

Software Engineering Department

Capstone Project Phase A

**Manipulated Reality**

**Video and Audio Analysis using Deep Learning for Deepfake Detection.**

<https://github.com/MaximL98/Manipulated-Reality>

**Supervisor: Dr. Reuven Cohen**

|  |  |  |
| --- | --- | --- |
| **Name** | Maxim Lebedinsky | Dmytri Kislitsyn |
| **ID** | 318171485 | 323349738 |
| **E-mail** | Maxim.Lebedinsky@e.braude.ac.il | Dmytri.Kislitsyn@e.braude.ac.il |

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Index** | **Subject** | **Page number** |
| **0** | **Abstract** | [**3**](#Abstract) |
| **1** | **Introduction** | [**3**](#Introduction) |
| **2** | **Background and Related Work** | [**4**](#BackgroundAndRelatedWork) |
| 2.1 | The Necessity of Deepfake Detection | [**4**](#b2_1) |
| 2.2 | Deep Learning | **[5](#b2_2)** |
| 2.3 | Deepfake | [**5**](#b2_3) |
| 2.4 | Deepfake Creation | [**6**](#b2_4) |
| 2.5 | Deepfake Detection | **[7](#b2_5)** |
| 2.6 | Existing Solutions | [**7**](#b2_6) |
| **3** | **Expected Achievements** | [**8**](#ExpectedAchievements) |
| 3.1 | Our Primary Objective | [**8**](#b3_1) |
| 3.2 | The Process | **[9](#b3_2)** |
| 3.3 | Success Criteria | [**9**](#b3_3) |
| **4** | **Engineering Process** | [**9**](#EngineeringProcess) |
| 4.1 | Research | [**9**](#b4_1) |
| 4.2 | Constraints and Challenges | [**11**](#b4_2) |
| **5** | **Methodology and Development Process** | [**12**](#MethodologyAndDevelopmentProcess) |
| 5.1 | Project Management Methodology | [**12**](#b5_1) |
| 5.2 | Phase A - Research Process | [**12**](#b5_2) |
| 5.3 | Phase B - Development Process | [**12**](#b5_3) |
| 5.4 | The Product/System | [**13**](#b5_4) |
| 5.5 | Project Requirements | [**13**](#b5_5) |
| 5.6 | Model Architecture | [**14**](#b5_7) |
| 5.7 | Use Case Diagram | [**16**](#b5_8) |
| 5.8 | Activity Diagrams | [**17**](#b5_9) |
| 5.9 | User Interface | [**20**](#b5_6) |
| 5.10 | Data Preprocessing Tools | [**21**](#b5_10) |
| 5.11 | Face Detection tools | [**22**](#b5_11) |
| 5.12 | Audio processing tools | [**22**](#b5_12) |
| 5.13 | Model Training Framework | [**24**](#b5_13) |
| **6** | **Verification and Evaluation** | [**25**](#VerificationAndEvaluation) |
| **7** | **References** | [**27**](#References) |

**Abstract**

Deepfakes, a synthetic type of media created using artificial intelligent techniques, pose a growing threat, leveraging the artificial intelligence sophisticated power to create convincing social, political, and general day-to-day misinformation. This capstone project investigates a deepfake detection approach utilizing deep learning techniques to create multimodal analysis models. The system will analyze eye and mouth movements within video frames, along with voice characteristics, to identify deepfakes. Users will be able to upload videos, the system will preprocess them using established libraries like Face-Recognition and OpenCV for facial landmark detection and extraction of mouth and eye region. Audio extraction leverages libraries like MoviePy and SpeechRecognition. The system will include three separate machine learning models, each built upon Convolutional Neural Networks which are deep learning algorithms particularly adept at image and video analysis, each model specifically trained for video and or audio data respectively, are employed for analysis. Training data consists of short-form videos from the FaceForensics++, Celeb-DF, DFDC datasets for video, and human speech audio files from WaveFake, In-The-Wild, Deep-Voice datasets for audio. Each model outputs a probability score indicating the likelihood of the input being a deepfake. The system integrates with a user-friendly web application built with Django and JavaScript to facilitate video/audio upload and result visualization. This work acknowledges the limitations and challenges inherent in developing these deepfake detection models. Especially, it considers the ongoing advancements in deepfake generation techniques that require continuous adaptation of detection models, alongside the ever-growing computational demands such models place on training resources.

**1. Introduction**

Deepfake, a subset of artificial intelligence, are generally created using two main kinds of neural networks: Generative Adversarial Networks (GANs[[1]](#footnote-1)) and Autoencoders (AEs[[2]](#footnote-2)).

Deepfake utilized to fabricate believable visual, auditory, and video fakes. The term summarizes both the technology enabling the creation of these fakes and the fakes themselves, a fusion of 'deep learning' and 'fake'. Deepfakes typically modify pre-existing source content, substituting one individual with another, or they generate entirely new content where an individual is depicted performing or uttering something they did not perform or say.

Soon, every audio/visual content consumed by individuals will require investigation due to its potential influence on social perception and behavior. This is particularly relevant given the potential for mass acceptance of manipulated media, especially in the present day, where fake news are clearly apparent, and are concerningly rampant. This field is expanding at an extremely fast pace and is surely going to evolve by the minute. In our opinion, this project, or other similar projects are a necessity for this field.

In this project, we intent to develop a unique system which will utilize three main methods to analyze synthetically generated video and or speech data to accurately determine the likelihood of it being fake. These three methods being special machine learning models trained to detect irregularities and artifacts generated by AI in the eye and the mouth regions, as well as human speech. Each model will be trained on the specific region for optimal accuracy. In the following pages, we define our system’s structure, Deepfake ML training datasets to be used, several already-existing video and audio tools for preprocessing of the data, and the general structure of the system. Additionally, we expand upon the project’s goal, our planned roadmap, and the detection success criteria of 70%-80% in Expected Achievements [[Section 3]](#ExpectedAchievements).

A person in a uniform

Description automatically generated

Figure 1: Deepfake example.

**2. Background and Related Work**

**2.1 The Necessity of Deepfake Detection**

The existence of realistic fake videos predates the advent of deepfakes, with historical examples such as Nazi propaganda during World War II, where fabricated films were used to portray events favorably. Similarly, prior to deepfakes, concerns have been raised regarding the authenticity of videos, including doubts about the Apollo moon landing footage. Despite these earlier forms of fake video content, the increase of deepfakes poses a unique and significant challenge due to their advanced capabilities and the ease with which they can be created.

This rise could lead to substantial cognitive costs as individuals struggle with distinguishing between real and fake videos. Machine learning enables a broader population to create deepfakes, making the issue more widespread and concerning [[17]](#bookmark_17).

