Report on Diffusion

Sheil Maniar

April 2025

1 Introduction

Diffusion models learn to generate data by first adding noise to the data, then learning to reverse that process, allowing them to generate new samples resembling the original data distribution. A diffusion model is most commonly used for generative tasks - images, audio, video, 3D-models, etc. The model takes pure noise and iteratively removes said noise to produce the required output.Traditionally, models such as Stable Diffusion use a CNN based backbone called a U-Net. These models are very capable but are not very scalable. To remedy this the Diffusion-Transformer (DiT) was introduced by replacing the CNN based UNet with a Transformer block.

2 Running *DiT.ipynb* from DiT repo

2.1 Part (a) CFG manipulation

In the context of diffusion models, particularly in image generation with tools like Stable Diffusion, "CFG" stands for Classifier-Free Guidance scale, a parameter that controls how strongly the model adheres to the input text prompt during image generation.
For our code, CFG scale is set between 1 and 10

Class label used for this : **[84]** which stands for Peacock

CFG Manipulation (number of sampling steps is same for all - **250**)

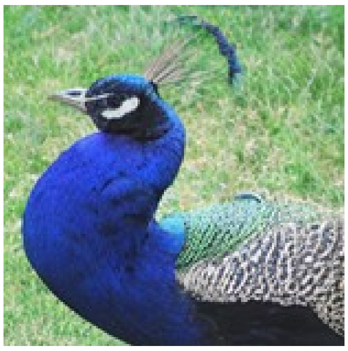


Figure 1: CFG = 1

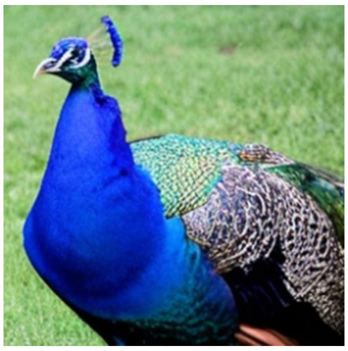


Figure 2: CFG = 4

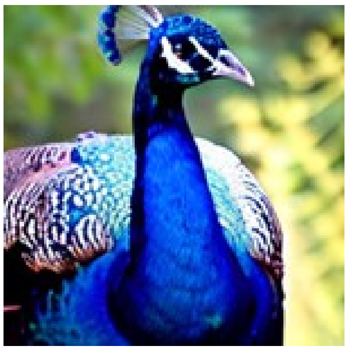


Figure 3: CFG = 10

When CFG is set to 1, we can notice how smooth and dull the picture is. It also generates "grass" which is different than our class -¿ peacock. What it means is, when CFG is low, generation contains unexpected features and doesn't strictly conform to our class. Picture is also less textured and bland. To sum it up, at low CFG, images may be creative or diverse with underfitting results
When CFG is 4, the generation somewhat adheres to it's class' image but also skirts around the boundaries of it. It can be used as a reference against the 2 extremes of CFG value - 1 and 10.
But when CFG is max (at 10) we get a sharper, more exaggerated and a vibrant peacock. This generation strongly adheres to it's class when negligible emphasis on background elements. At high CFG, image has negligible diversity and is overfitting.

2.2 Part (b) Change in number of sampling steps

In diffusion models, the number of sampling steps determines the iterative process of removing noise from a random input to generate a desired output, with more steps generally leading to higher quality, but also longer processing time. For our code, the the number of sampling steps is set between 0 and 1000

Class label used for this : **[84]** which stands for Peacock

Number of sampling steps (CFG value is same for all - **4**)

