Pixelating to the Edge: Generative AI Art on Edge Devices

Sonal Deepak Pardesi
National College of Ireland
Dublin, Ireland
x22209387@student.ncirl.ie

Cristina Hava Muntean
National College of Ireland
Dublin, Ireland
cristina.muntean@ncirl.ie

Anderson Augusto Simiscuka
Dublin City University
Dublin, Ireland
andersonaugusto.simiscuka@dcu.ie

***Abstract*—Generative AI is transforming the way people accomplish tasks, reshaping numerous industries and workflows. In this study, we focus on one of the most popular applications of generative AI: text-to-image generation, a technology that gained significant attention in 2021. Despite its popularity, even after three years, text-to-image generation remains less efficient and slower on edge devices like mobile phones compared to its web-based counterparts. This research investigates various model variations and analyzes the factors influencing inference speed in image generation. While Mobile Diffusion, though not yet commercially available, claims to achieve an impressive inference speed of 0.02 seconds, we explore whether architectural modifications and sampling techniques can further enhance performance without compromising image quality. Our findings indicate that adjustments to sampling operations, such as switching the scheduler from PNDMS to DDIM, resulted in a 6.57% increase in inference speed, albeit with a slight degradation in FID and CLIP scores. In contrast, architectural changes yielded significant improvements, achieving up to a 15.24% increase in speed while maintaining favorable results in FID and CLIP scores.**

***Index Terms*—Generative AI, GANs, text-to-image**

I. Introduction

In 2015, automated image captioning emerged, enabling machines to learn and understand objects in a picture, identify relationships between them, and generate human-like captions [1]. This breakthrough inspired researchers to reverse the process, creating high-resolution, natural-looking images from textual descriptions. This task required addressing two key components: language modeling and image generation, which leveraged sequential deep learning to build a conditional probabilistic model [2]. While earlier models could generate specific styles of images, such as landscapes, they were restricted to predefined styles.

The need arose for a model capable of generating diverse images from any textual description. OpenAI addressed this by releasing DALL·E in January 2021, followed by DALL·E 2 in April 2022, enabling advanced image generation technology for researchers and artists [3]. Currently, companies like OpenAI, Stability AI, Midjourney, and Runway ML are advancing this technology rapidly. However, text-to-image generation involves billions of parameters and demands extensive processing power, making it challenging to run on edge devices like mobile phones.

This paper explores the factors influencing the optimization of inference efficiency in text-to-image diffusion models to enable their operation on edge devices such as laptops. The paper is structured as follows: Section II reviews related works, Section III outlines the methodology, Section IV presents implementation details, Section V discusses evaluation and results, and Section VI concludes the paper and presents future research directions.

II. Related Works

A few companies dominate the field of text-to-image generation, and generative AI has significantly advanced, with major contributions from companies such as OpenAI (DALL-E), Stability AI, and Midjourney.

A. Generative Models and Their Evolution

Generative models are based on the principle that they can reproduce results similar to the training data on which they were originally trained. There are six main types of generative models: Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), Autoregressive Models, Flow-Based Models, Energy-Based Models, and Diffusion Models [4]. While each model has its own applications and importance, understanding GANs and VAEs is crucial for grasping the evolution of modern diffusion models used today in text-to-image generation.

GANs operate based on two main models: a generator and a discriminator. The generator is a Decoupled Convolutional Neural Network (DCN), while the discriminator is a Convolutional Neural Network (CNN). The generator is fed by a fixed-dimensional noise vector or latent variables, which it uses to produce fake images. These images are then mixed with real images from the dataset and fed into the discriminator, which classifies them as either fake or real. The classification accuracy is then used to update both the generator and discriminator. The generator's objective is to increase the classification error, while the discriminator's objective is to reduce it [5].

VAEs consist of three main components: the encoder, a latent space, and a decoder. The encoder takes a high-dimensional input, which is compressed and fed into the latent space. The decoder accesses this code from the latent space and decodes it as accurately as possible. The primary focus of VAEs is to reduce the noise in the data, recognize relevant features, and detect anomalies [6].

