Chapter 11
Deepfake Detection Using Multiple Data Modalities

Hanxiang Hao, Emily R. Bartusiak, David Güera, Daniel Mas Montserrat, Sriram Baireddy, Ziyue Xiang, Sri Kalyan Yarlagadda, Ruiting Shao, János Horváth, Justin Yang, Fengqing Zhu, and Edward J. Delp

Abstract Falsified media threatens key areas of our society, ranging from politics to journalism to economics. Simple and inexpensive tools available today enable easy, credible manipulations of multimedia assets. Some even utilize advanced artificial intelligence concepts to manipulate media, resulting in videos known as *deepfakes*. Social media platforms and their "echo chamber" effect propagate fabricated digital content at scale, sometimes with dire consequences in real-world situations. However, ensuring semantic consistency across falsified media assets of different modalities is still very challenging for current deepfake tools. Therefore, cross-modal analysis (e.g., video-based and audio-based analysis) provides forensic analysts an opportunity to identify inconsistencies with higher accuracy. In this chapter, we introduce several approaches to detect deepfakes. These approaches leverage different data modalities, including video and audio. We show that the presented methods achieve accurate detection for various large-scale datasets.

11.1 Introduction

The rapid proliferation of easy-to-use machine learning tools contributes to an ever-increasing amount of manipulated media. These tools enable users to create realistic and believable face swaps in images and videos. They also convincingly alter or replace audio tracks in videos. Some of these tools use machine learning (ML) and deep learning (DL) techniques. Videos (with or without audio) generated with deep learning methods are collectively referred to as the term *deepfakes*. Recently, many methods have been developed to effectively detect these deepfake videos. Since most of the deepfake videos still contain the artifacts that are caused by inaccurate face swapping (i.e., splicing artifacts), [1, 2] propose to detect these manipulated videos

H. Hao · E. R. Bartusiak · D. Güera · D. Mas Montserrat · S. Baireddy · Z. Xiang · S. K. Yarlagadda · R. Shao · J. Horváth · J. Yang · F. Zhu · E. J. Delp (✉)
Video and Image Processing Laboratory (VIPER), School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA
e-mail: ace@ecn.purdue.edu

C. Rathgeb et al. (eds.), *Handbook of Digital Face Manipulation and Detection*, Advances in Computer Vision and Pattern Recognition,
https://doi.org/10.1007/978-3-030-87664-7_11

by finding the temporal inconsistency of 3-D head pose and facial landmarks using Support Vector Machine (SVM). Most of the deepfake generation tools are based on the Generative Adversarial Networks (GANs). In [3, 4], several deep-learning-based detectors are proposed to discriminate between authentic images and GAN-generated images obtained from various GAN-based deepfake generators. In order to improve the generalizability of the detection methods, [5] uses metric learning and adversarial learning to enable to the deepfake detection method trained only with authentic videos without the requirement of manipulated videos. Please refer to [6–9] for the completed survey about the deepfake detection methods.

In this chapter, we present various methods to detect the manipulated videos by leveraging different data modalities (e.g., video, audio). We first propose an approach to detect deepfakes by utilizing spatiotemporal information present in videos. More specifically, we use Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) to extract visual and temporal features from video frame sequences to accurately detect manipulations. This technique focuses on face swapping detection by examining the visual and temporal features of facial regions in each frame. Some frames may contain blurry faces, hindering effective detection of manipulations. To solve this issue, we utilize a novel attention mechanism to emphasize reliable frames and disregard low-quality frames in each sequence.

Next, we present a method that analyzes audio signals to determine whether they contain real human voices or fake human voices (i.e., voices generated by neural acoustic and waveform models). Instead of analyzing the audio signals directly, the proposed approach converts the audio signals into spectrogram images displaying frequency, intensity, and temporal content and evaluates them with a CNN. We convert the audio signals into spectograms in order to leverage frequency information and provide a more amenable configuration of the data to a CNN. A CNN can analyze different frequency ranges more explicitly from a spectrogram, revealing artifacts in certain frequency ranges. This method can also aid in a deepfake detection task in which the audio as well as the visual content has been manipulated. Analysts can use our method to verify the voice tracks of videos and flag them as manipulated if either the audio analysis or the video analysis reveals manipulated content.

Finally, we extend the previous video-based and audio-based methods to detect deepfakes using audio-video inconsistency. As mentioned previously, ensuring semantic consistency across these manipulated media assets of different modalities is still very challenging for current deepfake tools. For a photo-realistic deepfake video, a visual analysis alone may not be able to detect the manipulations, but pairing the visual analysis with audio analysis provides an additional avenue for authenticity verification. Therefore, we also describe several existing methods to analyze the correlations between lip movements and voice signals via phoneme-viseme mismatching and affective cues. These methods incorporate both video and audio data modalities, which provide rich information for deepfake detection.

The remaining sections in this chapter are structured as follows. Section 11.2 discusses a deepfake detection method that relies only on video content. Section 11.3 presents a method that introduces audio analysis to detect manipulated audio. Finally,

Sect. 11.4 explores several methods to evaluate audio-video inconsistency for deepfake video detection, building off of the methods presented in Sect. 11.3.

11.2 Deepfake Detection via Video Spatiotemporal Features

With the fast development of deepfake techniques, deepfake videos seem more and more realistic, causing viewers to struggle to determine their authenticity. However, current deepfake techniques still suffer from temporal inconsistency issues, such as flickering and unrealistic eye blinking. In this section, we introduce a deep learning-based method to detect deepfakes by incorporating temporal information with video frames.

Figure 11.1 shows the block diagram of our spatiotemporal-based method. A shared CNN model first encodes input video frames into deep features. CNNs have achieved success in many vision tasks, such as image recognition and semantic segmentation. In our case, we utilize these CNN models to extract features for deepfake detection. In recent literature [10, 11], InceptionV3 [12], EfficientNet [13], Xception model [14], or an ensemble of these models have been used to extract deepfake features. Transfer learning is also used to fine-tune these models that are pretrained on some large-scale image datasets (e.g., ImageNet [15]) to speed up training processes and improve performance. We will compare the results achieved with these CNNs in Sect. 11.2.6. A shared CNN model also reduces the number of parameters that must be trained. This technique will force the model to extract the features that are agnostic to the input video content and manipulation methods, which is important to make the model generalize better to new deepfake videos.

