Capturing the Persistence of Facial Expression Features for Deepfake Video Detection

Yiru Zhao[1], Wanfeng Ge[1], Wenxin Li[1], Run Wang[1], Lei Zhao[1,2(✉)], and Jiang Ming[3]

[1] School of Cyber Science and Engineering, Wuhan University, Wuhan, China
leizhao@whu.edu.cn
[2] Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, Wuhan, China
[3] University of Texas at Arlington, Arlington, USA

Abstract. The security of the Deepfake video has become the focus of social concern. This kind of fake video not only infringes copyright and privacy but also poses potential risks to politics, journalism, social trust, and other aspects. Unfortunately, fighting against Deepfake video is still in its early stage and practical solutions are required. Currently, biological signal based and learning-based are two major ways in detecting Deepfake video. We explore that facial expression between two adjacent frames appears significant differences in generative adversarial network (GAN)-synthesized fake video, while in a real video the facial expression looks naturally and transforms in a smooth way across frames. In this paper, we employ optical flow to capture the obvious differences of facial expressions between adjacent frames in a video and incorporate the temporal characteristics of consecutive frames into a convolutional neural network (CNN) model to distinguish the Deepfake video. In our experiments, we evaluate the effectiveness of our approach on a publicly fake video dataset, FaceForensics++. Experimental results show that our proposed approach achieves an accuracy higher than 98.1% and the AUC score reaches more than 0.9981.

Keywords: Deepfake Detection · Temporal features · Spatial features · Optical flow

1 Introduction

With the remarkable progress of GANs in image synthesis, we cannot believe our eyes in the AI era. Fake videos can be easily generated with tools such as FaceSwap [2], Deepfacelab [1], FaceApp [5] by touching a few keys on devices. These tools leverage the power of GANs in image synthesis and provide quite interesting functionalities to users, for instance, users can swap one's face to others and create a nearly realistic fake video. As shown in Fig. 1, A physically

J. Zhou et al. (Eds.): ICICS 2019, LNCS 11999, pp. 630–645, 2020.
https://doi.org/10.1007/978-3-030-41579-2_37

discordant person can do a great dance by changing faces with someone from dance video. However, this also brings some security concerns and privacy issues to users when the image synthesis techniques are abused.

Everyone will be the victims including celebrities and politicians. A celebrity's face could be swapped to a naked body and an illusory official statement can be announced by a politician in an AI-synthesized fake video. Thus, it is crucial to call for effective ways for spotting these AI-synthesized fake videos which are also known as Deepfake [3]. There are three common types of Deepfake, namely face swapping, lip-sync, and puppet-master [11]. In our work, we mainly focus on face swapping which is widely used in free tools (e.g. ZAO [9]) and could easily incur misinformation dissemination in social networks.

Detecting manipulated media content is a longstanding research focus. However, traditional techniques face many challenges for detecting the Deepfake video. These techniques extract pix-level or color-level statistical features [15,18,19], which can be easily suffered by compression, resizing, etc. In generating videos, compression and resizing is common operations, thus traditional image forensics techniques failed in fake video detection.

Fig. 1. Deepfake video

Existing work on detecting Deepfake videos can be summarized into two categories, biological signal based and learning-based. Agarwal et al. [11] observed that individual exhibits distinct patterns of facial and head movement when speaking, while Deepfake video tends to disrupt these particular patterns. Yang et al. [24] investigated that current neural network synthesized faces appear mismatched facial landmarks. Li et al. [16] distinguished fake videos by capturing the frequency of eye blinking. Some researchers [17] noticed that the synthesized faces in Deepfake video always in a fixed size. Afshar et al. [10] proposed detecting Deepfake video with the basic insight that some frames in Deepfake video exhibit large blurred areas or a double facial contour. These work pay attention to extract obvious biological signal features that are unrealistic in real

videos. Some work leverage the power of neural networks in feature representation. The basic insight of these work is that the inconsistencies are introduced across frames in synthesizing fake videos, particularly temporal discrepancies across frames [14,21].

In our work, we explore another biological signal features which can be applied in distinguishing Deepfake videos. Facial expression in adjacent frames should be transformed naturally and has strong correlations, while it is hard to be overcome in synthesizing fake videos as facial expression patterns are always inadequate for individuals. In the meantime, we find optical flow can capture the subtle facial expression variations in consecutive frames effectively.