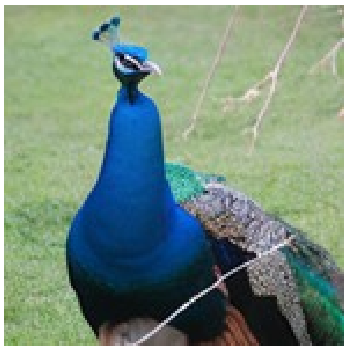


Figure 4: Sampling steps = 50

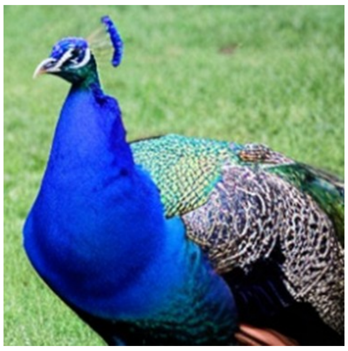


Figure 5: Sampling steps = 250

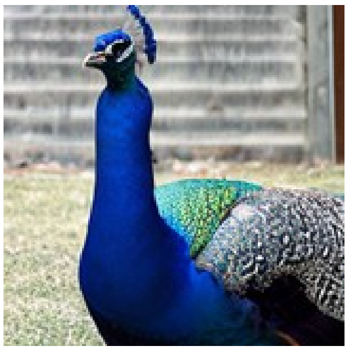


Figure 6: Sampling steps = 500

The image at 50 sampling steps has a lower image quality and has less details. Edges are not clear and has noise around them along with blurriness. Time to taken to generate this image is 8 seconds
At 250 sampling steps, we have balance between speed and quality. We get clean edges and good colour fidelity. Time taken to generate this image is 40 seconds.
At 500 sampling steps, we get a sharp image with very clean and distinct edges. We can see the difference in the level of detail in the tail feathers in the image at 50 and at 250. We get the finest details but at the cost of more time to generate images. Time taken to generate this image is 80 seconds.

3 xFormers Implementation

In the context of diffusion models and image generation, the xFormers library is a PyTorch-based library that offers optimized and composable Transformer building blocks, enabling faster and more memory-efficient image generation, within tools like Stable Diffusion.
While DiT (Diffusion Transformer) focuses on using transformers for diffusion models, xFormers is a library of optimized, modular building blocks for various transformer-based models, including those used in diffusion, offering performance improvements and flexibility.

xFormers git repo - xFormers

Our task was to replace DiT's Attention block with the xformers Attention block and measure the time taken to sample a certain amount of images

- DiT model used : DiT -B/8
- Batch Sizes : [50,100,150,200,250]
- Number of images sampled for each : 250

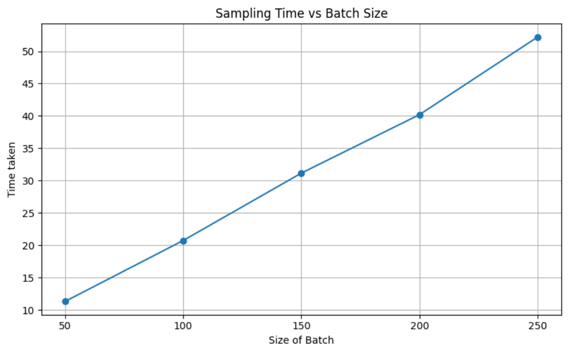


Figure 7: DiT's Attention block

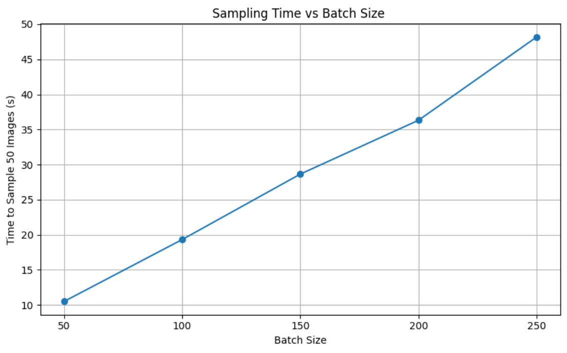


Figure 8: xFormers Attention block

Batch Size	Original Attention (s)	xFormers Attention (s)	Improvement (%)
50	11.31	10.50	7.16%
100	20.74	19.32	6.85%
150	31.16	28.65	8.05%
200	40.20	36.32	9.65%
250	52.20	48.18	7.70%

Table 1: Comparison of time taken (in seconds) to sample 200 images using Original Attention and xFormers Attention classes across different batch sizes

You can notice a marginal speedup in time to sample images while using xFormers, with improvement around 7-9
There may be other other bottlenecks, which I am not aware of, that are causing xFormers to not know it's true performance compared to the baseline model.
xFormers could be useful for large-scale diffusion tasks.

