Parnal Sinha
ITAI 2376 Midterm
Diffusion Model MNIST Dataset
Dr. Patricia McManus

Conditional Diffusion Model on MNIST: Implementation, Training, and Evaluation

Abstract

This report presents a comprehensive implementation of a class-conditional Denoising Diffusion Probabilistic Model (DDPM) on the MNIST dataset. Focusing on three core themes: technical mastery of the diffusion framework, model conditioning for controllable generation and lessons learned in ensuring stable training. The workflow demonstrates the attempted development of a conditional generative model for synthesizing digit images, though with mixed results. The implementation centers on a U-Net architecture with skip connections, incorporating sinusoidal timestep embeddings and classifier-free guidance to enhance model performance. Through rigorous analysis of training dynamics, sampling stability, and evaluation metrics, this report documents the significant technical challenges encountered and partial solutions developed during implementation. Results show that while the conditional diffusion model learned aspects of the MNIST data distribution, it struggled to consistently generate high-quality digit images on demand, with performance validated through both qualitative assessment and quantitative evaluation using CLIP. Despite limitations in the final output quality, the insights gained from this challenging implementation provide valuable lessons for future diffusion model applications.

Introduction

Diffusion-based generative models have emerged as a promising approach for image synthesis, demonstrating capabilities in producing outputs of varying quality and diversity. In this work, I implement and train a class-conditional Denoising Diffusion Probabilistic Model (DDPM) on the MNIST dataset, with the goal of mastering the diffusion process and conditioning mechanisms in a controlled setting.

My diffusion model is conditioned on class labels (digits 0-9), aiming to direct the generation process – an approach inspired by prior conditional diffusion frameworks (Ho et al., 2020; Nichol & Dhariwal, 2021). The implementation centers on a U-Net architecture with skip connections, enabling the model to learn to reverse the forward noising process and reconstruct digit images from pure noise, though with varying degrees of success.

This project began with ambitions to generate high-resolution, attribute-conditioned images (e.g., faces in CelebA) and progressed through multiple datasets, ultimately focusing on MNIST as a more tractable platform to demonstrate diffusion principles. Throughout the project, I emphasize technical rigor in reproducing the diffusion process and incorporate improvements from the literature, such as sinusoidal timestep embeddings and classifier-free guidance, to enhance model performance, though these enhancements didn't always yield the expected improvements.

Key themes of this report include:

Parnal Sinha
ITAI 2376 Midterm
Diffusion Model MNIST Dataset
Dr. Patricia McManus

1. **Technical mastery** of the diffusion framework, implementing forward noising, reverse denoising U-Net, timestep and class embeddings, and sampling strategies
2. **Model conditioning** for controllable generation, attempting targeted synthesis of specific digit classes with mixed success
3. **Lessons learned** in building and stabilizing a diffusion model, addressing numerical instabilities and hyperparameter sensitivities that proved more challenging than anticipated

Early experiments revealed significant challenges – for example, training on higher-complexity datasets led to numerical instabilities in the reverse diffusion sampling. These challenges were partially addressed by refining the noise schedule and guidance strategy and, ultimately, by constraining the scope to the MNIST dataset. This strategic refocus allowed me to validate some core diffusion model components, though the final results fell short of the clear, high-fidelity digit generation I had initially hoped to achieve.

Background & Motivation

Diffusion Models in Context

Generative adversarial networks (GANs) once dominated image generation, but diffusion models have emerged as a promising alternative in various tasks. The seminal DDPM formulation by Ho et al. (2020) introduced a Markovian noising-denoising chain that converts complex data distributions into tractable Gaussian noise and back, enabling likelihood-based training with a simple MSE loss.

Subsequent enhancements—cosine noise schedules, classifier-free guidance (CFG), latent diffusion—have further developed the field, contributing to systems such as DALL·E 2 and Stable Diffusion. These advances have established diffusion models as a significant approach for generative modeling, though implementing these techniques effectively remains challenging.

Why MNIST?

MNIST serves as a testbed for evaluation of diffusion principles for several reasons:

1. **Simplicity**: 10 balanced classes of 28×28 grayscale images allow experimentation without OOM errors.
2. **Benchmarking**: Established generative baselines facilitate comparison.
3. **Pedagogical Value**: The dataset's low resolution and clear semantics enable introspection of sampling dynamics and conditioning efficacy.

By constraining my scope to MNIST's 28×28 grayscale format, I aimed to validate fundamental diffusion components while working within limited computational resources, though even this simpler domain presented significant challenges.

Parnal Sinha
ITAI 2376 Midterm
Diffusion Model MNIST Dataset
Dr. Patricia McManus

Project Goals

1. **Technical Mastery**: Recreate forward and reverse diffusion processes from first principles; implement a U-Net with rigorous handling of skip connections, timestep embeddings, and class conditioning.
2. **Model Conditioning**: Demonstrate controllable digit generation via class embeddings and CFG, enabling on-demand synthesis of any digit 0-9, though with varying success across different digits.
3. **Lessons Learned**: Identify and attempt to mitigate numerical instabilities in sampling; refine hyperparameters and schedule choices for improved convergence, documenting both successes and persistent challenges.

By anchoring on these objectives, I aimed to produce not only a working MNIST generator but also a documented workflow that informs future scaling to more complex domains, with particular attention to challenges that might need to be overcome.

Methodology

Data Preparation

1. **Dataset**: Standard MNIST training (60,000 images) and test (10,000 images), normalized to [-1, 1].
2. **Batches**: 128 images per batch, shuffled each epoch.
3. *Forward Diffusion (Noising)*

The forward diffusion process gradually destroys structure in the data by adding noise in small increments over many time steps. At time t=0, I start with a real image (an MNIST digit). Then, at each diffusion step t=1,2,…,T, a bit of Gaussian noise is added to the image. This process is defined so that after T steps, the image is almost pure noise.

