SAN JOSÉ STATE UNIVERSITY

GAN

(Generative Adversarial Network)

Project Report

By

Vinit Kanani

Student ID: 016651323

To

Professor Shilpa Gupta

May 15, 2023

Problem Definition

Data augmentation plays a crucial role in improving the performance of machine learning models, especially in domains where labelled data is limited. Traditional data augmentation techniques such as rotation, translation, and scaling have been widely used. However, these techniques are limited in their ability to generate diverse and realistic data samples. The paper "Generative Adversarial Networks (GANs)"[1] proposes a new method for estimating generative models via an adversarial process. It have emerged as a powerful tool for data augmentation by leveraging the power of deep learning to generate synthetic data samples that closely resemble real data.

The goal of GAN-based data augmentation is to train a generative model that can generate high-quality synthetic data samples that capture the underlying distribution of the real data. GANs consist of two main components: a generator network and a discriminator network. The generator network takes random noise as input and generates synthetic samples, while the discriminator network aims to distinguish between real and synthetic samples. The two networks are trained in an adversarial manner, where the generator aims to fool the discriminator, and the discriminator aims to correctly classify real and synthetic samples.

By training a GAN on a dataset of real samples, the generator learns to capture the complex patterns and structures present in the data. This allows the generator to generate new samples that resemble the real data distribution. GAN-based data augmentation can be applied in various domains, such as computer vision and natural language processing, to increase the diversity of training data and improve the generalization and robustness of machine learning models.

The use of GANs for data augmentation has shown promising results in various applications. In computer vision, GAN-based data augmentation has been used to generate realistic images with different attributes, such as different poses, lighting conditions, or object occlusions. This enables the training of models that are more robust to variations in the input data. In natural language processing, GANs have been used to generate synthetic text samples that exhibit similar linguistic characteristics as the real data. This can help in training language models that better capture the semantics and syntax of the target language.

The "Generative Adversarial Networks" paper has had a significant impact across various domains, revolutionizing the field of artificial intelligence and machine learning. GANs have introduced new possibilities for generating realistic and high-quality synthetic data, enhancing creativity, and pushing the boundaries of what machines can achieve. This work forms a foundation for the following papers:

1. "A Simple Framework for Contrastive Learning of Visual Representations" by Chen, Ting et al. (2020) [2] proposes a novel integrated deep generative model, which is built by generative adversarial networks based on bidirectional long short-term memory and attention mechanism.

2. Frid-Adar, Maayan et al. (2018) presented "Synthetic data augmentation using GAN for improved liver lesion classification" [3] a data augmentation method that generates synthetic medical images using Generative Adversarial Networks (GANs).

Project Objectives

The objective of the project is to implement a Generative Adversarial Network (GAN) architecture. The GAN will be trained on the MNIST dataset and used to generate synthetic data samples for data augmentation. The performance of the augmented data will be compared to the original dataset. By comparing the results, the project aims to evaluate the effectiveness of GAN-based data augmentation in improving model performance and generalization. This analysis will provide insights into the potential benefits and limitations of using GANs for data augmentation in the specific domain of the project.

In this project, I have illustrated the iterative training of GANs, where the discriminator learns to differentiate between real and synthetic images and the generator learns to make synthetic images. I trained the GAN architecture for 50 epochs and observed the quality of the generated images. In addition, I explored the GAN with Self Attention architecture, which includes a self-attention mechanism to capture long-range dependencies and enhance image quality. I wanted to assess how well the two designs performed in terms of picture realism, small details, and overall visual quality. The goal of this study was to advance knowledge of GAN architectures and shed light on the advantages and disadvantages of employing SA-GAN for image generating tasks.

