Proceedings of the 3rd International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/4/2023315

The future of video-based face detection: Conventional methods VS. deep learning

Qingyao Liao

School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China

19281044@bjtu.edu.cn

Abstract. Recently, the problem of intelligent video analysis in the field of computer vision is receiving considerable attention worldwide. Several studies have proposed some classic methods to implement video-based face detection and face recognition. However, it has confirmed that the performance of video analysis can still improve in order to make it applicable to more scenarios. This article discusses using conventional methods and deep learning to achieve face detection. Two main issues need to be solved to accurately observe the performance of face detection, method selection, and model construction. AdaBoost and multi-task convolutional neural network (MTCNN) are selected to conduct the experiment. The results indicate that it is possible to combine these two methods to realize video-based face detection with higher performance.

Keywords: video analysis; face detection; video-based; AdaBoost; MTCNN.

1. Introduction

With the development of techniques for computer vision (CV), intelligent video analysis has become one of the hottest topics in the field of CV. Compared to still images, video data contains a vast number of real-time information to be exploited. Recently, many researches especially focus on face detection and face recognition from video sources. However, there is still plenty of room to improve precision and efficiency.

The techniques of face detection and face recognition will be discussed in this paper. At the begin of the 21st century, traditional algorithms were widely accepted by researchers. Viola and Jones [1] introduced a face detection technique using HAAR cascades and AdaBoost. Kwang [2] implemented a face recognition system by using principal component analysis (PCA). PCA expresses the original data with relatively important information through dimensionality reduction. Additionally, P.N. Belhumeur [3] proposed the Fisherface method and compared it with the Eigenface technique. Nowadays, with the emergence of deep learning, it's possible for people to achieve the state-of-art face detection and face recognition. Zhang [4] have designed the multi-task convolutional neural network (MTCNN) which can be used to solve the problem of face detection and alignment in an unconstrained environment due to various poses, lighting and occlusion. Yi [5] have put forward the DeepID3 network and Jiaolong [6] have presented the neural aggregation network, both have achieved great success on video face recognition. Through investigation and research, this study will find out a best approach that can be applied to video-based face detection by comparing the performance of AdaBoost and MTCNN.

The remaining of this paper will be divided into four parts. To begin with, a short literature review of history and latest researches of face detection and face recognition will be presented. Then, we will show the methodology of this project, which includes AdaBoost and MTCNN. In the next part, the results of these methods will be discussed. Last, a short conclusion will be made.

2. Literature Review

2.1. Previous works

Since the 1990s, numerous approaches of face detection and recognition have been proposed. Firstly, Matthew and Alex [7] developed the Eigenface method for face recognition in 1991. In addition to designing a system for automatic face recognition using feature surfaces, they also demonstrated a method for computing eigenvectors of covariance matrix, enabling computers of the time to achieve eigen decomposition of a large number of face images.

However, the Eigenface method didn't have a wonderful performance from a discrimination standpoint. In this case, the Fisherface method, which owned lower error rates that the previous method, was raised by Peter [3] in 1997. Its projection method was based on Fisher's Linear Discriminant and produced well separated classes in a low-dimensional subspace.

Then, Viola and Jones [1] introduced a classical face detection technique, which used Haar cascades to describe face features and AdaBoost to train hierarchical classifier. In order to enrich the simple features of [1], a extended set of rotated Haar-like features were proposed [8], which could be computed efficiently as well.

In 2002, Kwang [9] came up with Kernel Principal Component Analysis(KPCA). In general, PCA is suitable for linear dimensionality reduction of data. However, KPCA could achieve nonlinear dimensionality reduction of data, which was used to deal with linearly indivisible data sets. Therefore, a KPCA-based face extraction method produced a good performance on face recognition.

His [10] presented a face detection algorithm based on skin color segmentation. Firstly, the color space of the image is transformed from RGB to YCbCr. Secondly, skin region model and simple Gaussian model were established according to the

clustering performance of skin color points in color space. Finally, the skin color likelihood graph is obtained by calculating the skin color similarity, and the face position is determined by smoothing filtering, morphological processing and threshold segmentation.