In this work, we employ optical flow to characterize the temporal changes of facial expressions, that is, the temporal features proposed in this paper. Then we use the convolutional neural network to extract the spatial features of the original images. According to the Spatial-temporal characteristics consistency of subtle expressions, we use the convolutional neural network to extract the features at a deep level to detect the Deepfake video. Our contributions are:

1. In this paper, we observed that facial expression between two adjacent frames in the Deepfake videos appears significant differences, which can be used to better detect Deepfake videos.
2. We employ optical flow to capture the differences of facial expression between two adjacent frames based on our observation. Then we use the optical flow graphs to characterize the temporal features of the videos.
3. We evaluate the effectiveness of our approach on FaceForensics++, a publicly fake video dataset. The experimental results show that our approach achieves an accuracy higher than 98.1% and the AUC score reaches more than 0.9981.

The rest of this paper is organized as follows. Section 2 introduces the background. Section 3 describes our approach in detail. Section 4 presents the implementation and evaluation of our method, Sect. 5.

2 Background

In this section, we firstly introduce the generation of Deepfake videos. Then we further analyze its vulnerabilities.

2.1 Deepfake Video Generation

One way to generate Deepfake videos is to use an encoder-decoder model based on AI, which consists of two processes, the training process as well as the generation process. In the training process, two neural networks are trained, each of which is composed of an encoder network and a decoder network. For input pictures of two different faces A and B, the encoder network first compresses the face data in each picture into a low-dimensional vector and then uses the decoder network to decode the low-dimensional vector obtained in the previous step to generate the decoded picture. Then the network is optimized by minimizing the

difference between the decoded image and the input image. The encoder network of the two images must remain consistent during the training phase, in order to extract the consistency of the features in these two pictures. In the generation process, we use the trained decoder B to decode the encoded low-dimensional vector of A, so that we get the face-changing image of A.

In general, the overall process of the training process is to extract the features of two different faces (original face and target face) with the encoder of the same parameters and then recover the target face image with the decoder trained from the target face. The encoder is a convolutional neural network that extracts features such as facial features, expressions and so on from the image input. The decoder recovers the original image from the extracted features according to the parameters of the encoder [14]. The encoder contains a general method for extracting the typical features of human faces, and different decoders can recover different faces from the extracted features.

Another more common way to generate Deepfake videos is based on Generative Adversarial Network (GAN)[13]. As is shown in Fig. 2. A Generative Adversarial Network consists of two parts: the generator and the discriminator. The generator takes face samples that need to be swapped as input to generate false video. The discriminator is simply a classifier trained with supervised learning techniques to check if the video is real or fake. The generator and the discriminator are rivals of each other. All the parameters are trained until convergence.

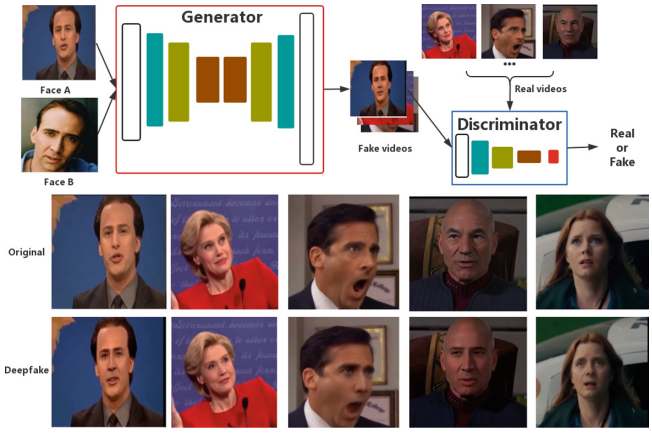

Fig. 2. The procedure of face swapping with GANs

2.2 Vulnerability of Deepfake Videos

Through the above analysis, we can see that when generating Deepfake videos, the face is replaced frame by frame, and then the complete video is synthesized. As is well known, video differs from images in that video is composed of

many consecutive frames. For real video, successive frames represent continuous changes in the target object over time, so they have strong consistency. In contrast, a Deepfake video will inevitably break the consistency between adjacent frames by processing each frame and then combining them. In other words, the facial expression of the target area in the Deepfake video will have a certain degree of distortion and anomalies. Therefore, extracting the characteristics of the variation between adjacent frames helps us to get a more comprehensive Deepfake video feature.

3 Our Approach

In order to extract the characteristics of the variation between adjacent frames, we need to get the change of pixels within the image sequence in the time domain and the correlation between adjacent frames to find the correspondence between the previous frame and the current frame. Calculating optical flow is such a method. In this paper, we characterize the temporal features by calculating the optical flow fields of the facial images in two consecutive frames.

