BACHELOR’S THESIS

Warsaw University of Technology
Faculty of Electronics and Information Technology
Department of Computer Science

Genarating new spiders using Generative Adversarial Network (GAN)

Valery Burau

Under The Supervision Of

Prof. dr hab. Andrzej Pacut

1. INTRODUCTION

Genarative Adversarial Network was invented by Ian Goodfellow and his colleagues in 2014.

Main idea behind GAN:

We have two most important parts while creating this type of network:

- **Generator**: replicates real data to produce fake data.
- **Discriminator**: distinguishes real data from fake data.

In this way generator and discriminator are playing mini-max game.

Characteristics:[1]

- class of machine learning systems.
- given a training set, this technique learns to generate new data with the same statistics as the training set.
- the *generative network* generates candidates while the *discriminative network* evaluates them.
- the generator trains based on whether it succeeds in fooling the discriminator.
- the generator is typically a *deconvolutional* neural network, and the discriminator is a convolutional neural network.

Usage:

- creating fake and realistic photos (e.g. *portraits, landscapes and album covers)* and videos.
- increasing amount of data.
- precision of some images(e.g. *improve astronomical images and simulation gravitational lensing for dark matter research*).
- reconstruction 3D models of objects from images and model patterns of motion in video.
- aging face photographs to show how an individual's appearance might change with age.
- visualization of the effect that climate change will have on specific houses.

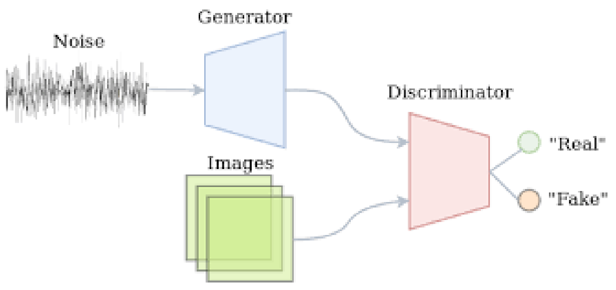
- model called Speech2Face can reconstruct an image of a person's face after listening to their voice.
- generating new ways to solve various problems (e.g. in construction) that people are even not able to foresee.
- generating fake voices.

Steps to train a GAN: [2]

1. Define the Problem (synthesize images from a caption, audio synthesis from sentences and etc.).
2. Define GAN Architecture (depends on the complexity of the problem) (multi-layer perceptron, neural network or a simpler model).
3. Train Discriminator to distinguish “real” Vs “fake” data (data labelled like real data, generated data labelled like fake data).
4. Train Generator synthesize data (we need to modify parameters of the generative model to maximize a loss of the discriminator).
5. Repeat steps 3 and 4 N times (N epochs).
6. Final result: the discriminator will not be able to distinguish the real and the fake samples.
7. Once training is complete, synthesize data from Generator.

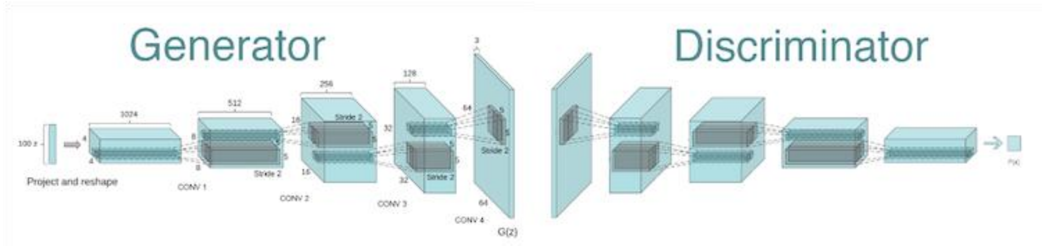
Types of GANs:

1. Original Vanilla GAN
    - A presentation of original GAN.



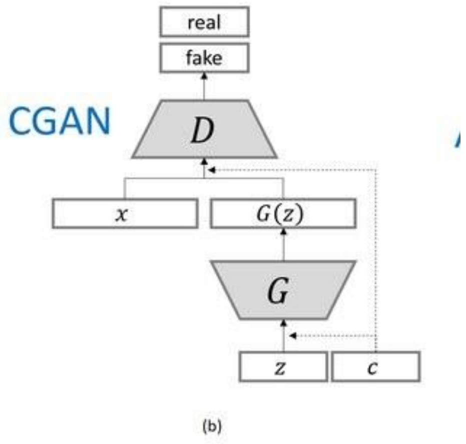
Picture: [3]

2. Deep Convolutional GAN (DCGAN)
    - CNNs used in unsupervised learning.
    - Generators are Deconvolutional Neural Networks.
    - Discriminators are CNNs.