Entering the 2010s, with the emergence of deep learning method, people became aware that it might be possible to achieve the state-of-art of face recognition using this method. At first, Yi [11] introduced the DeepID3, which included two very deep neural networks designed for face recognition. It was this paper that motivated later studies to further investigate the effectiveness of face detection and recognition with deep learning method.

2.2. Recent works

In this part, we are going to focus on the researches of the last 5 years.

Since more and more researchers have put a lot of time and energy into improvements and innovations, deep learning has become the overwhelming technique in face detection and recognition [12, 13]. With the emergence of some popular networks, such as Region-based CNN(R-CNN), Fast R-CNN, and Faster R-CNN, YOLO (You only look once) framework has been invented for detecting faces [14]. Moreover, several improvements have been proposed [15, 16] in attempts to enhance the performance of face feature discrimination. For example, the marginal loss function concurrently minimizes the intra-class variances as well as maximizes the inter-class distances by centering on the marginal samples [15]. With the combined utilization of softmax loss and marginal loss, we can readily obtain a more robust CNN network. Beyond that, the model trained with large margin cosine loss also have a similar effect [16]. Jiaolong [5] have presented the neural aggregation network (NAN). The aggregation module of NAN contains two attention blocks which adaptively aggregate the feature vectors to yield a singular feature inside the convex hull spanned by them.

On top of that, Zhang [4] have designed a novel framework called MTCNN, which is the representation of face detection method using deep learning. MTCNN uses multi-task cascaded convolutional networks for joint face detection and alignment and each convolutional network does their job respectively.

With the outbreak of Coronavirus Disease 2019 (COVID-2019), detecting faces with masks has become one of the hottest topics. To achieve this function, Amit [17] have proposed a dual-stage CNN architecture, which is capable of detecting masked and unmasked faces and ensures a safe public environment.

In spite of the prevalence of deep learning, there are also some researchers exploring traditional approaches. For instance, Rahmad [18] have made a comparison of Viola-Jones Haar cascade classifier and histogram of oriented gradients (HOG) for face detection. Besides that, Etemad, K., & Chellappa [19] have also achieved face detection by using Local Binary Pattern (LBP) and Support Vector Machine (SVM). The accurate rate of traditional approaches is moderate.

Last but not the least, Pavel and Sebastien [20] proposed a intriguing application of face detection, which is how to detect Deepfake videos successfully. Obviously, the further development of new efficient face detection technology is needed.

After browsing through the studies above, we have come to a conclusion that AdaBoost and MTCNN are both the basic of face detection. Therefore, it is meaningful to study these two methods in depth and propose a decent video face detection method based on them.

3. Methodology

3.1. Problem definition

Face detection is defined as a computer technology that can recognize the presence of human faces in digital images or videos. To achieve it, face detection apps use machine learning and formulas called algorithms to detect faces in larger images or dynamic videos, which may contain many objects that are not faces, such as landscapes, buildings, and other parts of humans (for example, legs, shoulders, and arms).

Face detection is a more extensive concept than face recognition. It has many applications, only one of which is face recognition. Face detection can also be used in autofocus cameras. It can be used to count the number of people entering a particular area. Moreover, it can even be used for marketing purposes.

Figure 1 represents the whole process of our experiment. We will detect faces in video by using AdaBoost and MTCNN. Then, the performance of two methods will be compared and analyzed. Finally, we will arrive at a credible conclusion based on the experimental results.

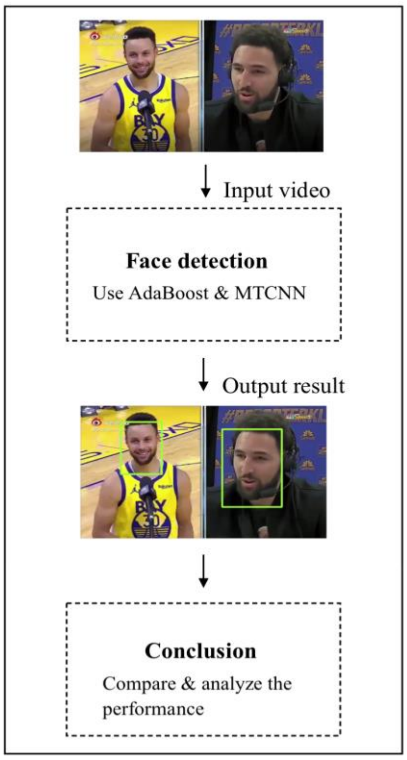

Figure 1. The whole process of experiment.

