Midterm Creating Images with Diffusion Models

Martin Demel

Department of Science, Technology, Engineering & Math, Houston Community College

ITAI 2376 Deep Learning in Artificial Intelligence

Patricia McManus

April 9th, 2025.

Introduction

My goal in this laboratory exercise was to explore the theory and application of diffusion models for image generation, specifically focusing on creating handwritten digits that could be recognized as MNIST characters. I began by setting up a suitable environment on my MacBook Pro with GPU acceleration enabled, installing the necessary Python packages, and fixing seeds for reproducibility. This foundational work was essential to ensuring a consistent training experience and minimizing any randomness that might confound my observations.

One key learning outcome was understanding why diffusion models systematically add noise to images and then learn to remove that noise step by step in reverse. This methodical "noise-adding and noise-removing" approach helped me see the gradual denoising in action, reinforcing how diffusion models capture each stage of image degradation and recovery. From a broader perspective, this project taught me that generative AI can be orchestrated in a more fine-grained manner than many other generative methods, leveraging incremental refinements across hundreds of time steps.

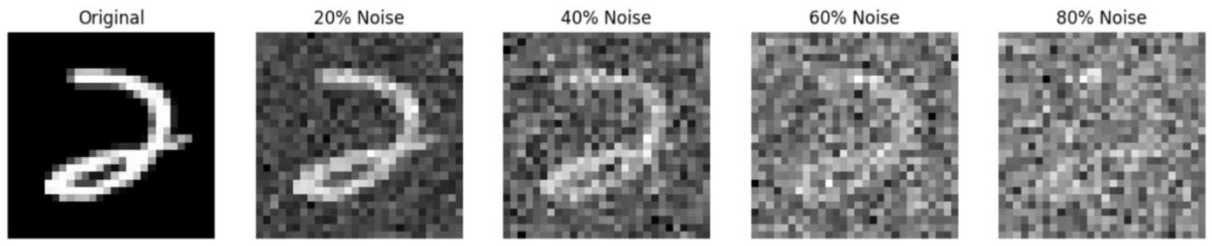
My choice of the MNIST dataset was guided by the requirement for a dataset that trained quickly and fit comfortably within my MacBook Pro's GPU memory. The images, at 28×28 in grayscale, provided a straightforward setting to test the diffusion pipeline without incurring prohibitive hardware demands. As part of the deliverables, I prepared a thorough analysis of the theoretical constructs, the architectural decisions (centered on a U-Net backbone), the diffusion schedule, the training process, and the evaluation with the optional CLIP integration.

Body

One of the initial hurdles was confirming that I had enough computational resources. After validating my GPU setup locally on the MacBook Pro, I divided the MNIST dataset into training (80%) and validation (20%), employing a transform pipeline that normalized the images to [–1, 1]. Basic data integrity checks, such as ensuring shapes were correct, data types were float, and that no NaN values existed, enabled me to proceed confidently with data loading.

A critical part of the laboratory requirement was implementing a U-Net for the diffusion tasks. This model leverages down-sampling blocks, a bottleneck layer, and up-sampling blocks, using skip connections to retain crucial spatial features at different resolution stages. In my code, each down block consisted of two convolutional layers followed by a pixel-based pooling operation that rearranged feature maps to halve spatial dimensions. The up blocks reversed this by concatenating skip connections, applying transposed convolutions to upsample, and then passing data through additional convolutional blocks.

This architecture is well-suited for diffusion because the skip connections maintain the image's essential structural details across different resolution levels. When the model is denoising images at various timesteps, these skip connections help the network reconstruct fine details that might otherwise be lost. A sinusoidal time embedding injects information about which denoising step the network is working on, effectively grounding the model in the “time” dimension of the diffusion process. Additionally, a class embedding allows the model to condition on a one-hot representation of digits, ensuring that the generated output can be steered toward a specific number.

