International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 05 | May 2021 www.irjet.net p-ISSN: 2395-0072

Forensic Tool for Deepfake Detection and Profile Analysis

Nirnay Khajuria[1], Rajas Chavadekar[2], Aditya Dhote[3], Sanyog Chavhan[4]

[1,2,3,4]*Dept. of Computer Engineering, Sinhgad College of Engineering (SPPU), Pune, India.*

Abstract - *To analyze the genuineness and quality of a person giving a speech or an interview while proctoring. Detect any forgery made by deepfake techniques using deep learning. Create a behavioral profile and analysis of that person which can be used for judging their peculiarity. In order to execute this, we have created four modules namely Face Detection Module, Sound Expression Module, Deepfake Detection Module and an Aggregator Module which combines the result of all the three given modules into one and provides output to the user in the form of a profile. The basic idea of our project is to present a forensic tool that will provide Deepfake, Audio and Video expression analysis in a single package. The system will determine whether the Candidate is giving a genuine interview or not. There would be no discrimination during the interview process, this system will remove all of the biases.*

Key Words: **deepfake, cyber-forensics, deep-learning, image processing, audio processing.**

1.INTRODUCTION

With increase in deep learning technology, we have seen advancements in video editing and forgery but there is also positive development in analysing one's interview performance. In current crisis, work from home culture is increasing, people are not meeting face to face which gives them chances for making forgery or deepfaking (identity tampering) in their video stream. In this scenario, recruiters might anticipate for a single application or service that will do the behavioural analysis of a candidate along with forgery detection.

In order to analyse the genuineness and quality of a person giving a speech or an interview while proctoring. Detect any forgery made by deepfake techniques using deep learning. Create a behavioural profile and analysis of that person which can be used for judging their peculiarity. The basic idea of our project is to present a forensic tool that will provide Deepfake, Audio and Video expression analysis in a single package. The system will determine whether the Candidate is giving a genuine interview or not.

The System provides Features as follows:

1) Deepfake Detection
2) Face Expression Analyzer
3) Sound Expression Analyzer
4) Complete Profile Analyzer

The Deepfake Detection module includes first of all gathering of data, then deepfake data training will be done, followed by Deepfake Detection. Whereas, in the Face Expression Analyzer, data gathering will be done firstly followed by training the model and then predicting the face expression according to the given user input to the system.

In the Sound Expression Analyzer Module, first of all data gathering will be done, then it will be trained so that it can set an expression mode for the sound sample. Following this procedure, a feature data will be created for that sound sample. The input audio frame will be directly provided to the sound expression analyser. This audio will be compared with the trained sound sample given from sound expression trainer and then predictions will be made for the input audio sample.

The Final prediction of profile will be given as output in a SRT file.

2. MATHEMATICAL MODEL

In this section we will construct a mathematical model that describes the functions of each of our modules in terms of set relations, input output tuples and domain to which they belong. We will discuss these domains first and then after construct their mapping.

2.1 Input Domain

We have an Audio Video screencapture input so input varies with time. Consider that is input stream, of two time-varying tensors,

$$ip(t) = \big(A(t), V(t)\big), ip \in AV\ stream$$

Where, $A(t) = [b_i]_{n\times 1}(t)$ is the audio stream,

and $V(t) = \left[b_{ijk}\right]_{w\times h\times 3}(t)$ is the r-g-b video stream.

Where $b_{ijk} \in \{0,1\}^8\ \forall\ i,j,k\ \in \mathbb{N}$ are bytes.

2.2 Output Domain

The output is also a stream of tuples which belong to following output domain,

The Output is composed of following entries:

1) t : time :time at which a reading was taken w.r.t video.
2) df : Boolean : 1 if deepfake frame else 0.
3) gs : float : genuineness score between 0 to 1.
4) pe : emotion ID : prominent expression found at time t.