Then, we input the features to a temporally aware network to leverage the relationship between frames. There are many types of temporally aware networks, including Recurrent Neural Networks (RNNs), Long Short-Term Memory networks

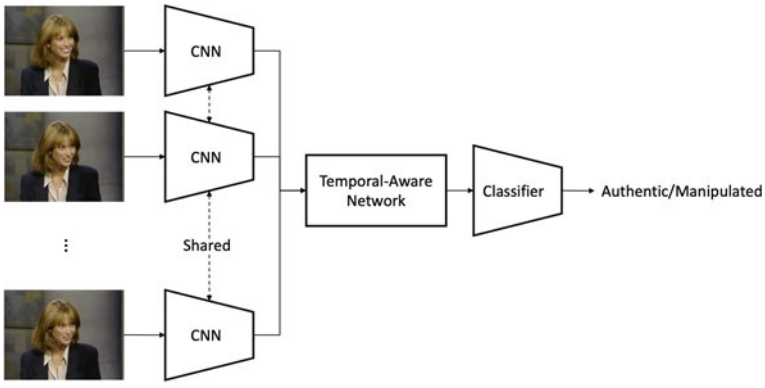

Fig. 11.1 Overview of the spatiotemporal deepfake detection system

(LSTMs) [16], and Gated Recurrent Units (GRUs) [17]. LSTMs and GRUs are special kinds of RNNs that are capable of learning long-term dependencies across sequences. For our deepfake detection task, a GRU will analyze CNN-extracted features from video frames to accumulate useful information related to deepfake artifacts. The GRU leverages temporal information implicitly to reveal deepfakes, rather then being explicitly designed to focus on temporal inconsistencies. This feature will also prepare the model to generalize better to different types of deepfakes. The result of the GRU is a new representation of the video in the latent space that contains discriminating information from the entire video.

Next, we use a classifier to label a video as authentic or manipulated. For deep learning models, people often use a multi-layer perceptron (MLP) (i.e., fully connected layers) as a classifier along with batch normalization and Rectified Linear Unit (ReLU).

11.2.1 Overview

In this section, we introduce the details of our video-based deepfake detection approach. The main workflow commences with CNN-based face detection followed by CNN-based facial feature extraction to determine a set of salient facial indicators that will aid in manipulation detection. Then, the facial features are analyzed by an Automatic Face Weighting (AFW) mechanism and a Gated Recurrent Unit (GRU) network to extract meaningful features to verify videos as authentic or manipulated. Additionally, a Boosting Network is used to aid the backbone network in learning to discriminate between authentic and manipulated videos.

11.2.2 Model Component

The next few sections detail the architecture of the main ensemble network, which consists of the CNN-based face detector, CNN-based feature extractor, AFW, and GRU.

Face Detection. For our analysis, we focus on faces, which typically are the primary target of deepfakes. This means that face regions generally contain indicators of a video's true nature. Thus, the first step in our approach is to locate faces within video frames. We use a Multi-task Cascaded Convolutional Network (MTCNN) [18] for this task since it produces bounding boxes around faces and facial landmarks simultaneously. MTCNN consists of three stages. In the first stage, a fully convolutional network, called Proposal Network, generates a large number of face bounding box proposals. In the second stage, another CNN, called Refine Network, improves the output from the first stage by rejecting a large number of false proposals. The remaining valid proposals are passed to the third stage, where the bounding boxes and facial landmark positions are finalized. Non-maximum suppression and

bounding box regression are applied in every stage to suppress overlapping proposals and refine the output prediction.

To speed up face detection, we downsample each video by a factor of 4 and extract faces from 1 in every 10 frames. We also expand the margin of detected face bounding boxes by 20 pixels to include possible deepfake artifacts around the edges of faces and hairlines. After face detection, we resize all face occurrences to 224×224 pixels.

Face Feature Extraction. After detecting faces in the frames, we begin to train our deepfake detection model to identify authentic and manipulated faces. We extract features with another CNN and perform binary classification to determine if the faces contain authentic or manipulated information. Because of the large amount of video data that needs to be processed, we prioritize CNNs that are both fast and accurate for this task. In the end, we chose to use EfficientNet-b5 [13] since it was designed with neural architecture search to ensure it is both lightweight and highly accurate.

We further enhance EfficientNet by training it with the additive angular margin loss (ArcFace) [19] as opposed to softmax and cross-entropy. ArcFace is a learnable loss function that modifies the regular classification cross-entropy loss to ensure a more efficient representation. It aims to enforce a margin between each class in the latent feature space obtained from the previously mentioned CNN models. This results in features that are forced to be highly discriminative, resulting in a more robust classification.

Automatic Face Weighting. After classifying each frame as manipulated or not, we have to determine a classification for the entire video. The straightforward option is to simply average the classifications of the frames to come up with a video classification. However, this may not be the best option. Generally, face detectors are accurate, but sometimes they incorrectly categorize background regions in images as "faces", which can impact frame-level and video-level classifications in downstream applications. Additionally, there is no limit on the number of faces in a frame, of which any number can be authentic or manipulated. Faces can also be blurry or noisy, which further complicates direct averaging of frame predictions.

To address this issue, we propose an automatic face weighting (AFW) mechanism that highlights the faces that reliably contribute to the correct video-level classification while disregarding less reliable faces. This approach can be considered similar to the attention mechanisms found in transformer networks [20]. We assign a weight w_j to the output label l_j determined by EfficientNet for the jth extracted face. Using these weights, we can calculate a weighted average of all the frames' labels to obtain a final classification for the video. Both labels l_j and weights w_j are estimated by a fully connected linear layer that takes the EfficientNet features as input, meaning that the EfficientNet features are used to determine a label for how much a face has been manipulated (l_j) as well as how confident the network is of its classification (w_j). The final output probability p_w of a video being manipulated can be calculated as

$$p_w = \sigma\left(\frac{\sum_{j=1}^{N} w_j l_j}{\sum_{j=1}^{N} w_j}\right), \tag{11.1}$$

where w_j and l_j are the weight value and label obtained for the jth face region, respectively, N is the total number of frames under analysis, and $\sigma(.)$ refers to the sigmoid function. To ensure that $w_j \geq 0$, we pass w_j through a ReLU function. We also perturb the values with a small value to avoid division by 0. This process ensures we have an adaptive approach to combine frame-level classifications to obtain a video-level classification.