However, we want to extract deeper features. We use a convolutional neural network (CNN) to obtain the spatial features of the frame and combine them with the temporal features. Based on the consistency of space-temporal characteristics, our CNN-based deep learning model can acquire deeper features.

We build a model that consists of four parts: data preprocessing, spatial feature extraction, temporal feature extraction, and deep learning model.

At first, we processed video data set into pictures frame by frame and intercepted the faces in the pictures. Then, we use CNN to extract the spatial features of human faces. At the same time, we calculate the optical flow between two continuous pictures of human faces to obtain the optical flow diagrams. After that, the powerful self-extraction ability of CNN is used to extract the temporal features of optical flow diagrams. According to the consistency of space-temporal characteristics, we combine the temporal and spatial characteristics so that CNN can acquire deeper features. Finally, we train the deep learning model to implement Deepfake video detection. The entire process is shown in Fig. 3.

3.1 Data Preprocessing

In general, data preprocessing is a common requirement for many learning algorithms and models. Data preprocessing describes any type of processing performed on raw data to prepare it for another processing procedure. In this paper, the data preprocessing stage mainly consists of three steps: extracting image frames from the video, extracting the face region from the image, and sequentially storing the images.

First, we selected training data sets and verification data sets from real videos and Deepfake videos. Then we separate frames from the original video data sets frame by frame. We do video segmentation and then extract the face region of every frame. At last, the processed face images are stored in the order of the video frame sequence number.

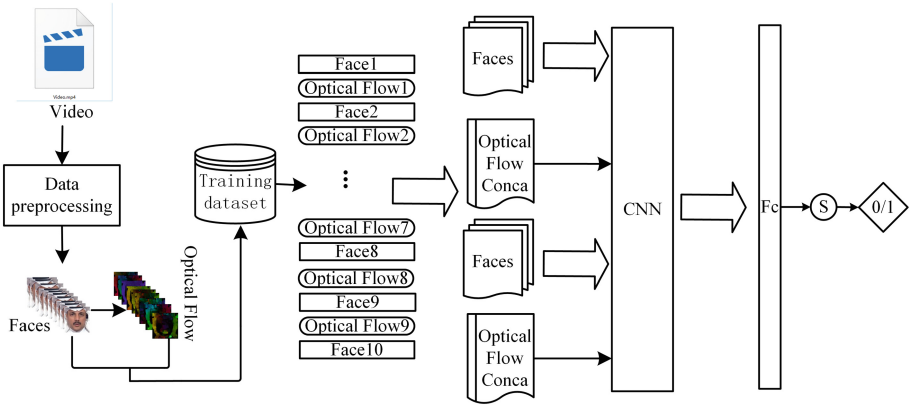

Fig. 3. Overall framework of the model

3.2 Temporal Feature Extraction Based on Optical Flow

We know that when generating Deepfake videos, the face is replaced frame by frame, and then the complete video is synthesized. Such a process neglects the consistency of slight changes in facial expressions between frames in the original video, which is likely to bring about disharmonious cases. In most cases, the emotions of people do not change suddenly, in other words, the expressions of human beings change continuously in normal videos. That is to say, the emotions reflected by their facial expressions between adjacent frames should be consistent. However, since each frame is independent, videos generated by the Deepfake technique do not always take into account this kind of situation, leading to inconsistent facial expressions between adjacent frames. It may be hard to tell with the naked eye, but it is not difficult for the machine to recognize.

Therefore, we need to find the correspondence between the previous frame and the current frame. Calculating optical flow is a method of calculating the change of pixels within image sequence in the time domain and the correlation between adjacent frames, thereby calculating the motion of the object between adjacent frames. Optical flow is the instantaneous velocity of a moving pixel within the image. If the time interval is small enough, the velocity can be expressed by displacement. In simple terms, assuming that the angle of observation is constant, the optical flow represents the movement of a point from the first frame to the second frame. For example, for two adjacent frames in a video, or frames extracted from a video at a small-time interval, the instantaneous velocity of pixel movement can be expressed by displacement. We usually regard it as a two-dimensional vector $u = (u,v)$ describing the instantaneous velocity of the pixel, which is also called the optical flow vector.

