National College of Ireland

Pixelating to the Edge - Generative AI Art on Edge Devices

MSc Research Project
MSc. AI for Business

Sonal Deepak Pardesi
Student ID: 22209387

School of Computing
National College of Ireland

Supervisor: Anderson Simiscuka

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Sonal Deepak Pardesi
Student ID:	22209387
Programme:	MSc. AI in Business
Year:	2023 - 2024
Module:	Practicum 2
Supervisor:	Anderson Simiscuka
Submission Due Date:	12 August 2024
Project Title:	Pixelating to the Edge - Generative AI Art on Edge Devices
Word Count:	4876
Page Count:	17

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Sonal Deepak Pardesi
Date:	15th September 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Pixelating to the Edge - Generative AI Art on Edge Devices

Sonal Deepak Pardesi
22209387

Abstract

Generative AI is now changing the landscape of how people are getting things done. In this study we are focusing on one of the most popular application of generative AI which is text to image generation, a technology which created huge craze in 2021. Even after 3 years, it is not as effective and speedy on the edge devices like mobile as its own web version. In this research, we will compare a number of model variations and analysis what factors affect the inference speed of the image generation. Currently Mobile diffusion, although commercially not available, has claimed that it can achieve 0.02 seconds of speed. I tried to study if any architectural changes and sampling techniques can improve the inference speed while maintaining quality of image. The changes in the sampling operations like changing the scheduler from PNDMS to DDIM gave a 6.57 percent increase in inference speed with a bit of degradation in FID and CLIP score. The architectural changes gave significant improvement of upto 15.24 percent increase and good results with FID and CLIP score. This research will explore the much needed generative AI technology's requirement on edge devices.
Keywords: Generative AI, GANs, text-to-image

1 Introduction

Images are not just an important piece of memory, from early childhood we can see images being a very important part of the education of a child it helps them to create a connection with the object and the language, for young and teenagers diagrams and schematics are one of the most effective ways of learning, understanding and explaining objects. Even in our adult lives diagrams, images and scemhatics are used in almost every aspect of our lives right from creating an interesting invitation card, adding images to presentations or lectures for better explainability, uploading an interesting banner for our social media profiles, creating a POC for business pitches or just to have a unique wallpaper to look at, using images is more integral part of our lives than we think. Seeing this opportunity in 2015 a new technology emerged know as automated image captioning, here a machine is expected to learn and understand different objects present in a picture and understand the relationship between them and then write a human like caption or description for them(Chohan et al.; 2020)

This technology was great and made researchers curious what if the process was reversed. Creating high resolution natural looking images from their textual descriptions had two major components to take care of, one is language modelling and another is image generation where sequential deep learning techniques were used to build a conditional

probablistic model(Mansimov et al.; 2015). While is was not new to train the machine on a model for specific images for example if you require an image on landscapes you can train the model on tons of lanscape images and then the model would be able to generate a landscape image when prompted but here the model was restricted to one particular style. The need of the hour was to input any textual description and get an image in return, an in January 2021 OpenAI anounced DALL E followed by a version upgrade ie. DALL E 2 in April 2022. OpenAI democratized the AI capabilities in terms of image generation empowering researchers and artist to leverage state of the art language and image model(Vayadande et al.; 2023). At present companies like OpenAI, Stability AI, Midjourney, Ranway ML etc are making progress in this technology and rolling out new advancements at a greater pace. However this text to image generation process requires billions of parameters and hence a huge amount of processing and computing resource and hence it is not possible to process this as an application on an edge device like a mobile phone.
In 2023 Google released a research paper on it s own model Mobile Diffusion which is similar to stable diffusion but with lesser parameters so that we can run this on an edge device like mobile. The inference speed for the model was a striking sub second speed for a 512x512 image(Zhao et al.; 2023).

In this research I will try to examine what factors can affect/improve the optimization of inference efficiency in text-to-image diffusion models to support its operations on an edge device like laptop?
The focus will be on two parameters 1. Architecture 2. Sampling Efficiency. The research will compare different modification on a self created image generator to examine the factors affecting the inference speed.