3.2. Method selection

In this part, we will discuss the methods of face detection from video sources. In conventional face detection methods, we will select AdaBoost for three reasons. First, before the emergence of deep learning, AdaBoost was the most popular and useful method to detect faces in images and videos. To some extent, it could represent most of face detection methods. Second, this method builds a fast and effective face detection framework, which could meet the immediate requirements. Third, it doesn't demand high computing power like support vector machine (SVM), which means it could run easily on embedded, resource-constrained devices.

For deep learning networks, MTCNN will be the most decent choice for this experiment since it is the most classic framework in CNN for face detection and the basis for keeping up with the latest researches. Besides that, its innovative use of cascade networks could improve the efficiency of detection significantly. Moreover, the structure of each network is not so complicated and easy to understand, which could make us conduct this experiment smoothly.

AdaBoost. As early as in 2001, Viola and Jones [1] proposed a classical face detection technique. This method is based on integral graph, cascade classification detector and AdaBoost algorithm. Its framework can be divided into the following four steps:

- Use Haar-like features for detection;
- Use integral graph to accelerate the evaluation of Haar-like features;
- Use AdaBoost to select some rectangular features (weak classifiers) that best represent human face, and these weak classifiers are constructed as a strong classifier according to the weighted voting method;
- Cascade strong classifiers together to improve accuracy.

Haar features, which include three types: edge feature, linear feature, and center feature, is used to simulate the relevant features in the face by using some fixed features. Each classifier extracts a corresponding feature from the image. Therefore, they are similar to CNN, in which each convolution kernel extracts corresponding features. All types of features are shown in figure 2.

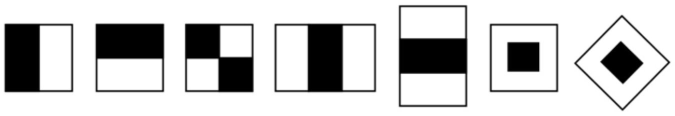

Figure 2. Three types of Haar features.

The feature template contains white and black rectangles, and defines the feature values of the template as the sum of white rectangle pixels minus the sum of black rectangle pixels. The value of one Haar feature reflects the change of the gray level of the image.

Rectangular features can be located at any position in the image and their size can be arbitrarily changed, so the rectangular eigenvalue is a function of the three factors: the type of the rectangular template, the position of the rectangle and the size of the rectangle. Therefore, the change of category, size and position makes a small detection window contain a lot of rectangular features. For example, the number of rectangular features in a detection window with a size of 24*24 pixels can reach 160,000. This leaves two problems to be solved:

- How to calculate so many features rapidly?
- Which rectangular features are most useful for classifiers?

To solve the first problem, we will introduce the concept of integral graph. In an image window, a large number of Haar rectangular feature areas can be extracted. If the rectangular feature areas are traversed every time when calculating Haar eigenvalues, a large number of repeated calculations will be executed and a significant amount of time will be wasted. Integral graph is a method to quickly calculate rectangular features. The main idea is to save the sum of the pixel values of the rectangular region formed between the initial pixel point of the image and each pixel point as an element, that is, to convert the original image into an integral graph (or summation graph).

When calculating the sum of the pixels of a certain rectangular region, it only needs to index the values of the four corner points in the rectangular region in the integral graph and perform ordinary addition and subtraction operations to obtain the Haar eigenvalues. In the whole process, the image only needs to be traversed once, and the time complexity of feature calculation is constant (O(1)), which can greatly improve the calculation efficiency. The formula for the elements in the integral graph is defined as follows:

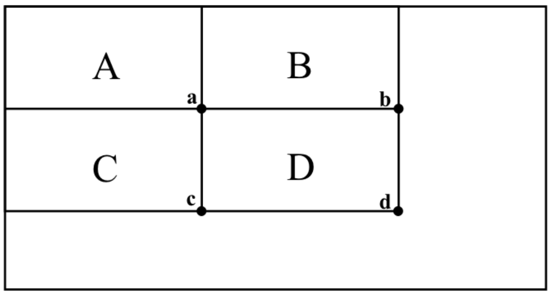

Figure 3. The sum of features.

