# BACHELOR’S THESIS

Warsaw University of Technology

[Faculty of Electronics and Information Technology](http://www.elka.pw.edu.pl/eng) Department of Computer Science

**Generation of new spiders using Generative Adversarial Network (GAN)**

Valery Burau

# Under The Supervision Of

**Prof. dr hab. Andrzej Pacut**

# **1. INTRODUCTION TO GAN**

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

## Main idea behind GAN:

There are 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]](https://en.wikipedia.org/wiki/Generative_adversarial_network#cite_note-GANnips-1)

* class of machine learning systems.
* given a training set, the 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](https://en.wikipedia.org/wiki/Convolutional_neural_network).

## Usage:

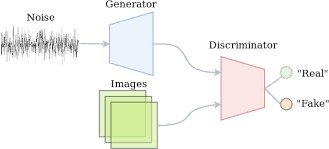
* creating fake and realistic photos (e.g. [*portraits, landscapes and album covers*](https://www.artbreeder.com/)*)* and videos.
* increasing amount of data.
* precision of some images (e.g. [*improve*](https://en.wikipedia.org/wiki/Image_restoration)*ment of* [*astronomical images*](https://en.wikipedia.org/wiki/Astrophotography) *and simulation gravitational lensing for dark matter research*).
* [reconstruction 3D models of objects from images](https://en.wikipedia.org/wiki/3D_reconstruction_from_multiple_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]](https://www.youtube.com/watch?v=O8LAi6ksC80&amp;list=PLTl9hO2Oobd-1jxZ01__NjibEY6h15Kha)

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 (parameters of the generative model should be modified 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.

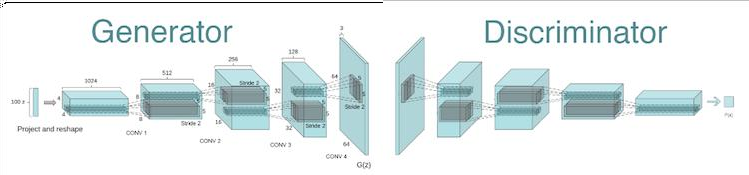
## Types of GANs:

1. Original Vanilla GAN
   * A representation of original GAN.



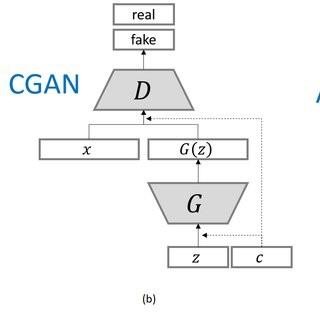
# *Picture 1: Original Vanilla GAN* [*(Jos van de Wolfshaar (2018)**Semi-supervised learning with GANs)[3]*](https://medium.com/@jos.vandewolfshaar/semi-supervised-learning-with-gans-23255865d0a4)

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



*Picture 2: DCGAN* [*(Roman Trusov GAN DEEP LEARNING ARCHITECTURES – REVIEW)[[4]*](https://sigmoidal.io/beginners-review-of-gan-architectures/)

1. Conditional GAN(CGAN)
   * Dictate the type of data generated through a condition.



*Picture 3: Conditional GAN (CGAN)* [*(Gerasimos Spanakis* (2018)*. “Fig 1- uploaded by Gerasioms (Jerry) Spanakis”)[5]*](https://www.researchgate.net/figure/GAN-conditional-GAN-CGAN-and-auxiliary-classifier-GAN-ACGAN-architectures-where-x_fig1_328494719)

# **LITERATURE**

* Goodfellow, Ian; Pouget-Abadie, Jean; Mirza, Mehdi; Xu, Bing; Warde-Farley, David; Ozair, Sherjil; Courville, Aaron; Bengio, Yoshua (2014). [*Generative Adversarial Networks*](https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf) (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:*](https://en.wikipedia.org/wiki/ArXiv)[*1606.03498*](https://arxiv.org/abs/1606.03498) [*[cs.LG*](https://arxiv.org/archive/cs.LG)*].*