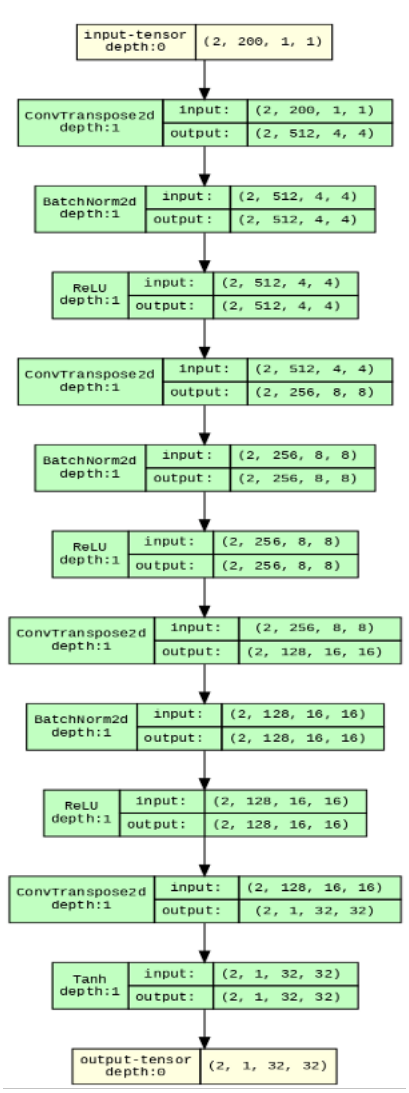
Analysis

Methodology

The first step involved training a Generative Adversarial Network (GAN) on the MNIST dataset. The GAN architecture comprises a generator network and a discriminator network. The generator network learns to generate realistic images, while the discriminator network distinguishes between real and synthetic images. Through adversarial training, the GAN improves its ability to generate increasingly realistic images.

The second step focused on training a Self-Attention GAN (SA-GAN) on the same MNIST dataset. SA-GAN is an advanced GAN architecture that incorporates a self-attention mechanism. This mechanism allows the generator to focus on crucial spatial relationships within the images, resulting in higher-quality generated images with finer details and improved resolution.

The third step was utilized a Conditional Generative Adversarial Network (cGAN) to generate synthetic images from the MNIST dataset, specifically for the purpose of data augmentation. To assess the performance of cGAN-based data augmentation, I conducted a comparative analysis with traditional data augmentation techniques. I employed commonly used techniques such as rotation, translation, scaling and blur to augment the original dataset. By comparing the performance of models trained on the augmented data from both cGAN and traditional techniques, I aimed to evaluate the effectiveness of cGAN-based data augmentation in improving model performance and generalization.



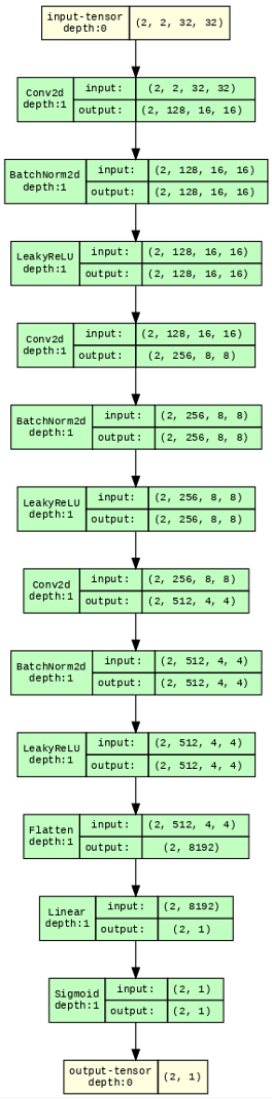


Figure 1: “Architecture of Generator and Discriminator”

Self-Attention Layer

The Self-Attention layer is a key component in Generative Adversarial Networks (GANs) that enhances the ability of the generator to capture spatial relationships within images. It achieves this by allowing the generator to focus on important regions or features during the image generation process. The Self-Attention layer calculates attention weights for each spatial position by considering the relationships between all positions in the feature map. This attention mechanism enables the generator to effectively capture long-range dependencies and generate images with fine details and realistic textures. The inclusion of Self-Attention layers in GAN architectures has shown remarkable improvements in image quality and resolution, making it a valuable technique for generating high-quality synthetic images.