4 Sliding Window Attention (SWA)

Sliding window attention (SWA) is a technique used by transformer models to limit the attention span of each token to a fixed-size window. This leads to reduced computational complexity and a more efficient model. Imagine a narrow spotlight in a dark storeroom, where only certain objects can be focused within a small area at any given time.

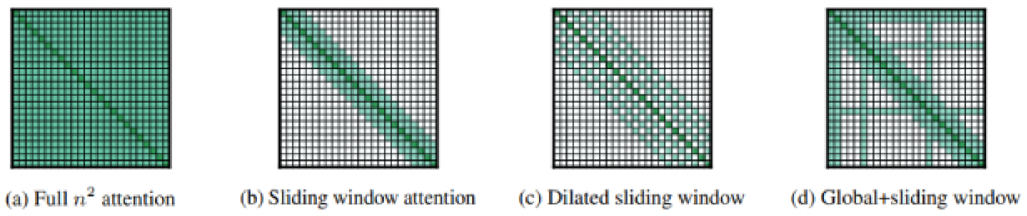

(a) Full n^2 attention
(b) Sliding window attention
(c) Dilated sliding window
(d) Global+sliding window

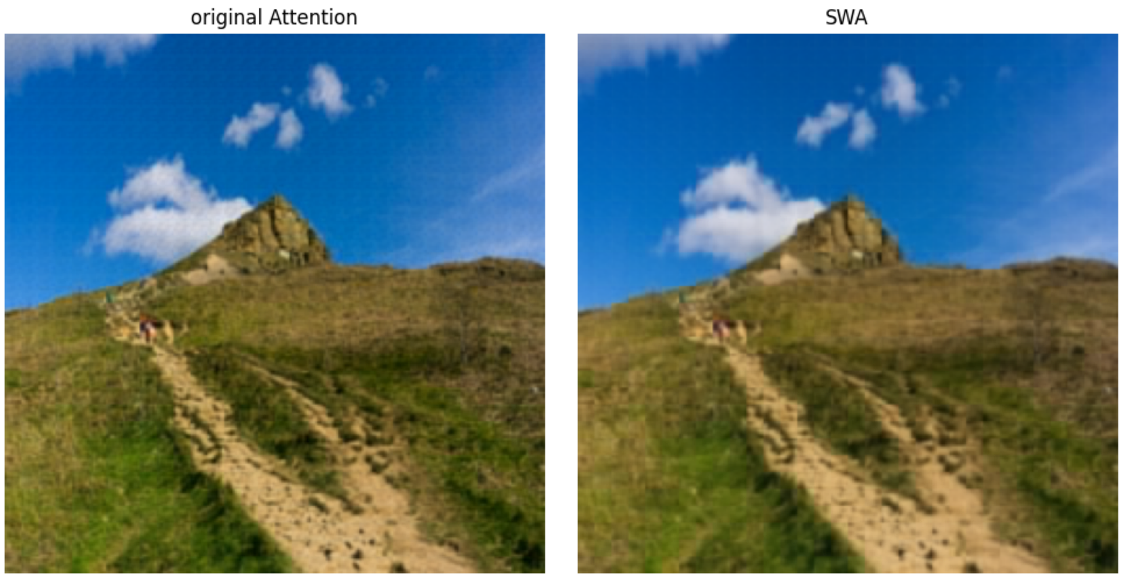
Figure 2: Comparing the full self-attention pattern and the configuration of attention patterns in our Longformer.

Our task is to train 2 DiT models on the landscape dataset without class conditioning from scratch (one with Original attention and with SWA) and compare their samples and FID.