The diffusion model, which is used in most text-to-image generator applications today, involves two processes: a predefined forward process that maps data distribution to a Gaussian distribution, and a reverse process that uses a trained neural network to reverse the effects of the forward process using ordinary or stochastic differential equations (ODE/SDE). These processes occur in the latent space, with the text encoder conditioning the mapping in the forward process [7]. Diffusion models are highly efficient in producing high-quality photorealistic images. They address issues like mode collapse, provide high stability, work well with different data types, and can be conditioned for various use cases.

B. Transformer Architectures in Image Generation

The transformer architecture, an attention-based alternative to recurrent networks, is used for various NLP tasks, particularly in text-to-image applications. It consists of six key components, starting with the self-attention mechanism, which provides the model with the weights of different words in a sentence. Then, positional encoding is applied, which indicates the position of the tokens in the sequence. The concept of multi-head attention allows for the simultaneous application of several attention mechanisms in parallel. After this, the data passes through the feed-forward network, which produces various encoded representations. Additionally, the encoder-decoder structure is integral to this model [8].

The transformer architecture is used in a variety of applications, ranging from NLP tasks like language modeling and text summarization to vision tasks like image classification, where Vision Transformers (ViT) are commonly employed.

In summary, implementing text-to-image models on edge devices like mobile phones presents challenges due to limited computational resources. To address these challenges, techniques such as model distillation and diffusion GAN fine-tuning have been proposed. Model distillation involves training smaller models to replicate the performance of larger models with reduced complexity. Key factors affecting inference efficiency include architecture and sampling. The MobileDiffusion model tackles these issues by replacing the U-Net architecture with separable convolutions, reducing computational requirements while maintaining image quality. It also refines transformer blocks and activation functions to improve performance on mobile devices. Additionally, sampling efficiency has been improved through methods like progressive distillation, which reduces sampling steps and enables real-time inference. These advancements make it feasible to deploy high-quality text-to-image models on mobile devices, paving the way for real-time applications. Ongoing research aims to further refine these techniques to balance computational efficiency with high-quality image generation, enabling broader use of AI models on smartphones.

III. Methodology

The research presented in this paper uses a pre-trained Stable Diffusion model to create high-quality images from simple text-based prompts provided by the user. Our methodology leverages the cutting-edge capabilities of diffusion models, specifically the CompVis/stable-diffusion-v1-4 variant, which was trained on large datasets of images and text annotations. The steps followed are as follows: downloading and installing the required libraries, loading the pre-trained model, receiving user prompts via an interactive interface, and generating an image based on these prompts. There is no need to collect or preprocess data, as the pre-trained model is already equipped to interpret the text and generate the corresponding image. Additionally, we used mixed precision and GPU acceleration to ensure faster image generation.

```
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)
```

Fig. 1. Code used to create an input widget.

To implement the text-to-image model, the following libraries were installed to ensure smooth operation: PyTorch/Torch, Diffusers, and Pillow [9]. A pre-trained Stable Diffusion model was used for this research from the Hugging Face model hub. To increase computational speed and reduce memory usage, mixed precision (fp16) was applied to the model.

The model works on the data or input provided by the user. Therefore, the following code (Fig. 1) was used to generate an input widget as part of the data collection process.

To obtain the image generated from the text prompt, the function ‘generate_image’ is used, which also saves the generated image to a file. To facilitate the image generation process, a button is created.

The text-to-image generation model was implemented using the Stable Diffusion model from Hugging Face’s diffusers library. Once the model is loaded, it can be executed within a Python environment, including the necessary import dependencies such as torch, diffusers, and optional requirements for interactive user input, such as images. It allows users to input text prompts via an easy-to-use widget interface, which then prompts the model to generate corresponding images. The generated images are saved in a folder for later analysis. 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 the methodology, images are successfully generated based on text prompts using the pre-trained Stable Diffusion model. The model outputs transformed image data based on the prompts provided by the user.

This work follows a pipeline for text-to-image generation, programmed in Python, utilizing the Stable Diffusion

model (CompVis/stable-diffusion-v1-4). The system includes text input widgets for user interaction, allowing users to generate high-quality images from the provided text. Key tools and libraries include PyTorch for tensor manipulation, GPU acceleration, and PIL for image preprocessing operations.