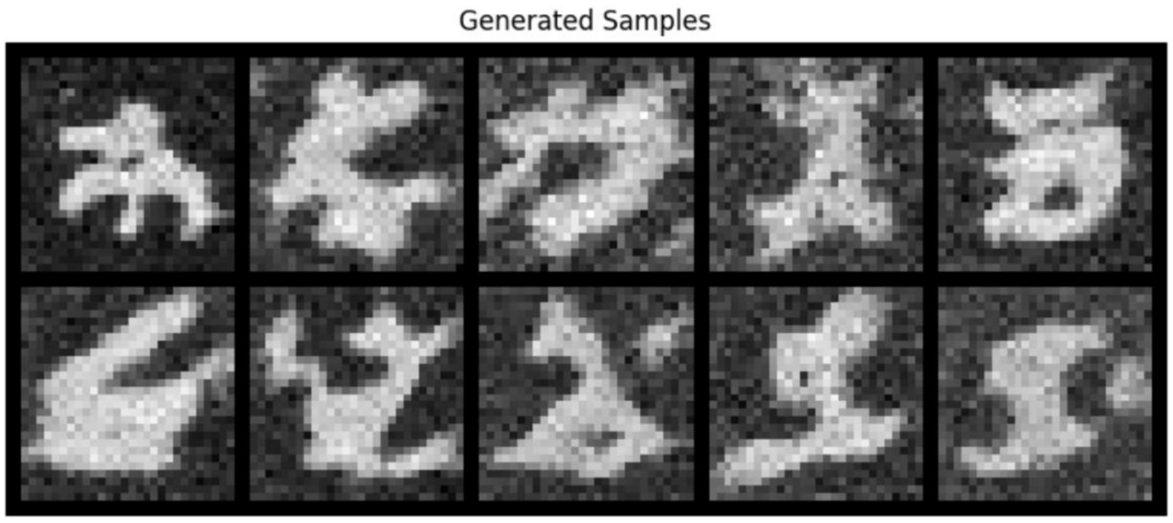


In the forward diffusion phase, I implemented a function that incrementally adds noise to the images, regulated by a schedule of beta values. This gradual increase in noise at each timestep contrasts with adding all the noise at once. The latter would make it nearly impossible for the model to learn the nuanced reverse mapping. By adding noise progressively, each intermediate step in the denoising stage corresponds to removing just a thin slice of noise rather than an overwhelming scramble. From my observations, an image generally becomes recognizable after about 70–80% of the steps in the reverse process, but this can vary with digit shape and clarity.