Formally, I construct a Markov chain $q(x_t|x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{1-\beta_t} \cdot x_{t-1}, \beta_t \cdot I\right)$, where β_t is a variance schedule controlling the noise level at step t. In my implementation:

1. **Variance Schedule**: Linear β_t from 0.0001 to 0.02 over T = 100 steps.
2. **Closed-Form Sampling**: Direct computation of $x_t \sim \mathcal{N}\left(\sqrt{\bar{\alpha}_t} \cdot x_0, (1-\bar{\alpha}_t) \cdot I\right)$ via precomputed $\bar{\alpha}_t = \prod_{i=1}^{t}(1-\beta_i)$.

This formulation allows me to take an original image x_0 and sample its noisy version at step t in one go. I verified that as t increases, the samples x_t visually lose clarity—e.g., an MNIST “5” gradually turns into a fuzzy cloud of gray pixels.

Parnal Sinha
ITAI 2376 Midterm
Diffusion Model MNIST Dataset
Dr. Patricia McManus

A key insight from my step-by-step recognition analysis was that information is lost slowly over the diffusion process. Using a pre-trained digit classifier on partially diffused images, I found that classification accuracy dropped as t increased, approaching random guessing around the point when about 50-60% noise was added. This experiment reinforces why the diffusion model's reverse process is theoretically feasible—because each step only adds a little noise, the model can potentially learn to remove that noise incrementally, though achieving this in practice proved challenging.

Reverse Model (U-Net)

I adopted a U-Net architecture as the core of my diffusion model, which is a common choice in recent diffusion research (Ho et al., 2020). The U-Net is well-suited for diffusion tasks because of its encoder-decoder structure with skip connections.

In my context, the network takes a noisy image x_t (plus conditioning) and outputs a prediction of the added noise $\hat{\epsilon}_\theta$. The architecture consists of:

1. **Architecture**: Four downsampling and four upsampling blocks with GELU activations and GroupNorm.
2. **Skip Connections**: Encoder features concatenated to decoder inputs to preserve high-resolution details.
3. **Residual Connections**: Added in each block to ensure stable gradient flow.

The encoder path progressively downsamples the image, capturing a hierarchy of features from local edges up to global shapes. The decoder path then upsamples back to the original resolution, attempting to reconstruct the signal (by predicting noise to remove).

The **skip connections** between corresponding encoder and decoder layers play a critical role: they allow high-resolution details from the encoder to bypass directly to the decoder. This means the model doesn't have to learn to transmit all fine detail through the bottleneck; it can rely on skip connections to recover local information (like the outline of a digit) when denoising.

In my experiments, the U-Net showed mixed effectiveness: while it learned to denoise noisy digit images to some extent, it often struggled with preserving subtle features and creating clear, consistent digit forms.

Conditioning Mechanisms

To condition the image generation on a class label (digit 0-9), I incorporated an embedding of the class into the U-Net. Specifically, I implemented two parallel embedding pathways:

1. **Time Embedding**: Sinusoidal positional encoding of t passed through an MLP to produce a vector e_t.

2. **Class Embedding**: One-hot vector of the digit class passed through an MLP to produce a vector e_c of the same dimension as e_t.

These embeddings are then added into the U-Net at a specific point in the architecture. In my design, I chose to inject e_t and e_c at the bottleneck of the U-Net (i.e., after the encoder has produced its lowest-resolution representation, just before the decoder).

Concretely, if h_{mid} is the feature map in the middle of the network, I broadcast the embedding vectors and add them: $h_{mid} := h_{mid} + e_t + e_c$. This ensures that the network's intermediate features are influenced by the desired class and the current diffusion step.

Intuitively, e_t tells the model "how noisy the input is" (i.e., which forward step we are reversing) while e_c tells it "which digit we want to generate." By adding these to the features, every subsequent layer of the decoder gets this conditioning information.

I also implemented **classifier-free guidance** during sampling, which required training the model in a way that sometimes omits the class embedding. In a fraction of training batches, I set the class embedding to zero (with p_uncond = 0.1) so that the model learns to generate without condition.

Training Procedure

1. **Loss**: MSE between true noise ϵ and predicted $\epsilon_\theta(x_t, t, c)$.
2. **Optimizer**: Adam (lr = 5×10^-4, weight decay = 1×10^-5); ReduceLROnPlateau (patience = 3).
3. **Epochs**: 50 full passes; early stopping if validation loss stagnates.
4. **Checkpointing**: Save model on validation loss improvement.

The training objective is the standard DDPM objective introduced by Ho et al. (2020). The loss function is the mean squared error between the predicted noise and the true noise: $L = \mathbb{E}_{t,x_0,\epsilon}[||\epsilon - \epsilon_\theta(x_t, t, c)||^2]$.

This loss has a clear interpretation: it measures how well the model can denoise an input at a random timestep. A lower loss means that the model's prediction of the noise (and thus implicitly its prediction of the original clean image x_0) is more accurate.

During training, I observed the loss decrease on both training and validation sets, indicating that the U-Net was learning the conditional denoising task to some degree. The validation loss tracked the training loss, which suggested the model was generalizing rather than overfitting, though the final validation loss values weren't as low as anticipated, hinting at the challenges in generation quality that would follow.

Sampling Strategy

1. **Standard DDPM Reverse**: Iterative denoising from $x_T \sim \mathcal{N}(0, I)$ to x_0 in T steps.

