Simple Diffusion: Architecture and Fine-tuning

Ezequiel Cutin ezecutin@umich.edu
Jason Zhang zhangjt@umich.edu
Aarti Phatke aphatke@umich.edu
Bo Pang bopang@umich.edu

1. Introduction

This project focuses on creating a simplified version of ControlNet. Training image diffusion models with additional conditional control through the use of ControlNets is an innovative approach to addressing the resource-intensive and time-consuming challenges of adding additional parameters to image diffusion models.

1.1. Motivation

Training image diffusion models with additional conditional control and inputs can be resource-intensive and time-consuming, often yielding sub-optimal results. These additional inputs can include pose, edges, and semantic segmentation. To address the challenges associated with training image diffusion models with additional inputs, this project proposes a strategy centered around the use of ControlNets. ControlNets are auxiliary networks trained separately using both small ($<$50k) and large ($>$1m) datasets. The smaller dataset facilitates quicker iterations and experimentation, while the larger dataset provides more comprehensive information, contributing to improved overall performance.

ControlNet's deterministic communication capabilities are leveraged to streamline and improve the efficiency of the image generation process. Notably, ControlNet's efficient architecture not only ensures stability but also minimizes data usage, aligning well with the requirements of modern high-performance computing environments.

In real-time applications, where the rapid generation of realistic and coherent images from given poses is crucial, the efficiency gained from this approach becomes a key factor. Industries such as computer graphics, content creation, and interactive systems stand to benefit significantly from the project's focus on improving stability, quality, and efficiency in image generation processes.

The significance of this project lies in reproducing the effectiveness of ControlNet architecture to simpler models to better understand how it functions. ControlNet has been successfully integrated with production-grade Stable Diffusion models, but not simpler more readable models.

1.2. Description & Contribution

We originally planned to implement ControlNet as described in the paper [1] "Adding Conditional Control to Text-to-Image Diffusion Models" by Lvmin Zhang, Anyi Rao, and Maneesh Agrawala from Stanford to be able to use human poses as an input to a production-grade stable diffusion model, for example, SD1.5. The ControlNet architecture is a network built on top of Stable Diffusion, where we lock the parameters for the already existing model, and add an external network that contains the same encoding layers as the Stable Diffusion model. Instead of decoding afterward, we have a couple of zero-initialized convolution layers, whose outputs are fed into the decoding stage of the original Stable Diffusion network. However, we found this difficult as the Stable Diffusion architecture was very complex and it was hard to understand how to integrate ControlNet into it. So we decided to train our own Simple Diffusion model. However, our initial attempt with the Human Poses dataset, characterized by the details and a collection of 20,000+ images, encountered challenges due to limited GPU resources and time constraints. The dataset was extremely varied and generating human-like features proved to be difficult. Thus, we shifted our focus to a more manageable flower dataset (Oxford Flowers 102) and to refining our diffusion model. This strategic pivot allowed us to efficiently operate within our resource constraints while working on the creation of higher-quality images.

2. Related work

2.1. Image to Image

The simple diffusion model in image generation is based on the idea of progressively adding noise to an image and then learning to reverse this process. This idea is essential to many picture-generating techniques because it enables the model to learn intricate structures and patterns from the data.

Image-to-image techniques complement the diffusion model's methodology nicely. These techniques take an input image, transform it (typically by adding noise or making other changes), and then create a new image by using the transformed image together with extra inputs (such as

text prompts). The important part here is that the model preserves important aspects of the original image (such as color and composition) while modifying it; this is a concept that is also shared by the diffusion model, which uses an image's gradually changed versions to teach itself.

One important benefit of the simple diffusion model is its compatibility with image-to-image approaches. These techniques are adaptable, enabling a variety of changes while preserving crucial elements of the source image, such as composition and color. Because of its adaptability, the basic diffusion model can be used for a wider variety of picture-creation activities.

2.2. Depth to Image

By including depth information, the Depth-to-Image model enhances the conventional Image-to-Image method. This can be viewed as an additional layer of complexity to the image-generating process within the diffusion model. The 'third dimension' is added by the model, which learns to reconstruct or produce images based on surface features (such as colors and textures) in addition to adding depth information. This is consistent with the diffusion model's idea of producing increasingly accurate and lifelike images by learning from complicated, multi-dimensional data.

