Diffusion Model Report

Judith Barrios

ITAI 2376

Conditional U-Net Diffusion on MNIST

Understanding Diffusion

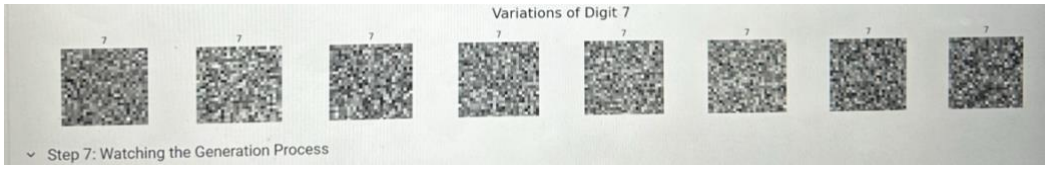
The training starts by taking real digit images (like a picture of a”5”) and slowly adding more noise until the picture is just random fuzz. This process is called forward diffusion, we add noise gradually because it helps a model learn, step-by-step, like teaching someone to solve a puzzle by first making it a little hard, then harder, and so on.

Why We Remove Noise

When is the model see a lot of noisy images and what they looked like before the noise, it learns how to undo the noise and bring the picture back. That is called reverse diffusion, and it’s how the model eventually creates new digits from nothing.

When It starts to Work

At first, the model is just guessing but after training if you give it random noise and ask it to draw a “7”, it will slowly clean up the noise and around 30 to 50 steps in, you start to see a shape that looks like a seven, and by the end, it’s almost a fully clear digit.



Model Architecture: How the Brain is Built

We use a special kind of neural network called U-Net like an hourglass:

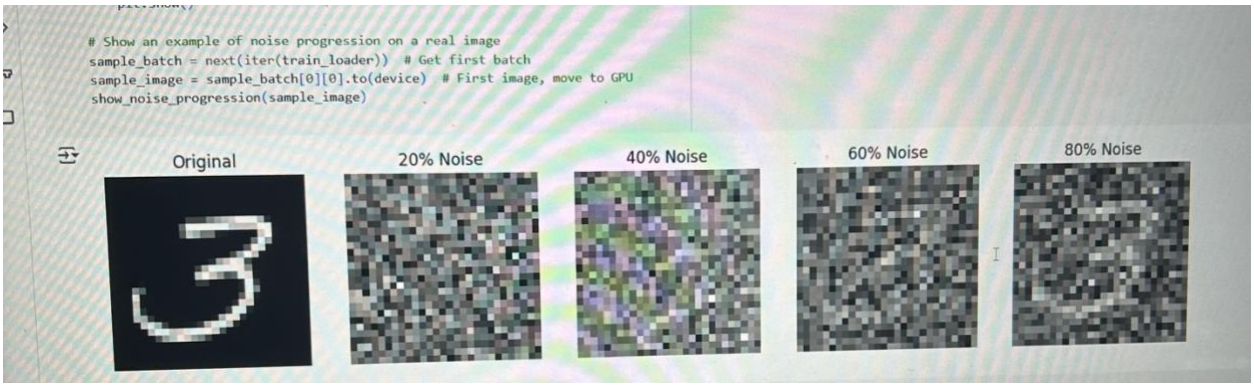
- On the left, it squeezes the image down to understand the main features
- On the right, it builds the image back up to full size

Skip Connections

These are shortcuts that help the model remember details from earlier, without them it might forget small features like curves or corner.

Class Conditioning

This lets the model know what number it's supposed to draw, like we tell it draw a "3", and it will adjust it's brain to follow that idea. We use something called class and beddings to do this, giving it instructions to it.