Gated Recurrent Unit. In this work, we choose Gated Recurrent Unit (GRU) [17] as the temporal-aware network. As previously mentioned, LSTM and GRU are special kinds of RNNs. Both of them improve the original RNN using multiple gated units to resolve the vanishing gradient issue in order to learn the long-term dependencies across sequences. Due to the less complicated structure, we choose GRU instead of LSTM to reduce the training time. GRU is used to analyze all previously computed values in a temporal manner to evaluate the information learned over time. More specifically, GRU operates on vectors describing each face detected in a video, where the vectors consist of 1,048 facial features extracted with EfficientNet for frame j, the logit l_j, the weight w_j, and the probability of manipulation p_w computed with AFW.

The GRU consists of three stacked, bi-directional layers, and a uni-directional layer with a hidden layer of 512. The final layer consists of a linear layer with a sigmoid activation function to estimate a final probability, p_{RNN}, which describes the likelihood that the video is manipulated.

Weight Initialization. Each network of the overall ensemble is initialized with weights in a manner that will help it best succeed. We use a pretrained MTCNN for face detection. The EfficientNet face extractor is initialized with weights pretrained on ImageNet, and the AFW and GRU are initialized with random weights. Before training the entire ensemble in an end-to-end fashion, we train the EfficientNet with the ArcFace loss on 2,000 batches of cropped faces selected randomly. Although this initial training step is not necessary to increase the accuracy of the overall approach, our experiments indicated that it aided the network in faster convergence with a more stable training process. This step ensures the parameters passed onto the rest of the network are more suited to our deepfake detection task.

Loss Function. The network utilizes three different loss functions. The first is ArcFace loss, which operates on the output of EfficientNet. It is used only to update the weights of EfficientNet to extract facial features based on batches of cropped faces from randomly selected frames and videos. The second loss function is a binary cross-entropy (BCE) loss, which operates on the AFW prediction p_w. It is used to update the weights associated with EfficientNet and the AFW. The third loss function is another BCE, which operates on the GRU prediction p_{RNN}. It is used to update the weights of EfficientNet, the AFW, and the GRU. The ArcFace loss evaluates frame-level classifications, while the BCE losses evaluate video-level predictions.

11.2.3 Training Details

In this work, we train and evaluate the proposed method on the Deepfake Detection Challenge (DFDC) Dataset [21]. We split the dataset into training, validation, and testing sets with the ratio of 3:1:1. Since our approach consists of many components that rely upon each other, it is important to train each portion properly to ensure the success of the overall ensemble. We train our facial feature extractor (i.e., EfficientNet), the AFW, and the GRU ourselves, but we do not train or update the MTCNN. The entire ensemble is trained end-to-end with the Adam optimizer [22] and a learning rate of 0.001.

Our method can only afford to evaluate one video at a time during training due to the size of the network, the number of frames, and GPU computational limits. However, the network parameters are updated after processing groups of videos. EfficientNet is updated with the ArcFace loss after 256 random frames, and the entire ensemble is updated with the BCE losses after 64 videos. During training, we oversample videos that contain genuine, authentic faces to balance the dataset so that the network is presented with balanced manipulated and authentic faces during the training process.

11.2.4 Boosting Network

In order to further improve the model performance, we also utilize a boosting network. The boosting network is a duplicate of the backbone with a different objective. Instead of minimizing BCE on class predictions, the boosting network strives to predict error in the logit domain between predictions and the true classifications for both the AFW and the GRU. More specifically, the output of the AFW layer is defined as

$$p_w^b = \sigma\left(\frac{\sum_{j=1}^{N}(w_j l_j + w_j^b l_j^b)}{\sum_{j=1}^{N}(w_j + w_j^b)}\right), \tag{11.2}$$

where w_j and l_j refer to the weights and logits produced by the main network and w_j^b and l_j^b refer to the weights and logits produced by the boosting network for the jth face region. N is the total number of frames under analysis, and $\sigma(.)$ refers to the sigmoid function. In a similar manner, the output of the GRU is defined as

$$p_{RNN}^b = \sigma(l_{RNN} + l_{RNN}^b), \tag{11.3}$$

where l_{RNN} refers to the logit produced by the GRU of the main network, l_{RNN}^b refers to the logit produced by the GRU of the boosting network, and $\sigma(.)$ refers to the sigmoid function. The main network is trained on the training data, while the boosting network is trained on the validation data. The main network and the boosting network interact in the AFW layer and after the GRU.

11.2.5 Test Time Augmentation

We leverage one other technique to enhance the performance of our approach: data augmentation during testing. Data augmentation has been used in training to reduce overfitting. However, in our experiments, we discover that using the following data augmentation procedure during testing can reduce the incorrect and overconfident predictions. Once the MTCNN identifies facial regions in a desired frame, we crop the designated areas in the desired frame, in the previous two frames, and in the following two frames. We repeat this for all frames in the test sequence, resulting in five sequences of video frames depicting faces. Next, we randomly apply a horizontal flip data augmentation to each sequence and run each of the sequences through our full model. The final classification prediction for a video sequence is the average of the five predictions on the shifted sequences. This technique decreases the number of incorrect and overconfident predictions since averaging smooths out anomalous predictions.

11.2.6 Result Analysis

We train and evaluate the proposed method on the Deepfake Detection Challenge (DFDC) Dataset [21]. In addition, we make quantitative comparisons with EfficientNet [13], Xception [14], Conv-LSTM [10], and a modified version of Conv-LSTM using the facial regions detected by MTCNN as input. For the EfficientNet [13] and Xception [14] networks, the final prediction result of each video is obtained by averaging the predictions of each frame.

We select a configuration for each model based on the validation set with balanced authentic/manipulated data. The corresponding Receiver Operating Characteristic (ROC) and Detection Error Trade-off (DET) curves are shown in Fig. 11.2. Since the Conv-LSTM method extracts the features based on the entire video frames, it cannot effectively capture the manipulations that occur in facial regions. However, when we use the detected facial regions instead of the entire frames as input, the detection performance improves significantly. The two typical CNN models EfficientNet-b5 [13] and Xception [14] have achieved good performance in manipulation detection based on video frames. The results of the proposed method indicate that performance of EfficientNet-b5 can be further improved by adding an Automatic Face Weighting layer (AFW) and a Gated Recurrent Unit (GRU).