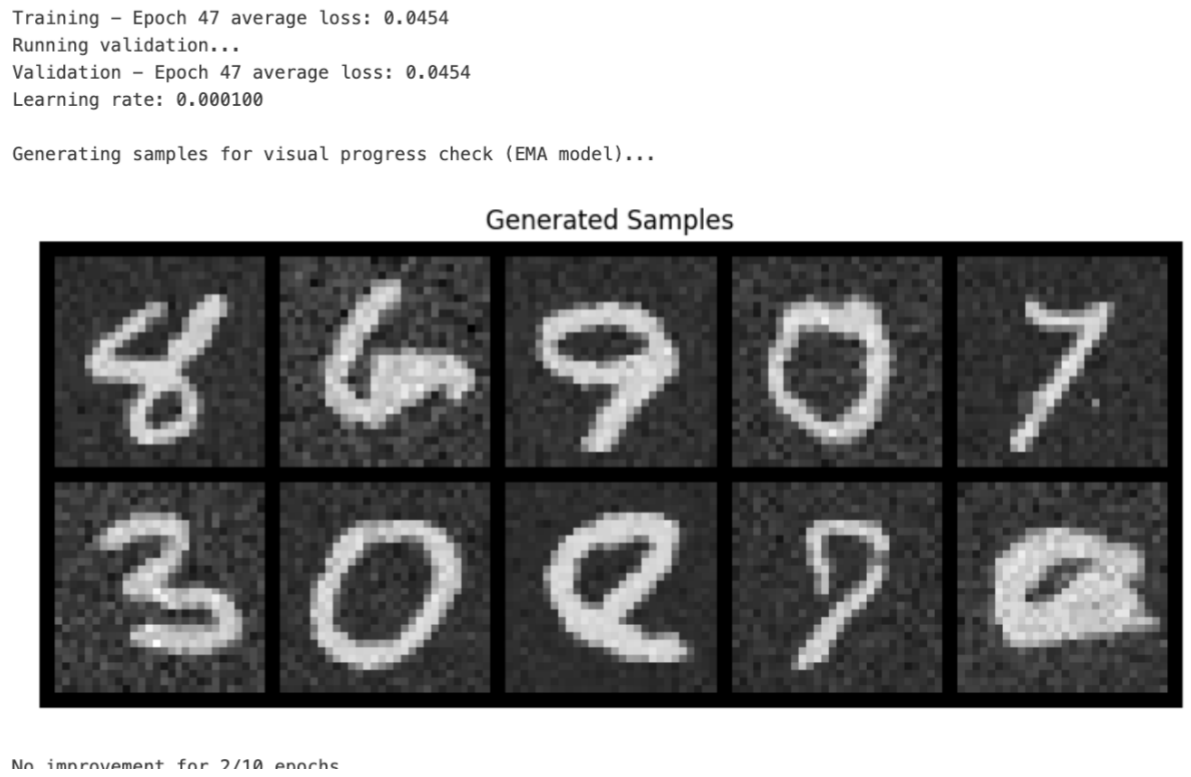
Initially, I set up only 100 diffusion steps. Although the model's loss decreased, my early attempts at sampling produced random blob-like shapes instead of clear digits. Reflecting on this challenge, I realized that the diffusion process greatly benefits from a higher number of steps. By extending the schedule to 300 timesteps, faint digit shapes began to emerge. With 500 timesteps, the generated digits sharpened significantly, highlighting that more, finer-grained denoising steps enable the model to recover features more accurately. Indeed, the model architecture and optimizer settings remained the same; merely changing the number of diffusion steps was the critical adjustment.

```
Training - Epoch 3 average loss: 0.0644
Running validation...
Validation - Epoch 3 average loss: 0.0635
Learning rate: 0.000200

Generating samples for visual progress check (EMA model)...
```



During training, my primary metric was the mean squared error between the predicted noise and the actual noise injected into each sample. Monitoring both the training and validation loss allowed me to adjust the learning rate and detect any signs of overfitting. I employed an exponential moving average (EMA) strategy for the model weights. The EMA model typically generated cleaner outputs than the raw model checkpoint, so all final samples and demos utilized the EMA version. By the end of training, the validation loss stabilized, and the generated images closely matched the intended digit class.