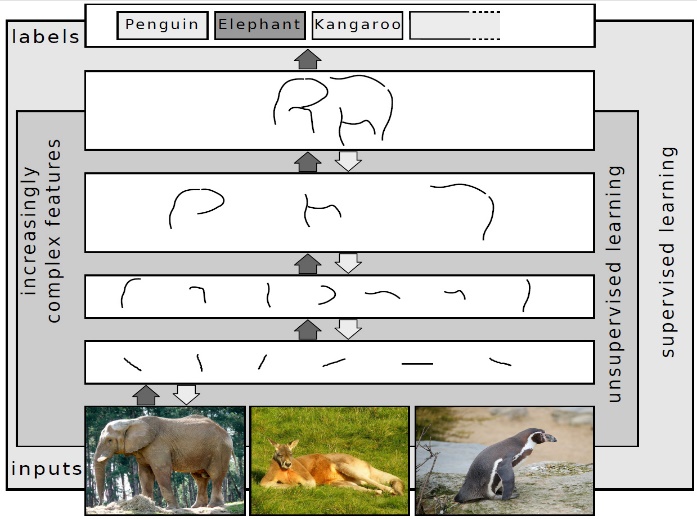
Deepfakes are a significant challenge threatening the dependability of face recognition systems and the integrity of information on the Internet. Faces play an important role in human interactions and biometrics-based person identification. Therefore, face manipulations can critically disrupt trust in digital communications and security applications (e.g., government, media) [[8]](#survey_8).

One of the primary dangers associated with Deepfakes is manipulation of elections, mistrust in significant public and private institutions, social division, harm to specific military or intelligence operations or capabilities, threats to the economy, and damage to international relations [[9]](#bookmark_9).

**2.2 Deep Learning**

Deep learning, a subset of machine learning, employs hierarchical architectures composed of multiple layers to iteratively obtain abstract representations from initial inputs. These representations are increasingly advanced, allowing the model to capture complex patterns and relationships within the data. For instance, in image recognition tasks, early layers might detect edges, while later layers may recognize higher-level objects such as numbers, letters, or facial features.

Alternatively, deep learning can be understood as a reproduction or automation of human cognitive processes, transforming raw input into a processed understanding. This perspective leads to the concept of "deeper" learning, which include a dual-process approach. Human learning initiates the process by extracting initial insights from the source material, and subsequent computer learning refines these insights to arrive at a final learned representation. The term "deepest" learning denotes the ultimate form of automation, where the system autonomously translates input into a fully learned object without explicit guidance.

  
Figure 2: Schematic overview of layer-wise learning of feature hierarchies. Increasingly complex features are determined from the input using unsupervised learning.

**2.3 Deepfake**

Deepfakes represent a form of synthetic media characterized by the usage of advanced computational models acquired from deep learning techniques to create visual and audio content that can more easily deceive the public. These digital creations can replace one individual's likeness with another's in a highly convincing manner. The creation of such manipulations is made possible by advanced machine learning algorithms, including the employment of autoencoders and generative adversarial networks (GANs), which are designed to generate new data instances that resemble the input data.



Figure 3: A fake Midjourney-created image of Donald Trump being arrested.

**2.4 Deepfake Creation**

There are various options for people to create deepfakes. Some of them are online apps, websites, open-source projects on GitHub. The main methods used for deepfake generation, implement the use of neural networks in a few clever ways:

**Generative adversarial networks (GAN):** Generative adversarial networks architecture is based on two networks: **Discriminator and Generator.** The discriminator network is a classifier. As for the generator, it gets random noise, which is the source of randomness that can give multiple answers to the same question effectively. With that noise, it generates an image. Those two networks are playing against each other in a minimax game, trying to beat each other. The generator creates an image, and the discriminator then tries to find errors in the image. If it does, it then notifies the generator. The generator tries again with a different image. And so on, so that after every step, they both improve. This competition drives both the discriminator to get better at identifying and the generator to produce the images [[1]](#bookmark=id.1fob9te).

**Autoencoder (AE):** An Autoencoder is a type ofneural network that learns to decompress data efficiently. It consists of three layers: **Encoder, Code, Decoder**. The Encoder layer compresses the input data into a compressed representation in a reduced dimension. The Code layer represents the compressed input fed to the decoder layer.

The decoder layer takes the encoded representation and attempts to reconstruct the original input data [[3]](#bookmark=id.2et92p0). But if we feed it a source image and a target image, it can generate an image that looks like the source image but with the target image features.

**2.5 Deepfake Detection**

Deepfake detection is done by analyzing multiple regions of a person’s face and detecting irregularities, deformed facial expressions, blurry areas around generated areas, etc. The analysis is done by various algorithms such as image processing, image segmentation, face detection, and pattern recognition.

Another method is audio analysis. Audio analysis recognizes irregularities in generated speech which are not present in human voices such as timing errors, tonal artifacts, and echo.

**2.6 Existing Solutions**

**Eye based deepfake detection:** There are a couple of interesting approaches regarding eyes-based deepfake detection. The first one noticed that generated faces often share a common flaw, which is an irregular iris shape. Article [[2]](#bookmark=id.3znysh7) suggests a system that uses a model that extracts the facial landmark points to localize the eyes and to perform pupil segmentation. This segmentation is then analyzed to determine if the pupil shape is irregular. Then it performs a parametric fitting to an ellipse using the mean squared error (MSE)[[3]](#footnote-3) optimization.

The second approach [[4]](#bookmark=id.3dy6vkm) noticed the lack of natural eye blinking. This system combines CNN[[4]](#footnote-4) and RNN[[5]](#footnote-5) techniques in a unique deep learning algorithm to look for the lack of eye blinking.CNN is used to distinguish between open-eye and closed-eye states in a frame.

In an image of a real human face captured by a camera, the corneal specular highlights of the two eyes are related as they are the results of the same light environment. But in deepfake videos, this feature is inconsistent. [[16]](#bookmark_16) Develops a method to automatically compare the corneal specular highlights of the two eyes and evaluate their similarity.

**Mouth based deepfake detection:** Teeth in deepfake videos often appear as a single mass with very little definition. This article [[7]](#bookmark=id.4d34og8) suggests using a model that detects facial images where the person has their mouth open with visible teeth (frames without visible teeth are eliminated). The model crops the mouth area and marks the mouth with static points. Then, they perform various calculations on the mouth to determine if the teeth are well-defined or not.

**Technical deepfake detection:** Detect statistical inconsistencies between altered faces and background images on deepfakes. In [[15]](#bookmark_15) they use 81 facial landmarks shape predictor in Dlib. Analyzing features at multiple scales and levels of abstraction detecting subtle inconsistencies or anomalies.

**Audio based deepfake detection:** Audio deepfake detection can be performed in a few ways, most notably using Bi-Spectral and Cepstral[[6]](#footnote-6) statistics on the audio signals, as suggested in [[5]](#bookmark=id.tyjcwt). Bi-Spectral analysis enhances features which AI-synthesized audio signals create and are very mild in authentic human voices. While Cepstral analysis reveals power components, which are audio features present in human speech but missing in AI-synthesized signals. These features are then extracted using Mel Frequency Cepstral Coefficient (MFCC[[7]](#footnote-7)), and a model can be used to detect them.

Method [[6]](#bookmark=id.1t3h5sf) on the other hand, suggests analyzing the dissimilarity between lip movement and speech. The model is trained to detect unsynchronized movement of the lips based on the audio input. Moreover, this technique can also locate the deepfake regions of a video. The training is done by splitting short semi-manipulated (Some segments are genuine, while others are synthesized) recordings of people talking to the camera and splitting them into 1 second segments. This allows the model to locate manipulated content more efficiently for each part of the video.

