Development of deepfakes detection model using deep learning framework

Summary

Deepfake technology, which utilizes deep learning algorithms to create realistic but fabricated media by altering a person's appearance or voice, poses significant challenges in various domains, including identity theft, financial fraud, and political manipulation. This project aims to address these concerns by developing a robust and accurate deepfakes detection model using image processing and deep learning techniques.

The project begins with an introduction to deepfakes, highlighting their potential consequences such as identity theft, financial fraud, and political manipulation. It also explores the practical applications of deepfakes, including de-ageing in movies, entertainment, animating historical photos, and the legal and ethical challenges they present.

The main objective of the project is to develop a model capable of discerning between authentic and synthetic media content. To achieve this, the project follows a systematic approach. A block diagram is presented, illustrating the ResNeXt and LSTM architectures used in the detection model.

The project's methodology involves conducting a literature survey on deepfakes detection techniques, gathering and pre-processing available datasets, training the ResNeXt architecture neural network, and analysing and evaluating the developed model. The literature survey explores various methods, such as eye blinking detection, smart watermarking, and inconsistency detection in facial features. The dataset is created by combining pre-existing datasets and pre-processing the videos to extract faces and resize them uniformly.

The outcomes of the project include a deepfakes detection model with high accuracy and robust performance. The model's effectiveness is demonstrated through the evaluation of performance metrics, including accuracy, loss, and confusion matrices. Furthermore, the project highlights the importance of considering biases and optimizing the model for real-world scenarios.

In conclusion, this project contributes to the development of a deepfakes detection model that can help mitigate the risks associated with synthetic media manipulation. By leveraging deep learning and image processing techniques, the model offers a reliable solution to identify and combat deepfake content, safeguarding individuals and organizations from potential harm.

Table of Contents

List of Tables

List of Figures

Abbreviation and Acronyms

- CNN - Convolution Neural Network
- GAN - Generative Adversarial Network
- LSTM - Long short-term memory
- ResNeXt - Residual Network
- ReLU - Rectified Linear Unit
- GPU - Graphical Processing Unit
- FPS - Frame Per Second
- TP - True Positive
- TN - True Negative
- FP - False Positive
- FN - False Negative
- MSE - Mean Squared Error
- ROC - Receiver Operating Characteristic
- SGD - Stochastic Gradient Descent
- ADAM - Adaptive Moment Estimation
- BLEU - Bilingual Evaluation Understudy

1. Introduction

1.1 Abstract

Driven by sophisticated machine learning algorithms, deepfake technology has become a noteworthy technological development with both advantages and disadvantages. This abstract highlights the dangers of deepfakes and offers instances of their possible misuse. Deepfakes are made-up media, usually videos, that purport to show people saying or acting in ways they never would have. Deepfakes pose major risks to many facets of society, even though they have many acceptable uses, such as amusement and age-delaying in motion pictures.

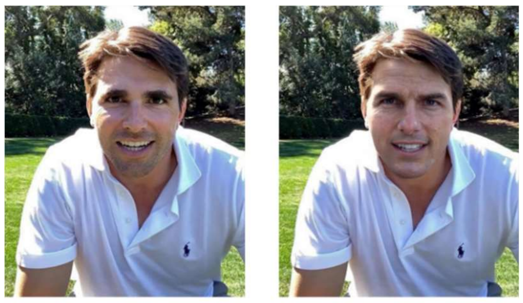


Fig 1.1.1: Deepfake example

Significant privacy breaches are caused by deepfakes. Deepfakes can cause harassment, blackmail, and harm to one's reputation both personally and professionally by superimposing someone's likeness over pornographic or compromising information. Such invasions of privacy may have detrimental psychological and emotional effects on the people who are the targets.

In addition, there are moral and legal issues with deepfakes. They can violate rights to intellectual property, cast doubt on the veracity of digital evidence used in court, and muddy the lines between fact and fiction. Consent, media manipulation, and responsible AI use all present ethical challenges.

In conclusion, deepfake technology has a lot of room for abuse even as it presents intriguing new opportunities. The spread of false information, fraud and scams, invasions of privacy, and ethical and legal issues are a few instances of possible misuse. It is imperative that individuals, organisations, and legislators tackle these concerns by means of technology innovations, public education initiatives, and strong regulatory structures in order to lessen the detrimental effects of deepfakes on society. We hope that by working on this project, we will be able to mitigate the dangers posed by deepfakes and shield society from any possible harm they may cause. Our goal is to use image processing and deep learning methods to create a deepfakes detection model.

1.2 Motivation

The prevalence of deepfake technology and its potential for misuse have become increasingly evident through real-world incidents like the ones you mentioned. These examples have served as powerful motivators for us to undertake this project, as we recognize the urgent need to address the pervasive threat posed by deepfakes and protect individuals, communities, and organizations from their harmful consequences.

Deepfakes may have an adverse effect on democratic processes, as demonstrated by the Telangana incident, where voters were emailed films showing a sitting minister pleading for votes against the current state administration. Election integrity can be compromised by such practices, which can also deceive voters and weaken public confidence in the democratic system. By creating a strong deepfake detection model, we hope to enable people to recognise and oppose these dishonest strategies, protecting the openness and equity of democratic processes.

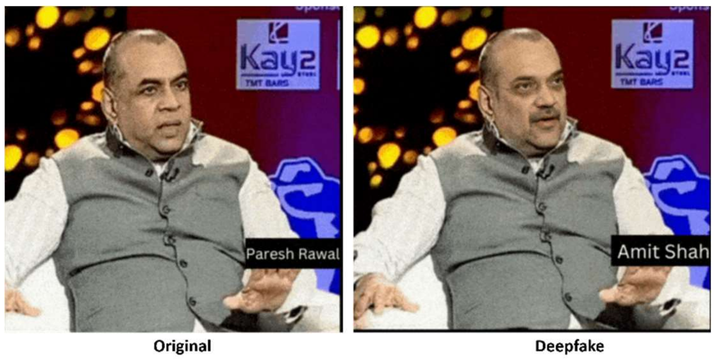


Fig 1.2.1: Deepfake for political manipulation

The ability of deepfakes to sway public opinion is demonstrated by the appearance of videos in Madhya Pradesh that used scenes from a well-known TV programme to incite viewers' animosity towards the incumbent. These tricks have the power to amplify societal turmoil, deepen rifts, and warp people's perceptions of reality. With the help of our project, people will be able to recognise and assess media information critically, empowering them to resist manipulation of public opinion and make wise judgements.

