Face Detection Using Cascaded Convolutional Neural Networks

Joakim Lilja and Caroline Heidenreich

Project for Image Based Recognition and Classification

Abstract. A simple cascaded convolutional neural network for single face detection is proposed. Due to the computationally heavy nature of deep networks a shallow initial net, *24Net*, is implemented for fast discrimination of the sub-images generated by a sliding window together with an image pyramid. This allows for a small portion of the input data to be passed to the deeper *48Net* for final detection. The architecture enables computationally efficient detections with a total time of 1260 ms for a 640x480 gray-scale image with an accuracy of 88%.

Keywords: Face detection, Convolutional Neural Networks, Deep learning, Python

1 Introduction

Face detection is a problem with many applications in security, photography, social media and human computer interaction. In many cases, the detection is carried out online and is thus required to be computationally efficient
Object detection in general is a difficult problem due to the immense variations of scale, illumination, perspective, occlusion, etc. Various methods have been proposed for developing robust detection algorithms focusing on extracting predefined features that are invariant to the aforementioned variations and then train a model based on these features. However, with the increase of available computational power, the use of large artificial neural networks, or specifically convolutional neural networks - *CNN*, for tackling this problem has gained in popularity. This is also referred to as *Deep Learning*. Here both the type of features along with the model is trained using large sets of data. Notably, at the ImageNet Large Scale Visual Recognition Competition (ILSVRC) workshop at the European Conference on Computer Vision 2012 (ECCV) in Florence, Italy, a submission using CNNs won the competion by decreasing the error rate of previous year winner by half.

Since CNNs are computationally heavy we present a simple variant inspired by [1]. A cascaded architecture of neural networks is proposed for quickly filtering out background areas from a large amount of images using a small CNN and then doing the more computational-intensive precise detection in a second step on a smaller amount of images but with a larger CNN. Since a full implementation of the setup used in [1] is beyond the scope of this project, we implement a simpler

detector based on the proposed cascaded model. Furthermore, we restrict our detector to grayscale images with a single face.

For training and evaluation the *"FDDB: A benchmark for face detection in unconstrained settings"* [2] is used.

This is a project report of our work conducted during the course DD2427 - Image Based Recognition and Classification at Royal Institute of Technology, Stockholm. The goal of this project is to

- Summarize state-of-the-art work related to this task
- Implement and apply a cascaded CNN for single face detection in grayscale images

The implementation is done in python where there are a number of libraries that can be used for deep learning. We chose the neural networks library `Keras` [3] that runs on top of the software package `theano` [4] that is optimized for the handling of large arrays.

2 Background

During the last decade, the algorithm proposed by Paul Viola and Michael Jones in 2001 has been considered being state of the art for face detection[5]. Their algorithm is based on recognizing a pattern that usually constitutes a face - a horizontal dark band that is characteristic for eyes that are shadowed and a vertical brighter line where the nose bridge is. The algorithms checks for both of those patterns in a cascade and if they are detected it looks for further characteristic features of a face. This face recognition method brings accurate and fast results for faces that are seen from the front but is not reliable for faces seen from different angles. [6]
Approaches to solve this problem with neural networks have already been developed during the past two decades, e.g. by [7] and [8]. Notably, the latter one proposed an approach in two stages. However, before 2012, neural networks were regarded as too prone to overfitting. Only the recent advances in convolutional neural networks have led to a breakthrough in image recognition in general and face detection in particular. To apply CNN's to the face detection problem multi-layered convolutional networks are trained with a large database of annotated images (in this case faces from different angles and in different illuminations). This approach was e.g. implemented by Farfade et al. in [9]. They built a database with 200,000 images of faces and 20 million non-face images and trained a convolutional neural network called AlexNet. The obtained tool - known as Deep Dense Face Detector - has shown to bring accurate results also for partly occluded faces and faces not seen from the front. Many face detection algorithms use a sliding window to tackle the problem of localization. A window slides across the image searching for faces in all of its parts. This approach was also followed in [9]. Furthermore the images are scaled down to allow for faces with different sizes. Similar techniques are used in other object detection tasks [10].