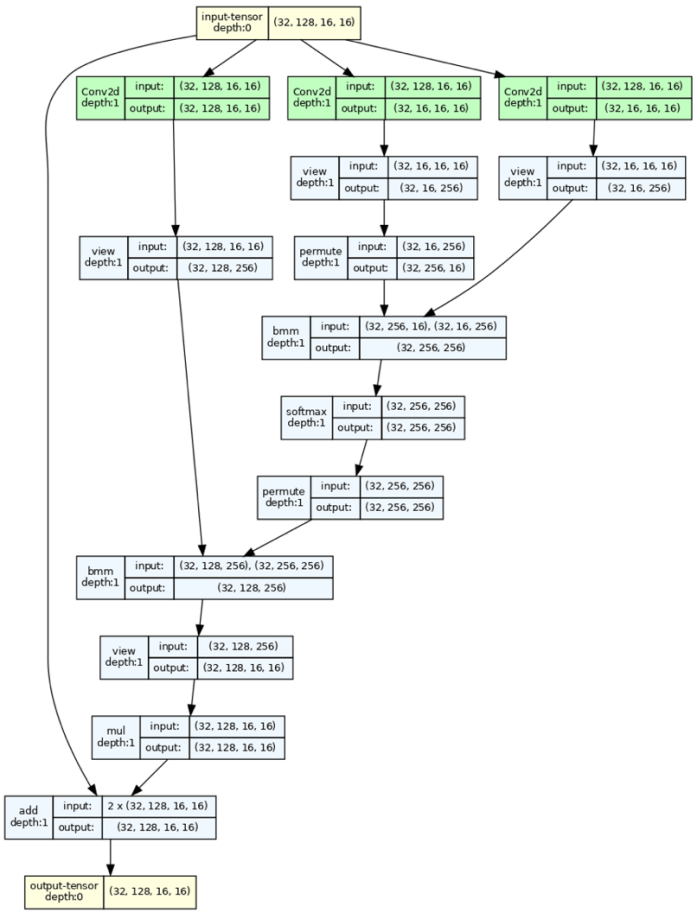


Figure 2: Self-Attention Layer

```
Generator(
  (model): Sequential(
    (0): ConvTranspose2d(100, 512, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): ConvTranspose2d(512, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (4): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU()
    (6): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (7): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ReLU()
    (9): SelfAttention(
      (query_conv): Conv2d(128, 16, kernel_size=(1, 1), stride=(1, 1))
      (key_conv): Conv2d(128, 16, kernel_size=(1, 1), stride=(1, 1))
      (value_conv): Conv2d(128, 128, kernel_size=(1, 1), stride=(1, 1))
    )
    (10): ConvTranspose2d(128, 1, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (11): Tanh()
  )
)
```

```
Discriminator(
  (model): Sequential(
    (0): Conv2d(1, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.01)
    (3): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (4): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): LeakyReLU(negative_slope=0.01)
    (6): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (7): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): LeakyReLU(negative_slope=0.01)
    (9): SelfAttention(
      (query_conv): Conv2d(512, 64, kernel_size=(1, 1), stride=(1, 1))
      (key_conv): Conv2d(512, 64, kernel_size=(1, 1), stride=(1, 1))
      (value_conv): Conv2d(512, 512, kernel_size=(1, 1), stride=(1, 1))
    )
    (10): Flatten(start_dim=1, end_dim=-1)
    (11): Linear(in_features=8192, out_features=1, bias=True)
    (12): Sigmoid()
  )
)
```

Figure 3: Model summary

Generator

The generator neural network model consists of several layers. The first layer is a transposed convolutional layer with a kernel size of (4, 4) and stride of (1, 1), followed by batch normalization and ReLU activation. The second layer is another transposed convolutional layer with a kernel size of (4, 4), stride of (2, 2), and padding of (1, 1), along with batch normalization and ReLU activation. The third layer is yet another transposed convolutional layer with similar parameters as the second layer, along with batch normalization and ReLU activation. The fourth layer is a self-attention layer that performs convolution operations on the input feature map. The fifth layer is a transposed convolutional layer with a kernel size of (4, 4), stride of (2, 2), and padding of (1, 1), followed by batch normalization. The final layer is a transposed convolutional layer with a kernel size of (4, 4), stride of (2, 2), padding of (1, 1), and Tanh activation function. This generator model is designed to generate images, taking a 100-dimensional input and producing an output with 1 channel.