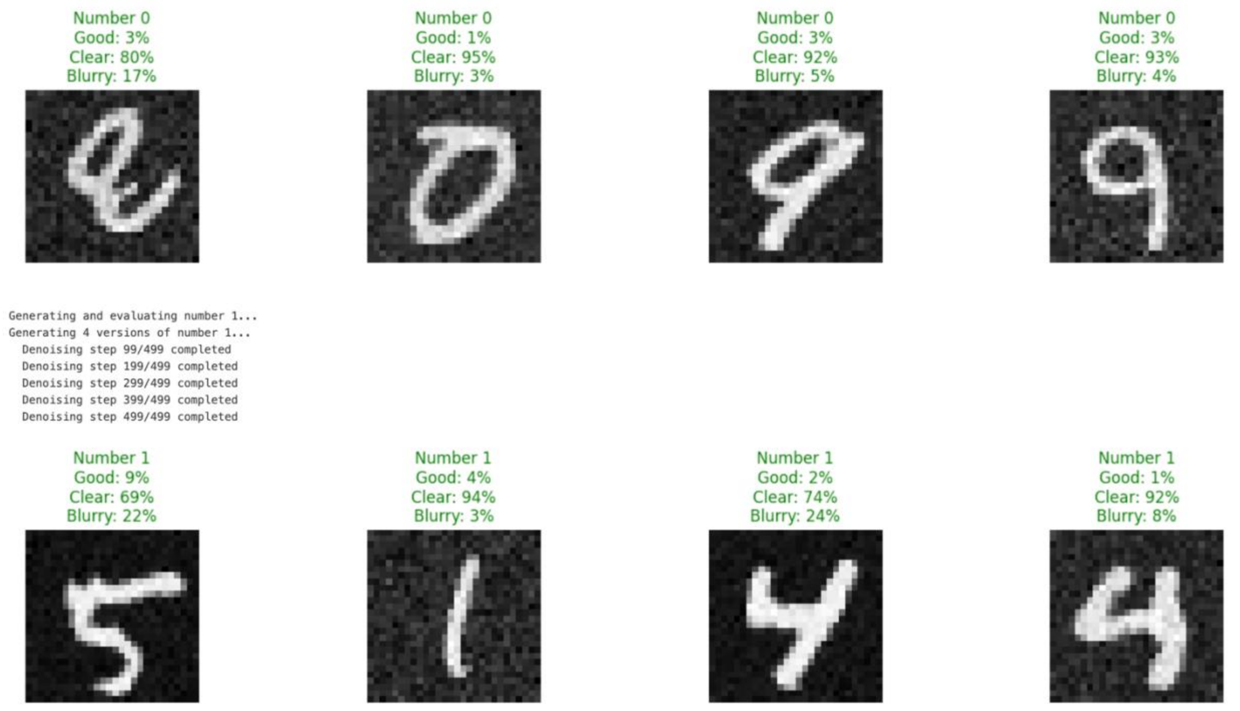


```
Training - Epoch 47 average loss: 0.0454
Running validation...
Validation - Epoch 47 average loss: 0.0454
Learning rate: 0.000100

Generating samples for visual progress check (EMA model)...
```

```
No improvement for 2/10 epochs
```

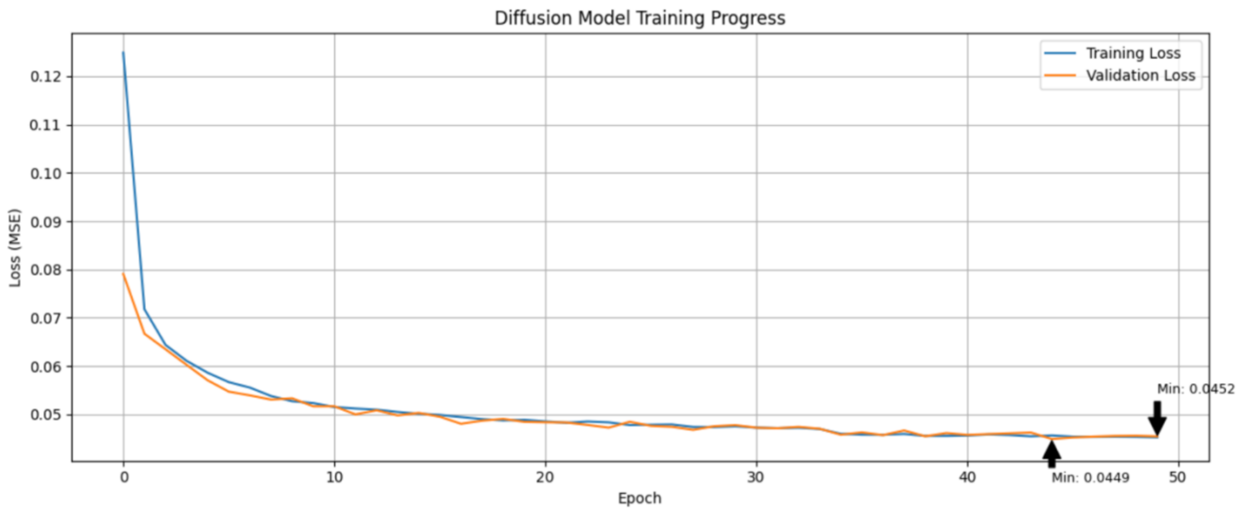
To further analyze the model's performance, I integrated a CLIP-based evaluation. While optional, this step provided an exciting perspective: CLIP can score how well the generated images align with textual prompts like "A handwritten number 7" or "A blurry, unclear number.” In practice, the best outputs matched high CLIP scores for the correct digit label and clarity.



```
Generating and evaluating number 1...
Generating 4 versions of number 1...
  Denoising step 99/499 completed
  Denoising step 199/499 completed
  Denoising step 299/499 completed
  Denoising step 399/499 completed
  Denoising step 499/499 completed
```