A face detection algorithm that shares our special focus on computational efficiency has been proposed by Li et al. in [1]. Their method to achieve accurate but less computational intense results is to use a cascade of six convolutional neural networks. The first shallow net scans coarsely through the image with a 12×12 pixels large window and rejects large parts of the background. The remaining windows are adjusted in the 12-calibration-net by perturbing the window parameters by either applying spatial or scale transformations with the goal to increase the confidence value of the window. The output of this net is then rescaled to 24×24 large images and given as input to the 24-net which is further reducing the number of images and also followed by a calibration net. The deeper 48-net is finally applied to the remaining images. It is computationally more demanding but only a small number of images is given as input to this network. Their algorithm results in a performance of 770ms on a 640×480 VGA image. A simplified implementation based on their approach is described below.

3 Approach

3.1 Convolutional Networks

Inspired by the work in [1] we propose a similar but simplified cascaded network structure. We define two networks, a shallow initial CNN - *24Net* and a more complex CNN for the final detection - *48Net*, see fig. 1. The model structures on these follow the definitions in [1] apart from the following differences

- In [1] they have two setups of the different CNNs, one for binary classification and one for calibrating the bounding boxes of the faces. We combine the purpose of these two into a single CNN for each detection window size (24 and 48). These output a scalar y in the range [0,1].
- We are not implementing their first CNN, the shallow *12Net*.
- We are not concatenating the outputs from *24Net* into the last layer of *48Net*, this is referred to as multi-resolution solution, see [1] for details.
- The local normalization layer is not implemented in Keras and was therefore omitted.

The models were simplified to allow for an implementation within in the scope of this project.

3.2 Pre-processing, Sliding window and Image pyramid

Since the task is not only to classify whether an image contains a face or not but also to localize it, the image is split up into multiple sub-images at various scales. A sliding window is then defined as a square of size 24×24 and for each scaled image the detection window slides over the scaled image with a chosen step size extracting a sub-image of the same size as the window which is then fed into the *24Net*. The procedure is visualized in fig. 2. For pre-processing prior to *48Net* the detection window is fixed at the center with size 48×48 and thus

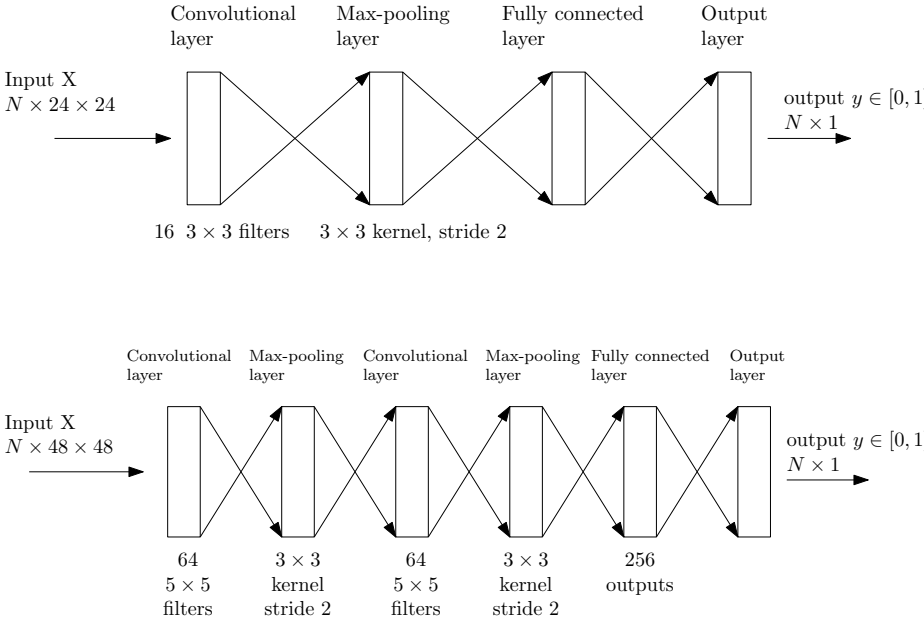


Fig. 1: Network architecture for the *24Net* and *48Net*

no "sliding" takes place. This was chosen in order to reduce the dimensionality of the input. The data is also normalized to zero mean and a standard deviation of one.

3.3 Cascaded Network Architecture

The cascaded architecture consists of generating the sub-images from the original images using the image pyramid and the sliding window. These are then passed through the *24Net*. The sliding windows with an output $y^{24} > T_{24}$ of the output are then re-sized by a factor of five and used to retrieve a new sub-image from the original image. This is done to increase the detection area and account for mis-adjusted squares. This sub-image generates a new image pyramid with a detection window of size 48×48 fixed at the center. This was chosen to minimize the number of inputs to the next net. These sub-images are then passed through *48Net* to yield the final prediction values $y \in [0, 1]$. The maximum value is then matched with its corresponding window and original image. Fig. 3 illustrates the procedure.

3.4 Labels

In order to train and evaluate our model a ground truth is needed. The data set, [2], provides annotations in ellipses. We re-calculate these into squares with size equal to twice the largest radius of the ellipse projected on the vertial axis. The

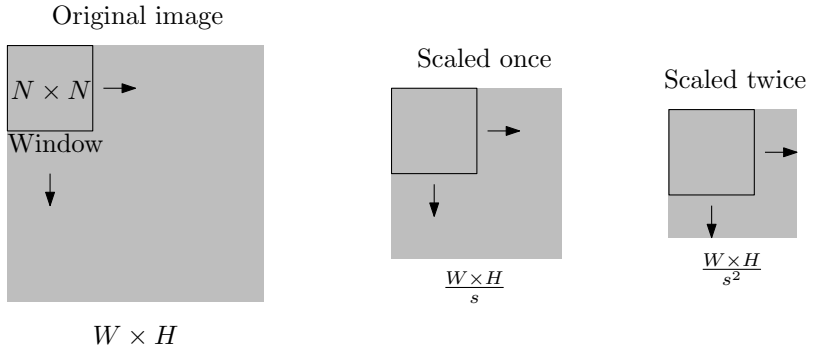


Fig. 2: Visualization of the image pyramid and sliding window. The original image is down-scaled by a factor s, n times and for each time the sub-images enclosed by the sliding window is used as input to the corresponding CNN.

label of a sub-image is then calculated based on how well it is aligned with the original image's square annotation, see fig. 4. The label, l, is then calculated as

$$\begin{aligned} m &= \frac{2}{w_a^2} \\ l_x &= \texttt{max}(1 - m(\Delta x_1^2 + \Delta x_2^2), 0) \\ l_y &= \texttt{max}(1 - m(\Delta y_1^2 + \Delta y_2^2), 0) \\ l &= \texttt{min}(l_x, l_y) \end{aligned} \tag{1}$$

where w_a is the width of the square annotation box. Note that when the misalignment, say in y direction for illustrating the point, $(\Delta y_1^2 + \Delta y_2^2)$ is equal to $1/m$, then $l = 0$. Further assume that $\Delta y_1 = \Delta y_2 = \Delta$. Then we get $\Delta^2 = \frac{1}{2m} = \frac{1}{4}w_a^2 \Rightarrow \Delta = \frac{1}{2}w_a$. A misalignment of $1/2$ of the annotation box width thus corresponds to $l = 0$.

3.5 Training

We split the data set into a training set and a test set. This results in 821 images containing a single face used as positive training samples. Furthermore we add 300 images to be used as negative training samples taken from the SUN data set [11]. The ground truth is calculated according to eq. (1). We then generate the sub-images prior to *24Net*, and use all of these to generate the sub-images prior to *48Net* as described above. For optimization we use the stochastic gradient descent method with Adadelta [12], to adaptively set the learning rate, together with the square error loss function. Multiple training sessions with batch sizes between 16-128, step size 24 and shuffled data were conducted.

