Assessment Questions

Now that you've completed the exercise, answer these questions include explanations, observations, and your analysis Support your answers with specific examples from your experiments:

1. Understanding Diffusion

- Explain what happens during the forward diffusion process, using your own words and referencing the visualization examples from your notebook.
    - During the forward diffusion process, our model incrementally adds noise to the image. Basically it takes a clear image and adds static to the image little by little. Through the noise progression visualizations, I was able to see how the initial clean image gradually became indistinguishable and lost its structure as the model, guided by the beta schedule, added noise at each step.
- Why do we add noise gradually instead of all at once? How does this affect the learning process?
    - Adding noise gradually allows the model to learn a distribution of the data at each level that noise is added. If we added the noise all at once then the model would have to learn how to take noise and instantly create a clear image. This affects the learning process because the model creates intermediate noisy versions of the data and learns to predict the noise added at each step. It uses what it learns and applies that to the reverse process. This makes learning manageable for the model and allows it to successfully denoise images during the generation process.
- Look at the step-by-step visualization - at what point (approximately what percentage through the denoising process) can you first recognize the image? Does this vary by image?
    - Based on the step-by-step visualization I was able to recognize the image starting at approximately 50 percent of the way through the denoising process. It does vary by type of image generated. For example, for generating a zero I was able to recognize the image at step 277/500. However it wasn’t until step 221/500 that I was able to recognize the three or the 7.

2. Model Architecture

- Why is the U-Net architecture particularly well-suited for diffusion models? What advantages does it provide over simpler architectures?
    - The U-Net architecture is well-suited for diffusion models because of its ability to preserve the fine-grained details obtained during the corruption process while still capturing both local and global features. This is because of its encoder-decoder structure using skip connections, this gives it an advantage over most other models. Through this structure the U-Net is able to efficiently predict noise during

the denoising process. Other architectures tend to struggle with preserving the spatial information that is needed to accurately predict noise at every pixel.

- What are skip connections and why are they important? Explain them in relations to our model
    - Skip connections are basically detours that information from earlier layers take so they can be applied to later layers. In the model they are implemented in the upblock. They take the output of a downblock and concatenate that information with the upblock's information before moving forward with processing. They are needed because the downblock loses spatial resolution but captures high level features. They are used to preserve and transfer the fine-grained details from the downblock to the upblock so the model can accurately reconstruct the image and predict the noise, especially around edges and textures.
- Describe in detail how our model is conditioned to generate specific images. How does the class conditioning mechanism work?
    - The class conditioning for our model to generate specific images is done through how we implement the class embedding module and the c input used in the U-Net's forward pass. To do this we first start with converting our class labels given as input into one hot vectors. It is then passed through the EmbedBlock which has a Linear Layer and a GELU activation followed by an Unflattening step to reshape it to [batch_size, c_embed_dim, 1, 1]. This is what creates a learned embedding for our classes. Then the class embedding is added to the feature maps created by the middle block. By adding the class embedding, the model learns a more nuanced and target aware internal representation of the input data. Now when we want to generate a number we have to specify which number. Then the model creates a corresponding one-hot encoded class. This is then passed to the model at each denoising step. This helps steer the model to generate an image of that class. I also found out from some research that the c_mask portion of the code is for potential classifier free guidance.

3. Training Analysis (20 points)

- What does the loss value of your model tell us?
    - The loss value of the model tells us how accurately it is predicting the noise that was added to the image at a given step. The lower the loss, the closer the model's prediction to the actual noise that was added. As with most models we want to reduce this loss to the lowest we possibly can. This indicates that the model is getting better and learning how to reverse or denoise the images.
- How did the quality of your generated images change throughout the training process?
    - In the first epoch, the images generated were just of random strokes, blotches, or noise with a little semblance of the number being predicted. Then around epoch 15 the images became clearer and more recognizable to their targets. When

training was over the images weren't perfect but were pretty good. They had some minor imperfections and variations in style. The improvement of quality over the course of the training was depicted by the loss curve decreasing each epoch. This indicates that the model was getting better at its predictions each time.

