Denoising Diffusion Model

Luu Hai Tung[1], Yan Meng[2], and Mengkedalai[3]

[1]haitungluu@edu.bme.hu
[2]yan.meng@edu.bme.hu
[3]kedalai.meng@edu.bme.hu

Abstract This study focuses on fine-tuning existing diffusion model research using the CIFAR-10 dataset. Our objective was to understand the operational dynamics of diffusion models and assess their performance against a standard dataset. Through these experiments, we performed corresponding visual studies on the dataset, configured the diffusion model, fine-tuned and optimized the model results. In order to explore the feasibility of model application, we created a visual application interface for the model. Our research results give us a clearer understanding of the functions and limitations of diffusion models, and help us further research on deep learning models.

1 INTRODUCTION

In recent years, the field of artificial intelligence has witnessed remarkable advancements, particularly in generative modelling, a domain focusing on the creation of new data instances that resemble a given data set[1]. Among the various approaches in this area, diffusion models have emerged as a groundbreaking technique, distinguishing themselves through their ability to generate high-quality, diverse samples across various domains like images, audio, and text.

Diffusion models[2] are inspired by the physical process of diffusion, which describes how particles move from areas of higher concentration to areas of lower concentration. In the context of machine learning, these models gradually transform data from a complex distribution (such as a set of natural images) into a simpler one (usually Gaussian noise) and then learn to reverse this process. This reversible transformation is key to their success, enabling the generation of new data instances by carefully reversing the diffusion steps.

The appeal of diffusion models lies in their flexibility and robustness. Unlike traditional generative models like Generative Adversarial Networks (GANs)[3] or Variational Autoencoders (VAEs)[4], diffusion models do not rely on adversarial training or explicit likelihood maximization. Instead, they operate by iteratively denoising data, can produce samples of remarkable clarity and variety.

Furthermore, diffusion models have demonstrated impressive performance in a range of applications. In image generation, they have been shown to produce images of comparable or even superior quality to state-of-the-art GANs. In natural language processing, they offer new ways to tackle text generation challenges. In audio synthesis, their ability to generate coherent and high-fidelity sounds opens up new avenues for realistic voice and music generation.

In this work, we undertake a detailed exploration of diffusion models by closely replicating and fine-tuning existing research, specifically leveraging the CIFAR-10 dataset[5]. Our objective is to unravel the intricate workings of these models, delving into their theoretical foundations and the practical aspects of their implementation. This replication serves as a foundation for a deeper understanding of how diffusion models are trained, the unique challenges encountered during their optimization, and their comparison with other generative models in the field.

Through this process, we gain valuable insights into the practical application of diffusion models, particularly how they perform with well-established datasets like CIFAR-10. By the conclusion of this article, readers will have gained a comprehensive perspective on diffusion models, informed by our empirical experience of replicating and fine-tuning an existing study. This exploration aims to provide a clear understanding of the technical intricacies and the practical challenges of these models, as well as their transformative potential within the realm of generative AI, especially when applied to familiar datasets like CIFAR-10."

2 EXPERIMENT

Our study aimed to replicate and understand the diffusion model, a sophisticated generative model, using the CIFAR-10 dataset. The implementation was carried out in Python, utilizing the PyTorch library for deep learning functionalities. The core components of our implementation included the model architecture, data

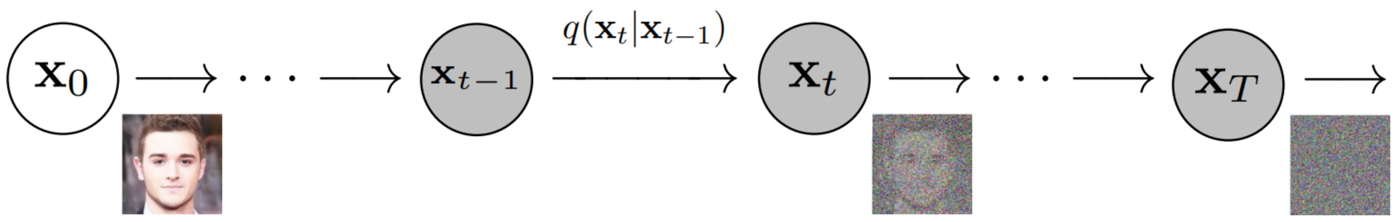


Figure 1: Diffusion model workflow

processing, training procedure, and evaluation metrics. Below we detail each aspect of our methodology.

2.1 Model selection & Architecture

In the diffusion model, a network architecture that can efficiently process and reconstruct complex images is needed.The U-Net architecture is uniquely suited in this regard because it can better handle both local and global information of an image, which is critical for generating high-quality images. In contrast, while conventional CNNs are very powerful in many image-related tasks, they may not have the same ability to retain detail and contextual information as U-Net, which is especially important in the diffusion model. Therefore, U-Net architectures are often preferred in diffusion models.
We utilized a U-Net-based architecture (`Unet` class) for the diffusion model[6]. The U-Net was configured with a base dimension of 64 and dimensional multipliers at subsequent levels set to (1, 2, 4, 8).

2.2 Data Processing

The CIFAR-10 dataset[5] was processed and loaded using our custom `get_data` function, which handled data retrieval and preprocessing, conforming to the specified batch size and image size (32x32 pixels).

2.3 Training Setup

Because the model takes a long time to train, we did not train it directly on Google Colab but locally. Here is the detailed training setup.

- Our training process was built around the `GaussianDiffusion` and `Trainer` classes.
- The diffusion model was configured to operate over 1000 timesteps.
- The `Trainer` class managed the training loop, with key parameters including a learning rate of 8e-5, a total of 20,000 training steps, and gradient accumulation every 2 steps.
- We employed an Exponential Moving Average (EMA)[7] with a decay of 0.995 and activated mixed precision training (`amp=True`).

2.4 Monitoring training with Wandb

For experiment tracking and logging, we integrated Weight & Biases (WandB) into our workflow. This allowed for real-time monitoring of the training process, including loss metrics and image samples.
The WandB project was set up under the name "denoise-diffusion-model" with specific experiment naming based on user input.

2.5 Model Saving and Loading

The trained model's state was periodically saved, with the path specified by the user. This feature facilitated the preservation of model checkpoints for further analysis or continued training.
The above methodology provided a structured approach to replicating the diffusion model. By adhering to these implementation details, we ensured a thorough and accurate replication of the diffusion model's training and evaluation process, as well as a comprehensive understanding of its operational dynamics.

3 EVALUATION

3.1 Evaluation Metrics

In assessing the performance of denoising diffusion models, one of the key evaluation metrics commonly employed is the Fréchet Inception Distance (FID) score[8]. The FID score quantitatively measures the similarity between the generated samples and a reference dataset in the feature space of a pre-trained neural network, typically InceptionV3 or a similar model. This metric has gained popularity in the field of generative modelling due to its ability to capture both the quality and diversity of generated samples.
The FID score is calculated using the following steps:

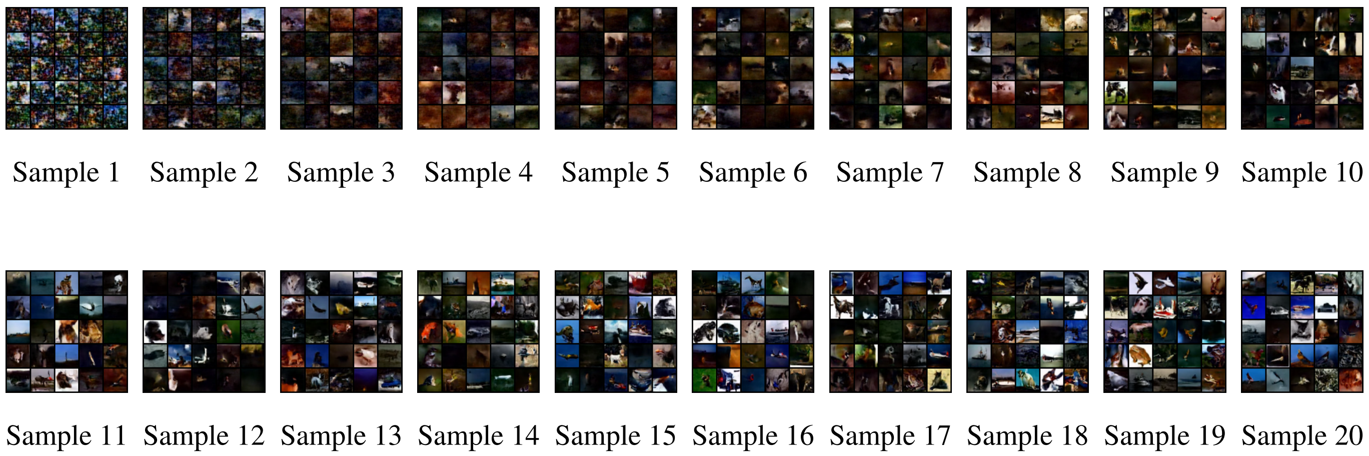


Figure 2: Denoising Processing
The image above illustrates the progressive enhancement of our model's denoising effect, showing the transformation from a noisy image (Sample 1) to a clear image (Sample 20). These series of images represent the denoising effect after the first training epoch, continuing up to the 20th epoch. Through these sequential samples, we can clearly observe how each training epoch progressively improves the image quality, reduces noise, and enhances the clarity of details.

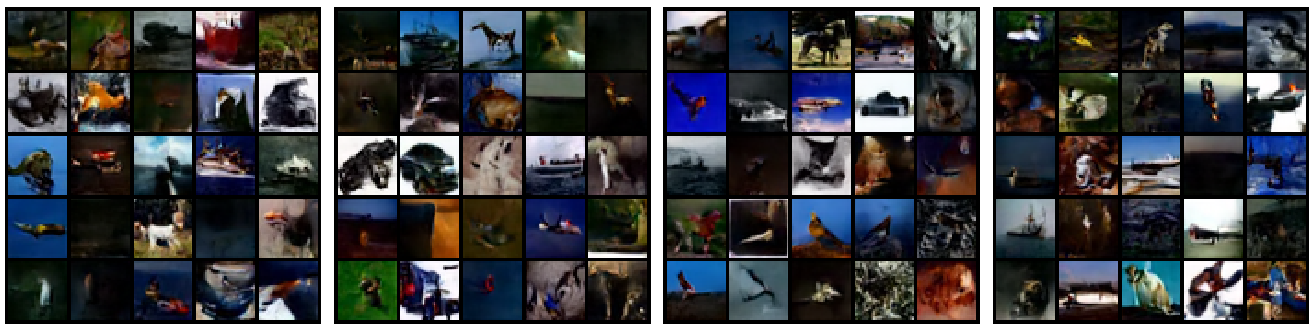


Figure 3: Evaluation overview

- Generation of Samples: Generate a set of synthetic samples using the denoising diffusion model.

- Feature Extraction: For both the generated samples and the reference dataset, extract high-level features using a pre-trained neural network, usually InceptionV3. These features capture important characteristics of the images.

- Statistics Computation: Compute the mean and covariance of the feature vectors for both the generated samples and the reference dataset. These statistics represent the distribution of features in both datasets.

- Distance Calculation: Calculate the Fréchet distance between the two multivariate Gaussian distributions defined by the mean and covariance of the feature vectors. The Fréchet distance is a measure of how similar the distributions are and is used as the FID score.

3.2 Baseline Comparisons

Our results were benchmarked against the original model's performance as reported in the seminal paper. Additionally, we compared our model's performance with other generative models like GANs[3] and diffusion model with pertained weights on the CIFAR-10 dataset[5].