Optical flow can be divided into two types, dense optical flow, and sparse optical flow [22]. The dense optical flow performs point-by-point matching on the image to calculate the offset of all points on the image, while the sparse optical flow only needs to specify a set of points with obvious characteristics for

tracking. In contrast, due to the denser optical flow vector, dense optical flow is significantly better than the sparse optical flow at the registration effect, but at the same time, the calculation amount is larger because the offset of each point is calculated. In our work, we select dense optical flow because the amount of image pixels that need to be calculated is limited and the demand for accuracy is high.

OpenCV [22] provides an algorithm for calculating dense optical flows: the Farneback dense optical flow algorithm. The principle is briefly described below [16]. The working principle of the optical flow method is based on the following assumptions [23]:

1. The brightness of the target pixel does not change between two consecutive frames of images.
2. There is a similar motion between adjacent pixels.

We define $X = [x, y]$ as one pixel position in the image and t as time. We note the brightness of $X = [x, y]$ at time t as $I(x, y, t)$.

For two consecutive frames in the image, since the brightness of the target pixel does not change, the brightness of the same pixel in these two frames does not change, which is expressed as:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t) \tag{1}$$

We use the first order Taylor expansion at $I(x, y, t)$ of formula (1), and get formula (2):

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt + \xi \tag{2}$$

In formula (2), ξ is the second order infinitesimal in Taylor's expansion, which can be ignored. Substituting formula 2 into formula 1, and divide both sides by dt,we get:

$$\frac{I(x, y, t)}{dt} = \frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0 \tag{3}$$

It is obvious that $\frac{dx}{dt}$ and $\frac{dy}{dt}$ represent the motion vectors of the tracked pixel points in the x-axis direction and the y-axis direction. Let $u = \frac{dx}{dt}$, $v = \frac{dy}{dt}$, then (u, v) is the desired optical flow. This feature is very suitable for the temporal feature extraction in this paper.

3.3 Spatial Feature Extraction Based on CNN

According to the consistency of space-temporal features, we want to extract the spatial features of adjacent frames. Considering that the convolutional neural network (CNN) has a strong self-learning ability, we hope to build our own network to learn and capture the characteristics such as low-level features, contour, grayscale, texture, and other features. These features can well reflect some subtle abnormalities of the human face in Deepfake videos, such as edge stiffness, jitter, distortion, color and so on.

3.4 Deep Learning Model

We obtain the optical flow diagram by calculating the optical flow and extract the spatial features between adjacent frames through the convolutional neural network. We combine the two features and use the convolutional neural network to further extract the deeper features. Considering the excellent performance of CNN in image classification, we decided to use this type of network to implement our detection. The general structure is shown in Fig. 4. Then we trained and verified the model on the data set, adjusted and modified the network structure with poor verification performance, and finally got the model that performed well on both the training set and the verification set.

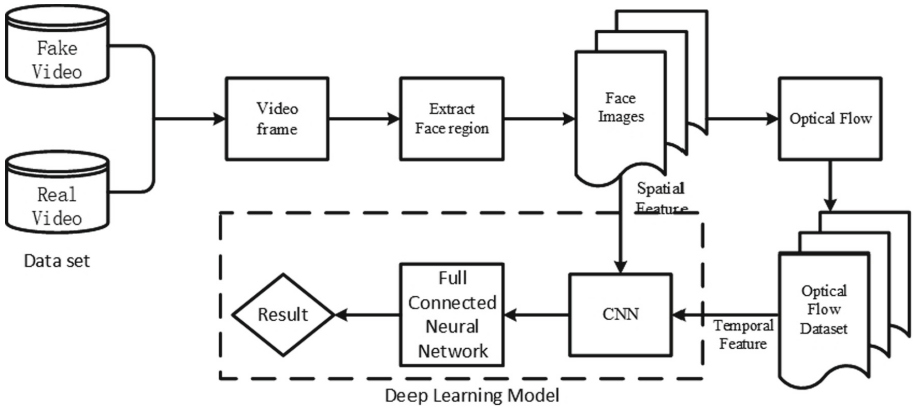

Fig. 4. The Structure of our deep learning model

The neural network has one input layer, four convolutional layers, four batch normalization layers, four pooling layers, one flatten layer, two dropout layers, and three dense layers. The specific information of each part is as follows:

1. Input layer: This layer accepts ten 256*256*3 original frames and two 256*256*8 optical flow diagrams as input at a time.
2. The parameters of convolutional layers and pooling layers, as well as the processing results of each layer, are shown in Table 1.
3. Batch normalization layer: This layer normalizes the data of each batch to ensure the rapid convergence of the model.
4. flatten layer: This layer is mainly used to "flatten" the data input from the convolutional layer, that is, to convert multidimensional data to one-dimensional input.
5. Dropout layer 1: Whenever parameters are updated each the time during training, the input neurons are randomly disconnected with a probability of 0.5.