We also evaluate how the boosting network and data augmentation affects the results in the testing phase. In order to do so, we use the log-likelihood error (the lower the better) to represent the system performance, since log-likelihood score can penalize heavily for being confident but wrong. The results are shown in Table 11.1. It demonstrates that by including both the boosting network and test augmentation at the same time, the log-likelihood can be decreased to 0.321.

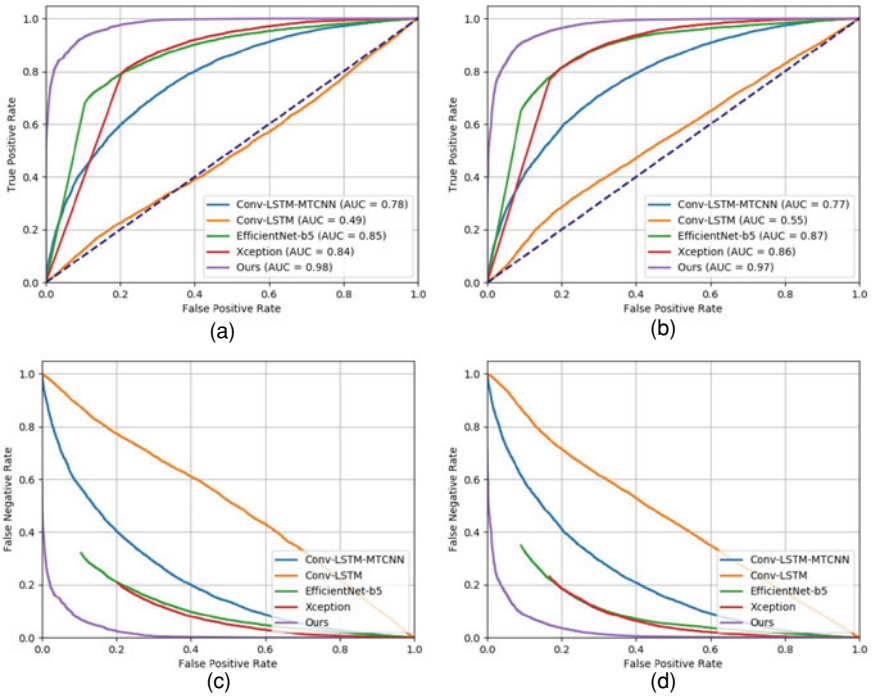

Fig. 11.2 The manipulation detection performance comparison. Figures **a** and **b** are the ROC curves obtained from validation and testing sets, respectively. Figures **c** and **d** are the DET curves obtained from validation and testing sets, respectively

Table 11.1 The log-likelihood error results

Method	Log-likelihood
Baseline	0.364
Baseline + Boosting network	0.341
Baseline + Boosting network + Test augmentation	0.321

11.3 Deepfake Detection via Audio Spectrogram Analysis

Visual content is just one data modality that can be altered. Audio attacks can be used to spoof devices to gain access to personal records. They may also be used to change the message delivered by a figure in a video. Such attacks may consist of only newly synthesized audio to achieve a nefarious objective. Other times, falsified audio may be used in deepfakes to sync with the newly generated faces (or just lips) in the videos [23]. We need methods to analyze standalone audio signals as well as signals that accompany visual content to verify the authenticity of the messages we hear.

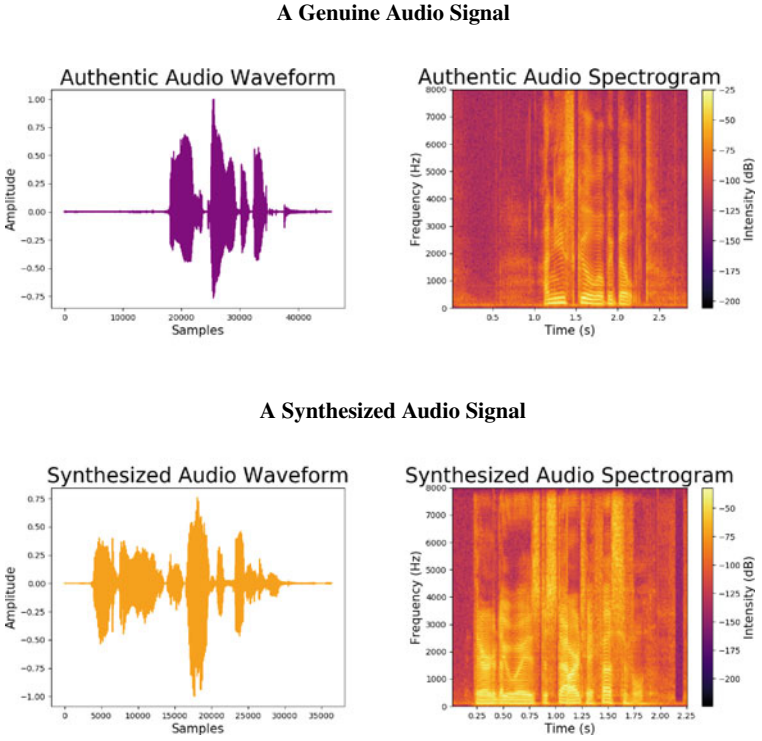

Fig. 11.3 *Left column:* Raw audio waveforms, where purple indicates an authentic audio signal and orange indicates a synthesized audio signal. *Right column:* Spectrograms corresponding to the raw audio waveforms, which serve as inputs to the CNN to classify the signals based on authenticity

In this section, we present a method that analyzes audio signals to determine their authenticity. Our approach works by analyzing audio signals in the form of spectrograms, as shown in Fig. 11.3, with a Convolutional Neural Network (CNN). This work can prevent spoofing attacks by analyzing audio signals on their own, or it can aid in the detection of deepfakes by adding audio analysis to a video analysis as shown in Sect. 11.4.

11.3.1 Overview

We present a method that analyzes a few seconds of an audio signal and identifies whether it is genuine human speech or synthesized speech. Figure 11.4 depicts an overview of our method. It consists of four main steps. First, we apply the Fourier Transform to raw audio waveforms. Then, we use the resulting Fourier coefficients to

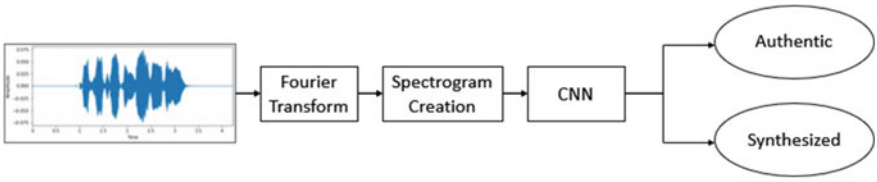

Fig. 11.4 Proposed Method. The proposed approach applies the Fourier Transform to raw audio signals to generate spectrogram "images"—the inputs to the CNN. The CNN classifies signals as *authentic* or *synthesized*

construct spectrograms of the audio waveforms. Next, we analyze the spectrograms with a CNN, and finally we classify audio signals as *authentic* or *synthesized*.