Where is the pixel value of the image at point. In order to save time and reduce repeated calculation, the integral graph of image can be recursively deduced as follows:

In this way, pixel integration can be performed in any rectangular region. The gray integration of all pixels in any rectangle of an image can be quickly calculated by the integral graph of the image. As shown in figure 3, the value of the integral graph of point a is (where Sum is the gray integration):

Similarly, the value of integral graphs of point 2, 3 and 4 are:

The gray integration of all pixels in the rectangular region D can be obtained from the value of integral graph of the rectangular endpoint:

To deal with the second problem mentioned above, we will use the AdaBoost algorithm to build up the classification framework. AdaBoost is an abbreviation of Adaptive Boosting. The core idea of AdaBoost is to train the same weak classifier for different training sets, and then aggregate the weak classifiers obtained from different training sets to form a final strong classifier. Its self-adaptation lies in that the weight of the sample incorrectly classified by the previous basic classifier will increase, while the weight of the sample correctly classified will decrease, and it will be used to train the next basic classifier again. At the same time, in each iteration, a new weak classifier is added, and the final strong classifier is not determined until a predetermined small error rate or a predetermined maximum number of iterations is reached.

Given a training data set: where is the input training sample and represents positive sample and negative sample respectively.

The algorithm flow of AdaBoost is as follows:

- First, initialize the weight distribution of training data. Each training sample is initially assigned the same weight. Hence, the initial weight distribution is:
- For each (T is the number of weak classifiers):
  - ① A weak classifier with the lowest error rate is selected as the th basic classifier, and the weak classifier is calculated. The error of this weak classifier on the distribution is:
  - ② Calculate the weight of this weak classifier in the final classifier (the weight of weak classifier is represented by):
  - ③ Update the weight distribution of training samples:
    where is the normalized constant.
- Finally, the weak classifiers are combined according to the weight of weak classifiers:
  Through the sign function, a strong classifier is obtained:
  were.

In the final step, several strong classifiers are connected in series to form a cascade classifier. Each layer of the cascade classifier is composed of strong classifiers trained by AdaBoost algorithm. When the classifier of the first layer gets the correct result of the suspected face, it will trigger the second layer for classification. When the classifier of the second layer gets the correct result of the suspected face, it will trigger the third layer for classification, and so on. Finally, the image of the suspected face is confirmed as a face. On the contrary, when triggered to a certain layer and determined that the result of the classifier is not a face, it will immediately stop the detection of the image. The structure of the cascade is similar to a pyramid, as shown in figure 4.

In fact, not only the strong classifier is a pyramid structure, but also every weak classifier in the strong classifier is also a pyramid structure.

MTCNN. Convolutional neural network is a multi-layer neural network composed of overlapping convolutional layers used for feature extraction, pooling layers used for feature processing, and fully connected layers used for combining all local features into global features. In the convolutional layer of convolutional neural network, one neuron only connects with some adjacent neurons. A convolutional layer of CNN usually contains several feature maps, and the neurons of the same feature map share the same weight, which is the convolution kernel. In general, the convolution kernel is initialized in the form of random decimal matrix. In the training process of the network, the convolution kernel will learn to get reasonable weights. The immediate benefit of sharing weights (convolution kernels) is to reduce connections between layers of the network while reducing the risk of overfitting. There are usually two forms of pooling, mean-pooling and max-pooling. Pooling can be regarded as a special convolution process. Convolution and pooling greatly simplify model complexity and reduce the number of model parameters.

Multi-task convolutional neural network (MTCNN), which was proposed by Kaipeng Zhang et al. [4] in 2016, is used to realize video face detection in this experiment. In this model, three cascaded networks are used, and the idea of candidate box plus classifier is used to detect face quickly and efficiently. The three cascading networks are: proposal network (P-NET) for fast candidate window generation, refine network (R-NET) for high precision candidate window filtering selection, and output network (O-NET) for final boundary box and face key point generation. The face detection process of MTCNN is shown in figure 5.