The deepfake scam involving a multinational corporation in Hong Kong showcases the significant financial risks associated with this technology. The meticulous approach employed by the fraudsters, utilizing advanced deepfake techniques to replicate high-ranking company officials, emphasizes the sophistication and potential impact of such scams. By developing an effective deepfake detection model, we aim to provide organizations with the means to identify and prevent deepfake-based fraud, safeguarding their financial resources and preserving trust in the business environment.
The cumulative illustration of these cases emphasises how urgently effective deepfake detection and mitigation techniques are needed. As accountable members of the

technology community, we are driven to help in the creation of practical remedies that can shield people from the negative impacts of deepfakes—individuals, groups, and institutions. We understand how critical it is to protect people's financial security, stop false information from spreading, and maintain the integrity of democratic processes.

Our objective is to stimulate responsible innovation and advance the appropriate application of AI by tackling the ethical and societal issues raised by deepfake technology. Our goal with this project is to help create policies and procedures that guarantee the moral application of deepfake technology while safeguarding people's security, privacy, and rights.

To sum up, these are a few of the instances that have strongly encouraged us to take on this job. Our motivations are to defend democratic procedures, dispel false information, reduce financial risk, and deal with the moral dilemmas raised by deepfake technology. Our goals are to improve society, promote trust, and lessen the possible harm that deepfakes may cause by creating an efficient deepfake detection model.

1.3 Literature survey

The detection and mitigation of deepfake videos have become critical research areas due to the increasing prevalence and potential harm caused by synthetic media. To develop your project, you have referred to several scientific papers that propose various techniques and approaches for deepfake detection.

1. Chang, M. C., Li, Y., and Lyu, S. [2] The lack of eye blinking, a physiological indicator that is poorly recorded in artificially created fake videos, is how Li et al. expose AI-generated phoney face videos in their paper. This method makes use of the distinctive features of spontaneous eye blinking to spot irregularities and tell authentic recordings from fakes.

2. In 2021, [5]-Afchar, Nozick, Yamagishi, and Echizen Smart watermarking is investigated by Afchar et al. as a preventative measure against deepfake picture alteration. They look at the use of generative adversarial networks (GAN) and convolutional neural networks (CNN) for deepfake detection with the goal of identifying and stopping image-based deepfake manipulations.

3. Sayali Nagbhidkar, Sanjana Singh, Aarti Karandikar, Vedita Deshpande, and Saurabh Agrawal (April of 2020) [7]— In order to identify anomalies in face characteristics, compression rate, and discrepancies presented during the development of deepfake films, Karandikar et al. offer a deep learning-based methodology that makes use of convolutional neural networks (CNN). The

programme attempts to detect telltale signals of deepfake manipulation by utilising CNN's capacity to understand intricate patterns.

4. Tanzim Reza, Tahera Khanom Tinni, Zami Al Zunaed Farabe, Zalish Mahmud, MD, Gabi Hasin Ishrak, and Muhammad Zavid Parvez (2024) [6] "- For deepfake video detection, Ishrak et al. provide a hybrid model in this research that combines Long Short-Term Memory (LSTM) with Convolutional Neural Network (CNN) and CapsuleNet. With regard to rotating or tilting images, as well as the information loss resulting from layer pooling, this model seeks to address CNN's shortcomings. The authors also stress how critical it is to include justifications for the detection choices made in order to improve the model's interpretability.

5. Thanh Thi Nguyena, Quoc Viet Hung Nguyenb, Dung Tien Nguyena, Duc Thanh Nguyena, Thien Huynh-Thec, Saeid Nahavandid, Thanh Tam Nguyene, Quoc-Viet Phamf, Cuong M. Nguyeng (August 2022) [31] – offers an extensive overview of deep learning methods applied to the production of deepfakes, or extremely lifelike AI-generated content that can manipulate faces, lip movements, and facial animations. This study analyses the current state-of-the-art in deepfake detection approaches, primarily based on deep learning, with the goal of assisting in the development of more reliable solutions to counter the threat posed by deepfakes.

6. A thorough and methodical evaluation of deep learning-based techniques for identifying deepfakes is given by Arash Heidari, Nima Jafari Navimipour, Hasan Dag, and Mehmet Unal (October 2023) [32]. It looks at different deep learning architectures, training sets, and assessment measures used in the field of deepfake detection studies. In this quickly developing field of study, it outlines the main obstacles, most recent developments, and potential future research areas.

All of these scientific articles provide a thorough summary of various methods and strategies for deepfake detection. They address topics like CNN-based detection, hybrid models, physiological signals, and watermarking. You can create a deepfake detection system for your project that is comprehensive and efficient by taking into account and expanding upon the methods and conclusions discussed in these papers.

2. Background Theory

2.1 Introduction

Deepfake videos, created using neural network tools like GANs (Generative Adversarial Networks) or Autoencoders, involve synthesizing human images onto source videos. These techniques utilize deep learning algorithms to seamlessly blend target images with the movements and expressions of individuals in the source videos, resulting in highly realistic-looking videos that can be difficult to distinguish from genuine footage.

Nonetheless, there are several restrictions on deepfake production tools that can be used to distinguish between pristine and deepfake videos. These tools frequently add little anomalies or artefacts into the frames during the development process, which are not visible to the human eye but can be picked up by trained neural networks. These unique artefacts are important markers for spotting deepfakes.

By taking advantage of the shortcomings of deepfake production tools, we present in this study a deep learning based approach that successfully differentiates artificial intelligence (AI) created fake videos from actual videos. Specifically, they concentrate on using Res-Next Convolutional Neural Networks (CNNs) to extract information at the frame level from the videos. CNNs can catch complex patterns and information within the frames, making them ideal for image analysis jobs.

Next, a Long Short-Term Memory (LSTM) based Recurrent Neural Network (RNN) is trained using the extracted frame-level characteristics. A particular kind of RNN called LSTMs is excellent at simulating temporal dependencies in sequential data, which makes them a good fit for video sequence analysis. Deepfake and genuine videos can be distinguished from one another thanks to the training of an LSTM-based RNN to determine whether a video has been altered in any way.

We suggest training the model on a variety of accessible datasets in order to improve its performance on real-time data. They take a large amount of videos from of datasets like Celeb-DF [3], FaceForensic++ [1], and Deepfake Detection Challenge [2]. The model gains the ability to recognise deepfake movies from a variety of sources and situations by learning a broad range of features from different kinds of images by training on a variety of datasets.