* + DCGAN. WRITTEN BY Li Yin[.Mar 8, 2017](https://medium.com/%40liyin2015/dcgan-79af14a1c247?source=post_page-----79af14a1c247----------------------).
  + GANS — PART2: DCGANs (deep convolution GANS) for generating images. Manish Chablani. [Jun 28, 2017](https://towardsdatascience.com/gans-part2-dcgans-deep-convolution-gans-for-generating-images-c5d3c7c3510e?source=post_page-----c5d3c7c3510e----------------------)
  + GAN — DCGAN (Deep convolutional generative adversarial networks). Jonathan Hui. [Jun 18, 2018](https://medium.com/%40jonathan_hui/gan-dcgan-deep-convolutional-generative-adversarial-networks-df855c438f?source=post_page-----df855c438f----------------------)

# **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.

*Picture 4: Dataset examples*

## Preprocessing data:

It is important to make all images the same size and get rid of unnecessary information before putting them to the network as input.

The following steps were taken to achieve this (for each image):

1. Center Cropping:
   * Choose which size is smaller: height or width;
   * Reduce the chosen size (it equals 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.



*Picture 5: Dataset preprocessed examples*

## Type of Model:

DCGAN model has been implemented. It’s considered as one of the most effective models for this kind of problem (image preprocessing).

CNN (Convolutional Neural Network) are mostly used in working with images.

* 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:

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

Using filters of size 4x4, stride equals to 2 and padding equals to 1 twice more feature maps are got and the size of each feature map is twice less comparing to the previous layer (except the first and the last layers).

Filters are used to extract different features (characteristics) about the

image (e.g. quantity of some similar objects, landscape, position of the object and etc.)

Final result of convolutional layers is 4x4x512 image (where 4x4 - the image size; 512 - number of feature maps).

In the end all feature maps are converted into a single layer using flattening and the final result shows that the image is real or fake (created by generator).

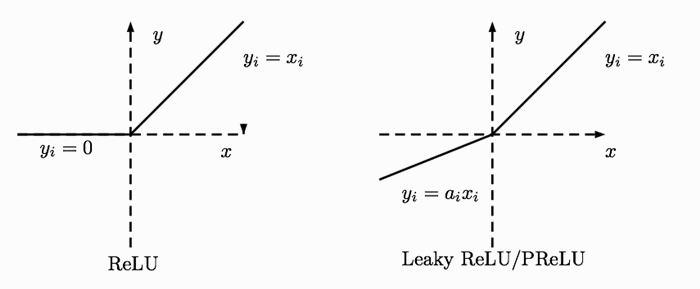
* Classification model.
* LeakyReLU activation function.
* Convolutional layers.

## Generator:

Generator takes random noise as input and transform the noise to a flattening layer. After this vice versa steps are done comparing with discriminator (starting from 4x4x512).

In the end a fake image of spider of size 64x64x3 is got.

* Deconvolutional layers.
* ReLU activation function (last layers – tanh function).



*Picture 6: ReLU and Leaky ReLU activation functions* [*(Byte-Master-101 (2018). “ReInventing Neural Networks”)[6]*](https://www.codeproject.com/Articles/1220276/ReInventing-Neural-Networks)

## Losses:

* Two loss functions (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 BCELoss.

## Training process:

GPU is used for training process. It takes several hours to train a usual model. It’s needed more powerful video card and more time for learning process to implement a stronger network.

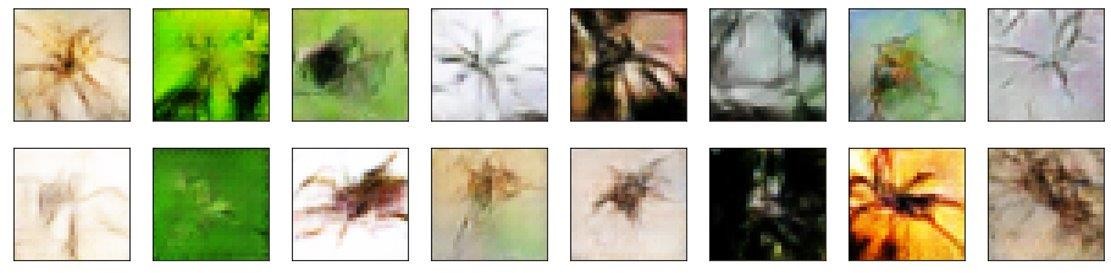
During the learning process the weights are changed. The weights represent the strength of the connection between neurons and decide how much influence the input has on the output.