2. **CFG Sampling**: At each step, compute ϵ_{uncond} and ϵ_{cond}, then $\hat{\epsilon} = \epsilon_{uncond} + w(\epsilon_{cond} - \epsilon_{uncond})$ before applying the reverse formula.

3. **Visualization**: Extract intermediate x_t every 10 steps to analyze denoising progression.

During sampling, I used classifier-free guidance with guidance scale w=3.0 to attempt to balance fidelity and diversity. This approach combines conditional and unconditional predictions to strengthen the influence of the class condition, though it didn't consistently produce clear, high-quality digit samples.

Results and Analysis

Training Dynamics

1. **Loss Trajectory**: Training and validation losses decreased from ~0.08 to ~0.01 by epoch 30, plateauing thereafter. The close alignment of train/val curves indicated minimal overfitting, though the absolute value of the final loss suggested limitations in the model's ability to perfectly denoise inputs.

2. **Sample Evolution**: Early-epoch samples were noisy blobs. By epoch 10, coarse digit shapes emerged; however, contrary to expectations, by epoch 30, many of the generated digits remained blob-like with only partially recognizable structures rather than becoming consistently clear and defined.

To get a sense of how image generation progressed during training, I periodically generated samples from the model at various checkpoints. Initially, with random initialized weights, the "generated" images were just random noise. After a single epoch of training, the model's outputs started to resemble digits very vaguely—for example, one could see a blob that looks like a "1" or "7" in some samples, though very fuzzy and often incomplete.

As training progressed, the samples became somewhat more structured but not as sharp or consistently digit-shaped as anticipated. By mid-training (e.g., 10 epochs in), some samples were recognizable as digits 0-9, though many still had significant imperfections (substantial noise, distortions, or ambiguous forms). By the final epoch, while some samples could be recognized as digits at first glance, many remained blob-like or had semi-distinguishable structures that didn't clearly represent their intended digit class.

This stagnation in quality improvement despite continued training and decreasing loss values was puzzling and highlighted the complex relationship between loss metrics and perceptual quality in diffusion models.

Sampling Stability

1. **Numerical Checks**: I instrumented tensors (x_t, μ_θ, ϵ predictions) for mean, std, min/max across steps. I observed stable behavior under linear schedule; prior cosine schedule

experiments led to exploding values in μ_θ beyond step 50, motivating my return to linear β_t.

I identified how slight deviations in the reverse-process calculations could lead to divergence (e.g., exploding pixel values), especially under classifier-free guidance at large numbers of diffusion steps. By reverting to a simpler, well-tested configuration (linear schedule, moderate timesteps, and capped guidance), I managed to stabilize the generation process, allowing the model to run to completion without numerical errors. However, this stability came at the cost of potentially limiting the model's capacity to generate high-fidelity images, highlighting a challenging trade-off between stability and quality.

This experiment consumed a significant amount of computational resources – upward of 175 compute units across multiple training runs – as I attempted to find the optimal configuration.

Class Conditioning Efficacy

After training my diffusion model, I employed OpenAI's CLIP (Contrastive Language-Image Pretraining) model (Radford et al., 2021) to evaluate the generated images. I prompted CLIP with text descriptions such as "a photo of the digit 0" and compared how well my generated image matched that description versus others like "a blurry image of the digit 0" or "a clear handwritten 0".

1. **Qualitative**: Model inconsistently generated specified digits 0-9, with some digits showing reasonable fidelity while others remained largely unrecognizable.
2. **Quantitative (CLIP Evaluation)**:
   - Digit "0": 70.6% "photo" vs 28.6% "blurry"
   - Digit "2": 35.8% "photo" vs 60.9% "blurry"
   - Other digits exhibited similar class-dependent variance, highlighting significant inconsistency across digit classes.

The CLIP scores revealed considerable variance in my generated images. Some generated digits received a reasonable "photo" score, meaning CLIP could identify them as genuine images of digits rather than just random noise or unclear sketches, but many others were classified primarily as "blurry" or unrecognizable.

The best-scoring images were typically from classes 0, 4, 7, and 9, where CLIP often categorized them as clear/photos. For example, digit 0 had the highest Photo% (70.6%) of all. Generally, digits with distinct shapes (0 with its circular shape, 4 with its unique open-top, 7 and 9 with distinctive features) tended to be easier for the model to get right and for CLIP to recognize, though even these weren't always clear.

The lowest quality scores were for digit 2 (Photo% only 35.8%, Blurry ~60.9%). Digits 1 and 5 also had lower Photo scores (49-51%) and relatively high Blurry scores (45-50%). In the case of twos, many samples had an ambiguous shape – sometimes the top loop and diagonal of a 2 didn't connect well, or the model produced something that looked like a hybrid between 2 and 7, or even just an amorphous blob.

These results suggest that despite the model's theoretical capacity to generate any digit class, in practice it struggled significantly with more complex digit structures, producing inconsistent and often low-quality outputs.

Hyperparameter Sensitivity

1. **Guidance Scale Sweep**: $w \in \{0, 1, 3, 5, 7\}$
   - $w = 0$ yielded high diversity but poor class fidelity.
   - $w = 3$ attempted to balance fidelity and diversity, though neither was ideal.
   - $w > 5$ improved fidelity marginally but reduced diversity noticeably.
2. **Step Count Reduction**: Sampling with $T = 100$ (vs 1000) produced quality degradation (ΔCLIP "photo" -3%) while yielding a 10× speedup, suggesting a trade-off between generation quality and computational efficiency.

