SAiDL Spring Induction Assignment

Soham Ulhas Pujari
2024A7PS0490G

1 Introduction

The SAiDL Spring 2025 Induction Assignment was a great opportunity to dive deep into the core ideas behind Artificial Intelligence, Machine Learning, and Deep Learning. It wasn't just about solving problems—it was about learning how to think like a researcher and build like a practitioner.

In this report, I've shared my approach and results for the two tracks I worked on: **Core ML** and **Diffusion**. Each section goes beyond just code and numbers. I've included the thinking process, the mistakes I made, what I learned from them, and how I worked along the way to get things working.

The tasks pushed me to explore resources outside the standard curriculum—from research papers to new tools and evaluation methods. Overall, it's been a challenging but really rewarding experience that's helped shape a stronger foundation for future work in AI. .

2 Core ML

Core ML project was quite a fundamental project and worked on how we can improve predictions of a model when the data we provided had unmatching labels, which was termed as noise.

2.1 Task 1 : Making labels noisy

For this project we basically had to make functions that would convert the CIFAR-10 dataset from PyTorch and made functions to make the labels unmatch with their respective images. FYI: There are 10 classes in CIFAR-10. so basically what we did was in place of an image of horse, we put a different label and controlled the ratio at which this shold be done.

2.2 Task 2: Making loss functions

As explained in the paper we had to work on different loss functions. But initially for some idea we had to evaluate and then construct plots for accuracy between CE and NCE on noisy data at different ratios. The plot is as follows:

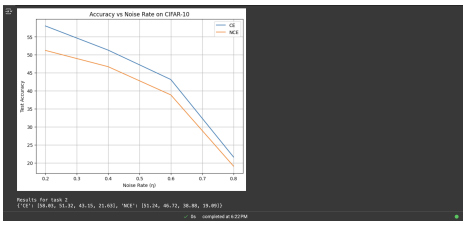

Figure 1: CE vs NCE performance

Since here the NCE has lesser accuracy than CE, it goes with our understanding that on clean test labels CE will be better. BUT as noise increases we observe that NCE doesn't fall as sharply as CE does(very slight difference in fall) showing that is more robust to noisy training data.

2.3 Task 3: Implement APL

In this part, we had to make an APL function. APL in this project was a result we got from NCE+RCE in some ratio (reverse cross entropy has the parameters of cross entropy in each other's place). After training and evaluating we got the following plot between the three, CE, NCE and APL.

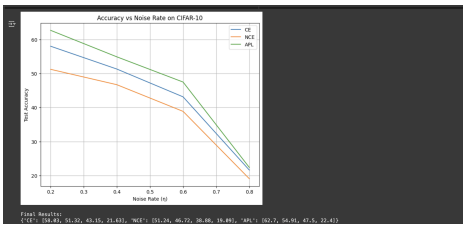

Figure 2: CE vs NCE vs APL performance

3 Diffusion

Diffusion was the second half of this project and in my experience the tougher of the two. It would have been rather impossible to complete this project without extensive help from mentors, LLMs and other provided resources. Diffusion is basically unblurring an image based on how we trained a particular model.

3.1 Task 1: Getting essence of different parameters.

1A : In this part of the project, we looked at how different values of Classifier-Free Guidance (CFG) affect the quality of generated images. Using the DiT notebook, we ran experiments with CFG set to 0 (no class conditioning) and then increased it to a high value like 10 (strong class conditioning). This allowed

us to visually compare how the images changed. CFG plays a key role in guiding the model—lower values let it generate more freely, while higher values make it stick closely to the given class label. These comparisons helped us see how class conditioning impacts both the quality and variety of the generated images.

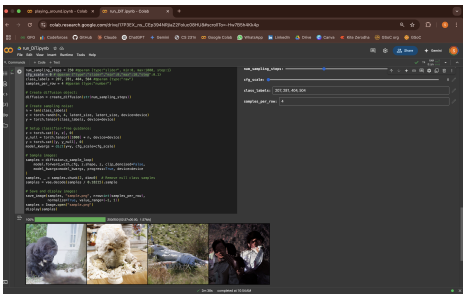

Figure 3: CFG 0

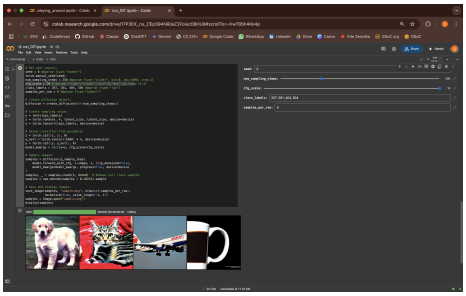

Figure 4: CFG 10

1B : Here, we explored how the number of sampling steps affects image generation. We tested values like 50, 250, and 500 to see how increasing the number of denoising steps changes the final output. With fewer steps, the model generates images faster, but they often turn out blurry or inconsistent. On the other hand, using more steps tends to produce sharper and more stable results, though it takes longer. This experiment helped us build a better understanding of the trade-off between speed and image quality in diffusion models.