2.2 Steps involved in detecting deepfake

Dataset Collection: In order to optimise the model for real-time prediction, we have collected data from multiple publicly available datasets, such as FaceForensic++ (FF), Deepfake Detection Challenge (DFDC), and Celeb-DF. In order to provide precise and fast identification of deepfake films, the goal is to build an extensive dataset that includes a variety of video formats. Half of the movies in the dataset are fictitious, and the other half are real, to prevent training bias.

Certain movies with altered audio can be found in the Deepfake Detection Challenge (DFDC) dataset; however, audio deepfakes are not included in this paper's scope. In order to exclude the audio-altered films from the DFDC dataset, a Python script was utilised for preprocessing.

The researchers took 1000 real movies and 1000 false videos from the DFDC dataset after preprocessing it. 500 genuine movies and 500 fraudulent videos were taken from the FaceForensic++ (FF) dataset. In addition, the Celeb-DF dataset had 500 authentic videos and 500 phoney videos. Four thousand videos total—two thousand real videos and two thousand fraudulent videos—make up this combined dataset.

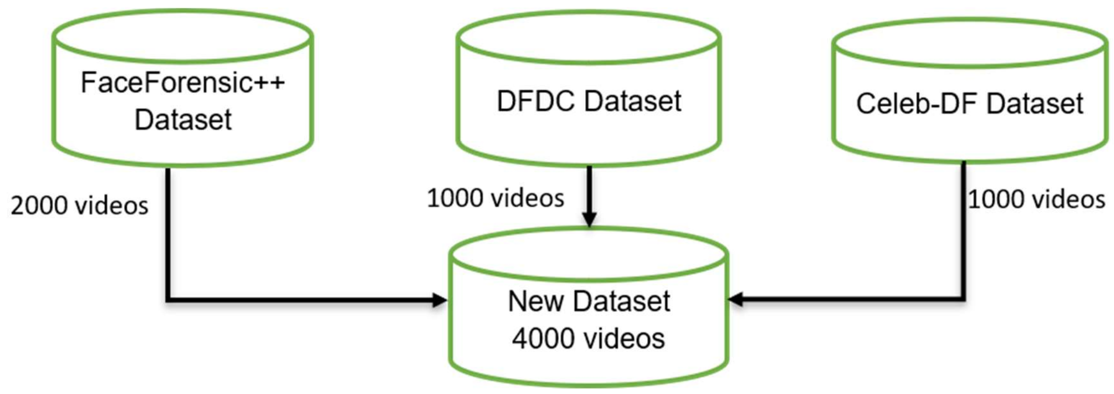


Fig 2.2.1: Dataset

Figure 2.2.1 illustrates the distribution of the datasets, highlighting the number of videos obtained from each dataset and their categorization as real or fake. This diverse dataset composition aims to provide a robust training set that covers a wide range of video types, enabling the model to detect deepfake videos accurately and in real-time.

Preprocessing: In the preprocessing step, the videos are processed to remove unnecessary noise and focus only on the required portion, i.e., the face. The following steps are involved:

1. Splitting the video into frames: The first stage involves dividing the video into separate frames. A single image from the video is represented by each frame.

2. Face detection and cropping: A face detection model or technique is used to detect the face in each frame. When a face is identified, the frame is cropped to just contain the area of the face. By taking this measure, it is made sure that only pertinent facial data is saved for later analysis.

3. Conversion to a new video: After cropping the frames, the cropped frames are combined to form a new video. This new video consists of the sequence of cropped frames, representing the face-only content of the original video.

In the pre-processing step, videos are processed using dlib's face recognition library for face detection and cropping. Leveraging dlib offers high accuracy and efficiency in locating facial features, ensuring robust detection across varying conditions. This approach enables the extraction of relevant facial information, contributing to the effectiveness of the subsequent deepfake detection process.

The mean total frame count of each video is used to determine a threshold value, which is then chosen to address computational constraints and preserve uniformity in the number of frames across videos.

A threshold value of 150 frames, for instance, might be used. This indicates that for processing and analysis purposes, just the first 150 frames of every video are taken into account.

The selection of this threshold is influenced by the computational power available, particularly the GPU's capabilities in the experimental environment. Processing a large number of frames simultaneously can be computationally intensive, so a manageable number of frames is chosen to balance computational efficiency and accurate analysis.

A frame rate of 30 frames per second (fps) and a size of 112 x 112 pixels are used to save the first 150 frames of the freshly made films. This helps to properly use the Long Short-Term Memory (LSTM) model, which works with sequential data, and standardises the video format.

The LSTM can efficiently identify temporal correlations and patterns in the video data for deepfake detection by analysing the frames in order.

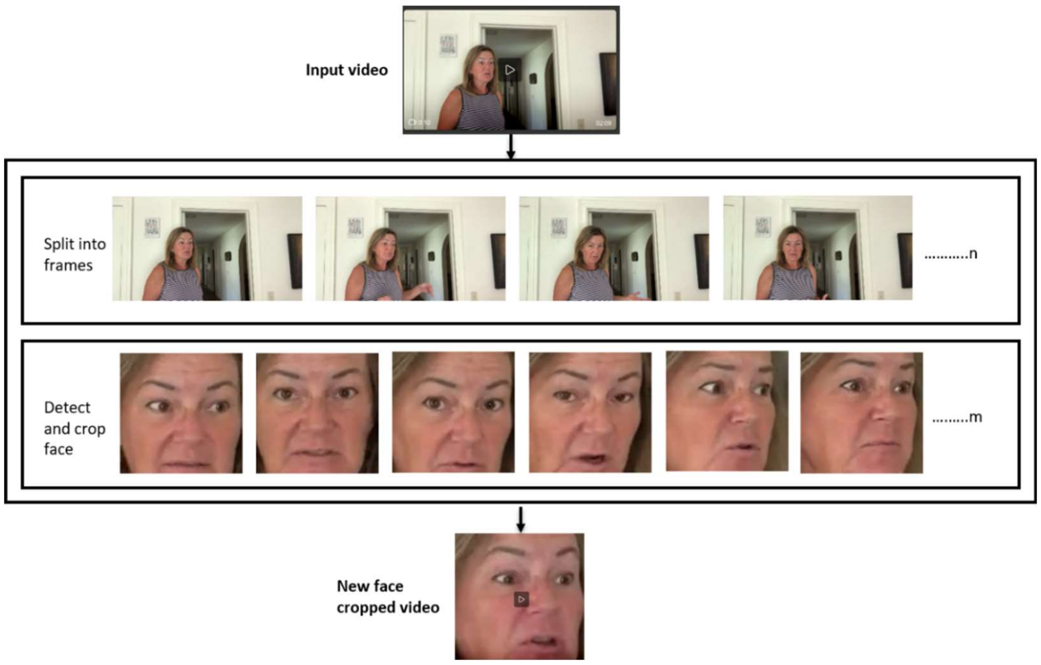


Fig 2.2.2: Pre processing

2.3 Tools and technologies used

Programming frameworks:

- **PyTorch:** The open-source deep learning framework PyTorch is a well-liked tool for creating and refining neural networks. Convolutional neural networks (CNNs) and recurrent neural networks (RNNs), such as long short-term memory (LSTM), are among the deep learning models that may be constructed on it with flexibility and efficiency. With PyTorch, you can easily define models, perform efficient computations, and optimise using gradients thanks to its high-level abstractions and dynamic computational graph.

