Towards Data and Energy Efficient GANs Using Decoders

Mara-Iuliana Dragomir
2727283@student.vu.nl

Pawel Piwowarski
2729221@student.vu.nl

Adam Rydzinski
2739038@student.vu.nl

Ioana Song
2730786@student.vu.nl

Teodora Taran
2737833@student.vu.nl

March 2023

Abstract

In this paper, we want to implement our own GAN, heavily based on previous research, for economical model training with minimized computational cost as well as a minimal few shot technique for minimising the required training data set. We will modify the already existing GAN architecture with additional decoders added to the Discriminator network and measure the results of the changes based on FID and LPIPS scores for the modelled distribution. We conclude that adding more decoding techniques to the discriminator can be overall beneficial for the results of the GAN game. The code for the paper is available at[1].

1 Introduction

In recent years, we have seen a worldwide rise in the usage of generative models. A straightforward example is OpenAI's Chat-GPT model for generating human-like text [1]. A special example of generative models are implicit latent variable models, which can model arbitrary data from latent variables[2].Especially in the field of generating images, one can note the rapid development of projects such as DALL E2.[3]. Training of latent variable models such as diffusion-based models or GANs suffers, however, from enormous computational cost, massive data set need, and training instability [4]. In this paper, we want to address some of the problems associated with training one of these latent variable models, the GAN network. More importantly, we want to study the results of the so-called GAN game, which turns out to be adversarial in nature, and propose our way of improving the results.

[1] https://github.com/pawelpiwowarski/Fast_GAN_Vrije

1.1 GAN Game

In most general terms, the result of the GAN game is defined as training a model G, let's call it a generator, that learns an implicit operation on some latent variable, f.e. a vector. $\vec{v}$ sampled from some distribution, f.e $N(0, I)$, transforming this vector into data coming from the modelled distribution X. However, Generator on its own is unable to achieve this; it needs another adversarial model, let's call it Discriminator. The purpose of D is to judge the output of G and output a probability of how likely it is that the output comes from the modelled distribution. X. This adversarial approach leads to a clear MinMax problem. Let's call it a function. $V(D, G)$ and that how it has been stated in the original GAN paper [5]. Taking our initial notation it can be written as:

$$\min_G \max_D V(D, G) = E_{x \sim X}[\log D(x)] + E_{z \sim N(0,I)}[\log(1 - D(G(z)))] \quad (1)$$

Essentially, this can be seen as the loss function for our two neural networks, and we can train G and D to minimize and maximize the loss, respectively, using backpropagation and stochastic gradient descent until reaching a theoretical Nash equilibrium. However, this approach is far from perfect. As we can read in the seminal work about GANs published in 2019 [6], this MinMax aproach is not used in practice often and that a different strategy called a Non-Saturating loss function for the Generator is used, it breaks up the one function MinMax approach into two separate loss functions for the networks. However, as the authors write, NS-GAN is a heuristic approach; theoretically, it shouldn't converge to Nash equilibrium, but in practice it does. In our view, this example underlines two important problems that play the biggest part of our research.

Firstly, as shown by the non-saturating GAN, the main issue is that of gradient saturation or vanishing; in short, D does not provide sufficient learning incentive gradients for G, thus G cannot learn anything and the GAN does not converge. Secondly, Previous academic research was mainly centered around different loss functions for GANs; as such, the architecture of the neural networks was most often marginalized and researchers focused much attention on mathematical guarantees and correctness, but as the example of NS-GAN shows, mathematical correctness sometimes doesn't lead to overall beneficial results for the GAN game.

1.2 Vanishing Gradients

The problem of vanishing gradients can be best described as a question of imbalance between the capabilities of the discriminator and the generator's abilities. It leads to the discriminator becoming too powerful at categorizing the data coming from the generator, or, in short, D overfits. This leads to no learning incentive for G, as the gradients get close to zero. The other way is also possible: G becomes too good at fooling D, and thus the learning does not progress. The example of vanishing gradients clearly illustrates how important the role of the discriminator is in the GAN game; it needs to constantly provide informative gradients to the generator while, on the other hand, not overfitting.

2 Related Work

As mentioned previously, past research focused on theoretical aspects of the GAN game, i.e., loss functions. We can note two major breakthroughs in research. The first was the work of Arjovsky et al. [7], in their work the researchers proposed to use the Wassertein distance for the loss function of the Discriminator which lead to beneficial results, the authors of [8] "Improved Training of Wasserstein GANs" made an additional improvement of introducing a concept of gradient penalties which normalised the gradients making the loss a 1-Lipschintz differentiable function, subsequently stabilising the training and convergence of GANs. In 2018 researchers at NVIDIA made Generator grow progressively , they also added the improvement of equalised learning rates. [9]. The 2018 and 2019 [9] [10] works the NVIDIA team shifted the focus away from loss functions onto the architecture of the networks. Developing the idea of focusing on the architecture, the work by Liu et al. [11], explored improving GANs via unconventional ways (Skip-Connections and Decoders), their research into data and energy efficient GANs is the starting point for our paper.

3 Design Choices

3.1 Architecture over loss function

The base of our network was a direct port of the architecture proposed by Liu et al. [11], with the addition of porting the architecture from pytorch to tensorflow, for better performance on M1 chips [12]. The cores of the architecture proposed Lie et al [11] is its simplicity and robustness.

3.2 Generator Architecture

The main part that plays a beneficial role and helps alleviate the problem of vanishing gradient flow in the generator is the so-called skip layer channel-wise excitation. This connection is a bridge connection between two output tensors of different sizes. The skip connection works by multiplying the tensors channel-wise, i.e., only the channel dimensions. Figure 1 explores the idea in more detail. The benefit of this approach is that it is computationally less expensive than element wise operations. The rest of the generator's network is a standard Generator architecture with the addition of Gated Linear Unit as an activation between some of the upSampling layers. The benefits of Gated Linear Unit have first been observed in [13] language based models, their properties also include providing a better gradient flow through the network. Additionally Noise Injection layer is added between every second upSampling layer, as adding noise has been proved beneficial for the entire GAN game [14]. Simplified version of Generator's architecture can be seen in figure 2. More detailed description of the Generator's architecture is to be found in [11]