IV. Implementation

The implementation followed a step-by-step breakdown of the problem statement, focusing on specific areas of improvement. The research uses a Stable Diffusion model, so there was no need for external data. However, we began by installing several essential libraries. Next, we loaded pre-trained models, including the base model for this research—the Stable Diffusion model—along with the FID and CLIP models for evaluation purposes. An initial evaluation of the model was then performed to establish a reference point, allowing us to compare percentage improvements in subsequent results.

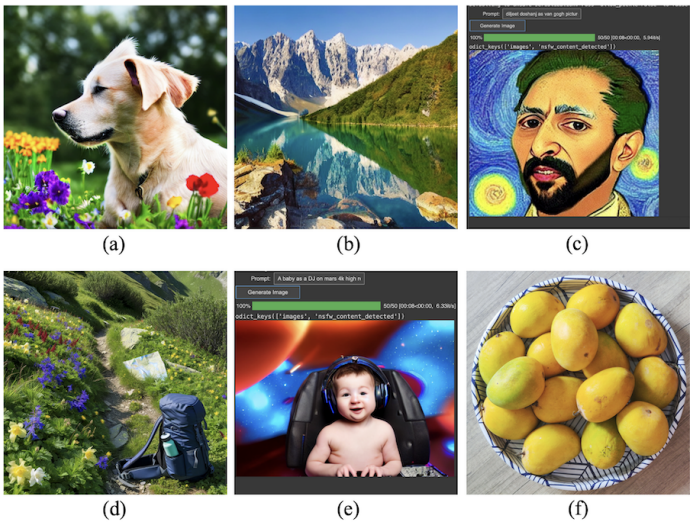
The next phase focused on improving the model to achieve better inference time, FID, and CLIP scores. While the changes made to the model resulted in some improvements, it was observed that the output varied with each new prompt, and there were no significant changes in the FID and CLIP scores. Therefore, the focus shifted to optimizing inference time. To achieve this, we modified the timesteps and scheduler, which led to improved inference times. Additionally, profiling was conducted to identify any bottlenecks in the model, and the results are shown in Table I.

The profiling indicates potential bottlenecks and issues within the model. While the attention layers consume a significant amount of time, the primary concern lies with the convolution operations, which are the main bottlenecks in both CPU and CUDA processing times. To address this, we implemented a custom UNet architecture utilizing depthwise separable convolutions. This modification helps reduce the number of parameters and computational cost compared to standard convolutions.

In our efforts to improve sampling efficiency, we incorporated DDIM (Denoising Diffusion Implicit Models) with as few as 5 timesteps, yielding improved results. Additionally, to enhance inference speed, we employed mixed precision, which not only improved computational efficiency but also reduced memory usage, making it particularly well-suited for large models with extensive datasets. The next subsections detail the implementation process.

A. Libraries, Models, Profiling

The important libraries used for the model include Diffusers, Transformer, Torch, TorchVision, SciPy, and ipywidgets. Several pre-trained models were used in the research to ensure code efficiency and access to authentic, large datasets. The models include Stable Diffusion for the execution and creation of the application, the CLIP model, and the FID model for evaluation. Profiling was conducted to identify any bottlenecks and address them with code modifications, as outlined in the next step.



(a) (b) (c)

(d) (e) (f)

Fig. 2. Images generated on user prompts: (a) A candid photo of a Labrador enjoying the breeze in a garden in 4K resolution; (b) A scenic landscape with mountains and a lake; (c) Diljit Dosanjh in a Van Gogh painting; (d) A backpack on a mountain trail; (e) A baby as a DJ on Mars; (f) A bunch of mangoes in a geometric bowl.

B. Improving the Model

Three main changes were made to the model. The first was the use of mixed precision. The second was the incorporation of the DDIM scheduler, where we initially started with 50 timesteps and later reduced it to as few as 5 timesteps, which improved sampling efficiency. The final change was the implementation of a custom UNet architecture with depthwise separable convolutions and the use of Swish activation instead of ReLU.