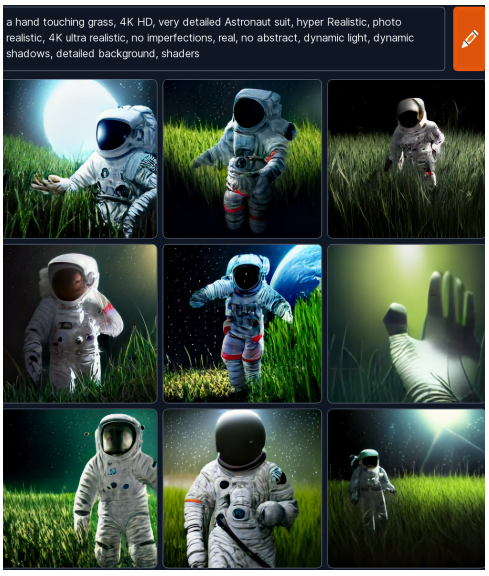

Figure 1: Unique and imaginative outputs from DALL E. (Face; 2022)

2 Literature Review

A few companies today dominates the field of text-to-image generation, generative AI has truly elevated its worth with major advancements made by companies such as OpenAI (DALL-E), Stability AI, and Midjourney. Artist across the world are fascinated and worried at the same time experimenting with the result and quality of the art work generated by this technology which is based on GANs and style transfer technologies that are able to interpret textual descriptions into artwork using complex models. But running such complex models on mobile / edge devices is still a problem because of latency issues, not to mention the processing power limitation. Since mobile is an integral part of human lives now and it is the fastest way for any technology to propagate, it's important to bring text to image to edge devices. In this paper, we discussed the architectural enhancements and performance boosting measures required to open the possibility of text-to-image generation at edge devices.

It has lot of potential, especially text to image generation using Generative AI for art and many practical applications. The deployment of these models on edge devices is hindered by problems to be solved at the architectural design and efficiency levels. Over time, as research improves such approaches, we are likely to see more advanced generative AI models with control synthesis, providing new ways of manipulating model behavior and making powerful AIs like GPT-3 available everywhere.

2.1 Generative Models and Their Evolution

The essence of generative models lies in the fact that they can reproduce similar results as the training data that they were trained on in the first phase. There are six types of generative models, namely GANs(Generative Adversarial Networks), VAEs(Variational Autoencoders), Autoregressive Models, Flow Based Models, Energy Based Models and Diffusion Models(Harshvardhan et al.; 2020). While each model has its own application and importance, the understanding of GANs and VAEs is important to understand the evolution of the modern diffusion models used in today's text to image generators.

GAN or Generative Adversarial Networks works on two main models one is a generator and other is a discriminator, while generator is a Decoupled Convolutional Neural Network(DCN) the discriminator is a Convolutional Neural Network (CNN). The generator is feed by a fixed dimensional noise vector or latent variables and it produces fake images which is then mixed with the real images from the dataset. This output is now fed to the discriminator which classifies whether it is a fake or real image. The classification accuracy is fed to both generator and discriminator, the objective of generator is to increase the classification error and that of the discriminator is to decrease classification error(Mi et al.; 2018).

VAEs or Variational Auto Encoders are another important generatie model which has three main components the encoder, a latent space and a decoder. The encoder is given a high dimensional input which is then compressed and fed into the latent space. The decoder access this code from the latent space and decodes the data as correctly as possible. The main focus here is to reduce noise in the data, recognise relevant features and detect anamolies(Kos et al.; 2018).

As we have a better understanding of both GANs and VAEs understanding the dif-

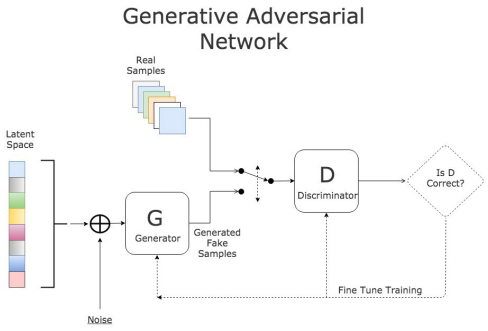

Figure 2: GAN Model.(KDnuggets; 2017)

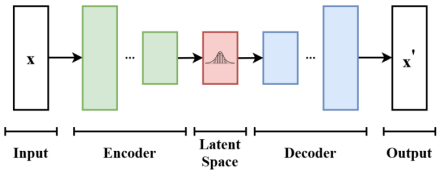

Figure 3: VAE Model. (Wikipedia; 2021)

sussion model which is used in almost every text to image generator application today will be much easier. In a diffsuion model we have two processes first we have a pre defined forward process which maps data distribution into a gaussian distribution. Then we have a reverse process which uses trained neural network to reverse the effects of forward process by using ordinary or stochastic differential equations (ODE/SDE), all this happens in latent space and the text encoder conditions the mapping in forward process(Cao et al.; 2024). Diffusion models are highly efficient in producing high quality photorealistic images, they bypass issues like mode collapse, provide high stability, works great on different data types and can be conditioned for various use cases.

