Vision and Perception Project Report

Isabella Claire Adriani, 1714559

September 2019

1 Introduction

The task of the project is to translate the cGAN written with keras from chapter 8 in a code that uses tensorflow. I worked alone on this project.

2 Conditional generative adversarial network (cGAN)

The cGan is deep network where both generator and discriminator are conditioned on some information, segmented images in this case. The cGan in general as many applications like image-to-image translation or text-to-image synthesis.

The following graph represents the cGAN model used in this project[1]:

Figure 1: Tensorflow graph

3 Changes

I made some changes to the code since during developing the project these changes made the model have better results.

For the generator I used the pip2pix loss version. For the discriminator I take the mean of the initial discriminator loss. I changed the learning rate to 6e-5. I used a separate discriminator for the fake images instead of using an adversary network. I resize and normalize the images when all the images are loaded instead of doing after each loaded image. This has save a lot of time and space.

[1]The model is very large therefore the image of the graph has been resized to fit the page. The full size image is stored in the same directory as this document.

4 Dataset

The dataset, cityscapes, is a set of segmented images and target images, images we want to reproduce. The 256x256 images are turn int black and white images and then normalized for better training. The images portray city life, cars, people, buildings, etc.

Since in the dataset there aren't testing images I created a function that splits the validation set into validation and test sets.

For computational purposes I ran the code only for 10 epoch and I used only use 1000 training images.

5 Results

As we can see in the graphs, the model learns how to generate images. The losses decrease over time. This is a task that usually is done with more epochs, though is performing well after ten epochs.

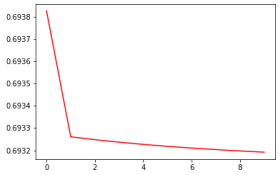

Figure 2: Discriminant losses distributed over ten epoch

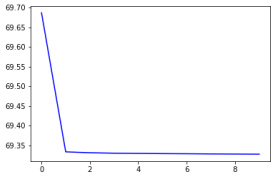

Figure 3: Generator losses distributed over ten epoch

The model generates images that are very close to the target image. Is possible to notice that already after the first epoch the model creates images very close to the target one except for some blurring. After the 10th epoch a lot of the blurring as faded away given a sharper look to the image. If it was possible to run the code for more epochs, the images would be exact reproductions of the target images.

In conclusion, despite the good results, if it were possible to run it on quite a few more images and epochs we would have even better results.

Figure 4: Real Image

Figure 5: Generated image after the first epoch

Figure 6: Generated image after the 10th epoch