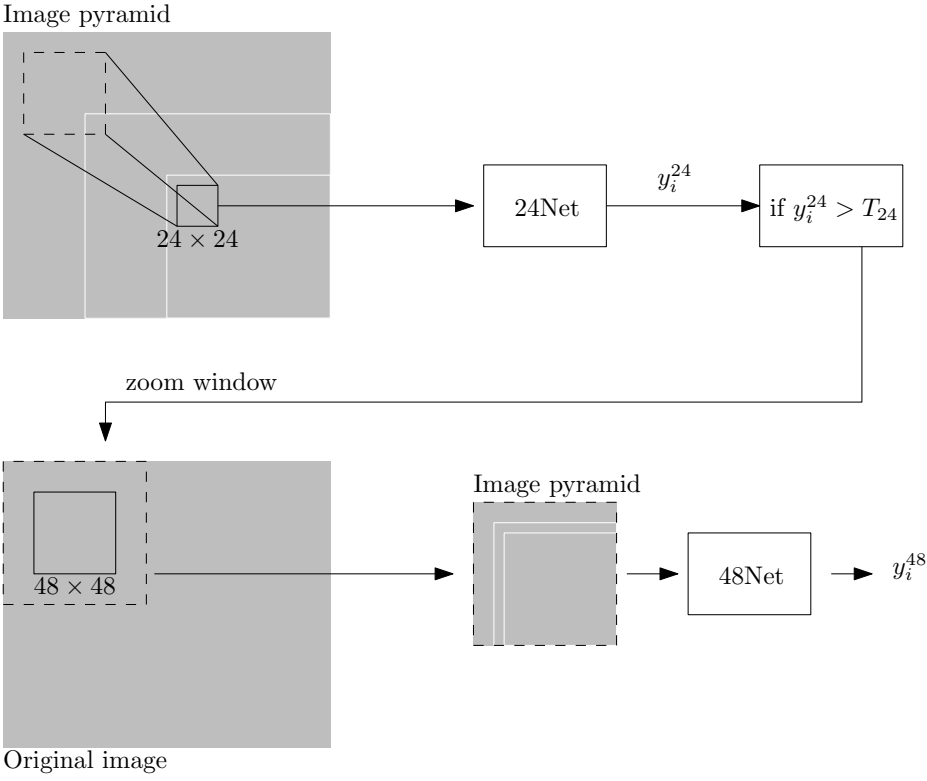


Fig. 3: Illustration of the cascaded network flow. The image subject to detection generates an image pyramid. For each scaled version of this image a window of size 24×24 slides over it extracting a sub-image that is passed through the *24Net*. The sliding window's corresponding bounding box with respect to the original image is stored (dashed line). If a sub-image, i, results in a label $y_i^{24} > T_{24}$ its corresponding bounding box is enlarged by a factor five and the underlying sub-image is retrieved. This, in turn, generates a new image pyramid for that specific sub-image with a detection window of size 48×48 centered within the bounding box. The sub-images generated by this detection window are then passed into *48Net*

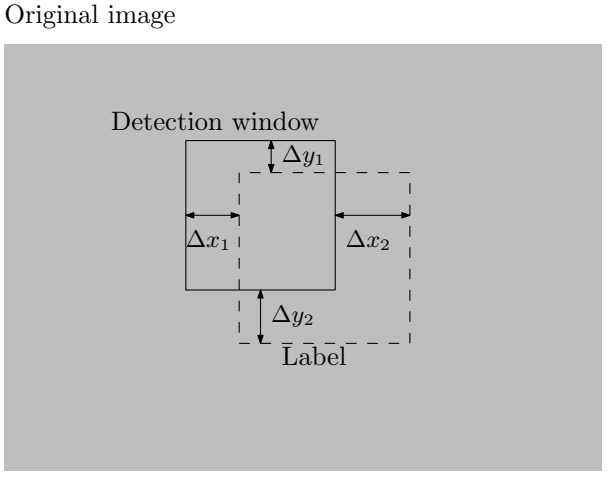


Fig. 4: Measure of alignment for a detection window with the annotation of the image.

4 Experiments

4.1 Setup

To evaluate the proposed network architecture the test set is pre-processed and passed through the trained models. The threshold T_{24} is chosen such that the top 10% of the outputs from *24Net* are chosen as input to the deeper *48Net*. These outputs, $y^{48}_{k,i}$ are mapped back to their corresponding image, k, and bounding box, i, and the bounding box with the highest prediction is chosen, i^*. If $y^{48}_{k,i^*} > T_{48}$ it is considered a detected face.

To evaluate if the bounding box of a detection is a detected face or not we calculate its similarity with the ground truth according to the same decision function for labeling data, see eq. (1). If the similarity is greater than zero, it is regarded as a detected face.

The metric for evaluating the performance is chosen as the number of correctly detected faces divided by the total number of images, since all images in our data set contains images with a single face. This is referred to as the accuracy of the detector. We further test two step sizes of 12 and 24, set the scaling factor to 1.5 for the image pyramid prior to *24Net* and fix the scaling factor prior to *48Net* such that three scaling levels are generated with the last one fitting the bounding box perfectly.

4.2 Results

With a step size of 24 an accuracy of 88% is achieved. If the step size is reduced to 12 the accuracy is bumped to 92%. This is at expense of the detection time. For a 640x480 image the time increases from 1260 ms to 2340 ms. By plotting

a histogram of the similarities, see fig. 5, between the bounding boxes we see that the model manages to fit the bounding boxes relatively well, this can be understood by noting that the majority of the similarities lies in the upper region. Fig. 6 shows the output bounding boxes for *24Net* with a step size of 12 and 24 respectively. Fig. 7 shows a sample of the successful detections while fig. 8 shows some of the false positive detections and misaligned bounding boxes.