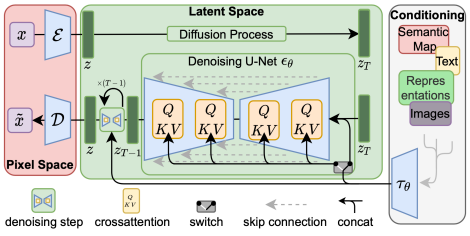

Figure 4: Diffusion Model.(Weng; 2021)

2.2 Transformer Architectures in Image Generation

It is an attention based alternative to recurrent networks used for various NLP tasks but mainly in text to image use cases. The transformer architecture consists 6 key components starting with the self attention mechanism which provides model with weighs of different

words in a sentence. Then we have positional encoding which provides the position of the tokens in the sequence. The concept of multi head attention provides simultaneous application of several attention mechanism in parallel. After this layer the data goes through the feed forward network which produces various encoded representations along with this is the encoder decoder structure(Vaswani et al.; 2017).
Transformer architecture is used in various applications ranging from NLP tasks such as language modelling and text summarizing. There are vision transformers (ViT) which are used for image classification purposes.

2.3 Enhancing Efficiency for Edge Devices

The main difficulty in implementing text to image models on an edge device like mobile is the requirement of high computational efficiency and limited resources on edge devices. To combat these problems, approaches such as model distillation and diffusion GAN fine-tuning have been proposed. Model distillation is when we train a smaller model using that huge model as the target of training, so essentially it will have an equivalent performance as that long model but not at all in complexity terms. There are two main factors, architecture and sampling, which are affecting the inference speed for such model and in this research we will also examine.

2.4 Factors to consider for inference efficiency

The two most important factors affecting inference efficiency are architecture and sampling efficiency. The method proposed in MobileDiffusion paper further simplifies the model by replacing a U-Net architecture with separable convolutions to significantly reduce computational requirements while still providing high-quality images. More over, by refactoring transformer blocks into more lower resolution ones and simplifying activation functions (GELU for Swish in our case)it expedite numerical stability and decrease computational cost towards mobile device applications

Beyond architectural progress, sampling efficiency is crucial. Approaches like progressive distillation have drastically cut down sampling steps from hundreds to a few, hence making it possible for real-time inference on mobile. This enormously enhances the efficiency of sampling, even allowing for one-step generation in combinatorial conjunction with diffusion-based GAN models (e.g., UFOgen). Together, these strategies make it possible to produce high-quality images in sub-second times thus demonstrating the feasibility of deploying advanced text-to-image models on resources-constrained devices such as smartphones.

These improvements are making text-to-image models more deployable on mobile platforms, as well as leading to new possibilities for real-time applications that will greatly enrich user experience and commoditize the application of generative models in common scenarios. Ongoing research in this direction is to refine these methods more, so that the balance should be of computational efficiency and generating quality images at realtime frame rate so as to enable for broader usage and accessibility out-of-the-box advanced AI models on a mobile phone.

3 Methodology

The research uses a pre-trained Stable Diffusion model to create high-quality images from a simple text-based prompt provided by the user. Our methodology aims to take advantage of the cutting-edge diffusion models' capabilities, and specifically the CompVis/stable-diffusion-v1-4 variant, which was trained on vast datasets of images and text annotations. The steps consists of downloading and installing the needed libraries, loading the pre-trained model, receiving user prompts via an interactive interface, and generating an image based on these prompts. Thus, there is no need to take additional measures to collect and preprocess data as the pre-trained model is already instructed on how to interpret the text and produce the desired results in an image form. Additionally, we use mixed precision and accelerate the process with the GPU to ensure faster image generation. The following sections describe the methodology in detail.

3.1 Installation of Necessary Libraries

To implement text to image model we need to install some specific libraries to run the model smoothly. Let us understand each library in detail and how it is important for this research.

3.1.1 Pytorch/Torch

Pytorch or torch is an open spurce library specifically used for deep learning purposes. It was developed by Facebook's AI research lab and it comprises a powerful collection to support tensor computation with GPU acceleration. It provides automatic differentiation and is mainly used for application like computer vision and NLP models(Paszke et al.; 2019). It provides great flexibility in working on models which uses stable diffusion.

3.1.2 Diffusers

It is one of the important libraries when working with text to image generation model as it iteratively tranforms a simple noise distribution into a complex data distribution. It also has pre trained libraries and APIs for inference(Bengesi et al.; 2024).

3.1.3 Pillow

Its is open source imaging library by python which helps with opening, manipulating and saving many different images with different file formats. Functions like resizing, cropping, rotating etc can be done using this library(Villán; 2019).

3.2 Loading the Model

A pre trained stable diffusion model was used for this research from the hugging face model hub. To increase the computation speed and reducing memory usage mixed precision(fp16) was used in the model.