Discriminator

This is a discriminator neural network model that consists of several layers. The first layer is a convolutional layer with a kernel size of (4, 4), stride of (2, 2), and padding of (1, 1), followed by batch normalization and LeakyReLU activation with a negative slope of 0.01. The second layer is another convolutional layer with similar parameters as the first layer, along with batch normalization and LeakyReLU activation. The third layer is yet another convolutional layer with similar parameters, followed by batch normalization and LeakyReLU activation. The fourth layer is a self-attention layer that performs convolution operations on the input feature map. The fifth layer is a flatten layer that converts the output tensor into a 1-dimensional vector. The sixth layer is a fully connected linear layer that maps the flattened input to a single output neuron. The seventh layer applies a sigmoid activation function to produce the final output of the discriminator, representing the probability of the input being real or synthetic. This discriminator model is designed to classify input images into real or synthetic categories.

Results

Synthetic images were generated using the cGAN and augmented the dataset using traditional techniques such as rotation and blur.

To evaluate the effectiveness of these augmentation techniques, I trained a CNN MNIST classifier on the original dataset as well as the augmented datasets. The classifier achieved an accuracy of 98% on the traditionally augmented dataset, while the GAN-augmented dataset achieved an accuracy of only 42%.

The reason for the lower accuracy with GAN-augmented data can be attributed to the fact that GAN-generated images were almost identical and did not capture the complete spectrum of variations present in the original dataset. This led to inconsistencies in the synthetic data, which negatively impacted the performance of the classifier.
On the other hand, traditional augmentation techniques such as rotation and blur manipulate the existing images in a way that does not alter the fundamental features of the digits, allowing the classifier to perform better.

The following table shows model accuracy for augmented data.

Architecture	GAN Data Augmentation	Traditional Data Augmentation
Accuracy	42%	98%

Table: Model accuracy for augmented data

Discussion

Limitations of GAN

- **Mode Collapse:** GANs are prone to mode collapse, in which the generator only generates a small variety of outputs and fails to fully capture the diversity of the target distribution. As a result, created samples could become monotonous and lack variation.
- **Training Stability:** Finding the right balance between the generator and discriminator networks may be difficult while training GANs. Convergence issues and oscillations might result from an unstable training procedure. To accomplish consistent training, fine-tuning architectural decisions and hyperparameters is frequently required.

Advantages of GAN

- **Realistic Image Generation:** GANs are excellent at creating incredibly realistic and aesthetically pleasing pictures. They are able to capture precise features and sophisticated patterns throughout the adversarial training process, producing pictures that closely resemble genuine samples from the training dataset.
- **Unsupervised Learning:** Since GANs don't need labeled training data, they may be used for unsupervised learning. The sample generator gains the ability to produce samples simply based on the input noise vector, without the use of labels or annotations.

Limitations of SA-GAN

- **Computational Demands:** The self-attention mechanism in SA-GAN has increased the architecture's computational complexity. Because of the additional training time and resource needs resulting from this, SA-GAN may not be as useful in situations when computational resources are limited.
- **Model Complexity:** SA-GAN adds extra layers and parameters, which complicates the model. When working with smaller datasets this complexity might raise the danger of overfitting, which requires strict regularization procedures to minimize the problem.

Advantages of SA-GAN

- **Enhanced Image Quality:** When compared to conventional GAN architectures, SA-GAN produces images with higher visual quality. Sharper and more realistic results are produced as a result of the attention layer's assistance in capturing complex patterns and global relationships.
- **Flexibility in other Domains:** SA-GAN is well suited for producing pictures in other domains beyond basic datasets like MNIST because to its capacity to model long-range relationships and capture fine details. It has produced realistic faces, landscapes, and other intricate visual material, among other jobs, with encouraging results.

