Parnal Sinha
ITAI 2376 Midterm
Diffusion Model Supplemental Report
Dr. Patricia McManus

Diffusion Model Implementation and Troubleshooting Report

Introduction

This project began with ambitious intentions: training a conditional diffusion model on the CelebA dataset, aiming to generate high-resolution, attribute-conditioned facial images. CelebA, with its complex feature space and attribute vectors, seemed like the perfect playground to stretch the limits of diffusion models.

However, after burning through nearly 175 compute units, suffering multiple hallucinated outputs (later identified as numerical instability), training collapses, and a progressive erosion of sanity, the project scope was adapted – first to CIFAR-10, and ultimately, to MNIST.

In a way, this descent mirrors the trajectory of the project itself: big dreams slowly constrained by the realities of limited compute power, unstable training pipelines, and the relentless clock of assignment deadlines. Despite these setbacks, the core mission remained unchanged: to build, train, and analyze a class-conditional diffusion model using a UNet-based architecture, and to observe its capacity for reconstructing structured image data under a diffusion framework.

What follows is not merely a technical methodology, but a documentation of the compromises, adaptations, and lessons learned in the process of dragging a working model into existence under less-than-ideal conditions. The Methodology section below outlines the specific training pipeline, dataset handling, architectural choices, and sampling strategies employed for the various datasets prior to choosing MNIST dataset for the purpose of documentation and learning value.

Methodology

The core objective was to implement and train a conditional Denoising Diffusion Probabilistic Model (DDPM) capable of generating images based on class labels. The methodology evolved significantly as challenges arose, adapting across datasets (CelebA, CIFAR-10, Fashion-MNIST) and incorporating various techniques based on standard practices and debugging attempts.

1. *Diffusion Models in Context*

- **Forward Process (Noising)**: The foundation relies on gradually adding Gaussian noise to clean images (x_0) over a predefined number of discrete timesteps (T). The state at timestep t (x_t) is sampled from a Gaussian distribution conditioned on x_{t-1}, with variance controlled by a noise schedule β_t. This can be expressed efficiently by sampling x_t directly from x_0:

  $x_t = \sqrt{\bar{\alpha}_t} \cdot x_0 + \sqrt{1 - \bar{\alpha}_t} \cdot \epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$, $\alpha_t = 1 - \beta_t$, and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$.

The add_noise function implemented this process. Visualizations confirmed this forward process worked correctly, transforming clean images into noise as expected.

- **Reverse Process (Denoising/Sampling)**: The goal is to learn the reverse process $p_\theta(x_{t-1}|x_t, c)$, conditioned on the class label c. DDPMs approximate this by training a model (typically a U-Net) $\epsilon_\theta(x_t, t, c)$ to predict the noise ϵ added at timestep t. Given the predicted noise ϵ_θ, the previous state x_{t-1} can be estimated using the formula derived from Bayes' theorem:

  $$\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t, c)\right)$$

  $x_{t-1} = \mu_\theta(x_t, t) + \sigma_t z$, where $z \sim \mathcal{N}(0, I)$ and $\sigma_t^2 = \beta_t$ (for DDPM).

  Sampling starts with pure noise $x_T \sim \mathcal{N}(0, I)$ and iteratively applies this reverse step from $t = T - 1$ down to $t = 0$. The remove_noise function (template version) and the CFG-enabled generation functions implemented variations of this reverse step.

2. *Noise Schedules*

- **Linear Schedule (Template Default)**: Initially, a linear schedule was used as per the template, where β_t increases linearly from β_{start} (e.g., 0.0001) to β_{end} (e.g., 0.02) over $T = 100$ steps. This is simple but sometimes considered less optimal than other schedules.
- **Cosine Schedule (Optimization Attempt)**: Based on common practices suggesting improved performance, particularly for complex datasets and longer training, a cosine schedule was implemented during the CIFAR-10 and Fashion-MNIST attempts. This schedule, defined by $\bar{\alpha}_t = \frac{f(t)}{f(0)}$ where $f(t) = \cos\left(\frac{t/T+s}{1+s} \cdot \frac{\pi}{2}\right)^2$, provides a slower noise addition rate initially and accelerates later compared to the linear schedule. The implementation used $T = 1000$ (for CIFAR-10) and later $T = 200$ (for Fashion-MNIST).