Despite being less complex than the Depth-to-Image model, the simple diffusion model has some benefits that make it better in some situations. One of its main advantages is that it is simpler than models that include depth information or other layers of complexity; It is easier to understand and frequently uses less processing power. This increases the accessibility and ease of implementation of the basic diffusion model, particularly in situations where computational efficiency is critical.

2.3. Adding Conditional Control to Text-to-Image Diffusion Models

ControlNet [1], designed to enhance large, pre-trained text-to-image diffusion models, integrates with these models which are grounded in the diffusion process of gradually transforming images by adding and then reversing noise. By doing so, ControlNet taps into their innate ability to generate detailed and realistic images from textual descriptions. ControlNet enhances the overall functionality and applicability of the diffusion model in producing controlled, high-fidelity images by adding an extra layer of control while maintaining the diffusion model's fundamental generative capabilities.

2.4. Denoising Diffusion Probabilistic Models

The diffusion probabilistic models described by Jonathan Ho [2] represent a refined class of latent variable models influenced by non-equilibrium thermodynamics, sharing a core principle with the simple diffusion model in the realm of image generation. Both approaches involve a process of systematically transforming an image by adding noise and then mastering the reversal of this noise addition, to either recreate the original image or generate new visuals.

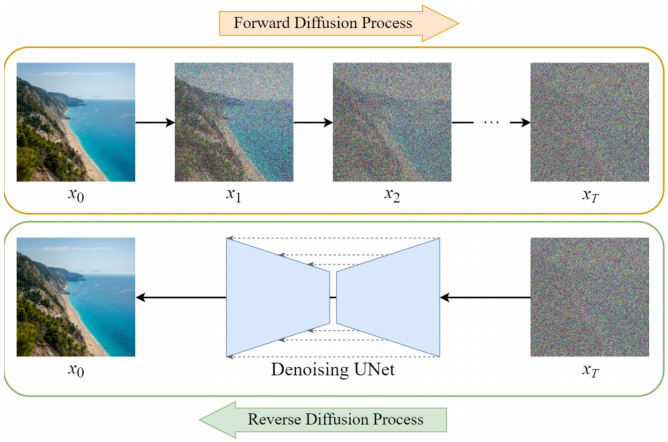


Figure 1. Process of training diffusion

While the diffusion probabilistic models described by Jonathan Ho are advanced, there are certain scenarios where the simple diffusion model may be considered better. The simple diffusion model offers a more straightforward approach, making it easier to implement and understand, especially for those new to the field of AI image generation. This simplicity can be a significant advantage in educational or resource-limited settings. In addition, simple diffusion models require fewer computational resources.

3. Method

3.1. Data Preprocessing

We preprocessed the images from the Oxford 102 Flowers to be able to feed them into the model efficiently. We resized the pictures to 64x64 resolution in order to make the images consistent as well as save memory. We also applied a random flip to the images to augment the data. Then the image data was rescaled from [0, 255] to [-1, 1] to facilitate faster and better training.

3.2. Architecture of the Simple Diffusion Model

The architecture of our simple diffusion model integrates a closed-form forward diffusion process for generating noisy images with a UNet-based model designed to predict and remove noise, as visualized in Figure 1. The timestep embedding introduces the concept of shared parameters across time, and our training process is built to consider memory efficiency and the backward pass with pre-computed noise variances, aiming to efficiently generate realistic images while controlling noise.

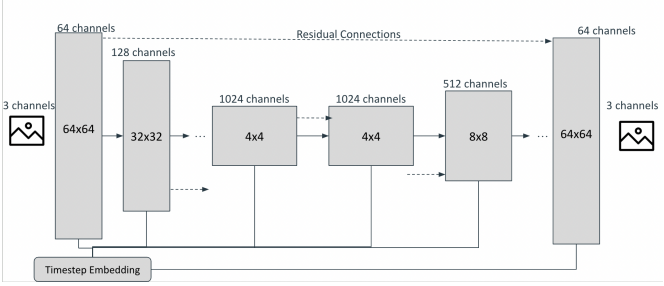


Figure 2. Our UNet architecture

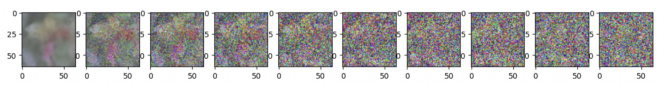
Leveraging the well-established UNet architecture, a proven framework in image processing, our simple diffusion model incorporates an encoder, decoder, and timestep embedding component. The encoder captures essential features from the input image, effectively embedding information using increasing amounts of layers. In contrast, the decoder uses this encoded data to reconstruct the image, ultimately generating a denoised version. The inclusion of a timestep embedding is crucial, introducing a temporal element that enhances the model's ability to handle the diffusion process over various time steps.

