[Face Detection]

[Activity Report No.1]

Radu Beche

[In this report I have presented the what I have achieved so far after completing the research phase. I will describe the algorithm that I am going to implement, how it works and what the roadmap of this project will be.]

[Face Detection]

[Activity Report]

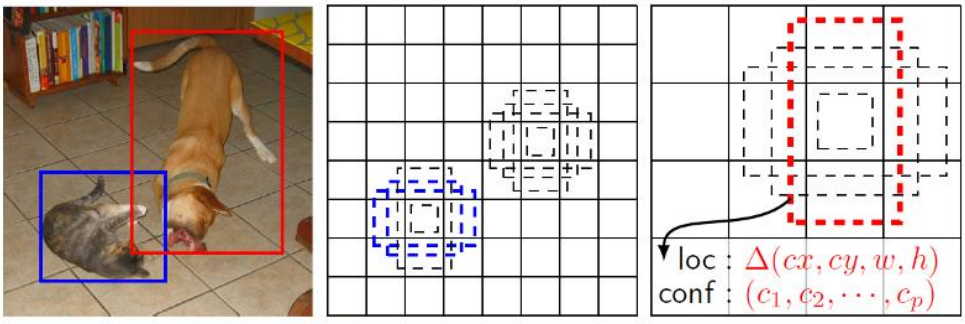
What's new since last time?

Since last week I have documented myself a little bit more about the architecture of an "object detectors" and how do they do what they do. I have started my own implementation of SSD (**S**ingle **S**hot **D**etector) and I have decided that, as a first step I will train it on a dummy dataset* (thus I being more lightweight for my computer during training). This dummy dataset will be made by me.

What is Single Shot Detector?

The Single Shot Detector is a real-time, one stage, deep learning object detection algorithm that was first published in 2016 that achieved state of the art performances. SSD300 (input image is 300x300 pixels) can run at 40 FPS on an NVIDIA GTX1080 without any optimizations having an average accuracy of 75% IOU. By applying quantization, pruning and reducing the input image size, this model can be ported even to a mobile phone with decent FPS and accuracy performance

How does Single Shot Detector work?

