SURVEILLANCE THROUGH FACE DETECTION

Uppala Sehouriey
CS19BTECH11015

Ambati Sanjay Krishna
CS19BTECH11013

Nalari Sushma
CS19BTECH11006

L. Harshavardhan Reddy
CS19BTECH11023

Abstract

In recent times, with the current evolving technology, there is a higher need to ensure the safety and security of the people. This has been a significant problem, so many research experts are trying to improve the security conditions to overcome these security issues. Hence for security purposes, computer vision techniques such as face detection, motion detection, pattern detection, etc., can be used for better surveillance systems. In this paper, we mainly focus on surveillance issues caused around households, offices, etc. Also, we will be discussing how we can overcome them with our proposed advanced computer vision techniques on surveillance, which can identify any breach in the security or detect any suspicious behavior that will cause security issues, etc. In Further, we will mainly go through how face detection can be used for surveillance systems and detailed analysis of the algorithm and approach used.

1. Introduction

Continuous surveillance monitoring will be a mundane task for humans born to do much more. With Constant monitoring devices such as CCTV surveillance cameras, we can record high-quality footage, but it still needs human observations for decision making. Security has been a challenging and fundamental problem.
On the other side, we have a rising deep learning technology configured to detect and categorize objects through visual data. Using this ability in the surveillance field shall reduce tedious work of going through complete video footage.

1.1. Motivation

Video Surveillance networks worldwide are large with millions of cameras, and With the rise in facial recognition and deep learning techniques, we can make drastic security enhancements. Computer vision for surveillance sought out to be a viable solution. Object detection can detect unfriendly or hostile situations and immediately notify. The goal here is to use deep CNNs to build an autonomous surveillance system that we can use for different purposes like face detection objects detection(for example, detecting the number plates in parking space). It helps in identifying spots of crime quickly and improves public safety.

2. Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks

Face detection and alignment, which are essential to face applications, may become difficult due to occlusions, pose variations, lightings, etc. There are many DL techniques that can yield significantly good results, but there are drawbacks to most designs. In this paper, let's see the cascaded convolutional networks, how they came to use, frameworks used, Architecture and training, etc.

3. Approach

Framework : There will be three stages in total. We will see each step in detail. The image will be initially resized into different scales to build the image pyramid. This image pyramid will be input for stage 1.

Stage 1 : In this stage, we use the Proposal Network (P-Net, a fully conventional network) to get the candidate windows and their bounding box regression vectors. These bounding box regression vectors will be used to calibrate the candidates. Finally, the highly overlapped candidates will be merged using NMS (Non maximum suppression).

Stage 2 : In this stage the candidates obtained from stage 1 will be passed to Refine Network, where many false candidates are removed, and calibration with bounding box regression, NMS candidate merge are also performed.

Stage 3 : The main focus is to identify the facial landmarks in this stage. Generally, the network identifies

five different facial landmark positions. The rest of the things are the same as above.

Architecture: Previously, even after using multiple CNNs for object detection, the performance was limited. As we are doing face detection, we might not need many filters; so we will be reducing the 5*5 filter to a 3*3 filter. With these changes, we will get better performance and less runtime.

3.1. Training

We have three tasks to train CNN detectors according to the three stages mentioned above: face/ non-face classification, bounding box regression, and facial landmark localization.

Face classification : This is a two-class classification problem. For classification, the cross-entropy loss is calculated for each sample.

Bounding box regression : Here, for each candidate window, the offset is predicted between itself and the nearest ground truth. Euclidean loss is employed for each sample.

Facial landmark localization : This is also the same as above. We minimize Euclidean loss for facial landmarks coordinate and ground-truth coordinate.

3.2. Experiments

We will compare our face detector against the standard Face Detection Data Set and Benchmark (FDDB), WIDER FACE, and Annotated Facial Landmarks in the Wild (AFLW) benchmark. In Proposal Network, we randomly crop several patches from WIDER FACE to collect positive, negative and part faces. For landmark faces, we crop faces from CelebA. After evaluating the performance of our face detection method against FDDB and WIDER FACE, our detector outperforms all other approaches by a large margin. Also, face alignment outperforms all approaches. The runtime of our joint face detection and alignment is also significantly less.

3.3. Conclusion

We have seen the multi-task cascaded CNNs based framework for joint face detection and alignment. Also, the results produced from our methods outperform several benchmarks by some margin, all while keeping run time relatively low. So this can be regarded as one of the successful methods for face detection and alignment.

For more detailed information, you can cite the literature's we have summarized [6] [3]

4. Further Literature Review

Let us see some of the other DL techniques used for face detection and alignment. We can compare these techniques with cascaded convolutional networks and see the advantages and disadvantages.