**3.** **Expected Achievements**

**3.1 Our Primary Objective**

We plan to develop three machine learning models, designed to identify Deepfake video and audio files. All three models will be built separately, using the tools mentioned in the Engineering Process part. Then, we plan on merging all three models, and make them interact with each other using a percentage estimator which we will build ourselves. The percentage estimator will get the classification each model returned and will calculate the output based on the results. We will only use videos that contain a person’s face. The first model will be based on classification of visual aspects. It will contain two methods for visual analysis, eye analysis, and mouth analysis. The second model will only analyze audio. The third model will be a merged model capable of both visual and auditory data. The models are the key component for our system, processing user-uploaded video or audio files to assess their authenticity.

The proposed system will feature an intuitive user interface that allows the uploading of media files. These files will then undergo analysis by the selected model, which will then provide the user with a probability estimate, indicating the likelihood of the uploaded content being a Deepfake.   
This approach allows us to use advanced machine learning techniques with user-friendly design to provide a practical solution for identifying Deepfake content accessible for everyone.

**3.2 The Process**

To achieve our objectives, the initial phase involves assembling a complete dataset comprising of a wide array of Deepfake videos. This dataset will serve as the base for training and refining our model. Following the collection and preparation of the dataset, the next step involves training the models. This process includes several key techniques including face detection, image processing, and audio analysis, which are essential for identifying and differentiating between genuine and synthetic content. Finally, we will develop a user-friendly interface that simplifies the process for users to upload and analyze their media files. This interface will allow users to easily interact with our ML model, providing them with real-time percentage evaluation of the authenticity of the content provided.

A diagram of a model

Description automatically generatedFigure 4: Manipulated Reality model and usage structure.

**3.3 Success Criteria**

Based on our extensive research, most successful results in this field of deepfake detection, boast a successful detection rate range of roughly 70%-90%. Though high, most solutions focus on a limited number of datasets, and are not suited for unseen or unusual data. Thus, we will aim to stay within this range or even surpass, using a broader range of training datasets and multiple ML models to achieve a good score for unusual data. In addition, our model should analyze data only when a face or speech is detected. The data analyzed should be as clear as possible, with clear face representation, good lighting, and good overall video quality for the face feature detection, as well as good speech samples, that will be lacking background noise as much as possible. These constraints on the training data will ensure the accuracy and quality of our project’s outcomes.

**4. Engineering Process**

**4.1 Research**

**Detecting deepfakes**

To date, our investigations have primarily focused on comprehending the mechanisms underlying the creation of deepfakes and their potential impacts on societal norms, both negative and positive. Subsequently, we proceed into the complicated domain of deepfake detection, which involve a multitude of complex solutions. Additionally, we explored the challenges associated with audio deepfakes, a specialized subset within the broader spectrum of deepfakes. Our objective was to gain a deeper understanding of existing solutions in both areas with the aim of developing a unified model capable of addressing both sets of challenges.

Presently, our emphasis is on collecting an extensive dataset to enhance the precision of our model. Given the computational constraints, our training data must include a broad spectrum of genuine and synthetic video and audio files. We will take video analysis by gathering large amounts of videos and cropping them to the eyes and the mouth and discarding everything else, and will be inspecting eye behavior indicative of deepfakes, including blinking patterns and abnormal iris shapes, as well as inconsistencies in mouth movements. For audio processing, which requires the same or even higher computational capabilities, we will crop the parts not containing speech and will try to integrate the most effective currently available solutions described later in the most efficient way possible.

**The continued nature of deepfake improvement**

Upon reviewing the literature and articles mentioned in [Section 2: Background], including articles and expert analyses, we have obtained insights indicating that our attempt to develop a model for fighting deepfakes may indeed evolve in a wrong direction due to the inherent properties of deepfake creation. It appears that any model capable of identifying imperfections in the deepfakes may, paradoxically, enable the creators of these deepfakes to improve their techniques and enhance the authenticity of the fake content. This suggests an escalating arms race between the development of deepfake detection algorithms and the advancement of deepfake generation capabilities. Such a scenario presents a never-ending challenge where the quality of deepfakes continues to improve until they become indistinguishable from genuine content, posing a formidable problem that both humans and machines may struggle to differentiate effectively.

**Dataset**

We must acquire a substantial dataset composed mainly of current deepfakes. Given the rapid advancements in deepfake generation techniques, the inclusion of older deepfakes in the training set could prove to be counterproductive. Modern deepfakes exhibit significantly higher quality and smoothness compared to earlier iterations, making them more representative of the current landscape of deepfake production. As a result, training models on outdated deepfakes may yield diminishing returns and may not appropriately prepare the model for identifying the latest advancements in deepfake technology**.** The chosen datasets are mentioned in the future development process section [below](#videoDatasets).

**Detection model**

To achieve maximum accuracy in our model for deepfake detection, it is essential to train it on an exceptionally large dataset consisting of video and audio files. The sheer volume of data requires a robust computational infrastructure capable of handling the processing demands. Without powerful computing resources, the training phase could extend to an impractically long duration and set back the suitable deployment of the model. Thus, we will mainly work with Keras (Python machine learning framework), as described later in [Section 5.13].

**4.2 Constraints and Challenges**

**Data storage:** This project faces a significant challenge due to the substantial storage requirements for video and sound data. Training the model necessitates hundreds of gigabytes of memory, which presents a logistical obstacle.

**Limited computational power:** The large dataset size necessitates training on average graphic cards, which typically have lower computational power compared to specialized hardware. This limitation leads to significantly extended training times.

* **Proposed solution:** **Iterative development with smaller data batches.** To address the challenges associated with data storage and computational limitations, the project will adopt an iterative development approach. This will involve working with smaller subsets of data during the initial development stages.
* **Transfer Learning:** Leveraging pre-trained models on large-scale image and video datasets like ImageNet or Kinetics can provide a robust foundation for deepfake detection models. This can reduce training time and improve initial performance, especially when data is limited.

**Evolving Deepfakes:** Deepfake creation techniques are constantly evolving, making it difficult for detection models to remain effective. The model needs to be continually updated with new training data encompassing the latest deepfake generation methods to maintain accuracy.

* **Proposed solution:** **Make code futureproof.** We will split our training and actual working product phases into a clear structure, explained and commented on thoroughly to allow future upgradeability and modification. This process will allow adjustment and training of a new model if the technology of deepfake generation improves further.
* **Active Learning:** Implementing active learning techniques allows the model to select the most informative data points for further training. Trimming content which does not contain faces or speech prior to detection will yield faster runtimes, less noise, and clearer examples for the model to learn from accurately. This will significantly improve model performance while reducing the overall amount of data required for training.

**Privacy Concerns:** Utilizing large amounts of video and audio data for training raises ethical concerns about privacy and data protection. For this reason,

* **Proposed solution:** **Analyze the uploaded data, return result, don’t save data.** we will discard of the uploaded content after providing the results of the deepfake analysis.

**5. Methodology and Development Process**

**5.1 Project Management Methodology**

We chose to follow the Agile software development method due to its ideal fit for small teams. Agile encourages close collaboration between both team members and promotes iterative development in small chunks. This allows for early error detection and correction, crucial for a project with limited resources and a constantly changing deepfake landscape. Additionally, Agile's adaptability allows to quickly integrate new techniques or datasets as they emerge in this rapidly evolving field. Finally, the emphasis on frequent feedback loops ensures we receive continuous input on model performance and project direction, enabling course correction and a more efficient development cycle.