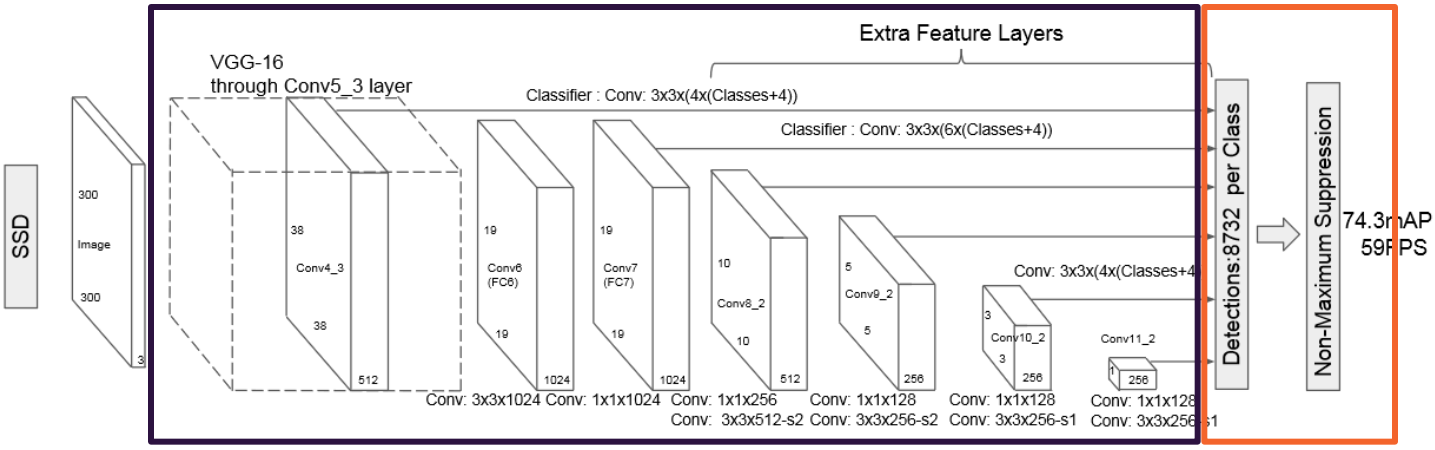


(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Project Roadmap

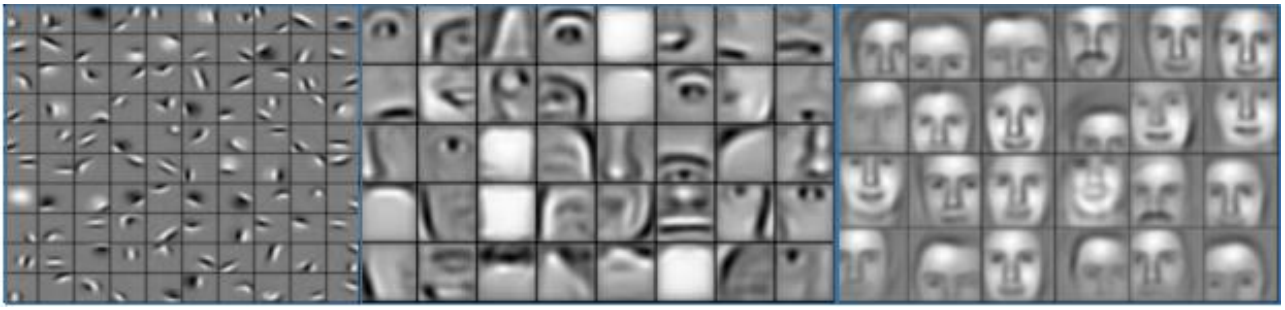
- Research period
- Implementation and training on a dummy dataset
- Training on an actual face dataset
- Deploy application

To understand this, I will try to explain and exemplify each part of the below algorithm.



The algorithm is composed from 2 parts, a network that will extract the features of the face and one part that interprets that output.

The first part, is a convolutional neural network encoder. Its job is to extract the features of the object. Each layer's job is to extract a certain part of the object at different scale. For example, the first layers job will be to detect the smallest edges of a face, the second, will be to combine them and detect more complex features like a nose or a mouth, and finally the whole face. Note that this just a dummy explanation of how a convolutional encoder works. This is the trainable part of the algorithm.



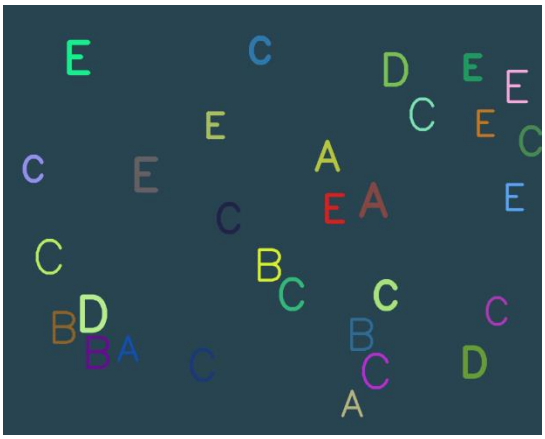
The second part is the way you interpret and decide if a certain region of an image really represents a face. The image is divided into a grid named, these grids will have a center point named anchor. For each anchor the following will be predicted: a bounding box x, y, height and width and a classification (a list of numbers that will represent the class of that object) and a confidence parameter.

These boxes will be filtered using a non-maximum suppression algorithm. And voila, these are the bounding boxes for our objects.

My dummy dataset

This is going to be a dataset composed of images with multiple letters in them. I will try to detect the letter and position using the above described algorithm.

A sample image is:



My implementation

My implementation is going to be done using Python with the well-known framework Keras and Tensorflow. As from know I have implemented the first part, but I am in a little bit of trouble getting the second part to work accordingly thus there are a lot of math formulas. After finishing the second part I will start the training the network. My progress so far can be found on the GitHub page.