- Why do we need the time embedding in diffusion models? How does it help the model understand where it is in the denoising process?
    - In a diffusion model, each time step corresponds to a phase where the model adjusts the image slightly. Time embeddings are numerical representations of time-related data, specifically designed to capture the sequential nature and relationships within time series data. They are essentially just vector encodings of the downsampling path at each time step. This helps our model understand where it is by differentiating between the early, middle, and late stages of denoising.

4. CLIP Evaluation (20 points)

- What do the CLIP scores tell you about your generated images? Which images got the highest and lowest quality scores?
    - The good and clear scores convey how close CLIP believes my generated images are to the real thing. The Blurry score is self explanatory, it depicts how pixelated or blurry my generated images are. The images that got the best scores were out of the batches of eight, three, and seven.
- Develop a hypothesis explaining why certain images might be easier or harder for the model to generate convincingly.
    - My hypothesis for this is that some numbers look like a backwards version of others. For example a five looks like a backwards two, five could look like a 6, and a 6 could look like a backwards 2. The underlying structure of these digits are so close that the model gets confused since it is extracting features from the pixels of their underlying structures.
- How could CLIP scores be used to improve the diffusion model's generation process? Propose a specific technique.
    - You could use an actor critic approach to learning similar to what is done in reinforcement learning or how GANs are trained. I could make the model check the CLIP scores and adjust hyperparameters or weights accordingly. The model's noise predictions can then be slightly adjusted at each time step to increase the CLIP similarity between the partially denoised image and the target digit.

5. Practical Applications (20 points)

- How could this type of model be useful in the real world?
    - This model is useful in many applications in the real world. It is used in image generation by creating realistic and diverse images. It is also used in image editing to manipulate images by inpainting, outpainting, style transfer, and

super-resolution. There is also a use for it in Data augmentation by generating synthetic data to expand training datasets. Lastly, it helps with content creation by assisting content creators in bringing their ideas to reality through the generation of images, videos, and audio from simple text prompts.

- What are the limitations of our current model?
    - The first limitation is the image resolution of the training data, our model is trained on 28x28 pixel images. This makes the model generate rather pixelated and blurry images. Next limitation would be lack of diversity being that it is only trained on digits. This means it cannot be used to generate images other than digits. The next limitation would be training time. This model took a lot of time to train and drained my compute units in colab in no time. The next limitation is the lack of control over thickness, style, or other visual attributes.
- If you were to continue developing this project, what three specific improvements would you make and why?
    - If I had more compute the first thing I would improve would be to use one of the more feature rich datasets like the CIFAR-10 dataset. This would bring the model closer to a real world application that generates images people actively try to generate every day.
    - I would like to implement some kind of actor critic approach to the training process. This would enhance the models ability to understand what is a good generated image and what isn't keeping it on track as it trains and potentially decrease my overall loss value further.
    - I could also apply attention mechanisms which would enhance the models ability to extract more meaningful features from the data. From my research i've seen that attention layers can help with datasets that have consistent global structures like digits do. This should hopefully help the model improve the coherence and quality of the generated images.

Bonus Challenge (20 points):

- If you were to continue developing this project, what three specific improvements would you make and why?
    - If I had more compute the first thing I would improve would be to use one of the more feature rich datasets like the CIFAR-10 dataset. This would bring the model closer to a real world application that generates images people actively try to generate every day.
    - I would like to implement some kind of actor critic approach to the training process. This would enhance the models ability to understand what is a good generated image and what isn't keeping it on track as it trains and potentially decrease my overall loss value further.

- I could also apply attention mechanisms which would enhance the models ability to extract more meaningful features from the data. From my research i've seen that attention layers can help with datasets that have consistent global structures like digits do. This should hopefully help the model improve the coherence and quality of the generated images.

Best Generated Number Images: