Face Detection

Amil George
TUM
IMAT:03658271
Email:amilgeorge@gmail.com

Daniil Pakhomov
TUM
IMAT:03658275
Email:daniil.j.pakhomov@gmail.com

Tanuj Ghinaiya
TUM
IMAT:03662075
Email:tanujg82@gmail.com

Ansh Kapil
TUM
IMAT:03657530
Email:anshkapil@gmail.com

Abstract

Face detection task poses many challenges such as lightning conditions, intensity variations, occlusions such as beards, glasses and different poses. It is important to note that Face Detection is a subtask of a much bigger task called Object Detection. Various algorithms have been developed in recent years regarding the problem after Viola and Jones revolutionized the face detection task by introducing a real-time capable face detection algorithm using boosting and attentional cascade back in 2001. Neural Network based algorithms provide even more discriminative power. We will discuss different approaches for face detection; using both boosting techniques and Convolutional Neural Network based techniques; their implementation and comparisons in this project. First we discuss a slightly modified version of Viola-Jones algorithm [1] which we implemented for open source library with real-time performance. Secondly, we show how we implemented convolutional neural network called lenet-5 [2] and show the results after training. And, finally, we show a combined approach [3] that uses attentional cascade and convolutional neural networks as weak classifiers inspired by Vila-Jones approach.

1 Introduction

1.1 Goals of Face Detection

According to [4], the goals of face detection is “Given an arbitrary image, the goal of face detection is to determine whether or not there are any faces in the image and, if present, return the image location and extent of each face“. The detection should be accurately localized i.e. the detection should be at the place where the face is and only one detection per face should be there. Secondly, the amount of false positive detections should be as low as possible i.e. the detections where there is no face in reality but it is detected by the algorithm. Thirdly, the amount of false negatives should also be as low as possible i.e the amount of faces which actually exists but are not detected as faces.

1.2 Challenges of Face Detection

1. **Need for high accuracy**: If error rate of classifier is 0.001, on the first two scales of an image of size 1000×1000 with detection window size 24×24, we get 1858 error detections.
2. **Compicated false positives**: Some parts of an image can resemble a face. For example, a simple drawing of a face will be detected as a face.

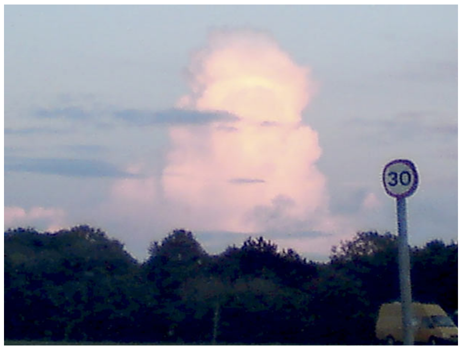

Figure 1: Face-like object.

3. **Extensive detection of the same face**: Due to fact that some classifiers have a small invariance to translation and scale, there will be a lot of detections around a face. We only want one detection for a face. Special algorithms should be applied like Non-Maximum Suppression.

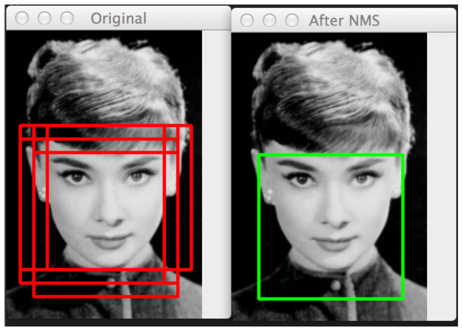

Figure 2: Extensive detection of one face.

4. **Negative examples choice problem**: Since the Face Detection task is a task of binary classification we have to gather a dataset with two classes: faces and non-faces. The non-faces part is a problematic section because literally anything can be used for the second class.

2 Datasets

Face detection task requires that the model used for face detection should be resilient and be able to handle variations in images in form of variances of lighting conditions, complex false positives, occlusions, poses and intensity variation. This can be incorporated in the current implementation by training the model using a dataset which incorporates all the variations that are to be considered for the face detection task. Another important aspect of the task is the ability to recognize non faces in the image, which in turn requires the dataset to incorporate non face images as part of the dataset to effectively distinguish between face and non-face patches of images.

Thus the dataset should be comprised of both face and non-face images. For the purposes of this project we created a dataset by combining various datasets into single dataset which we use for training our model. The final dataset contains about 70.000 face and 200.000 non face images after post processing of the raw data. The following datasets were used in the construction of the final dataset.

2.1 Positive examples

As positive examples for our classifiers we used the Caltech 10,000 Web Faces database [5] The dataset has a total of 10,524 faces of in various settings. The coordinates of the eyes, nose and the center of the mouth are provided. After preprocessing and augmentation we have 70.000 examples.

2.1.1 Preprocessing

In order for classifiers to learn properly from the faces examples, the examples faces should be consistently aligned and have some variation in intensity and position.

We applied the following preprocessing steps before using the specified dataset:

- **Rotate image to align the line between eyes**: As faces in the dataset are differently al-ligned and we also want to augment our dataset, we decided to first align all faces to be in the same manner. In order to do so, we took the line between eyes and rotate the image to make this line horizontal.
- **Crop faces consistently**: with the same offset above and below eyes, and from the left and right. So, the classifier won't be confused.
- **Save the proportion when doing it**: Very important point because otherwise we train our classifier on the faces that are not of a real proportion and the results may be bad.
- **Augmentation**: Augment the dataset by perspective transformation and by placing it on a random background. We have also added intensity variations.

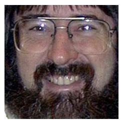

Figure 3: Consistently cropped face example.

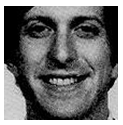

Figure 4: Consistently cropped face example.

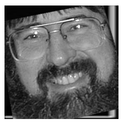

Figure 5: Random transformation example.

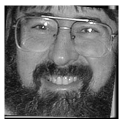

Figure 6: Random transformation example.

2.2 Negative examples

Usually for negative examples people take just background images without any humany body parts (pictures of buildings, rooms, houses and so on). None of these contain people. And a lot of classifiers make mistakes on the people's clothes or some other part. We experienced this after training our classifier on usual dataset without human body parts. This observation led us to an idea to use pictures containing human body but with the faces replaced with random noise.

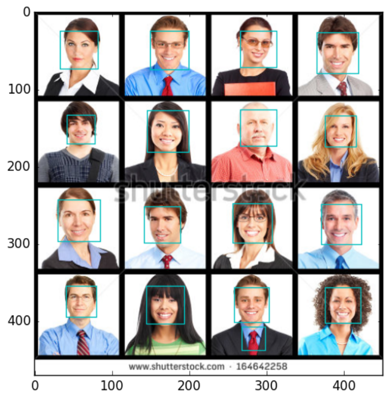

Figure 7: Adaboost Attentional Cascade trained on usual dataset. The tie on the second image from the bottom right corner is detected as a face. Consequence of using negative examples without human body.

We used Facial Landmarks in the Wild [6] dataset for this purpose. The annotated Facial Landmarks in the Wild [6] dataset contains 25,993 annotated faces. The main interest for us in this dataset was that it contains a large part of human body and also background scenes.

2.2.1 Preprocessing

In order to use the the provided dataset we first preprocessed it in the following manner:

- **Replace rectangles containing faces with random noise**: This was done in order to avoid placing a face in negative examples which will confuse the classifier.
- **Extract random patches from the dataset on different scales**: In order for the dataset to capture more variation we extracted random patches on different scales from the images with eliminated faces.

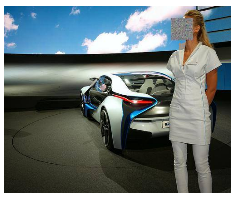

Figure 8: Example of an image with eliminated faces from which the negative samples were taken.

3 Classifiers

3.1 LeNet-5 Convolutional Neural Network architecture

These are the facts about the CNN that we implemented:

- LeNet-5 is a convolutional network originally designed for handwritten and machine-printed character recognition.
- Softmax function is optimized.

Figure 9: Example of an image with eliminated faces from which the negative samples were taken.

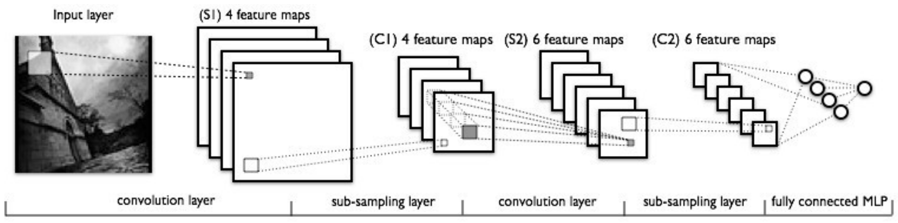

Figure 10: LeNet-5 CNN architecture.

- Gradient descent.
- Minibatch training with the size of a batch 60.
- Random initialization is as suggested for $tanh$ function.
- No regularization is used.
- Mean is computed from training set and subtracted from each image during training and evaluation.
- Uses grayscale images.
- A network with big structure. Takes time to compute.

3.1.1 Results after training

After traning we were able to achieve the following results:

- 0.01 test set error.
- 0.008 best validation set error during training.
- 0.001 training set error.

Test set and validation set were created from the same dataset patches that weren't used for training.

3.1.2 Results on a real image with scale search

We ran our classifier on the sample lena image and got the following results after search on multiple scales:

3.1.3 Code

The code containing classifier can be found the the repository of our group. Also, while implementing this exercise we made a small optimization to Theano that was merged `https://github.com/Theano/Theano/pull/2865`

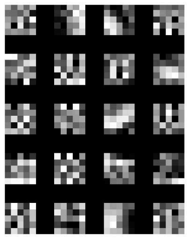

Figure 11: LeNet-5 first convolutional layer weights.

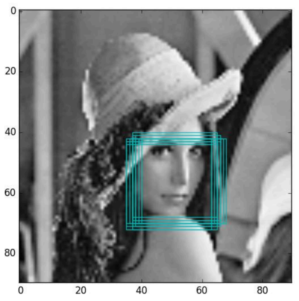

Figure 12: LeNet-5 multi scale search without the non-maximum suppresion to show the extensive detection.

3.2 Adaboost attentional Cascade

3.3 Adaboost attentional Cascade architecture

These are the facts about the classifier that we implemented:

- Uses simple, integer, fast-to-compute, handcrafted features: Multiblock Local Binary Patterns.
- Uses Attentional Cascade of classifiers that speeds up the process of evaluation significantly.
- Uses integral image and features are scaled instead of image. Much faster.
- Real-time performance.
- Uses gentle adaboost to train classifiers on each stage of cascade.
- Uses whole dataset for training.

3.3.1 Results after training

- 0.01 best validation set error during training.
- 0.005 training set error.
- 0.011 test error.
- Number of positive samples 3000.
- Number of negative samples 1500.

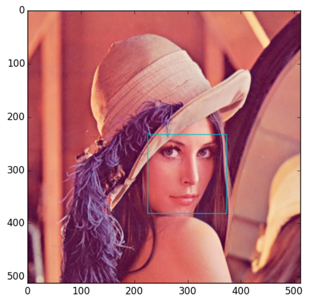

Figure 13: Adaboost Cascade multi scale search wit non-maximum suppresion.

3.3.2 Results on a real image with scale search

3.3.3 Code

This classifier was implemented for scikit-image using Cython. The following pull requests contain all the code: `https://github.com/scikit-image/scikit-image/pull/1536` `https://github.com/scikit-image/scikit-image/pull/1570`

3.4 Cascade of Convolutional Neural Networks

3.5 CNN cascade architecture

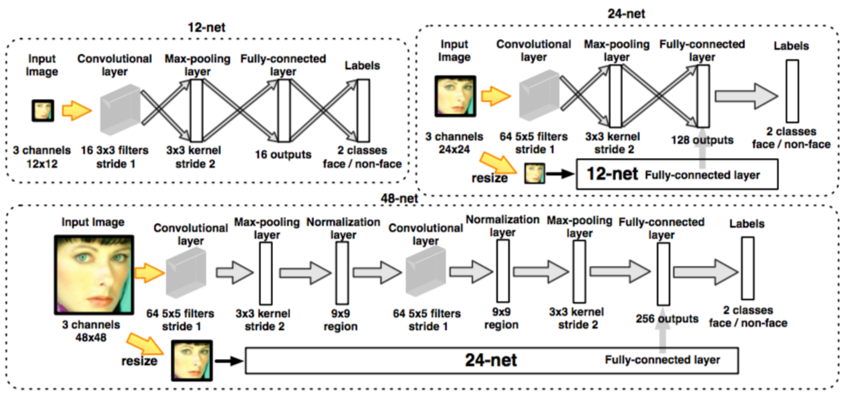

Figure 14: CNN Cascade Architecture.

Facts about the CNN cascade:

- Uses softmax error function, same minibatch size and regularization as lenet-5.
- Same cascade approach as in Adaboost Cascade.
- Faster to compute because negative examples are rejected faster on the first stages of cascade.
- Real-time performance.

The cascaded convolutional neural network consists of multiple stages. The image pyramid is scanned exhaustively using the CNN cascade. Low confidence regions are rejected in the low resolution stages like the 12 Net. Higher confidence regions or regions that cannot be rejected in the low resolution stages are passed to the next layer for careful examination by a larger network.

In our work we only used first two stages from the picture above.

3.5.1 Results after training

- 0.0145 best validation set error during training.
- 0.0053 training set error.
- 0.015 test error.

3.5.2 Results on a real image with scale search

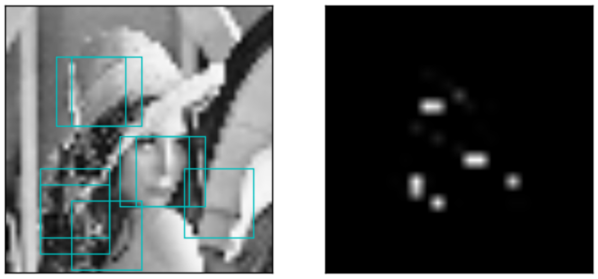

Figure 15: 12 net evalutaion and confidence map. Selected windows are passed to 24 net.

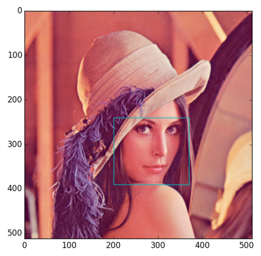

Figure 16: Final output of the cascade.

3.5.3 Code

The code can be found in the repository of our group.

References

[1] Face detection based on multi-block lbp representation L Zhang, R Chu, S Xiang, S Liao, SZ Li - Advances in biometrics, 2007 - Springer

[2] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998d). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 22782324.

[3] A Convolutional Neural Network Cascade for Face Detection H Li, Z Lin, X Shen, J Brandt - Proceedings of the IEEE , 2015 - personal.stevens.edu

[4] Yang et al, Detecting Faces in Images: a Survey, Pami 2002

[5]`http://www.vision.caltech.edu/Image_Datasets/Caltech_10K_WebFaces/`

[6]`https://lrs.icg.tugraz.at/research/aflw/`