Landscape dataset
SWA Research paper

- Model used : DiT -B/8
- Window size : 8
- Learning rate for optimizer : 0.0003 (for both)
- Epochs : 35

Sample result :

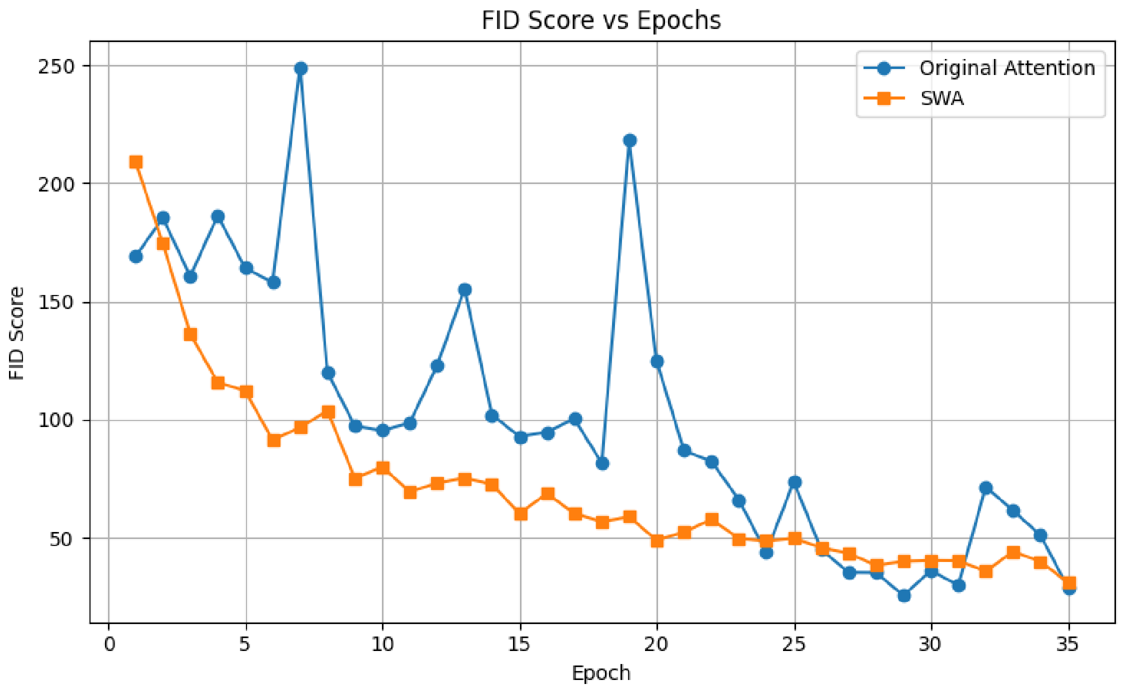


Epoch 32, Step 0, Original Loss: 0.0017
Epoch 32, Step 10, Original Loss: 0.0016
Epoch 32, Step 20, Original Loss: 0.0017
Epoch 32, Step 30, Original Loss: 0.0010
Epoch 32, Step 40, Original Loss: 0.0006

Epoch 32: FID (original) score = 71.57, FID (SWA) score = 35.89

As per the results, one can notice that images generated with original Attention block are grainy and pixelated. Square patches are visible in the entire image, that gives it a grid like feeling. The image doesn't look smooth.
Whereas, images generated using SWA are more smooth. Square patches are not visible at all. Overall, SWA improves image generation quality.

Talking about FID scores,
The Fréchet Inception Distance (FID) score is a metric used to evaluate the quality of images generated by generative models, particularly GANs, by measuring the similarity between the distributions of real and generated images in a feature space extracted by a pre-trained neural network like Inception v3. Lower FID scores indicate a better match between the generated and real images.



Original Attention's Loss function is quite janky with unwanted sudden spikes and doesn't do a good job in gradually decreasing FID scores.
Whereas in SWD, the FID score decreases steadily with no unwanted huge spikes in between. It also quickly lowers the score with wasting any more epochs, which means that it is doing a great job in refining our model and is way better at it than Original attention.

5 Conclusions

- Varying CFG and number of sampling steps changes the generated image. Manipulating it can lead us to getting our desired image for a particular label.
- xFormers Attention block is slightly faster $\sim$ (9%) than original DiT Attention block. xFormers can be implemented for large scale tasks.
- SWA is a better performer in generating images than Original DiT and also has lower FID scores for the same.