Federal State Autonomous Educational Institution for Higher Education
National Research University Higher School of Economics

Faculty of Computer Science
Applied Mathematics and Information Science

BACHELOR'S THESIS

Research project

"Training Generative Adversarial Networks with Adversarial Attacks"

Prepared by the student of group 193, 3rd year of study,
Pirogov Vyacheslav Grigoryevich

Supervisor:
Visiting Lecturer, Alanov Aibek

Moscow 2022

Contents

Abstract

First Generative Adversarial Networks (GANs) appeared only a few years ago, but they quickly start to show the best results in the image generation, and also quite good for creating sound. At the same time, GANs are flexible tool, which can generate a variety of entities. There are many ways to make the GAN work better, and often training is slow or perform badly, making it worth to use absolutely everything that could help. Training with Adversarial Attacks is an effective, albeit time-consuming, method which this work describes. All code is written in Python, models are created using PyTorch version 1.10.

Аннотация

Первые Генеративно Состязательные Сети появились всего несколько лет назад, однако они быстро стали показывать лучшие результаты по генерации картинок, а также весьма неплохие для создания звука. При этом данные модели являются гибким инструментом, с помощью которых можно генерировать самые различные сущности. Существует множество способов заставить Генеративно Состязательную Сеть работать лучше, при этом зачастую обучение проходит медленно, либо показывает плачевные результаты, из-за чего стоит использовать абсолютно все, что может помочь. Обучение при помощи Состязательных Атак как раз и является полезным, хоть и трудоемким способом, которое и будет рассмотрено в этой работе. Весь код написан на Python, модели созданы с помощью PyTorch версии 1.10.

Keywords

DL, GAN, Adversarial Attacks, FGSM, DCGAN, SNGAN, WGAN

1 Introduction

1.1 Description of the subject area

1.1.1 Generative Adversarial Networks

The first article about Generative Adversarial Networks by I. Goodfellow et al. (2014), describes a new generative architecture, which trains via an adversarial process. Authors propose to train two models at the same time. One of them is called Generator (G), and the other is Discriminator (D). G is trying to capture the distribution of the dataset, D aims to estimate the probability that a sample came from the training data rather than G. As a result, D and G are trying to solve the minimax problem with value function $V(G, D)$:

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

In the image 1.1 we can see an example of generated numbers for the MNIST dataset.

Figure 1.1: Example of generated images by GAN on MNIST dataset from "Generative Adversarial Nets" by I. Goodfellow et al. (2014)

1.1.2 Adversarial Attacks

Deep Neural Networks are a very powerful tool to recognize patterns in data, and, for example, perform image classification on a human-level. However, can we somehow "trick" the model and find failure modes? In the picture 1.2 we can see that the image of the panda with an addition of some noise will be recognized as gibbon, with 99,3% confidence. So, that noise addition was our adversarial attack. We have changed our image a little and now model cannot classify that right, when human would easily say that it's still a panda.

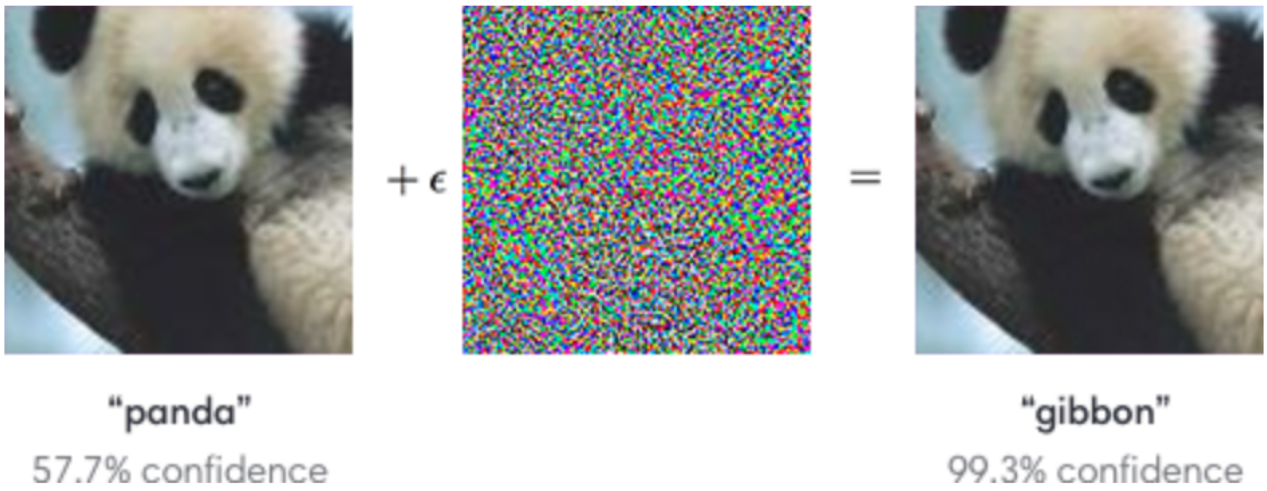


Figure 1.2: Panda Adversarial Attack example from openai. With small addition of some noise, the classifier can be fooled

1.2 Formulation of the problem

The purpose of this work is to explore different ways of building GANs and compare them with GANs that have been trained using Adversarial Attacks. The main idea is to stabilize learning with attacks on Discriminator, that will work like regularization. So we will create "attack" images and extra train D with them.

The problem of instability training in GANs is one of the main throughout the existence of architecture. Almost all articles try to solve this problem in one way or another. There are also many papers like "Towards Principled Methods for Training Generative Adversarial Networks" (Martin Arjovsky and Bottou (2017)), where authors explore this field. Other works are aimed at improving quality of

generated images. However, currently there is only one article about research in training GAN with Adversarial Attacks, called "Rob-GAN: Generator, Discriminator, and Adversarial Attacker" by Liu and Hsieh (2019), where the main goal was to investigate how Adversarial Attacks can affect convergence.

There are many options for Adversarial Attack, they are all divided into 2 types: Black Box and White Box. White Box Attacks assume that we have access to the model parameter and can, for example, calculate the gradients with respect to the input (similar as in GANs). Black Box Attacks on the other hand have the harder task of not having any knowledge about the network, and can only obtain predictions for an image, but no gradients or the like. In this work, we will focus on White Box Attacks as they are usually easier to implement and follow the intuition of Generative Adversarial Networks.

Nowadays, we have many different implementations of GANs, types of Adversarial Attacks and datasets. We will not be able to implement everything, however, identifying the best patterns is the goal of the work.

1.3 Main results

Adversarial Attacks were investigated on the following GANs:

- WGAN: Huge improvement of key metrics, stabilizing Generator loss.
- WGAN-GP: Similar results as in the classic version, destabilizing Discriminator loss.
- DCGAN: Similar results as in the classic version.
- SNGAN: Small improvement in key metrics, stabilizing Generator loss.

1.4 Structure of the work

1 Introduction to Adversarial Attacks, realization of few methods on multiple datasets.

2 Realization of some popular GANs, calculation and comparison of key metrics on CIFAR-10 dataset.

3 Development of GAN Adversarial training theory, and implementation of it with different GANs and hyperparameters.

2 Related work

2.1 Generative Adversarial Networks

In 2016 Facebook's AI research director Yann LeCun called adversarial training "the most interesting idea in the last 10 years in ML". First idea was described in the paper "Generative Adversarial Nets" by I. Goodfellow et al. (2014). However, nowadays the "vanilla" realization of GAN is DCGAN, proposed by Radford, Metz, and Chintala (2015), which led to more stable models. We can see the difference in the picture 2.1. Most GANs today are at least loosely based on the DCGAN architecture.

The Wasserstein GAN, or WGAN for short, was introduced by Martín Arjovsky, Chintala, and Bottou (2017), in their paper titled "Wasserstein GAN". Authors propose to slightly change the architecture: instead of using a Discriminator that predicts whether the given image is real or fake, the WGAN replaces D with a "Critic" model, which scores the realness or fakeness of an image. A new way of training is more stable and less sensitive to the choice of hyperparameters and model architecture. In the picture 2.2 we can see the difference between WGAN and GAN samples.

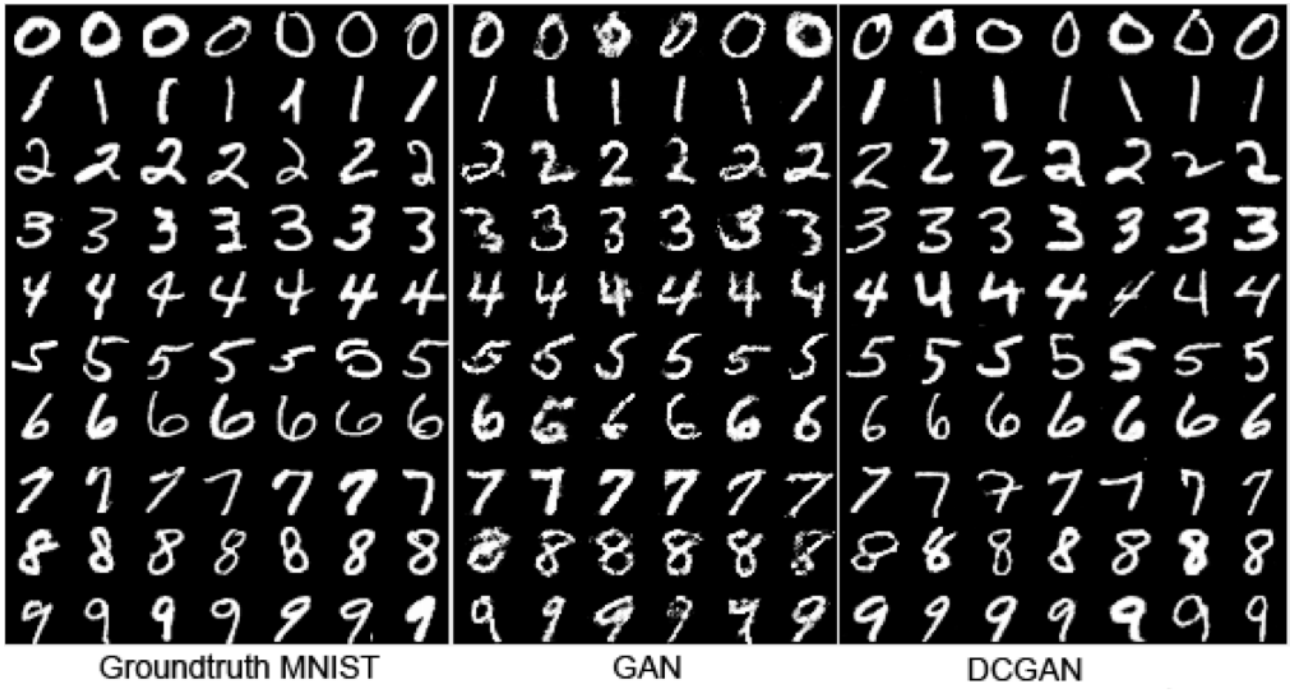


Figure 2.1: GAN comparison on MNIST from opengenus

Also there is an upgrade of WGAN, called WGAN-GP, which penalizes the norm of gradient of the Discriminator with respect to its input, instead of clipping. It is described in the paper "Improved Training of Wasserstein GANs" by Gulrajani et al. (2017)

Figure 2.2: Examples of generated images from WGAN paper (Martín Arjovsky, Chintala, and Bottou (2017)). Left: WGAN. Right: standard GAN

Another configuration of GAN is SNGAN, which uses Spectral Normalization after the last layer. Idea was described in the paper "Spectral Normalization for Generative Adversarial Networks" by Miyato et al. (2018). In the picture 2.3 we can see some samples from SNGAN.

Figure 2.3: Examples of generated images by SNGAN from SNGAN paper Miyato et al. (2018)

2.2 Adversarial Attacks

For Deep Learning, the first paper about Adversarial Attacks was published by I. J. Goodfellow, Shlens, and Szegedy (2015) under the title "Explaining and Harnessing Adversarial Examples". Article describes White Box Attack called Fast Gradient Sign Method (FGSM). We get our pictures following the formula:

$$\widetilde{x} = x + \epsilon \cdot sign(\nabla x J(\theta, x, y))$$

The term $J(\theta, x, y)$ represents the loss of the network for classifying input image x as label y; ϵ is the intensity of the noise, and $\widetilde{x}$ the final adversarial, example. The equation resembles SGD and is actually nothing else than that. We change the input image x in the direction of maximizing the loss $J(\theta, x, y)$. This is exactly the other way round as during training, where we try to minimize the loss. The sign function and ϵ can be seen as gradient clipping and learning rate specifically. We only allow our attack to change each pixel value by ϵ. Also the attack can be performed very fast, as it only requires a single forward and backward pass. We can see an example of an attack in the picture 2.4.

Another way to fool the model called Adversarial Patch method. The authors Brown et al. (2017) of the article "Adversarial Patch" say: "We present a method to create universal, robust, targeted adversarial image patches in the real world. The patches are universal because they can be used to attack any scene, robust

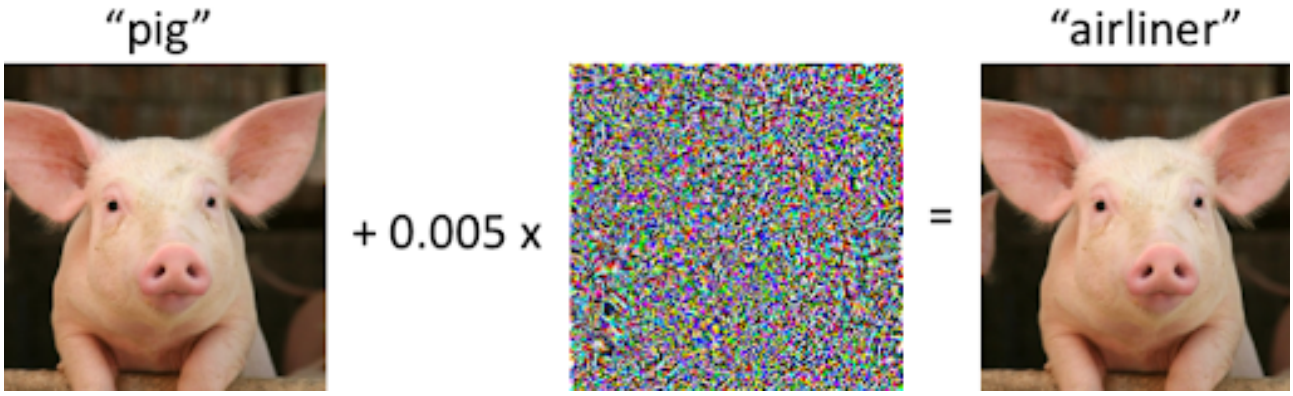
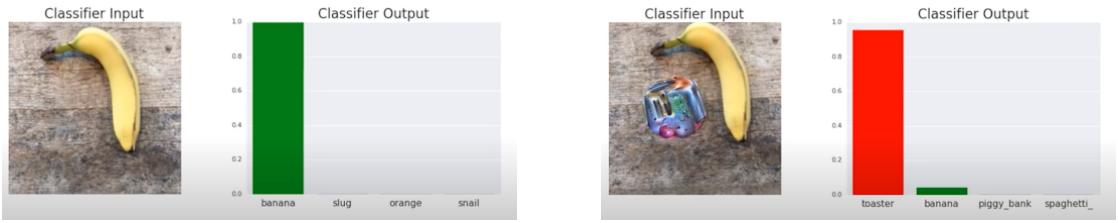


Figure 2.4: Example of misclassification due to FGSM attack from medium.

because they work under a wide variety of transformations, and targeted because they can cause a classifier to output any target class. These adversarial patches can be printed, added to any scene, photographed, and presented to image classifiers; even when the patches are small, they cause the classifiers to ignore the other items in the scene and report a chosen target class." The main idea is to add a small picture of the target class (here toaster) to the original image, the model does not pick it up at all. A specifically designed patch, however, which only roughly looks like a toaster, can change the network's prediction instantaneously. So let's try to change not all image, only part of it. This form of attack is an even bigger threat in real-world applications than FSGM. As we can see in the picture 2.5, classifier cannot recognize banana with the addition of a toaster.