Picture: [4] Source.Radford et aL, 2015

3. Conditional GAN(CGAN)
   - Dictate the type of data generated through a condition.



Picture: [6]

2. LITERATURE

- Goodfellow, Ian; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron; Bengio, Yoshua (2014). *Generative Adversarial Networks* (PDF). Proceedings of the International Conference on Neural Information Processing Systems (NIPS 2014). pp. 2672–2680
- *Salimans, Tim; Goodfellow, Ian; Zaremba, Wojciech; Cheung, Vicki; Radford, Alec; Chen, Xi (2016). "Improved Techniques for Training GANs" .arXiv:1606.03498 [cs.LG].*
- DCGAN. WRITTEN BY Li Yin.Mar 8, 2017.
- GANS — PART2: DCGANs (deep convolution GANS) for generating images. Manish Chablani. Jun 28, 2017
- GAN — DCGAN (Deep convolutional generative adversarial networks). Jonathan Hui. Jun 18, 2018

3. METHODOLOGY AND DATA

Main task:

Generate new species of spiders using GAN.

Libraries:

- PyTorch
  - Powerful library that provides the user with various helpful functions(such as a training on GPU, functions for preprocessing data, different learning algorithms, etc.).
- Matplotlib
  - Using for drawing pictures.
- Numpy
  - Using for mathematical operations.

Dataset:

- 4820 images in RGB system.
- Different size of images.
- All spiders are mostly situated in the very center of the image.

Examples:

Preprocessing data:

Before putting images to the network as input it's important to make all images of the same size and get rid of unnecessary information.

To achive this goal I've used(for each image):

1)Center Cropping

- Choose which size is smaller: height or width;
- Reduce the chosen size(equal 0.8*smallest_size_value)
- New size is using for both: height and weight.
- Make center crop of the image using received values

2)Resizing

- Make the size of image to be equal 64x64.

Examples:

Type of Model:

I’ve implemented DCGAN model. It’s considered as one of the most effective models for this kind of problem.

Working with images we mostly use CNN(Convolutional Neural Network).

- Ridding of pooling layers.
    - Usage of padding instead of pooling helps us to save more information about the image.
- Batch Normalisation for the generator and the discriminator.

Discriminator:

As input discriminator takes a fraud/real image of spider of size 64x64x3 (3 due to RGB image).

Then using filters of size 4x4, stride equal 2 and padding equal 1 i get twice more feature maps and twice less the size of each feature map with each new layer(except the first and last one).

We’re using filters to extract different features(characteristics) about the image as much as possible.

I get 4x4x512 image(where 4x4-image_size; 512-number of feature maps) from 64x64x3 image.

In the end i transform all feature maps to one layer using flattening and get the result if the image is real or fake(was generated by generator).

- Classification model.
- LeakyReLU activation function.
- Convolutional layers.

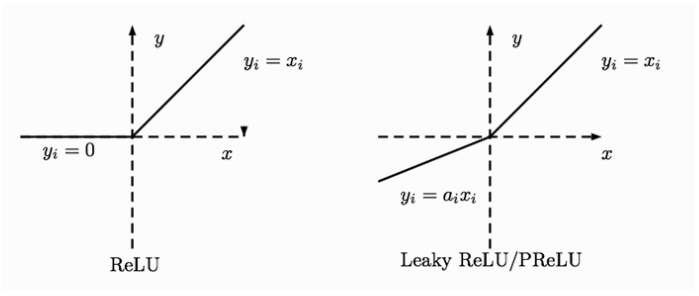
Generator:

As input generator takes random noise and transform it to a flattening layer. After this i do vice versa steps comparing with discriminator(starting from 4x4x512).

In the end i get a fake image of spider of size 64x64x3.

- Deconvolutional layers.

- ReLU activation function(last layers – tanh function).



Picture: [7]

Losses:

- Two functions of losses(real_loss, fake_loss).

For discriminator we should sum

d_loss = fake_loss(D_fake) + real_loss(D_real)
D_fake – put to discriminator fake images
fake_loss(D_fake) - tells what a loss when it shows that images are real
D_real – put to discriminator real images
real_loss(D_real) - tells what a loss when it shows that images are fake

g_loss = real_loss(D_fake)
real_loss(D_fake) - tells what a loss when it shows that images are fake

Type of loss:

Binary cross entropy with logits loss. Reasons:

- a binary classification model(two categories: real or fraud).
- better performance than *Misclassification rate* and *MSE (Mean Squared Error)*.
- *misclassification rate* fails to state how wrong / how correct predictions are (it computes the number of misclassified predictions but it does not take into account how off these predictions were from the real ones).
- eliminates *vanishing gradient* during training in neural network architecture (the change in the weights does not become zero). Learning is not stalled.
- it applies sigmoid activation instead of us, and we should add it manually if we use nn.BCELoss.

