A Brief History of Generative Adversarial Networks

Parth Sharma
luckyparth1401@gmail.com

ABSTRACT

Generative Adversarial Networks (GANs) are a class of Deep Neural Networks that have the ability to generate realistic synthetic data including but not limited to images, text, or even videos. The ability to learn and generate new data with the same statistics as the training set has made it the de-facto standard for generative networks, making older architectures (like autoencoders) obsolete. This paper investigates GANs from the ground-up, analysing various architectures , their advantages, caveats, and use cases in pursuit of understanding its widespread success.

1 Introduction

First proposed by Goodfellow et al. [1] , the GAN training loop involves two Deep Neural networks [2] - the generator and the Discriminator - that work against one another in a min-max game [3]. The Generator is tasked with the task of replicating the input data, using inputs from a latent vector [4] to generate data with the same dimensions as the training data. The Discriminator works as the "adversary", trying to differentiate between real data from the training set and the "synthetic" outputs from the Generator. As Goodfellow puts it - "The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency."

One thing worth mentioning is that the models are complementary, that is, the two models cannot work without one another. The output from the discriminator is used to optimize the generator's performance, creating data that becomes more representative of the training dataset. On the flipside, the new, generated data is used to improve the discriminator's performance. This feedback loop is best known as the min-max optimisation strategy.

[3].

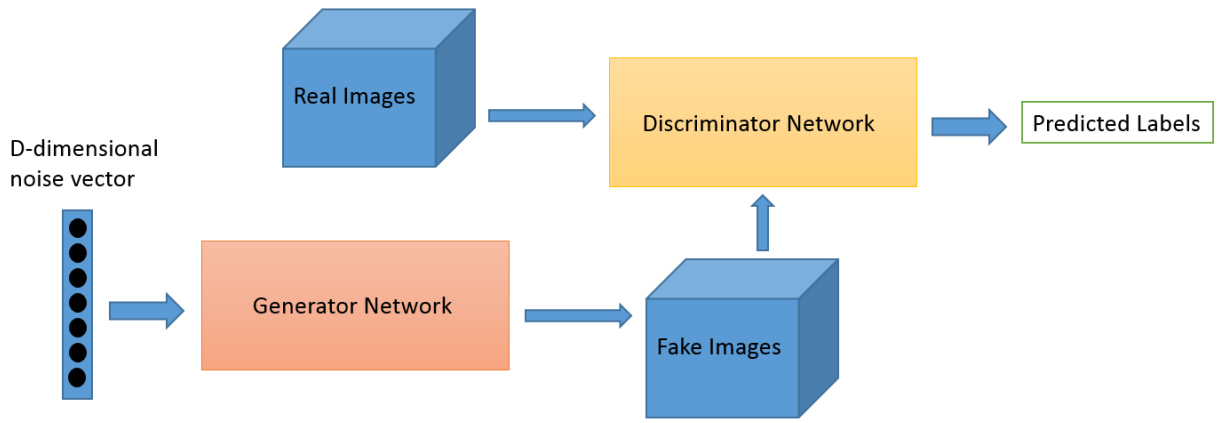

Figure 1: Overview of a simple GAN architecture

2 Different types of Unconditional GAN architectures

Ever since their inception in 2014, GANs have seen a lot of improvements and modifications. Be it new loss functions, training strategies or both, these models have been the forefront of many new technologies, and so it is important to carefully analyse each one of them.

2.1 DCGAN (2016)

Deep Convolutional GANs(DCGANs) are exactly what they sound like- They are generative networks almost entirely composed of Convolutional Neural Networks [5]. First introduced by Alec et al. [6], the DCGAN generator makes use of Transpose Convolution (or Upsampling) [7]layers to to produce an image from a seed (random noise). The Discriminator is just a standard binary CNN classifier with binary entropy as the loss function.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_data}[log(D(x))] + \mathbb{E}_{z\sim p_z}[1 - log(D(G(z)))]$$

where x is a sample image from the training data, z is the image seed, G and D are the Generator and Discriminator respectively.

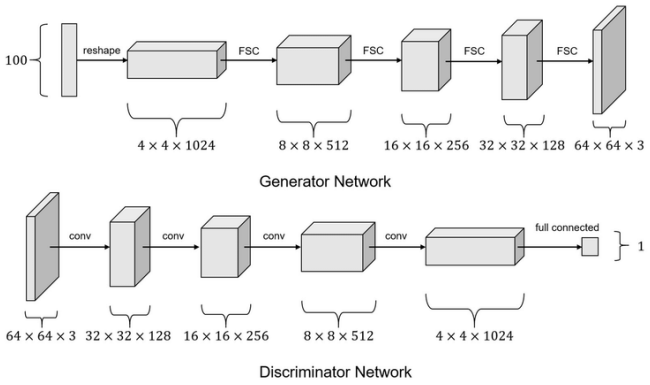

Figure 2: Architecture of a Deep Convolutional Generative Adversarial Network (DCGAN)

2.1.1 Drawbacks of DCGAN

Perhaps the biggest caveat is its susceptibility to mode collapse [8] , which refers to the inability of GAN to generate different class images. So if your data has N categories and your generator consistently produces images of $K < N$ classes, the model suffers a mode collapse.

Depending on the nature of the dataset, the DCGAN model may also completely fail to converge onto a global minima [9], leading to a Convergence failure. Unlike mode collapse, a convergance failures implies that the generator produces incomprehensible images.

These failures can be linked to the rigid model architecture, crude training strategy, and the loss function.

(a) convergence failure

(b) mode collapse

Figure 3: Common DCGAN failure modes on the MNIST dataset

2.2 ProGAN (2018)

To solve the problems introduced by DCGANs, Karras et al. [10] from NVIDIA introduced a progressively growing GAN model, most commonly known as Progressive GAN or a ProGAN. The idea is to first train the model at a lower resolution, and then progressively add new layers to capture finer details. This new training stratagem reduced training time and improved model stability to an unprecedented extent, which allowed the model to generate high-quality photo-realistic images even at 1024^2 resolution.

The real success of ProGANs can be attributed to three key components:

- Layer introduction Mechanism
- Minibatch standard deviation
- Pixel-wise Feature Normalization

2.2.1 Layer Introduction

The layer introduction mechanism refers to the process of smoothly adding new convolution layers in both the generator and the discriminator to increase the depth of the neural networks. This allows-

- the Generator to produce higher-resolution images
- the Discriminator to take these higher-resolution images as input and classify them as either real or fake

The end result is that the model gained incredible stability and convergence speed, and as per the authors, comparable result quality were often obtained up to 2–6 times faster than traditional GANs.

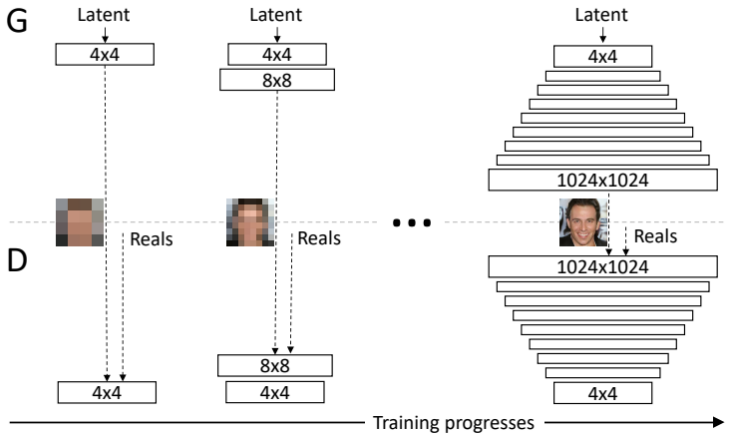

Figure 4: ProGAN architecture for 1024x1024 image with Layer Introduction Mechanism

2.2.2 Minibatch Standard Deviation

Following a paper on improving GAN performance by Salimans et al.[11] , the authors introduced a "Minibatch standard deviation" layer at the end of the Discriminator. In essence, the minibatch standard deviation layer updates the activation maps in the Discriminator by using their standard deviations and averaging them over the minibatch. This technique was groundbreaking as it provided a fail-safe against mode collapses [12], a phenomenon that DCGANs were prone to.

2.2.3 Pixel-wise Feature Normalization

To disallow the scenario where the magnitudes in the generator and discriminator spiral out of control as a result of competition, the authors normalised the feature vector in each pixel to unit length in the generator after each convolutional layer [13]. While this technique doesn't actually help the model converge faster, it does prevent the model output from explosively increasing.

2.2.4 Drawbacks of ProGANs

Even though Progressive GAN undoubtedly overcame all of the flaws that DCGAN possessed, in the process it brought one of its own - latent space entanglement[14]. An entangled representation identifies latent factors that each map to more than one aspect of the generative process. In other words, in an entangled latent space, one or more image features can be encoded in a single dimension. This was a clear sign that we still didn't have much control over the Generator Architecture and the latent space [15].

2.3 StyleGAN (2019)

While ProGANs revolutionised the training schema and Discriminator architecture, the Generator was still largely unexplored. So Karras worked on improving the Generator Architecture, taking inspiration from neural style transfer [16]. The new and improved model was thus dubbed StyleGAN. [17].

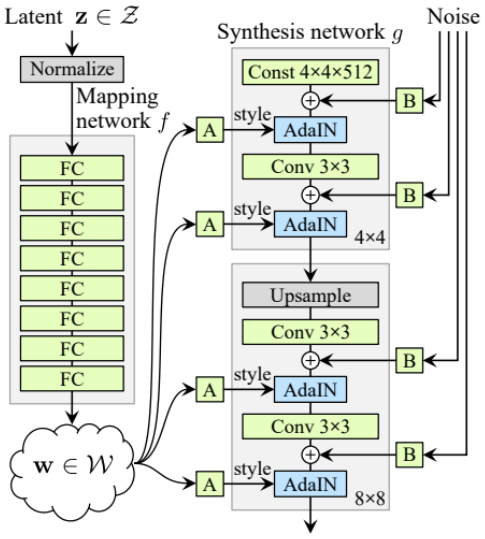

Figure 5: StyleGAN generator. The network on the left creates an intermediate latent vector w that controls the generator through the AdaIN layer. A is a learned affline transformation network, and B is a learned Noise controller. Output from the noise controller is concatenated with the network as x_i

The StyleGAN generator took a massive detour from the traditional Generator architecture by omitting the input layer altogether and starting from a learned constant instead. The Latent seed $z \in Z$ was passed through a a non linear mapping network f: $Z \rightarrow W$ to give $w \in W$, which was then introduced to the generator through an Adaptive Instance Normalization(AdaIN) layer. Instance Normalization is largely used in neural style transfer [16].

$$\text{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i} \frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i},$$

Where xi is the Convolutional Output, and $y = (y_s, y_b)$ is an Affline transform[18] of w. To put it simply, Adaptive Instance Normalization is a normalization method that aligns the mean and variance of the content features with those of the style features. These design changes, especially the mapping network and the AdaIN layer, minimise latent space entanglement, while improving the image quality drastically.

2.3.1 Drawbacks of StyleGANs

The complex architecture of StyleGAN fixed most of the fundamental problems of the preceding models. But the generator was still prone to visual artefacts [19] as shown in $Figure6$ and $Figure7$. These drop-like artefacts were mostly observed when the model reaches 64x64 resolution or more.

Figure 6: Drop-like artefacts in generated images

Figure 7: Other visual artefacts in the CelebA dataset. Notice the lack of symmetry and visible marks in ears. There's also some artefacts in the eyes, glasses, and lips

2.4 StyleGAN 2 (2020)

In an attempt to improve the StyleGAN model, Karras et al. [20] revised the architecture, normalization techniques and training schema of the generator to address its shortcomings.

As it turns out, the visual artefacts could be traced back to the AdaIN operation. In the words of the author - "We pinpoint the problem to the AdaIN operation that normalizes the mean and variance of each feature map separately, thereby potentially destroying any information found in the magnitudes of the features relative to each other". This hypothesis was supported by the fact that the artifact completely disappeared once the Normalization function was removed.

So the solution was simple. To achieve better results the authors needed to -

- Replace AdaIN with a demodulation operation and normalization layer to update the feature maps
- Add bias and the Noise Controller output (B) after the normalization layer. This preserves the "style" of the said convolutional block as now it's only purpose is to modify the standard deviation of the activation map.