(a) banana (b) toaster

Figure 2.5: Banana or toaster? Adversarial Patch Attack on banana picture, from Adversarial Patch paper (Brown et al. (2017)). With an addition of a toaster, classifier fails to recognize the banana

2.3 Quality

In GANs, quality cannot be measured using accuracy or with other common metrics. So, the paper "Improved Techniques for Training GANs" by Salimans et al. (2016) provides a solution to this: a new score called "Inception score" (IS). In short, our GAN generates about 50k images, which passes through the pretrained model "Inception", that gives the distribution of each image by class (picture 2.6). Further, with various calculations, IS is considered. All we need to know is that the more IS, the better and more diverse classes are represented by our GAN. However, there are also many problems with IS, it is far from perfect.

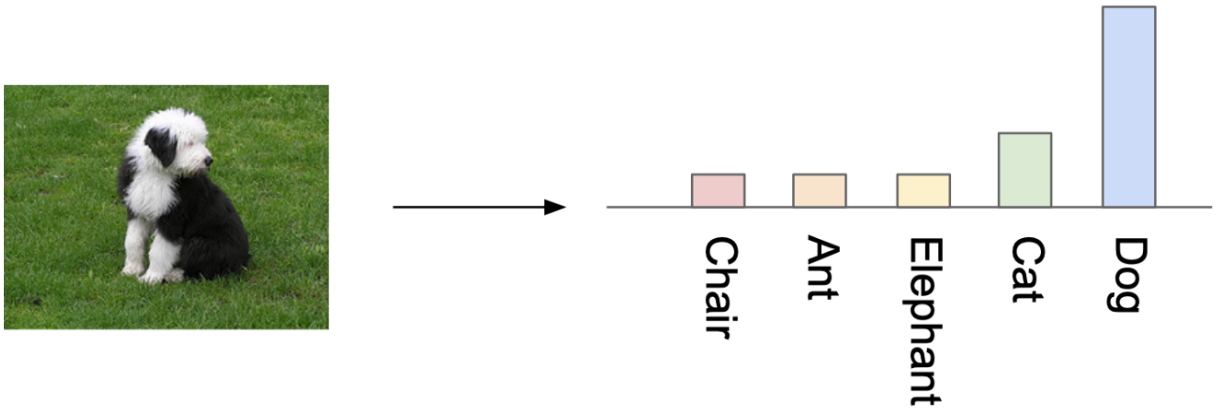


Figure 2.6: Class distribution from some classifier on a picture of a dog, from medium

Another way to calculate quality for GANs described in the paper "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium" by Heusel et al. (2017). FID is an improvement on IS: much more like human perception. FID counts the Frechet distance. The smaller it is, the more similar the resulting distribution is to the real one. Nevertheless, as we can see on picture 2.7, some problems are still with us. But at present, these two measures are the most popular and useful.

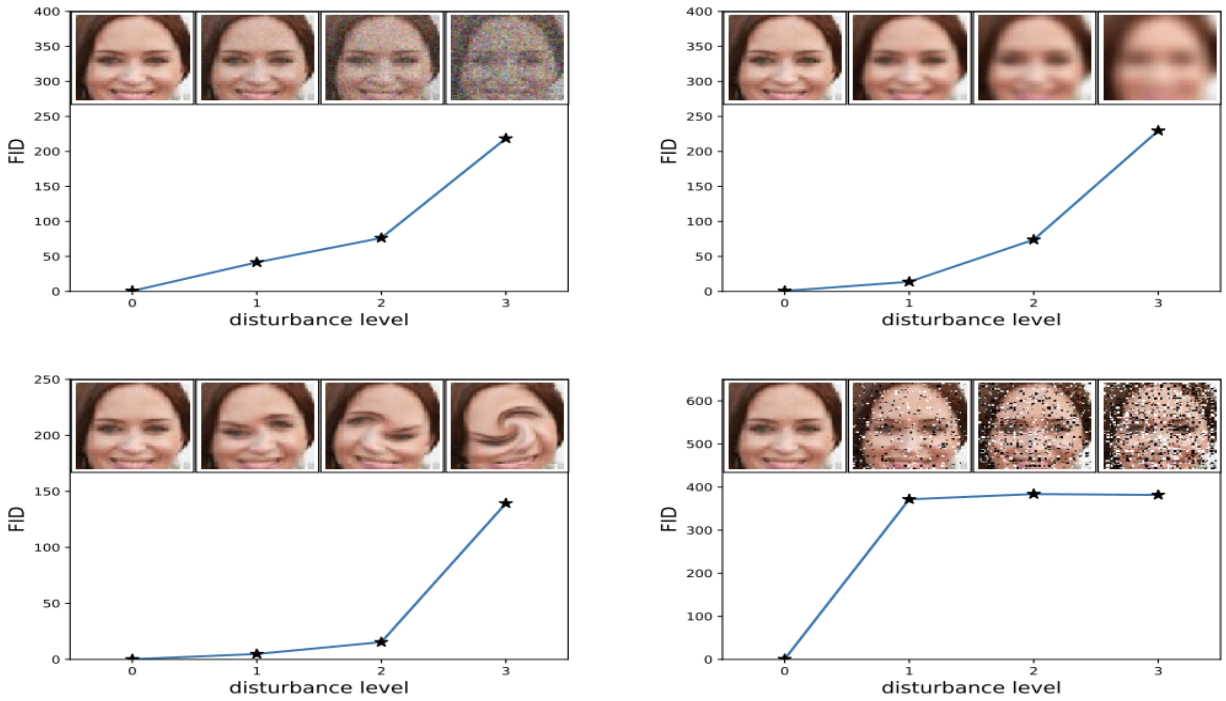


Figure 2.7: FID example from "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium" paper (Heusel et al. (2017)). The FID captures the disturbance level very well by monotonically increasing. But sometimes doesn't see the difference between a bad and a terrible image.

2.4 Attack on GAN

Currently there is only one article about research in training GAN with Adversarial Attacks, called "Rob-GAN: Generator, Discriminator, and Adversarial Attacker" by Liu and Hsieh (2019). Paper explores how PGD attacks (Madry et al. (2018)) affects convergence speed of GAN training and the robustness of Discriminator. Illustration of the training process we can see in the picture 2.8

First of all, authors use a powerful and time-consuming method of Adversarial Attack, called PGD-attack. Given an example x with ground truth label y, PGD computes adversarial perturbation δ by solving the following optimization with Projected Gradient Descent:

$$\delta := \underset{||\delta|| \leq ||\delta_{max}||}{\operatorname{argmax}} \; l(f(x+\delta; w), y),$$

where $f(\cdot; w)$ is the network parameterized by weights w, $l(\cdot, \cdot)$ is the loss function and $||\cdot||$ is l_∞-norm. Secondly, authors architecture based on AC-GAN (Odena,

Olah, and Shlens (2017)), this type of GAN didn't become popular over time. And what's most important, the paper doesn't research the quality of Rob-GAN at all, the purpose of the work was to improve convergence speed and robustness.

In our work, the goal is to create a Generator, which will train not much longer than the classic version and will improve key metrics (mostly FID). So we will use more powerful GANs and faster attack than PGD, because it is very time-consuming to solve an optimization problem on each step. Also adversarial images can alternate with dataset images when they are fed to Discriminator.