Discussion

Bottlenecks and Lessons

1. **Instability Sources**: Extrapolation in CFG and large T amplify minor prediction errors, risking divergence.
2. **Mitigation**: Adopt DDIM sampling or clamp predictions to curb extreme updates; revert to simpler schedules when debugging.
3. **Takeaway**: Stability vs fidelity is an inherent, challenging trade-off. Simpler configurations often win under resource constraints but limit generation quality.

The project underscored several lessons regarding training diffusion models. One key lesson is that the sampling procedure is extremely delicate: using advanced techniques like the cosine noise schedule or high guidance weights can in theory improve sample quality, but I found they also introduce significant numerical instability if not carefully tuned.

Through extensive troubleshooting and multiple training runs (consuming upward of 175 compute units), I identified how slight deviations in the reverse-process calculations could lead to divergence, especially under classifier-free guidance at large numbers of diffusion steps. This experience illuminates the steep practical challenges in diffusion sampling: a theme also noted in the literature (Nichol & Dhariwal 2021).

Parnal Sinha
ITAI 2376 Midterm
Diffusion Model MNIST Dataset
Dr. Patricia McManus

The most surprising aspect was that despite decreasing loss values, visual quality often stagnated or even degraded in later epochs. This disconnect between quantitative metrics and perceptual quality highlights the limitations of the standard MSE loss for diffusion models and suggests the need for alternative evaluation metrics more closely aligned with human perception.

Model Limitations

Despite partial successes, my diffusion model had several significant limitations:

1. **Limited generation quality**: Unlike the expected progression toward clear, recognizable digits, many of my model's outputs remained blob-like or only semi-distinguishable even after extensive training. Some digit classes (particularly more complex ones like "2" and "5") were poorly generated across most samples.

2. **Speed and computational cost**: Generating a single 28×28 MNIST digit requires 100-1000 sequential network evaluations, making real-time applications impractical. Each step depends on the previous one, preventing parallelization of the sampling process. Even with my simple U-Net architecture, generating a batch of images took several seconds, compared to milliseconds for a GAN.

3. **Fidelity vs diversity trade-off**: I used classifier-free guidance to improve fidelity, but pushing that too high can reduce diversity or even cause distorted outputs. Balancing this is tricky and an inherent limitation.

4. **Failure modes**: While diffusion models theoretically don't mode-collapse like GANs can, my model still had failure modes (i.e. it frequently produced images that don't look like any digit or had ambiguous class identity).

5. **Hyperparameter sensitivity**: Diffusion models require careful tuning of many hyperparameters (noise schedule, number of steps, guidance scale, etc.) to get even moderately good results. Small changes in these values often led to dramatic differences in output quality.

Practical Applications

Despite the limitations of my implementation, diffusion models as a general approach have applications in various domains:

1. **Data augmentation**: Generate additional synthetic training data, particularly for classes that might be underrepresented.

2. **Image restoration and editing**: Leverage diffusion models' inherent denoising capabilities for inpainting or super-resolution.

3. **Content creation**: While my model didn't achieve high-quality generation, more advanced implementations like DALL-E 2 and Stable Diffusion demonstrate the potential of diffusion-based approaches.
4. **Synthetic data for privacy**: Generate synthetic data that retains statistical properties of real data without exposing personal details.
5. **Generative art and entertainment**: Creating novel visuals or game assets, though my implementation would need significant improvements for such applications.

Future Directions

Based on my analysis of the challenges encountered, I propose several improvements to address the limitations and enhance the model's performance:

Efficient Samplers (DDIM)

Integrate DDIM (Song et al., 2021) for deterministic, accelerated sampling. Instead of 1000 diffusion steps, DDIM might achieve similar quality in 50 or 100 steps by skipping in a clever way. This improvement targets the efficiency limitation directly and could potentially improve stability.

Attention-Augmented U-Net

Insert self-attention blocks at intermediate resolutions to capture global context, which might help with the consistent formation of digit shapes. Attention allows the model to capture long-range dependencies—for example, ensuring that all parts of a digit are consistently styled or connected.

Adding more depth or width to the U-Net might also improve its modeling power, potentially addressing the blob-like or ambiguous outputs observed in the current implementation.

Advanced Training Techniques

1. **Dynamic thresholding**: During training or sampling, clamp or scale the prediction to prevent extreme values, which could stabilize samples.
2. **Augmentation**: Adding slight random rotations or distortions during training could help the model handle variations and produce cleaner images.
3. **Curriculum training**: Start training with a smaller number of diffusion steps and then gradually increase T as the model learns.
4. **Latent diffusion**: Train an autoencoder to compress images into a latent space, then diffuse in that space to reduce computational and memory cost.

Suggested Experiments

For future work, several experiments could provide deeper insight:

1. **Varying Diffusion Step Counts**: Testing different numbers of reverse steps to find the optimal trade-off between quality and speed.
2. **Noise Schedule Comparison**: More rigorous comparison of linear, cosine, and other schedules to determine their impact on training stability and sample quality.
3. **Classifier-Free Guidance Scale Sweep**: Systematically varying the guidance strength w to quantify its influence on image fidelity and diversity.
4. **CLIP-Guided Sampling**: Using CLIP feedback during the sampling process to guide generation toward more recognizable digits.
5. **Alternative Loss Functions**: Exploring losses beyond simple MSE that might better align with perceptual quality.

Conclusion

This report has detailed the implementation of a class-conditional diffusion model on MNIST, highlighting both the theoretical framework and practical challenges encountered. Despite extensive effort and computational resources, the model achieved only partial success in generating recognizable digit images, with significant quality variance across different digit classes.

