MID-TERM ASSIGNMENT REPORT

My Midterm Project: Generating Fashion-MNIST Images with a Diffusion Model

For this assignment, I chose to work with the **Fashion-MNIST dataset**. The main goal was to build a diffusion model that could generate realistic images of clothing items from scratch. This involved several key stages: understanding the diffusion process, implementing a U-Net architecture, training the model by progressively adding and then removing noise, and finally, analyzing the quality of the images I generated. I also took on the optional challenge of using CLIP (Contrastive Language-Image Pre-training) for evaluation.

1. Getting Started and Setting Up

I started by setting up my environment in Google Colab, ensuring GPU acceleration was enabled, which is crucial for training these kinds of models. I used the provided Jupyter notebook (L09_Diffusion_student_notebook_ITAI_2376.ipynb) as my base.

2. Implementing the Diffusion Model

This was the core of the assignment, and I had to complete several code sections.

- **Dataset Setup (Fashion-MNIST):** I loaded the Fashion-MNIST dataset, which consists of 28x28 grayscale images of 10 different clothing categories. I implemented the necessary preprocessing, including transforming images to tensors and normalizing them. I also split the dataset into training and validation sets and created data loaders to feed the data to the model efficiently.
- **Model Architecture (U-Net):** I implemented a U-Net model. The U-Net is great for these tasks because its encoder-decoder structure with skip connections helps capture both high-level semantic information and fine-grained details.
  - I built components like ConvBlock (using GELU activation and GroupNorm), DownSample, and UpSample blocks. The UpSample blocks were designed to explicitly accept channel counts from skip connections.
  - Crucially, I implemented **time and class conditioning**. Time conditioning, using sinusoidal position embeddings similar to Transformers, tells the model which step of the diffusion process it's currently in. Class conditioning, using one-hot encoded class

labels projected into an embedding space, guides the model to generate images of a specific category. These two embeddings are combined and projected to influence the U-Net's bottleneck.

- **Diffusion Process:**
  - **Forward Diffusion:** I implemented the process of gradually adding Gaussian noise to the images over a set number of TIMESTEPS (I used 100). The amount of noise is controlled by a linear schedule of betas (variance). The equation I used for adding noise at timestep t to an image x0 was xt=αtx0+1−αtϵ, where αt is the cumulative product of αs=1−βs.
  - **Reverse Diffusion:** The U-Net model's job is to learn to reverse this process – to predict and remove the noise at each timestep, starting from pure noise and gradually generating a clean image. My remove_noise function took the noisy image xt, the current timestep t, the model, and conditioning information (class label) as input, and predicted the noise to subtract.
- **Training:** I set up the training loop using the Adam optimizer and a Mean Squared Error (MSE) loss between the predicted noise and the actual noise added during the forward process. I trained the model for 30 epochs with a constant learning rate of 0.001 and a batch size of 64. I also implemented gradient clipping and saved checkpoints whenever the validation loss improved.

3. Training Analysis and Results

- **Loss Value:** The loss value (MSE) indicates how well the model is predicting the noise that was added to the image at each step. My training loss started higher (around 0.1695 in the first epoch) and progressively decreased to about 0.1075 by the end of training, with validation loss also showing a similar trend (ending around 0.1109, though the best was ~0.1069 around epoch 29). This decreasing loss showed that the model was indeed learning to reverse the diffusion process effectively.
- **Generated Image Quality:**
  - In the early epochs, the generated images were, as expected, quite fuzzy and didn't look like much.
  - As training progressed, the quality significantly improved. The images started becoming more organized and clearly resembled Fashion-MNIST items like shirts, trousers, dresses, etc.
  - I visualized the step-by-step generation process (reverse diffusion). Starting from pure noise at t=99, by around t=39 or t=19 (about 30-40% into the reverse process), distinct features of the clothing items started to emerge, becoming quite recognizable by t=3.

4. CLIP Evaluation (Bonus)

I decided to implement the CLIP evaluation to get a more quantitative measure of how well my generated images aligned with textual descriptions.

- I loaded a pre-trained CLIP model (ViT-B/32).
- For each generated image, I created a prompt like "a clear, well-detailed Fashion-MNIST image of class 5" (for class 5, which is 'Sandal').