3.3 Creating Input Widget

The model works on the data or input provided by the user hence the following code was used to generate an input widget as a part of data collection process.

```
import ipywidgets as widgets
from IPython.display import display

# Create a text input widget for the prompt
prompt_input = widgets.Text(
    value='A scenic landscape with mountains and a lake',
    placeholder='Type something',
    description='Prompt:',
    disabled=False
)
display(prompt_input)
```

Figure 5: Code used to create input widget

3.4 Generating Images

To obtained the image generated through the text prompt the function 'generate image' is used, this function also helps save the image generated to a file. To faciliate the generation process a button is also created.

The text-to-image generation model was implemented using the Stable Diffusion model from Hugging Face 'diffusers' Library. Once the model is loaded it can be executed using a python environment including import dependencies like 'torch', 'diffusers', and optional requirements for interactive user input as image etc., as illustrated here by 'pillow' (PIL) which lacks options of opening with GUI-input, adjusted according to widget on Jupyter-plots ('ipywidgets'). It allows users to input text prompts via an easy-to-use widget interface which then causes the model to produce matching images. The saved images are stored in a folder to be analyzed later. The performance of the model is evaluated using three key metrics Fréchet Inception Distance (FID), CLIP score and inference time.

In the final step of this implementation, images will be successfully output based on text using a Stable Diffusion pre trained model. It outputs transformed image data using prompts that the user has given to it.

Outputs Produced:
Derived Data: Text to high-quality image where the text is given by users.
Written Code: A pipeline for text-to-image generation, programmed in Python.
Model Trained: The Stable Diffusion model (ie, CompVis/stable-diffusion-v1-4)
Used Tools: Text input widgets for user interaction and prompt input
Tools and Languages Used:
Programming Languages: Python
Libraries:
We use pytorch for tensor manipulations and gpu acceleration.
PIL : for image(preprocessing) operations

4 Implementation

The final stage of implementation was based on a step-by-step breakdown of the problem statement and then working on specific areas. The research uses a stable diffusion model,

hence there was no need for any external data, however, I first started with installation of certain essential libraries. In the next step, we loaded some pre trained models which includes the base model of this research - Stable Diffusion and FID and CLIP models for evaluation purposes. I then performed an intial evaluation of the model to make sure that we can set a reference point for further results to compare the percentage improvement. The next step involved the improvement in the model to obtain better inference time, FID and CLIP score. The changes in the model provided with some improvement but it was also seen that the output differed with every new prompt, also there was not a significant change in the FID and CLIP score. Hence there on the focus was to improve the inference time. To achieve this changes in the timesteps and the scheduler where made and that provided with improved inference time. I also did profiling on the model to identify any bottlenecks and the following table shows the results.

aten::convolution	0.69	47.458ms	15.55	1.065s	211.585us	0.000us	0	4.150s	824.223us	5035
aten::conv2d	0.49	33.438ms	16.02	1.097s	217.949us	0.000us	0	4.140s	822.286us	5035
aten::cudnn_convolution	6.75	462.638ms	10.29	705.347ms	140.089us	3.524s	45.35	3.935s	781.549us	5035
aten::scaled_dot_product_attention	0.45	30.603ms	3.49	239.254ms	146.512us	0.000us	0	1.860s	1.139ms	1633
aten::_scaled_dot_product_efficient_attention	0.51	34.968ms	3.05	208.651ms	127.772us	0.000us	0	1.860s	1.139ms	1633
aten::_efficient_attention_forward	0.82	56.045ms	1.96	134.174ms	82.164us	1.834s	23.61	1.860s	1.139ms	1633
fmha_cutlassF_f16_aligned_64x64_rf_sm75(PyTorchMemEf...)	0	0.000us	0	0.000us	0.000us	1.563s	20.11	1.563s	3.064ms	510
void cudnn::ops::nchwToNhwcKernel¡__half, __half, float¿	0	0.000us	0	0.000us	0.000us	1.239s	15.95	1.239s	218.332us	5676
aten::linear	1.51	103.276ms	15.72	1.077s	111.328us	0.000us	0	1.162s	120.092us	9677

Table 1: Profiling of the Model showing resource usage

The profiling shows that there potential bottlenecks and concerns in the model, while the attention layers consume time but the main concern are the convolution operations and they are main bottlenecks in CPU and CUDA time. To improve the model I used a custom UNet architecture were depthwise separable convolutions were used, this helps in reducing number of parameters and computation cost as compared to standard convolutions.
In the research to improve the sampling efficency I also incorporated DDIM (Denoising Diffuser Implicit Models) with as low as 5 timesteps and it provided improved results. To improve the inference speed I also used mixed precision which provided better computational efficiency, reduced memory usage and most suitable for large models with big data. Below is a summary of the entire implementation.