6. Dense layer 1: Fully connected layer with 16 units. It calculates the dot product between the input vector and the weight vector to obtains 16 outputs, and inputs the result to the Leaky ReLU activation function for nonlinear processing.
7. Dropout layer 2: The input neurons are randomly disconnected with a probability of 0.5 each time the parameters are updated during the training.
8. Dense layer 2: Fully connected layer with 1 unit. It calculates the dot product between the input vector and the weight vector to get 1 output.
9. Dense layer 3: Fully connected layer with 1 unit. This layer accepts the processing result of 10 video frames and 2 optical flow pictures as input, and then output the final classification result.

Table 1. Parameters and results of convolutional layers and pooling layers

Layers	Filter size	Step length	Number of convolutional kernels	Activation function	Feature map
Convolutional layer C1	**3*3**	**1**	**8**	**Relu**	**256*256*8**
Pooling layer M1	**Pooling window size: 2*2**				**128*128*8**
Convolutional layer C2	**5*5**	**1**	**8**	**Relu**	**128*128*8**
Pooling layer M2	**Pooling window size: 2*2**				**64*64*8**
Convolutional layer C3	**5*5**	**1**	**16**	**Relu**	**64*64*16**
Pooling layer M3	**Pooling window size: 2*2**				**32*32*16**
Convolutional layer C4	**5*5**	**1**	**16**	**Relu**	**32*32*16**
Pooling layer M4	**Pooling window size: 4*4**				**8*8*16**

Each layer of CNN plays its role. The convolutional layer extracts the features of the image, and its weight sharing and partial connection structure reduces the number of parameters that need to be optimized. The batch normalization layer speeds up the convergence of the model. The pooling layer compresses the data, reducing memory consumption. The Dropout layer helps avoid over-fitting. After the network completes the training, the obtained CNN model and its parameters can be used for verification and testing. We modify and adjust the structure of the model several times according to the verification performance. Finally, we choose the neural network with the highest accuracy. Figure 5 show the feature map of the input optical flow map after every convolutional layer.

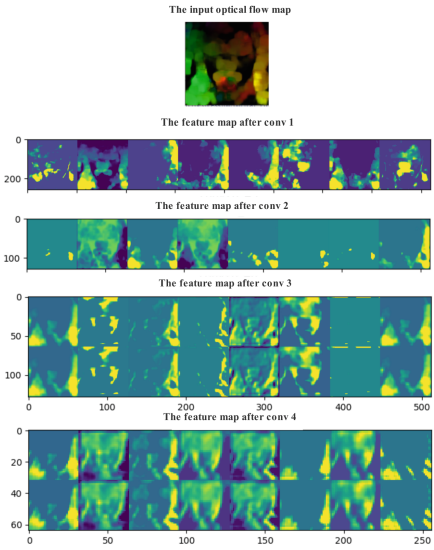

Fig. 5. The intermediate results

4 Implementation and Evaluation

In this section, we briefly introduce our dataset and experimental process. Finally, we evaluate our model's performance and analyze the experiment results.

4.1 Experimental Setup

To evaluate our method, we use the FaceForensics++ Datasets published on github [1], which is an open dataset containing 1000 Deepfake standard-definition videos and 1000 real standard-definition videos collected from several social media platforms. Since the dataset is open source and diverse, evaluation based on it ensures that our method is effective and robust.

The experiments in this paper were performed on a Windows 10 desktop computer with an Intel(R) Core(TM) i7-8700CPU@3.20 GHz, a memory of 16 GB, and a GPU for the Nvidia GTX1080ti. The deep learning model was built using Keras 2.2.4 [7] and used Tensorflow 1.8.0 [8] as the backend engine.

4.2 Experiment Process

Our experiment contains several steps.

Firstly, we randomly select 850 videos from 1000 real videos and 850 from 1000 Deepfake videos as the training dataset. As for the testing dataset, we randomly select 100 videos from each category respectively.

Secondly, we separate frames from the original video data set frame by frame. In this paper, we use FFmpeg [6] to separate frames from the original video

data set frame by frame. FFmpeg is the leading multimedia framework, able to decode, encode, transcode, mux, demux, stream, filter and play pretty much anything that humans and machines have created. We use the Python language to call the FFmpeg program for video framing.

