Project Report on

Deepfake Detection Based on Audio-Visual Mismatch

Submitted by

Reuben Suju Varghese	**B190670CS**
Rameez Naufal	**B190785CS**
Sanjana Reji Kallingal	**B190629CS**

Under the Guidance of

Lijiya A.

Department of Computer Science and Engineering
National Institute of Technology Calicut
Calicut, Kerala, India - 673 601

February 17, 2023

[illegible]
21/2/2023
Lijiya A

Deepfake Detection Based on Audio-Visual Mismatch

Reuben Suju Varghese Rameez Naufal Sanjana Reji Kallingal

Abstract: As public engagement with digital content continues to rise, we are increasingly reliant on technology platforms. However, the anonymity of the digital world has created an opening in the realm of deepfakes. We propose a novel approach for deepfake detection based on audio-visual mismatch. This strategy seeks to use discrepancies between a video's audio and visual elements to spot probable deepfakes. These variations can be explained by the disparity in mouth shapes and associated phonetic sounds in the audio. Our design can discern if a video is real or fake by calculating the percentage of phonetic mismatches using a mismatch estimator.

1 Introduction

You must have seen videos on the Internet of celebrities such as Barack Obama, Donald Trump, and others that may seem distorted or mismatched. These kinds of videos are known as deepFakes, and they pose a huge threat to our media security. The generation of such videos, especially of people in power, may cause a drastic shift toward chaos since it holds so much malicious power. Over the past few years, the technology to create such videos has become widely accessible, with apps generating realistic videos even with low computational power.

It is vital to understand the method with which malicious users create deepfakes. Creation methods fall under a few categories. First, face swapping is the process by which the face in the original video is swapped with the face of the target subject. Puppet mastery is when the expressions of the target subject are controlled by a source video. In lip-syncing, a method that falls in our domain of research, a video is generated with a consistent mouth region using audio recordings. Finally, audio deepfakes deal with the modification of the target subject's audio content.

However, most researchers only considered deepfake detection to be an image-based problem because most existing articles on the topic concentrate on image-based features collected from videos. Their primary operations are to first extract the frames of the fake video and then extract some visual features from the deepfake picture for further operations. These experimentation techniques just take into account the visual data, not all the elements from fake videos. Nevertheless, there are numerous audio features in the fake video that have been improperly utilized.

Therefore, here we introduce a revised approach towards deepfake detection based on audio-visual mismatch. The phoneme-vismeme model mainly extracts the audio and visual information from the video separately and further compares the information of two characteristics. The open/closed mouth labeling threshold and mismatch estimator threshold contribute to discerning if the video is real or fake. The main contributions of our work are summarized as follows:

- To extract the phonemes and visemes from the audio and video, respectively, we first use a number of libraries, including Speech Recognition, Librosa for feature extraction, TensorFlow for a pre-trained model on phonetic sounds, and P2FA for video-audio alignment.
- Our current workflow suggests using a lip-state detection algorithm to categorize visemes for related MBP phonemes as open or closed.

- Finally, we assess the input video's authenticity using the mismatch estimator and data on real and fraudulent videos from the datasets that are readily available.

2 Problem Statement

To design a model that can detect even subtle discrepancies between the mouth movements and speech content, indicating a phoneme-viseme mismatch, for a given video as input and ultimately, produce a binary output labeling the input as real or fake.

3 Literature Review

3.1 Detection Techniques

3.1.1 Facial Artifacts

Deepfake algorithms generate images of limited resolution with face warping, which leaves distinctive artifacts in the facial region. This limitation is what [1] exploits to detect fake images. [1] suggests that it is possible to generate negative examples containing facial artifacts for the convolutional neural network (CNN) to train on. The backbone of this methodology is a Neural Network trained on mapping the facial expressions of the source to the target subject. The facial region is detected and extracted to create negative examples on which the CNN classifier can train. This extracted portion of the image is then scaled and smoothed using Gaussian Blur. Finally, this smoothed face undergoes affine warping to produce the above-mentioned artifacts. Similar work in this area explores the use of Generative Adversarial Networks (GANs) [2] and Resampling Detection [3].