In the context of deepfake detection, PyTorch is likely used for various tasks such as:

1. Setting up and refining the convolutional neural network (CNN) Res-Next to extract features at the frame level.

2. Constructing and refining the Recurrent Neural Network (RNN) for classification using Long Short-Term Memory (LSTM) as its foundation.

3. Defining custom loss functions and metrics specific to the deepfake detection task.

4. Managing the data pipeline, including loading and preprocessing the video frames.

5. Conducting model evaluation and inference on new videos.

Deep learning model implementation and deepfake detection research are well-suited for PyTorch because to its adaptability, user-friendliness, and robust community support.

IDE:

- **Google colab:** Python code can be written, run, and collaborated on using Google Colab, an online cloud-based integrated development environment (IDE). It provides a Jupyter Notebook-like interface and offers free access to computational resources, including GPU acceleration. Google Colab is widely used in the field of deep learning and machine learning due to its convenience and accessibility.

  In this project, Google Colab is used for several reasons:

  1. GPU Acceleration: Most of the deep learning models frequently need a lot of processing power, particularly when they're being trained on large-scale datasets. GPUs are made available through Google Colab, which speeds up training considerably.

  2. Cloud-based Computing: By using Google Colab, researchers can leverage the cloud infrastructure to perform computationally intensive tasks without relying solely on local resources.

  3. Collaboration: Google Colab facilitates group work on a single notebook, which makes it simpler for developers and academics to exchange and cooperate on projects.

  4. Preinstalled Libraries and Packages: PyTorch, NumPy, pandas, and other well-known Python libraries and packages are preinstalled in Google Colab, which speeds up setup and guarantees compliance with project specifications.

- **Jupyter Notebook:** Jupyter Notebook is an open-source web tool that lets users create and share documents with written explanations, live code, and visualisations. It offers an interactive environment for exploring, analysing, and developing models in data, and supports a number of computer languages, including Python.

Jupyter Notebook is commonly used in machine learning and data science projects, including deepfake detection, for the following reasons:

1. Interactive Development: Jupyter Notebook allows researchers and developers to execute code cells interactively, making it convenient for experimenting with different algorithms, exploring data, and visualizing results.

2. Code Documentation: Jupyter Notebook supports the inclusion of explanatory text, equations, and visualizations alongside the code, providing a well-documented and self-contained environment for sharing project insights and findings.
3. Easy Iteration and Debugging: Jupyter Notebook supports iterative development, allowing developers to modify and rerun specific code cells without restarting the entire notebook. This facilitates quick debugging and experimentation.

4. Data Visualization: Creating and displaying visual representations of data is made easier by Jupyter Notebook's good integration with popular visualisation frameworks like Matplotlib and Seaborn.

Jupyter Notebook is utilised in this deepfake detection project for activities including data preprocessing, model construction, model evaluation, and outcome analysis. It offers an adaptable and interactive environment for investigating and applying deep learning models as well as for examining the results of deepfake detection.

Libraries:

- **torch and torchvision:** torch is the primary library of the PyTorch framework, while torchvision is a PyTorch package that provides tools for image and video processing. These libraries are fundamental to this project as they enable the implementation and training of deep learning models for deepfake detection.

  1. To create and train neural networks, manage tensors, and carry out effective GPU computations, torch offers the required modules and functionalities.

  2. Pre-trained models, datasets, and transformation functions created especially for computer vision tasks—such as processing images and videos—are available through torchvision.

Both libraries are used in this project to:

1.Establish and hone deep learning models for feature extraction and classification, such as the Res-Next CNN and LSTM.

2.For the purpose of transfer learning or feature extraction, start with pre-trained models, like the ones included in torchvision.models.

3. Use picture and video transformations to preprocess the input data, such as scaling, normalisation, and data augmentation.

- **numpy:** Large, multi-dimensional arrays and matrices are supported by the robust Python numerical computing module numpy. Because of its extensive variety of mathematical operations and functions, it is crucial for numerical computations in deep learning and machine learning applications.

In this deepfake detection project, numpy is used for various tasks, including:

1. Handling numerical data in the form of arrays and matrices.
2. executing matrix operations—which are essential to deep learning computations—such as element-wise and dot product operations.
3. Manipulating and reshaping data arrays to match the required input formats of the deep learning models.

- **CV2:** CV2, also referred to as OpenCV (Open Source Computer Vision Library), is a well-liked open-source computer vision library. Face detection and image modification are only two of the many features and techniques it offers for processing images and videos.

In this project, CV2 is used for:

1. Implementing and applying face detection algorithms to identify and extract faces from video frames.
2. Processing images and videos, including scaling, cropping, and colour corrections.
3. Visualizing and displaying images and videos during the preprocessing and analysis stages.

- **matplotlib:** With the help of the popular Python plotting library matplotlib, a variety of visualisations, such as scatter plots, histograms, and charts, can be created. It offers an adaptable and user-friendly interface for data visualisation.

In this project, matplotlib may be used to:

1. Show and examine how data is distributed, including how real and fake videos are distributed within the dataset.

2. Make graphs and plots to display performance metrics, like loss and accuracy, while training and assessing models.

3. Create visualisations that show the true positive and false negative rates, among other outcomes of deepfake detection.

- **face_recognition:** Face recognition can be achieved with ease using the face_recognition Python library. It locates and recognises faces in pictures and videos using deep learning models that have already been trained. In this project, face_recognition is used for:

1. Video frame face detection and cropping, which makes sure that only the face region is taken into account when detecting deepfakes.

2. Extracting facial features or landmarks that can be used for further analysis or comparison.

- **Pandas:** For Python data analysis and manipulation, pandas is a very well-liked library. It offers strong data structures, like DataFrames, and tools for effectively managing structured data.

