# Preprocessing a car dataset for GANs

Tomasz Garbus tg370795
Jan Ludziejewski jl371158
Michał Naruniec mn360386

February 2019

## Introduction

We were inspired by the article [1]. What we noticed about the generated car photos (and seems true about most GAN applications in general) is that they all had a random background. We decided to preprocess a car dataset by removing the backgrounds and replacing them with on color. We also tried to train two GAN architectures and compare results with the preprocessing and without it.

## 1 Snowball and segmentation

### Manually labeled data

To remove and easily substitute the background in the pictures, we utilized architectures solving the semantic segmentation problem.

We decided to use Cars dataset. Unfortunately, we did not have the ground truth labels for the pixels. Using GIMP's intelligent marking feature, we have manually labeled 50 pictures of the data set. We also created 60 element validation set and 60 element test set. It would be better to have larger validation and test sets, but due to other responsibilities this is all we had time for. However, the project's goal was to check if the concept of removing backgrounds might improve effects of GANs' training and not achieving state of the art segmentation accuracy. Therefore, we assumed that reasonable intersection over union metric on these validation and test sets, along with manual check, will be enough to carry out this proof of concept.

### U-Net snowball

The first architecture we decided to use for the segmentation task was U-Net. We needed to expand the training set, so we implemented a snowball - we trained UNet on first 50 samples. We also wrote script that collects yet unlabeled pictures and generates a prediction one by one. It is shown to the user, and they can accept, reject or correct[7]. If they decide to correct, a simple image editor allows to save manually enhanced marking. From time to time, training was ran on new samples to make the job easier. We were running the training on a GPU in one of our laptops.

We have labeled 900 pictures using that method, but even then we had to make a lot of corrections. To cope with that, we decided to use different network - FCN32.

### FCN32 snowball and prediction

We used FCN32 based on VGG, which we learned about from laboratories. We trained smaller version of the network on Google Colaboratory (2048 convolution filters per layer) and plugged it into the snowball script. We were very pleased with the results, as the 100 images we labeled needed few or no corrections. Having 1000 training pictures now, we trained both U-Net and bigger version of FCN32 (4096 convolution filters per layer) an ICM server. After benchmarking, we generated predictions for roughly 7000 more training images, which we could then use in GANs' training.

### U-Net and FCN32 comparison

We managed to achieve those mean IoU:

- U-Net: 81 % on validation set
- U-Net: 80 % on test set
- FCN32: 89 % on validation set
- FCN32: 87 % on test set

Our FCN32 was also learning faster. U-Net was producing labels with "holes", while FCN appeared not to have this problem, which is the next reason why it suited us better.

## 2 GANs

After finishing the segmentation task, we proceeded to try out GANs on preprocessed dataset.

### Preprocessing

The preprocessing consisted of 4 phases:

- Erasing the background using our segmentation network
- Cropping the image to square size
- Resizing the image (to either 32x32, 64x64 or 128x128)
- Data augmentation (flipping)

### DCGAN

With DCGANs we have encountered many learning stability issues and thus experimented with various hyperparameters. The advised[5, 6] hyperparameters for training DCGANs are:

- mini-batch size 128
- Leaky ReLU slope 0.2
- Adam optimizer with learning rate = 0.0002 and $\beta_1 = 0.5$
- Weights initialized randomly with normal distribution, mean = 0, st. dev = 0.02

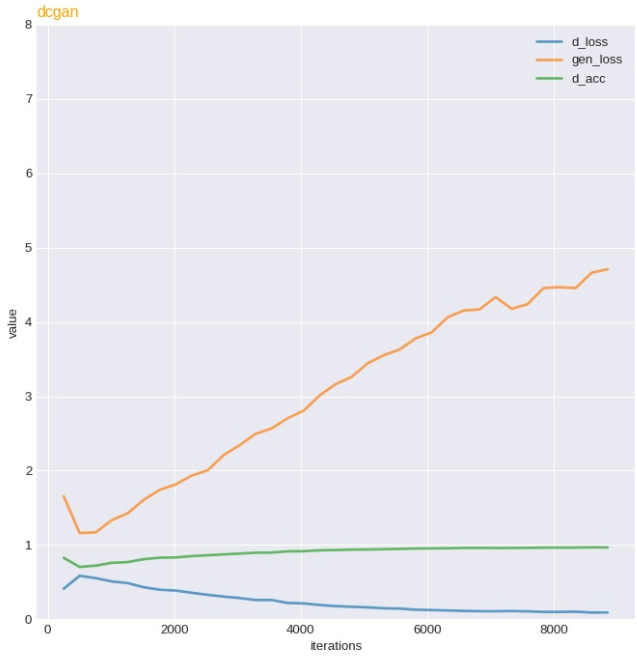
Some sources also suggest traning the discriminator multiple times per one iteration of generator. We have tried different ratios of discriminator/generator iterations and some changes in generator architecture (filters number, removing one layer for faster learning). However in all experiments either:

- Discriminator was learning much faster and generator was not really improving
- Or generator started overfitting, achieving small losses without producing meaningful images

In most experiments we were using 64x64 images, though we have also tried 32x32 and 128x128.