Model	FID Score
BigGAN [9]	14.73
DF Model with pretrained weights[10]	14.30
OURS	77.1

Table 1: FID Score Comparison on CIFAR-10

The training loss and fid score are presented below. The following results give us a potential method to continue working on this model. The FID score starts at a high value, indicating the initial image quality was relatively poor compared to the target distribution, and there was a sharp decrease early in the training, showing rapid improvement in image quality. As training progresses, the FID score decreases but at a slower rate, suggesting diminishing returns on image quality improvement per training step. Towards the later steps, the FID score appears to plateau, indicating that the model may be approaching its performance limit given the current architecture and training data.

In addition, the total loss shows fluctuations but a general downward trend, which is expected as the model learns to minimize its error over time. The fluctuations in total loss may suggest that the model is still learning from the data and there's room for optimization. It could also reflect the stochastic nature of the training process, especially if the batch size is not very large. The overall decline in loss is a good sign, although it hasn't stabilized, which typically indicates that training could still be beneficial.

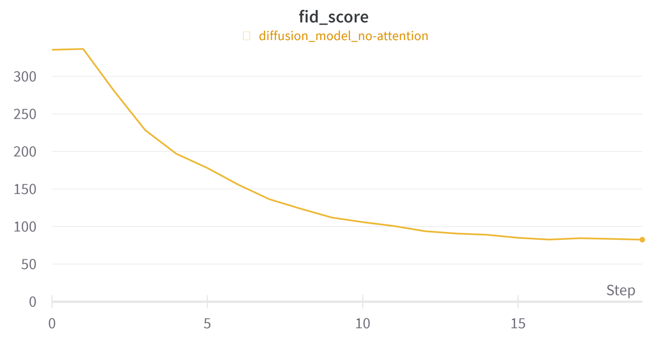


Figure 4: FID score

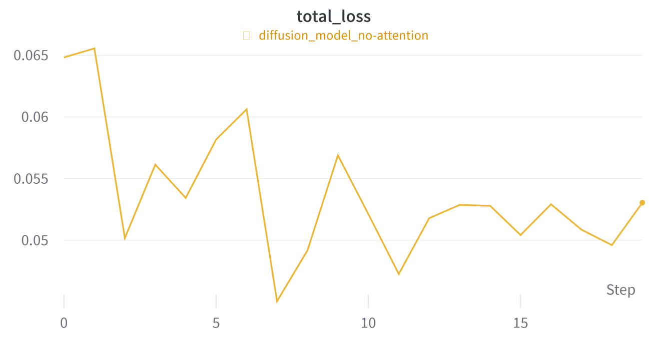


Figure 5: Total loss

Overall, the trends suggest that this model has not yet fully converged, as evidenced by the gradual improvement in the FID score and the lack of stabilization in the total loss. The model's performance, as indicated by the FID score, is considerably above the threshold of top-performing models in the field, such as the Consistency Trajectory Model (CTM), with an FID of 1.73 on CIFAR-10. The FID score's plateauing could imply that the model's current capacity has been maximized for the complexity of the CIFAR-10 dataset. To further improve the FID score, it might be necessary to introduce model refinements or additional training techniques. The total loss's downward trend implies that the model may continue to improve with further training. However, if the loss does not begin to stabilize, it could indicate a need to review the learning rate or other hyperparameters to ensure stable convergence.

To further evaluate the performance of our model, we conducted a comparative experiment using a state-of-the-art diffusion model with pre-trained weights as a benchmark. In our comparative analysis, we focus on two core aspects: quantitative evaluation and qualitative observation. First, we quantitatively compare the two models by means of the Fréchet Inception Distance (FID) Score. Although there is a gap between our model and the reference model regarding FID Score, this is not a comprehensive picture of all aspects of model performance. Therefore, we further performed a qualitative analysis, i.e., a visual comparison of the images generated by the two models (as shown in Figures 6 and 7).

In the visual comparison, although the FID scores showed differences in performance, we observed that the two were extremely similar in terms of visualization. This direct image comparison provides us with a more intuitive performance evaluation that demonstrates the visual effectiveness of our model. The ability of our model to generate high-quality images comparable to the benchmark model indicates that our experiments were successful.