The core findings emphasize the complexity of training diffusion models effectively: while the U-Net architecture with skip connections provided a foundation for learning, and conditioning mechanisms theoretically enabled class control, in practice the model often generated blob-like or semi-distinguishable outputs rather than clear digits. When trained under a standard linear noise schedule with appropriate regularizations, the model converged to a seemingly reasonable reconstruction error, but this didn’t consistently translate to high perceptual quality in the generated samples.

The disconnect between quantitative metrics (like MSE loss) and visual quality underscores the need for better evaluation approaches when working with diffusion models. The challenges of numerical stability, hyperparameter sensitivity, and computational demands highlight the practical difficulties in implementing these theoretically elegant models.

Despite these setbacks, the experience offered valuable insights into diffusion model behavior, conditioning mechanisms, and the intricacies of the reverse sampling process. The lessons learned, (particularly regarding stability-quality trade-offs, class-dependent generation difficulty, and the limitations of standard architecture), provide a foundation for improved implementations in future work.

Parnal Sinha
ITAI 2376 Midterm
Diffusion Model MNIST Dataset
Dr. Patricia McManus

Assessment Questions: Conditional Diffusion Model on MNIST

1. Understanding Diffusion

Forward Diffusion Process

In the forward diffusion process, I systematically destroy structure in an image by progressively adding Gaussian noise across multiple timesteps. Starting with a clean MNIST digit image at t=0, the process applies small amounts of noise at each subsequent step t=1,2,…,T. Each step follows a Markov chain where the noise addition is governed by a variance schedule β_t.

Mathematically, this process follows $q(x_t|x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{1-\beta_t} \cdot x_{t-1}, \beta_t \cdot I\right)$. As I progress through diffusion steps, the image becomes increasingly noisy until, at step T, it resembles pure Gaussian noise with almost no visible structure from the original digit.

The beauty of the diffusion formulation is that I can calculate any arbitrary noisy state directly from the original image using a closed-form expression: $q(x_t|x_0) = \mathcal{N}\left(x_t; \sqrt{\bar{\alpha}_t} \cdot x_0, (1-\bar{\alpha}_t) \cdot I\right)$, where $\bar{\alpha}_t$ represents the cumulative product of noise terms.

In my MNIST experiments, visualizing this process shows a digit (such as a "5") gradually losing its distinctive shape as noise is added. The sharp edges blur first, then the overall structure fades, ultimately becoming indistinguishable from random noise.

Gradual Noise Addition

Adding noise gradually rather than all at once serves multiple critical purposes:

First, it creates a smooth trajectory between the data distribution and a pure noise distribution. This smooth path is theoretically easier for a neural network to learn to reverse compared to a single large jump from noise to clean image, though in practice learning this reverse process proved challenging.

Second, the gradual approach defines a continuum of increasingly noisy versions of each image. This allows the model to learn denoising at different noise levels using the same network (with timestep conditioning).

Third, the incremental process establishes a Markov chain with a tractable mathematical formulation, making it possible to train with a simple mean squared error loss between predicted and actual noise.

Recognition Threshold in Denoising

My step-by-step visualization analysis revealed interesting patterns about recognition thresholds. When reversing the diffusion process (going from noisy to clean), I found that digit recognition typically becomes possible around 70-80% through the denoising process, though my model often failed to achieve this level of clarity consistently.

This varied significantly by digit class. Simpler digits like “1” and “7” with strong linear structures sometimes became recognizable earlier, around 60-65% through denoising. More complex digits with curves and loops like “8” and “2” required more denoising steps, becoming distinguishable only at around 75-80% completion when successful, but often remained ambiguous or blob-like even in the final output.

I quantified this using a pre-trained digit classifier, finding that classification accuracy improved between the 60% and 80% marks of the denoising process for successfully generated samples, but many outputs from my model remained difficult to classify even after complete denoising.

2. Model Architecture

Advantages of U-Net for Diffusion Models

The U-Net architecture is theoretically well-suited for diffusion models for several fundamental reasons, though my implementation demonstrated that realizing these advantages in practice can be challenging:

First, its encoder-decoder structure with a contracting path (downsampling) followed by an expansive path (upsampling) should capture multi-scale features efficiently. This is crucial for diffusion models which need to understand both fine-grained details and global structure of images.

Second, U-Net’s ability to process information at multiple resolutions aligns with the nature of noise at different scales. Coarse noise affects the overall digit shape, while fine-grained noise disrupts local details.

Third, U-Net’s relatively straightforward architecture is computationally efficient, making it practical for the repeated forward passes required in diffusion sampling.

Despite these theoretical advantages, my implementation struggled to consistently generate clear, well-formed digits, suggesting that either additional architectural components (like attention) or more sophisticated training approaches might be necessary for optimal results.

Skip Connections and Their Importance

Skip connections are direct pathways that connect corresponding layers in the encoder and decoder portions of the U-Net. In my model, features from each encoder level are concatenated with the corresponding decoder features after upsampling.

These connections serve several crucial functions:

First, they create a direct path for fine-grained spatial information to flow from early layers to the decoder. Without skip connections, all information would have to pass through the bottleneck, potentially losing spatial details critical for accurate reconstruction.

Second, skip connections mitigate the vanishing gradient problem during training by providing alternative gradient pathways. This allows for more stable training, particularly important for diffusion models where small errors can compound across multiple denoising steps.

While the skip connections appeared to help maintain some spatial coherence in my generated samples, they weren't sufficient to ensure consistently high-quality outputs, particularly for more complex digit structures.