3.1.2 RNN

Recurrent Neural Networks (RNN) have had some success in the field of deepfake detection. [4] uses a CNN to extract frame-level features and these features are used to train the RNN to see if the video has been manipulated. The RNN we commonly use for this process is an LSTM (long short-term memory) network that is perfect for handling complete data streams such as speech and video. The set of features that are obtained by the CNN is passed onto the LSTM for analysis. Two autoencoders are trained separately but with shared weights to avoid incompatibility between them. Due to this, there will have to be two sets of training images; first of the source subject and second of the target subject. Note that deepfake video generation is also possible through this method by passing the face of the original subject through the decoder of the swapped subject. This process repeated for each frame will cause the decoder to reconstruct the face from the new subject.

3.1.3 Phoneme-Viseme Mismatch

Deepfake videos can range from slight temporal and spatial alterations to full-face forgeries. The former category includes lip-sync deepfakes, sometimes known as total mouth-audio synthesis. When aligning audio to video during deepfake creation, phoneme-viseme pairing is often overlooked.[5] outlines a forensic technique that exploits the inconsistencies of viseme with spoken phonemes, namely 'm', 'b', and 'p' to detect audio based and text-based lip-sync deep fakes. The position of phonemes (M, B, P) is extracted using Google's speech-to-text API to transcribe and P2FA to align the audio. Video frames surrounding the beginning of the occurrence of the MBP phoneme are chosen, and three approaches—broadly categorized as manual and automatic—are evaluated in order to establish the predicted closed mouth viseme. In the manual approach, using the reference frame from the same video, an analyst labels the sequence of frames as open/closed. In one of the automatic approaches, a vertical intensity profile is extracted from the middle of the mouth after extracting the lip region. While in the other approach, a CNN is trained to classify if a mouth is open or closed in a single video frame. Although the automatic techniques proved to provide detection accuracies above 96%, there were inevitable failures due to factors such as differences in lip shapes from the reference frame and asymmetrically open mouth.

The application of phoneme-viseme pairing goes beyond the MBP category thanks to related research in this field. [6] proposes a multi-phonemes selection-based framework for lip forgery detection tasks, which takes full advantage of the visual and aural information in lip forgery videos. All video frames are categorized using 12 phoneme classes, and for each phoneme-lip set, the difference between the real and fake lip forms' open-close amplitude is measured. A sub-model is then trained for the real/fake classification step. Finally, phonemes with the top 5 deviations are selected, and their models are integrated into an ensemble to maximize the discriminability of real/fake videos. Due to the shortcomings of the available public datasets, researchers additionally organized a new dataset in [6] that is specifically targeted at lip forgery detection. For each subdataset, different combinations of phoneme sets are used. XceptionNet is the baseline model trained on all frames combined with ResNet-50, which performs well in image classification.

3.1.4 GAN Discriminators

There have been numerous works in the field of deepfake creation using GAN (Generative Adversarial Networks). However, a surprising technique has been theorized by [7] that uses the power of GANs to detect deepfakes. With the use of MesoNet, a forgery detection technology, as the baseline, a GAN is trained, and the discriminator is extracted, which serves to distinguish deepfakes from the others. The generator and discriminator networks of the GAN play adversarial roles, i.e., the former creates the deepfake images and the latter tells them apart. Different models have been proposed where the discriminator has been trained on different datasets and the GAN in various orders. These methods have given an average classification score of 90%.

3.2 Prevention Techniques

3.2.1 Smart Watermarking

Luochen L. suggested a [8] watermarking approach by embedding unperceptive watermarks on the backgrounds of the images. These images, when processed by a deepfake creation tool, cause the synthesized images to be blurry and disrupted, especially on the facial areas. These could be easily perceived by human eyes or a machine with several metrics. The same could be achieved on videos by applying the same on multiple or selected frames in a video. The smart watermark proposed uses two modules: a watermark module to generate the unperceptive watermark and an attention module to guide the watermark generation along the facial area. The generation of the watermark along facial regions results in a better distortion than random placement due to the fact that deepfakes are primarily concerned with the manipulation of the facial regions. The added advantage comes with a lesser required watermark to produce, thereby reducing computational and generation time.

