Kolapo Mogaji

Professor Patricia McManus

ITAI 2376

April 9, 2025

1. Understanding Diffusion

A. Forward Diffusion Process:

In the forward diffusion process, noise is gradually added to a clean image over a series of steps until the image becomes indistinguishable from pure noise. At each step, a small amount of Gaussian noise is added, based on a pre-defined beta schedule. The progressive noise addition allows the model to learn the inverse mapping—i.e., how to recover the clean image from noisy inputs.

B. Why Gradual Noise Matters:

Adding noise incrementally provides the model with structured learning tasks. Rather than jumping directly from noise to image, the model learns small refinements at each step, improving overall stability and convergence.

C. Recognizable Images During Denoising:

During the denoising process, recognizable features of the image (e.g., digit outlines in MNIST) typically begin to emerge around the halfway point through the reverse steps (e.g., $t \approx 150$ out of 300). Earlier steps generate blurred, noisy approximations that become more defined over time.

2. Model Architecture

A. Why U-Net is Suitable:

U-Net is ideal for diffusion tasks because it combines local and global context through downsampling and upsampling layers with skip connections. The encoder compresses spatial features, while the decoder reconstructs the image using features from corresponding encoder layers.

B. Importance of Skip Connections:

Skip connections directly connect encoder and decoder layers at the same resolution. This helps preserve fine-grained features and prevents information loss during downsampling, crucial for high-fidelity image reconstruction.

C. Class Conditioning Mechanism:

Though not fully implemented in this version, class conditioning can be integrated by incorporating class embeddings into input channels or intermediate layers. This technique allows the model to control the generation process based on specific class labels.

3. Training Analysis

A. Loss Value Interpretation:

The loss function measures the difference between the model's predicted noise and the actual noise added during the forward process. A decreasing loss, as observed in training (e.g., Loss: 1.3057), indicates the model is learning to reverse the noise and recover the original image.

B. Image Quality Over Time:

Generated image samples show that during early training, outputs are heavily blurred and noisy. However, by the end of training, generated digits become more coherent, displaying outlines and structure similar to actual MNIST digits.

C. Role of Time Embedding:

Time embeddings enable the model to distinguish between different steps in the diffusion process. This is essential because the type and amount of noise vary across steps, and time awareness allows for appropriate denoising at each stage.

4. Implementation Details

A. Model Setup:

Implemented a lightweight U-Net using GroupNorm and GELU activations, with downsampling via Conv2D and upsampling via ConvTranspose2D layers. The architecture includes skip connections for effective feature reuse.

B. Diffusion Class:

Manages the forward noise schedule with 300 time steps. Uses linearly spaced beta values and calculates cumulative alpha products to simulate forward noise and facilitate reverse sampling.

C. Training Execution:

- Used the MNIST dataset, normalized and loaded in batches of 128.
- Training loop included resizing of noisy targets when input/output shape mismatches occurred.

- Optimized with Adam, learning rate = 1e-4.

D. Image Generation Function:

- Generates images from pure noise using the trained model.
- Iteratively denoises in reverse order of time steps.
- Final image displayed using Matplotlib with RGB transposition.

5. CLIP Evaluation (Optional - Not Attempted)

While CLIP evaluation was not implemented in this version due to time and resource constraints, the proposed idea would involve comparing image embeddings generated by CLIP with corresponding text labels to evaluate semantic consistency.

6. Practical Applications

A. Real-World Use Cases:

- AI Art Generation (e.g., DALL·E)
- Medical Imaging Enhancement
- Data Augmentation for ML pipelines
- Restoration of corrupted images

B. Limitations:

- High computational cost
- Sensitive to training hyperparameters
- Potential for bias or hallucination in outputs

C. Proposed Improvements:

1. Add Class Conditioning: Improve control over generation.
2. Experiment with CIFAR-10: Train on more complex data.
3. Implement CLIP Scoring: Evaluate generated images semantically.

Conclusion

This assignment provided a comprehensive understanding of diffusion models, blending theory, architecture, and practice. By building a model capable of generating MNIST digits from noise, I gained practical insights into the mechanics behind state-of-the-art generative models.