4.1 Libraries

The important libraries used for the model includes diffuser, transformer, torch, torch vision, scipy and ipywidget.

4.2 Model

A number of pre trained models were used in the research to ensure efficiency of the code and availability of authentic and huge dataset. The models include stable diffusion for the execution and creation of the application, CLIP model, FID model for evaluation.

4.3 Profiling

This was done to identify any bottlenecks and improve them with changes in the code as mentioned in the next step.

4.4 Modification in the Model

There were 3 main changes made in the model the first was to use mixed precision. Second was to use the DDIM scheduler, here I started with 50 time steps and later in the model reduced it to as low as 5 time step using which improved the sampling efficiency. The last change was to use a custom UNet architecture with depthwise separable convolutions and using swish activation instead of relu.

5 Evaluation

We focus our main evaluations of text-to-image generation models on three key metrics: inference time, the Fréchet Inception Distance (FID), and CLIP score. Each of these parameters provides a different insight into the ability and effectiveness of the model. Through measuring these metrics we try to determine whether the model is good enough in all those aspects which help us make it a better real-world application.
Inference Time: The metric here represents the time it takes for the model to synthesize an image based on a text prompt given into it. In practical scenarios such as a production environment, where real-time or near-real time inferences are desired.
Quicker inference times mean that it is friendlier for users and also more cost-effective in terms of deployment: be it set dressing an interactive art piece, generating content against a deadline or running on a VR installation. By means of benchmarking the inference time, we can determine if how efficient our model is and areas for possible improvements.
What is FID (Fréchet Inception Distance) : The Fréchet Inception distance measures the 'distance' between two or more features of images. This approach measures the distance in feature vector space between images generated by generators and real images and compares it to benchmarks on a large-scale image recognition task. Lower FID scores mean that the images are more similar real images combining quality and diversity.
Therefore, assessing the FID score helps us to measure if our model can produce high fidelity content that is same as natural images which are crucial for many possible applications of digital image production in sectors including e.g. art and media entertainment(Chong and Forsyth; 2020).
CLIP Score: A metric that shows how well generated images match their textual prompts. We use the CLIP (Contrastive Language-Image Pretraining) model to compute cosine similarity between embedding of texts and images. The reason a higher CLIP score is better because that shows how accurate the model can convey the text descriptions into image(Liu et al.; 2021).
This metric is useful for things like visual storytelling, educational content creators and assistive tech applications that provide a text-to-image correspondence. We start with sampling a bunch of images from different text prompts using the Stable Diffusion model in our evaluation process. After that for each image generation, we test the generation time (inference speed) to analyze its speed. The images are generated, and then compared to a dataset of real images in order to determine the FID score which is an indicator for how far away from training data distribution our generator output lies regarding image quality and diversity.
We also report the CLIP score to test how well images are generated with corresponding texts. Through examining these three measurements, the goal is to provide a complete assessment of model performance and mainly identify what its strengths are as well areas

that can be built upon in order for it have additional applications within real life scenarios.

5.1 Image Generation / Case Study 1

First we tried to analyze how accurately the model is able to generate images depending on the give promtps. Below are some results of images generated on user prompts.

Figure 6: Fig 1. A candid photo of a Labrador enjoying breeze in a garden, 4k, high resolution
Fig 2. A scenic landscape with mountains and a lake
Fig 3. Diljit Dosanj in a Van Gogh Painting
Fig 4. A Backpack on a mountain trail
Fig 5. A baby as a DJ on Mars
Fig 6. A bunch of mangoes in geometric bowl

5.2 Initial model / Case Study 2

The initial model is based on stable diffusion CompVis v1-4 with a CLIPImageProcessor for feature extraction. In the initial model I also used PNDMS (Pseudo Numerical Method for Diffusion Model) Scheduler for sampling efficiency. for the choice of architecture UNet2DConditionModel was used from the UNet architecture and AutoencoderKL for VAE.

Initial Model
This is the first step performed to create a benchmark for the rest of the image generation inference time and other evaluation parameters. The values obtained from this shows

inference time as 9.26 seconds, FID score as 11036.33 and CLIP score as 29.23. For the initial model all the 3 parameters were not so satisfactory and hence changes were made to the model.

