CS485 Term Project Final Report SWaTGAN: Sliding Window Attention for Efficient Image Generation

Mennan Gök
22003074
Department of Mathematics
Bilkent University
mennan.gok@ug.bilkent.edu.tr

Ömer Tuğrul
22002723
Department of Electrical and Electronics Engineering
Bilkent University
o.tugrul@ug.bilkent.edu.tr

I. Introduction

In this project, we propose the novel architecture Sliding Window Attention-GAN (SWaTGAN), which utilizes the sliding window attention (SWA), proposed in [2]. Initially proposed for natural language processing tasks, SWA limits the receptive field of each token to a pre-determined window size. In this way, in the early layers of transformer blocks, the similarity score is calculated between the near-by tokens. The receptive field of each token increases towards the latter transformer layers, enabling the model to capture long-range dependencies despite starting with a narrow receptive field. Using SWA enables SWaTGAN to achieve a competitive performance in unconditional image generation tasks in a more efficient way than vanilla transformer based architectures, e.g. [4]. We trained SWaTGAN model with CelebA dataset, taken from Kaggle [7], which contains celebrity face images. We also trained TransGAN [4], and DCGAN [3] models with the same dataset in order to compare the results, and the training times of these models.

In Section 2 Related Works, we briefly explain the previous work on unconditional image generation, sliding window attention, TransGAN, and DCGAN. In Section 3 Model Details, we give the architecture details of SWaTGAN, by explaining the underlying working principles of the model, and progressive growing in the generator. In Section 4 Experiments, we explain the training details and configurations of the model, and the details of the dataset we used during training. Lastly, in Section 5 Results, we state and discuss the qualitative and quantitative evaluation results, comparing SWaTGAN's performance with the performance of TransGAN [4] and DCGAN [3].

II. Related Works

A. Unconditional Image Generation

Unconditional image generation is a process where a model generates images from scratch without any specific input conditions like labels or descriptions. The model learns the distribution of the training data and creates new images that look like the training images. One of the common generative model architecture is **generative adversarial networks(GANs)** [1]. The *generator* network in GANs produces images, while the *discriminator* network evaluates them, with both networks training simultaneously in a competitive manner, which can be demonstrated as a two player mini-max game. The primary challenge in unconditional image generation is achieving high-quality and diverse images, which is addressed through advanced training techniques and architectural innovations.

B. DCGAN: Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

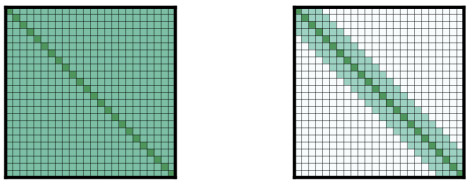
DCGAN[3], introduced by Radford, Metz, and Chintala in 2015, replaced the linear layers in Generative Adversarial Networks[1] with *deep convolutional layers* and has become a foundational model in the field of generative adversarial networks (GANs). It uses deep convolutional layers in both the generator and discriminator. The model introduces techniques to stabilize GAN training, such as avoiding max-pooling and using batch normalization. However, DCGAN relies on convolutional layers, which are inherently local in nature. Hence, they are struggling to capture long-range dependencies.

C. TransGAN: Two Pure Transformers Can Make One Strong GAN

TransGan [4], introduced by Jiang, Chang, and Wang in 2021, marks a significant difference from traditional GAN architectures [3] by replacing the convolutional layers with *transformer blocks*. This paper is the first paper that entirely replaced convolutional layers with transformer blocks. Unlike the convolutional layers that are good at handling local dependencies, the multi head self-attention mechanism of transformers enables the model to capture long-range dependencies in the image data. This capability is crucial for generating high-resolution images. In TransGAN, both the generator and discriminator are composed of multiple layers of transformer blocks. However, the matrix multiplications in the self-attention mechanism is computationally expensive and requires large memory that can limit the scalability of the model. Furthermore, the transformers are data hungry and need too much data for training. The paper addresses this issue with data augmentation, while the problem with computational inefficiency still persists.