Figure 1: Sample of generated "cars".

Figure 2: Learning process plot. It reflects the unstability of the model. Discriminator was getting better much faster than the generator so the generator loss was increasing.




## BEGAN

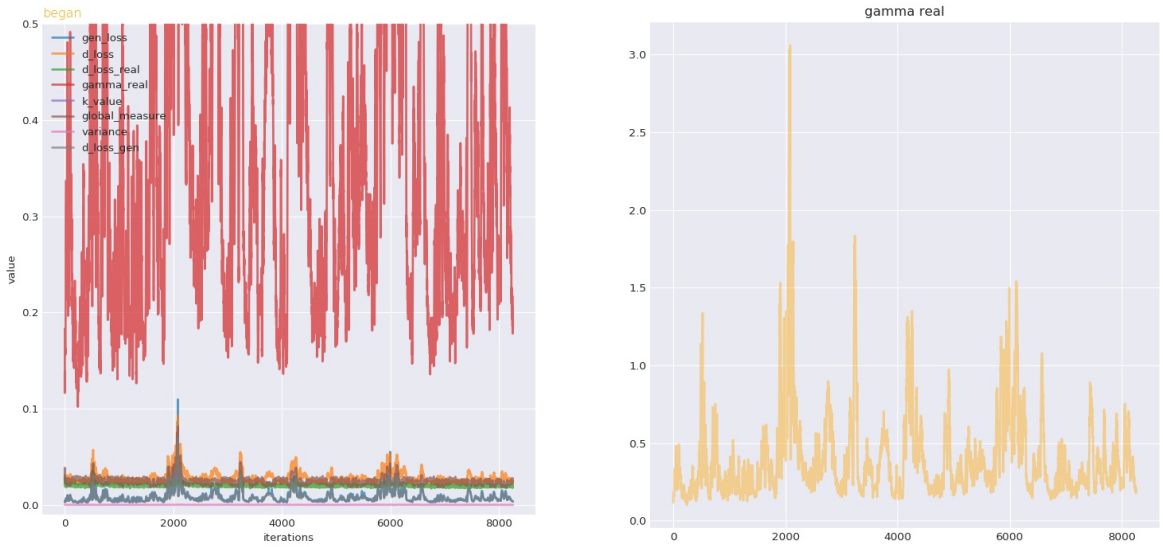
Complementary to standard DCGANs, we experimented with currently state-of-the-art generative architecture called Bounded Equilibrium Generative Adversarial Networks, using two new techniques:

- Discriminator is autoencoder trying to perfectly reconstruct dataset images, and badly reconstruct generator images with parameter k.
- Parameter k is adjusted "momentum-like" to achieve designated ratio between discriminator and generator, called gamma.

Model is created using two modules, encoder which is fully convolutional network with 3x3 convolutional kernels and 2x2 max polling, without any regularization (as in BEGAN paper), and with decoder being its reverse. Generator is decoder, and discriminator is encoder with decoder. However,

experiments where equilibrium was reached ended with generator generating black images with some random shades, independently of chosen gamma. BEGAN architecture in its original paper had assumption about normal distribution of input data, but with note that experimentally it works anyway, we conjecture that cars are more diverse and less normalized then, especially for a weak classifier. I. e. pixelwise mean of normalized faces is recognizable as a human (and is a good point of start for a generator), but ours car dataset would give a black image with blur in the middle, state which was sometimes achieved by our network but which also quickly returned to plain black. This is also connected to locality of features, i. e. parts of image of face are different organs chosen independently retrieve a better result than independently chosen parts of non-normalized by position car (and probably more horrific).

Figure 3: BEGAN late learning process. No progress despite stable (but highly variant) equilibrium (desired gamma is 0.5).




## 3 Summary

Cars seem to be a difficult domain for GANs, especially since they are less normalized than faces. We were hoping to help the network learn by removing the background from images. While it didn't seem to improve the quality of generated images, we have achieved good results in segmentation task.

## 4 References


1. https://medium.com/@utk.is.here/keep-calm-and-train-a-gan-pitfalls-and-tips-on-training-generative-adversarial-networks-edd529764aa9

2. Cars dataset used in paper: 3D Object Representations for Fine-Grained Categorization
   Jonathan Krause, Michael Stark, Jia Deng, Li Fei-Fei
   4th IEEE Workshop on 3D Representation and Recognition, at ICCV 2013 (3dRR-13). Sydney, Australia. Dec. 8, 2013.
   https://ai.stanford.edu/ jkrause/papers/3drr13.pdf

3. U-Net paper:
   https://arxiv.org/pdf/1505.04597.pdf

4. FCN32 paper:
   https://arxiv.org/pdf/1605.06211.pdf

5. DCGAN paper:
https://arxiv.org/pdf/1511.06434.pdf

6. DCGAN hyperparameters recommendations:
https://gist.github.com/shagunsodhani/aa79796c70565e3761e86d0f932a3de5

7. Simple paint in Python which inspired the label editor:
https://gist.github.com/nikhilkumarsingh/85501ee2c3d8c0cfa9d1a27be5781f06