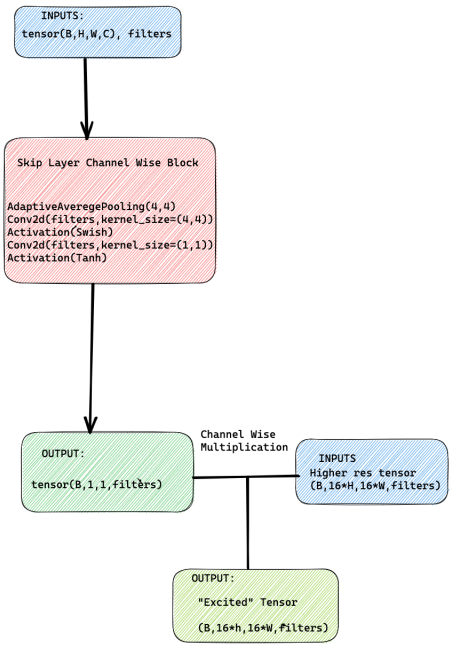

Figure 1: Skip layer connection

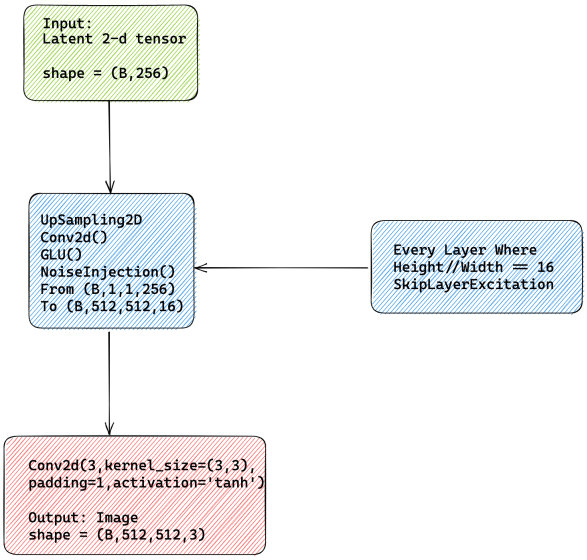

Figure 2: Simplified Generator's Architecture

B - Batch Size, H - Image Height, W - Image Width, C - Number of Channels

3.3 Discriminator Architecture

The Discrimniator network can be divided into two sub-networks. The first is responsible for providing logs corresponding to the presumed value of authenticity of the images, a set of values between 0 and 1. The closer to 1, the more "real" the data is, and vice versa. The second part of the discriminator network is the decoder, which is responsible for converting the intermediate feature maps to reconstructions based on various data coming from the real images. This approach can best be described as a semi-self-supervised approach to training GANs; the idea behind it can be traced back to the original cGAN architecture.[15] and more later to [16]. However unlike the previous approaches the FastGAN proposed by Liu et al [11], applies supervision to the Discriminator and not the Generator network. The idea behind this supervision is simple we train a decoder that when given a feature map let's call it f_1, results in a reconstruction of the original real image but resized to the resolution of the decoded image. In this example there is no other operation on f_1 before passing it to the decoder, but any operations are allowed. It is important to note that this can be seen as a supervised approach because the decoding is done only on the real samples and thus we are implicitly labeling the data. For a clearer picture of the idea see Figure 3.

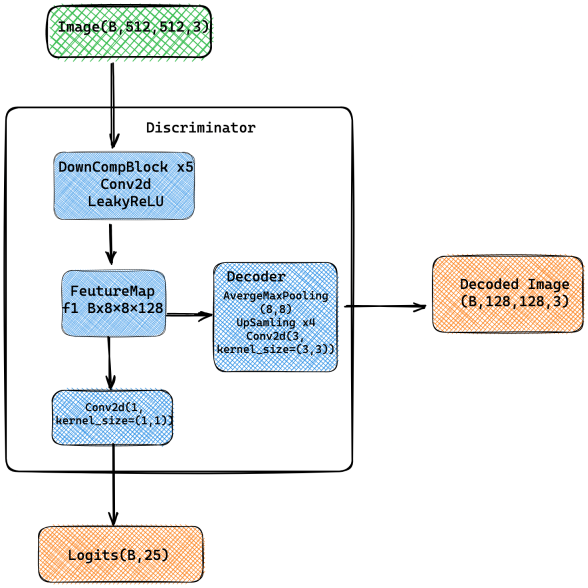

Figure 3: Decoder Overview

3.4 Data Augmentation and Loss Functions

One of the goals when implementing the FastGAN architecture is efficient treatment of data. The reason is that when dealing with a limited dataset, it is very hard to avoid mode collapse since the number of trainable examples is much too low. One of the most recent ways of mitigating this issue was described in a paper by Zhao et al.[17]. Differentiable augmentation proposed by the team augments both real and fake samples with random noise and resizing. The idea is to make a small dataset scalable for the GAN game. This greatly helps with making the GAN data efficient.

When it came to the question of loss functions we experimented with using both the cross-entropy and hinge loss. We noted however similarly to [11] that using a non-saturating version of the hinge loss, provides the most stability and fast convergence. The equation of the loss for the discriminator part of the network can be written as follows with respect to our original notation.

$$L_D = E_{x\sim X} \min(0, -1 + D(x)) + E_{z\sim N(0,I)} \min(0, 1 - D(G(z)) \tag{2}$$

We speculate that hinge loss perform better than other loss function in the course of the GAN game because of its lower computational cost which has been described in [18]. One can also use gradient penalty as on option to normalize the gradients of D if possible, but we found that implementing it lead to to much computation cost and we judged it to be infeasible. Additionally to the discriminator's loss another loss function for the decoder had to be provided; it can be denoted as.

$$L_{\text{Deocder}} = \sum_{i=1}^{n} E_{f_i\sim D_{\text{encode}}(x), x\sim X} \left|S_i(f_i) - T_i(x)\right| \tag{3}$$

Where n represents the number of feature maps to be decoded, function S_i represents an operation on a feature map f_i such as decoding and f.e cropping. Function T_i represent the equivalent operation on real sample x corresponding to the real image. The loss of the decoder utilises the learned perceptual distance metric, between two images as implemented in the work by Zhang et al[19]. Taking both losses into one eqaution the total loss for the Discriminator can be written as

$$L_{totalD} = L_D + L_{Decoder} \tag{4}$$

As for the Generator's loss function a standard non saturating version of loss of the Discriminator has been applied.

$$L_G = -L_D \tag{5}$$

4 Experimental Setup

We will conduct three experiments, in all of the experiments we will add new feature map f_i, function S_i and T_i (equation 4), and leave all of the other parameters unchanged. In more concrete terms the function S_i represents the processing on the intermediate feature map of D, and the reconstruction provided by the decoder part of D.

$$|S(f_i) - T(x)| \tag{6}$$

In the first experiment the we will only use the feature map f_1, we will simply decode the intermediate feature map f_1, to (Bx128x128x3) and measure the loss with respect to the original image but resized to 128x128.