3. *Model Architecture (U-Net)*

- **Core Structure**: A U-Net architecture was employed, consistent with standard diffusion model practices. This architecture is well-suited due to its ability to process information at multiple spatial resolutions through downsampling (encoder) and upsampling (decoder) paths, while preserving high-resolution details via skip connections between corresponding encoder and decoder layers.
- **Building Blocks**: The template provided basic building blocks:
  - **GELUConvBlock**: Contained Conv2d, GroupNorm, and GELU activation. Group size handling was adjusted for compatibility.

  - **RearrangePoolBlock**: Used einops.Rearrange for 2x spatial downsampling by rearranging pixels into channels, followed by a GELUConvBlock.
  - **DownBlock (Template)**: Sequentially combined two GELUConvBlocks and a RearrangePoolBlock. *Initial analysis revealed a flaw: this structure didn't allow easy access to features before pooling for skip connections.*
  - **UpBlock (Template)**: Implemented ConvTranspose2d for upsampling, concatenated the result with an incoming skip connection, and processed through two GELUConvBlocks.
- **Architectural Variations**:
  - *Template MNIST*: down_chs=(32, 64, 128), t_embed_dim=8.
  - *CIFAR-10/Fashion-MNIST (Advanced)*: Attempted deeper/wider architectures, e.g., down_chs=(64, 128, 256) or (64, 128, 256, 512), with larger t_embed_dim (e.g., 256 or 512).
- **Skip Connection Handling**: A significant modification was made to the U-Net's **init** and forward pass (deviating from the template's flawed DownBlock structure) to correctly capture feature maps *before* pooling in the downsampling path and pass them via skip connections to the corresponding UpBlock during the upsampling path. This involved separating the feature extraction (GELUConvBlocks) from the pooling (RearrangePoolBlock) within the U-Net definition.

4. *Conditioning*

- **Time Conditioning**: Timesteps t were embedded using SinusoidalPositionEmbedBlock (inspired by Transformers) followed by linear layers and activation, producing t_emb.
- **Class Conditioning**: Class labels c were provided as input. An EmbedBlock (using linear layers and activation) converted the class label (either an index converted to one-hot, or a pre-computed one-hot vector) into an embedding class_emb. The Deepseek fix ensured the EmbedBlock output shape was correctly [B, embed_dim, 1, 1].
- **Embedding Injection**: The time and class embeddings (t_emb, class_emb) were added to the feature maps at the bottleneck (middle) of the U-Net.
- **Conditioning Mask (c_mask)**: A boolean mask ([B]) was passed to the U-Net's forward method to control the application of embeddings, primarily for implementing Classifier-Free Guidance during training and generation. The Deepseek fix clarified how this mask should be applied element-wise to both t_emb and class_emb before adding them to the bottleneck features.

5. *Training*

- **Objective**: The model ϵ_θ was trained to predict the noise ϵ added during the forward process, given the noisy image x_t, the timestep t, and the class condition c.

Parnal Sinha
ITAI 2376 Midterm
Diffusion Model Supplemental Report
Dr. Patricia McManus

- **Loss Function**: Mean Squared Error (MSE) between the predicted noise $\epsilon_\theta(x_t, t, c)$ and the true noise ϵ was used: $L = \mathbb{E}_{t,x_0,\epsilon}\left[|\,|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_t} \cdot x_0 + \sqrt{1-\bar{\alpha}_t} \cdot \epsilon, t, c\right)|\,|^2\right]$.
- **Optimizer**: Adam optimizer was used.
- **Learning Rate Scheduling**: ReduceLROnPlateau scheduler was employed to decrease the learning rate if validation loss stagnated.
- **Gradient Clipping**: Applied torch.nn.utils.clip_grad_norm_ to prevent exploding gradients.
- **Validation**: After each training epoch, the model's performance was evaluated on a validation set (split from the training data) using the same MSE loss objective (calculated without backpropagation).
- **Early Stopping**: Training was configured to stop if the validation loss did not improve for a set number of epochs (early_stopping_patience).
- **Classifier-Free Guidance (CFG) Training**: During the "advanced" setup attempts, CFG was incorporated into the train_step. With a certain probability (e.g., 10%), the class condition c provided to the U-Net was effectively masked out (by passing c_mask=False), forcing the model to learn both conditional and unconditional noise prediction. This is crucial for enabling CFG during sampling.