- Then, I calculated the cosine similarity between the CLIP embeddings of the generated images and the text prompt.
- The CLIP scores I obtained were generally in the range of 0.26 to 0.28. While these might not seem very high on an absolute scale, they provide a good relative measure. For example, for class 5 ('Sandal'), the scores were around [0.2664, 0.2593, 0.2832, 0.2654, 0.2781].
- **Hypothesis on Generation Difficulty:** I hypothesized that images/classes with more distinct and simpler structures (like T-shirts or Trousers) might be easier for the model to generate and thus achieve higher CLIP scores compared to more complex or varied items (perhaps certain types of bags or intricate shoe designs).
- **Improving with CLIP:** I thought that CLIP scores could potentially be used to improve the diffusion generation process itself, perhaps by using them as a feedback signal to reinforce training on higher-scoring samples or even guide the generation process directly (e.g., CLIP-guided filtering or reinforcement learning).

5. Understanding the Concepts

- **Forward Diffusion:** This is the process where I systematically destroy structure in an image by adding Gaussian noise over many timesteps. It's a Markov chain, meaning each noisy version xt only depends on xt-1. Adding noise gradually is key because it creates a sequence of progressively noisier images, providing a clear learning signal for the model to learn how to reverse this degradation step-by-step. If all noise was added at once, it would be a much harder problem for the model.
- **U-Net Architecture Suitability:**The U-Net is excellent for diffusion models because:
    - Its **encoder-decoder structure** allows it to process images at multiple scales, capturing both global context and local details.
    - **Skip connections** are vital. They pass high-resolution feature maps directly from the encoder to corresponding decoder layers. This helps the decoder reconstruct finer details that might be lost during downsampling and also helps with gradient flow during training.
- **Class Conditioning:** In my model, this mechanism allowed me to guide the generation towards a specific Fashion-MNIST class. I achieved this by taking the one-hot encoded class label (e.g., a 10-dimensional vector for Fashion-MNIST), passing it through an embedding layer (a small neural network) to get a class embedding. This class embedding was then combined (summed) with the time embedding for the current timestep. This combined conditioning vector was then projected to match the channel dimension of the U-Net's bottleneck and added to the feature maps at that stage, influencing the denoising process to produce an image of the target class.
- **Time Embedding:** This is crucial because the model needs to know *how much* noise to remove at each step of the reverse diffusion process. The amount of noise changes with each timestep. By providing a time embedding (I used sinusoidal positional embeddings), the model can learn to adapt its denoising strength based on the current timestep t. It essentially gives the model a "sense of time" in the diffusion sequence.

6. Practical Applications and Potential Improvements

- **Real-World Applications:**

    - **Content Creation:** Generating unique images for art, design, advertising, or virtual environments.
    - **Data Augmentation:** Creating synthetic training data for other computer vision models, especially when real data is scarce.
    - **Image Editing & Restoration:** Tasks like inpainting (filling in missing parts of an image) or super-resolution.
    - **Fashion Design:** Prototyping new clothing designs or variations.
    - **Medical Imaging:** Assisting in tasks like medical image segmentation or reconstruction, where detail is critical.
- **Limitations of My Current Model:**
    - **Low Resolution:** Fashion-MNIST images are only 28x28, which limits detail.
    - **Computational Cost:** The iterative nature of diffusion models makes generation (and training) computationally intensive, especially for higher resolutions. I personally found that attempting the CIFAR-10 dataset was not feasible due to GPU limitations on Colab (I ran out of credits while troubleshooting and sometimes my sessions would run too long).
    - **Limited Diversity:** While recognizable, the generated images might lack true diversity or creativity, partly due to the simplicity of the dataset and model.
- **Proposed Improvements:**
    1. **Increase Network Capacity:** Using a deeper or wider U-Net (more layers or channels) could allow the model to capture more complex patterns and generate higher-fidelity images. This would involve adding more down/up sampling blocks or increasing channel counts, while carefully maintaining skip connections.
    2. **Advanced Noise Scheduling:** Instead of a simple linear beta schedule, exploring other schedules like a cosine schedule, or even a learned noise schedule, could lead to faster convergence or better sample quality.
    3. **CLIP-Guided Training/Generation:** More deeply integrating CLIP, perhaps by using its scores as part of the loss function during training (reinforcement learning approach) or to guide the sampling process directly, could help generate images that better align with specific textual descriptions or desired attributes.

Conclusion

Overall, this midterm was a challenging but incredibly rewarding experience. I successfully implemented a diffusion model that could generate recognizable Fashion-MNIST images. The process involved a lot of debugging, especially around tensor shapes and channel dimensions in the U-Net, but seeing the model gradually learn to denoise images from pure noise to coherent forms was fascinating. The optional CLIP evaluation also provided a valuable quantitative perspective. I feel this project significantly deepened my understanding of both the theory and practical application of diffusion models in generative AI.