3.2.2 Residual Fingerprint

Deepfake detectors have done a good job of detecting deepfakes in the past. However, there now exists a type of attack that threatens the validity of the deepfake detectors, namely adversarial attacks [9]. The attack method adds subtle perturbations to normal images that force a classification model to give incorrect predictions. Hence, as a defense technique, [10] proposes Residual Fingerprinting. This method aims at maximizing the difference between real and fake images by transforming residual fingerprints with a Discrete Cosine Transform (DCT) Component. The impact of adversarial examples on a deepfake detector is analyzed. Based on this analysis, a reconstruction network is created to restore adversarial deepfakes into their pure versions. The evaluation suggests that the deepfake detector's performance has gone up from 50% to 84%.

3.2.3 Blockchain and Smart Contracts

Proof of authenticity systems were required for digital content to identify trusted valid authors of the data, especially for media that could be deepfaked like audio, video, and images. Haya R. H. et al in [11] proposed the concept of using blockchain technology to maintain the authenticity of the videos created in a decentralized manner. It provides transparency, traceability, and history tracking, which involves tracking and tracing that may involve multiple versions. The blockchain they proposed uses Ethereum smart contracts and IPFS for storage. The calculated gas cost was also very minimal and negligible. It was tested and proved resilient against attacks like a man-in-the-middle attack, replay, and DDoS attacks. The same can be applied to other hypermedia like audio, images, or manuscripts.

4 Project Design

After extensive research in the field of deepfake detection based on audio-visual mismatch, the following design was proposed to detect lip-sync deepfakes based on the MBP closed-mouth viseme to phoneme mismatch. There are two types of flows, the training flow and the prediction flow, as shown in figure 1. The training flow sheds light on the preprocessing steps taken before the training data is input to the CNN model to identify closed-mouth viseme sequences. On the other hand, the prediction flow gives insight into the pre-processing steps taken before the trained model, and the mismatch estimator can determine if a video is real or a deepfake.The motivation for this design stems from [5] that proposed a model to detect the MBP phoneme-viseme mismatch.

4.1 Training Flow

4.1.1 Datasets

The dataset comprises video recordings of individuals processed to detect and calculate the landmark distances of the mouth region.Since the landmark distances of lips differ from person to person, CNN is thought to deliver superior accuracy for person-specific datasets; hence, the aforementioned data types are acquired from Google and Facebook datasets.

4.1.2 Video to Image Sequence

The video recordings contained in the initial dataset are split into frame sequences using python OpenCV.

4.1.3 Pre-processing stage

If an image is aligned, the training of the model will be more successful because the center of the face will be equal for all images in a dataset.[12] Studies show that many facial recognition algorithms can benefit from facial alignment before trying to detect the face.This step can be drawn parallel to data normalization as it includes normalized rotation, translation, and scale representation of the face. Using tools such as DLIB[13], Mediapipe[14], and python OpenCV, etc faces in each of the frames are aligned, cropped, and rescaled.

4.1.4 Lip State Detection model

As in [15], We simplify the lips-state model with a set of six key landmarks and use their distances for the lips-state classifications, building on two well-known lips landmark detectors, DLIB and MediaPipe. After computing the mouth aspect ratio [16], a temporary threshold value of 0.65 is set. The viseme is labeled as open if the mar value is higher than the threshold value.The neural network receives this data and trains on it for a number of epochs.

4.1.5 CNN

The proposed model utilizes a typical and very lightweight dense neural net with one dense layer of 32 units and Rectified Linear Unit (RELU) activation, as depicted in Fig 1. For a compilation of the model, the optimizer is set to Adam, and the loss is binary cross-entropy. The CNN is trained on the landmark distances and corresponding labels. Hypothetically, higher distance values should correspond to mouth open positions, whereas lower distance values correspond to closed mouth positions.