D. Longformer: The Long-Document Transformer

The Longformer[2], introduced by Beltagy, Peters, and Cohan in 2020, addresses the challenge of processing long documents with transformer models. Traditional transformers with their quadratic complexity with respect to input length are inefficient for handling the long sequences. Longformer paper introduces a novel attention mechanism, namely **the sliding window attention** shown in Figure 1.(b), that reduces this complexity, enabling efficient handling of long documents. The sliding window attention mechanism limits each token's attention to a fixed size window of neighboring tokens. This reduces the computational complexity from quadratic to linear with respect to the sequence length.



(a) Full n^2 attention (b) Sliding window attention

Fig. 1: A visual comparison of vanilla attention mechanism (a) and the sliding window attention mechanism (b). The figure is taken from the Longformer paper [2], which first proposes the sliding window attention mechanism.

III. Model Details

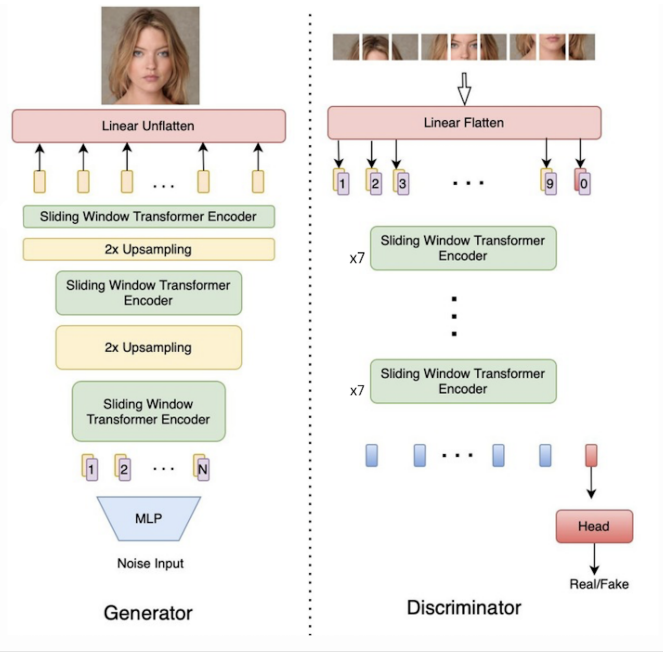


Fig. 2: The generator and discriminator architecture of SWaT-GAN model

The proposed model SWaTGAN is a transformer-based GAN model for unconditional image generation tasks. SWaT-GAN utilizes the efficient sliding window attention layers [2] in order to achieve good performance in a memory-efficient way. As explained in Section 2, in sliding window attention layers, the similarity score of each token is computed only within a local neighborhood of each token, whose size is determined by the *window size* parameter in our model.

As illustrated in Figure 2, the **generator** of the model progressively increases the spatial resolution, while decreasing the channel size in order to ensure a memory-friendly training, a technique first proposed in [8]. The generator receives a random noise vector of dimension 1024. Then it passes the noise through **a multi-layer perceptron (MLP)** block which consists of several linear layers, and GELU activation function. Then we use three consecutive **Sliding Window Transformer Encoder blocks (SWTE)**. As illustrated in Figure 3, each SWTE utilizes several **encoder blocks**, each of which contains the sliding window attention layer, layer normalizations, and an MLP block. The number of encoder blocks in each SWTE is determined by the parameter named *depth*. In Section 4 Experiments, we will share the details of the experiments we conducted with different choices of depth to see its effect on the performance and training time. The three consecutive SWTE blocks are trained to generate the latent vectors required to generate 8x8, 16x16, and 32x32 images respectively, in a progressive manner. To achieve this progression, we used two **upsampling layers** with nearest-neighbor interpolation, one in between each SWTE blocks. Lastly, we unflatten the latent vector generated by the last SWTE, using a 2D convolution layer, to obtain the generated image. In the **discriminator**, we use a SWTE block with 7 encoder blocks, i.e. depth is 7. The input image is flattened through a 2D convolution layer and passed through the single SWTE with 7 encoder blocks. The discriminator, then, outputs if the input image is real or fake.