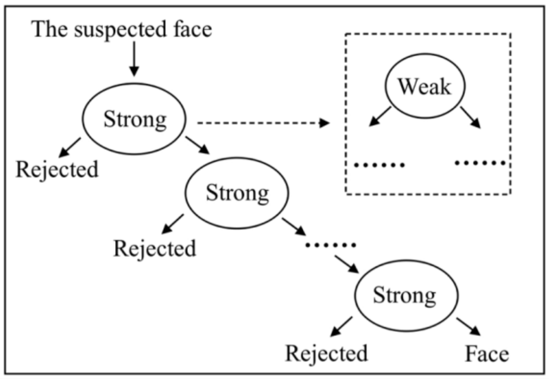

Figure 4. The structure of the cascade classifier.

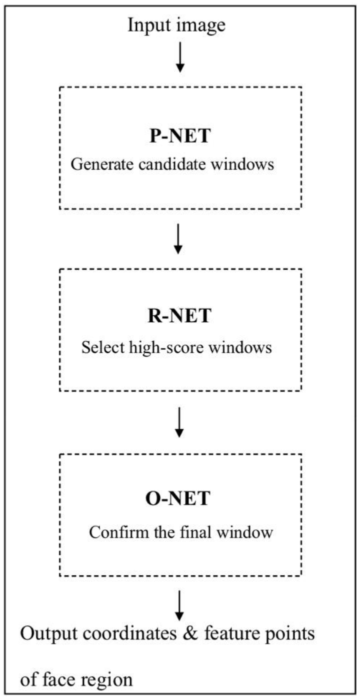
Input image
P-NET
Generate candidate windows
R-NET
Select high-score windows
O-NET
Confirm the final window
Output coordinates & feature points of face region

Figure 5. Face detection process of MTCNN.

P-NET is a face region area suggested network. The network will input the image into three convolution layers. Then, it will determine whether the region of this image is face or not through a face classifier. At the same time, using the border regression and a facial point locator to generate face region's initial proposals. Finally, this part will output a lot of pieces of possible face region, which are fed into R-NET for further processing.

After the image passes through P-Net, there are many prediction windows left, which represent a significant amount of possible face regions. We will feed all the prediction windows into R-Net. In this network, the input will be refined and most of the wrong proposals will be eliminated. Then, it will use the border regression and face key locator again, and finally output more credible face regions for O-NET to use. Compared with p-NET, which yields 1x1x32 features through full convolution, R-NET uses a fully connection layer of 128 after the last convolutional layer, which retains more image features and has better accuracy performance than P-NET.

O-NET is a more sophisticated convolutional network and has more input features. At the end of the network, there is also a large fully connected layer of 256 retaining more image features. The final output are the upper left and lower right coordinates of the face region and the five feature points of the face region, which are also the final output of the model.

As many convolutional neural network models dealing with image problems, MTCNN also uses some traditional methods like image pyramid, border regression and non-maximum suppression (NMS). In this experiment, NMS will replaced by Soft-NMS, which is of great benefit to the final result. The process of Soft-NMS can be divided into the following three steps:

- Sort candidate boxes in descending order of confidence;
- Calculate the values of Intersection-over-Union (IOU) between the first candidate box and the remaining boxes, and reduce the score of the box according to the values of IOU;
- Finally, delete the boxes uniformly according to the score threshold and only preserve the highest score box;

3.3. Model construction

AdaBoost. In this experiment, 2600 positive samples and 5500 negative samples will be used for training process and there are four Haar classifiers to be trained respectively. We will set following key parameters that are related to the performance of each classifier. Number of stages represents the number of classes in the course of training and the aim of each class is to obtain a strong classifier. Number of splits is identical to the number of spit child nodes or features in a weak classifier. The value of min hit rate determines the positive detection rate of positive samples of each stage. Max false alarm rate means maximum error detection for each classifier. The value of weight trimming coefficient will make the training process focus more on the samples that cannot be correctly classified, therefore, improve the training efficiency. Besides the parameters mentioned above, we will set the initialization weights of all samples to be equal. Last but not the least, the types of Haar features used in the training include vertical features and 45-degree rotation features.