4.1. Faceness-Net

Faceness-Net briefly consists of three stages. They are generating partness maps, ranking candidate windows by faceness scores, and refining face proposals for face detection. Let us see in detail what will happen in these three stages.

In the first stage, the entire image will be given as input for five CNNs, and each of the five CNNs will give a partness map as output. Here, each map will indicate the composition of specific facial components in the input image, such as hair, eyes, nose, mouth, and beard. Finally, all these five partness maps will be combined to give a partness map that will indicate the location of the face in the given input image.

In the second stage, candidate windows will be extracted from partness maps according to the facial part configurations. Then, these candidate windows will be ranked according to their faceness scores. As the final faceness scores are calculated by taking an average of individual parts there might be a large number of false positive windows produced.

In the third stage, the candidate windows that are produced in the above stage will be refined by using a trained multitask CNN. The face classification and bounding box regression will also be jointly optimized in this stage. More information on FACEness-NET can be found in [4]

4.2. Multi-scale Detection Cascade

In this technique we will use multi-scale two-stage cascade framework and use the divide and conquer strategy. In this technique, there are two stages. We will generate multi-scale proposals from a fully convolutional network in the first stage. In the second stage, we will use the windows generated from the first stage to generate a face and non-face prediction using a multi-task convolutional network. Let us see in detail what will happen in these three stages.

In the first stage, where the multi-scale proposals are generated, we will generally take a set of fully convolutional networks and train them jointly for face classification and scale classification. The images will be divided into

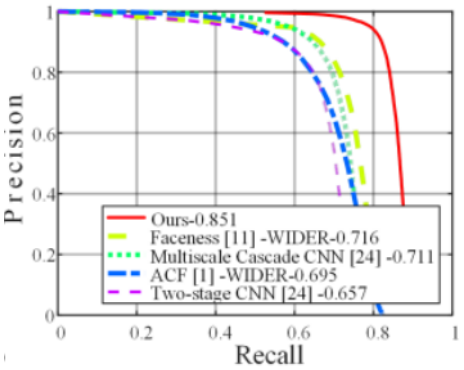


Figure 1. Easyset

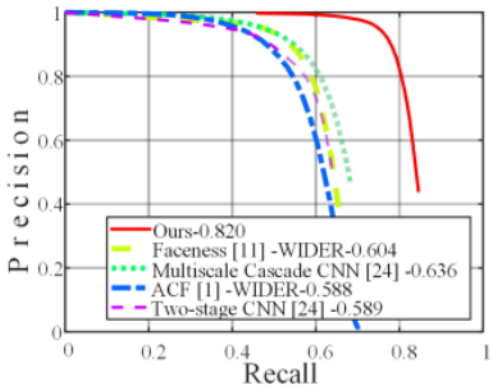


Figure 2. Mediumset

several categories and again divided into several subclasses using predefined scale subclasses in each group. Finally, the probability of the presence of a face will be calculated, which uses some functions and methods such as stochastic gradient descent with backpropagation.

In the second stage, the windows got in the previous stage will be refined. Similar to the last stage, the face classification and bounding box regression will be jointly trained. In bounding box regression we will predict a position of its nearest ground truth bounding box for each window. Also, the Euclidean and cross-entropy losses are used for bounding box regression and face classification. More information on Multicascade-NET can be found in [5]

4.3. Conclusion on above described algorithms

As we now have some basic idea about Faceness-net and Multi-scale detection cascade, we can now see and conclude that these two techniques are not as good as our initial multi-task cascaded convolutional networks. We can observe from the figure(obtained from [6]) that the multi task cascaded convolutional networks will perform far better than Faceness-net, Multi-scale detection cascade, and various other methods.

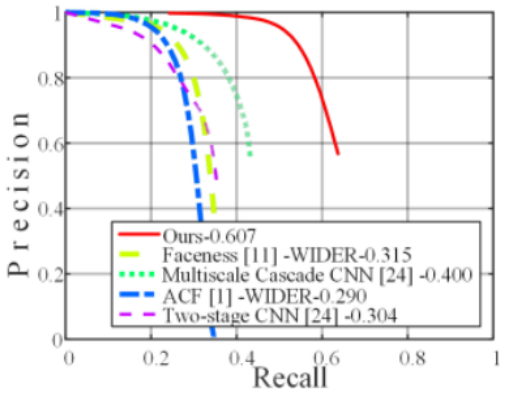


Figure 3. Hardset

5. Preliminary Results

5.1. Results replicated from literature

We have used mtcnn packages and modules to replicate and test the models on an image. Output Format :

```
{
    'box': list [ ],
    'keypoints':
    {
        'nose': ( ),
        'mouth_right': ( ),
        'right_eye': ( ),
        'left_eye': ( ),
        'mouth_left': ( )
    },
    'confidence': __
}
```