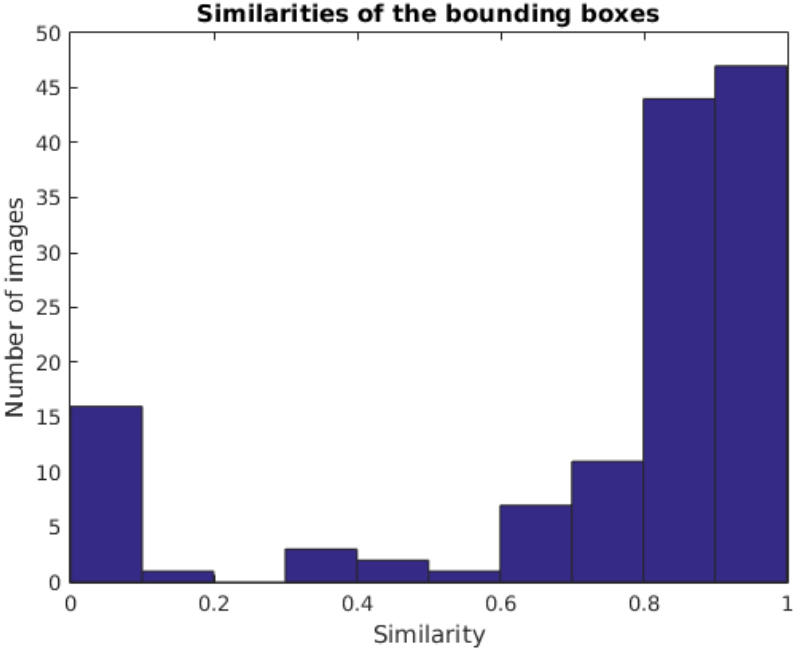
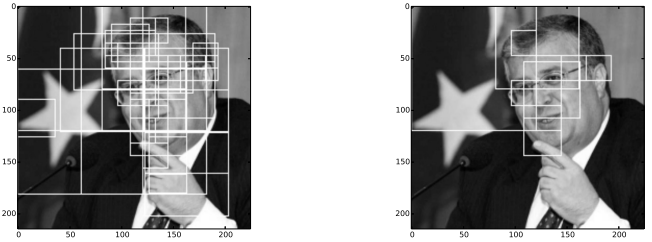
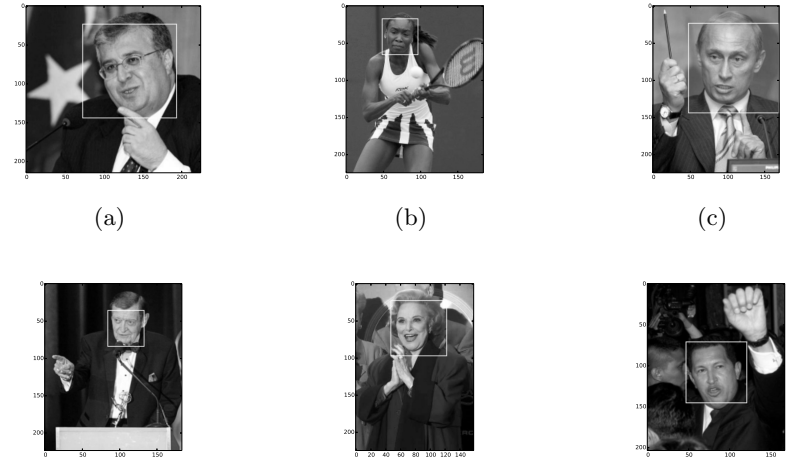


Fig. 5: Histogram of similarities between the bounding boxes of ground truth and prediction



(a) (b)

Fig. 6: Example for bounding boxes for the 10% best predicitions from *24Net* for stepsizes of 12 and 24

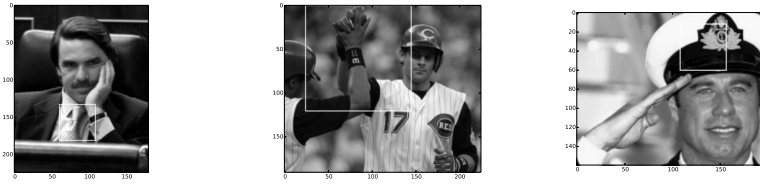


(a) (b) (c)