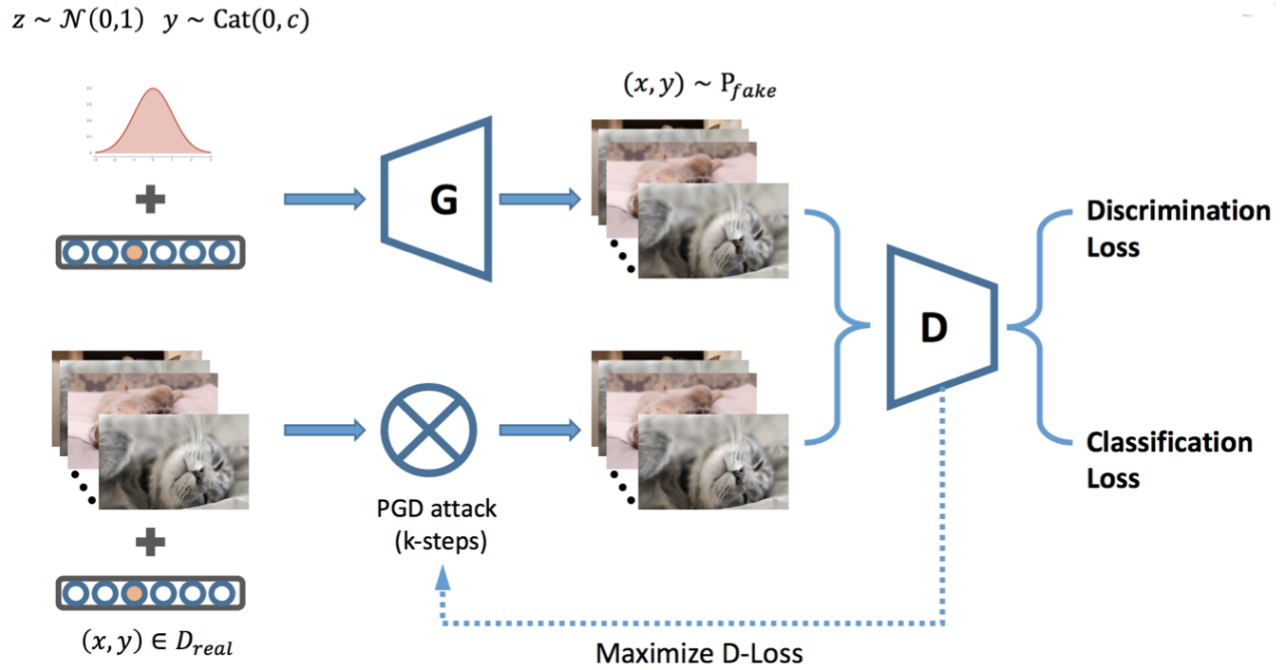


Figure 2.8: Illustration of the training process from Rob-GAN paper (Liu and Hsieh (2019)). This is similar to the standard GAN training, the only difference is that whenever feeding the real images to the D network, authors first invoke adversarial attack

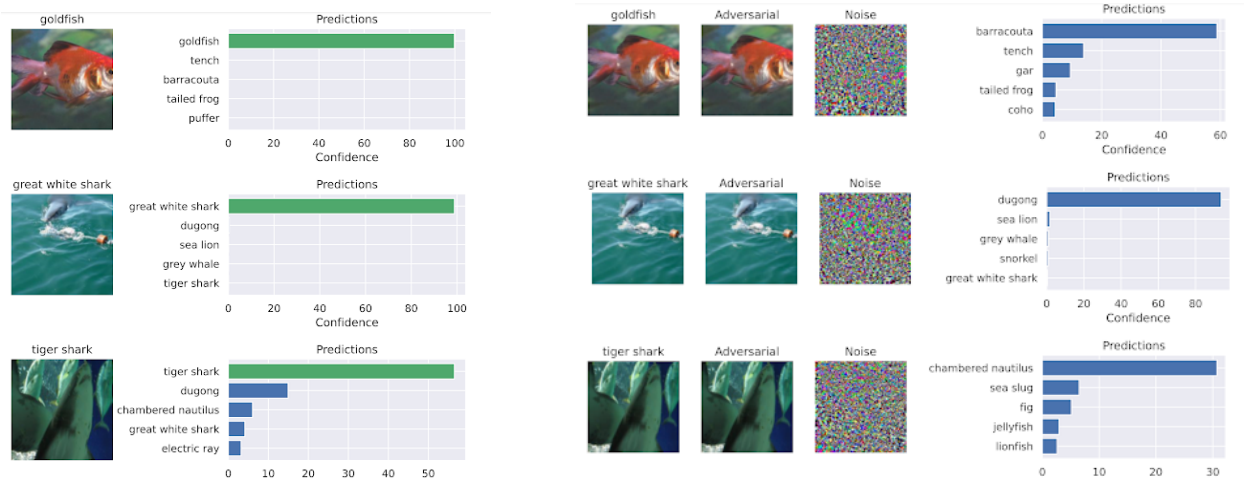
3 First steps

All code can be found in github.

3.1 Adversarial Attacks

First of all, Adversarial Attacks were carried out on the ImageNet dataset. As we can see on picture 3.1, only with one FSGM step, none of the right labels in

the top 5, when for human pictures are the same. Also model has big confidence on the adversarial image. And top 5 error now = 60%, when with dataset images top 5 error = 4,3%.



(a) dataset images (b) FGSM images

Figure 3.1: FGSM ImageNet. Example of attack on ImageNet dataset

After that, Adversarial Patches showed excellent results on the same dataset: as we can see in the picture 3.2, we fool our model in 100% of cases. Also patches were successfully transferred to the DenseNet: using the "pineapple patch" of size 64x64, the accuracy of the model drops to 65.41% in top-1, and to 82.81% in top-5.

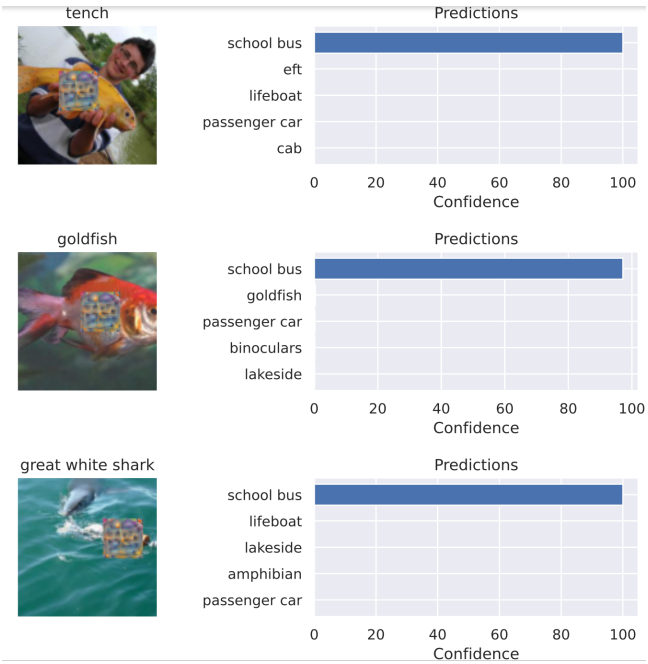


Figure 3.2: Adversarial patches example on ImageNet dataset. With the addition of pineapple we can fool model in 100% of cases

Also, FGSM method has been tested on MNIST and CIFAR-10 dataset. Model for MNIST classification has an accuracy about 0.987, so it was not so easy to deceive it in one step. Examples we can see in the picture 3.3. The plot of error versus epsilon for CIFAR-10 is presented on 3.4.