4.2 Prediction Flow

4.2.1 Phoneme-Viseme Extraction

The timestamps of the phonemes are extracted using TensorFlow deep learning models, speech recognition, librosa libraries etc. Following this, the visemes are extracted by selecting 6 frames around the start

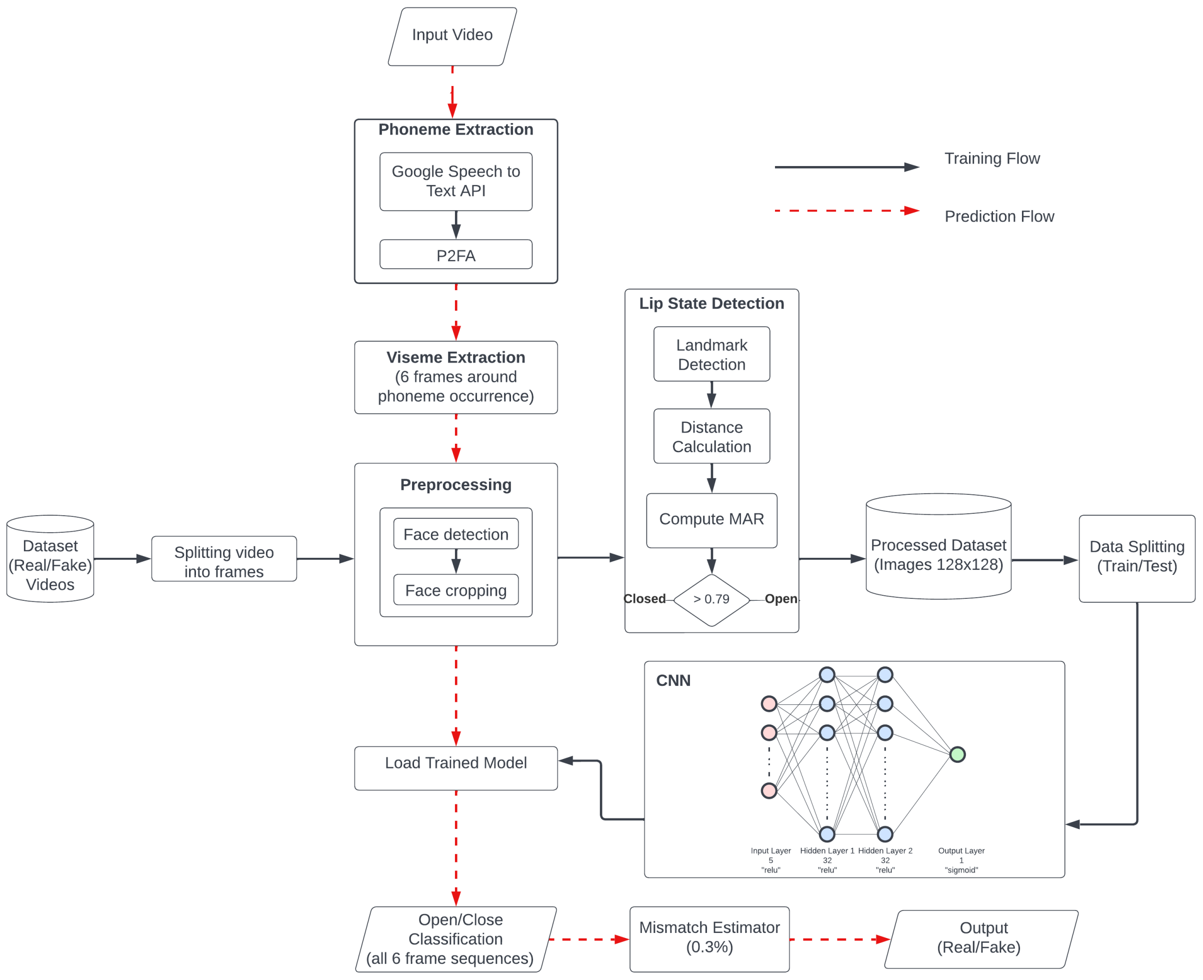

Figure 1: Proposed Design for phoneme-viseme mismatch detection

of the occurrence of the MBP phoneme. These frames can be obtained using python OpenCV.