In this project, pandas may be used for:

1. Preprocessing and loading the video metadata, including timestamps and labels.

2. Organizing and manipulating the metadata to facilitate data exploration and analysis.

3. Combining and merging multiple datasets.

- **sklearn:** A vast array of tools and algorithms for data preprocessing, feature extraction, model training, and evaluation are provided by the extensive Python machine learning library sklearn (scikit-learn).

In this project, sklearn is used for tasks such as:

1. Dividing the dataset into subsets for training and testing in order to assess the model.

2. Applying feature scaling or normalization to the extracted features or input data.

3. Using different algorithms, like support vector machines (SVM) or logistic regression, to train and assess classification models.

- **random:** Python's random module offers functions to create random sequences, random numbers, and sampling. For tasks that call for randomization or involve randomness, it is frequently utilised.

In this project, random is used for:

1. Shuffling and randomizing the order of video frames or the dataset during training to introduce variability.

2. Randomly selecting samples or subsets of data for validation or testing purposes.

3. Adding randomness or noise to the data or model training process, such as data augmentation techniques.

Overall, these libraries are utilized in this deepfake detection project to facilitate a variety of essential tasks. These include implementing deep learning models for accurate detection, preprocessing data to prepare it for analysis, and visualizing data to gain insights into patterns and model performance. Additionally, feature extraction is performed to highlight important data characteristics, and face detection is used to identify faces in video frames. Data manipulation helps organize and format the data correctly, while model evaluation assesses the accuracy and reliability of the deep learning models.

3.Problem Definition.

3.1 Problem statement

The problem statement of our project is to develop a deepfake detection system using an LSTM-based artificial neural network. The goal is to address the increasing concern regarding the widespread creation and dissemination of deepfake content, which can have significant negative implications such as political tension, fake terrorism events, blackmail, and misinformation.

While tools for creating deepfakes have become easily accessible, the detection of deepfakes remains a major challenge. The lack of reliable and efficient deepfake detection tools has allowed these manipulated videos to spread rapidly across social media platforms, leading to potential harm and the distortion of truth.

Our project aims to contribute to the detection and prevention of deepfakes by developing an effective model to detect deepfakes.

To achieve our goal, we plan to expand on the problem statement by considering the following aspects:

1. **Dataset collection:** A wide range of authentic and representative deepfake videos will be gathered. The training and assessment of our deepfake detection system will be based on this dataset. To make sure the model can successfully generalize to various modifications, it is crucial to gather a broad variety of deepfake samples.

2. **Preprocessing and feature extraction:** The videos will be preprocessed in order to extract pertinent features that will aid in the distinction between authentic and deepfake content. Techniques including motion analysis, facial landmark detection, and frame extraction may be used in this. Through the use of significant features to represent videos, we are able to accurately describe the temporal and visual patterns typical of deepfake manipulation.

3. **Architecture for an LSTM-based neural network:** To detect deepfakes, we will create and deploy an artificial neural network based on LSTMs. For simulating sequential data, such as video frames or temporal patterns, Long Short-Term Memory (LSTM) networks operate best. The LSTM network will be trained using the features that were taken out of the dataset so that it can discover the fundamental qualities of deepfake movies.

4. **Model evaluation and training:** The dataset will be split up into testing, validation, and training sets. Using suitable loss functions and optimization strategies, the LSTM model will be trained on the training set. The validation set will be used to choose the model and

adjust the hyperparameters. Lastly, the testing set performance of the trained model will be examined to determine its accuracy.

5. **Performance measurements and analysis:** We'll use a number of metrics, including accuracy, precision, recall, and F1 score, to evaluate how well our deepfake detection algorithm performs. To find any particular weaknesses or vulnerabilities, we may also examine the model's performance under various deepfake manipulations.

3.2 Project objectives

1. To conduct literature survey on deepfakes detection techniques.
2. To develop the required dataset by pre-processing available datasets.
3. To Train the ResNeXt architecture neural network using the developed dataset.
4. To analyze and evaluate the developed model.
5. To mitigate potential vulnerabilities and limitations of the developed model.

Our project aims to address the increasing concern surrounding the creation and dissemination of deepfake content by developing an effective deepfake detection system. We begin by conducting a comprehensive literature survey to gain a deep understanding of existing deepfake detection techniques and methodologies. We then focus on developing a diverse and representative dataset by preprocessing available datasets, ensuring it contains both real and deepfake videos.

We train a ResNeXt architecture neural network with this dataset, taking use of its skills in picture categorization tasks. By examining the retrieved attributes, the network has the ability to differentiate between authentic and deepfake films. We carefully examine and assess the created model, determining any vulnerabilities and limits and gauging its performance using a range of measures.

In order to address these problems, we investigate methods to improve the resilience and generalization capacities of the model. Our main goal is making a significant contribution to the field of deepfake detection by offering a dependable and practical way to lessen the negative effects of deepfake content and shield people and communities from the false information and manipulation that can have damaging repercussions.