In set theory,

$$t \in T\ (time), df \in \{0,1\}, gs \in [0,1], pe \in E,$$

$$where\ E = \{'angry', 'disgust', 'fearful', 'happy',$$

$$'neutral', 'sad', 'surprised'\}$$

Let op be output tuple and O is output domain,

$$O = T \times \{0,1\} \times [0,1] \times E$$

Hence op is 4-tuple as follows,

$$op(t) = (t, df(t), gs(t), pe(t)) \in O$$

2.3 Functions or Modules

Our project or application can be viewed as a function F that takes in input ip and gives output op.

$$op(t) = F(ip(t))$$

Where mapping $F : I \rightarrow O$

2.3.1: Module 1: DeepFake Detection

DeepFake detection can be defined as a function that takes a video frame at time t and gives binary result 1 if deepfake else 0.

$$df(t) = DF(V(t))$$

Where mapping $DF: V \rightarrow \{0,1\}$

Furthermore, DF is composed of 2 functions:

1) $FC: V \rightarrow V$: Face detection, grayscale conversion and Cropping
2) $DFD: V \rightarrow \{0,1\}$: DeepFake Detection

$$DF = (DFD\ o\ FC)$$

2.3.2: Module 2: Face Expression Detection:

Face Expression detection can be defined as a function that takes a video frame at time t and gives one of the expression IDs in E.

$$fe(t) = FE(V(t))$$

Where mapping $FE: V \rightarrow E$

Furthermore, FE is composed of 2 functions:

1) $FC: V \rightarrow V$: Face detection, grayscale conversion and Cropping
2) $FED: V \rightarrow E$: Face Expression Detection

$$FE = (FED\ o\ FC)$$

2.3.3: Module 3: Sound Expression Detection:

Sound Expression detection can be defined as a function that takes an audio frame at time t and gives one of the expression IDs in E.

$$se(t) = SE(A(t))$$

Where mapping $SE: A \rightarrow E$

Furthermore, SE is composed of 2 functions:

1) $MFCC: A \rightarrow F$: Mel-Frequency Cepstral Coefficients features extraction where F is features set.
2) $SED: F \rightarrow E$: Sound Expression Detection.

$$SE = (SED\ o\ MFCC)$$

2.3.4: Module 4: Aggregator:

Aggregator is a function that takes in deepfake reading df, face expression ID 'fe', and sound expression ID 'se'. Let's say that we define AGG as aggregator function then,

$$op(t) = AGG(df(t), fe(t), se(t))$$

Its mapping is $AGG: \{0,1\} \times E \times E \rightarrow O$

Aggregator has a function to calculate the genuineness score, GS which takes two arguments,

1) fea : array : array of face expressions for t = t - 5 to t = t
2) se : E : sound expression generated at t = t recorded every 5 sec.

$$fea(t) = \{fe(t)\ \forall\ t = t - 5\ to\ t\}$$

Genuineness score can be defined as:

$$GS(fea, se) = \frac{|\{se\} \cap fea|}{|fea|}$$

Further the Aggregator function can be defined as,

$$op(t) = AGG\big(df(t), fe(t), se(t)\big)$$

$$= (t, df(t), GS(fea(t), se(t)), se(t))$$

Here prominent expression $pe(t) = se(t)$.

Hence the final project function F can be defined as,

$$op(t) = F(ip(t))$$

$$= AGG(DF(ip(t).V), FE(ip(t).V), SE(ip(t).A))$$

Which completes our mathematical model.

3. FEASIBILITY ANALYSIS

This subsection discusses on how the algorithms or structures used in our modules are feasible and can be implemented mathematically. Here we will analyze the computational modules we will be using in our implementation with respect to their space and time complexity whether it is P or NP complete.

For this we need to first understand the fundamental unit of computation we will be using in all of our modules involved in audio-video processing, the Convolutional Neural Network which is in turn based on Neural Networks. So, any inferences we make on neural networks will implicitly apply on Convolutional Neural networks and in turn on the higher order models used. And thus, on the architecture we will be using.

The Neural Network has two main functionalities, forward propagation and backward propagation. Every operation in neural network is a matrix multiplication, addition or function mapping out of which addition and mapping of m by n matrix has time complexity *O(mn)* and multiplication of m by n and n by p matrix has *O(mnp)* when applied one element at a time also these operations will produce a matrix which has finite space complexity i.e.,$O(n^2)$. As Neural network is composed of matrices and its operations, a Neural network is P complete and hence can also be effectively further reduced in terms of time by using parallel computation.