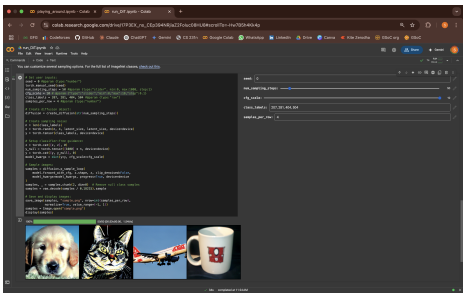

Figure 5: 50 steps

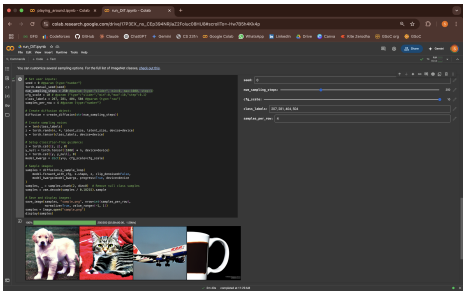

Figure 6: 250 steps

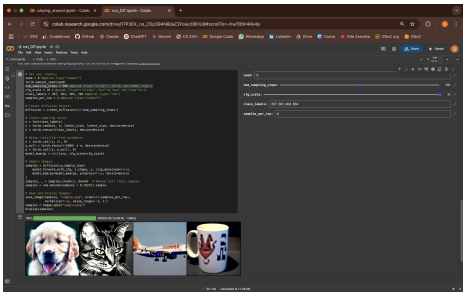

Figure 7: 500 steps

3.2 Task 2: Understanding attention

2A: In this task, the goal was to boost the efficiency of the DiT model by replacing its default attention block with the Xformers implementation. Xformers is built for better speed and memory usage, so we benchmarked how long it took to generate 50 images using both the original and updated models. The goal was to see if faster attention could noticeably improve runtime without sacrificing image quality.

PS: While the setup seemed straightforward, the results didn't quite reflect the expected improvements. There were a few quirks in the process that may have affected the outcome, so the findings here are more exploratory than conclusive. Still, it offered some valuable lessons on performance tuning and model optimization in practice.

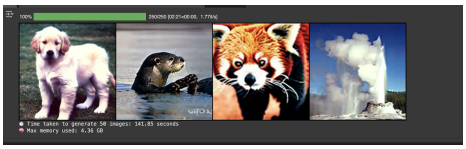

Figure 8: Before replacing default with Xformers

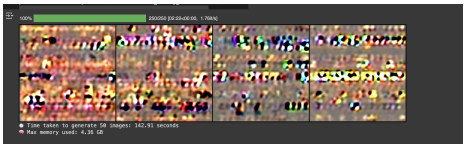

Figure 9: After replacing

2B: In this part, we trained two DiT models on a landscape datase from kaggle—one using the standard full attention mechanism, and the other using sliding window attention (SWA). SWA limits attention to smaller, local windows instead of the whole image, which makes it more efficient and scalable for larger inputs. To compare them, we looked at both the visual quality of the generated images and their FID scores. The goal was to see if SWA could cut down on compute without a noticeable drop in output quality.

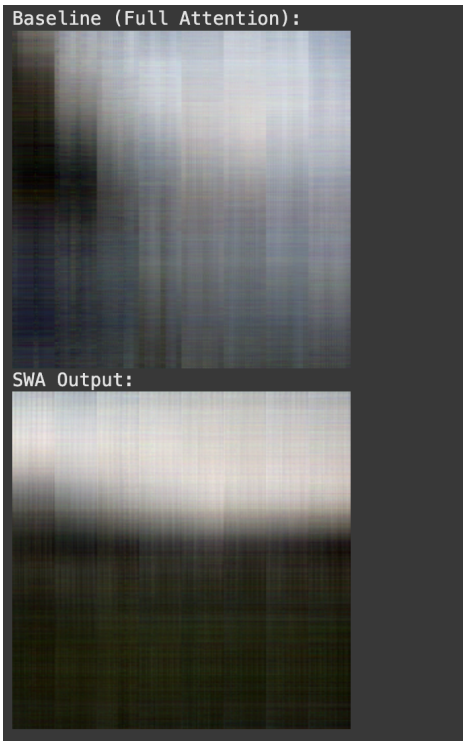

Figure 10: Standard vs SWA

3.3 Task 3: Using CMMD for evaluation

3A: In this task, we used the CLIP Mean Maximum Discrepancy (CMMD) metric to evaluate images generated by both the Full Attention and SWA versions of the DiT model. Unlike FID, which relies on Inception embeddings, CMMD uses CLIP embeddings to measure how close the real and generated distributions are in terms of high-level semantic content. This gave us a different perspective on image quality—focusing more on the meaning and structure of the images, not just how realistic they look on the surface.

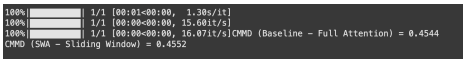

Figure 11: Comparison using CMMD

3B: Building on the CMMD setup, this task involved swapping out the CLIP model with other vision-language models like SigLIP and ALIGN. The idea was to see how different embedding models affect the CMMD scores and whether newer models might give a more reliable sense of visual-semantic quality. After

running the evaluations, we looked at how the differences in their scores reflect on the perceived quality and meaning of the generated images.

```
SigLIP CMMD (Baseline – Full Attention) = 0.4634
SigLIP CMMD (SWA – Sliding Window)      = 0.4634
ALIGN CMMD (Baseline – Full Attention) = 0.6323
ALIGN CMMD (SWA – Sliding Window)      = 0.6325
```

Figure 12: After swapping CLIP with SigLIP,ALIGN