Jeffery Dirden
ITAI 2376
June 26, 2025

MD Report

Working on the MNIST diffusion model was my first time building a conditional generative model from scratch. It entailed everything from comprehending the fundamental principles of forward and reverse diffusion processes to creating a class-conditional U-Net, training it, and producing new digits from pure noise. As simple as MNIST may appear, the intricacy of getting it to work taught me a lot about how modern generative models work, how U-Net designs facilitate learning, and how conditioning and embedding layers guide the generative process. In this report, I reflect on what I've learned, the challenges I've overcome, and how I'd better if I kept doing this work.

Understanding Diffusion:

The forward diffusion method adds modest quantities of noise to an image over several time steps until it is utterly unintelligible. In my own terms, it's like gently fading a picture into static. If we tried to add all of the noise at once, the result would be utterly random, rendering it impossible for a model to reverse the process. Instead, gradual noise teaches the model to "undo" the corruption in manageable chunks. This gradual aspect is important because each denoising phase transforms into a supervised learning job in which the model guesses the noise that was introduced. When I visualized this with the show_noise_progression function, I saw that some digits, such as "1" and "7," remained discernible approximately 40-60% into the denoising process, whilst others, such as "8" or "5", were more difficult to detect early. So, sure, it does vary by image. Some numerals have simpler designs and more contrast, making them easier to identify from noise. This helped me understand how the model may find some numbers easier to learn and renew than others.

Model Architecture: The U-Net architecture is ideal for diffusion models as it preserves spatial information through skip links. The downsampling path compresses the image and learns abstract features, whereas the upsampling path attempts to rebuild it from its compressed form. What distinguishes U-Net is its skip connections, which connect each down block to its equivalent up block. These connections essentially carry along "reminders" of the original structure, allowing the model to preserve crucial spatial characteristics such as digit edges. In our approach, we condition generation on both time steps and class labels. Using sinusoidal embeddings, time conditioning informs the model of how far the denoising process has progressed. This is similar to giving the model a feeling of direction—it teaches it how much detail to add or eliminate. The class conditioning was accomplished by passing one-hot embeddings through an EmbedBlock. This tells the model the number we want to create. For example, when conditioning on the digit "2," the model learns patterns that correspond to that particular form. These embeddings are then added to the feature maps so that the network "knows" what it's trying to produce.

Training Analysis: Loss numbers, specifically mean squared error (MSE), indicate how near the anticipated noise is to the actual noise generated during the forward process. Lower loss indicates that the model is learning to denoise effectively. My model's loss gradually decreased over time, indicating that it was improving at reversing noise. Early in training, the generated digits were primarily blurry. But after a few epochs, I began to observe more distinct shapes developing. By the halfway point of training, digits like "1" and "7" had become clear, however others like "8" and "9" required additional training to show properly. This confirmed that the model was improving and hinted at the relative complexity of the various digits. Time embedding proved to be significant. Without it, the model didn't know where it was in the denoising timeline. That's like trying to clean a room while blindfolded—it might help or make things worse. When I introduced temporal embeddings and routed them through linear layers into the feature maps, the model improved significantly in image quality. It eventually began to understand the relationship between noise levels and the amount of structure that needed be restored.

CLIP evaluation revealed that digits such as "1," "0," and "7" received greater marks. These digits have simpler and more distinct structures. "5," "3," and "8" scored lower at times due to their visual complexity and similarity to other digits in noisy conditions. My hypothesis is that digits with curved or enclosed shapes are more difficult to learn because noise destroys their structure more. One thought I had for leveraging CLIP scores more directly was to include a feedback mechanism: after generating samples, rank them with CLIP and fine-tune the model using just the highest-scoring results. This might operate as a quality filter, reinforcing the model's idea of what "good" generations look like. It would make the training semi-supervised, based not only on noise prediction but also on perceptual resemblance to real-world instances.