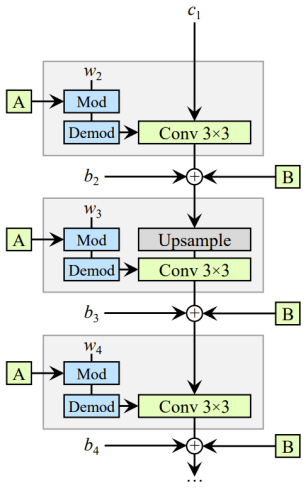

Figure 8: Convolutional block, or "style" blocks from StyleGAN2 generator. c_1 is a constant and b_n are the bias values

This improved architecture finally solved all shortcomings of previous GAN models, creating high-quality, photo-realistic images with no observable artefacts. Some of the results can be seen in $Figure 9$

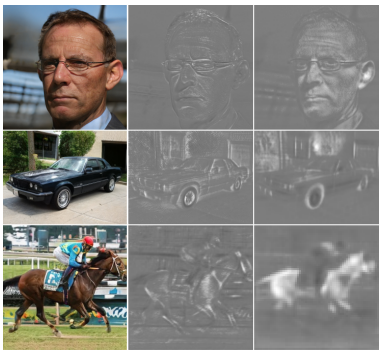

Figure 9: Sample output from a pretrained StyleGAN2 Generator

3 Conditional GAN (2014)

Even though models such as ProGAN [10], StyleGAN [17], and StyleGAN2 [20] showed unprecedented image generation qualities in an unsupervised environment, they lacked control over the type of generated image. Therefore Mirza et al. [21]proposed a new technique that allowed greater control over image features by including class labels.

Simply put, these networks made use of the class labels in the dataset, concatenating the label data to both the Generator and Discriminator respectively. This allowed us to Conditionally Generate images, giving us greater control over the Generated Output.

The cGAN loss function was almost identical to the binary cross-entropy loss proposed in the original GAN paper [1], with some changes to accommodate the class labels y:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_data}[log(D(x|y))] + \mathbb{E}_{z\sim p_z}[1 - log(D(G(z|y)))]$$

In the Generator, the latent space vector was concatenated with the class label vector(see Figure 10) y. The resultant vector becomes the true hidden representation for the Generator.

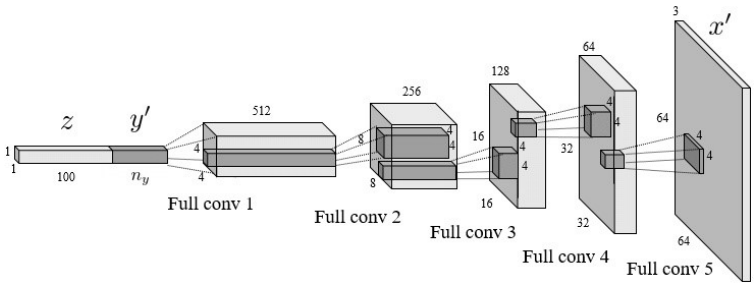

Figure 10: Sample cGAN Generator for image editing

On a similar note, the Discriminator was updated to a Multi-input model , that accepts both Images and and label vectors as inputs.

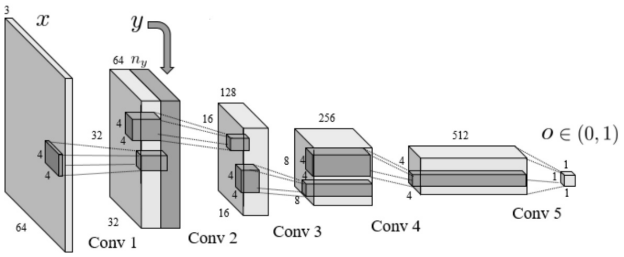

Figure 11: Sample cGAN Discriminator for image editing

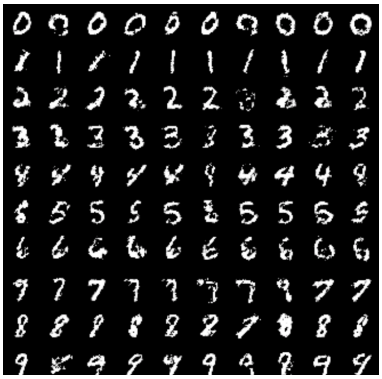

Figure 12: Unimodal conditional generation on the MNIST dataset

Conditional Generation was perhaps the most important aspect of current-day GANs, as they enhance model functionality and help in multiple use cases, including but not limited to - super resolution, image-to-image conversion, image captioning, text-guided image generation, image manipulation, etc. We will discuss some of these use cases in this section.