11.3.2 Dataset

For our experiments, we utilize the dataset [24] of the 2019 Automatic Speaker Verification Spoofing and Countermeasures Challenge (ASVspoof2019) [25]. This large-scale dataset contains 121,467 audio tracks. Some of the audio samples are authentic and contain recordings of humans speaking. Other samples contain audio to be used in spoofing attacks. The inauthentic audio samples were generated via voice conversion (VC), speech synthesis (SS), and replay methods. Since our ambitions focus more on deepfake detection than spoofing attacks, we only consider audio signals that have been synthetically generated to replicate human voices, which is included in the VC and SS subsets. This data was generated with neural acoustic models and Artificial Intelligence, including Long Short-Term Memory networks (LSTMs) and Generative Adversarial Networks (GANs). For training and evaluating our CNN classifier, we utilize the official dataset split of the ASVspoof2019 challenge, which divides the full dataset into 25,380 training audio tracks, 24,844 validation tracks, and 71,243 testing tracks.

11.3.3 Spectrogram Generation

The first step in our audio verification method is to apply the Fourier transform to raw audio signals. A Fast Fourier Transform (FFT) is a method that efficiently computes the Discrete Fourier Transform (DFT) of a sequence. We utilize the FFT to compute the Fourier coefficients of an audio signal under analysis. Then, we convert the Fourier coefficients to decibels. The second step in our approach is to construct spectrograms of the audio signals. We create spectrogram "images" of size 50x34 pixels to analyze with our CNN. Examples of the spectrograms created for our dataset are shown in Fig. 11.3.

Spectrograms convey information about the intensity of an audio signal over time and frequency. One axis depicts time and the other depicts frequency. The intensity of an audio signal is represented via color at a specific time and frequency. Brighter colors that are closer to shades of yellow indicate greater intensity and volume of the audio signals. On the other hand, darker colors that are closer to shades of purple or black indicate lower intensity and quieter volume of the audio signals. Although these colors assist us in seeing the differences in intensity over time and frequency of an audio signal, we do not use them in the spectrograms analyzed by the CNN. After the spectrogram images are constructed, we remove the color and convert the images to grayscale. We also normalize their values to prepare them for analysis by the CNN.

11.3.4 Convolutional Neural Network (CNN)

Since our method analyzes spectrogram "images", our CNN employs 2-D convolutions. This is in contrast to a CNN that analyzes a raw audio waveform, which would utilize 1-D convolutions across the 1-D sequence. By using 2-D convolutions to analyze spectrograms, our method incorporates intensity information over frequency and time.

Table 11.2 outlines the specifics of the network architecture. It mainly consists of two convolutional layers. Next, it utilizes max pooling and dropout to introduce regularization into the network and decrease the chances of overfitting. After two dense layers and more dropout, the CNN produces a final class prediction, indicating whether the audio signal is authentic or synthesized. We train the CNN for 10 epochs using the Adam optimizer [26] and cross-entropy loss function.

Table 11.2 CNN Details. This table specifies the parameters of the developed CNN. Each row in the table describes *(from left to right)* the function of the layer, its output shape, and the number of parameters it introduces to the CNN. (N, H, W) refers to the number of feature maps produced by the layer (N), along with their height H and width W

Layer	Output shape (N, H, W)	Parameters
conv_1	(32, 48, 32)	320
conv_2	(30, 46, 64)	18,496
max pooling	(15, 23, 64)	0
dropout_1	(15, 23, 64)	0
flatten_1	(22080)	0
dense_1	(128)	2,826,368
dropout_2	(128)	0
dense_2	(2)	258

Table 11.3 Results. This table presents results achieved with the baseline random classifier and our CNN approach

Method	Accuracy (%)	Precision (%)	Recall (%)	F-1 (%)
Baseline (random)	49.98	50.12	50.34	40.69
Proposed method	**82.54**	**66.00**	**81.38**	**68.93**

11.3.5 Experimental Results

Table 11.3 summarizes the results of our method. We evaluate our results based on accuracy, precision, recall, and F1-score. We also calculate Receiver Operator Characteristic (ROC), Detection Error Trade-off (DET), and Precision-Recall (PR) curves. We demonstrate the success of our method over a random classifier, which serves as a baseline for comparison. The random classifier randomly guesses whether an audio signal is authentic or synthesized according to a uniform random distribution. Results indicate that our method outperforms the baseline random classifier based on all metrics.

Figure 11.5 shows Receiver Operating Characteristic (ROC), Detection Error Trade-off (DET), and Precision-Recall (PR) curves for our results in comparison to the baseline. Our approach achieves a high ROC-AUC of 0.8975, which outperforms the baseline ROC-AUC of 0.5005. The PR-AUC exhibits similar behavior. Our method achieves PR-AUC of 0.4611, while the baseline PR-AUC settles at 0.1024. All metrics included in both the table and the figures indicate that our method accomplishes better verification of audio signals than the baseline for both the validation and testing sets.

Considering that the testing dataset contains new audio attacks which were never seen before in training and validation, these results are very promising. Analysis of audio signals in the frequency domain formatted as spectrograms is effective for an audio verification task. It can also be used as audio features for audio-video inconsistency analysis in the following section.

11.4 Deepfake Detection via Audio-Video Inconsistency Analysis

The previously mentioned audio analysis technique can aid in the detection of deepfake videos by extending the scope to include two different media modalities. For videos in which only the audio has been altered, this method will complement a pixel analysis method. For some realistic deepfakes, a visual analysis alone may not be able to detect the manipulations, but pairing the visual analysis with audio analysis provides an additional avenue for authenticity verification.