4.2.2 Frames pre-processing

The pre-processing stage is the same as that of the training flow.

4.2.3 Load Trained Model

Past the pre-processing stage, the closed-mouth viseme sequences are passed to the trained model to label the sequence as open or closed.The model labels each frame in the 6-frame sequence as open (0) or closed (1) and takes the maximum value for the sequence as (0) or (1).The labels of all 6 frame sequences are thereafter passed to the mismatch estimator.

4.2.4 Mismatch Estimator

The mismatch estimator calculates the total number of closed-mouth MBP visemes that were labeled as a closed sequence and discerns if the original video was fake or real.Theoretically, the threshold fixed for the model is at 0.3%, which indicates that up to 0.3% mismatch can still classify the video as original, beyond which the model detects it as fake.This percentage can vary during the implementation of the model.

5 Work Done

Semester 7: In the phoneme-viseme detection methodology, we found that there exist many research gaps in the model. Therefore, we have proposed a design that rectifies all gaps by proposing more efficient methodologies in place of some components. The base model is designed around the phoneme-viseme detection model on M, B, and P phonemes [5]. Manual labeling of frames in the preprocessing stage has been replaced with lip state detection preprocessing [15] and mouth aspect ratio from an available code base [16]. The CNN model (XceptionNet) has also been replaced due to the relatively small size of the dataset that will be used. The neural architecture has been modified to contain just 2 hidden relu layers to improve the overall efficiency of the model.

Semester 8: Work done in the respective submodules for each flow has been documented below.

5.1 Training Flow

5.1.1 Video to Image Sequence

Since the training flow will take as input the videos from the dataset, we have to split this video into frame sequences for our requirements. For this purpose, we use the OpenCV library to capture the frames and store them separately on successful reading of frames.

We have also explored the MediaPipe library to detect and draw hand landmarks on the frames of the video. The module then draws the identified hand landmarks onto the processed frames, as shown in Fig 2. We will be able to extract the mouth landmark information using a similar method for a different module.

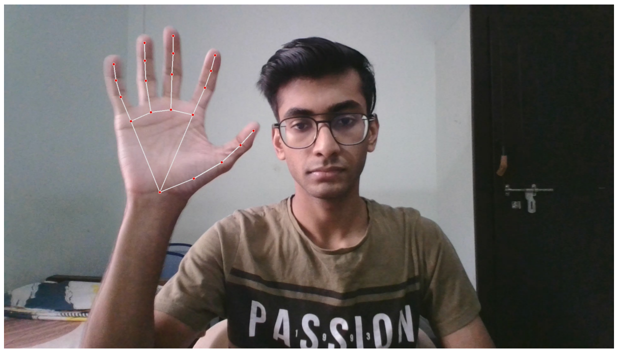

Figure 2: Detecting hand landmarks on the frame

5.1.2 Pre-processing: Face Cropping and Alignment

It is vital to perform pre-processing on the input frames before passing it to the Lip State Detector to increase the accuracy. Our pre-processing module implements face cropping and alignment for better results. The code uses the dlib library to detect the facial region and resizes the cropped image to a width of 800 pixels. Furthermore, it performs a facial alignment in case the subject's head is tilted to give maximum accuracy from the Lip State detection module. This facial alignment is done by setting the paths to the shape predictor model file to the imultis library, as shown in Fig 3. This is all done to obtain a better face recognition performance.

5.1.3 Lip State Detection

Since pre-grouping of datasets is essential before passing it onto the CNN to train. We implemented a Python script to detect closed lip state and log the result into a file. The script uses the Dlib library to detect facial landmarks and compute the mouth aspect ratio (MAR), which is used as an indicator to check for closed and opened mouth positions. The threshold for MAR is set to be 0.65. If the threshold goes below 0.65, the state of the mouth is considered