We are using pytorch library to achieve this which has functionalities of creating matrices and tensors which are parallelly computed in dedicated Graphic Processing Units (GPUs). These processors compute all rows and all columns at constant time so any operation for matrix which was taking *O(mn)* time will take *O(1)* time with same space complexity. So, addition and mapping will take *O(1)* time and multiplication will take *O(n)* time, which is also of polynomial order. It turns out that Neural Networks, hence convolutional Neural Networks and **thus our models are P-complete.**

From above statements we conclude that our architecture is feasible with respect to space and time mathematically.

4. ARCHITECTURE

The architecture of forensic tool for deepfake detection and profile analysis is an integrated architecture of Deepfake Detector, Face Expression Analyzer, Sound Emotion Classifier and Aggregator.

The architecture of our forensic tool is shown in Figure1.

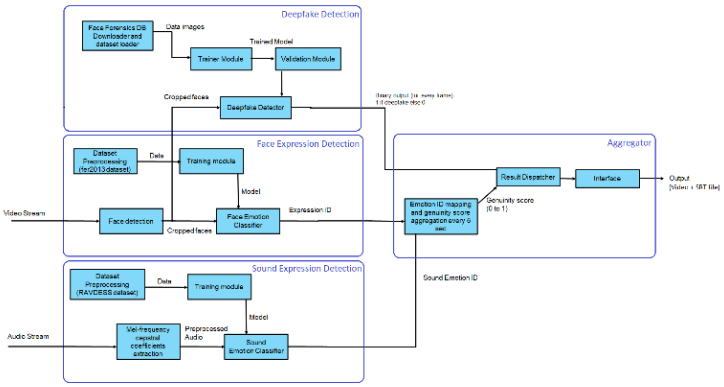

Fig 1: The architecture of the forensic tool

In Deepfake Detection module the image data is taken from Face Forensics DB downloader and dataset loader and is given to Trainer Module. The Validation Module takes this and passes it to the Deepfake Detector. The cropped faces data is given as input to Deepfake Detector which compares them with the trained data and generates binary output 1 or 0.

In Face Expression Detection module, the cropped face data from the fer2013 dataset and is given to Face Expression Classifier, simultaneously the captured screenshot of the candidate is given to Face Expression Classifier as input which then will compare these screenshots with the training module data and generate ExpressionID accordingly.

In Sound Expression Detection the sound data is taken from RAVDESS dataset and given to Training module. Trained model is given as input to Sound Emotion Classifier, and simultaneously audio stream of candidate is given as input to it. Sound Emotion Classifier will generate SoundExpressionID by comparing audio stream with trained data.

The Aggregator module takes input from all three modules, it will take both ExpressionID and SoundEmotionID and map them every 5 seconds and will send binary output to Result Dispatcher. Binary output from Deepfake Detector is given to the Result Dispatcher.

The Result Dispatcher compares both binary values and generates a final score which will be given to the Interface. The Interface will generate the output in an SRT and mp4 file.

5. IMPLEMENTATION AND TOOL INFRASTRUCTURE

5.1 Getting Input, Audio Processing and Aggregator

The video input is provided by the screencapture by 'mss' python module. It captures the entire screen available. This enables to capture video even from online interviews or meetings as we may not have direct access to capture the video stream from browser or video conferencing application. Every frame is captured and written in the shared memory on the go. Coming to audio input, the processing of audio stream takes place in entirely different process. The sound expression module then communicates its result (expression enum) to the aggregator via shared memory in async. Every time after sound module communicates the genuineness score, prominent expression and deepfake percentage is calculated (aggregated) and its subtitle is appended.

5.2 Dumping Output

As specified above the output is in the video and subtitles file (written by 'srt' python module) which gives the deepfake percentage, genuineness score and prominent expression for approximately every 5 seconds. The main reason for such output format is the real time playback of the recorded video and associated output.