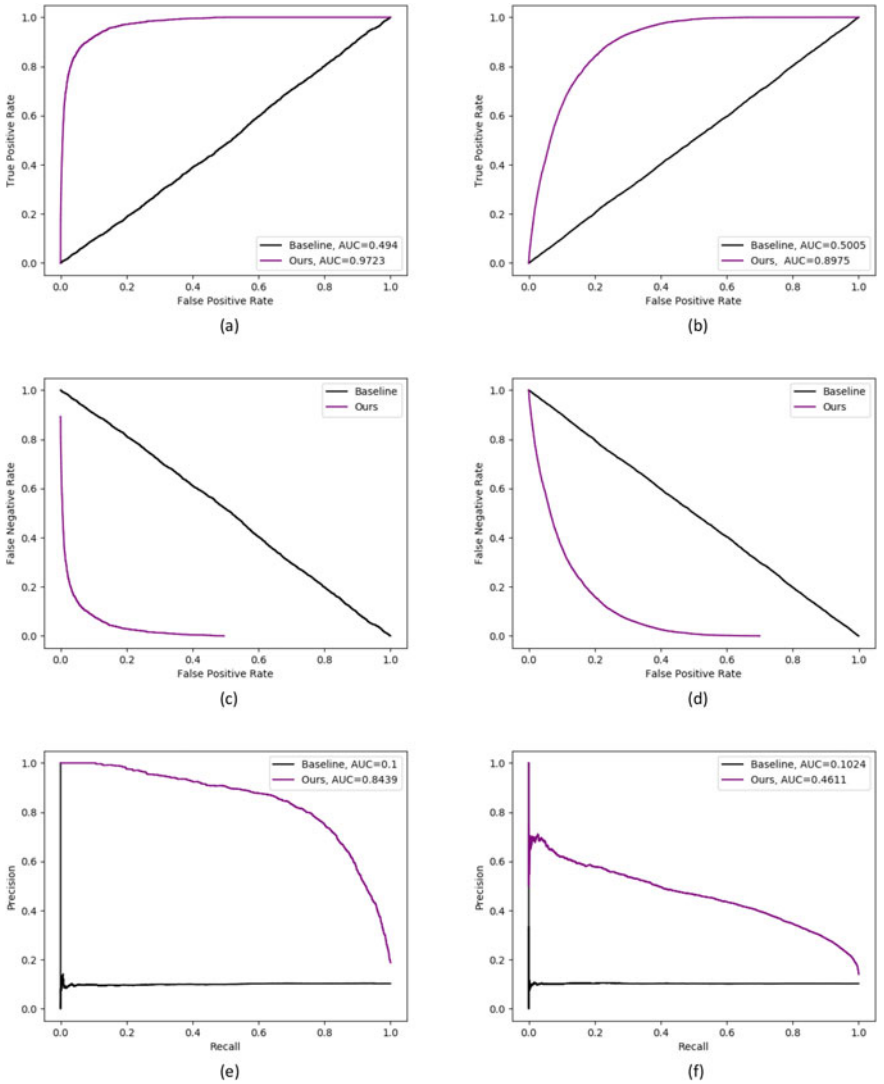

Fig. 11.5 ROC, DET, PR curves. Figures **a** and **b** show the ROC curves obtained from the validation and testing sets, respectively. Figures **c** and **d** show the DET curves obtained from the validation and testing sets, respectively. Figures **e** and **f** show the PR curves obtained from the validation and testing sets, respectively

In this section, we discuss detecting deepfakes by analyzing the natural correlations that manifest when lip movements are coherent with the voice in videos of speaking persons. Then, the absence of such correlations in videos will point to plausible manipulations. Several works [27, 28] have explored this direction. For example, Korshunov et al. [27] propose to use lip keypoints obtained from 68-point facial landmarks and audio Mel-frequency cepstrum to check their consistency. These lip and audio features are concatenated together via Principal Component Analysis (PCA) for dimensionality reduction. Then, we can use these features to train a classifier (e.g., Gaussian mixture model, SVM, or LSTM) for deepfake detection.

However, simply concatenating the visual features and audio features does not always work, especially due to the large variation of possible facial and head movements and individual appearance differences. In the following sections, we will describe several deepfake detection methods based on the work [28, 29] to provide more reliable approaches using audio and video inconsistency analysis.

11.4.1 Finding Audio-Video Inconsistency via Phoneme-Viseme Mismatching

As described earlier, current deepfake techniques are still not able to produce coherent lip-sync manipulated videos. To exploit this, Agarwal *et al.* [28] propose to explicitly detect the mismatch of phonemes and visemes. A phoneme is a distinct unit of human speech, while viseme is the counterpart of a phoneme for lip movement. In their work, they focus only on the close-mouth phoneme, such as the phoneme group of *M* (e.g., mother), *B* (e.g., brother), and *P* (e.g., parent), since detecting closed lips is more accurate than other lip movements. If the audio narrative text is available, the closed-lip phoneme can be found directly through phonograms. If only audio data is provided, there are tools available to transcribe the audio track into text, such as the Speech-to-Text API from Google.[1] After finding the closed-lip phoneme, we describe an approach to detect the viseme.

Figure 11.6 shows how we detect the closed-lip viseme. 68-point facial landmarks are first detected given a RGB frame using an online tool.[2] As shown in Fig. 11.6, the landmark points include both inner and outer loops of the lips. To find if the lips are closed or open, we compute the two middle points of the upper and lower lips and collect the intensities of the pixels along the line segment shown as the red line in Fig. 11.6. Note that we use bilinear interpolation to obtain the pixel intensity along the line segment. The right two plots in Fig. 11.6 show the corresponding pixel intensity plot given the images on the left after converting to grayscale. We apply moving average with a window size of 10 to smooth the plots. Then we find the local maxima and local minima and their prominences, h_i and l_i, using the MATLAB function *findpeaks* for frame i. h_i measures the intensity drop from upper lip to the

[1] https://cloud.google.com/speech-to-text.

[2] https://github.com/1adrianb/face-alignment.