6. *Sampling / Generation*

- **Basic (Template)**: The original template used the remove_noise function, implementing the basic DDPM reverse step without CFG.
- **CFG Sampling (Advanced)**: For the improved attempts, generation functions (generate_samples_cfg, redefined generate_number, visualize_generation_steps) implemented CFG. At each step t:
    1. The model predicted noise unconditionally: $\epsilon_\theta(x_t, t, c_{null})$ (by setting c_mask=False).
    2. The model predicted noise conditionally: $\epsilon_\theta(x_t, t, c)$ (by setting c_mask=True).
    3. The final noise estimate was calculated by extrapolating: $\hat{\epsilon} = \epsilon_\theta(x_t, t, c_{null}) + s \cdot \left(\epsilon_\theta(x_t, t, c) - \epsilon_\theta(x_t, t, c_{null})\right)$, where s is the cfg_scale.
    4. This combined $\hat{\epsilon}$ was used in the DDPM reverse step formula to calculate x_{t-1}.

7. *Datasets*

- **CelebA**: Initial target. Abandoned due to compute limitations and complexity.

- **CIFAR-10**: Second target. 32x32 RGB images. Normalized to [-1, 1]. Trained with advanced setup (Cosine, 1000 steps, CFG, deeper U-Net). Showed low loss but failed visually due to sampling instability.
- **Fashion-MNIST**: Third target. 28x28 grayscale images. Normalized to [-1, 1]. Trained with advanced setup (Cosine, 200 steps, CFG, slightly deeper U-Net). Showed promising early visuals but ultimately suffered similar sampling instability.
- **MNIST**: Final fallback. 28x28 grayscale images. Normalized to [-1, 1]. Attempted with both advanced setup (failed sampling) and basic template setup (poor visual quality). The final attempt focused on the basic setup with potential CFG during generation only.

Troubleshooting & Attempted Optimizations

The journey from CelebA to MNIST was marked by significant troubleshooting efforts aimed at achieving stable training and visually coherent image generation.

1. *Initial Challenges (CelebA & Early CIFAR-10)*

- **Compute Constraints**: Training on CelebA proved infeasible with available Colab compute units, leading to the switch to CIFAR-10.
- **Out-of-Memory (OOM) Errors**: Early attempts with larger batch sizes or deeper U-Nets on CIFAR-10 frequently resulted in OOM errors, necessitating reductions in BATCH_SIZE (e.g., to 32 or 64) and careful architecture choices.
- **Slow Convergence / Poor Visuals**: Initial runs (potentially using simpler settings or shorter training) yielded poor visual results, prompting the adoption of more advanced techniques.

2. *Advanced Setup Implementation (CIFAR-10 & Fashion-MNIST)*

- **Goal**: Improve stability and visual quality based on standard diffusion model practices.
- **Methods Introduced**:
  - **Cosine Noise Schedule**: Replaced the template's linear schedule, aiming for smoother noise transitions (N_STEPS increased to 1000 for CIFAR, later 200 for Fashion-MNIST).
  - **Deeper U-Net**: Increased channel depth (down_chs) and embedding dimensions (t_embed_dim) compared to the template's MNIST defaults to better handle dataset complexity.
  - **Classifier-Free Guidance (CFG)**: Implemented CFG during training (randomly masking condition) and generation (combining conditional/unconditional predictions) to enhance sample quality and class adherence.

  - **Corrected Skip Connections**: Modified the U-Net forward pass to correctly handle skip connections, addressing a flaw in the template's DownBlock design.