Thirdly, we do video segmentation and then extract the face region of every frame. We use Dlib [4], which is a modern C++ toolkit containing machine learning algorithms and tools, including the HOG-SVM algorithm for face detection and multiple detection algorithms based on CNN. Through experiments, we found that the CNN-based detection algorithms are better, so we choose the latter to get face position information. Then we use OpenCV [22] to crop the obtained face position information and save it as a 256*256 3-channel .png format image. The number of images in the dataset is shown in Table 2.

Table 2. Experimental data

Class	Training	Verification
Deepfake frames	**17116**	1924
Real frames	**17191**	1962
Total frames	**34307**	3886

Then, we calculate the optical flow fields of the facial images in two consecutive frames. We use OpenCV, which provides a function called calcOpticalFlowFarneback to implement the Farneback dense optical flow algorithm. The image of the human facial area extracted from the video is an RGB image, but the Farneback dense optical flow algorithm can only calculate a grayscale image. So it is necessary to first convert the RGB image into a grayscale image. Besides, we need to take care that the input of this function is an 8-bit single-channel picture of 256*256 pixels of two consecutive frames, while the output is a CV_32FC2 format optical flow image of the same size as the input picture which is a two-channel image. We calculate 4 optical flow maps between 5 frames, and we define the combination of the above-mentioned pictures as a group. We use two continuous groups as an input as is shown in Fig. 6, and then mark it with 1 for fake video or 0 for real video.

At last, we incorporate the temporal features with spatial features of consecutive frames into a convolutional neural network (CNN) model to distinguish the Deepfake video.

4.3 Experimental Result

In the experiment, we set the iteration of the model to 80 times, and the loss function of the deep learning model to MSE (mean square error). The calculation method is as follows:

$$Loss = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y_i} - y_i)^2 \tag{4}$$

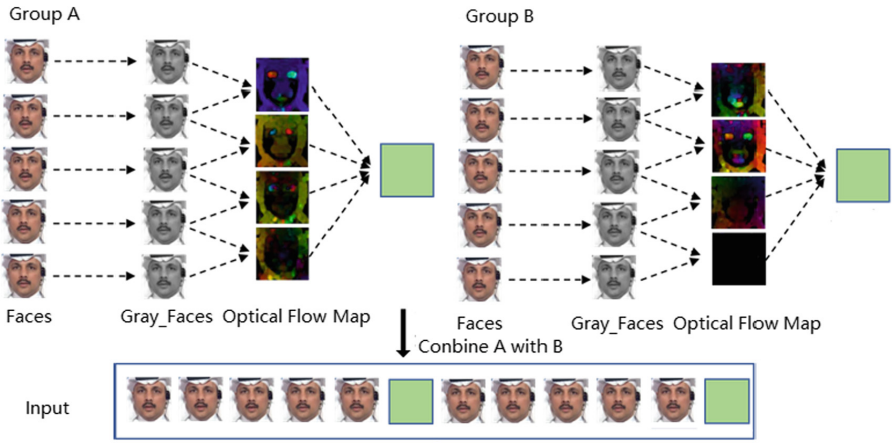

Fig. 6. The generation of input sequence

$\hat{y}_i$ is the predicted value of the model, and y_i is the label of the sample. Loss function can well represent the fitting degree between the predicted results of the model and the real label, and the smaller the value, the better. As shown in Fig. 7, With the increase of training times, the loss function value of the model gradually decreases.

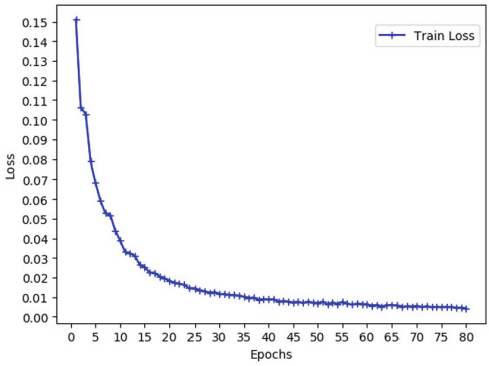

Fig. 7. The changing curve of loss

Accuracy is an important evaluation index for the classification model. The accuracy is defined in a standard way as formula 5. TP stands for True Positive, the number of Deepfake images correctly classified. TN is True Negative, referring to the number of True images correctly classified. FP is False Positive and refers

to the number of Deepfake images misclassified. FN is False Negative and is the number of true images wrongly classified. The higher the accuracy, the better the accuracy of the model.