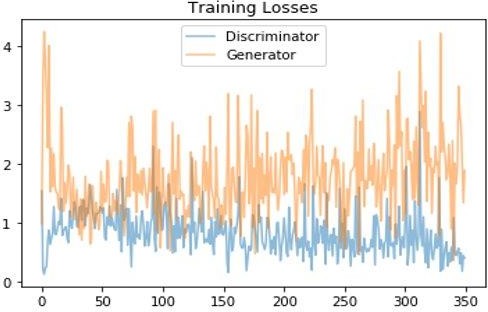
## Code:

[github](https://github.com/valeblond/myThesis/blob/master/project/gan_spiders2.py)

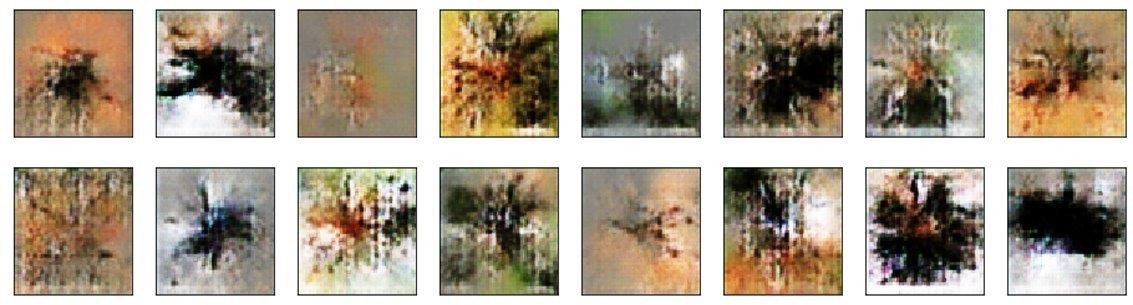
# **RESULTS**

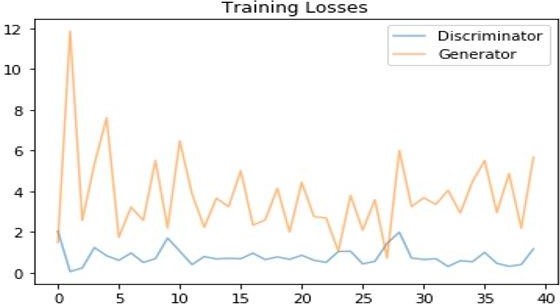
* batch size=32; epochs=40; image size=64x64





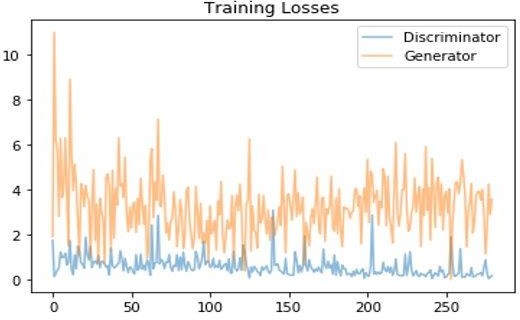
* batch size=32; epochs=40; image size=64x64





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





|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

# **BIBLIOGRAPHY**

1. Wikipedia. “[Generative adversarial network](https://en.wikipedia.org/wiki/Generative_adversarial_network#cite_note-GANnips-1)”.
2. CodeEmporium (2018). “[Generative Adversarial Networks – FUTURISTIC & FUN AI !](https://www.youtube.com/watch?v=O8LAi6ksC80&amp;list=PLTl9hO2Oobd-1jxZ01__NjibEY6h15Kha)”.
3. Jos van de Wolfshaar (2018). “[Semi-supervised learning with GANs](https://medium.com/%40jos.vandewolfshaar/semi-supervised-learning-with-gans-23255865d0a4)”.
4. Roman Trusov. “[GAN Deep Learning Architectures - review](https://sigmoidal.io/beginners-review-of-gan-architectures/)”.
5. Gerasimos Spanakis (2018). “[Fig 1- uploaded by Gerasioms (Jerry) Spanakis](https://www.researchgate.net/figure/GAN-conditional-GAN-CGAN-and-auxiliary-classifier-GAN-ACGAN-architectures-where-x_fig1_328494719)”.
6. Byte-Master-101 (2018). “[ReInventing Neural Networks](https://www.codeproject.com/Articles/1220276/ReInventing-Neural-Networks)"