3. *Diagnosing Training vs. Generation Issues*

- **Successful Forward Process**: The show_noise_progression visualization consistently worked, confirming that adding noise via add_noise and the chosen schedule (Linear or Cosine) was numerically correct.
- **Stable Training**: Training logs across multiple runs (CIFAR-10, Fashion-MNIST with advanced setup) showed consistently decreasing training and validation loss (MSE), reaching low values (e.g., < 0.1). Periodic train_step debug prints confirmed that intermediate tensor values (x_t, noise, predicted_noise) remained stable during the training updates. This indicated the model *was* learning the noise prediction task numerically.
- **Generation Instability**: The primary failure point emerged during the reverse sampling process (generation). Debug prints added to the CFG generation functions revealed:
  - The model's noise predictions (eps_cond, eps_uncond, combined eps) were generally stable and within reasonable ranges.
  - However, the intermediate image representation x and the calculated mean term in the DDPM update formula exploded to extremely large positive and negative values over the course of the sampling steps (e.g., Std reaching hundreds/thousands).
  - This numerical explosion occurred even with N_STEPS reduced to 100/200 and persisted across CIFAR-10 and Fashion-MNIST using the advanced setup.
- **Visual Outcome**: The numerical explosion during sampling resulted in the final generated images appearing as corrupted noise or abstract patterns, despite the low training loss.

4. *Attempts to Stabilize Generation*

- **Clamping Predicted Noise (eps)**: Based on the diagnosis, torch.clamp(eps, -1.5, 1.5) was added inside the generation loop after CFG combination but before the mean calculation. **Result**: This did *not* prevent the explosion of x and mean values in the subsequent steps, indicating the instability was more deeply rooted in the iterative application of the formula with the chosen schedule/model.
- **Varying CFG Scale**: Tested different cfg_scale values (1.0, 3.0, 5.0, 7.0, 9.0) during generation. **Result**: While different scales produced slightly different noise patterns, none resolved the underlying numerical instability or produced clear images with the unstable setup.

- **Reducing N_STEPS (with Cosine/CFG)**: Reduced diffusion steps from 1000 (CIFAR) to 200, then 100 (Fashion-MNIST) while keeping the Cosine schedule and CFG. **Result**: Instability persisted even at 100 steps in this configuration.

- **Reverting to Basic Setup (MNIST)**: As a final fallback, reverted to the template's simplest MNIST configuration (Linear schedule, n_steps=100, basic U-Net params, no CFG). **Result**: Training was stable, but visual results were very poor and slow to converge (blurry blobs after 10 epochs).

- **Applying CFG to Basic Model (Option 2)**: Tested applying CFG *only* during generation using the model trained with the basic setup. **Result**: While potentially slightly better than no CFG, the underlying basic model likely hadn't learned enough for CFG to produce sharp results quickly.

- **Deepseek Fix**: Investigated a suggested fix related to EmbedBlock shape and U-Net masking logic. While the fix was applied to the UNet definition for the final basic MNIST run, it targeted a different potential issue (shape mismatch in embedding application) and did not resolve the observed generation instability, which occurred later in the sampling loop.

5. *Environment & Other Factors*

- **Runtime Restarts**: Frequent runtime restarts (due to timeouts or manual resets) necessitated rerunning setup cells, increasing the chance of configuration errors.

- **Template Quirks**: The template's DownBlock skip connection issue and potential inconsistencies in mask handling required modifications for correct U-Net functionality.

- **Compute Resources**: Significant compute resources (approximately 175 units) were expended during these troubleshooting cycles, highlighting the cost of debugging complex generative models.

Analysis & Examination

Despite extensive efforts and multiple iterations across different datasets and configurations, the project consistently encountered a critical bottleneck: **numerical instability during the reverse diffusion (sampling/generation) process.** This manifested as exploding intermediate image values (x and mean), leading to corrupted, noise-like final outputs, even when the model demonstrated successful learning based on training and validation loss metrics.