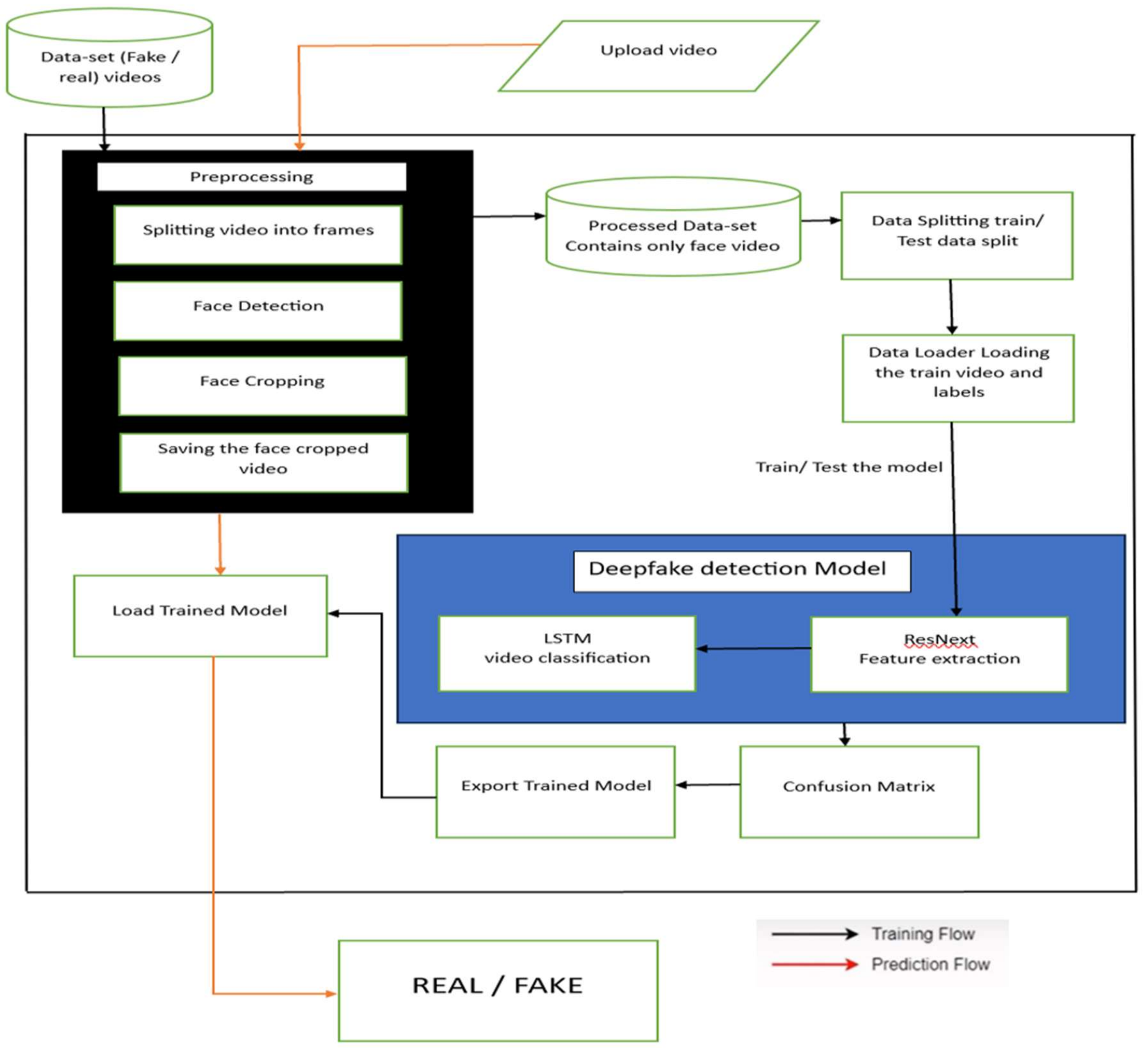
3.3 Methods and Methodology

Objective	Methodology	Description
1.	Literature Survey	Conduct survey on various deep learning and ML approaches on deepfake generation and deepfake detection.
2.	Developing dataset required for training the model	Create a dataset of videos by gathering and combining the datasets available.
3.	Preprocessing the dataset	Preprocessing is done on the gathered dataset to make necessary changes to detect deepfake.
4.	Training the ResNeXt CNN with LSTM for prediction	Uploading the created dataset to the architecture to train the model.
5.	Testing the developed model for prediction	Test the trained model with new untrained data.
6.	Evaluation using standard metrics	Using standard metrics like accuracy, precision and recall to analyze model performance.
7.	Recalibrating the model	Investigate potential biases in the model predictions and optimize the model's performance under such conditions.
8.	Scientific Report	A report highlighting all the features of the proposed system is documented as per the university template.

Tabel 3.3.1 - Methods and Methodology

4.Problem Solving.

4.1 Introduction



Data-set (Fake / real) videos
Upload video
Preprocessing
Splitting video into frames
Face Detection
Face Cropping
Saving the face cropped video
Processed Data-set Contains only face video
Data Splitting train/ Test data split
Data Loader Loading the train video and labels
Train/ Test the model
Deepfake detection Model
LSTM video classification
ResNext Feature extraction
Load Trained Model
Export Trained Model
Confusion Matrix
REAL / FAKE
Training Flow
Prediction Flow

Fig 4.1.1: Block diagram of deepfake detection

A deepfake detection system's architecture and operation are depicted in the block diagram. There are two primary sections to it: the prediction flow and the training flow. The steps involved in training the deepfake detection model are presented in the training flow section. Data preprocessing is the first step, in which methods like motion analysis, frame extraction, and facial landmark detection are used to prepare the dataset. These methods assist in removing pertinent elements from the videos, which are essential for differentiating between authentic and deepfake material.

The model training phase starts after the dataset has been preprocessed. Using the prepared dataset, the ResNeXt architecture-based deepfake detection model is trained. During this process, the model's parameters are optimised using the proper loss functions and optimisation strategies. The objective of the training phase is to make the model capable of understanding the fundamental traits and structures of both authentic and deepfake videos.

The model evaluation phase of the training flow comes after the model has been trained. Evaluation metrics like accuracy, precision, recall, and F1 score are used to gauge how well the trained model performs. This assessment sheds light on the advantages and disadvantages of the deepfake detection model and aids in determining its efficacy.

In the prediction flow section, the procedures for utilising the trained model for deepfake detection in real-time scenarios are illustrated. In the data input stage, the trained model is fed video data. Either genuine or possibly manipulated deepfake content can be found in this data.

In the model prediction phase, the input video data is analysed using the trained deepfake detection model. The model determines whether the input video is a deepfake or real by using the features and patterns it has learned. This forecast is predicated on the model's capacity to discern between the traits unique to deepfakes and those found in real videos.

The outcome of the deepfake detection procedure is finally shown in the output stage. It displays the classification of the input video—real or deepfake. Users can identify and take appropriate action against potentially harmful or misleading content with the help of this output, which offers valuable information.

In general, the block diagram offers a summary of the steps and elements needed to train and operate a deepfake detection system. It serves as an example of how crucial data preprocessing, model training, assessment, and prediction are to the efficient detection and identification of deepfake content.

4.2 Problem solving approach

It is true that comprehension of the deepfake video production process is essential for successful detection. When using tools like GANs and autoencoders, a source image and a target video are usually used. After splitting the video into frames, the target face is substituted for each frame's face. To combine the replaced frames and improve the overall quality of the deepfake video, pretrained models are used to eliminate any traces left over from the creation process.
We employ a similar methodology in our deepfake detection approach. We admit that pretrained neural network-generated deepfakes can resemble real videos quite a bit,

making it challenging to distinguish between them visually. But frequently, these deepfake creation tools leave behind minute traces or artefacts that might not be immediately apparent. To accurately distinguish between deepfake and real videos, our goal is to find these subtle traces and artefacts. Our goal is to create a strong deepfake detection system by examining the distinct traits and patterns found in deepfakes. The system will utilise its comprehension of these subtle traces and artefacts to efficiently categorise videos as authentic or deepfakes. Our method goes beyond visual inspection by revealing the hidden signs of deepfake manipulation through the use of sophisticated algorithms and models.