V. Evaluation

This study evaluates the performance of text-to-image generation using the Stable Diffusion model, with three key metrics: inference time, Fréchet Inception Distance (FID), and CLIP score. Inference time measures how quickly the model generates an image from a text prompt, which is crucial for real-time or near-real-time applications, such as interactive art or VR applications. FID measures the distance between feature vectors of generated images and real images, with lower scores indicating higher image quality and diversity, which is important for applications like art and media [10]. The CLIP score evaluates how well generated images align with textual descriptions by computing the cosine similarity between the text and image embeddings, with higher scores indicating better text-to-image accuracy [11]. By measuring these metrics when images are generated by the Stable Diffusion model, a comprehensive assessment of the model's performance is conducted, and areas for improvement to enhance its real-world applicability can be identified.

A. Image Generation

First we analyze how accurately the model is able to generate images depending on the given promtps. Fig. 2

TABLE I
PROFILING OF THE MODEL SHOWING RESOURCE USAGE

Kernel Name	Self CPU %	Self CPU	CPU Total%	CPU Total	CPU Time Avg	Self CUDA	Self CUDA%	CUDA Total	CUDA Time Avg	No of Calls
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

```
Comparison of Initial and Modified Model Performance:
Initial Inference Time: 9.26 seconds
Modified Inference Time: 8.65 seconds
Time Reduction: 6.57%
Initial FID Score: 11036.34
Modified FID Score: 14057.75
FID Score Improvement: -3021.41
Initial CLIP Score: 29.23
Modified CLIP Score: 28.87
CLIP Score Improvement: -0.36
```

Fig. 3. Comparision of the outputs

```
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
```

Fig. 4. Comparison of Modified Model with Initial Model

illustrates a set of images generated on user prompts.

B. Initial model

The initial model is based on Stable Diffusion CompVis v1-4 with a CLIPImageProcessor for feature extraction. In the initial model, we also used the PNDMS (Pseudo Numerical Method for Diffusion Model) Scheduler for sampling efficiency. For the architecture, the UNet2DConditionModel from the UNet architecture and AutoencoderKL for VAE were used.

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 (as seen in Fig. 3) show an inference time of 9.26 seconds, an FID score of 11036.33, and a CLIP score of 29.23. For the initial model, all three parameters were unsatisfactory, so changes were made to the model.

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

C. Sampling Efficiency

1) Initial Model: In the previous result, although we achieved satisfactory results for inference time, the quality of the image was being affected. To address this, we kept the use of mixed precision as it was and reduced the timesteps to as low as 5. The use of DDIM ensured that the image quality was maintained, and reducing the timesteps made the image generation faster.

2) Modified Model: As seen in Fig. 4, it can be observed that we achieved a 10 percent improvement in inference time. Not only that, but the FID also improved significantly, with a score of 8919, down from 11036. The CLIP score remained largely unchanged.

Fig. 5. Image Generated by User Prompt “Monkey as DJ”

D. Output with User Prompt

In this section, we will examine some images generated from user prompts. What was striking is that the inference time varied for each image generated. While this depends on the complexity of the prompt, we also experimented with a timestep of 3.

1) User Prompt Images: The first image (Fig. 5) was generated with the user prompt ”monkey as DJ” and shows an 11 percent improvement in inference speed (as seen in Fig. 6). However, the FID score was affected, and the CLIP score remained almost the same. To address this and maintain the improved inference speed, the final modification to the architecture was made, which is discussed in the next section.

Another image generated was “scarecrow with pink hoodie” (see Fig. 7). For this image, a 14 percent improvement in inference speed was observed, with satisfactory FID and CLIP scores, as seen on Fig. 8.