Visualized in Figure 2, the UNet architecture comprised of multiple blocks of double-convolutions that increase in thickness but decrease in width and height. 5 downsampling blocks made up the encoder, and 5 upsampling blocks made up the decoder. For the timestep embedding, we used a sinusoidal embedded. This architecture has demonstrated effectiveness across diverse image-related tasks, making it a fitting choice for our diffusion model.

Our model used a systematic approach to calculate loss in the training process. Through batch iteration, the F1 losses are computed for each batch, and the Adam optimizer is applied to optimize model parameters based on these calculated losses.

3.3. Results - Simple Diffusion V1

Our initial results were promising, we had successfully implemented a model that had decreasing loss shown in Figure 3. The losses for this initial model always converged at around 0.15. The pictures that it generated were definitely a step in the right direction, however, they were extremely blurry and the shapes were indistinct. Furthermore, the RGB values seemed to never go close to zero, as most of the colors had a greyish muted tone, indicating similar values in all 3 channels.



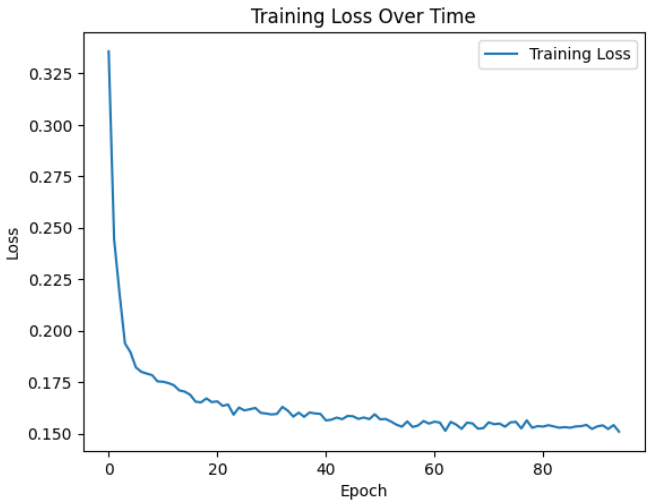


Figure 3. Loss graph of Simple Diffusion V1

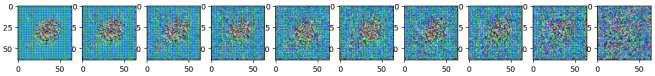


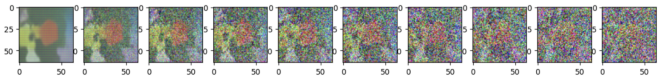
Figure 4. V2 error

3.4. Improvements

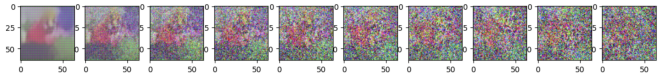
After considering what weaknesses our current model had and doing further research into how to improve the model, we came up with a few improvements to our design. The first improvement that we made was to initialize weights according to the Xavier Uniform distribution. $U(-a, a)$, where $a = gain * \sqrt{\frac{6}{fan_in + fan_out}}$. Good weight initialization can make loss decrease faster and also help the model converge to a better minimum. The second improvement was to include a ReLU step after the batch norm instead of before it. However, that resulted in an interesting issue of certain patches of the diffused images not being altered, shown in Figure 4 above. We hypothesized that this was due the a vanishing gradient issue, so we replaced ReLU with LeakyReLU(0.2). Third, increasing the depth allowed the model to process more complex details. Lastly, as stated in [3] multiplying the residual connections by a constant value, $\frac{1}{\sqrt{2}}$, improves results as well.

3.5. Results - Simple Diffusion V2

After making the aforementioned changes, our results started to look much different. Our model produced images that had much more vibrant colors, with more defined clear edges. The shapes were much easier to identify, however, they were very abstract and globular. The loss at this stage converged to around 0.135.