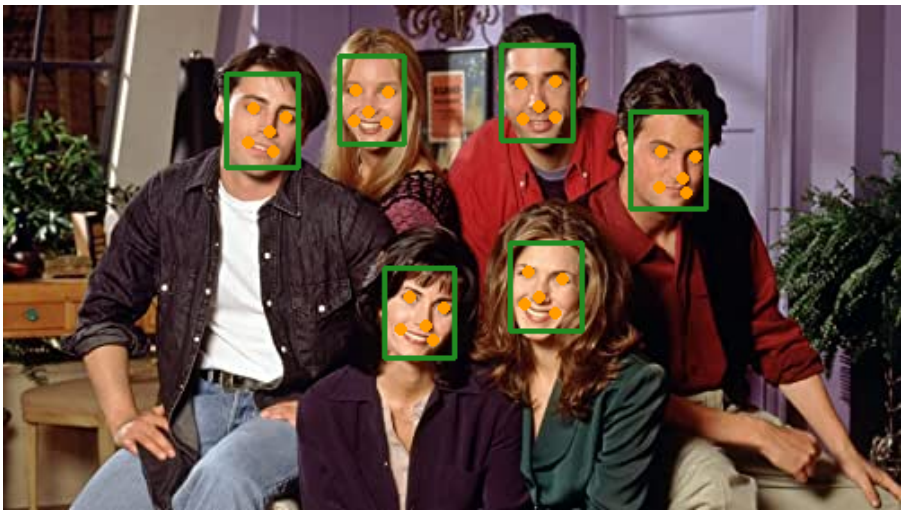
You can see the results of testing mtcnn algorithm on two images one containing a single face and other containing multiple faces in 4.

6. Observations

The Mtcnn model has been very much renowned for its performance. Coding the mtcnn model involves three CNN architectures PNet, RNet and ONet.

Before we dive through some code, let's look at some of the images used for training the mtcnn model. We will consider the WIDERFACE image training dataset throughout this paper. An interesting point to observe is that most of the images consist of straight faces, that is the orientation of faces in most images is almost the same. Few sample

(a) Einstein face detection



(b) Friends face detection.

Figure 4. Face Detection using MTCNN face detector.

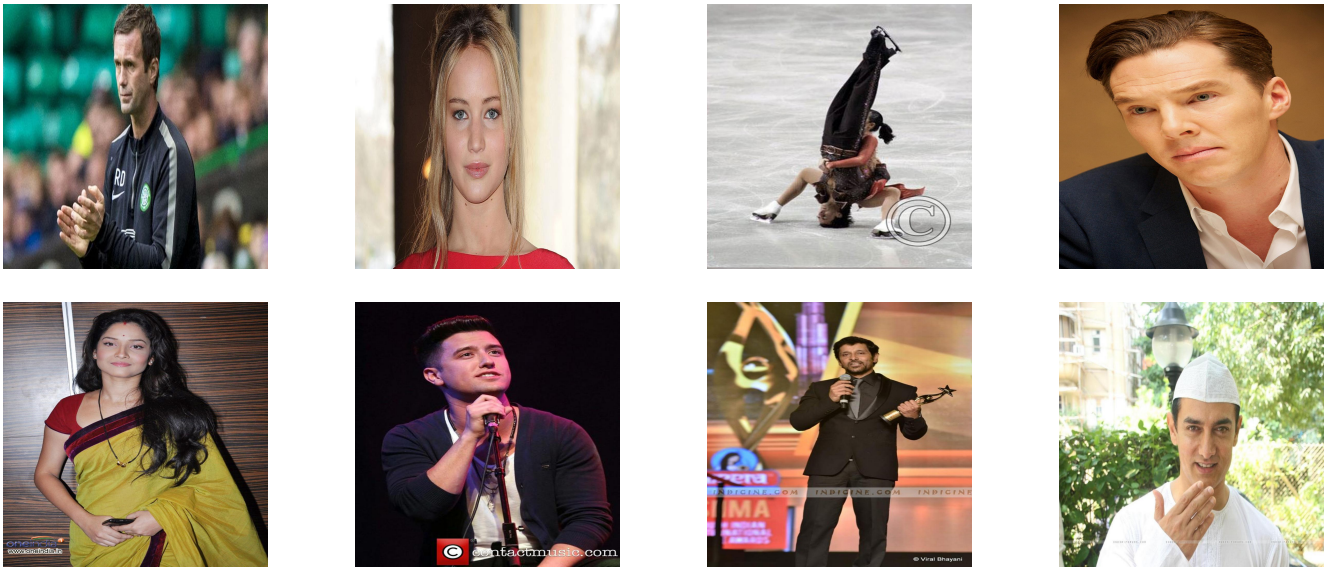


Figure 5. Sample Images in WIDERFACE training dataset

images from WIDERFACE dataset are here 5. There isn't much of a rotation. In further sections of this paper, we will try to build a mini-scale version of mtcnn and see its performance on **inverted faces**.