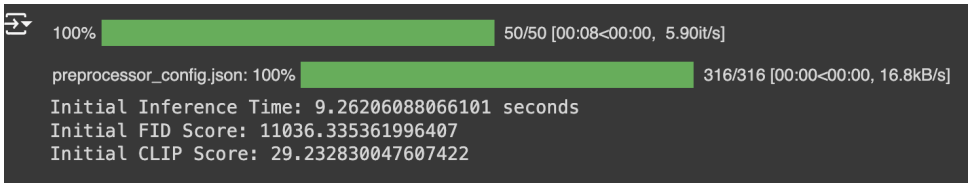

Figure 7: Results of the initial execution of model

Modified Model

In the modified model I first changed the scheduler from PNDMS to DDIM with initial timesteps of 50 which was later reduce to 25, also incorporated mixed precision. the first result obtained shows around 6 percent reduction in inference time, however the FID and CLIP score were greatly affected.

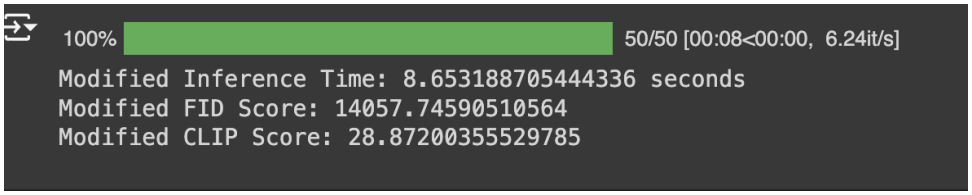

Figure 8: Results from the Modified Model showing improvement in inference time but degradation in FID and CLIP score

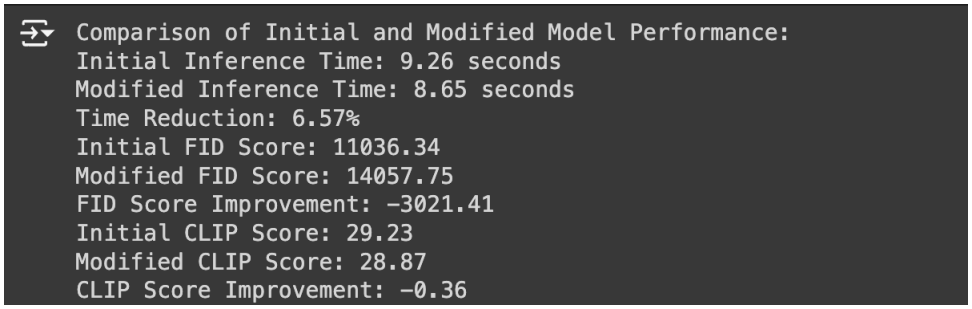

Figure 9: Comparision of the outputs

5.3 Sampling Efficiency / Case Study 3

Initial Model

In the previous result although we were achieving satisfactory result for inference time the quality of the image was being affected. To solve this I kept the use of mixed precision as it is and reduced the timestep to as low as 5. Use of DDIM ensured that the quality

```
Modified Inference Time: 8.313452959060669 seconds
Modified FID Score: 8919.541743755424
Modified CLIP Score: 29.658824920654297
Comparison of Initial and Modified Model Performance:
Initial Inference Time: 9.26 seconds
Modified Inference Time: 8.31 seconds
Time Reduction: 10.22%
Initial FID Score: 11036.34
Modified FID Score: 8919.54
FID Score Improvement: 2116.80
Initial CLIP Score: 29.23
Modified CLIP Score: 29.66
CLIP Score Improvement: 0.43
```

Figure 10: Comparision of Modified Model with Initial Model

of the image is maintained and reducing time steps made the generation of image faster.

Modified Model
It can be observed that we got 10 percent improvement in the inference time and not only that but the FID also improved greatly with a score of 8919 from 11036. The CLIP score remained fairly unchanged.

5.4 Output with User Prompt / Case Study N

In this section we will see some images generated from user prompts and what was striking is that for each image generated the inference time was difference. While this depends on the complexity of the prompt as well, I also tried with time step 3. Below are some of the images generated.

User Promt Images
The first image is generated with user prompt "monkey as DJ" and it shows 11 percent improvement in inference speed. But the FID score was affected and CLIP score remained almost same. To correct this and maintain the inference speed last modification was made in the architecture discussed in next section.