4 User Interface

Gradio provides an easy and efficient way to build and share interactive demos of machine learning models or data science projects, all within the Python ecosystem. This accessibility and ease of use make it an excellent tool for data scientists and machine learning engineers looking to showcase their work and collaborate with others.

We have successfully integrated our de-noising model into the Gradio UI interface. The importance of this step is that it provides an intuitive platform for any user to easily use our model to generate de-noised images. Figure 8 shows our UI interface. We can change the batch-size of the images inputting in the model, from 1 to 3. Because in our model, if we input the images more than 4, we can not get the better results.

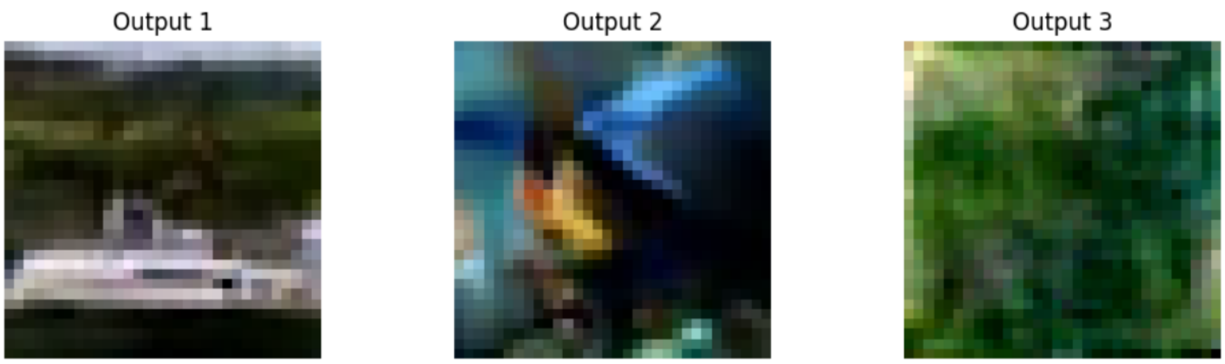


Figure 6: Our Model's Output On Gridio

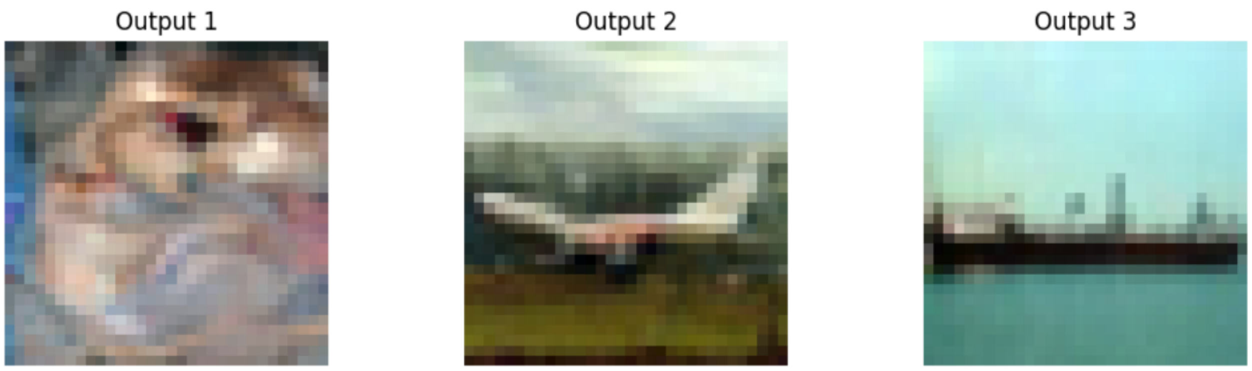
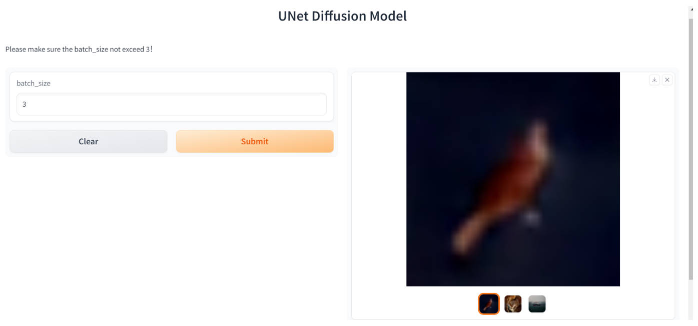