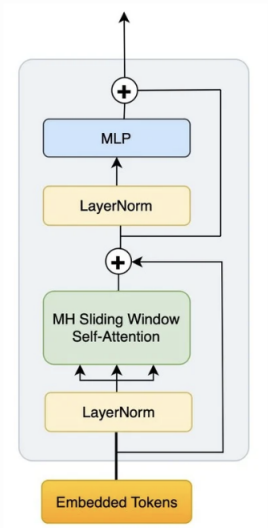


Fig. 3: A single Sliding Window Transformer Encoder Block

Traditionally, the generator and discriminator are both trained via the adversarial mini-max game [1], represented by the equation

$$\min_G \max_D V(D,G) = E_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log D(\mathbf{x})] + E_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]. \quad (1)$$

,where G and D represent the generator and discriminator functions respectively, p_{data} represents the distribution of the training dataset, and $p_{\mathbf{z}}$ represents the generated data distribution. Here, G tries to minimize this loss function by trying to fool D into believing that the generated images are real, while D tries to maximize this loss function by learning to tell generated images from real images.

Our model, SWatGaN, offers several advantages over the traditional GAN architectures by incorporating the SWA mechanism. By stacking multiple transformer blocks on top of each other, SWaTGAN effectively captures both local and long-range dependencies in images, which results in improved quality in the generated output images. SWA reduces the computational complexity compared to vanilla self-attention, quadratic to linear, and make it more efficient, scalable for high-resolution image generation. Furthermore, it improves training stability by balancing local and global feature handling that results in better performance compared to conventional convolutional GANs like DCGAN.

IV. Experiments

A. Dataset Details

For this project, we used the Large-scale CelebFaces Attributes (CelebA) dataset, which is very common dataset among the state of art architectures. It contains 202,599 celebrity images. The images are originally have a resolution of 178x218 pixels, but we were resized the images to 32x32 pixels to be able to fit our computational resources. The dataset features diverse facial expressions, poses, and attributes, making it ideal for tasks like facial attribute recognition and image generation. In Figure 4, we demonstrate a small sample of the 32x32 resized images from our dataset. More details about the CelebA dataset can be found on its Kaggle page [7].

Fig. 4: A sample of images from CelebA dataset. Resized to 32x32.

B. Training Details

We used a 12GB Nvidia RTX 4080 GPU for model training. In Table I, we showed the 4 models we trained. In the **Depth** column, we showed the number of encoder blocks, 3, per layers of SWTE blocks in the generator (3 layers in total). In the second and third columns, we showed the **Training Time** and **GPU Usage** during training, respectively. In the last column, we show the **FID** scores of each model.
We trained all 4 models with a batch size of 64, a learning rate of 1e-4 both for generator and discriminator, and 1e-3 weight decay, for 20 epochs. Since we do not have enough computational resources to conduct more detailed experiments, unfortunately we could not optimize the training parameters for each model.

1) TransGAN: We trained one TransGAN model with depths 5, 4, and 2 in the generator. In the discriminator, we used a single SWTE with 7 encoder blocks, the same as in all of the 4 models. We trained the TransGAN model for 20 epochs, which took approximately 10 hours, utilizing approximately 11.6GB of our 12GB GPU. In the end, we got and FID score of 144.38.

2) SWaTGAN: We trained three SWaTGAN models with varying depths in the generators. These varying depths are 5-4-2, the same as the TransGAN, 7-7-7, and 15-12-9. In the discriminators, we used a single SWTE with 7 encoder blocks, the same as the TransGAN. We trained each SWaTGAN model for 20 epochs. The training of the 5-4-2 model took approximately 6 hours, utilizing approximately 5.4GB GPU. In the end, we got and FID score of 177.89.

The training of the 7-7-7 model took approximately 7 hours, utilizing approximately 8GB GPU. We got an FID score of 164.05. Lastly, the training of 15-12-9 model took approximately 9 hours, utilizing approximately 11.2GB GPU, close to the TransGAN model. In the end, we got and FID score of 155.32. With the 15-12-9 model, we tried to limit-test our model by observing how does the depth parameter affects the training time and GPU usage. We increased the depth size as far as our GPU allowed, to observe how SWaTGAN performs compared to TransGAN with limited computational resources.