**5.2 Phase A - Research Process**

We familiarized ourselves with deepfakes, including their creation using Generative Adversarial Networks (GANs), potential harms, and the ongoing advancements in deepfake techniques.

1. We conducted a scientific literature review on existing deepfake detection models and their underlying research.
2. Following this review, our interest shifted towards deepfake detection models based on human facial features.
3. To meet computational demands, we identified the mouth and eyes only, as key facial features to target for deepfake detection within the video frame.
4. Recognizing the potential for improved accuracy, we explored audio processing and feature extraction techniques alongside video analysis in the detection process.
5. To gain practical insights, we plan to develop two prototypes: one for video and one for audio data. Each prototype will involve preprocessing a small dataset and feeding it into a corresponding simple deep learning model. This initial exploration with these models is intended to inform further research and development efforts.

**5.3 Phase B - Development Process**

1. Gather and pre-process video datasets (FaceForensics++, Celeb-DF, DFDC) of deepfakes and real videos.
2. Gather and pre-process audio datasets (WaveFake, In-The-Wild, Deep-Voice) of deepfakes and real speech recordings.
3. Train and evaluate the initial model on a validation set.
4. Explore hyperparameter[[8]](#footnote-8) tuning to improve model performance.
5. Implement data augmentation techniques such as, spatial and color augmentation, time shift (shifting the order of frames or speed variations) to increase training data diversity.
6. Evaluate the model on an unseen test set to gauge generalizability.
7. Refine and document the model code and training process.
8. Develop basic visualizations for model performance and explanations, confusion matrix, ROC curve, precision-recall curve, etc.
9. Evaluate the model's performance on a separate test set to obtain a reliable estimate of its generalizability to unseen data. We will combine the results gathered from both video and audio data to try and develop.
10. Create a web-based user interface for simple interaction with the model.
11. Prepare a comprehensive project report summarizing the development process, results, and challenges encountered.
12. Discuss potential future work for next cycle, based on project outcomes, such as further model improvement, deployment strategies, or exploring different deepfake detection methods.

**5.4 The Product**

This work proposes a machine learning-based approach for detecting deepfakes in user-uploaded multimedia content featuring human subjects. The system analyzes the provided video or audio files and outputs a probability score indicating the likelihood of the content being a deepfake.

**5.5 Project Requirements**

Functional requirements

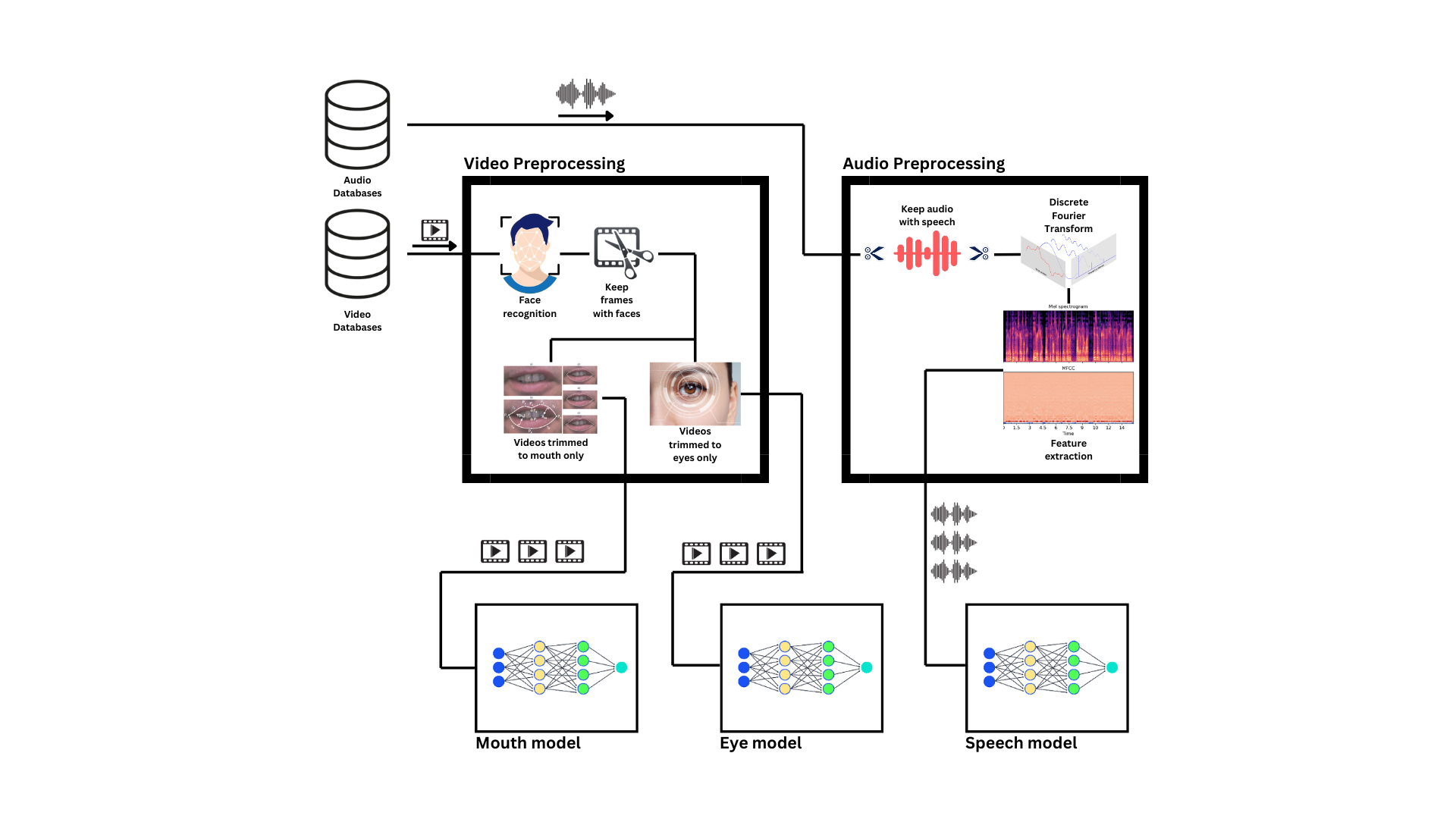
|  |  |
| --- | --- |
| 1 | The system will allow the user to upload videos/audio for analysis. |
| 2 | The system will inform user of probability of video/audio being a deepfake. |
| 3 | The system will consist of three machine learning models. Eye, mouth, and speech deepfake detection. |
| 4 | The system will allow the user to select which model they would like to use. Video only, audio only or combined. |
| 5 | The system will only use videos with human faces. It will notify the user if a video is unfit for detection. |
| 6 | The system will trim out audio which does not contain human speech prior to detection. |
| 7 | The system will preprocess the videos by discarding everything but the eyes and mouth. |
| 8 | The system will use a percentage estimator to estimate the contents’ genuineness. |