In the second experiment we will add f_2, with cropping that is the intermediate feature map f_2 of size (Bx16x16x64), is cropped based on the part parameter to a tensor sized (Bx8x8x32). This operation on the feature map can be denoted as $P(f_2)$, where P is the cropping function. Then the result is fed to the decoder an the output is a reconstruction of the part of the image sized Bx128x128x3.

In the third experiment will add another intermediate feature map f_3 sized Bx32x32x32 from which a feature centered intermediate feature map will be taken corresponding to the central part of the image. The processing which we can denote as $C(f_3)$ produced a tensor sized Bx8x8x32 which is fed to the decoder and the loss with respect to the central part of the image is calculated.

The training dataset consisted of 768 images of oil paintings painted by Vincent Van Gogh [20]. We wanted a dataset that was limited in size and displayed a unique and hard to randomly reproduce style. All of the models have been denoted correspondingly: Model 1(Baseline), Model 2, Model 3. We speculated that after training each model will have a different degree of details i.e the more feature maps which reconstruct part of the image the more detail in the Generator's outputted distribution. The exact time when to stop training GANs is extremely difficult to estimate we have decided in parts considering our limited resources that after noting that the reconstruction loss stays stagnant for a longer period of time we stopped the training and generated 50 samples from Generator and measured its FID and LPSIPS metric with respect to the originals samples. All of the training loops were trained using Google GPU accelerated Colab [21] notebooks, and each took around 5-6 hours to complete.

5 Summary and Findings

The first model that we evaluated was model 1 (Baseline) with one decoder used. The summary of the model can be found on (Figure 4,5). We noted that the Generator's output captured the background of the paintings quite well but failed at getting any details into the picture f.e faces or hands. Additionally we noted that the reconstruction loss quickly went stagnant after 200 epochs.

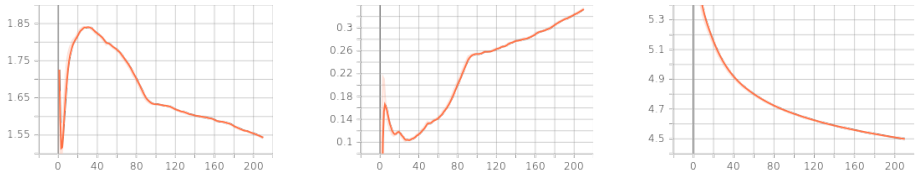

Figure 4: D loss, G loss, Rec Loss - Model 1(Baseline) - 200 Epochs

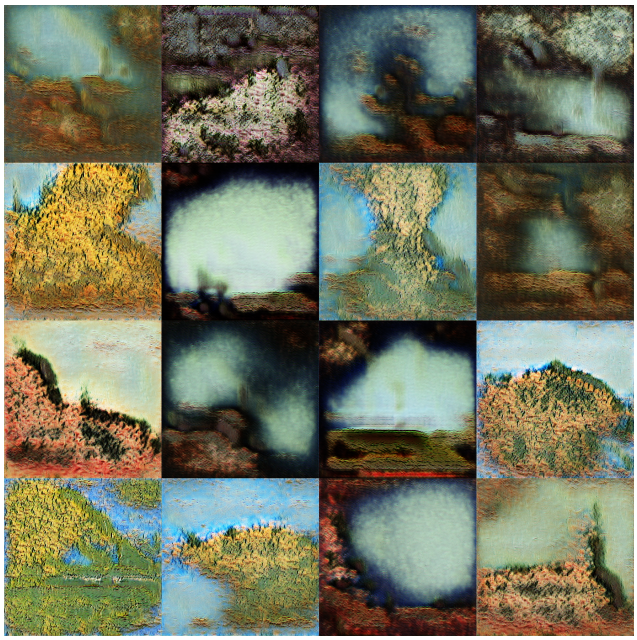

Figure 5: Generated Images after 200 epochs - Model 1

The second model that we evaluated was model 2 that used two decoders (figures 6,7). We noted that the images became more detailed with the Generator's ability to reconstruct details of trees or outlines of faces improving.