Figure 11: Image Generated by User Prompt " Monkey as DJ"

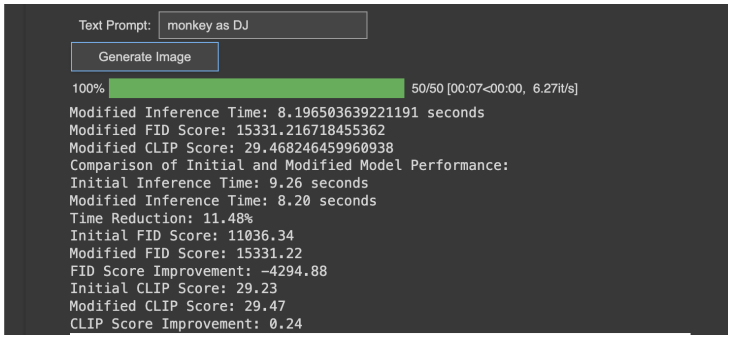

Figure 12: Results of the above generated image

Another image generated was "scarecrow with pink hoodie" for this image a 14 percent improvement in inference speed was observed with satisfactory FID and CLIP score. Below is the image generated and the results obtained.

Figure 13: Image generated with prompt "scarecrow with pink hoodie"

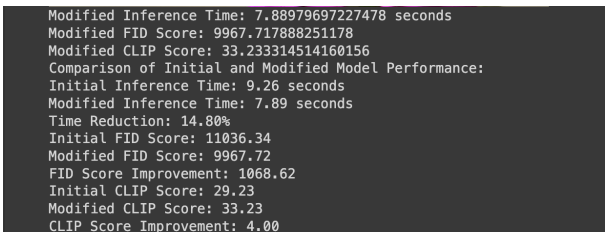

Figure 14: Comparision of result - initial model vs current output

Results After UNet Changes

In the final modification I changed the UNet architecture for more accurate image and hence used the depthwise seperable convolution and changed the activation from relu to swish. The results not only improved but the images were better as well. We can see the inference speed was increased from 11 percent to 14 percent for the "monkey as DJ" image and similarly the for " scarecrow with pink hoodie" the inference speed increased from 14 percent to 15 percent with improved image.

Figure 15: Image generated after UNet modification for "monkey as DJ"

```
Modified Inference Time: 7.899375677108765 seconds
Modified FID Score: 7510.65845794967
Modified CLIP Score: 37.1183052629883
Comparison of Initial and Modified Model Performance:
Initial Inference Time: 9.26 seconds
Modified Inference Time: 7.90 seconds
Time Reduction: 14.69%
Initial FID Score: 11036.34
Modified FID Score: 7510.66
FID Score Improvement: 3525.68
Initial CLIP Score: 29.23
Modified CLIP Score: 37.12
CLIP Score Improvement: 7.89
```

Figure 16: Improved Results for above image

Figure 17: Improved image for "scarecrow with hoodie"

```
Modified Inference Time: 7.848771333694458 seconds
Modified FID Score: 9710.651087720886
Modified CLIP Score: 32.67625427246094
Comparison of Initial and Modified Model Performance:
Initial Inference Time: 9.26 seconds
Modified Inference Time: 7.85 seconds
Time Reduction: 15.24%
Initial FID Score: 11036.34
Modified FID Score: 9710.65
FID Score Improvement: 1325.69
Initial CLIP Score: 29.23
Modified CLIP Score: 32.68
CLIP Score Improvement: 3.45
```

Figure 18: Improved results for "scarecrow with hoodie"

Sampling Efficiency Change	Inference Speed (s)	Speed Improvement (%)	FID Score	FID Score Change	CLIP Score	CLIP Score Change
Benchmark	9.26	N/A	11036	N/A	29.23	N/A
PDMS to DDIM	8.65	6.57	14057	-3021.41	28	-0.36
Mixed Precision & Reduced Timestep (5)	8.31	10.22	8919.54	2116.80	29.66	0.43

Table 2: Results obtained after applying sampling changes.

Comparison Metric	Benchmark	Monkey as DJ - Std UNET	Monkey as DJ - Custom UNET	Scarecrow with Pink Hoodie - Std UNET	Scarecrow with Pink Hoodie - Custom UNET
Inference Speed (s)	9.26	8.2	7.9	7.89	7.85
Speed Improvement (%)	N/A	11.48%	14.69%	14.8%	15.24%
FID Score	11036	15331.22	7510.66	9967.72	9710.65
FID Score Change	N/A	-4294.26	3525.68	1068.62	1325.69
CLIP Score	29.23	29.47	37.12	33.23	32.68
CLIP Score Change	N/A	0.24	7.89	4	3.45

Table 3: Results obtained after making architectural changes.

5.5 Discussion

The results of our tests were examined, and a number of important insights about the capabilities and drawbacks of text-to-image generation on edge devices were discovered. Particularly when dealing with challenging and abstract prompts, our initial model had difficulty with both image quality and inference speed. Changes, like changing to a DDIM scheduler and utilizing blended accuracy, further developed inference speed yet at first debased picture quality. The utilization of depthwise separable convolutions and an optimized UNet architecture were two additional refinements that successfully achieved a balance between speed and quality, leading to significant enhancements in both metrics. Despite these advancements, the model's performance varied depending on prompt complexity, highlighting the need for ongoing refinement and the possibility of incorporating more advanced architectural enhancements and training methods to better align with current models. While these results depict improvement, they also highlight the particular difficulties associated with optimizing generative AI for edge device environments.