3.1 Resolution Enhancement/Super Resolution GAN (2017)

The task of estimating a high-resolution image from its low-resolution counterpart is referred to as super-resolution. While many techniques such as bicubic interpolation [22] and directional bicubic interpolation [23] exist, they fail to capture finer details in the image.

Super-Resolution GAN (SRGAN) [24] tackled this issue by modifying a ResNet-based [25] DCGAN generator such that low-resolution images can be fed as the input seed for the Generator. The authors also proposed an interesting loss function:

$$l = l_{vgg} + \alpha * l_{adversary}$$

where $l_{adversary}$ is the adversarial loss of the GAN architecture (determined by the Discriminator), α is an arbitrary constant (assumed 10^{-3} by the authors). and l_{vgg}, or the content loss, is the euclidean distance between the feature representations of the reconstructed image and the reference image, determined from a pretrained vgg54 network [26].

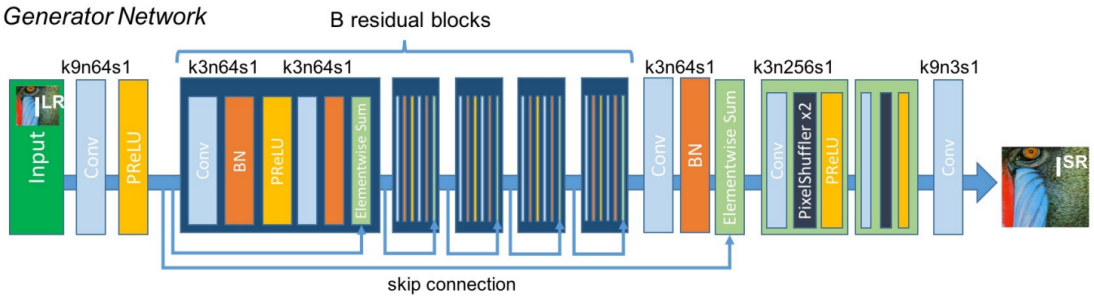

Figure 13: SRGAN generator. Notice the residual blocks and skip connections

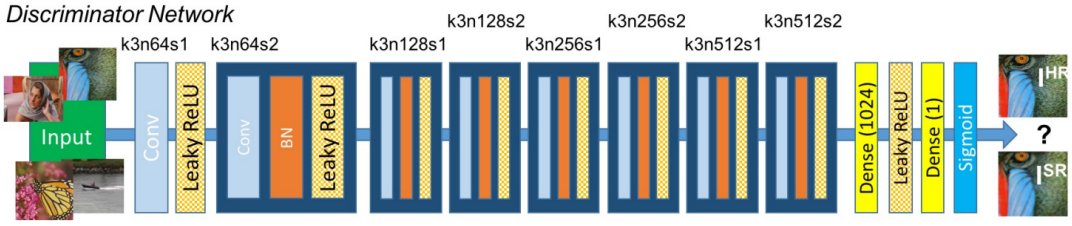

Figure 14: SRGAN discriminator. In comparison to the Generator, the discriminator is fairly simple

SRGANs proved to be incredibly versatile, being able to generate high-quality images upto 4x upscaling factor with little to no visual artefacts. This technology still finds active use in medical imaging, satellite imaging, surveillance and security, astronomical imaging, amongst others.