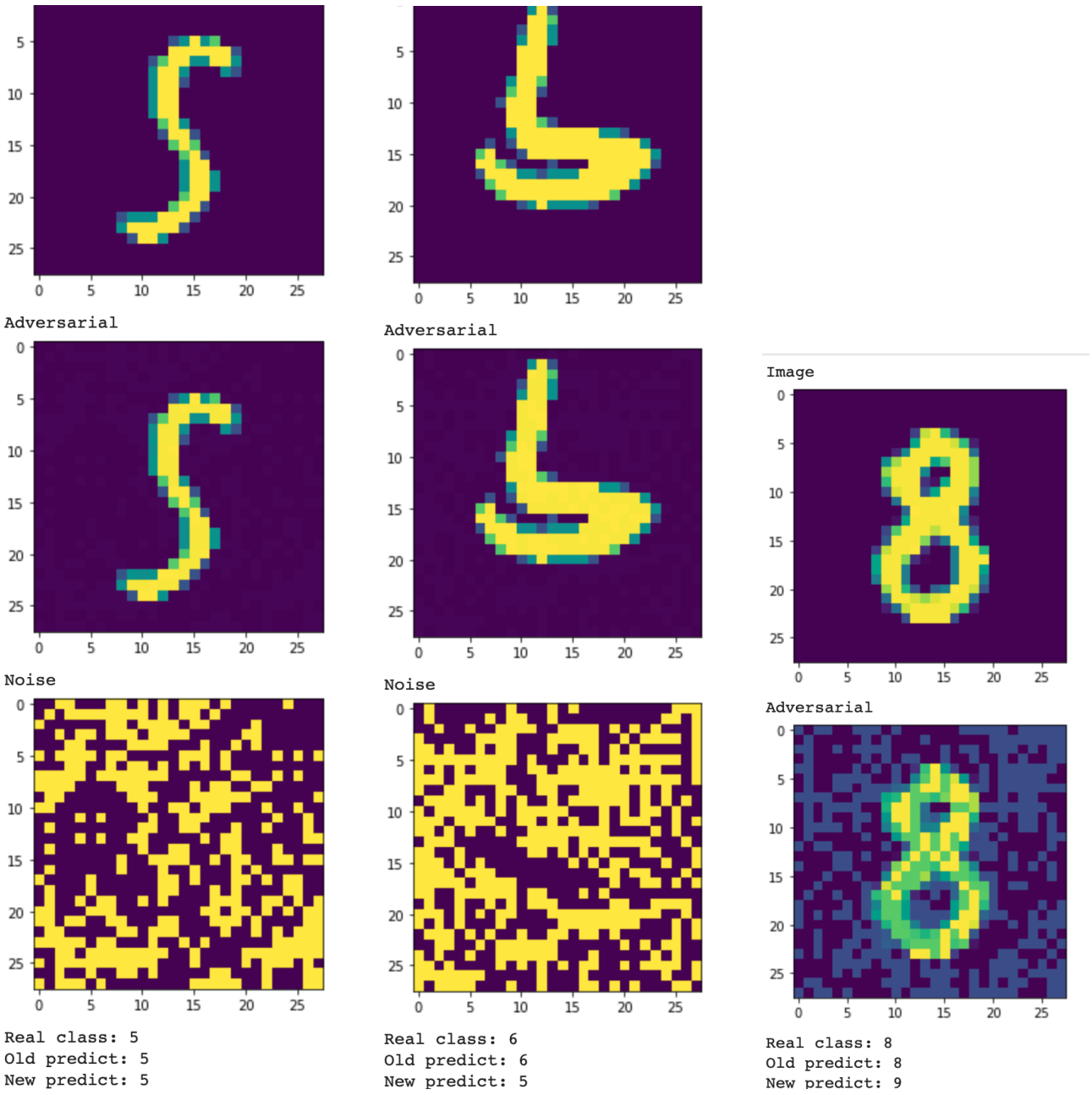


Figure 3.3: In the left picture, we failed to fool the model, but in the middle image FGSM attack was successful. Thus we have to increase epsilon to achieve stable attack, one of which is shown in the right figure.

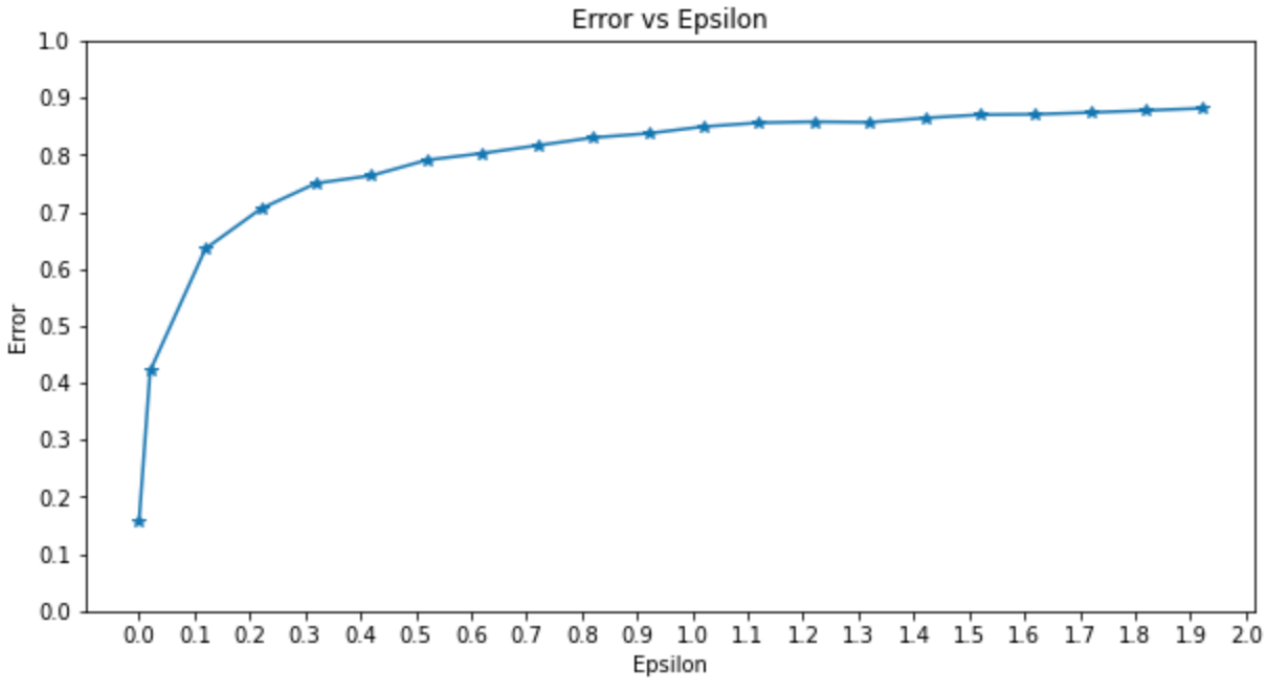


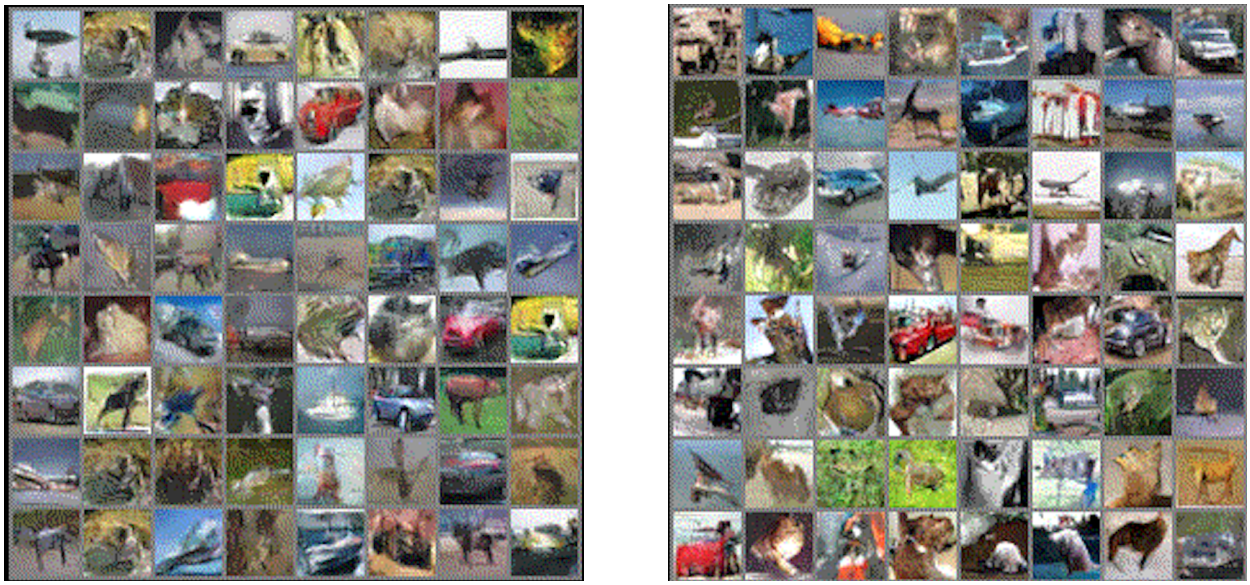
Figure 3.4: Error vs Epsilon with FGSM attack on CIFAR-10 dataset. Even with a small epsilon, we can fool model in most cases

3.2 GANs

It's hard to completely repeat the author's metrics of each GAN, but some of them have even been improved. As we can see in the table 3.1, our implementation of DCGAN and WGAN-GP(CNN) turned out to be better than the classic ones due to a more successful selection of hyperparameters on CIFAR-10. We can also look at the generated images by our DCGAN and the best images from the classic implementation of SNGAN(CNN) - picture 3.5.