5.3 The Role of Shared Memory

It is observed that the face and deepfake modules do not process video frames in real time, i.e., while processing one frame they may drop some frames for processing after which they will give their output. In order to enable real time analysis, we need to keep operation of video capturing and deepfake, face expression in async manner in entirely different processes. The videocapture (from aggregator.py) continuously writes the video stream in allocated shared memory and the deepfake and face detector module can get it on demand according to the pace of their processing.

However, after processing they need to communicate their results back to the aggregator so here, we use another shared memory where these modules will share their enumerations to the aggregator. The shared memory utility is provided by the 'SharedArray' python module.

5.4 The Process Launch Sequence

The diagram below illustrates the launching sequence of modules in their processes. The hatched region represents that the module and its libraries are in loading state.

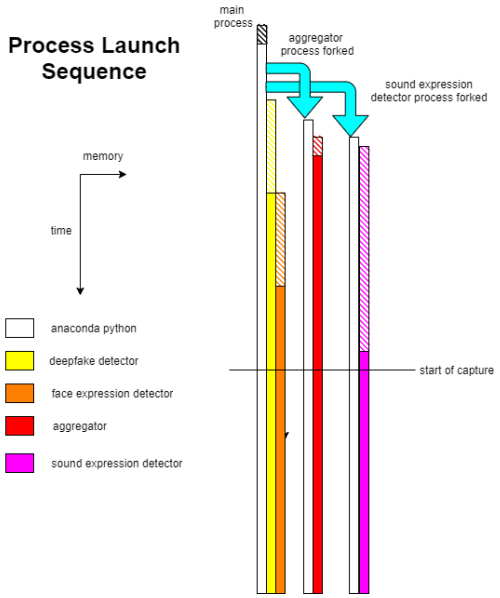

Fig 2: Process Launch Sequence Diagram

In order to capture and process the incoming audio and video streams, there is a need to execute their process in parallel in async manner and then collect their periodic results into a parallelly running aggregator. The operating system provides two kernel APIs to execute tasks in parallel i.e., forking a new process (fork syscall in posix) or creating a new thread within a same process (clone syscall in posix). They have their own limitations so; we will choose what is optimal for our project. For us memory is a major concern as every module has its own deep learning framework that consumes memory that may cause overflows if launched in other thread as threads are meant for low resource consuming codes. Moreover, the call stack of threads is limited and may not accommodate the calls of the deep learning framework. This is the reason we launch sound expression module separately in different process as it is totally independent task with different input device. However, we observed that the deepfake detection and face expression needed the same cropped face for processing so we have kept them in the same process (main process). The videocapture and collection, aggregation of results in aggregator has to be done in parallel in async hence we launch it in an entirely new process.

Now focusing on the loading time of modules, the sound expression detector takes more time for loading as it has to initialize the pulseaudio microphone device plus its deep learning framework (tensorflow) whereas the face expression and deepfake detector loads in reasonable time and aggregator instantaneously loads and halts the videocapture until the most lately loading module i.e., the sound expression detector module loads completely and start capturing microphone. This communication happens via the shared memory allocated for sharing expression enums that if there is any non-negative sound expression enum at that given time.

5.5 The Process Termination Sequence

The diagram below illustrates the termination sequence of modules in their processes. The hatched region represents that the module and its libraries are in de-loading state.

In order to terminate the ongoing application, we need to understand that we have two child processes running viz. the aggregator and sound expression detector out of which there is huge post processing in aggregator module in result dispatching (saving srt file and recorded mp4 video) and audio video synchronization of video for real time playback. We need to safely interrupt the child processes to break their processing loops, let them do their respective post processing, terminate and report to their main (parent) process which is in the waiting state to let children complete their tasks.