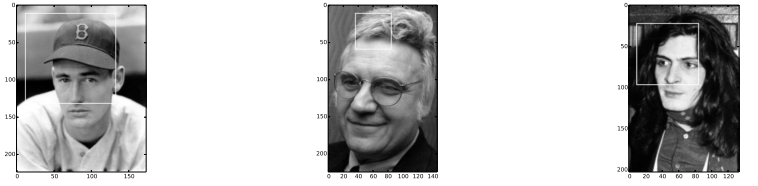
(d) (e) (f)

(g) (h) (i)

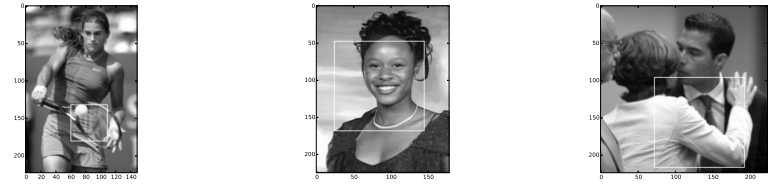
Fig. 7: Examples for cases where our algorithm brings satisfactory results



(a) (b) (c)



(d) (e) (f)



(g) (h) (i)

Fig. 8: Examples for cases where our algorithm fails. In some cases, as a), b), c), g), i), another object (tie, ball, hand) gets higher predictions than the face, in some cases the bounding box is misplaced. For the latter case better results could have been obtained by introducing spatial/scaling pertubations

5 Conclusions

Although limited by computational resources and model complexity our results are still promising with an accuracy of 88% on the test set with a step size of 24. For a 640x480 image the total detection time is 1260 ms on a Quad core 2.83 GHz CPU. If the step size is reduced to half, we bump the accuracy to 92% at the cost of a detection time of 2340 ms. Our results confirm that a cascaded network of the form proposed in [1] is a computationally efficient method for face detection using convolutional neural networks. It is also noteworthy to conclude that our method for labeling the date yields satisfactory results. There are, however, multiple improvements that can be made.

- First and foremost, a larger training set would be beneficial. Due to our limited computer resources we only trained on half of the benchmarking set FDDB [2] together with 300 images from the SUN data set [11]. There are larger data sets available.
- For simplicity we only created an image pyramid of the inputs to *48Net*. To get better aligned bounding boxes some spatial perturbations of the inputs are assumed to yield more accurate results.
- Adding an additional shallow CNN might increase the discrimination of sub-images prior to the last net further lowering the computational load.
- The model tends to favour certain objects before faces, such as ties and hands. A training session could be conducted where such objects are negatively weighted to see if better results can be achieved.
- The obvious limitation that our algorithm only detects single faces. This was due to an initial, not very thought-through, decision. Later it demanded too large changes in the implementation for us to be able to finish within the time frame of the project. It is however not a restriction to the model but the implementation alone.

References

1. Li, H., Lin, Z., Shen, X., Brandt, J., Hua, G.: A convolutional neural network cascade for face detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 5325–5334
2. Jain, V., Learned-Miller, E.G.: Fddb: A benchmark for face detection in unconstrained settings. UMass Amherst Technical Report (2010)
3. Chollet, F.: Keras. https://github.com/fchollet/keras (2015)
4. Theano Development Team: Theano: A Python framework for fast computation of mathematical expressions. arXiv e-prints **abs/1605.02688** (May 2016)
5. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. Volume 1., IEEE (2001) I–511
6. Review, M.T.: The face detection algorithm set to revolutionize image search. https://www.technologyreview.com/s/535201/the-face-detection-algorithm-set-to-revolutionize-image-search/ Accessed: 2016-05-23.
7. Garcia, C., Delakis, M.: Convolutional face finder: A neural architecture for fast and robust face detection. Pattern Analysis and Machine Intelligence, IEEE Transactions on **26**(11) (2004) 1408–1423
8. Vaillant, R., Monrocq, C., Le Cun, Y.: Original approach for the localisation of objects in images. In: Vision, Image and Signal Processing, IEE Proceedings-. Volume 141., IET (1994) 245–250
9. Farfade, S.S., Saberian, M.J., Li, L.J.: Multi-view face detection using deep convolutional neural networks. In: Proceedings of the 5th ACM on International Conference on Multimedia Retrieval, ACM (2015) 643–650
10. Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y.: Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229 (2013)
11. Xiao, J., Hays, J., Ehinger, K.A., Oliva, A., Torralba, A.: Sun database: Large-scale scene recognition from abbey to zoo. In: Computer vision and pattern recognition (CVPR), 2010 IEEE conference on, IEEE (2010) 3485–3492
12. Zeiler, M.D.: Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701 (2012)