Non-functional requirements

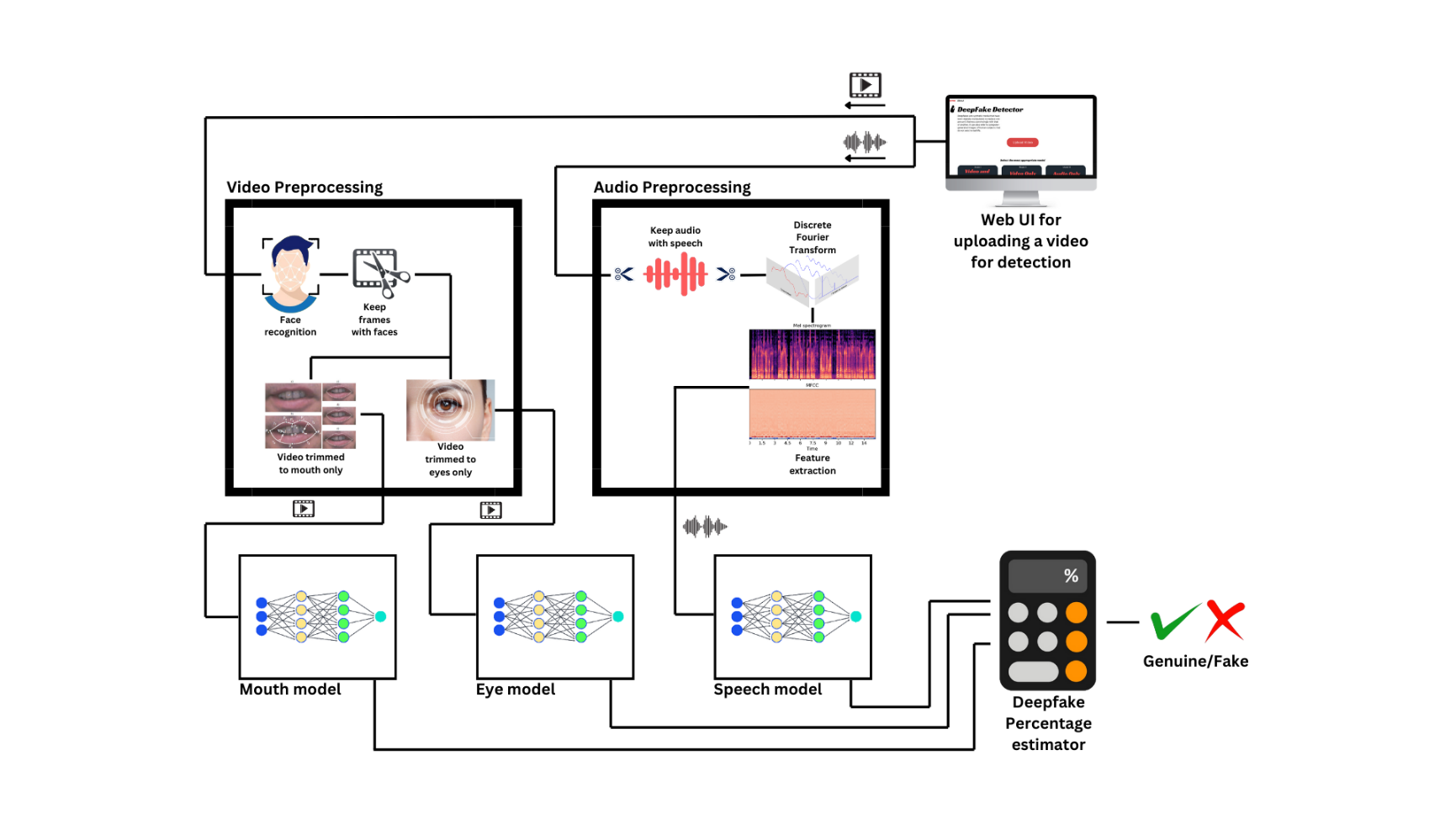
|  |  |
| --- | --- |
| 1 | The user graphical interface will include how-to-use tips and short explanations. |
| 2 | The system will display which models out of the three detected ingenuine content. |
| 3 | To prevent any personal data in the content provided from leaking, the data will be discarded after the model provides the result. |
| 4 | Our system will be easy to run and will be able to be used from any desktop or mobile browser. |
| 5 | The user interface will contain a thorough explanation about the model structure, training process, architecture and the steps taken in the detection process. |
| 6 | The code will be structured and thoroughly commented on to allow for future use, upgradeability, and scalability. |
| 7 | The system will require a minimum audio length of ~3 seconds. |
| 8 | The web UI will be created by the Django Web Framework. |

**5.6 Model Architecture**

**Phase 1 - Training:** Our development process consists of two phases, the first is the model training. The audio part will include using SpeechRecognition (A python library) to extract human speech and to discard everything else in the audio recording. It will then undergo feature extraction, such as MFCC, delta, and delta squared to create the training set. The extracted features from each audio sample will then be fed to a CNN model. The video part will include using face-recognition (A Python library) to extract only the human faces and will discard frames and parts of the video which are not relevant. The videos will then be cropped again using OpenCV, to extract the mouth and eyes to further isolate the required data regions. The preprocessed videos are then passed to the corresponding model in the standard CNN training methodology.

Figure 5: Training phase architecture.

**Phase 2 – Assembly:** The second development phase is the creation of the deepfake detection system. Below is a diagram describing the planned architecture of the detection system, intended for use by users. A user chooses to upload a video containing sound, or sound only and will choose detection either for the audio, video, or both video and audio data. As mentioned already, the system will have an easy-to-use user interface (GUI) implemented using Django, a high-level Python web application framework, which will allow the user to choose which content they want to assert the genuineness of. The system will perform audio segmentation (if audio is present) from the video file utilizing the MoviePy library (Python library for video editing). Audio will undergo preprocessing by cropping non-speech parts, while also extracting the features vector, and sent to the already-trained speech deepfake model. Video will undergo preprocessing by initially cropping the video to human faces while discarding anything else. Then, the video will be cropped and split into two video segments, one of the mouth region, and one of the eyes region. Subsequently, both videos will be sent to the already-trained mouth and eye detection models for detection. Then, the results of all three models will be calculated using a percentage estimator that will determine the data genuineness.

Figure 6: Detection model architecture.

**5.7 Use Case**

**Actors:**

* User.

**System:**

* Video and Audio Analysis for Deepfake Detection.

**Description:**

The user can interact with the system through the following options:

1. Upload video:

* The user uploads a video file to the system.
* The system offers the option to analyze the video for, visual aspects only or both visual and audio (speech) aspects.
* The system processes the video and returns a percentage score indicating the likelihood of the video being synthetically generated.

1. Upload audio:

* The user uploads an audio file containing speech.
* The system allows the user to specify a specific timestamp within the audio for targeted analysis.
* The system processes the audio and returns a percentage score indicating the likelihood of the speech being synthetically generated.