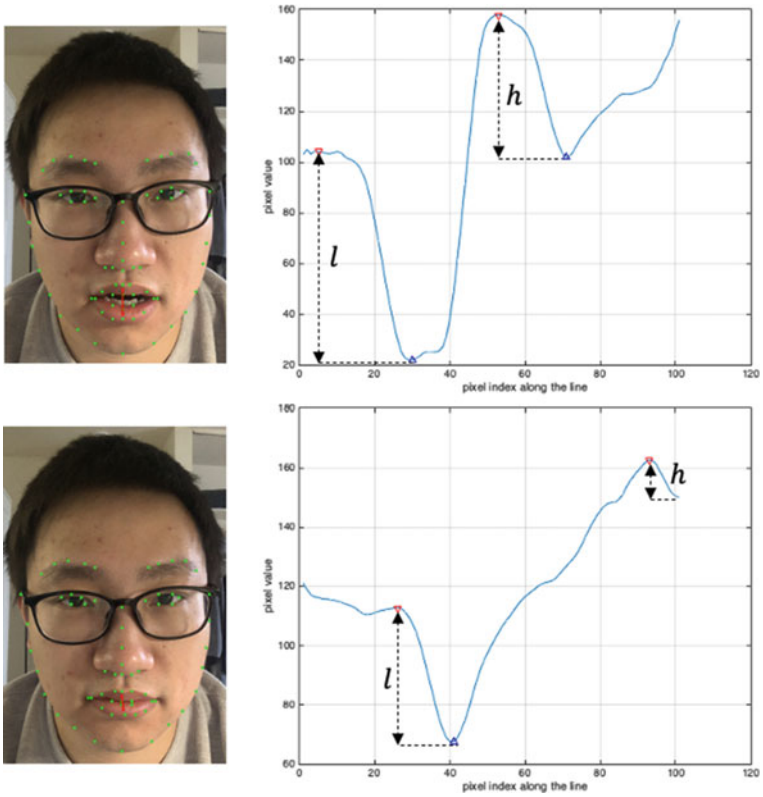

Fig. 11.6 Viseme detection. The first row shows the viseme profile of open mouth and the second row shows the case of closed mouth. The two images on the left are the original RGB images with inpainted landmarks and profile line (red line). The two plots on the right are the corresponding profile feature plots with local minima (blue triangle) and maxima (red triangle)

mouth interior, while l_i measures the intensity boost from mouth interior to lower lip. To detect a closed-lip viseme, given the reference h_r and l_r from a ground truth closed-lip frame, we measure the distance $|l_i - l_r| + |h_i - h_r|$. If the distance is smaller than a threshold value, it will be classified as a closed-lip viseme.

Given a closed-lip phoneme event at a specific event frame, we will first collect several frames before and after the event frame. If there is at least one closed-lip viseme that can be found in the selected frames, we consider the phoneme and viseme to match. Otherwise, we consider the phoneme and viseme mismatched. With this approach, we determine if the given video is deepfake or not by detecting phoneme-viseme mismatching.

This approach explicitly finds phoneme-viseme mismatching to detect audio-video inconsistency. However, it is not always necessary to explicitly find such a mismatch. In the following section, we introduce a method that uses a deep learning model to automatically detect deepfakes from audio and video data.

11.4.2 Deepfake Detection Using Affective Cues

In this section, we will introduce a method based on [29] that does not rely on the hand-designed audio and video features mentioned in Sect. 11.4.1. Instead, we will guide the model to learn a latent space that disentangles the manipulated/authentic data for both audio and video modalities. Different from the work in Sect. 11.2.1, which learns a manipulated/authentic discriminative latent space for video only, the presented work aims to find such a space for both audio and video, simultaneously.

Figure 11.7 shows the block diagram of our presented method. Given an image sequence, face features are extracted first using a CNN-based method, such as the method previously shown in Sect. 11.2.1. To extract audio features, we can use the same approach as proposed in Sect. 11.3 using spectrograms as audio features. Then, we pass the video feature f and audio feature s to two separate CNN models (i.e., video and audio modality embedders) to map input features into a latent space that is discriminative for manipulated/authentic data. Emotion features can also be extracted from f and s using a pretrained Memory Fusion Network (MFN) [30]. MFN is a deep learning model that aims to detect human emotion from different data modalities like facial expressions and speech. Similarly, we use two separate MFNs as video and audio emotion embedders to map the input features into the latent space that is discriminative for manipulated/authentic data. After obtaining the embeddings of video and audio modality features (m^f and m^s) and the embeddings of video and audio emotion features (e^f and e^s), we compute the feature distance (e.g., Euclidean distance or cosine distance) to determine if the input is a deepfake or not. There are many loss functions that are applicable to obtaining a discriminative latent space for the manipulated and authentic data, such as triplet loss [31] and ArcFace [19] (as described in Sect. 11.2.1).

As described above, we show that instead of solely relying on video modality, we can detect deepfakes using both audio and video modalities. These methods are more robust to new attacks (i.e., new deepfake techniques) because they consider more

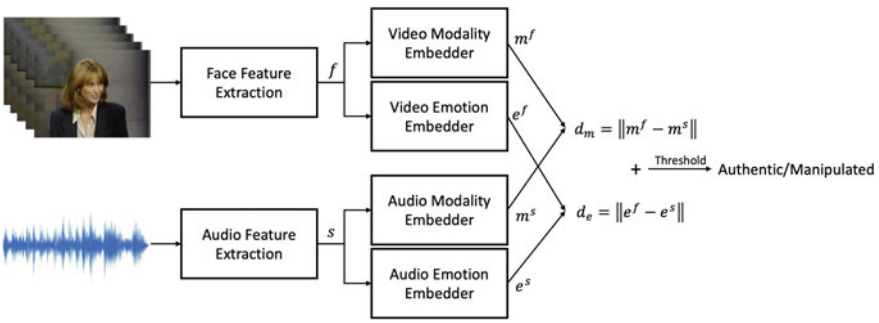

Fig. 11.7 Deepfake detection model using affective cues. The presented method extracts data modality features and emotion features from both audio and video. Then the detection result is obtained by jointly comparing the audio-video feature distances from data modality and emotion

information. As deepfakes continue to become more realistic, focusing on multiple data modalities can give us a better opportunity for accurate detection. Video and audio data modalities are not the only modalities that can assist in deepfake detection. Other data modalities (e.g., video metadata [32]) are also useful to improve the robustness of the detection algorithm. We believe that with the help of multi-modality and cross-modality analysis, detection methods will be more robust against future deepfake attacks.

11.5 Conclusion

In this chapter, we introduce several approaches that analyze deepfake features to determine their authenticity. First, we design a deepfake detection method that relies on spatiotemporal features obtained from video frames. Then, we pivot to incorporate an audio analysis to further improve our deepfake detection. We develop an audio-based method to detect synthetic speech based on spectrogram analysis. Next, we describe several methods that utilize both video frames and audio speech to detect deepfakes via audio-video inconsistency. We show that the presented approaches successfully identify deepfake videos from various large-scale datasets with high accuracy. The true potential of deepfakes is still untapped. We continue to evolve and innovate as new technology becomes available.