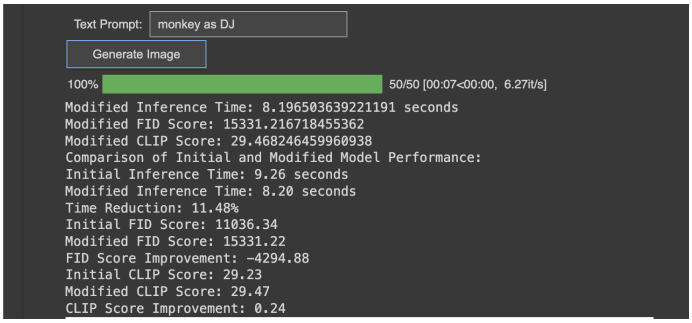


Fig. 6. Results of the "Monkey as DJ" image

Fig. 7. Image generated with prompt "scarecrow with pink hoodie"

2) Results After UNet Changes: In the final modification, we changed the UNet architecture to achieve more accurate images by using depthwise separable convolutions and switching the activation function from ReLU to Swish. The results not only improved but the images were also better, as seen in Figs. 9 and 10. We can observe that the inference speed increased from 11 percent to 14 percent for the "monkey as DJ" image (Fig. 11), and similarly, for the "scarecrow with pink hoodie" image, the inference speed increased from 14 percent to 15 percent, with an improved image (Fig. 12). The improvements are summarized in Tables II and III.

VI. Conclusion and Future Work

This study aimed to improve the efficiency of text-to-image generation on edge devices while enhancing both image quality and inference speed. The focus was on engineering modifications and testing their effectiveness, proposing updates for scalable applications. Key changes included replacing standard diffusion models with depthwise separable convolutions, using a DDIM scheduler, blending accuracy techniques, and optimizing the UNet architecture. These adjustments led to notable improvements in inference speed and image quality, as evidenced by comparisons of inference times and evaluation metrics such as FID and CLIP scores. The study demonstrated the feasibility of deploying generative AI on edge devices, achieving significant time savings and better image fidelity. However, challenges remain, particularly with complex

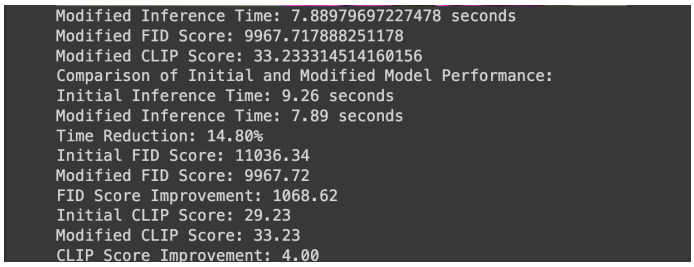


Fig. 8. Comparision of result - initial model vs current output

Fig. 9. Image generated after UNet modification for "monkey as DJ"

Fig. 10. Improved image for "scarecrow with hoodie"

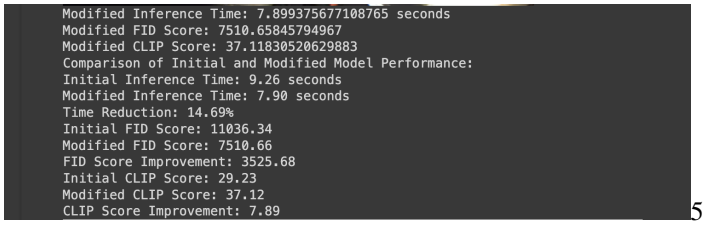


Fig. 11. Improved Results for "monkey as DJ"

TABLE II
RESULTS OBTAINED AFTER APPLYING SAMPLING CHANGES

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 III
RESULTS OBTAINED AFTER MAKING ARCHITECTURAL 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

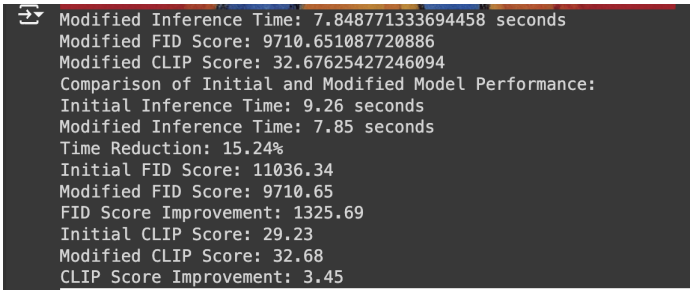


Fig. 12. Improved results for "scarecrow with hoodie"

prompts, which affected both speed and image quality. Further advancements in attention layers, model architectures, adaptive learning techniques, and real-world testing are necessary to optimize performance. These improvements could enable cost-effective, user-friendly generative AI applications on edge devices, expanding their potential in creative and professional fields. Despite the progress made, the study highlighted the ongoing challenges of fine-tuning generative models for edge environments, particularly in balancing speed and image quality across diverse use cases. As future work, other AI-based techniques and models [12], [13], [14], [15], [16], and [17] could be investigated and compared with the current solution.