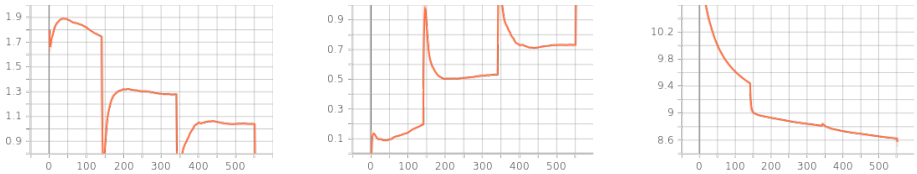

Figure 6: D loss, G loss, Rec Loss - Model 2- 500 Epochs

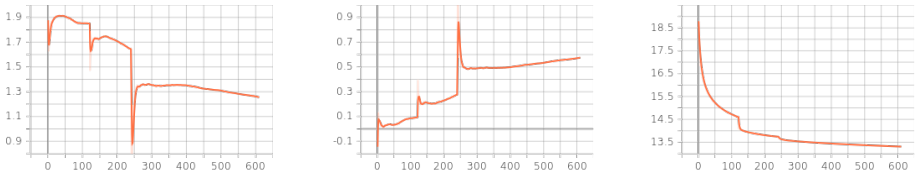

Figure 8: D loss, G loss, Rec Loss - Model 3- 500 Epochs

Figure 7: Generated Images after 300 epochs - Model 2

The third model that we evaluated was model 3 that used three decoders (figures 8,9). The images still lacked much detailed and the Generator's ability to produces original samples improved slightly.

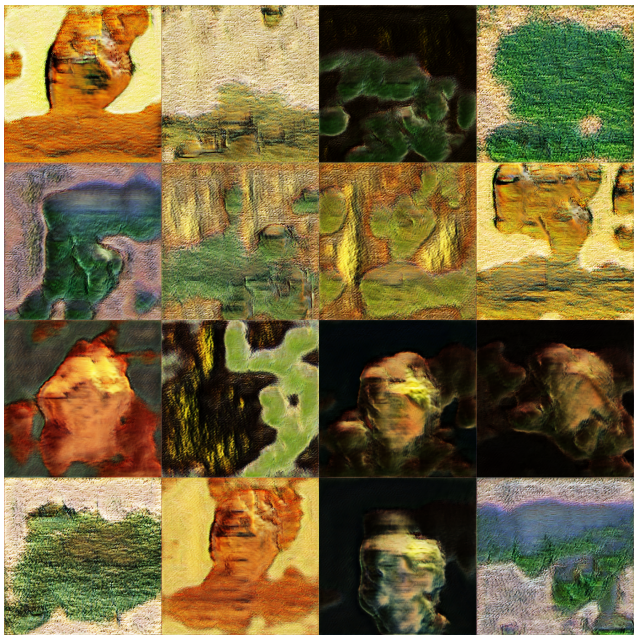

Figure 9: Generated Images after 300 epochs - Model 3

We used both FID metric [22] and LPIPS [19] for measuring the quality of generated samples. We noted a big improvement compared to the baseline model with using two or three decoders. The images became much more detailed, and the FID and LPIPS scores correspond to that. We noted no significant difference between model 2 and model 3, we speculate the feature map f_3 lies to deep in the Discriminator network and as such its weights do not make much difference in the long run, simply for the reason of gradient flow.

	FID	**LPIPS**
Model 1 (Baseline)	317.997	0.7787
Model 2 (D+Crop)	288.366	0.7804
Model 3(D+Crop+Center)	295.675	0.7679

Table 1: Evaluation Of The Models (The lower the score the better)

6 Future Works

Improving FastGAN through using more decoders is a question left open, perhaps growing the number of decoders horizontally rather than vertically can bring better results than the ones we observed.