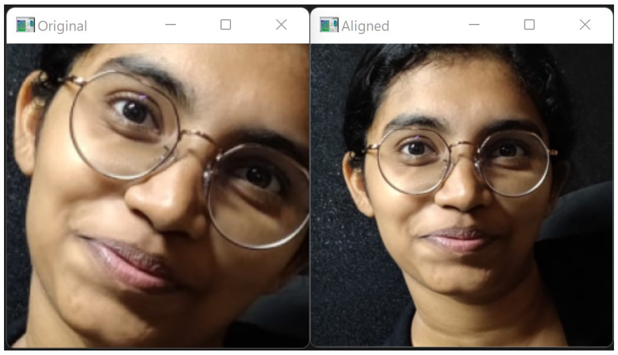

Figure 3: Face Alignment and Cropping

closed or vice versa The corresponding values and timestamps of those closed mouth frames are outputted onto the text files. The threshold is likely to be updated in the future based on the analysis and outcome of the model on various datasets. Additionally, the code will be modified in the future to take as input a collection of frames cropped around the mouth and output the frames along with the labels, compiling the processed dataset to feed to the CNN.

The program was initially made to overlay the lip-detected boundaries on top of the video and the computed mouth aspect ratio value on the top left, which is then saved as a different file in the output directory instead of the newly added text file with a timestamp and its corresponding aspect ratio values. This was done for better analysis and debugging purposes of the code.

In the main loop, the script reads frames from the video stream, detects faces using the face detector, and computes the mouth aspect ratio for each face. The script uses OpenCV to display the frames and to capture keyboard events and stops when the 'q' key is pressed. Overall, this script provides a simple and effective way to detect the state of the lips in a person's face from a file using the mouth aspect ratio as an indicator.

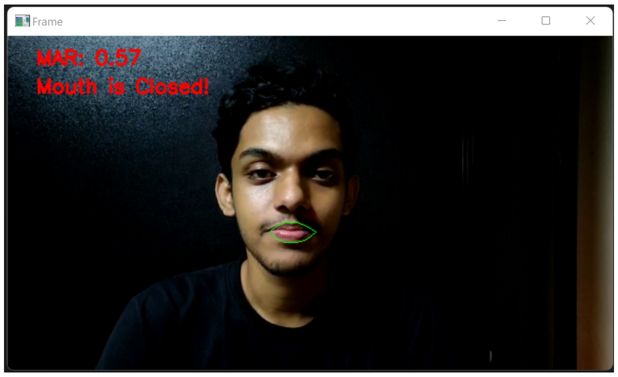

Figure 4: Detecting closed mouth viseme using MAR

5.2 Prediction Flow

5.2.1 Phoneme Extraction

The foundation of the design relies on the precise detection of the aforementioned phonemes from an audio file with accuracy in the millisecond range. To achieve this, we inspected numerous speech recognition models for the extraction of the closed-mouth phonemes ('M', 'B', 'P'). A common approach was to transcribe the speech into a sequence of phonemes. The phoneme extraction module works on the wav file, which is obtained from the input video using the moviepy library. Through the use of different python libraries and tensorflow, we explored different models and toolkits as seen below:

speech_recognition At first, we attempted to transcribe the audio file into text using the SpeechRecognition library. The input audio file was first recorded, stored, and given as input to the google speech recognizer included in the SpeechRecognition library. By splitting the transcribed text and assuming a sample rate of 11025 Hz, we calculated the timestamp of the occurrence of the closed-mouth phonemes. However, we were only able to collect the start and end timestamps of the word that contained the occurrence of a closed-mouth phoneme. The accuracy obtained through this method was far too low for our use, and hence we discarded it.

vosk/kaldi toolkit To perform speech recognition on the wav file, we instantiated a Vosk model and created a KaldiRecognizer object. Feeding the audio data in chunks to the recognizer, we collected the json dictionaries containing the transcribed words, the start and end timestamp, and the confidence of transcription. Our plan was to use these timestamps to approximate the position of the closed-mouth phoneme in the word and find its corresponding timestamp. Unfortunately, the vosk library is known to be less accurate than standard speech recognition modules (such as google speech recognizer), and this was seen in our implementation of the toolkit, due to which we rejected it.