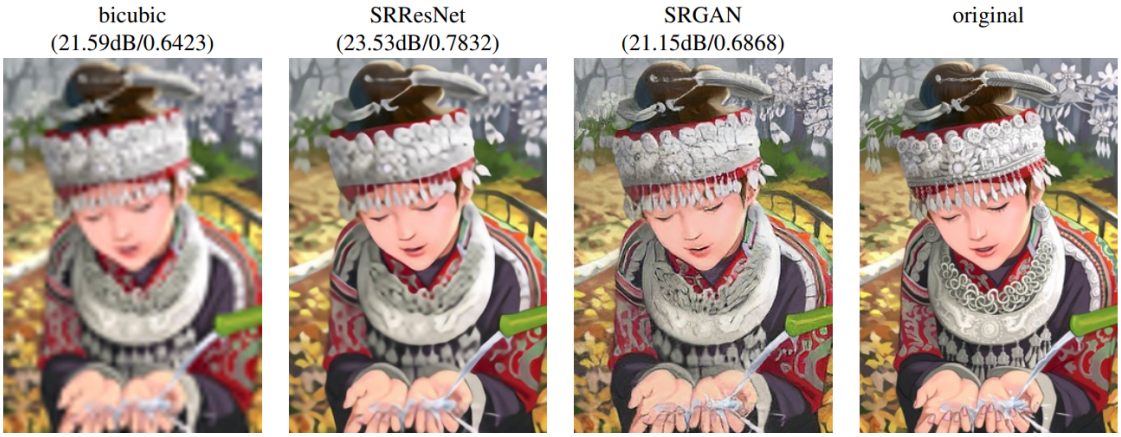

Figure 15: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image SRGAN is able to reconstruct finer features without compromising on the bigger details

3.2 Pix2Pix GAN (2018)

First proposed by Isola et al. [27]from UC Berkley, Pix2Pix GAN is an Image-to-Image translation model that can learn interesting mappings from one image to another. These networks not only learn the mapping from input image to output image, but also learn a loss function to train this mapping. This makes it possible to apply the same generic approach to problems that traditionally would require very different loss formulations. It can be applied to a wide range of tasks, including synthesizing photos from label maps, generating colorized photos from black and white images, and even transforming sketches into photos

The incredible success of the model lied in its architecture. The Generator model is a U-Net [28], a popular image-segmentation model. In essence, U-Net is an encoder-decoder model with skip connections.

The loss function proposed by the authors looked something like this:

$$l = arg\min_{G}\max_{D}\mathcal{L}_{cGAN}(G, D) + \lambda\mathcal{L}_1(G)$$

Where $\mathcal{L}_1(G)$ represents $\mathcal{L}_1$ or the Manhattan distance [29] between the generated image and the ground truth. λ is just a constant.

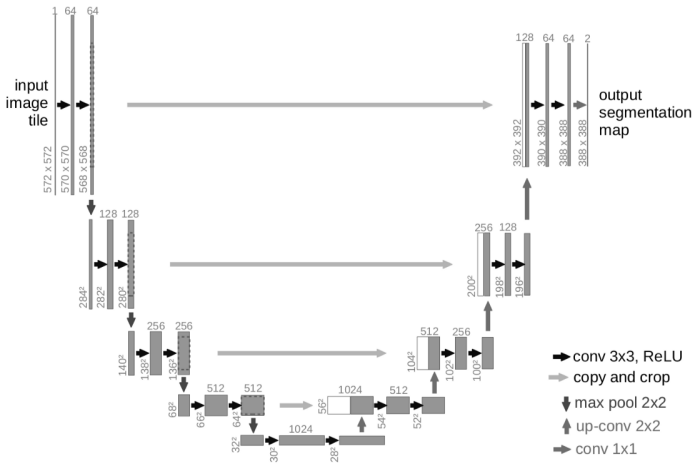

Figure 16: U-Net architecture

The Discriminator architecture was rather unique. Instead of running on the entire image, the Discriminator ran patchwise, which means that it penalized the structure at the scale of local image patches. The penalties were then averaged to give the final result. Due to this unique behaviour, the Discriminator was dubbed PatchGAN [30].

The combination of a Versatile Generator, carefully designed loss function, and a sensitive Discriminator make Pix2Pix GANs incredibly powerful for all types of image-to-image translation.