Table 3.1: GANs quality

Model	Dataset	Inception Score	FID
our DCGAN	**CIFAR10**	**6.40(0.06)**	**41.42**
DCGAN	**CIFAR10**	**6.26(0.06)**	**41.92**
our WGAN-GP(CNN)	**CIFAR10**	**7.71(0.11)**	**18.67**
WGAN-GP(CNN)	**CIFAR10**	**7.66(0.10)**	**19.83**
our WGAN(CNN)	**CIFAR10**	**6.00(0.08)**	**48.38**
WGAN(CNN)	**CIFAR10**	**6.62(0.09)**	**40.03**
our SNGAN(CNN)	**CIFAR10**	**7.76(0.13)**	**18.38**
SNGAN(CNN)	**CIFAR10**	**7.84(0.12)**	**17.81**



(a) our DCGAN sample

(b) SNGAN sample

Figure 3.5: Example of generated pictures from our DCGAN realization vs classic SNGAN

4 Theory of GAN Adversarial Training

Training algorithm, presented in the picture 4.1, is very similar to GANs, the only difference is that sometimes G and D will learn on the adversarial images.

As mentioned before, in this work, we will focus on FGSM attacks. For this choice, there 2 reasons: first of all, we can easily apply FGSM to G and D, we only need to calculate gradient by loss, and nothing extra, like in Adversarial Patch, where we need to find a patch from somewhere. Secondly, it is very fast, unlike PGD, FGSM making only one step by gradient, which leads to worse results, but we are achieving high speed, GANs are still very slow in learning.

We want to explore the distribution of the real dataset, that's why we don't need adversarial attacks on every image. First of all, carry out an attack on G in the first epochs of training may have a bad influence on quality, because at the start of the training, G generates low quality images. So, let's say that we will start to attack images from 10% of epochs (kind of hyperparameter), with some chance C (another hyperparameter), and ϵ in FGSM (different hyperparameter for various datasets and image quality).

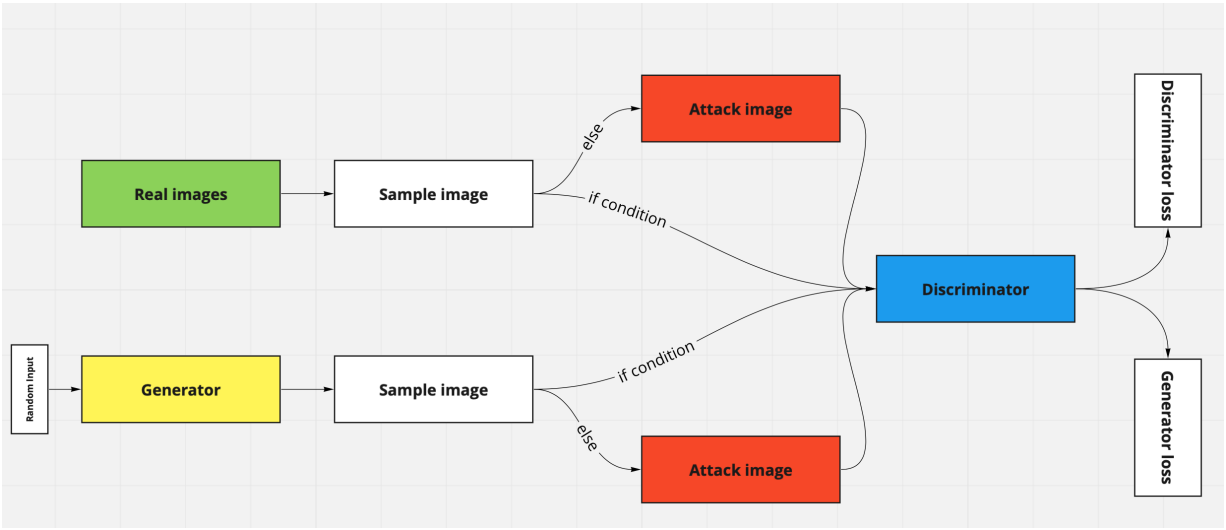
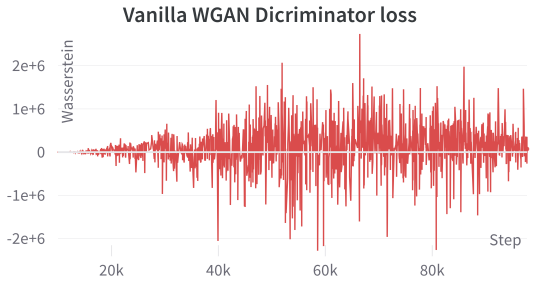
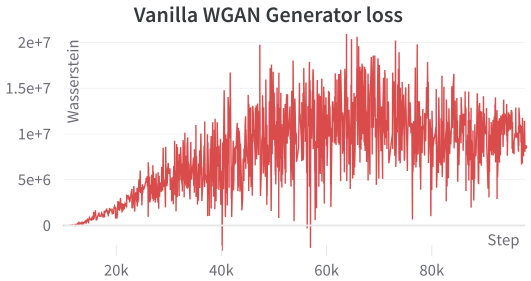


Figure 4.1: Illustration of the training process. This is similar to the standard GAN training, i.e. alternatively updating the generator G and discriminator D networks. The main difference is that, with some condition, we first invoke adversarial attack, so the Generator and Discriminator are trained with adversarial examples.

The goal of the work is to improve GAN metrics, but in that time, we want to monitor robustness and stability of the architecture. For these reasons, we can easily track Discriminator and Generator losses. If the graphs become more smooth, we will achieve what we want. In a bad case, we can also split the Discriminator loss into two two parts: one is responsible for the real images and another for the fake ones. Then we can figure out what went wrong: Generator overtook Discriminator or vice versa. For example, when we do forward pass in Wasserstein loss for D, we compute loss for real as minus mean of the real, and loss for fake as mean of the fake, then just summarize these values. We can see examples from WGAN training on 4.2 and 4.3

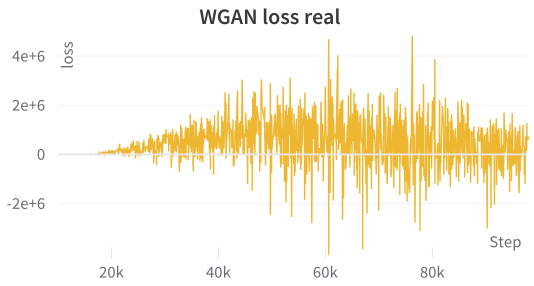


(a) WGAN D loss

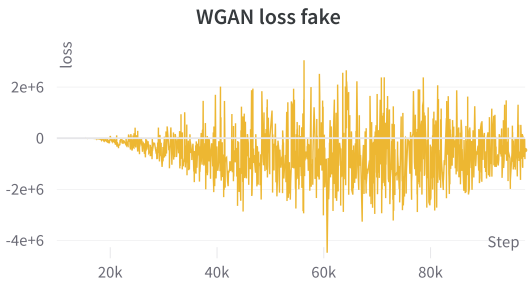


(b) WGAN G loss

Figure 4.2: Loss functions of Discriminator and Generator in GAN training can be both positive and negative, as well as large values



