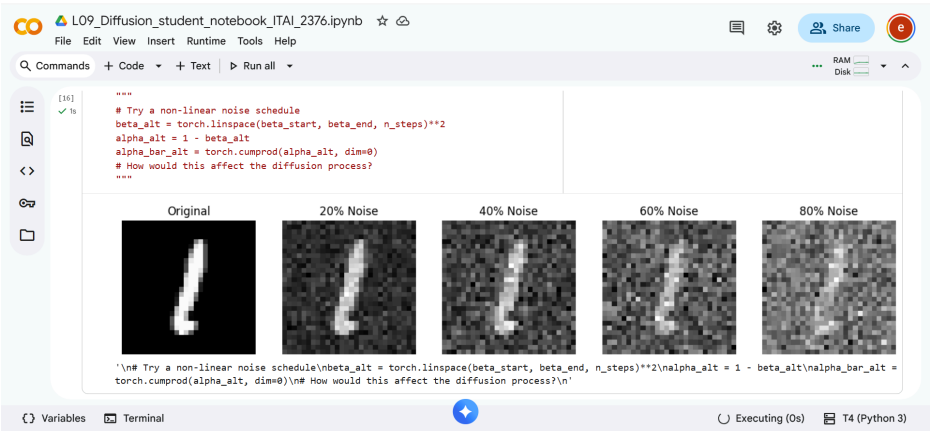
Diffusion Model Experiment Report

In this experiment, I worked on generating different images of handwritten digits using a diffusion model. The main goal was to understand how noise can be added to and removed from images step by step, and how the model learns to reconstruct clear images from that process. Diffusion models are based on probability and mathematics — they start by turning an image into random noise and then try to reverse this process to get back the original data. In simple terms, the model "learns" to clean noisy images until they look like the original training samples.

I began by generating four versions of each digit (from 0 to 9) to observe how the model handled variations. The function generate_number() was used to create these samples, and the results were displayed using matplotlib. The model progressively removed noise through many time steps (from high noise to low noise), which allowed me to visualize the diffusion process clearly. Each denoising step represented the model improving its guess about what the image should look like. The mathematics behind this process involves gradually reducing a noise term using learned parameters so that the model approximates the probability distribution of the data. This can be described by a function where the predicted output x_t becomes less random as t decreases.

To explore randomness, I used the helper function generate_with_seed() to generate images of the same digit but with different noise seeds. A seed sets the starting point for random number generation — changing it makes the model create slightly different styles each time even though it's generating the same digit. For example, when I generated multiple versions of the number "3" with different seeds, I noticed that some looked more curved, while others were bolder or slightly tilted. This variation shows how diffusion models can capture multiple possible representations of the same concept.



Mathematically, this randomness comes from how Gaussian noise is sampled and later reduced by the model. The diffusion process adds noise based on a variance schedule, and the reverse process estimates the original data distribution p(x0) from the noisy samples xt. Even though the underlying math is complex, the main idea is that the model learns the statistical pattern of digits through repeated noise predictions. Over time, it becomes very good at removing noise in a way that reflects what the real image should look like.

During the experiment, I also faced a few problems. One issue was not knowing exactly where to place the code sections in the file at first. I learned that indentation and placement in Python matter a lot — placing the code after the wrong loop or outside a function caused errors like “IndentationError” and “NameError.” I also had to make sure the right device (CPU or GPU) was being used, otherwise the model would run slowly or throw device mismatch errors. Another problem was forgetting to include torch.manual_seed() inside the function, which made the results inconsistent every time I ran it. Fixing these issues helped me understand how deterministic randomness works in experiments and how important debugging is in machine learning.

From this experiment, I learned that diffusion models are not only powerful but also creative — they can produce different outputs from the same input just by changing the random seed. I also saw how mathematical ideas like noise addition, Gaussian distribution, and probability functions play a key role in how the model learns to denoise images. Overall, this exercise helped me understand both the coding and the mathematical intuition behind generative AI models.

Assessment Questions and Answers

1. What did the model learn to do in this experiment?

The model learned how to reverse the noise-adding process to reconstruct clear, realistic images from random noise. It essentially learned to predict what an image should look like at each step of the denoising process until it became a recognizable digit.

2. How did you explore randomness using seeds, and what did you notice?

I used different random seeds with the generate_with_seed() function to create multiple versions of the same digit. Each seed produced unique styles — some digits appeared thicker or more curved. This showed me how diffusion models can generate a variety of possible outputs even when the target digit is the same.

3. What mathematical concepts were involved in this experiment?

This experiment used mathematical ideas from probability and statistics, especially Gaussian noise,

variance scheduling, and conditional probability. The diffusion process is governed by equations that add and remove noise based on a time variable t, helping the model learn to approximate the data distribution step by step. The mean squared error (MSE) loss was used to measure how close the model's predictions were to the real data at each step.

4. What challenges or problems did you face while working on the experiment?

I initially had trouble knowing where to place the new code blocks, which caused indentation and function scope errors. I also forgot to set the random seed in some cases, which led to inconsistent results. Additionally, I learned that device mismatches between CPU and GPU could cause runtime errors. Solving these problems helped me understand how coding structure, reproducibility, and hardware compatibility affect experiments.

5. What did you learn about diffusion models and their potential?

I learned that diffusion models are extremely flexible and capable of creating high-quality, unique outputs from randomness. They don't just copy training images — they learn patterns deeply enough to create new versions that still fit the data distribution. This has real-world potential for creative fields like art and design, as well as technical applications like data generation, restoration of damaged images, and synthetic dataset creation for AI training.

6. How does the concept of overfitting relate to this experiment?

Overfitting happens when a model learns the training data too specifically and struggles to generalize to new data. In diffusion models, if training continues for too long or with poorly balanced data, the model might generate repetitive images instead of creative variations. Observing the loss ratio between validation and training data helps detect overfitting — if validation loss starts increasing while training loss decreases, it's a sign that the model is memorizing instead of learning.

7. What was the most interesting part of the project?

The most interesting part was seeing how just changing the random seed produced different versions of the same digit. It made the model feel creative — almost like it had its own artistic interpretation of each number. This gave me a deeper appreciation of how randomness and probability can enhance generative AI, making it more precise and expressive.

References

- Ho, J., Jain, A., & Abbeel, P. (2020). *Denoising Diffusion Probabilistic Models*. NeurIPS.

- Kingma, D. P., & Welling, M. (2013). *Auto-Encoding Variational Bayes*. arXiv:1312.6114.

- PyTorch Documentation. (2024). *torch.manual_seed — Reproducibility in PyTorch*. https://pytorch.org/docs/stable/torch.html

- Hugging Face. (2024). *Diffusion Models Explained Simply*. https://huggingface.co/blog/annotated-diffusion