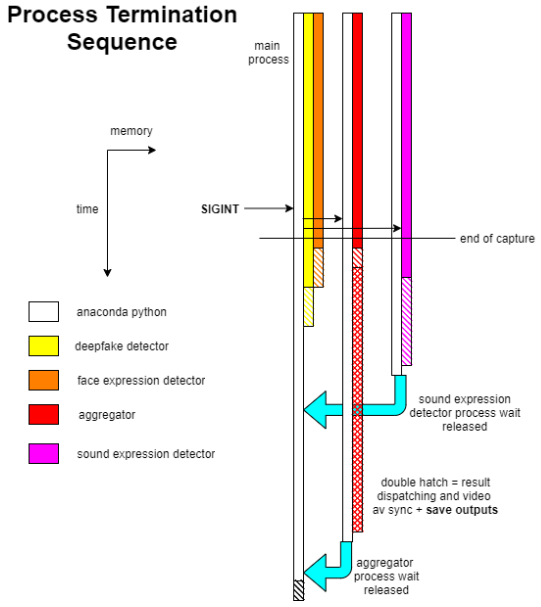

Fig 3: Process Termination Sequence Diagram

Whenever the main process gets a SIGINT (Interrupt signal) it interprets that we have to stop the application. The video loop (that analyzes deepfake and face expression) gets terminated and the sound expression and aggregator processes are explicitly sent (kill syscall) SIGINT so they stop their respective loops and begin post processing. The associated modules and libraries of main process are de-loaded and resources are released in parallel.

However, in aggregator there is huge post processing. Firstly, to save the recorded video and dump the 'srt' file and secondly, to synchronize the video, its srt and audio for real time playback. The algorithm of audio video synchronization as a part of result dispatcher is listed below.

```
algorithm av_sync(video_file, actual_playback_time):
    get video_file duration in video_duration using ffmpeg
    change_ratio := actual_playback_time / video_duration
    scale video_file change_ratio times using ffmpeg
    concat wavfiles sequentially into audio.wav using sox
    get output_video_file by merging video_file and audio.wav using ffmpeg
    return output_video_file
```

6. CONCLUSIONS

We have introduced a new Forensic Analysis Tool along with Behavioral Analysis Features such as Face and Sound Emotion and Detection in a single application that will do this all.

Although, there are tools in market for vocal and facial expression analysis, we have provided Deepfake Detection with the above tools in a single package. However, this implementation is at POC (Proof of Concept) level, one might extend its features to Body Posture Detection, Lie Detection, etc.

We have developed a module named Aggregator which takes the output of all the three modules and provides a detailed analyzed profile of the candidate. Hence, we have successfully analyzed the profile of a person based on online interaction via videos with its Forensic validity.

REFERENCES

[1] Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, Matthias Nießner, "FaceForensics++: Learning to detect Manipulated Facial Images", New York, United States, arXiv.org, 2019.

[2] Rohit Pathar; Abhishek Adivarekar;Arti Mishra; Anushree Deshmukh, "Human Emotion Recognition using Convolutional Neural Network in Real Time", Chennai, India, ICIICT, 2019.

[3] Marco Giuseppe de Pinto; Marco Polignano; Pasquale Lops; Giovanni Semeraro, "Emotions Understanding Model from Spoken Language using Deep Neural Networks and Mel-Frequency Cepstral Coefficients", Bari, Italy, EAIS, 2020.

[4] Francois Chollet, "Xception: Deep Learning with Depthwise Seperable Convolutions", New York, United States, arXiv.org, 2017.

[5] Bin Zhang; Changqin Quan; Fuji Ren, "Study on CNN in the Recognition of Emotion in Audio and Images", ICIS, Okayama, Japan, 2016.

[6] Ali Aliev, Avatarify-Python. Computer software, 18 May, 2020.

[7] S. Davis, P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences", IEEE, New Heaven, CT, USA, 1980.

[8] Beth Logan, "Mel Frequency Cepstral Coefficients for Music Modeling", ResearchGate, Washington, D.C., DC, United States, November, 2000.

[9] P. Viola, M. Jones, "Rapid object detection using a boosted cascade of simple features", IEEE, Kauai, HI, USA, December. 2001.

[10] Darius Afchar, Vincent Nozick, Junichi Yamagishi, Isao Echizen, "MesoNet: a Compact Facial Video Forgery Detection Network", IEEE, Hong Kong, China, January 2019.