A balanced split of 70% for training and 30% for testing separates the dataset into train and test sets. With this division, each set will have an equal amount of real and deepfake videos.

We combine CNN and RNN architectures in our deepfake detection model. For feature extraction, we employ a pre-trained ResNext CNN model. Our experiment used a ResNext model with 50 layers and 32x4 dimensions. We are able to efficiently extract pertinent features from the video frames by utilising the pre-trained ResNext model. We use an LSTM network to classify the videos as real or deepfake. The LSTM network's input consists of 2048-dimensional feature vectors that were extracted from the final pooling layers of the ResNext model. With a dropout rate of 0.4 to avoid overfitting, the LSTM layer has 2048 latent dimensions and 2048 hidden layers. By processing the frames in this manner, the LSTM is able to perform temporal analysis by comparing frames taken at various times.

The model additionally integrates a linear layer with 2048 input features and 2 output features, as well as a Leaky ReLU activation function, to facilitate the learning of correlations between the input and output. The desired output size is attained by using an adaptive average pooling layer with an output size of 1. Batch training uses a 4-batch batch size and a sequential layer. The model's confidence during prediction is obtained using a SoftMax layer.

The overall strength of RNNs—more especially, LSTM networks—for sequential analysis of video frames and CNNs for feature extraction are combined in our deepfake detection model. The model is adjusted to maximise its efficiency and allow for precise categorization of videos as authentic or deepfake.

Hyperparameter tuning:

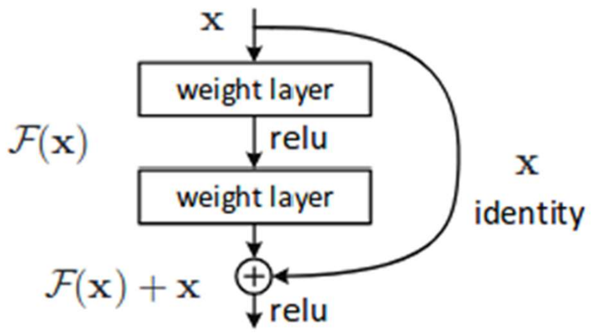
A deepfake detection model's hyper-parameter tuning is a crucial step in maximising accuracy. Based on the dataset and model performance, it entails choosing the best values for a variety of hyper-parameters. The user message discusses using an adaptive learning rate Adam optimizer with a weight decay of 1e-3 and a learning rate of 1e-5 (0.00001). In order to maximise convergence and reduce loss during training, these values have been adjusted.

Cross-entropy, which works well for classification issues like deepfake detection, is employed as the loss function. In this particular development environment, a batch size of 4 is selected for batch training in order to optimise the computational resources.

Several parameters that are utilised as features in the deepfake detection model are identified. Eye blinking, teeth enchantment, greater distances between eyes, moustaches, double edges, eyes, ears, and nose, iris segmentation, facial wrinkles, uneven head posture, face angle, skin tone, facial expressions, lighting, various positions, double chins, hairstyles, and higher cheekbones are some of these parameters. These parameters are probably used to record different patterns and visual cues that can be used to differentiate between deepfake and real videos.

Model details:

1. ResNeXt CNN: Specifically, the resnext50_32x4d model of the ResNext CNN is employed in the deepfake detection system. The dimensions of this pre-trained model are 32 by 4, with 50 layers. With its strong convolutional neural network architecture, deeper networks can be handled with efficiency.



x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Fig 4.2.1: ResNeXt working

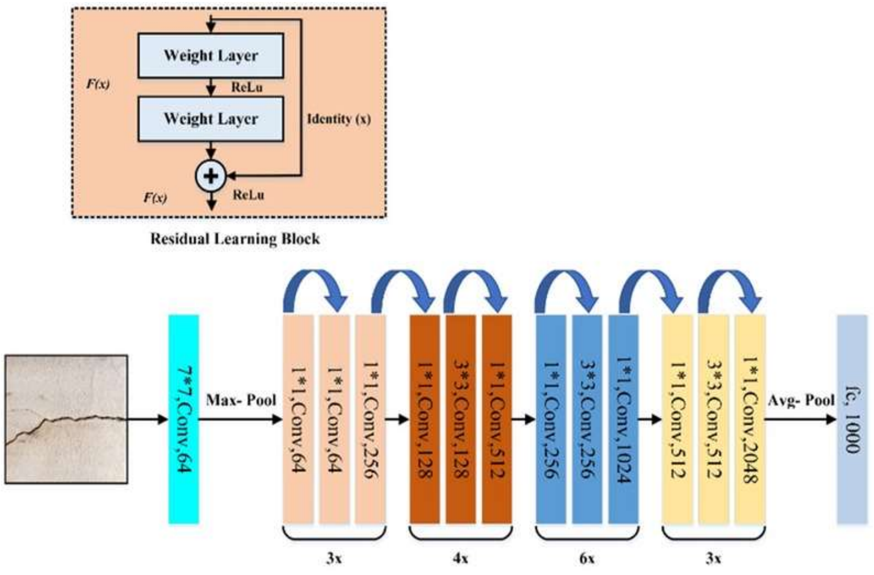


Fig 4.2.2: ResNeXt architecture

2. Sequential layer: Using the Sequential layer, the feature vectors that the ResNext model returns are organized and stored in a container. As a result, features can be passed sequentially to the LSTM layer.

3. LSTM layer: This layer is used for temporal change capture between video frames and sequence processing. The ResNext model's 2048-dimensional feature vectors are used as the LSTM layer's input. Here, a dropout rate of 0.4 is employed to avoid overfitting and a single LSTM layer with 2048 latent dimensions and 2048 hidden layers is employed. Through the sequential processing of the frames by the LSTM layer, temporal analysis is made possible by comparing the frame at time 't' with frames from earlier time occurrences 't-n', where 'n' can be any number of frames prior to 't'.

