During this project, I explored the implementation of diffusion models for generating images from the MNIST dataset. This hands-on experience allowed me to develop a deeper understanding of forward and reverse diffusion processes, U-Net architecture, and model training using PyTorch. Despite facing technical challenges, the iterative debugging process helped refine my approach and enhanced my problem-solving skills in machine learning. In this report, I will summarize my findings, outline the diffusion model pipeline, reflect on the challenges encountered, and discuss potential improvements for future research. Additionally, I will analyze specific machine learning concepts such as time embedding, CLIP evaluation, and model conditioning mechanisms.

Understanding Diffusion

Diffusion models function by progressively corrupting images with Gaussian noise in a step-by-step process known as forward diffusion. This process transforms clear images into randomized noise, making them unrecognizable. The reverse diffusion process then learns to remove the noise incrementally, reconstructing the image based on learned patterns.

Why Noise is Added Gradually

Gradual noise addition is crucial because it enables the model to learn the underlying structure of images while being exposed to varying degrees of distortion. If noise were added all at once, the model would struggle to capture meaningful relationships between pixels and would fail to reconstruct accurate images. The staged corruption helps the model build a structured denoising process, improving its ability to recover realistic images.

At What Point Do Images Become Recognizable?

During early stages of denoising, the images remain heavily distorted, resembling amorphous blobs. However, as the model removes noise over successive steps, patterns begin to emerge. Around the middle-to-late stages of diffusion, recognizable MNIST digits form, though with some artifacts. By the final steps, the reconstructed images closely resemble their original forms, proving the effectiveness of the denoising function.

Model Architecture

The U-Net architecture was chosen for this project due to its effectiveness in image reconstruction. U-Net consists of an encoder-decoder structure with skip connections, which allows it to preserve fine details while removing noise from images.

Why U-Net is Well-Suited for Diffusion Models

U-Net's convolutional layers extract spatial features, while upsampling layers reconstruct the image. The downsampling process compresses image information into meaningful representations, helping the model focus on key structural details needed for accurate denoising.

Importance of Skip Connections

Skip connections link encoder layers directly to decoder layers, allowing the model to retain original image features throughout processing. This is essential in diffusion models because the reconstruction process benefits from high-frequency details captured during the encoding phase. Without skip connections, small but important pixel-level features would be lost.

Class Conditioning Mechanism

Class conditioning helps the model generate images associated with a particular class label (e.g., specific digits in MNIST). This is implemented by adding class embeddings to the model's feature maps, guiding the reconstruction process toward producing images from a specific category. While MNIST doesn't require complex conditioning, this method is crucial for more advanced datasets where multiple object types exist.

Training Analysis

Interpreting Loss Values

Loss measures the difference between the generated image and the original image. A decreasing loss over epochs indicates successful training, as the model becomes better at predicting noise and removing it accurately.

Changes in Generated Image Quality During Training

Early in training, generated images are highly distorted, showing little resemblance to actual MNIST digits. By mid-training, digits start taking shape but still contain blur and artifacts. Toward the final epochs, the reconstructions become increasingly accurate, closely mirroring real handwritten numbers.

Time Embedding in Diffusion Models

Time embedding provides the model with information about how far an image is in the diffusion process. It allows the U-Net to adjust its reconstruction approach based on how much noise is present at a given step, helping it predict the correct amount of denoising needed at each stage.

CLIP Evaluation

Understanding CLIP Scores

CLIP evaluates the quality of generated images by comparing them with human-labeled text descriptions. Higher CLIP scores indicate that the AI-generated images align well with the expected output.

Why Some Images Are Harder to Generate

Certain images are more challenging to reconstruct due to complex pixel arrangements and overlapping features. For instance, digits with similar structures (e.g., ‘5’ and ‘6’) may introduce ambiguity, making the model struggle to distinguish subtle differences.

Improving Diffusion Models with CLIP Scores

CLIP scores can help improve the diffusion process by rewarding realistic generations and fine-tuning the loss function. Adjusting the training objective based on CLIP evaluations can lead to better image fidelity, reducing reconstruction artifacts.

Practical Applications

Real-World Uses of Diffusion Models

Diffusion models have vast applications, including:

- Image Restoration: Enhancing old or damaged images through AI-powered reconstruction.
- Synthetic Data Generation: Creating artificial datasets for AI training, useful in medical imaging and security.
- Art and Creative Design: Used in AI-generated artwork and stylized content creation.

Limitations of the Current Model

While effective for MNIST, this diffusion model struggles with high-resolution images, lacks advanced conditioning mechanisms, and requires significant computational power for large-scale training.

Three Specific Improvements for Future Models

- Use deeper U-Net layers to improve reconstruction accuracy.
- Train on higher-resolution datasets like CIFAR-10 to enhance complexity handling.
- Implement CLIP-based evaluation to refine image realism during training.

Bonus Challenge: Exploring Advanced Architectures

As part of the bonus challenge, I attempted to modify the standard U-Net by integrating residual connections to enhance performance. This variation improved feature retention and stabilized training, but required additional memory, highlighting the trade-offs in complex models.

Findings from Bonus Challenge

- Residual layers improved reconstruction quality but increased model size.
- Batch normalization stabilized training, reducing artifacts in generated images.

- Higher learning rates caused instability, requiring precise tuning.

These insights emphasize that model architecture plays a key role in the success of diffusion models and that strategic modifications can improve performance without compromising efficiency.

Conclusion

Implementing a diffusion model from scratch provided invaluable experience in deep learning and generative AI. While the debugging process was challenging, overcoming technical obstacles strengthened my coding and analytical skills. Moving forward, I am excited to expand upon these findings by experimenting with higher-resolution datasets, alternative architectures, and hybrid generative models.