Figure 7: Other's Model[10] Output On Gridio


UNet Diffusion Model

Please make sure the batch_size not exceed 3!

batch_size: 3

Figure 8: UI Interface

5 CONCLUSION

Our replication and exploration of diffusion models using the CIFAR-10 dataset primarily served as an educational endeavour rather than one aimed at surpassing the achievements of the original research. While our results did not exceed those found in the seminal paper, the process provided us with a deeper understanding of diffusion model intricacies and the challenges inherent in such complex generative models. This study has been instrumental in enhancing our knowledge and skills in the field of generative AI, offering valuable lessons that extend beyond mere performance metrics.

6 Future work

Observing the results of the current training, we propose a number of methods to improve the result further.

- Model Architecture Refinement: Given the current FID score, further refinements in the model architecture are crucial. Incorporating advanced techniques such as self-attention mechanisms or exploring larger and more diverse datasets for training might yield significant improvements.

- Training Duration and Data Augmentation: Our results indicate a positive trend with prolonged training. Investigating the impact of extended training epochs and integrating sophisticated data augmentation strategies could enhance the model's ability to generalize and produce higher-quality images.

- Hyperparameter Optimization: Adjusting hyperparameters, including the learning rate, batch size, and noise schedule, could improve model performance. Automated hyperparameter tuning methods, such as Bayesian optimization, might offer substantial benefits.

- Comparative Analysis with Advanced Models: Engaging in a more comprehensive comparative analysis with state-of-the-art models, such as CTM, would provide deeper insights. This involves not only comparing FID scores but also examining qualitative aspects of the generated images.

- Adversarial Training Techniques: Integrating adversarial training techniques, as seen in models like SR3+, which achieved a significant improvement in FID scores, could be a promising avenue. This approach might help in better capturing the distribution of the CIFAR-10 dataset.

References

[1] C. G. Turhan and H. S. Bilge. “Recent trends in deep generative models: a review”. *2018 3rd International Conference on Computer Science and Engineering (UBMK)*. IEEE. 2018, pp. 574–579.

[2] J. Ho, A. Jain, and P. Abbeel. “Denoising diffusion probabilistic models”. *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.

[3] A. Creswell, T. White, V. Dumoulin, et al. “Generative adversarial networks: An overview”. *IEEE signal processing magazine* 35.1 (2018), pp. 53–65.

[4] C. Doersch. “Tutorial on variational autoencoders”. *arXiv preprint arXiv:1606.05908* (2016).

[5] A. Krizhevsky, G. Hinton, et al. “Learning multiple layers of features from tiny images” (2009).

[6] O. Ronneberger, P. Fischer, and T. Brox. “U-net: Convolutional networks for biomedical image segmentation”. *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.

[7] J. S. Hunter. “The exponentially weighted moving average”. *Journal of quality technology* 18.4 (1986), pp. 203–210.

[8] M. Heusel, H. Ramsauer, T. Unterthiner, et al. “Gans trained by a two time-scale update rule converge to a local nash equilibrium”. *Advances in neural information processing systems* 30 (2017).

[9] A. Brock, J. Donahue, and K. Simonyan. “Large scale GAN training for high fidelity natural image synthesis”. *arXiv preprint arXiv:1809.11096* (2018).

[10] P. Esser. *PyTorch Implementation of Diffusion Models*. `https://github.com/pesser/pytorch_diffusion`. 2021.