Key Observations

1. **Loss-Perception Gap**: The most striking observation was the significant disconnect between the low MSE loss achieved during training (often below 0.1 or even 0.07) and the poor perceptual quality of the generated images. This highlights a known limitation of MSE loss for generative tasks – it prioritizes pixel-level accuracy, which doesn't always

correlate with human perception of structure or realism. The model learned to predict the *average* noise well but failed to synthesize coherent global structures during generation.

2. **Sampling Instability**: Debug prints definitively showed that the numerical explosion occurred within the iterative denoising loop (for t_val in range(n_steps - 1, -1, -1): …). While the model's single-step noise predictions (eps) were stable, the repeated application of the DDPM update formula (mean = …; x = mean + variance*noise) led to divergence. This suggests sensitivity to the accumulation of small errors amplified by the schedule coefficients (α_t, β_t, $\bar{\alpha}_t$) over many steps.

3. **Impact of Advanced Methods**:
   - **CFG**: While CFG is known to improve sample quality, its introduction (requiring separate conditional/unconditional passes and scaling) might have exacerbated the numerical sensitivity in the sampling loop, especially with higher scales. However, the training itself remained stable with CFG.
   - **Cosine Schedule & Increased Steps (N_STEPS=200 or 1000)**: These aim for better theoretical performance but increase the number of iterations where instability can accumulate. The instability persisted even when reducing steps back to 100 while keeping the Cosine schedule and CFG.
   - **Deeper U-Net**: Provided more capacity but didn't prevent the sampling failure.
4. **Dataset Complexity**: The problem was most severe and appeared earlier with CIFAR-10 compared to Fashion-MNIST, suggesting the higher complexity and color channels of CIFAR-10 made the sampling process even more sensitive to instability. However, the failure even on Fashion-MNIST indicates the issue wasn't solely dataset-dependent but likely related to the specific implementation/hyperparameter combination.

5. **Ineffectiveness of Common Fixes**: Standard methods to address instability, such as clamping the predicted noise (eps) before the update step, did not resolve the exploding values in this case. This suggests the instability might stem from the coefficients in the DDPM formula itself or their interaction with the model outputs over many steps.

6. **Basic Setup Performance**: Reverting to the absolute basic template setup for MNIST (linear schedule, 100 steps, simple U-Net, no CFG) resulted in stable training and generation but produced visually poor, blurry outputs even after 10-20 epochs, highlighting the significant impact of techniques like CFG on visual convergence speed.

Root Cause Hypothesis

The most likely root cause is a fundamental numerical instability inherent in the specific combination of:

- The DDPM sampling formula used (especially the coefficients $1/\sqrt{\alpha_t}$ and $\frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}$).
- The chosen noise schedule (particularly the Cosine schedule over many steps, or even the Linear schedule if implementation details are sensitive).

- The specific architecture and learned weights of the U-Net model.

Small errors in the noise prediction ϵ_θ, when amplified by CFG scaling and multiplied by potentially large schedule-dependent coefficients within the iterative sampling loop, accumulate over hundreds of steps, leading to divergence and exploding values in the intermediate image representation x_t. While clamping eps helps in theory, it didn't fix the underlying sensitivity of the update formula itself in this context. More robust sampling algorithms (like DDIM) or different model parameterizations might be required for stability.

Future Direction

While the project faced significant challenges in achieving high-fidelity image generation, the process yielded valuable insights and highlighted several avenues for future work and improvement:

1. *Implement Alternative Sampling Methods (e.g., DDIM)*

The core issue identified was numerical instability during the DDPM sampling loop. Denoising Diffusion Implicit Models (DDIM) offer an alternative sampling process that uses a different update rule derived from a non-Markovian forward process. DDIM often allows for much faster sampling (fewer steps) and can be numerically more stable than DDPM, potentially avoiding the exploding value issue. Implementing and testing DDIM sampling with the trained models would be the highest priority next step.