Evaluation and Reflection

The produced pictures from the GAN and SA-GAN architectures were examined and contrasted visually throughout the evaluation process. After 50 iterations of training, it was noted that the GAN architecture produced relatively high picture quality for the MNIST dataset. The produced numbers could be recognized and had some resemblance to actual handwritten numbers. However, the photos that were created had certain restrictions in terms of fine details and sharpness.
However, the SA-GAN architecture proved to be more effective in producing MNIST digits. Sharper and more realistic images were produced as a result of the SA-GAN's improved ability to catch complicated patterns and features with the addition of the attention layer. The SA-GAN architecture demonstrated its capacity to produce digits that are both visually captivating and extremely believable, outperforming the performance of the conventional GAN design.
Based on the quality of the produced MNIST digits, the performance of both the basic GAN and SA-GAN architectures was evaluated during the assessment phase.

The visual quality of the produced digits was not as clear and sharp as intended after 82,800 steps of training the basic GAN. The generated images lacked fine details, resulting in slightly blurry and less realistic digits.

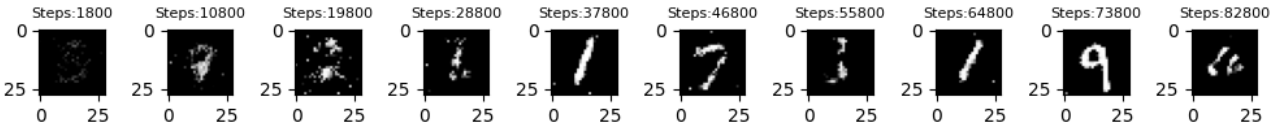


Figure 4: Training of Simple GAN

On the other hand, a measurable improvement in picture quality was seen after only 9,000 steps of training the SA-GAN architecture. In comparison to the simple GAN, the produced digits had sharper edges and finer features. The SA-GAN architecture's self-attention mechanism, which made it possible to capture long-range dependencies and increase the visual quality of the produced pictures, can be credited with this development.

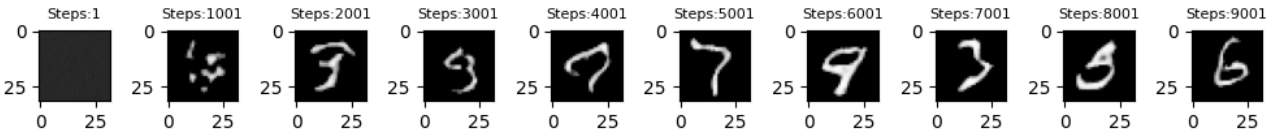


Figure 5: Training of Self-Attention GAN

Conclusion

In this project, I set out to explore and evaluate the performance of two different GAN architectures, the traditional GAN and the Self-Attention GAN (SA-GAN), for image generation on the MNIST

dataset. Through our experiments and discussions, I gained valuable insights into the strengths and limitations of these architectures, as well as the impact of different network designs on image quality.

Our evaluation of the simple GAN revealed that, after 82,800 steps of training, the image quality of the generated MNIST digits fell short of being sharp and clear. Despite reasonable resemblance to real handwritten digits, the simple GAN struggled to capture fine details, resulting in slightly blurry outputs.

On the other hand, the SA-GAN architecture showcased remarkable improvements in image quality. After just 9,001 steps of training, the SA-GAN-generated images exhibited sharper edges and finer details. This can be attributed to the inclusion of the self-attention mechanism, which allowed the SA-GAN to capture long-range dependencies and focus on important spatial relationships within the images. The attention layer played a crucial role in enhancing the visual quality and realism of the generated digits.

These findings highlight the advantages of using advanced GAN architectures like SA-GAN for image generation tasks. The self-attention mechanism proved to be a valuable addition, enabling the generation of crisper and more visually appealing images. However, it is important to consider the increased computational demands and model complexity associated with the SA-GAN architecture.

References

[1] Goodfellow, Ian J., et al. ‘Generative Adversarial Networks’. *ArXiv [Stat.ML]*, 2014.
[2] Chen, Ting et al. "A Simple Framework for Contrastive Learning of Visual Representations." *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 2020.
[3] Frid-Adar, Maayan et al. "Synthetic data augmentation using GAN for improved liver lesion classification." *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*. 2018.