V. Results

A. Quantitative Results

TABLE I: Training Time and FID Score Comparison

Model	Depth	Training Time (h)	GPU Usage	FID
TransGAN	5-4-2	10	11.6G	**144.38**
SWaTGAN	5-4-2	6	5.4G	218.5549
SWaTGAN	7-7-7	7	8G	164.05
SWatGAN	15-12-9	9	11.2G	155.32

We compared the TransGAN model and three SWaTGAN models with varying depths, based on training time and Fr echet Inception Distance (FID) scores. TransGAN, achieved the best FID score of 144.38 but required 10 hours of training. The most minimal SWaTGAN model with 5-4-2 depth configuration in the generator took almost half the time and utilized half the GPU as the TransGAN model, but performed worse than the TransGAN model based on the FID scores (144.38 in TransGAN vs. 218.5549 in SWaTGAN 5-4-2). The second SWaTGAN model with 7-7-7 depth configuration in the generator performed slightly better than the previous SWaTGAN model by sacrificing a slight training time and gpu

utilization. This motivated us to limit-test our computational resources by increasing the depth parameters as far as our GPU allows. As a result, we trained our largest model SWaTGAN with 15-12-9 depth configuration, which performed better than other two SWaTGAN models. The last SWaTGAN model took almost as much time as the TransGAN model took for training, and utilized a close GB of GPU. However, based on the FID scores we conclude that the TransGAN model performed better than our SWaTGAN model with similar computational resources. Nonetheless, as we stated earlier, we could not conduct experiments to optimize the hyperparameters of the SWaTGAN model due to the lack of enough computational resources. We believe that if we optimize the hyperparameters of SWaTGAN and can train the models longer, we can possibly observe promising results of our SWaTGAN model.

SWaTGAN, utilizing sliding window attention, balanced efficiency and quality, completing training in 6 hours with a FID score of 218.5549. While Trans- GAN produced higher-quality images, SWaTGAN offered a more efficient training process, making it suitable for scenarios with limited computational resources.

B. Qualitative Results

In this section, we demonstrate some samples of images generated by the models we trained.

Fig. 5: A sample of images generated by TransGAN model

Fig. 6: A sample of images generated by the SWaTGAN model with 5-4-2 configuration

Fig. 7: A sample of images generated by the SWaTGAN model with 7-7-7 configuration

Fig. 8: A sample of images generated by the SWaTGAN model with 15-12-9 configuration

VI. Conclusion & Future Work

In this project, we introduced SWaTGAN, a novel GAN architecture leveraging Sliding Window Attention (SWA) for efficient image generation. By using SWA, SWaTGAN effectively captures both local and global dependencies in images, providing a balanced approach to generate high-quality images while maintaining computational efficiency. We trained and evaluated SWaTGAN on the CelebA dataset, comparing its performance against DCGAN and TransGAN. Our results shows that SWaTGAN offers a trade-off between the training time and image quality, which makes model a better choice compared to the TransGAN if there is a limited computational resource. Although TransGAN achieved the best Fréchet Inception Distance (FID) score, SWaTGAN's efficiency and competitive performance make it a strong candidate for future research and applications in generative modeling. Future work could explore further optimizations of hyperparameters and the model architecture of SWATGAN that results in better performance compared to TransGAN and other transformer based generative adversarial networks. Also, future work will include to extend th SWaTGAN to handle higher resolution images and more diverse datasets.

Acknowledgment

We would like to express our deepest gratitude to Professor Ayşegül Dündar Boral for her profound inspiration. Additionally, we extend our sincere thanks to Ahmet Burak Yıldırım for his invaluable contributions throughout our machine learning journey that made us capable of completing this project.

We sincerely thank Professor Atiye Tuğrul for providing us with the necessary resources for this project, including the GPU and the computer.

References

[1] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In Advances in neural information processing systems (pp. 2672-2680).

[2] Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The Long-Document Transformer. arXiv preprint arXiv:2004.05150.

[3] Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. arXiv preprint arXiv:1511.06434.

[4] Jiang, Y., Chang, S., & Wang, Z. (2021) TransGAN: Two Pure Transformers Can Make One Strong GAN, and That Can Scale Up. [Online]. Available: arXiv:2102.07074v1

[5] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention Is All You Need. arXiv preprint arXiv:1706.03762.

[6] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2020). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. arXiv preprint arXiv:2010.11929.
[7] https://www.kaggle.com/datasets/jessicali9530/celeba-dataset/
[8] Karras, T., Aila, T., Laine, S., Lehtinen, J. (2018). Progressive Growing of GANs for Improved Quality, Stability, and Variation. arXiv preprint arXiv:1710.10196.