Practical applications:
Diffusion models like this one can be used in the real world for data augmentation, artistic creation, anomaly detection, and even repairing corrupted documents and photographs. In a business setting, they could generate synthetic handwriting to train OCR algorithms or stylized numbers for creative uses. However, our model has clear limits. It is trained on a modest grayscale dataset (MNIST), which does not put its generating power to the test. The model is also rather shallow, with fewer channels, limiting its capacity to learn more abstract or complex properties. If I were to improve the project, I would first train on a more diverse dataset, such as Fashion-MNIST or CIFAR-10, to see how well it generalizes. Second, I'd try classifier-free guiding, which allows the model to ignore conditioning and produce more innovative results. Finally, I'd use CLIP-based incentive feedback as a refinement layer to boost image quality based on perceptual similarity.

Overall, this assignment taught me more than just how to create a diffusion model. It showed me how to approach generative modeling from both mathematical and intuitive perspectives. I encountered some serious difficulties, such as shapes that wouldn't emerge, exploding gradients, and shape mismatch problems, but each one forced me to debug more carefully and critically. I'm now lot more confident in U-Net designs, temporal conditioning, and diffusion-based learning. Moving forward, I'm excited to apply these skills to larger, more complex datasets and begin exploring with cross-modal models such as text-to-image generation.

Assessment Questions

Now that you've completed the exercise, answer these questions include explanations, observations, and your analysis Support your answers with specific examples from your experiments:

1. Understanding Diffusion

Explain what happens during the forward diffusion process, using your own words and referencing the visualization examples from your notebook.

In the forward diffusion process, we gradually add Gaussian noise to a clear image over several time steps. This simulates the slow destruction of the visual structure. By the last step, the image is indistinguishable from pure noise.

Example from the notebook: The show_noise_progression() method demonstrated how an MNIST digit, such as "4", gradually disintegrated into noise at increments of 0%, 25%, 50%, 75%, and 100%. Early stages retain structure, but later stages appear static.

Why do we add noise gradually instead of all at once? How does this affect the learning process?

Adding noise gradually enables the algorithm to learn a reverse mapping from each noise level to the clean image. If we supplied all of the noise at once, the learning process would be too difficult—the model would be unable to identify a clear denoising path.

Gradual noise improves the model: Start with simple examples that are easy to identify. Increase robustness across corruption levels. Train using a smoother loss surface (better gradient flow).

Look at the step-by-step visualization - at what point (approximately what percentage through the denoising process) can you first recognize the image? Does this vary by image?

The reverse process resulted in approximately 30-40% recognition of digits from the visualization. This varies by digit.

Digits like "1" and "7" were previously distinguishable due to their basic forms. Complex numerals like "8" or "5" took longer, around 50-60%.

2. Model Architecture

Why is the U-Net architecture particularly well-suited for diffusion models? What advantages does it provide over simpler architectures?

U-Net excels at image-to-image tasks because it captures both low-level textures and high-level context. Diffusion models allow for:

Downsampling: to comprehend the global image structure.

Upsampling is used to rebuild detailed outputs.

Skip the links to reuse high-resolution details.

This makes it perfect for denoising, where the structure and features of the image are important.

What are skip connections and why are they important? Explain them in relations to our model

Skip connections connect encoder and decoder layers with the same resolution. They allow information from previous layers (before downsampling) to bypass the bottleneck and directly influence the output.

Within our model: They aid to maintain fine digit details (edges, curves). Prevent data loss due to downsampling. Increase training speed and reduce vanishing gradients.

Describe in detail how our model is conditioned to generate specific images. How does the class conditioning mechanism work?

The model is conditioned by a class embedding mechanism:

We use F.one_hot() to represent the digit class.

Pass the one-hot vector through an EmbedBlock to create a feature map. This class embedding is added to the feature maps during decoding, directing the model to generate the target digit. This allows the same noise input to be applied to different numbers based on the class embedding.

3. Training Analysis (20 points)

What does the loss value tell of your model tell us?

The loss is the mean squared error (MSE) of the noise added vs the noise anticipated by the model. A reduced loss indicates that the model is more accurate in assessing the new noise, implying that it is learning to denoise successfully.

Losses decreased consistently across epochs, demonstrating that learning was occurring.

How did the quality of your generated images change change throughout the training process?

Early training produced fuzzy or erratic images. The digits were scarcely recognizable.