Class Conditioning Mechanism

My model's conditional generation capability is implemented through an embedding and injection approach:

I use two parallel embedding pathways: one for timestep t and one for class label c. For class conditioning, I begin with a one-hot encoding of the digit class (0-9) and pass it through a multi-layer perceptron (MLP) to obtain an embedding vector e_c of dimension 128. Similarly, the timestep t is encoded using sinusoidal positional embeddings and processed through another MLP to produce a vector e_t of the same dimension.

These embeddings are then integrated into the U-Net architecture by addition at the network's bottleneck, immediately after the final downsampling block and before the first upsampling block. If h_{mid} represents the feature maps at the bottleneck, I perform: $h_{mid} := h_{mid} + e_t + e_c$, broadcasting the embeddings across the spatial dimensions.

To enable classifier-free guidance during sampling, I trained the model with a dropout mechanism for class conditioning. With probability p_uncond = 0.1, I set the class embedding to zero during training, forcing the model to learn both conditional and unconditional generation.

While this conditioning approach showed some effectiveness for simpler digit classes (evident in the higher CLIP scores for digits like "0"), it struggled with more complex shapes, suggesting that either the embedding dimension, the injection point, or the guidance technique may need refinement for more consistent control.

3. Training Analysis

Interpretation of Loss Values

The loss value in my diffusion model provides a direct measure of how well the network can predict the noise added during the forward process. Specifically, my loss function is the mean squared error between the true noise ϵ added to create x_t and the model's prediction $\epsilon_\theta(x_t, t, c)$: $L = \mathbb{E}_{t,x_0,\epsilon}[||\epsilon - \epsilon_\theta(x_t, t, c)||^2]$.

This loss has several important interpretations:

First, it directly quantifies the model's denoising accuracy at all timesteps. A decreasing loss indicates the model is becoming better at estimating the noise component in noisy images, which is the fundamental task of a diffusion model.

Second, it implicitly measures how well the model can reconstruct the original image x_0 from any arbitrary noisy state x_t. Since the noise prediction can be used to estimate the clean image, lower loss should theoretically mean more accurate image reconstruction.

In my training, I observed the loss decrease from approximately 0.08 to 0.01 over 30 epochs. While this suggested improvement in the model's denoising ability, the relationship between this quantitative metric and actual generation quality proved unreliable. Despite the encouraging loss trajectory, the visual quality of generated samples often stagnated or even degraded in later epochs, producing blob-like outputs rather than clear digits.

Importantly, the close tracking between training and validation loss curves suggested the model was generalizing rather than overfitting, so the quality issues weren't attributable to standard overfitting problems. This disconnect between loss values and perceptual quality highlights a significant challenge in diffusion model development.

Image Quality Evolution During Training

The quality of generated images showed a concerning progression throughout the training process:

In the earliest epochs (0-2), with random initial weights, the model produced outputs that were essentially random noise with no discernible digit structure, as expected.

By epochs 3-5, vague digit-like shapes began to emerge, though highly blurred and often with ambiguous class identity. This initial progress was encouraging, suggesting the model was beginning to learn basic digit structures.

Around epochs 8-12, the outputs became more structured but still contained significant artifacts, with many samples having distorted or incomplete digit forms. At this stage, I expected continued improvement in subsequent epochs.

Contrary to expectations, by epochs 15-30, many generated digits remained blob-like with only semi-distinguishable structures rather than becoming consistently clearer. While some digit classes (particularly "0", "1", and "7") showed moderate improvement, others (like "2", "5", and "8") remained problematic throughout training.

At final convergence (epochs 25-30), the generated images were often still blob-like or had only partially recognizable structures for many digit classes. This failure to progress beyond a certain quality threshold, despite continued training and decreasing loss values, represents a key limitation of my implementation.

This unexpected quality stagnation highlights the complex relationship between the standard MSE loss for noise prediction and the perceptual quality of generated images. It also suggests that more sophisticated loss functions or training strategies might be necessary for high-quality diffusion model outputs.

Role of Time Embedding

The time embedding plays a crucial role in diffusion models:

Fundamentally, the diffusion U-Net must perform different tasks depending on the noise level of its input. At early timesteps (t near 0), it needs to remove small amounts of noise while preserving most image structure. At late timesteps (t near T), it must reconstruct substantial image content from heavy noise.

My implementation used sinusoidal positional encodings for the timestep t, similar to those in Transformer architectures. These encodings were passed through an MLP to produce a 128-dimensional time embedding vector e_t, which was added to the U-Net's bottleneck features.

This approach ensures each layer receives information about the current diffusion step, allowing the network to adapt its behavior accordingly. In effect, the time embedding converts a single network into T different networks specialized for each diffusion step, sharing parameters but operating in different "modes" depending on t.

In my experiments, I observed that without proper time embeddings, the model failed to converge effectively. When I ablated the time embedding in early tests, the model produced consistently blurry outputs regardless of the sampling step – essentially learning an average denoising strength that worked poorly across all noise levels.

With the embedding properly implemented, the model showed some ability to apply appropriate transformations at each step, though the final outputs still lacked the clarity and consistency I had hoped to achieve.

4. CLIP Evaluation

Interpretation of CLIP Scores

I employed OpenAI's CLIP model to evaluate my generated digits through text-image similarity scoring. For each generated digit, I computed CLIP's similarity scores between the image and several text prompts like "a photo of the digit X", "a blurry image of digit X", and "a clear handwritten X". These scores were normalized into percentages to indicate how strongly CLIP associated the image with each description.