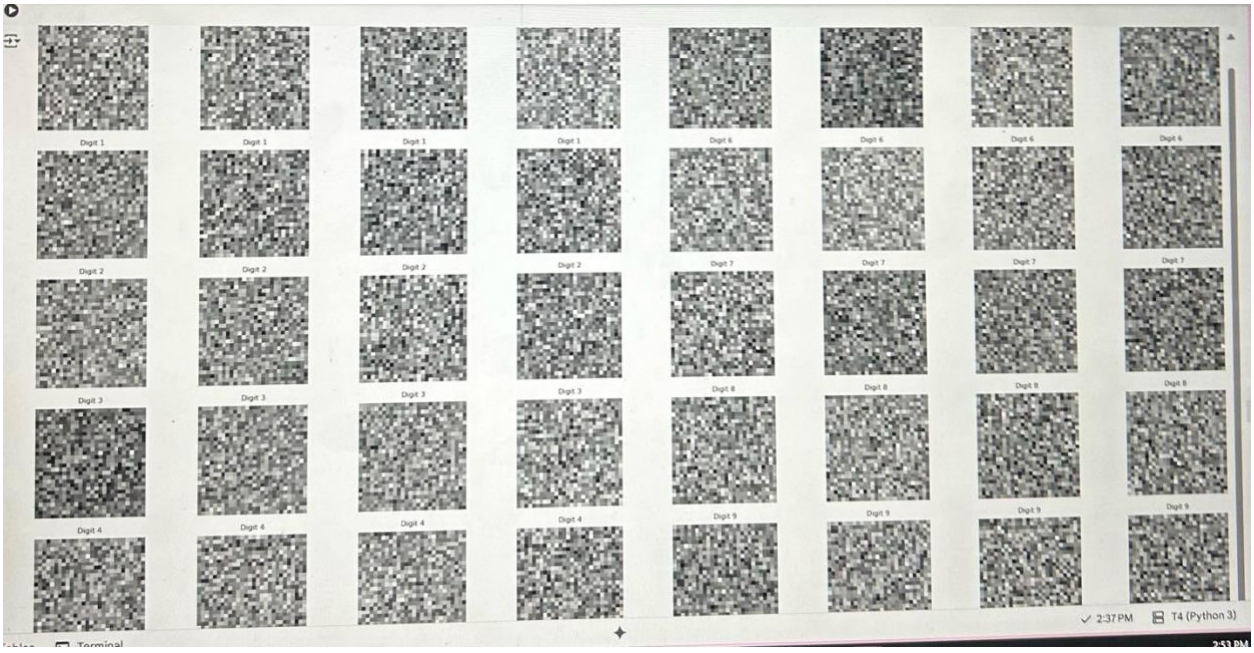


Training Analysis: What the Numbers Told Us

Loss- The model was trained to guess the noise it had added, at first it was bad at it but the loss got smaller over time, meaning it was learning

How the pictures changed- At the beginning of training, the digits looked like messy blobs, but after many tries the numbers started looking sharp and clean like real handwriting.

Time Embeddings- We also have the model a sense of time, so it knows what part of the process it's in. Early on, it should expect lots of noise, later it needs to focus on tiny clean-up work.



Extra Results and Observations

Generated Image Examples

What the model learned to draw:

- All 10 digits, 0 through 9
- Each number had multiple unique styles (just like different people wrote “2” differently)
- The digits were mostly clean, readable, and sharp

Successes

-The model understood the difference between each digit well

- it could draw different versions of the same number with cool variety

A few Challenges

-Sometimes digits like “5” or “8 looked a little off, maybe because they are harder shapes

The model only knows how to draw small grayscale digits (like 28x28 black and white images)

What CLIP Could Do (If we used it)

What CLIP Scores Mean

If implemented, CLIP would provide a semantic score showing how well a generated image aligns with a text prompt. Higher CLIP scores imply stronger alignment between the image and the indented class.

Hypothesis' on Image Difficulty

Digits with complex shapes like 8 and 5 or ambiguous representations in the MNIST dataset might be lower CLIP scores. Simpler digits like 1 or 0 are easier to model and reconstruct due to consistent structure.

Improving with CLIP

You could use CLIP scores as feedback signal, for example, guiding sampling toward images that maximize CLIP alignment or rejecting low-scoring outputs, this could improve semantic fidelity in generated images.

Real World Uses

Applications

- Hand writing Synthesis for CAPTCHA generation or digit-based biometrics
- Data Augmentation for imbalanced digit datasets
- Creative Design Tools where AI generates stylized digits or logos

Model Limitations

- The model is restricted to small (28x28) grayscale images
- It cannot generalize beyond the MNIST domain without retraining
- Quality depends on time step resolution and conditioning accuracy

Proposed Improvements

-Add CLIP Based Evaluation: Incorporating CLIP would allow semantic scoring of outputs and better feedback during generation

- Use Attention Mechanisms: Adding self-attention layers like in DDPM++ could help model long-range dependencies

- Train on More Complex Datasets: Using CIFAR-10 or fashion-MNIST would test the models robustness and generalizability to real world image generation

How I would make it even better

- Train on More Realistic Images – Like clothes or animals (CIFAR-10)
- Add CLIP Evaluation – To see how "Human-like" the output is
- Use Attention Layers – These help the model focus on the important parts of the image better

Final Thoughts

This project showed how powerful diffusion models can be, by teaching a computer to destroy and then repair images, it can learn to draw from scratch. And by giving it just a label like 7 it knows exactly what to generate.

References:

Palucha, Szymon. “Understanding Openai’s Clip Model.” *Medium*, Medium, 14 Apr. 2024, medium.com/@paluchasz/understanding-openais-clip-model-6b52bade3fa3.

“Introduction to Diffusion Models for Machine Learning.” *AssemblyAI*, www.assemblyai.com/blog/diffusion-models-for-machine-learning-introduction. Accessed 28 June 2025.

Ahirwar, Kailash. “A Very Short Introduction to Diffusion Models.” *Medium*, Medium, 26 Sept. 2023, kailashahirwar.medium.com/a-very-short-introduction-to-diffusion-models-a84235e4e9ae.

“Mnist Dataset : Practical Applications Using Keras and Pytorch.” *GeeksforGeeks*, GeeksforGeeks, 31 May 2024, www.geeksforgeeks.org/machine-learning/mnist-dataset/.