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{5}$$

Besides, the ROC curve is also selected as the evaluation standard, which can well describe the generalization performance of the model. The ROC curve plots the TPR (True Positive Rate) against the FPR (False Positive Rate) for every test case. We then calculate the AUC (Area Under the ROC curve) to characterize how well the model performs. The closer is AUC to 1, the better the model performance. The formula of TPR, FPR, and AUC is as follows:

$$TPR = \frac{TP}{TP + FN} \tag{6}$$

$$FPR = \frac{FP}{FP + TN} \tag{7}$$

$$AUC = \frac{1}{2} \sum_{i=1}^{m-1} (FPR_{i+1} - FPR_i)(TPR_i + TPR_{i+1}) \tag{8}$$

Figure 8 shows the system accuracy on both training sets and test set as the epoch number grows. It indicates that our method could achieve accuracy greater than 98% on the training set and 96% on the testing set after 40 iterations. Figure 9 shows the ROC curve. In paper [20], the author compares many models using the FaceForensics++ data set. We chose the two models MesoNet [10] and XceptionNet [12] to compare with our model, because only MesoNet and XceptionNet are the detection models for Deepfake. Then we reproduce these two models and experiment with our models on the same training and validation sets. Since some parameters in the experiment are different from those in Paper [20], such as data set division and CPU parameters. In order to compare the effects of models in various aspects, AUC is also selected as the comparison standard. our experimental results are shown in the Table 3. It is obvious that our method achieves far higher accuracy on the standard definition dataset than other deep learning models.

In contrast, our model has achieved ideal results in all indicators while maintaining a simple model structure. Since fake videos detecting applications always require detecting algorithm to be low time and computation consumed, our system meets the demand with high practicality and accuracy.

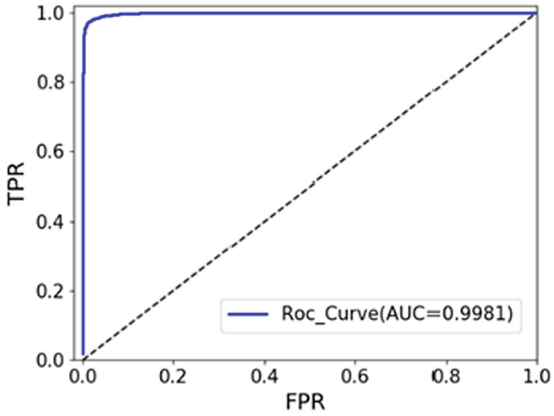

Fig. 8. ROC curve

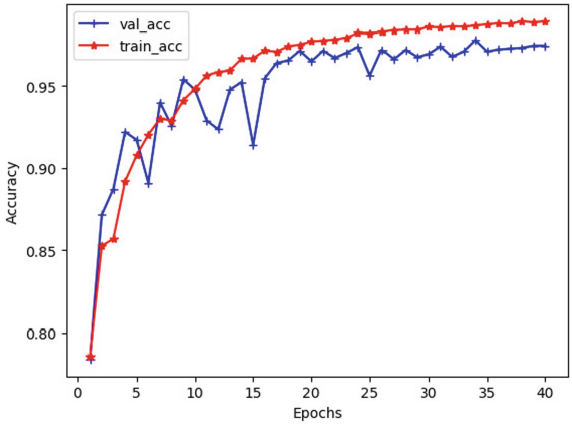

Fig. 9. The changing curve of model accuracy

Table 3. Comparison of accuracy results

Deep learning model	Accuracy (%)	AUC results
MesoNet [10]	**92.00%**	**0.9859**
XceptionNet [12]	**95.73%**	**0.6653**
Our model	**98.10%**	**0.9981**

5 Conclusion

Nowadays, fighting against Deepfake videos has become more and more important. In this paper, by analyzing the generation process of video, we find that facial expressions between adjacent frames are inevitably abnormal. In response

to this phenomenon, we propose a Deepfake video detection method base on subtle facial expressions. We employ optical flow to capture the obvious differences of facial expressions between adjacent frames in a video and incorporate the temporal characteristics of consecutive frames into a convolutional neural network (CNN) model to distinguish the Deepfake video.

According to experiment results, our model achieves great performance, with an accuracy much higher than most of the existing models, and at the same time, the complexity of the model can be greatly reduced.