6 Conclusion and Future Work

The goal of this study was to make text-to-image generation on edge devices more efficient while also improving image quality and inference speed. The targets were to look at changed engineering and testing effectiveness techniques and propose upgrades for versatile organizations. We carried out and changed a standard diffusion model to depthwise separable convolutions, DDIM scheduler, blended accuracy, and improved UNet design. Although performance varied depending on prompt complexity, these modifications resulted in significant enhancements to inference speed and image quality depicted by results which included inference speed comparisons and FID and CLIP score. The feasibility of implementing generative AI on edge devices was demonstrated by the significant time savings in inference and improved image fidelity. Although there are still limitations, such as variability in the generation of complex images, the implications of the research point in the right direction for real-time applications. Future work ought to investigate further developed attention layer alterations and various architectures to additionally enhance execution. Also, incorporating adaptive learning techniques and real world testing could upgrade results. Financially, these headways could create functional, easy to use generative artificial intelligence applications on edge devices, growing availability and utility in different imaginative and proficient fields.

References

Bengesi, S., El-Sayed, H., Sarker, M. K., Houkpati, Y., Irungu, J. and Oladunni, T. (2024). Advancements in generative ai: A comprehensive review of gans, gpt, autoencoders, diffusion model, and transformers., *IEEE Access* .

Cao, H., Tan, C., Gao, Z., Xu, Y., Chen, G., Heng, P.-A. and Li, S. Z. (2024). A survey on generative diffusion models, *IEEE Transactions on Knowledge and Data Engineering* .

Chohan, M., Khan, A., Mahar, M. S., Hassan, S., Ghafoor, A. and Khan, M. (2020). Image captioning using deep learning: A systematic, *image* **11**(5).

Chong, M. J. and Forsyth, D. (2020). Effectively unbiased fid and inception score and where to find them, *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6070–6079.

Face, H. (2022). Hugging face - the ai community building the future. Accessed: 2022-XX-XX.
URL: *https://huggingface.co*

Harshvardhan, G., Gourisaria, M. K., Pandey, M. and Rautaray, S. S. (2020). A comprehensive survey and analysis of generative models in machine learning, *Computer Science Review* **38**: 100285.

KDnuggets (2017). Kdnuggets - analytics, data science, and machine learning. Accessed: 2017-XX-XX.
URL: *https://www.kdnuggets.com*

Kos, J., Fischer, I. and Song, D. (2018). Adversarial examples for generative models, *2018 ieee security and privacy workshops (spw)*, IEEE, pp. 36–42.

Liu, X., Gong, C., Wu, L., Zhang, S., Su, H. and Liu, Q. (2021). Fusedream: Training-free text-to-image generation with improved clip+gan space optimization.
URL: *https://arxiv.org/abs/2112.01573*

Mansimov, E., Parisotto, E., Ba, J. L. and Salakhutdinov, R. (2015). Generating images from captions with attention, *arXiv preprint arXiv:1511.02793* .

Mi, L., Shen, M. and Zhang, J. (2018). A probe towards understanding gan and vae models, *arXiv preprint arXiv:1812.05676* .

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L. et al. (2019). Pytorch: An imperative style, high-performance deep learning library, *Advances in neural information processing systems* **32**.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I. (2017). Attention is all you need, *Advances in neural information processing systems* **30**.

Vayadande, K., Bhemde, S., Rajguru, V., Ugile, P., Lade, R. and Raut, N. (2023). Ai-based image generator web application using openai's dall-e system, *2023 International Conference on Recent Advances in Science and Engineering Technology (ICRASET)*, IEEE, pp. 1–5.

Villán, A. F. (2019). *Mastering OpenCV 4 with Python: a practical guide covering topics from image processing, augmented reality to deep learning with OpenCV 4 and Python 3.7*, Packt Publishing Ltd.

Weng, L. (2021). From gan to wgan. Accessed: 2021-XX-XX.
URL: *https://lilianweng.github.io/lil-log/2021/05/26/gan-and-wgan.html*

Wikipedia (2021). Wikipedia, the free encyclopedia. Accessed: 2021-XX-XX.
URL: *https://www.wikipedia.org*

Zhao, Y., Xu, Y., Xiao, Z. and Hou, T. (2023). Mobilediffusion: Subsecond text-to-image generation on mobile devices, *arXiv preprint arXiv:2311.16567* .