deepspeech From toolkits and libraries, we decided to move towards neural architecture and landed on deepspeech by Mozilla. Similar to previous approaches, we fed the audio file, obtained the transcription, and approximated the timestamp using the position of the closed-mouth phoneme. However, the deepspeech model showed a few drawbacks. It took an immense time to process even small files, which made it impossible to use. The model also had to be downloaded on the local machine, which decreases our ability to work with the latest version while consuming hardware resources.

praat/parselmouth In order to extract phonemes from the audio file, We used the parselmouth library to access the underlying praat software. A sound object was made to deal with the audio file. Which contained a method to extract the pitch details from the file. All of this had to be placed in a python textgrid in order to iterate and display the time period along with the occurrences of these phonemes.

But due to the latest versions of Parselmouth not having a textgrid module inbuilt in it due to security reasons. We had to explore the tgt module in python which gave us an alternative to work around it. This also included a few lines scripted in a .praat file which had to be executed externally by calling it inside the python main module.

pocketsphinx For speech recognition, we also explored the pocketsphinx module. This module was really helpful in teaching us about the formation and structure of various dictionaries, language and acoustic models. This was once a research project from Carnegie Mellon university that has been open-sourced now to be used by the public.

We used the default dictionary and language model provided inbuilt by the library. For the acoustic model, we had to import it externally from the internet since that was more robust than the one already available. Few alterations in the pitch clipping frequencies, phoneme confidence intervals had to be made. At the end we were able to detect phonemes with its frame number with a confidence level of 82%.

wav2vec2 The state-of-the-art model for automatic speech recognition, wav2vec2, is one of our most successful attempts at phoneme extraction. This attempt was fruitful since different versions of the model can be loaded over internet connectivity to suit our requirements. The model works at an impressive speed at speech transcription and displays a much higher recognition accuracy. From there, we were able to accurately parse out the timestamps of the various closed-mouth phonemes with a millisecond error rate. We were happy with these results and decided to move in this direction.

5.2.2 Viseme Extraction

The visemes pertaining to the phoneme timestamps were retrieved using the OpenCV machine vision toolkit. The algorithm was initially written to extract a single frame from a video corresponding to that timestamp in seconds as input.

The code was subsequently modified to include timestamps in seconds and milliseconds and to read all the timestamps from a text file. This is based on the presumption that the previous module outputs a text file containing the timestamps of the necessary phonemes.

As per the design requirements, six frames must be extracted around each phoneme occurrence. Hence, the said number of frames was extracted for each timestamp, taking the edge cases into consideration. The edge cases covered include the occurrence of the frame pertaining to the phoneme timestamp at the very beginning of or at the end of the audio file. In such cases, unlike extracting three frames before and after the timestamp, the algorithm extracts 6 frames after or before the timestamp respectively.

The final output, the six-frame sequence for each occurrence, is stored in an output folder as jpg files.

5.2.3 Mismatch Estimator

To finally determine whether the video is a fake or not, we calculated the mismatch of the close-mouth phonemes. Calculating this mismatch required the input of classification data pertaining to 6-frame sequences and the identification of instances where closed mouth lip state was absent. If the computed mismatch value was found to exceed a critical threshold, we classified the video as a deepfake.

6 Work Plan

After achieving the primary objective of our design without the CNN, our effort would now concentrate on a number of approaches and modules, including

- Compiling the aforementioned datasets to feed a neural network.
- Using a neural network, such as the CNN mentioned earlier, to attain greater accuracy.
- Employ toolkits like HTK in an attempt to boost speech recognition's precision in extracting phoneme timestamps.
- Incorporate global audio-to-video alignment to ensure the two are perfectly synched.This will also boost the accuracy of the overall model, especially by increasing the precision of the phoneme timestamps in microseconds.
- Add future design alterations based on an empirical approach to improve the overall accuracy, such as grayscale video conversion and audio signal pre-processing.
- Automate the procedure to link the modules and output a single binary value of real or fake for a specific video input.