ACKNOWLEDGMENT

Work supported by Research Ireland via the Frontiers Projects grant 21/FFP-P/10244 (FRADIS) and Research Centres grant 12/RC/2289_P2 (INSIGHT), and by the European Union (EU) Horizon Europe grant 101135637 (HEAT Project).

REFERENCES

[1] M. Chohan, A. Khan, M. S. Mahar, S. Hassan, A. Ghafoor, and M. Khan, "Image captioning using deep learning: A systematic," *image*, vol. 11, no. 5, 2020.

[2] E. Mansimov, E. Parisotto, J. L. Ba, and R. Salakhutdinov, "Generating images from captions with attention," *arXiv preprint arXiv:1511.02793*, 2015.

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

[4] G. Harshvardhan, M. K. Gourisaria, M. Pandey, and S. S. Rautaray, "A comprehensive survey and analysis of generative models in machine learning," *Computer Science Review*, vol. 38, p. 100285, 2020.

[5] L. Mi, M. Shen, and J. Zhang, "A probe towards understanding gan and vae models," *arXiv preprint arXiv:1812.05676*, 2018.

[6] J. Kos, I. Fischer, and D. Song, "Adversarial examples for generative models," in *IEEE Security and Privacy Workshops (spw)*. IEEE, 2018, pp. 36–42.

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

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

[9] A. F. Villán, *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, 2019.

[10] M. J. Chong and D. Forsyth, "Effectively unbiased fid and inception score and where to find them," in *IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 6070–6079.

[11] X. Liu, C. Gong, L. Wu, S. Zhang, H. Su, and Q. Liu, "Fusedream: Training-free text-to-image generation with improved clip+gan space optimization," 2021. [Online]. Available: https://arxiv.org/abs/2112.01573

[12] I.-S. Comşa, A. Molnar, I. Tal, P. Bergamin, G.-M. Muntean, C. H. Muntean, and R. Trestian, "A machine learning resource allocation solution to improve video quality in remote education," *IEEE Transactions on Broadcasting*, vol. 67, no. 3, pp. 664–684, 2021.

[13] I.-S. Comșa, A. Molnar, I. Tal, C. Imhof, P. Bergamin, G.-M. Muntean, C. H. Muntean, and R. Trestian, "Improved quality of online education using prioritized multi-agent reinforcement learning for video traffic scheduling," *IEEE Transactions on Broadcasting*, vol. 69, no. 2, pp. 436–454, 2023.

[14] A. Menon, A. Siddig, C. H. Muntean, P. Pathak, M. Jilani, and P. Stynes, "A machine learning framework for shuttlecock tracking and player service fault detection," in *Deep Learning Theory and Applications*, D. Conte, A. Fred, O. Gusikhin, and C. Sansone, Eds. Cham: Springer Nature Switzerland, 2023, pp. 71–83.

[15] A.-N. Moldovan and C. H. Muntean, "Qoe-aware video resolution thresholds computation for adaptive multimedia," in *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, 2017, pp. 1–6.

[16] A. A. Simiscuka, D. A. Ghadge, and G.-M. Muntean, "Omniscent: An omnidirectional olfaction-enhanced virtual reality 360° video delivery solution for increasing viewer quality of experience," *IEEE Transactions on Broadcasting*, vol. 69, no. 4, pp. 941–950, 2023.

[17] A. A. Simiscuka, M. A. Togou, R. Verma, M. Zorrilla, N. E. O'Connor, and G.-M. Muntean, "An evaluation of 360° video and audio quality in an artistic-oriented platform," in *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, 2022, pp. 1–5.