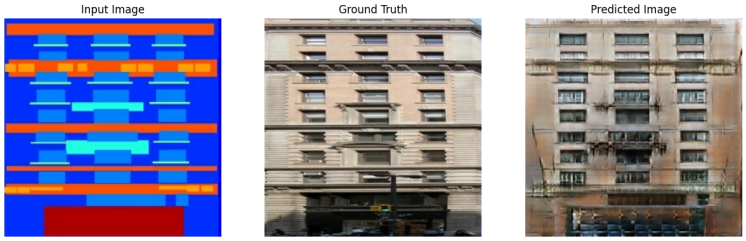

Figure 17: Image-to-image translation by the Pix2Pix Generator

4 Conclusion

Generative Adversarial Networks are incredibly powerful and versatile given their ability to learn (almost) any data distribution from the given training data. This generalization ability has given researchers and the Deep Learning community the ability to do things that were previously deemed impossible. GANs (and generative networks in general) have created new Deep Learning applications that expand beyond the generic classification/regression problem, and in the process, have raised the stakes to new levels. To call GANs influential would be an understatement- their inception was quintessential in the expansion of Generative modelling. This paper is meant to be treated as introductory literature for researchers who want to get into this field. So we carefully analyse the most influential GAN algorithms- their architectures, losses, and pros and cons. In the process we also discuss the most common GAN errors (mode collapse, convergence failures, latent space entanglement) and how each subsequent model tackles these problems. As a final step in appreciating the versatility of these models we try to understand conditional generators and two popular conditional architectures, namely SRGAN and Pix2Pix GAN.

References

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[2] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[3] Hassene Aissi, Cristina Bazgan, and Daniel Vanderpooten. Min–max and min–max regret versions of combinatorial optimization problems: A survey. *European journal of operational research*, 197(2):427–438, 2009.

[4] Zachary C Lipton and Subarna Tripathi. Precise recovery of latent vectors from generative adversarial networks. *arXiv preprint arXiv:1702.04782*, 2017.

[5] Wei Zhang, Jun Tanida, Kazuyoshi Itoh, and Yoshiki Ichioka. Shift-invariant pattern recognition neural network and its optical architecture. In *Proceedings of annual conference of the Japan Society of Applied Physics*, pages 2147–2151, 1988.

[6] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[7] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[8] Hoang Thanh-Tung and Truyen Tran. Catastrophic forgetting and mode collapse in gans. In *2020 international joint conference on neural networks (ijcnn)*, pages 1–10. IEEE, 2020.

[9] Jerry Li, Aleksander Madry, John Peebles, and Ludwig Schmidt. On the limitations of first-order approximation in gan dynamics. In *International Conference on Machine Learning*, pages 3005–3013. PMLR, 2018.

[10] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[11] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

[12] Arnab Kumar Mondal, Vineet Jain, and Kaleem Siddiqi. Mini-batch graphs for robust image classification. *arXiv preprint arXiv:2105.03237*, 2021.

[13] Yoon-Jae Yeo, Min-Cheol Sagong, Seung Park, Sung-Jea Ko, and Yong-Goo Shin. Image generation with self pixel-wise normalization. *arXiv preprint arXiv:2201.10725*, 2022.

[14] Jacob C. Kimmel. Disentangling a latent space.

[15] Karn N Watcharasupat and Alexander Lerch. Evaluation of latent space disentanglement in the presence of interdependent attributes. *arXiv preprint arXiv:2110.05587*, 2021.

[16] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pages 1501–1510, 2017.

[17] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[18] Eric W Weisstein. Affine transformation. *https://mathworld. wolfram. com/*, 2004.

[19] Mohammad Hoque, Md. Sadek Ferdous, Mohsin Khan, and Sasu Tarkoma. Real, forged or deep fake? enabling the ground truth on the internet. *IEEE Access*, PP:1–1, 11 2021.

[20] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020.

[21] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[22] Anil Gavade and Prasana Sane. Super resolution image reconstruction by using bicubic interpolation. 10 2013.

[23] Jing Liu, Zongliang Gan, and Xiuchang Zhu. Directional bicubic interpolation. 11 2013.

[24] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[26] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[27] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[28] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[29] Susan Craw. *Manhattan Distance*, pages 639–639. Springer US, Boston, MA, 2010.

[30] Thittaporn Ganokratanaa, Supavadee Aramvith, and Nicu Sebe. Unsupervised anomaly detection and localization based on deep spatiotemporal translation network. *IEEE Access*, PP:1–1, 03 2020.