New research suggests that by using transformer based architectures [23], replacing convolutions can greatly increase the quality of the synthesises images. An approach worth considering is mixing the convolution approach with the transformer based approach. Another interesting idea worth exploring proposed by Dhariwal and Nichol [24], is getting rid of the GAN adversarial approach in favour of a more principal approach of diffusion models. We can clearly note the disadvantages of GAN based architecture like: instability of training, mode collapse while on the other hand no other approach has allowed for image generation with so few resources. We believe tgat the question of one general solution for image generation will be an interesting one for the researchers in the years to come

References

[1] *OpenAI Chat*. https://chat.openai.com/chat.

[2] Jakub M. Tomczak. *Deep Generative Modeling*. Springer, 2022, pp. 57–58. ISBN: 978-3-030-93158-2. URL: https://doi.org/10.1007/978-3-030-93158-2.

[3] *DALL.E2*. https://openai.com/product/dall-e-2.

[4] Ling Yang et al. "Diffusion Models: A Comprehensive Survey of Methods and Applications". In: *arXiv preprint arXiv:2209.00796* (2022), p. 22.

[5] Ian Goodfellow et al. "Generative Adversarial Networks". In: (2014). URL: https://doi.org/10.48550/arXiv.1406.2661.

[6] Jakub Langr and Daniel J Bok Vladimir. *GANs in action*. Manning Publications, 2019. ISBN: 9781617295560.

[7] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein GAN". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 214–223.

[8] Ishaan Gulrajani et al. "Improved training of Wasserstein GANs". In: *Advances in Neural Information Processing Systems* 30 (2017), pp. 5767–5777.

[9] Tero Karras, Samuli Laine, and Timo Aila. "Progressive growing of GANs for improved quality, stability, and variation". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=Hk4b5yWCW.

[10] Tero Karras, Samuli Laine, and Timo Aila. "Analyzing and Improving the Image Quality of StyleGAN". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8110–8119.

[11] Bingchen Liu et al. "Towards faster and stabilized GAN training for high-fidelity few-shot image synthesis". In: *IEEE Transactions on Image Processing* 30 (2021), pp. 5738–5752. DOI: 10.1109/TIP.2021.3084234.

[12] Pawel Piwowarski. *Full Code of The 3 models*. `https://github.com/pawelpiwowarski/Fast_GAN_Vrije`. Accessed: March 29, 2023. 2021.

[13] Yann N. Dauphin et al. "Language Modeling with Gated Convolutional Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. 2017, pp. 933–941.

[14] Tim Salimans et al. "Improved Techniques for Training GANs". In: *Advances in Neural Information Processing Systems*. 2016.

[15] Mehdi Mirza and Simon Osindero. "Conditional generative adversarial nets". In: *arXiv preprint arXiv:1411.1784* (2014).

[16] Julius Monello Adeel Mufti Biagio Antonelli. "Conditional GANs For Painting Generation". In: *arXiv preprint arXiv:1903.06259*. 2019.

[17] Shengyu Zhao et al. "Differentiable Augmentation for Data-Efficient GAN Training". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 340–348.

[18] Rayan S. Wali. "Xtreme Margin: A Tunable Loss Function for Binary Classification Problems". In: *Cornell University Research Paper Machine Learning Artificial Intelligence* (2022). URL: `https://arxiv.org/pdf/2211.00176.pdf`.

[19] Richard Zhang et al. "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2018, pp. 586–595.

[20] Pawel Piwowarski. *Oil dataset for Fast GAN Vrije*. `https://github.com/pawelpiwowarski/Fast_GAN_Vrije/tree/main/data/oil`. Accessed: March 29, 2023. 2021.

[21] Google. *Google Colaboratory*. `https://colab.research.google.com`. accessed on 20 March 2023.

[22] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. "On Aliased Resizing and Surprising Subtleties in GAN Evaluation". In: *CVPR*. 2022.

[23] Yifan Jiang, Shiyu Chang, and Zhangyang Wang. "TransGAN: Two Pure Transformers Can Make One Strong GAN, and That Can Scale Up". In: *arXiv preprint arXiv:2102.07074* (2021).

[24] Prafulla Dhariwal and Alex Nichol. "Diffusion models beat GANs on image synthesis". In: *arXiv preprint arXiv:2105.05233* (2021).