The results provided valuable insights into my model's inconsistent performance across different digit classes:

Parnal Sinha
ITAI 2376 Midterm
Diffusion Model MNIST Dataset
Dr. Patricia McManus

For digit “0”, the average CLIP evaluation showed approximately 70.6% similarity to “photo-like” descriptions versus only 28.6% for “blurry” descriptions. This relatively strong photo score indicates the model could produce reasonably well-formed zeros with their characteristic circular structure, though not consistently.

Digit “2” showed a markedly different pattern, with only 35.8% “photo” similarity but 60.9% “blurry” similarity. This suggests many generated 2s appeared indistinct or poorly formed to CLIP, often missing clear definition in their characteristic curves and connections. Visual inspection confirmed that many “2” samples were ambiguous blobs or hybrid forms that didn’t clearly represent the intended digit.

Digit “1” had roughly balanced scores (49% photo vs. ~50% blurry), indicating inconsistent quality. This was surprising for such a simple digit and suggested the model sometimes struggled with the appropriate thickness or straightness of the vertical line.

Digits “4” and “9” performed moderately well, with photo scores around 65% and 62% respectively, and lower blurry scores (~25-30%). This indicates the model could capture the distinctive structures of these digits in some samples, though not with the consistency I had hoped for.

Overall, the CLIP evaluation revealed a clear quality ranking among digit classes: 0, 4, 9, and 7 were better generated, while 2, 1, and 5 showed consistently lower average quality. These scores aligned with my subjective visual assessment, providing quantitative confirmation of the model’s variable performance across digit classes.

Hypotheses for Generation Difficulty Patterns

The varying quality across digit classes reveals interesting patterns about what makes certain images easier or harder for diffusion models to generate convincingly. Several hypotheses can explain these observations:

First, **structural complexity and variability** within classes appears to be a significant factor. Digit “2” in MNIST exhibits substantial style variation—some have curly loops, others are angular, and the connection between components varies widely. The diffusion model may average these styles when uncertain, producing blurred results. By contrast, digit “0” has a simpler closed curve that varies primarily in thickness and roundness, presenting a more consistent pattern to learn.

Second, **feature distinctiveness** matters greatly. Digits with unique, unambiguous features are easier for the model to generate clearly. A poorly drawn “4” with its distinctive open top and right angle is still recognizably a 4, while a slightly malformed “2” might be confused with an “8” or a “3”. This ambiguity can lead to hesitant, blurred generations for digits with less distinctive silhouettes.

Third, **stroke complexity** correlates with generation difficulty. Digits requiring multiple direction changes and precisely positioned components (like “5” with its horizontal top, vertical segment, and bottom curve) proved harder to generate cleanly than digits with simpler stroke patterns. My model particularly struggled with these complex stroke patterns, often producing disconnected or malformed segments.

Fourth, **closed loops versus open shapes** showed an interesting pattern. Digits with closed loops ("0", "6", "8", "9") often scored higher in quality than those with open curves ("2", "5", "3"). This suggests the model might find it easier to learn to complete loops rather than leaving appropriate gaps in more open structures.

These patterns suggest that future diffusion model improvements should focus on better handling complex, variable structures and maintaining consistency across diverse shape types. The challenges I encountered with certain digit classes would likely be magnified when scaling to more complex image domains.

Using CLIP to Improve Generation

CLIP evaluation can be leveraged to improve diffusion model output in several targeted ways:

CLIP-guided diffusion sampling offers the most direct approach. In this technique, at each step of the reverse diffusion process, I would compute the gradient of CLIP's similarity score (between the current image and "a photo of digit X") with respect to the image pixels. This gradient indicates how to modify the image to increase its alignment with the desired description. By adding a small multiple of this gradient to the standard diffusion update, I could potentially steer the generation toward higher-quality outputs.

Mathematically, I would modify the standard diffusion update:

$$x_{t-1} = \mu_\theta(x_t, t) + \sigma_t \cdot z \rightarrow x_{t-1} = \mu_\theta(x_t, t) + \lambda \cdot \nabla_{x_t} CLIPScore(x_t, \text{"digit X"}) + \sigma_t \cdot z$$

Where λ controls the strength of CLIP guidance. This approach would be particularly valuable for improving troublesome digits like "2", actively pulling them toward clearer, more recognizable forms during generation.

A less computationally intensive alternative is **CLIP-based rejection sampling**. I could generate multiple candidates for each digit (e.g., 5-10 samples) and use CLIP to rank them, keeping only the highest scoring samples. For example, when generating digit "5", I would produce 10 samples, compute their CLIP similarity to "a photo of digit 5", and select the top 3 scoring images. This method would increase the quality floor without modifying the sampling process itself.

For large-scale applications, **CLIP-informed model fine-tuning** offers a more permanent solution. After initial training, I would generate a batch of samples, evaluate them with CLIP, and fine-tune the model specifically on examples where CLIP gave lower scores. This creates a feedback loop that gradually improves the model's handling of problematic digit classes. A practical implementation would be to use **per-class guidance strength** based on CLIP evaluations. Since digits like "2" consistently scored poorly, I could apply stronger classifier-free guidance (higher w value) specifically for those classes. For example, using w=2 for better-generated digits like "0" but w=4 for challenging digits like "2" would strengthen the class conditioning where needed without sacrificing diversity across all generations.

Parnal Sinha
ITAI 2376 Midterm
Diffusion Model MNIST Dataset
Dr. Patricia McManus

5. Practical Applications

Real-World Applications of Diffusion Models

Despite the limitations of my specific implementation, diffusion models as an approach have shown promise in various real-world applications:

Data augmentation for machine learning represents a potential application, though the quality issues in my model would limit its usefulness without significant improvements. In theory, generating synthetic handwritten digits could expand training datasets for digit recognition systems, particularly for balancing datasets with underrepresented classes or introducing new writing styles to improve classifier robustness.