Table1 shows the settings for the parameters of Haar classifier1.

Table1. Parameters setting of haar classifier1.

Parameters	Value
Number of stages	18
Number of splits	2
Min hit rate	0.995
Max false alarm rate	0.4
Weight trimming	0.9
Equal weights	TRUE
Types of Haar features	ALL

Table 2. Parameters setting of haar classifier2.

Parameters	**Value**
Number of stages	18
Number of splits	2
Min hit rate	0.995
Max false alarm rate	0.5
Weight trimming	0.95
Equal weights	TRUE
Types of Haar features	ALL

Table 3 shows the settings for the parameters of Haar classifier3.

Table 3. Parameters setting of haar classifier3.

Parameters	**Value**
Number of stages	20
Number of splits	2
Min hit rate	0.995
Max false alarm rate	0.5
Weight trimming	0.95
Equal weights	TRUE
Types of Haar features	ALL

Table 4 shows the settings for the parameters of Haar classifier4.

Table 4. Parameters setting of haar classifier4

Parameters	**Value**
Number of stages	15
Number of splits	2
Min hit rate	0.995
Max false alarm rate	0.5
Weight trimming	0.9
Equal weights	TRUE
Types of Haar features	ALL

MTCNN There are two different MTCNNs to be trained. We add some batch normalization layers in MTCNN1, which were not included in MTCNN2. The activation function of R-NET and O-NET in MTCNN1 is sigmoid function; while the activation function in MTCNN2 is softmax function.

Table 5. Network structure of mtcnn1.

Parameters	P-NET	R-NET	O-NET
Structure	3 convolutional layers, 1 pooling layers	3 convolutional layers, 1 pooling layers, 1 fully connected layer	4 convolutional layers, 2 pooling layers, 1 fully connected layer
Number of kernels of convolutional layers	10, 16, 32	28, 48, 64	32, 64, 128, 128
Convolutional layer kernel size	(3, 3)	(3, 3), (2, 2)	(3, 3)
Convolutional layer stride	(1, 1)	(1, 1)	(1, 1)
Pooling layer activation function	PReLU	PReLU	PReLU
Pooling layer kernel size	(2, 2)	(3, 3)	(3, 3), (2, 2)
Pooling layer stride	(2, 2)	(2, 2)	(2, 2)
Fully connected layer size	----	128	256
Normalization	BatchNorm2d	BatchNorm2d	BatchNorm2d
Activation function	----	Sigmoid	Sigmoid

Table 6 shows the settings for the network structure of MTCNN2.

Table 6. Network structure of mtcnn2.

Parameters	P-NET	R-NET	O-NET
Structure	3 convolutional layers, 1 pooling layers	3 convolutional layers, 1 pooling layers, 1 fully connected layer	4 convolutional layers, 2 pooling layers, 1 fully connected layer
Number of kernels of convolutional layers	10, 16, 32	28, 48, 64	32, 64, 128, 128
Convolutional layer kernel size	(3, 3)	(3, 3), (2, 2)	(3, 3)
Convolutional layer stride	(1, 1)	(1, 1)	(1, 1)
Pooling layer activation function	PReLU	PReLU	PReLU
Pooling layer kernel size	(2, 2)	(3, 3)	(3, 3), (2, 2)
Pooling layer stride	(2, 2)	(2, 2)	(2, 2)
Fully connected layer size	----	128	256
Normalization	----	----	----
Activation function	----	Softmax	Softmax

4. Experimental Discussion

4.1. Dataset

The training dataset we used in our experiment is WIDER Face Dataset, which is published by the Chinese University of Hong Kong in 2015. It contains 32,203 images and 393,703 faces, with great changes in face size, posture, occlusion, expression, makeup and illumination. Since its release, it has been widely used in the evaluation of convolutional neural networks with more powerful performance than traditional methods.