Training process:

We use GPU for this purpose. It takes several hours to train a usual model. To implement a stronger network it's needed more powerful video card and more time for learning process.

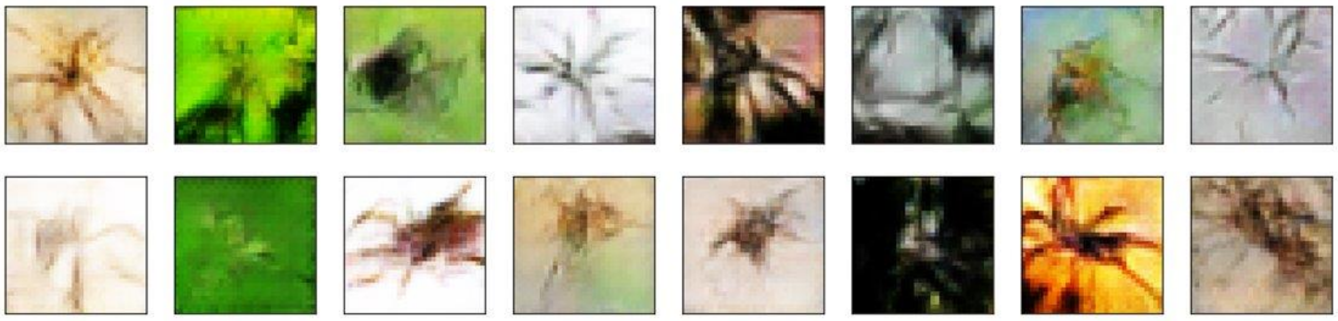
During learning process we're changing the weights that represent the strength of the connection between neurons and decide how much influence the inputwill have on the output.

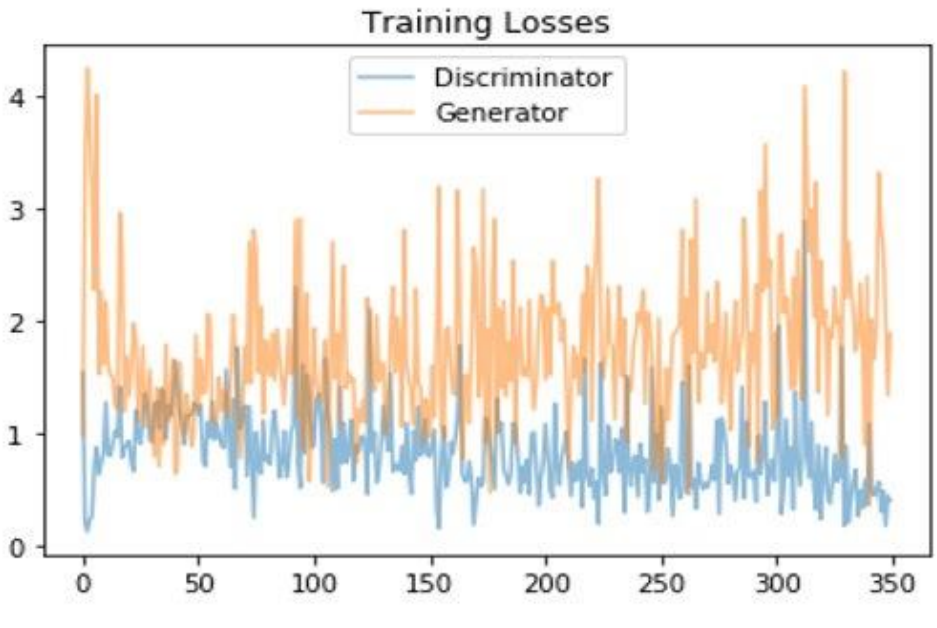
Code:

github

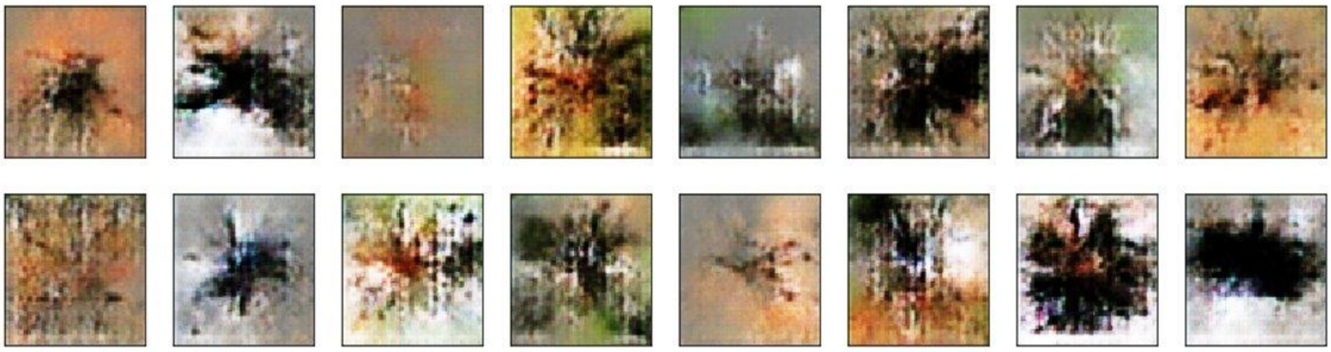
4. **RESULTS**

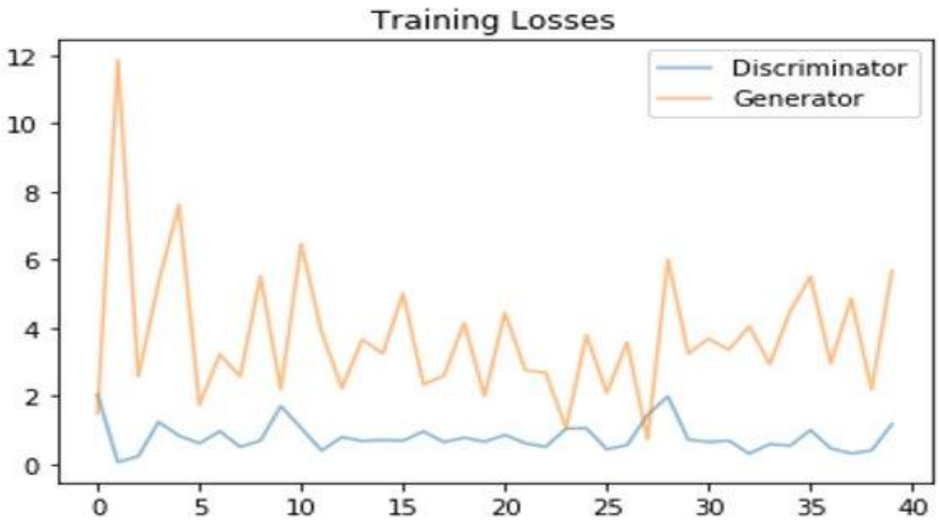
- batch size=32; epochs=40; image size=64x64



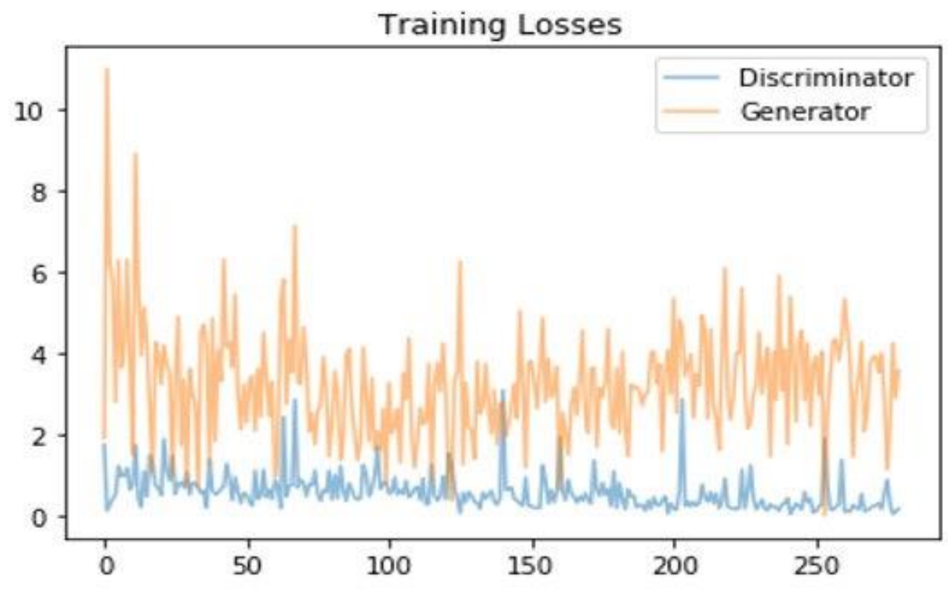


- batch size=32; epochs=40; image size=64x64





- batch size=32; epochs=40; image size=64x64



5. **BIBLIOGRAPHY**

1. Wikipedia. "Generative adversarial network".
2. CodeEmporium. "Generative Adversarial Networks – FUTURISTIC & FUN AI !".
3. Picture. Original Vanilla GAN. "Semi-supervised learning with GANs".
4. Picture. DCGAN. "GAN Deep Learning Architectures - review".
5. Picture. Generating faces. "18 | Impressive Applications of Generative Adversarial Networks (GANs)".
6. Picture. CGAN. "Fig 1- uploaded by Gerasioms (Jerry) Spanakis".
7. Picture. ReLU and Leaky ReLU. "ReInventing Neural Networks"