2. *Hyperparameter Optimization*

- **Learning Rate**: The learning rate (e.g., 5e-4 or 1e-3) might have been too high, especially in later stages of training, potentially contributing to less stable model weights. Experimenting with lower initial learning rates, more aggressive decay schedules (e.g., cosine annealing), or adaptive optimizers could improve model robustness.
- **CFG Scale during Training/Sampling**: The fixed cfg_scale=5.0 or 3.0 used during generation might not have been optimal. Systematically evaluating a wider range of CFG scales during inference is crucial. Furthermore, exploring techniques that adapt the CFG scale during training or sampling might improve stability.
- **Noise Schedule Tuning**: While Cosine is standard, experimenting with the s parameter in the Cosine schedule or trying other schedules like variance-preserving (VP) or variance-exploding (VE) SDE-based schedules might yield different stability properties. Carefully tuning β_{start} and β_{end} for the linear schedule in the basic setup could also be beneficial.

3. *Architecture Refinement & Regularization*

- **U-Net Modifications**: While the template U-Net was used (with skip connection fixes), exploring variations like different normalization layers (e.g., BatchNorm instead of GroupNorm, though GroupNorm is common), adding dropout layers (as suggested in some DDPM papers for CIFAR), or adjusting the number/placement of attention blocks could impact stability and performance.
- **Weight Regularization**: Experimenting with different weight_decay values or other regularization techniques might prevent the model from learning weights that lead to unstable predictions during sampling.

4. Investigate Loss Function Alternatives

While MSE is standard for noise prediction, exploring alternative losses like L1 loss or perceptual losses (though more complex to implement) might lead the model to learn representations that are more robust during sampling, even if the MSE isn't perfectly minimized.

5. Code Verification & Simplification

Given the persistent issues, a thorough, line-by-line verification of the mathematical implementation of the DDPM sampling step against established libraries or papers would be warranted. Simplifying the U-Net architecture significantly (e.g., fewer layers/channels) for initial baseline testing could also help isolate problems.

Conclusion

This project undertook the challenging task of implementing and training conditional diffusion models for image generation, starting with ambitious goals for CelebA and progressively adapting to CIFAR-10, Fashion-MNIST, and finally MNIST due to computational and stability constraints. While the models demonstrated successful learning numerically, achieving consistently low training and validation MSE loss across various configurations, a critical failure point emerged during the reverse diffusion (sampling) process.

Extensive troubleshooting, including implementing advanced techniques like Cosine scheduling and Classifier-Free Guidance, modifying the U-Net architecture, adding debugging prints, and attempting numerical stabilization methods like clamping predicted noise, revealed a persistent numerical instability. Intermediate image representations exploded to extreme values during the iterative denoising loop, particularly with longer step counts (100-200 steps), leading to corrupted final visual outputs resembling noise rather than structured images. This occurred despite the model making reasonable single-step noise predictions.

The primary conclusion is that the specific combination of the DDPM sampling algorithm, the chosen noise schedules (especially Cosine over many steps), and the template U-Net architecture proved numerically sensitive and prone to divergence during generation in this implementation. While techniques like CFG showed promise in accelerating early visual structure formation (as seen briefly in Fashion-MNIST), they could not overcome the underlying sampling instability.

Parnal Sinha
ITAI 2376 Midterm
Diffusion Model Supplemental Report
Dr. Patricia McManus
Reverting to the most basic template configuration for MNIST resulted in stable but visually poor generation, highlighting the importance of advanced techniques like CFG for achieving quality results efficiently.

Despite not achieving the initial generative goals, the project provided significant practical experience in implementing diffusion model components, navigating common training pitfalls (OOM errors, hyperparameter sensitivity), diagnosing complex numerical issues through logging and analysis, and understanding the critical gap that can exist between optimizing a numerical loss function (MSE) and achieving desired perceptual quality in generative modeling. The documented failures and troubleshooting steps themselves represent a valuable learning outcome, emphasizing the need for robust sampling strategies (like DDIM), careful hyperparameter tuning, and potentially architectural adjustments when working with diffusion models, particularly on complex datasets or with limited computational budgets. The final pivot to MNIST, while a compromise, provides a stable baseline for future exploration and refinement.