7 Conclusion

To sum up, we analyzed a variety of popular detection and prevention strategies in this research, as well as proposed a design for identifying lips-sync deepfakes based on phoneme-viseme mismatch. One of the strengths of the proposed approach is its ability to detect deepfakes even when the video quality is high, and the facial expressions are well-simulated. This is because the model focuses on phoneme-viseme alignment, a key aspect of natural speech production. It can be utilized in many different contexts, including forensics, news and entertainment, and the detection of deepfake videos on social media.

We have carefully designed our suggested approach for the detection of deepfakes in an attempt to offer a model with good theoretical performance. We posit that feeding the lip-state detector's labeling for the compiled dataset into the CNN would strengthen the detection model and provide us the freedom to choose the training variables. Our proposed work plan hereafter also incorporates modifications to the pre-processing module, such as experimenting with gray-scale conversions and preprocessing audio signals. These changes would guarantee a higher accuracy than what is obtained at present.

References

[1] Li, Yuezun & Lyu, Siwei. (2018). Exposing Deepfake Videos By Detecting Face Warping Artifacts.

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In NIPS, 2014.

[3] Nahuel Dalgaard, Carlos Mosquera, and Fernando P ́erezGonz ́alez. On the role of differentiation for resampling detection. In Image Processing (ICIP), 2010 17th IEEE International Conference on. IEEE, 2010.

[4] D. Güera and E. J. Delp, "Deepfake Video Detection Using Recurrent Neural Networks," 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 2018, pp. 1-6, doi: 10.1109/AVSS.2018.8639163.

[5] S. Agarwal, H. Farid, O. Fried and M. Agrawala, "Detecting Deep-Fake Videos from Phoneme-Viseme Mismatches," 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2020, pp. 2814-2822, doi: 10.1109/CVPRW50498.2020.00338.

[6] Lin, Jiaying, Wenbo Zhou, Honggu Liu, Hang Zhou, Weiming Zhang, and Nenghai Yu. "Lip Forgery Video Detection via Multi-phoneme Selection." (2021).

[7] S. A. Aduwala, M. Arigala, S. Desai, H. J. Quan and M. Eirinaki, "Deepfake Detection using GAN Discriminators," 2021 IEEE Seventh International Conference on Big Data Computing Service and Applications (BigDataService), 2021, pp. 69-77, doi: 10.1109/BigDataService52369.2021.00014.

[8] L. Lv, " Smart Watermark to Defend against Deepfake Image Manipulation", in IEEE the 6th International Conference on Computer and Communication Systems, April 2021.

[9] A. Gandhi and S. Jain, "Adversarial perturbations fool deepfake detectors," in 2020 International Joint Conference on Neural Networks (IJCNN). IEEE, 2020, pp. 1–8.

[10] J. Jiang et al., "A Residual Fingerprint-Based Defense Against Adversarial Deepfakes," 2021 IEEE 23rd Int Conf on High Performance Computing & Communications; 7th Int Conf on Data Science & Systems; 19th Int Conf on Smart City; 7th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys), 2021, pp. 797-804, doi: 10.1109/HPCC-DSS-SmartCity-DependSys53884.2021.00129.

[11] H. R. Hasan, K. Salah, "Combating Deepfake Videos Using Blockchain and Smart Contracts", IEEE Access, vol. 7, pp. 41596-41606, March 2019.

[12] https://pyimagesearch.com/2017/05/22/face-alignment-with-opencv-and-python/

[13] King, D.E.: Dlib-ml: A machine learning toolkit. Journal of Machine Learning Research 10, 1755–1758 (2009)

[14]]Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C.L., Yong, M.G., Lee, J., et al.: Mediapipe: A framework for building perception pipelines. arXiv preprint arXiv:1906.08172 (2019)

[15] arXiv:2112.04752v2 Modelling Lip-state detection using CNN for non verbal communication.

[16] https://github.com/mauckc/mouth-open/blob/master/detect_open_mouth.py

[17] Masood, M., Nawaz, M., Malik, K.M. et al. Deepfakes generation and detection: state-of-the-art, open challenges, countermeasures, and way forward. Appl Intell (2022). https://doi.org/10.1007/s10489-022-03766-z