7. MTCNN PNet Algorithm and Data Preprocessing

Let's consider the PNet (Stage-1) of the mtcnn model only for testing.

Data Preprocessing

The WIDERFACE dataset consists of 32,203 images and label 393,703 faces. The training dataset consists of an image and the label data for the image consists of the bounding boxes of faces present in the image. Keeping the bounding boxes aside, training the PNet requires both positive and negative samples. The required training data is acquired in the following steps.

Step-1 : Consider images with one face, since preprocessing images with multiple faces will be complicated.

Step-2 : Reshape the images such that the face part(surrounded by bounding box) can be viewed in

four different scales(preferably less than 12 since the input to PNet is a 12x12 image)
Step-3 : Crop the reshaped images with 12x12 kernels
Step-4 : Label the cropped images as positive if the face part is completely present in the image
Label the cropped image negative if there is no Intersection of the cropped part with the face part.
Step-5 : Leave the rest of the cropped images which have partial face-part embedded in them, as they might affect our models accuracy.

Ensuring the same number of positive and negative samples would be better.
Consider another dataset which has the images and inverted images of the present dataset. Use an PNet model with CNN architecture in a similar manner as in [1]

Train two PNet models independently, One with only straight images dataset and other with the dataset containing inverted images also.

8. Results

Testing the PNet models on a testset containing straight and inverted images, both the models gave around 50 percent accuracy for a training dataset of around 10,000 images and a testing dataset of 5000 images. A great advantage of mtcnn model is that even though PNet accuracy is low, RNet and ONet could still manage to refine the results.

The key Takeaway here is that, even though we have trained the second model on both straight and inverted images, the performance of it on differently configured faces has not increased significantly.

The above results were also observed in the Complete MTCNN model. MTCNN model was observed to give false results when an inverted positive labeled image is given as input whereas it gives correct results for the same un-inverted image 6 and 7 . When it comes to surveillance, we need a robust model that can overcome the face configurations in the image.

Here comes the idea of introducing Affine Transformations on input images rather than training the model for different configurations of input image.

9. Affine Transformations

Affine transformations are geometric transformations that preserves the lines and parallelism in but not necessarily the distances and angles. Affine transformations were used after face detection and before sending the data into face recognition models, thus making the models more robust [2] .
The addition of using face normalization and affine transformations will improve our models robustness in the face detection step also. Affine transformation Algorithms can act as a Black-box like layer in between the input and CNN layers that sends multiple transformed images of the input. The final output can be taken as the Union of all the outputs the MTCNN model produces. This allows the MTCNN model to be more robust.

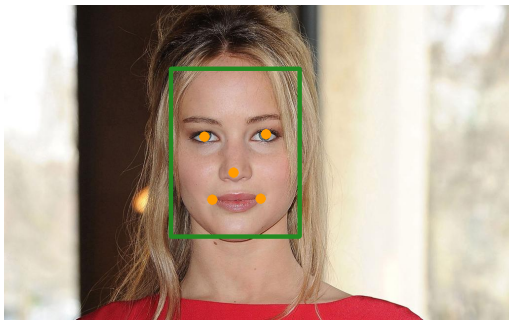


Figure 6. Straight Face getting detected by MTCNN

Figure 7. Inverted Face not getting detected by MTCNN

10. CONCLUSIONS

Security automation using Deep learning can bring drastic changes in the field of security processes. The approach of using models like MTCNN can reduce human efforts and provide faster response rates. Automated Security analysis could also help in improved investigation accuracy. Training a model with all possible data configurations is not the only way to improve its accuracy. Application of basic Affine Transformations can help you detect better and improve the models performance.

Further Study can be made as an insight into addition of traditional Face Recognition model added in sequence to the Face Detection model with changes discussed in this paper.

References

[1] Sachin Sudhakar Farfade, Mohammad Saberian, and Li-Jia Li. Multi-view face detection using deep convolutional neural networks, 2015. 5

[2] Xiaofang Jin and Yijia Xiao. An enhanced expression recognition based on sparse representation using data normalization. In *2021 8th International Conference on Computational Science/Intelligence and Applied Informatics (CSII)*, pages 43–48, 2021. 5

[3] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *International journal of computer vision*, 128(2):261–318, 2020. 2

[4] Shuo Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. From facial parts responses to face detection: A deep learning approach. *CoRR*, abs/1509.06451, 2015. 2

[5] Shuo Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. WIDER FACE: A face detection benchmark. *CoRR*, abs/1511.06523, 2015. 3

[6] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, Oct 2016. 2, 3