Some partially malformed or poorly shaped images received relatively higher scores for being “blurry” or “unclear.” This finding suggests that CLIP could guide the diffusion process by reinforcing well-formed results. One potential improvement is using such feedback to rank the generated samples, discarding the lower-scoring ones or even steering the training objective with CLIP in a reinforcement-like manner.

The robust forward and reverse diffusion framework, combined with a well-structured U-Net, demonstrates why diffusion models are so effective for image generation tasks. They systematically model each step of noise injection and removal, carrying fine-grained knowledge about how to reconstruct details from partial noise states. While the MNIST dataset is relatively simple, this exercise provides an excellent introduction to diffusion fundamentals before tackling more complicated data like CIFAR-10 or high-resolution images.



Throughout the experiment, I found that the time embedding is crucial in helping the network understand “where” it is in the denoising pipeline. Without a sense of the current timestep, the model would not know how aggressively to remove noise or which features to restore. This aligns with the theoretical idea that each diffusion step can be treated almost like a small generative model conditioned on a fraction of the remaining noise distribution.

In practical terms, diffusion models like this can be applied to a range of applications requiring diverse, high-fidelity sample generation, from art creation to data augmentation. However, it is essential to acknowledge certain limitations in the current setup. First, the model is constrained to a small image size (28×28) and a single dataset class. Second, training time can increase significantly for large images due to the numerous diffusion steps. Finally, while the results are good for simple digits, real-world images would require more robust conditioning, larger U-Nets, or advanced scheduling (and significantly more computational resources).

If I were to expand this project, my plans would include experimenting with style conditioning, exploring advanced noise schedules, and integrating CLIP-based feedback loops to prioritize only the highest-quality images for further refinement. Such improvements could make the generative process even more precise and versatile. In scenarios like medical imaging or other critical domains, these carefully tuned diffusion models are invaluable for generating new, realistic images that still honor the underlying data distribution.

Conclusion with reflection

This lab integrated theoretical diffusion knowledge with practical experience in implementing, training, and evaluating a U-Net-based diffusion model. Introducing noise in small increments and iteratively denoising the images highlighted the importance of many timesteps for clarity and fidelity. The training loss consistently decreased, and the final samples were crisp, legible digits, providing strong evidence of the model's learned capability.

The synergy of skip connections in the U-Net, class conditioning, and time embeddings underlined how each design decision influenced the quality of the generative process. Through systematic experimentation, the number of diffusion steps can be adjusted, and eventually, CLIP can be leveraged for further evaluation. I deeply appreciated how generative diffusion can achieve clear, structured results. This project also offered a perspective on potential applications, from creative image generation to more specialized tasks, highlighting the importance of methodical iteration improvements.

Resources

GeeksforGeeks. (2024, June 6). What are Diffusion Models? GeeksforGeeks. https://www.geeksforgeeks.org/what-are-diffusion-models/

bot66/MNISTDiffusion: Implement a MNIST(also minimal) version of denoising diffusion probabilistic model from scratch.The model only has 4.55MB. (2022). GitHub. https://github.com/bot66/MNISTDiffusion

GeeksforGeeks. (2024, May 31). MNIST Dataset : Practical Applications using KERAS and PyTorch. GeeksforGeeks. https://www.geeksforgeeks.org/mnist-dataset/

GeeksforGeeks. (2021, December 12). How to calculate an exponential moving average in Python? GeeksforGeeks. https://www.geeksforgeeks.org/how-to-calculate-an-exponential-moving-average-in-python/