and added a new contextual embedding.

## References

[1] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[2] K. Kreis, R. Gao, and A. Vahdat. Denoising diffusion-based generative modeling: Foundations and applications. *CVPR*, 2022.

[3] A. Krizhevsky, V. Nair, and G. Hinton. CIFAR-10 (Canadian Institute for Advanced Research), 2009.

[4] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv:1711.05101*, 2017.

[5] A. Q. Nichol and P. Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021.

[6] O. Ronneberger, P. Fischer, and T. 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.

[7] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *arXiv:1606.03498*, 2016.

[8] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[9] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models, 2022.

[10] N. W. Varuna Jayasiri. labml.ai annotated paper implementations, 2020.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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