(a) WGAN loss of real part

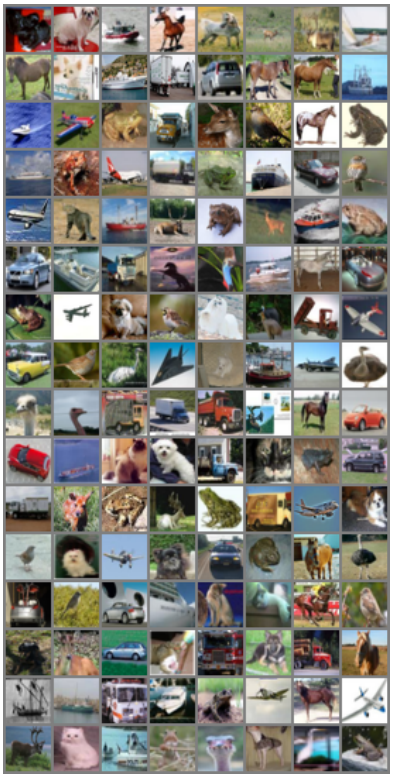


(b) WGAN loss of fake part

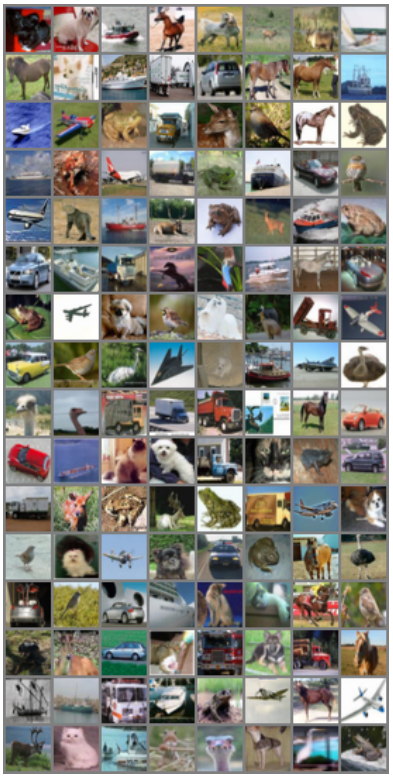
Figure 4.3: Real and fake parts of loss function of Discriminator in GAN training have the same problems

5 Experiments

All experiments were carried out on the CIFAR-10 dataset, 1xV100 and 8xCPU. Default hyperparameter ϵ was 0.02. With this value, Discriminator will make mistakes more often. But most importantly, attack images will not change much, and human will not be able to recognize the difference. We have to preserve image quality for better IS and FID. Example we can see in the picture 5.1.



(a) Real images from CIFAR-10



(b) Attacked by FGSM images from CIFAR-10

Figure 5.1: Human can't see the difference between these pictures, but many pixels in the right image were changed by the FGSM attack

Choosing hyperparameter C (chance that FGSM will be applied) is quite a difficult task. With C close to 0, the effect of adversarial training will be weak and the results will not differ significantly. With a chance close to 1, Generator will capture a completely different distribution, which will have a bad influence on the key metrics. So, we have to find the golden mean. The following values of C were considered in all experiments: 0.2, 0.4, 0.6, 0.8, then most successful values or cases where training exploded, were investigated more.

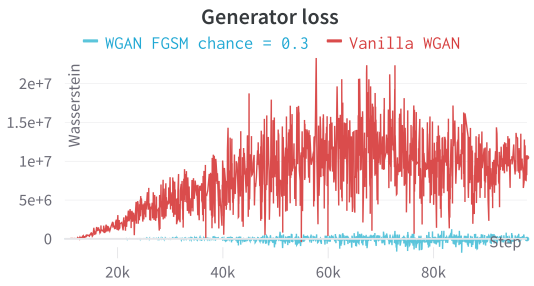
5.1 WGAN

WGAN-FGSM has the best performance among all considered GANs. Model with best hyperparameters ($\epsilon = 0.01, C = 0.3$) improved IS by 10% and FID by almost 30%, compared to the classic version! Over 25 full experiments were carried out, the best results are in the table 5.1.

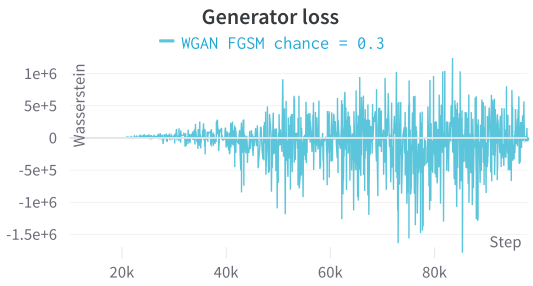
Table 5.1: WGAN-FGSM quality on CIFAR-10

Model	FGSM chance	ϵ	Inception Score	FID	Time (min)
Baseline WGAN	–	–	6.00(0.09)	48.38	**502**
WGAN-FGSM	0.2	0.02	6.58(0.09)	35.21	538
WGAN-FGSM	0.3	0.02	6.53(0.06)	**33.60**	537
WGAN-FGSM	0.4	0.02	6.73(0.10)	35.56	595
WGAN-FGSM	0.3	0.01	**6.77(0.07)**	33.78	537

Loss functions become much more stable, which makes training run more smoothly. Example of plot behavior is presented in the picture 5.2, generated images in the picture 5.3



(a) Generator losses for classic WGAN and WGAN-FGSM



(b) Generator loss for WGAN-FGSM

Figure 5.2: GAN Adversarial Training can stabilize training by influencing on the losses

(a) Sample generated by WGAN

(b) Sample generated by WGAN-FGSM

Figure 5.3: Generated samples from WGAN and WGAN-FGSM. The same noise was in the input

5.2 WGAN-GP

WGAN-GP penalizes the norm of gradient of the Discriminator with respect to its input. There are several options for FGSM-attack:

1. Carry out an attack, then calculate the gradient penalty.
2. Calculate gradient penalty, then carry out an attack with the resulting loss.
3. Sometimes calculate gradient penalty, other times carry out FGSM-attack

First option looks like the best, but a little out of our approach. In the second one we can't calculate the gradient of the resulting loss in the usual ways. So the third option is our choice. However, results, which we can see at the table 5.2 are

disheartening: metrics and time got worse, especially for large values. Also losses have the same behavior.

Table 5.2: WGAN-GP-FGSM quality on CIFAR-10

Model	FGSM chance	Inception Score	FID	Time (min)
Baseline WGAN-GP	–	**7.71(0.11)**	**18.67**	**613**
WGAN-FGSM	0.2	7.64(0.09)	19.97	645
WGAN-FGSM	0.4	3.35(0.03)	106.6	673
WGAN-FGSM	0.6	3.51(0.03)	112.57	693
WGAN-FGSM	0.8	4.19(0.07)	97.37	721

5.3 DCGAN

The performance of DCGAN almost doesn't change with FGSM-attacks: losses have not changed their behavior, metrics are similar, what we can see at the table. This means attacks don't necessarily improve weak GANs.