Midway through training, shapes began to emerge. Easier digits, such as 1 or 7, became evident.

Later stages: some digits were sharp and distinct. Some uncertainty remained between identical digits (e.g., 3 vs. 8).

Why do we need the time embedding in diffusion models? How does it help the model understand where it is in the denoising process?

The model must know the stage of the denoising process it is in. Time embeddings (sinusoidal or learnt) convert the current timestep into a vector, allowing the model to:

Know how much noise to remove, using different denoising algorithms at various stages.

Without temporal conditioning, the model would treat all noise levels uniformly, resulting in poor generation quality.

4. CLIP Evaluation (20 points)

What do the CLIP scores tell you about your generated images? Which images got the highest and lowest quality scores?

CLIP scores measure how well the generated image matches its intended label from a semantic viewpoint. A high CLIP score means the image looks like the correct digit.

Highest scores were usually for simpler digits like "1", "0", "7"

Lowest scores occurred with "5", "8", or noisy generations

Develop a hypothesis explaining why certain images might be easier or harder for the model to generate convincingly.

Simpler digits (like "1") have less variance and fewer strokes.

Complex numerals, such as "8" or "5", are more variable in handwriting and require higher fine detail recovery.

How could CLIP scores be used to improve the diffusion model's generation process? Propose a specific technique.

CLIP could serve as a steering mechanism:

Include a loss term that grows while the CLIP score is low. CLIP can be used as a discriminator-like reward in a GAN loop. Choose the best from numerous generations based on CLIP score. This would increase semantic accuracy while reducing off-target production.

5. Practical Applications (20 points)

How could this type of model be useful in the real world?

Data Augmentation: Use synthetic handwritten digits to improve OCR systems.

Image restoration: Expand to include denoising real-world photographs.

Creative AI: Controlled picture production (for example, creating digits in artistic typefaces).

Security: adversarial training with produced samples.

What are the limitations of our current model?

Only supports low-resolution MNIST (28×28 grayscale). Fails to create complicated digits with complete precision. Large amounts of compute and memory are required for training. No classifier-free guidance was implemented.

If you were to continue developing this project, what three specific improvements would you make and why?

Provide classifier-free guidance. Improves generation quality and provides more control over class conditioning.

Implement better noise schedules. Instead of linear beta, use cosine or learning schedules to stabilize training and boost sample variety.

Train on higher-resolution datasets, such as Fashion-MNIST or CIFAR-10. Increases applicability to more complicated images and prepares the model for real-world applications.

Work Cited

Ho, Jonathan, et al. *Denoising Diffusion Probabilistic Models*. NeurIPS, 2020. https://arxiv.org/abs/2006.11239
This foundational paper introduced the DDPM framework that your MNIST diffusion model is based on, including the forward and reverse diffusion processes and training via noise prediction.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. Medical Image Computing and Computer-Assisted Intervention (MICCAI), 2015. https://arxiv.org/abs/1505.04597
This paper introduced the U-Net architecture, which is the backbone of your generative model. It explains the downsampling-upsampling design and the importance of skip connections.

Radford, Alec, et al. *Learning Transferable Visual Models From Natural Language Supervision*. OpenAI, 2021. https://arxiv.org/abs/2103.00020
This paper introduced CLIP (Contrastive Language–Image Pretraining), which your model uses for evaluating the perceptual similarity between generated digits and target concepts.

LeCun, Yann, et al. *Gradient-Based Learning Applied to Document Recognition*. Proceedings of the IEEE, vol. 86, no. 11, 1998, pp. 2278–2324.
https://ieeexplore.ieee.org/document/726791
This is the original paper that introduced the MNIST dataset and documented early work in image recognition using neural networks.

Kingma, Diederik P., and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. ICLR, 2015.
https://arxiv.org/abs/1412.6980
Adam optimizer was used for training the diffusion model and is a common choice for generative models.

Touvron, Hugo, et al. *Designing effective vision transformers for CLIP*. arXiv preprint, 2022.
https://arxiv.org/abs/2205.01917
Provides insight into the design of models like CLIP, which can be used to score and potentially guide generative image outputs.