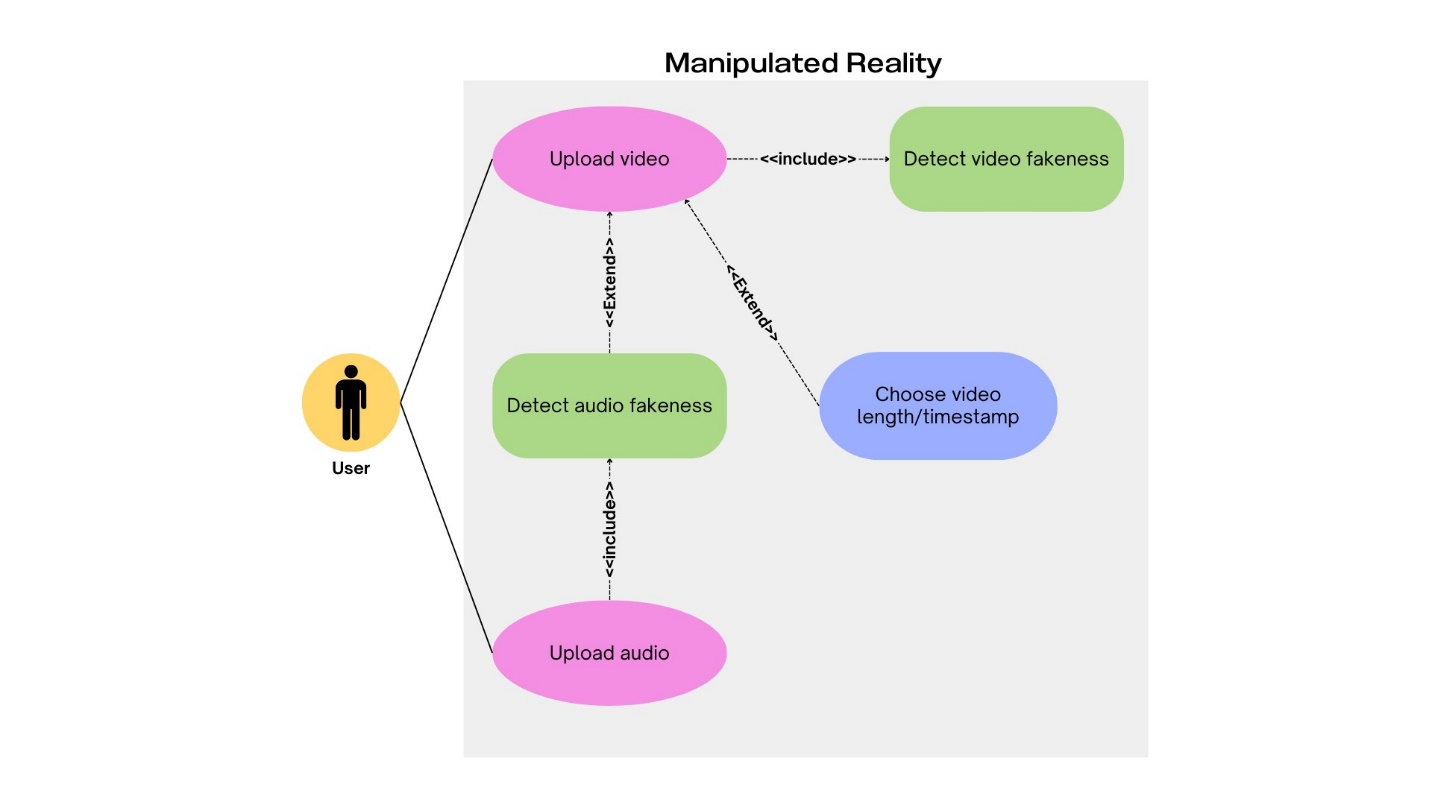


Figure 7: Use case diagram.

**5.8 Activity Diagrams**

Below are activity diagrams depicting the entire process of detection while using our system. This does not include the training phase, as that is something that will not be accessible to simple users of our system.

**Video deepfake detection activity diagram:** Below is an activity diagram of a process of detection done on a video by a user. The user starts by uploading a video to the website, the website then calculates the video length and lets the user choose the timestamps they wish to analyze. After choosing the timestamps, the user presses on the “Detect” button to start the deepfake detection. The website UI sends the video over to the backend, where it goes through pre-processing as described in [Section 5.6]. The backend checks to determine if a human face is present in the video, if not, it will prompt the user to upload another video for detection. If yes, the video is then analyzed, and the result is displayed to the user through the website UI.

**Audio deepfake detection activity diagram:** Similarly to the video deepfake detection,if the user chooses to detect deepfakes in an audio sample, the website allows the user to upload the audio. The audio file length is calculated using the website UI and the user can choose a timestamp to detect. After choosing the timestamps, the user presses on the “Detect button to start the deepfake detection. The website sends the audio sample to the backend, where it checks for the presence of human speech. If no human speech is present, the website then prompts the user to upload a different audio sample. If human speech is present in the audio, the sample is analyzed using the speech deepfake detection model and the result, either fake or genuine is displayed to the user through the website UI.

A diagram of a company

Description automatically generated

Figure 8: Video activity diagram.

A diagram of a company

Description automatically generated

Figure 9: Audio activity diagram.

**5.9 User Interface**

The user interface is a web-based application designed to facilitate deepfake detection. The UI consists of three primary stages:

**Landing Page (**[Figure 10](#Figure_landing_page)**):** This initial page serves as the entry point for user interaction. It provides functionalities for uploading video content to be analyzed for deepfake characteristics, and to choose different model depending on his needs.

A screenshot of a computer

Description automatically generated

Figure 10: Landing page gives the options to upload video or select appropriate model.

**Preprocessing Stage (**[Figure 11](#Figure_preprocessing_page)**):** Following video upload, an automated preprocessing step is initiated. This stage prepares the video data for subsequent deep learning model analysis.

A screenshot of a computer

Description automatically generated

Figure 11: Preprocessing step, after user upload data to be analyzed.

**Result Display (**[Figure 12](#Figure_result_page)**):** Upon completion of analysis, the UI presents the user with the outcome. This includes a probability score indicating the likelihood of the uploaded video being a deepfake. Additionally, the UI offers the option to upload another video for analysis.

A screenshot of a computer

Description automatically generated

Figure 12: Result display page.

**5.10 Data Preprocessing Tools**

Our deepfake detection project’s data preprocessing can be achieved with several powerful tools and libraries. The choice of tool(s) will depend on the specific requirements of our dataset and the preprocessing tasks at hand. Here are some recommended options:

1. **OpenCV**: OpenCV (Open-Source Computer Vision Library), It provides a comprehensive set of functions for loading, manipulating, and transforming image/video data. OpenCV can be used for tasks such as resizing, normalization, data augmentation (e.g., flipping, rotating, scaling), and converting data to formats compatible with our deep learning models.
2. **Pandas and NumPy**: Pandas and NumPy are essential libraries for data manipulation and numerical computing in Python. They can be used for tasks such as loading data from various sources (e.g., CSV, SQL databases), handling missing values, and performing necessary data transformations before feeding the data into our deep learning models.
3. **Pillow (PIL)**: Pillow, also known as the Python Imaging Library (PIL), is a library for image processing. It can be useful for tasks such as loading, resizing, converting image formats, and applying basic image transformations.

It is worth noting that we can combine multiple tools and libraries to create a thorough data preprocessing pipeline fitted to our specific requirements.

Additionally, we may consider implementing custom preprocessing functions or utilizing third-party libraries specific to our deepfake detection task if the available tools do not meet our needs. The choice of tools will depend on factors such as dataset size, complexity, performance requirements, and ease of integration with our deep learning modeling pipeline.