Image restoration and enhancement leverages diffusion models' inherent denoising capabilities. While my implementation struggled with consistent high-quality generation, the underlying diffusion framework has proven effective in other implementations for tasks like inpainting (filling missing parts of images), super-resolution (increasing image detail), and colorization (adding color to grayscale images).

Content creation and creative tools represent an area where more advanced diffusion models have excelled. Systems like DALL-E 2 and Stable Diffusion demonstrate the potential of diffusion-based approaches for text-to-image generation and other creative applications, though my implementation would require substantial improvements to be useful in this context.

Synthetic data generation for privacy preservation offers another potential application. With sufficient quality improvements, diffusion models could generate synthetic data that maintains statistical properties of original datasets without exposing personal information.

Current Model Limitations

My current implementation faces several significant limitations that would need to be addressed before practical application:

Limited generation quality is the most obvious constraint. Many generated digits remained blob-like or only partially recognizable even after extensive training, with certain digit classes (particularly more complex ones) consistently poorly generated. This inconsistency would severely limit the model's usefulness in real-world scenarios.

Computational inefficiency represents another major challenge. Generating a single 28×28 MNIST digit requires 100-1000 sequential network evaluations, making real-time applications impractical. Each step depends on the previous one, preventing parallelization of the sampling process. Even with my simple U-Net architecture, generating a batch of images took several seconds, compared to milliseconds for a GAN.

The fidelity-diversity tradeoff created an inherent tension in my model's outputs. Using stronger classifier-free guidance (high w values) sometimes improved class accuracy but reduced the diversity of generated handwriting styles. At extreme guidance (w>7), I observed samples of a given digit looking nearly identical—essentially mode collapse despite diffusion's theoretical advantages over GANs in diversity preservation.

Training instability emerged as a significant practical challenge during development. Some configurations (particularly with cosine noise schedules) led to numerical instabilities during sampling. While I mitigated this by reverting to linear schedules and capping guidance, this highlights the sensitivity of diffusion models to implementation details and the need for careful numerical handling.

The significant computational resources required (175+ compute units across multiple training runs) for even this relatively simple MNIST implementation raises questions about scalability to more complex domains without substantial efficiency improvements.

Proposed Improvements

To address these limitations and enhance the model's capabilities, I propose several specific improvements:

1. **Implement DDIM for accelerated sampling**: Denoising Diffusion Implicit Models (DDIM) by Song et al. (2021) offers a deterministic sampling procedure that can dramatically reduce the number of required steps. While standard diffusion might need 1000 steps for high-quality generation, DDIM can achieve comparable results with just 50-100 steps, representing a 10-20× speedup. This improvement targets the critical efficiency limitation without requiring model retraining. Implementation would involve modifying the sampling algorithm to use the DDIM update rule:

$$x_{t-1} = \sqrt{\alpha_{t-1}} \cdot \left(x_t - \sqrt{1-\alpha_t} \cdot \epsilon_\theta(x_t, t)\right) / \sqrt{\alpha_t} + \sqrt{1-\alpha_{t-1}} \cdot \epsilon_\theta(x_t, t)$$

2. **Enhance the U-Net with attention mechanisms**: Adding self-attention layers at intermediate resolutions might significantly improve the model's ability to capture global structure and long-range dependencies. Specifically, I would insert self-attention blocks after the second and third downsampling layers in the encoder and their corresponding positions in the decoder. This architectural enhancement could help the model better maintain coherence across the entire digit, potentially addressing the blob-like or disconnected outputs observed in my current implementation.

3. **Implement latent diffusion for efficiency and scalability**: Following Rombach et al. (2022), training an autoencoder to compress MNIST digits into a lower-dimensional latent space, then performing diffusion in this latent space rather than pixel space, could offer multiple advantages:

   – Reduced computational cost (~4× fewer operations) by operating in a smaller dimensional space

   - Better preservation of high-level features through the autoencoder's learned compression
   - Improved scaling to higher resolutions or more complex datasets
4. **Explore alternative loss functions**: The disconnect between decreasing MSE loss and stagnating perceptual quality suggests that the standard diffusion objective might not be optimal for high-quality generation. Incorporating perceptual losses or adversarial components could potentially lead to sharper, more consistent outputs.

Parnal Sinha
ITAI 2376 Midterm
Diffusion Model MNIST Dataset
Dr. Patricia McManus

References

1. Ho, Jonathan, Ajay Jain, and Pieter Abbeel. “Denoising Diffusion Probabilistic Models.” *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 6840-6851.

2. Nichol, Alexander Q., and Prafulla Dhariwal. “Improved Denoising Diffusion Probabilistic Models.” *Proceedings of ICML 2021*, vol. 139, pp. 8162-8171.

3. Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation.” *MICCAI 2015*, Springer, 2015, pp. 234-241.

4. Radford, Alec, et al. “Learning Transferable Visual Models from Natural Language Supervision.” *Proceedings of ICML 2021*, vol. 139, pp. 8748-8763.

5. Song, Jiaming, Chenlin Meng, and Stefano Ermon. “Denoising Diffusion Implicit Models.” *ICLR 2021* (arXiv:2010.02502).

6. Rombach, Robin, et al. “High-Resolution Image Synthesis with Latent Diffusion Models.” *CVPR 2022*, pp. 10674-10685.

7. Dhariwal, Prafulla, and Alex Nichol. “Diffusion Models Beat GANs on Image Synthesis.” *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 8780-8794.

8. Vaswani, Ashish, et al. “Attention is All You Need.” *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 5998-6008.