After adding another layer to the model and letting it run longer we got to a model that generated realistic-looking flowers. The colors became slightly more muted. Structural features of the flower such as petals, leaves, and stamen were visible and the colors generated started to make more sense. The loss at this stage converged to around 0.12. Pictured below is one of the best images we were able to generate. It looks like a flower with 3 pink petals and yellow stamens.



4. Experiments

To comprehensively fine-tune the performance of the proposed image diffusion model, several experiments were conducted, focusing on hyperparameters that influence the model's training and output.

4.1. Batch Size Variation

The impact of batch size on the model's training dynamics and output quality was systematically investigated. Different batch sizes were experimented with to assess their effects on convergence speed, training stability, and the overall quality of denoised images. This exploration aimed to find an optimal balance between computational efficiency and model performance, considering the available GPU resources and the size of the dataset. Online sources suggested that smaller batch sizes led to better generalization but longer runtimes, and larger batches had the opposite effect. However, through our study, we discovered that batch sizes had a different impact on time, increasing the runtime until a batch size of 64 and then decreasing dramatically afterward, as shown in Table 1. In general, smaller batch sizes gave us better results and lower losses but incurred longer runtimes. We settled for a batch size of 128 in the end.

4.2. Timestep Variation

We also investigated the impact on the number of timesteps that we set for the diffusion process. The default values for image diffusion seemed to be $T = 1000$. However, setting $T = 1000$ led to runtimes being dramatically longer, which we could not afford due to our limited resources. As the value of T increased, the amount of time it took to run an epoch also increased as shown in Table 1. Through our thorough investigation, we decided that $T = 500$ was the best value that offered good results while still being relatively efficient to train.

Batchsize	32	64	128	256
$T = 300$	58.18	59.36	52.99	50.87
$T = 500$	57.43	59.54	52.16	52.51
$T = 1000$	60.32	63.21	52.45	53.32

Table 1. Number of seconds it takes to run one epoch

5. Conclusions

We conclude that even with a simple bare-bones implementation of diffusion, a model can generate powerful results with enough training and refinement to its architecture. The dataset we trained on was relatively small (around 7000 images total), yet we were able to get somewhat realistic results. This work can be easily reproduced using our Jupyter Notebook. Due to the simplicity of our design, it doesn't generate as realistic of images as other models like Stable Diffusion, but it is much less computationally intense.

6. Future Work

6.1. Attention

Attention mechanisms have proven to be a powerful tool in enhancing the performance of various machine learning models, particularly in tasks involving sequential or spatial data. The core idea behind attention is to allow the model to dynamically focus on different parts of the input, assigning varying weights to each element based on its relevance to the current context.

In our model, we attempted to add a self-attention mechanism to each block of the diffusion model. This attention mechanism, based on scaled dot-product attention, allows the model to attend to different parts of the input feature maps dynamically, enhancing its ability to capture long-range dependencies and spatial relationships. This ultimately contributes to improved synthesis and generation of realistic images. However, these benefits come with trade-offs, including computational complexity, potential overfitting risks, and hyperparameter sensitivity. While our model architecture was relatively simple, it increased the computation time for training substantially. Due to our limited resources, we were not able to train the model for enough epochs to get a satisfactory outcome.

6.2. Patch Diffusion: Faster and More Data-Efficient Training of Diffusion Models

In recent research, the work titled "Patch Diffusion: Faster and More Data-Efficient Training of Diffusion Models" [4] by Zhendong Wang addresses the substantial time and data requirements associated with training powerful diffusion models. The proposed Patch Diffusion framework introduces a generic patch-wise training approach, substantially reducing training time costs and enhancing data efficiency.

References

[1] Zhang, L., Rao, A., & Agrawala, M. (2023). Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 3836-3847).

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

[3] Dhariwal, P., & Nichol, A. (2021). Diffusion models beat GANs on image synthesis. *Advances in neural information processing systems, 34*, 8780-8794.

[4] Wang, Z., Jiang, Y., Zheng, H., Wang, P., He, P., Wang, Z., ... Zhou, M. (2023). Patch diffusion: Faster and more data-efficient training of diffusion models. *arXiv preprint arXiv:2304.12526*.