Acknowledgements This material is based on research sponsored by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory (AFRL) under agreement numbers FA8750-16-2-0173 and FA8750-20-2-1004. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA and AFRL or the U.S. Government.

References

1. Yang X, Li Y, Lyu S (2019) Exposing deep fakes using inconsistent head poses. In: Proceedings of the IEEE international conference on acoustics, speech and signal processing, May 2019
2. Yang X, Li Y, Qi H, Lyu S (2019) Exposing gan-synthesized faces using landmark locations. In: Proceedings of the international workshop on information hiding and multimedia security, July 2019
3. Marra F, Gragnaniello D, Cozzolino D, Verdoliva L (2018) Detection of gan-generated fake images over social networks. In: Proceedings of the IEEE conference on multimedia information processing and retrieval, April 2018
4. Gragnaniello D, Cozzolino D, Marra F, Poggi G, Verdoliva L (2021) Are GAN generated images easy to detect? A critical analysis of the state-of-the-art. In: Proceedings of the IEEE international conference on multimedia and expo, July 2021
5. Cozzolino D, Rössler A, Thies J, Nießner M, Verdoliva L (2021) Id-reveal: Identity-aware deepfake video detection. In: arXiv preprint arXiv:2012.02512, December 2021

6. Verdoliva L (2020) Media forensics and deepfakes: an overview. IEEE J Select Topics Signal Process 14(5):910–932
7. Tolosana R, Vera-Rodríguez R, Fiérrez J, Morales A, Ortega-Garcia J (2020) Deepfakes and beyond: a survey of face manipulation and fake detection. In: arXiv preprint arXiv:2001.00179, January 2020
8. Mirsky Y, Lee W (2021) The creation and detection of deepfakes: a survey. In: ACM Computing survey, vol 54, No 1, January 2021
9. Nguyen TT, Nguyen CM, Nguyen DT, Nguyen DT, Nahavandi S (2021) Deep learning for deepfakes creation and detection. In: arXiv preprint arXiv:1909.11573, April 2021
10. Güera D, Delp EJ (2018) Deepfake video detection using recurrent neural networks. IEEE international conference on advanced video and signal based surveillance, November 2018. Auckland, New Zealand, pp 1–6
11. Montserrat D, Hao H, Yarlagadda S, Baireddy S, Shao R, Horvath J, Bartusiak ER, Yang J, Guera D, Zhu F, Delp E (2020) Deepfakes detection with automatic face weighting. In: IEEE conference on computer vision and pattern recognition workshops, June 2020, pp 2851–2859
12. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016) Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition, June 2016, Las Vegas, pp 2818–2826
13. Tan M, Le QV (2019) Efficientnet: Rethinking model scaling for convolutional neural networks. In: arXiv preprint arXiv:1905.11946
14. Chollet F (2017) Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, July 2017, Honolulu, pp 1251–1258
15. Russakovsky O, Deng J, Hao S, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg AC, Fei-Fei L (2015) ImageNet large scale visual recognition challenge. Int J Comput Vis 115(3):211–252
16. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(7)
17. Cho K, van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proceedings of the conference on empirical methods in natural language processing, October 2014
18. Zhang K, Zhang Z, Li Z, Qiao Y (2016) Joint face detection and alignment using multitask cascaded convolutional networks. In: IEEE signal processing letters, vol 23, April 2016
19. Deng J, Guo J, Xue N, Zafeiriou S (2019) ArcFace: additive angular margin loss for deep face recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, June 2019, Long Beach
20. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. In: Proceedings of advances in neural information processing systems, December 2017, Long Beach, pp 5998–6008
21. Dolhansky B, Howes R, Pflaum B, Baram N, Ferrer CC (2019) The deepfake detection challenge (dfdc) preview dataset. In: arXiv preprint arXiv:1910.08854
22. Kingma D, Ba J (2015) Adam: A method for stochastic optimization. In: Proceedings of the IEEE conference on international conference for learning representations, May 2015
23. Suwajanakorn S, Seitz SM, Kemelmacher-Shlizerman I (2017) Synthesizing obama: learning lip sync from audio. ACM Trans Graph 36(4)
24. Yamagishi J, Todisco M, Sahidullah M, Delgado H, Wang X, Evans N, Kinnunen T, Lee K, Vestman V, Nautsch A (2019) Asvspoof 2019: The 3rd automatic speaker verification spoofing and countermeasures challenge database. University of Edinburgh, The Centre for Speech Technology Research
25. Todisco M, Yamagishi J, Sahidullah M, Delgado H, Wang X, Evans N, Kinnunen T, Lee K, Vestman V, Nautsch A (2019) Asvspoof 2019: Automatic speaker verification spoofing and countermeasures challenge evaluation plan. In: ASVspoof consortium, January 2019
26. Kingma D, Ba J (2015) Adam: a method for stochastic optimization. In: Proceedings of the international conference for learning representations, May 2015, San Diego

27. Korshunov P, Marcel S (2018) Speaker inconsistency detection in tampered videos. In: Proceedings of the IEEE European signal processing conference, September 2018, pp 2375–2379
28. Agarwal S, Farid H, Fried O, Agrawala M (2020) Detecting deep-fake videos from phoneme-viseme mismatches. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, June 2020, pp 2814–2822
29. Mittal T, Bhattacharya U, Chandra R, Bera A, Manocha D (2020) Emotions don't lie: an audio-visual deepfake detection method using affective cues. In: Proceedings of the ACM international conference on multimedia, October 2020, Seattle, pp 2823–2832
30. Zadeh A, Liang PP, Mazumder N, Poria S, Cambria E, Morency L-P (2018) Memory fusion network for multi-view sequential learning. In: Proceedings of the AAAI conference on artificial intelligence
31. Schroff F, Kalenichenko D, Philbin J (2015) FaceNet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE computer vision and pattern recognition. Boston, pp 815–823
32. Güera D, Baireddy S, Bestagini P, Tubaro S, Delp EJ (2019) We need no pixels: Video manipulation detection using stream descriptors. In: Proceedings of the international conference on machine learning, synthetic-realities: deep learning for detecting audiovisual fakes workshop, June 2019, Long Beach