**5.11 Face Detection tools**

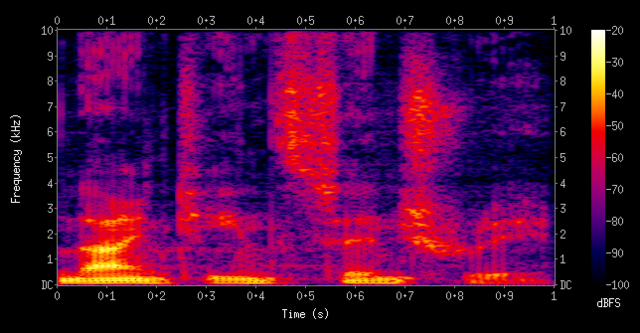
The first step in deepfake detection is face detection. This requires the use of dedicated frameworks and libraries. We present 3 useful tools, and will be selecting the most suitable, based on our computing capabilities and the accuracy of each detector.

1. **OpenCV** [**[11]**](#bookmark_11): OpenCV comes with several pre-trained face detectors that we can use out-of-the-box. One of the most used face detectors is the Haar Cascade classifier, known for its efficiency and adaptability. But the Haar Cascade classifier is a traditional approach and might not be as accurate as modern deep learning-based face detectors.
2. **Face-recognition** [**[12]**](#bookmark_12): Popular and powerful open-source library for face recognition and facial feature extraction in Python. It is built on top of the state-of-the-art face recognition algorithms and can be used for various tasks, including face detection, recognition, and facial landmark detection. The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark.
3. **MediaPipe** [**[13]**](#bookmark_13)**:** An open-source framework developed by Google for building machine learning pipelines to process time-series data such as video and audio. It is primarily designed for machine learning teams and software developers who implement production-ready ML applications, as well as students and researchers who publish code and prototypes as part of their research work. However, it's worth noting that MediaPipe is a relatively new and evolving solution, and its performance and accuracy may vary depending on the specific use case and requirements.

**5.12 Audio processing tools**

Audio will play a crucial part in the development of this model; our goal is to use audio detection to improve the overall accuracy. This means that if the video detection fails, the audio detection can play a crucial part in determining the validity of the content uploaded.  
We plan on trying out two methods of preprocessing and model training and going with the best accuracy and runtime performing one:

1. **Audio to spectrogram analysis:** This process will involve preprocessing the audio and extracting key features as mentioned in [[5]](#bookmark=id.tyjcwt). The input for the audio CNN model will be the extracted audio features fed to the neural network as images.

  
Figure 13: Audio spectrogram

1. **Audio to vector analysis**: This process will involve preprocessing the audio and extracting key features as vectors. Instead of using spectrograms which complicate model training as their dimensions are larger, we will provide the neural network model with a feature vector of the audio samples with the key extracted features.

**Libraries:**

**Scipy**: This python library allows us to use Fast Fourier Transform to generate the audio vectors from the audio data.  
**Librosa**: This library allows us to use the MFCC feature extraction functions.  
**Soundfile**: Allows us to turn stereo audio samples into mono samples.  
**Numpy**: A crucial part in trimming audio vectors and matrices, as well as matrix and vector manipulation and concatenation.

**Datasets:**

**WaveFake**: Large scale dataset of over 100K generated audio clips.  
**In-The-Wild**: A dataset of 58 politicians and celebrities split into real and fake samples.  
**Deep-Voice**: A dataset containing real human speeches and deepfake versions of those speeches.

**5.13 Model Training Framework**

**Keras** [**[14]**](#bookmark_14)**:** Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It is user-friendly, modular, and scalable. It was developed with a focus on enabling fast experimentation, and it allows for easy and fast prototyping, supporting both CNN’s and RNN’s, as well as combinations of the two.



We will leverage the Keras deep learning library for several key stages of our methodology, as mentioned previously. Specifically, Keras will be employed for the following tasks:

1. **Model Architecture:** Provides several options and utilities for defining and customizing our model’s architecture. Such as **Sequential, Functional, Pre-trained Models, Transfer Learning, Custom Layers and Architectures and Model Subclassing.**
2. **Model Definition: We can** Specify the input shape, convolutional and pooling layers, fully connected layers, and the output layer (e.g., a dense layer with sigmoid activation for binary classification).
3. **Model Compilation:** We will choose an appropriate loss function (e.g., binary cross-entropy for binary classification). Select an optimizer (e.g., Adam, RMSprop) and a metric to monitor during training (e.g., accuracy). Compile the model with the specified loss, optimizer, and metrics.
4. **Training:** We will use model.fit() to train our model on the training data, and monitor the training and validation metrics to ensure the model is learning and not overfitting.
5. **Model Evaluation:** We will evaluate the trained model's performance on a test set using model.evaluate(), analyze the model's accuracy, precision, recall, and other relevant metrics.
6. **Model Tuning and Improvement:** We will experiment with different architectures, hyperparameters (e.g., learning rate, batch size), and regularization techniques (e.g., dropout, L1/L2 regularization) to improve model performance.
7. **Model Saving and Loading:** We will save our trained model using model.save() for later use or deployment, load a saved model using keras.models.load\_model() for inference or further fine-tuning.

**6. Verification and Evaluation**

This table presents a series of test cases designed to evaluate the functionality of our program. The chosen testing framework is the "Unittest" module, a built-in component of the Python standard library, adhering to the recommendations outlined within the Django documentation.

|  |  |  |
| --- | --- | --- |
| Case | Test Case | Expected Result |
| 1 | Upload a video file (e.g., mp4) | The video is loaded and preprocessed, user taken into next page. |
| 2 | Upload an audio file (e.g., mp3) | The audio is loaded and preprocessed, user taken into next page. |
| 3 | Select other detection model | The system acknowledges the user's selection. Highlighting the chosen model. |
| 4 | Press “apply detection” | The system displays a clear indication that analyzing has begun using a progress bar. |
| 5 | Press “choose other video” | Allows the user to upload a different video. |
| 6 | Upload a video/audio in an unsupported format | Error is shown:  “Wrong file format, please input mp4/mp3 file.” |
| 7 | Upload a corrupted video/audio file | System displays an error message indicating corrupted file. |
| 8 | Upload a large video/audio file (exceeding size limit) | System displays an error message indicating file size limitation. |
| 9 | Upload an empty file | System displays an error message indicating no file uploaded. |
| 10 | Attempt to upload video into audio only model | Error is shown:  “Wrong model selection, please input audio file, or select other model.” |
| 11 | Attempt to upload audio into video only model | Error is shown:  “Wrong model selection, please input video file, or select other model.” |
| 12 | No result output in the final step | Error is shown: “An error occurred while analyzing the video/audio. Please try again.” |

**GUI Unit Testing:** Unit testing with Unittest can be employed to ensure the Graphical User Interface (GUI) functions as intended and identify potential errors or bugs that could hinder user experience.

**Validation**

**Front-end:** We will employ Selenium, an automated web application testing framework, to conduct comprehensive functional testing of the user interface.