4.2. Experimental Result

In this part, the best Haar classifier and MTCNN will be selected at first. Then, the face detection performance of AdaBoost and MTCNN will be compared and analyzed. To be specific, two experiments will be conducted respectively. In the first experiment, we attempt to detect two human faces from an interview video clip to test accuracy and detection speed of two models, since in this video, the state of human faces is relatively fixed. In the second experiment, a TV drama clip, which contains more human faces and the state of faces is more random, is used to measure some important performance indicators of two models.

The detection results of four Haar classifiers are shown in Table 7.

Table 7. Accuracy of the classifiers.

Video	Classifier1	Classifier2	Classifier3	Classifier4
Interview Clip	77.01%	80.02%	81.92%	13.17%
Drama Clip	59.14%	62.15%	64.99%	6.78%

As shown in Table 7, classifier3 has the highest accuracy on both interview clip and drama clip, which demonstrates that classifier with higher number of stages and larger value of max false alarm rate and weight trimming coefficient is conducive to implement a better face detection performance. In this case, classifier3 will be selected as the representation of the best face detection performance of AdaBoost. The detection results of two MTCNNs are shown in Table 8.

Table 8. Accuracy of the mtcnns.

Video	MTCNN1	MTCNN2
Interview Clip	98.10%	96.28%
Drama Clip	90.56%	88.62%

By appending batch normalization layers to the network and replacing softmax function with sigmoid, MTCNN1 owns better accuracy than MTCNN2. Therefore, we're going to choose MTCNN1 as the representation of MTCNN.

To compare the face detection performance of classifier3 and MTCNN1, the accuracy and frame per second (FPS) of two methods have been calculated. The detection results of the interview clip are shown in Table 9.

Table 9. the detection results of the interview clip

Index	MTCNN1	CLASSIFIER3
Accuracy	98.10%	81.92%
FPS	9.92	10.43

The detection results of the drama clip are shown in Table 10.

Table 10. the detection results of the drama clip

Index	MTCNN1	CLASSIFIER3
Accuracy	90.56%	64.99%
FPS	5.09	6.67

According to TABLEⅨ, we can come to a conclusion that the face detection accuracy of MTCNN1 is much better than classifier3 because MTCNN can find the most accurate bounding box from hundreds of proposal boxes after processing by three networks. Whereas, AdaBoost method will accept all proposal faces and not eliminate the wrong bounding boxes that we can see from figure 6.
By contrast, AdaBoost method has a better FPS than MTCNN since it doesn't have such a complicated structure as MTCNN. Therefore, AdaBoost classifier doesn't need to consume a significant amount of time to find faces in videos.

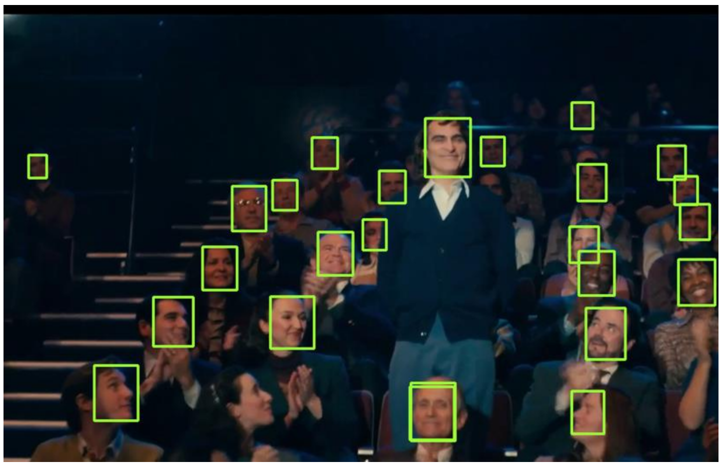

Figure 6. A frame of face detection result of AdaBoost.

5. Conclusion

To conclude, this paper intended to find a best approach to implement video-based face detection by comparing the performance of conventional methods and deep learning. We selected the most representative method in both conventional methods and deep learning, namely AdaBoost and MTCNN. Final discussions were made of the comparisons between the best AdaBoost classifier and the best MTCNN structure in detecting faces from video resources. From the discussion, the conclusion can be reached that the face detection accuracy of MTCNN is more adequate than AdaBoost's; while AdaBoost is faster than MTCNN due to its simpler framework. Our experimental results could provide promising insights into the research of face detection under the context of resource constraints.