In conclusion, our method can ensure both effectiveness and practicability. Our future work includes doing further research on deep fake videos with different quality levels, and realize automatic adjustment of parameters.

Acknowledgment. This work is partly supported by National Natural Science Foundation of China under Grant No.61672394 and 61872273. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

References

1. Deepfacelab. http://deepfakes.com.cn/. Accessed 22 Apr 2019
2. Deepfake github. http://github.com/deepfakes/faceswap. Accessed 20 Apr 2019
3. Deepfake wikipedia. https://en.wikipedia.org/wiki/Deepfake. Accessed 12 Sept 2019
4. Dlib. http://dlib.net/. Accessed 20 Mar 2019
5. Fakeapp. http://www.fakeapp.com/. Accessed 22 Apr 2019
6. Ffmpeg. http://ffmpeg.org/. Accessed 20 Mar 2019
7. Keras. http://keras.io/. Accessed 10 July 2018
8. Tensorflow. http://tensorflow.google.cn/. Accessed 10 July 2018
9. Zao. https://apps.apple.com/cn/app/zao/id1465199127. Accessed 12 Sept 2019
10. Afchar, D., Nozick, V., Yamagishi, J., Echizen, I.: Mesonet: a compact facial video forgery detection network. In: 2018 IEEE International Workshop on Information Forensics and Security, WIFS 2018, Hong Kong, China, 11–13 December 2018, pp. 1–7 (2018). https://doi.org/10.1109/WIFS.2018.8630761
11. Agarwal, S., Farid, H., Gu, Y., He, M., Nagano, K., Li, H.: Protecting world leaders against deep fakes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 38–45 (2019)
12. Chollet, F.: Xception: deep learning with depthwise separable convolutions. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017, pp. 1800–1807 (2017). https://doi.org/10.1109/CVPR.2017.195
13. Goodfellow, I.J., et al.: Generative adversarial networks. CoRR abs/1406.2661 (2014). http://arxiv.org/abs/1406.2661
14. Güera, D., Delp, E.J.: Deepfake video detection using recurrent neural networks. In: 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), pp. 1–6 (2018). https://doi.org/10.1109/AVSS.2018.8639163
15. Li, H., Li, B., Tan, S., Huang, J.: Detection of deep network generated images using disparities in color components. arXiv preprint arXiv:1808.07276 (2018)

16. Li, Y., Chang, M., Lyu, S.: In ictu oculi: exposing AI created fake videos by detecting eye blinking. In: 2018 IEEE International Workshop on Information Forensics and Security, WIFS 2018, Hong Kong, China, 11–13 December 2018, pp. 1–7 (2018). https://doi.org/10.1109/WIFS.2018.8630787
17. Lyu, Y.L.S.: Exposing deepfake videos by detecting face warping artifacts. CoRR abs/1811.00656 (2018). http://arxiv.org/abs/1811.00656
18. McCloskey, S., Albright, M.: Detecting GAN-generated imagery using color cues. arXiv preprint arXiv:1812.08247 (2018)
19. Nataraj, L., et al.: Detecting GAN generated fake images using co-occurrence matrices. arXiv preprint arXiv:1903.06836 (2019)
20. Rössler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J., Nießner, M.: Faceforensics: a large-scale video dataset for forgery detection in human faces. CoRR abs/1803.09179 (2018). http://arxiv.org/abs/1803.09179
21. Sabir, E., Cheng, J., Jaiswal, A., AbdAlmageed, W., Masi, I., Natarajan, P.: Recurrent convolutional strategies for face manipulation detection in videos. Interfaces (GUI) **3**, 1 (2019)
22. Taheri, S., Veidenbaum, A.V., Nicolau, A., Hu, N., Haghighat, M.R.: Opencv.js: computer vision processing for the open web platform. In: Proceedings of the 9th ACM Multimedia Systems Conference, MMSys 2018, Amsterdam, The Netherlands, 12–15 June 2018, pp. 478–483 (2018). https://doi.org/10.1145/3204949.3208126
23. Wang, L., et al.: Temporal segment networks: towards good practices for deep action recognition. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9912, pp. 20–36. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46484-8_2
24. Yang, X., Li, Y., Lyu, S.: Exposing deep fakes using inconsistent head poses. In: IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, 12–17 May 2019, pp. 8261–8265 (2019). https://doi.org/10.1109/ICASSP.2019.8683164