**Model:** Our model will focus on three main aspects in the data: mouth, eyes, and audio. For the mouth and eyes, our model will try to detect the fakeness percentage only if a face was recognized. For data that does include a person’s face, we expect to achieve high to perfect detection accuracy for low-quality deepfakes and medium to high detection accuracy for high-quality deepfakes. This is due to low-quality deepfakes often exhibiting more outstanding artifacts that are easy to identify. On the contrary, high-quality deepfakes may present a greater challenge. The model's performance in this area is expected to be limited by the currently available training datasets, which are often skewed towards lower quality content. However, an accuracy range of 50-60% is still anticipated for high-quality deepfakes.

Audio validation will be restricted to data containing speech, either standalone audio files or videos with speech components. As in the video model, high to perfect accuracy is expected for low-quality manipulated audio while high-quality synthetic audio detection accuracy might prove to be lower.

**User experience: Before the final presentation, we will conduct UI environment and deepfake detection tests with help from fellow students. We plan on conducting these tests on at least 5 students. This process will provide outside feedback for the goal of improving the model and useability of the system.**

* **UI environment:** We will conduct user testing to gather feedback on the application's ease of use.
* **Detection model:** We will generate a deepfake video of the subjects’ face and use it to test our models’ performance.

**7. References**

**[1]** Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. arXiv preprint. **[arXiv:1406.2661]**.

**[2]** Guo, H.; Hu, S.; Wang, X.; Chang, M.C.; Lyu, S. Eyes Tell All: Irregular Pupil Shapes Reveal GAN-Generated Faces. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Singapore, 22–27 May 2022; (pp. 2904–2908).   
**[**[**arXiv:2109.00162**](https://arxiv.org/abs/2109.00162)**].**

**[3]** Autoencoders: <https://www.datacamp.com/tutorial/introduction-to-autoencoders>

**[4]** Yuezun Li, Ming-Ching Chang, Siwei Lyu, In Ictu Oculi: Exposing AI Generated Fake Face Videos by Detecting Eye Blinking, 7 Jun 2018, **[arXiv:1806.0287[**.

**[5]** [Arun Kumar Singh](https://arxiv.org/search/cs?searchtype=author&query=Singh,+A+K) (1), [Priyanka Singh](https://arxiv.org/search/cs?searchtype=author&query=Singh,+P) (2) ((1) Indian Institute of Technology Jammu, (2) Dhirubhai Ambani Institute of Information and Communication Technology) ,Detection of AI-Synthesized Speech Using Cepstral & Bispectral Statistics, **]arXiv:2009.01934[**.

**[6]** Not made for each other- Audio-Visual Dissonance-based Deepfake Detection and Localization, [Komal Chugh](https://arxiv.org/search/cs?searchtype=author&query=Chugh,+K), [Parul Gupta](https://arxiv.org/search/cs?searchtype=author&query=Gupta,+P), [Abhinav Dhall](https://arxiv.org/search/cs?searchtype=author&query=Dhall,+A), [Ramanathan Subramanian](https://arxiv.org/search/cs?searchtype=author&query=Subramanian,+R), **[arXiv:2005.14405]**.  
<https://github.com/abhinavdhall/deepfake>

**[7]** DFT-MF: Enhanced deepfake detection using mouth movement and transfer learning, Ammar Elhassan, Mohammad Al-Fawa'reh, Mousa Tayseer Jafar, Mohammad Ababneh, Shifaa Tayseer Jafar, SoftwareX Volume 19, July 2022, 101115. **[DOI:**[**10.1016/j.softx.2022.101115**](http://dx.doi.org/10.1016/j.softx.2022.101115)**].**

**[8]** Zahid Akhtar, Deepfakes Generation and Detection: A Short Survey, Journal of Imaging (ISSN 2313-433X), 13 January 2023. **[DOI:**[**10.3390/jimaging9010018**](http://dx.doi.org/10.3390/jimaging9010018)**]**.

**[9]** Danielle K. Citron & Robert Chesney, Deep Fakes: A Looming Challenge for Privacy, Democracy, and National Security, in 107 California Law Review 1753 (2019). <https://scholarship.law.bu.edu/faculty_scholarship/640/>

**[10]** Jeff Donahue, Lisa Anne Hendricks, Marcus Rohrbach, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, Trevor Darrell, Long-term Recurrent Convolutional Networks for Visual Recognition and Description, arXiv:1411.4389, 17 Nov 2014. **[**[**arXiv:1411.4389**](https://arxiv.org/abs/1411.4389)**].**

**[11]** OpenCV: <https://pypi.org/project/opencv-python/>.

**[12]** Face-recognition, a python library: <https://pypi.org/project/face-recognition/>.

**[13]** Mediapipe, a google framework designed for ML applications <https://pypi.org/project/mediapipe/>.

**[14]** Keras, a neural network API <https://keras.io/>

**[15]** Detecting Deepfakes with Self-Blended Images Kaede Shiohara Toshihiko Yamasaki. The University of Tokyo {shiohara, [yamasaki}@cvm.t.u-tokyo.ac.jp](mailto:yamasaki%7d@cvm.t.u-tokyo.ac.jp) **[**[**arXiv:2204.08376**](https://arxiv.org/abs/2204.08376)**]**.

**[16]** EXPOSING GAN-GENERATED FACES USING INCONSISTENT CORNEAL SPECULAR HIGHLIGHTS Shu Hu, Yuezun Li, and Siwei Lyu Computer Science and Engineering University at Buffalo, State University of New York, USA **[**[**arXiv:2009.11924**](https://arxiv.org/abs/2009.11924)**]**.

**[17]** Fallis, D. The Epistemic Threat of Deepfakes. *Philos. Technol.* 34, 623–643 (2021). <https://doi.org/10.1007/s13347-020-00419-2>

1. Generative adversarial networks are two neural networks that compete against each other. The first network generates images, and the second identifies the mistakes. They then work in a continuous loop to try and improve each other. [↑](#footnote-ref-1)
2. An Autoencoder is a type ofneural network that learns to reconstruct its output data (such as images, text, audio, etc.) by compressing the input, decompressing the output, and comparing both input and output using a loss function. [↑](#footnote-ref-2)
3. Mean squared error (MSE) average squared difference between the estimated values and the actual value. [↑](#footnote-ref-3)
4. Convolutional neural networks (CNN) are a deep learning algorithm that is especially good for image recognition and processing tasks. It is made up of multiple layers, including convolutional layers, pooling layers, and fully connected layers. [↑](#footnote-ref-4)
5. Recurrent neural network (RNN) is a neural network where the output from the previous step is fed as input to the current step. [↑](#footnote-ref-5)
6. Cepstral & Bi-spectral statistics are statistical tools used in signal processing, mainly used in audio and speech analysis. Cepstral analysis is used to help us understand the frequencies present in an audio signal, and how it changes over time. Bi-spectral Analysis is a method for looking at the relationships between three dimensions of an audio signal to understand how the sound waves interact with each other. [↑](#footnote-ref-6)
7. MFCC are features extracted from an audio signal, widely used in speech and audio processing. They represent the spectral characteristics of sound in a manner that is suitable for various machine learning tasks. The coefficients capture the shape of the power spectrum of a sound signal, emphasizing features of the audio signal that are important for human speech perception, and discarding less relevant information. [↑](#footnote-ref-7)
8. Settings that define how the learning process occurs. They aren't directly learned by the model, but rather control the learning algorithm itself. [↑](#footnote-ref-8)