This research, however, is subject to several limitations. We only implement video-based face detection with low computing power scenario. To achieve higher precision and efficiency, this experiment can be conducted in high-performance devices. Moreover, we only compared a portion of methods and trained a limited number of classifiers and networks in this research, which could lead to further discussions of various of methods. Future research should consider the potential applications of video-based face detection such as security system and image search. We highly recommend researchers combine convolutional methods and deep learning to create a method with prominent accuracy as well as fast speed.

References

[1] Viola, P., & Jones, M. (2001, December). Rapid object detection using a boosted cascade of

simple features. In Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001 (Vol. 1, pp. I-I). Ieee.
[2] Gottumukkal, R., & Asari, V. K. (2004). An improved face recognition technique based on modular PCA approach. Pattern Recognition Letters, 25(4), 429-436.
[3] Belhumeur, P. N., Hespanha, J. P., & Kriegman, D. J. (1997). Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. IEEE Transactions on pattern analysis and machine intelligence, 19(7), 711-720.
[4] Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. IEEE signal processing letters, 23(10), 1499-1503.
[5] Sun, Y., Liang, D., Wang, X., & Tang, X. (2015). Deepid3: Face recognition with very deep neural networks. arXiv preprint arXiv:1502.00873.
[6] Yang, J., Ren, P., Zhang, D., Chen, D., Wen, F., Li, H., & Hua, G. (2017). Neural aggregation network for video face recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4362-4371).
[7] Turk, M., & Pentland, A. (1991). Eigenfaces for recognition. Journal of cognitive neuroscience, 3(1), 71-86.
[8] Lienhart, R., & Maydt, J. (2002, September). An extended set of haar-like features for rapid object detection. In Proceedings. international conference on image processing (Vol. 1, pp. I-I). IEEE.
[9] Kim, K. I., Jung, K., & Kim, H. J. (2002). Face recognition using kernel principal component analysis. IEEE signal processing letters, 9(2), 40-42.
[10] Hsu, R. L., Abdel-Mottaleb, M., & Jain, A. K. (2002). Face detection in color images. IEEE transactions on pattern analysis and machine intelligence, 24(5), 696-706.
[11] Sun, Y., Liang, D., Wang, X., & Tang, X. (2015). Deepid3: Face recognition with very deep neural networks. arXiv preprint arXiv:1502.00873.
[12] Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3431-3440).
[13] Wen, Y., Zhang, K., Li, Z., & Qiao, Y. (2016, October). A discriminative feature learning approach for deep face recognition. In European conference on computer vision (pp. 499-515). Springer, Cham.
[14] Garg, D., Goel, P., Pandya, S., Ganatra, A., & Kotecha, K. (2018, November). A deep learning approach for face detection using YOLO. In 2018 IEEE Punecon (pp. 1-4). IEEE.
[15] Deng, J., Zhou, Y., & Zafeiriou, S. (2017). Marginal loss for deep face recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops (pp. 60-68).
[16] Wang, H., Wang, Y., Zhou, Z., Ji, X., Gong, D., Zhou, J., ... & Liu, W. (2018). Cosface: Large margin cosine loss for deep face recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 5265-5274).
[17] Chavda, A., Dsouza, J., Badgujar, S., & Damani, A. (2021, April). Multi-stage cnn architecture for face mask detection. In 2021 6th International Conference for Convergence in Technology (I2CT) (pp. 1-8). IEEE.
[18] Rahmad, C., Asmara, R. A., Putra, D. R. H., Dharma, I., Darmono, H., & Muhiqqin, I. (2020). Comparison of Viola-Jones Haar Cascade classifier and histogram of oriented gradients (HOG) for face detection. In IOP conference series: materials science and engineering (Vol. 732, No. 1, p. 012038). IOP Publishing.
[19] Etemad, K., & Chellappa, R. (1997). Discriminant analysis for recognition of human face images. Josa a, 14(8), 1724-1733.
[20] Korshunov, P., & Marcel, S. (2018). Deepfakes: a new threat to face recognition? assessment and detection. arXiv preprint arXiv:1812.08685.