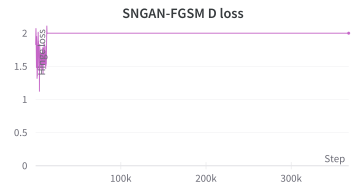
Table 5.3: DCGAN-FGSM quality on CIFAR-10

Model	FGSM chance	Inception Score	FID	Time (min)
Baseline DCGAN	–	6.40(0.06)	41.42	**590**
DCGAN-FGSM	0.2	**6.52(0.07)**	40.23	596
DCGAN-FGSM	0.4	6.15(0.09)	59.78	604
DCGAN-FGSM	0.6	6.34(0.04)	**39.10**	627
DCGAN-FGSM	0.8	5.97(0.05)	53.50	642

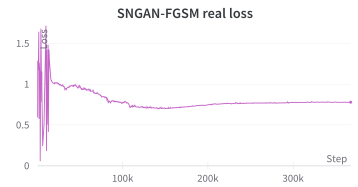
5.4 SNGAN

SNGAN is the most powerful among all the proposed in work GANs. However, significant results were not achieved, and many problems were encountered with the model at $\epsilon = 0.6, 0.8$. The left plot in figure 5.4 shows that Discriminator loss becomes constant 2. This can happen in two cases: G generates too high-quality and close to data distribution pictures, that D cannot distinguish them from the real ones. Or vice versa, Discriminator has become to smart for Generator and passes to little information for G learning. But 2 right plots in figure 5.4 shows us that real and fake parts of loss changes during training. This means that G

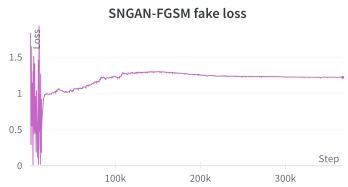
is trying to generate something new, but Discriminator is far ahead of him. This could happen because D received to much information about data distribution due to spectral normalization and frequent FGSM-attacks. If we have to fix it, but don't want to change the FGSM chance, we can simply decrease the starting epoch for attacks to zero, then Discriminator and Generator will learn from a slightly different distribution and there will be no such problems. Results at the figure 5.5.



(a) Loss of Discriminator after few steps from start FGSM-attacks becomes constant 2

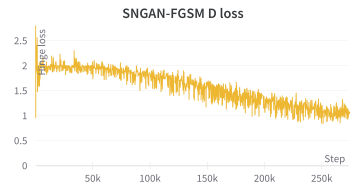


(b) Real part of loss is not a constant, but changes slowly

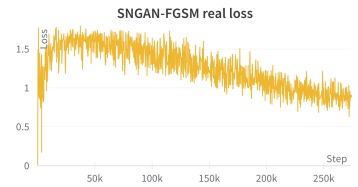


(c) Fake part of loss also is not a constant and changes slowly

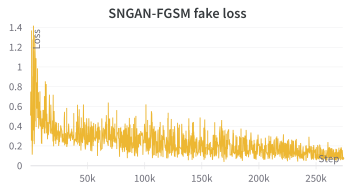
Figure 5.4: SNGAN-FGSM problem with high FGSM chance ($C = 0.8$)



(a) Loss of Discriminator



(b) Real part of loss



(c) Fake part of loss

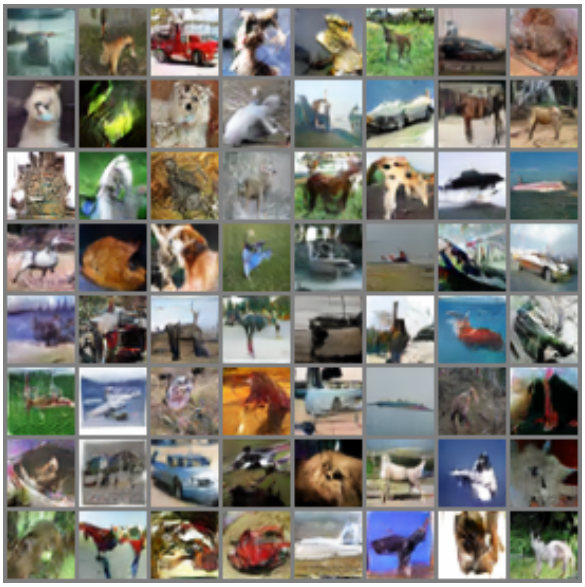
Figure 5.5: Losses are normal if we decrease the start epoch of FGSM attacks for models with high FGSM chance ($C = 0.8$)

Over 50 experiments were carried out, the best results are in the table 5.4. Images from SNGAN and best SNGAN-FGSM model in the picture 5.6

Table 5.4: SNGAN-FGSM quality on CIFAR-10

Model	FGSM chance	start FGSM	Inception Score	FID	Time (min)
Baseline SNGAN	–	–	**7.84(0.12)**	**17.81**	**503**
SNGAN-FGSM	0.2	10%	7.54(0.13)	19.20	750
SNGAN-FGSM	0.4	10%	7.36(0.03)	22.84	793
SNGAN-FGSM	0.6	5%	6.86(0.05)	28.34	580
SNGAN-FGSM	0.8	0%	6.64(0.08)	33.40	614

(a) Sample generated by SNGAN



(b) Sample generated by SNGAN-FGSM

Figure 5.6: Generated samples from SNGAN and SNGAN-FGSM. The same noise was in the input

6 Conclusion

During the coursework, almost 100 experiments were conducted with different implementations and hyperparameters, spent more than 30 days of computing resources, 4 different GAN architectures were explored with FGSM attacks. Some of them have improved quality and stabilized learning, WGAN metrics were increased significantly, which indicates the potential of Training Generative Adversarial Networks with Adversarial Attacks.

The first next steps would be to investigate more deeply the reasons for changes in training with Adversarial Attacks, perhaps a more detailed comparison with the article about Rob-GAN (Liu and Hsieh (2019)). After that, applying FGSM-attacks on most powerful modern GANs like StyleGAN (Karras, Laine, and Aila (2019)) and BigGAN (Brock, Donahue, and Simonyan (2019)). And than, with getting good results, we could try other popular datasets and Adversarial Attacks methods. With more than 1 step methods we can achieve completely different results.

7 Acknowledgments

This research was supported in part through computational resources of HPC facilities at HSE (University Kostenetskiy, Chulkevich, and Kozyrev (2021))

References

1. Martin Arjovsky and Léon Bottou. "Towards Principled Methods for Training Generative Adversarial Networks". In: *stat* 1050 (Jan. 2017).
2. Martín Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein GAN". In: *NIPS* abs/1701.07875 (2017).
3. Andrew Brock, Jeff Donahue, and Karen Simonyan. "Large Scale GAN Training for High Fidelity Natural Image Synthesis". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=B1xsqj09Fm.
4. Tom B. Brown et al. "Adversarial Patch". In: *NIPS* abs/1712.09665 (2017). arXiv: 1712.09665. URL: http://arxiv.org/abs/1712.09665.
5. Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems* 27 (2014).
6. Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6572.
7. Ishaan Gulrajani et al. "Improved Training of Wasserstein GANs". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 5769–5779. ISBN: 9781510860964.
8. Martin Heusel et al. "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf.

9. Tero Karras, Samuli Laine, and Timo Aila. "A Style-Based Generator Architecture for Generative Adversarial Networks". In: June 2019, pp. 4396–4405. DOI: 10.1109/CVPR.2019.00453.

10. P. S. Kostenetskiy, R. A. Chulkevich, and V. I. Kozyrev. "HPC Resources of the Higher School of Economics". In: *Journal of Physics: Conference Series* 1740.1 (Jan. 2021), p. 012050. DOI: 10.1088/1742-6596/1740/1/012050. URL: https://doi.org/10.1088/1742-6596/1740/1/012050.

11. Xuanqing Liu and Cho-Jui Hsieh. "Rob-GAN: Generator, Discriminator and Adversarial Attacker". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019.

12. Aleksander Madry et al. "Towards Deep Learning Models Resistant to Adversarial Attacks". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=rJzIBfZAb.

13. Takeru Miyato et al. "Spectral Normalization for Generative Adversarial Networks". In: *ICLR* abs/1802.05957 (2018). arXiv: 1802.05957. URL: http://arxiv.org/abs/1802.05957.

14. Augustus Odena, Christopher Olah, and Jonathon Shlens. "Conditional Image Synthesis with Auxiliary Classifier GANs". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 2642–2651. URL: https://proceedings.mlr.press/v70/odena17a.html.

15. Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015).

16. Tim Salimans et al. "Improved Techniques for Training GANs". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran

Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper/2016/file/8a3363abe792db2d8761d6403605aeb7-Paper.pdf.