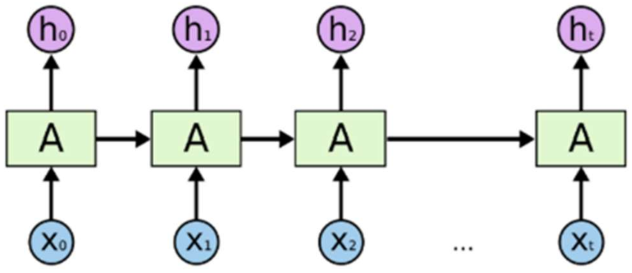


Fig 4.2.3: Overview of LSTM architecture

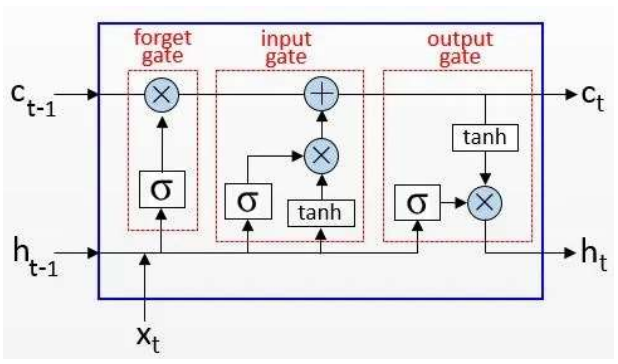


Fig 4.2.4: Internal LSTM architecture

4. ReLU: The activation function in the model is called ReLU (Rectified Linear Unit). ReLU keeps the raw input value in all other cases and outputs 0 for input values less than 0. It is a non-linear activation function designed to resemble how biological neurons might behave. ReLU has no vanishing gradient issue related to the sigmoid function and is computationally efficient.

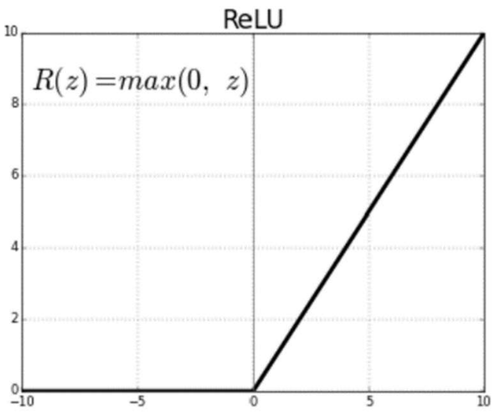


Fig 4.2.5: ReLU activation function

5. Dropout layer: The deepfake detection model uses the dropout layer as a crucial part to avoid overfitting. Here, a dropout layer is used, and its dropout rate is 0.4. The dropout layer strengthens and reduces the sensitivity of the model to the individual weights of the neurons by randomly removing neurons during training. As a result, the model is motivated to acquire more universal characteristics that may be applied to a wider variety of inputs.

6. Adaptive Average Pooling layer: The Adaptive Average Pooling layer is a technique that lowers computational complexity and extracts low-level characteristics from the neighbourhood. Here the model has a 2-dimensional Adaptive Average Pooling layer. Localised feature map information is compiled and summarised with the aid of this layer.

Overall, the combination of ResNext CNN, Sequential layer, LSTM layer, ReLU activation function, and Adaptive Average Pooling layer contributes to the deepfake detection model's ability to effectively process video frames and capture temporal information for accurate classification.

Model training details

1.Test train split: The dataset is divided into train and test datasets for the project, with a ratio of 70% for the train dataset and 30% for the test dataset. There are an equal number of real and fake videos in each split because the split is balanced.

2.Data loader: The videos and their corresponding labels are loaded using a data loader with a batch size of four. As a result, data processing during training can be done effectively.

3.Training: 30 epochs of training are conducted using the Adam optimizer, a weight decay of 1e-3 (0.001), and a learning rate of 1e-5 (0.00001). Deep learning models frequently employ the adaptive learning rate optimisation algorithm known as the Adam optimizer.

4.Cross entropy: The cross-entropy method is used to compute the loss function during training. For classification tasks, such as differentiating between authentic and fraudulent videos, cross-entropy is a frequently used loss function.

5.SoftMax layer: In the neural network model, the SoftMax layer is applied as the last layer. Squashing functions like the SoftMax function restrict the function's output to the interval 0 to 1. Because it gives probabilities for each class prediction, it is frequently used in multi-class classification problems. The two output nodes in this instance of the SoftMax layer indicate the classes "REAL" and "FAKE," and they offer the probability or confidence of each prediction.

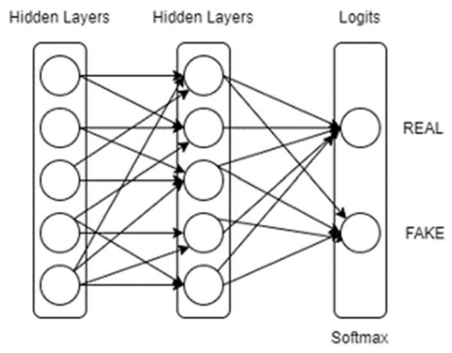


Fig 4.2.6: SoftMax layer

Important code snippets

```
import json
```

```python
import glob
import numpy as np
import cv2
import copy
import torch
import torchvision
from torchvision import transforms
from torch.utils.data import DataLoader, Dataset
import os
import matplotlib.pyplot as plt
import face_recognition
from tqdm.autonotebook import tqdm

# Function to extract frames from video
def get_frames(vid_path):
    vid_cap = cv2.VideoCapture(vid_path)
    success = True

    while success:
        success, frame = vid_cap.read()
        if success:
            yield frame

# Function to create face-only videos
def generate_face_videos(vid_files, save_dir):
    existing_vids = glob.glob(os.path.join(save_dir, '*.mp4'))
    print("Number of existing videos:", len(existing_vids))

    for vid_file in tqdm(vid_files):
        save_path = os.path.join(save_dir, os.path.basename(vid_file))

        if glob.glob(save_path):
            print("File already exists:", save_path)
            continue

        frames_buffer = []
        vid_writer = cv2.VideoWriter(save_path,
cv2.VideoWriter_fourcc('M','J','P','G'), 30, (112, 112))

        for idx, frame in enumerate(get_frames(vid_file)):
            if idx <= 200:
                frames_buffer.append(frame)

                if len(frames_buffer) == 4:
                    face_locations =
face_recognition.batch_face_locations(frames_buffer)
                    for i, face_loc in enumerate(face_locations):
                        if face_loc:
                            top, right, bottom, left = face_loc[0]

                            try:
                                face_frame = 
cv2.resize(frames_buffer[i][top:bottom, left:right, :], (112, 112))
                                vid_writer.write(face_frame)

                            except Exception as e:
                                print(f"Error processing frame: {e}")
                    frames_buffer = []

        vid_writer.release()

# !mkdir '/content/drive/My Drive/FF_REAL_Face_only_data'
output_path = '/content/drive/My Drive/FF_REAL_Face_only_data'
generate_face_videos(vid_files, output_path)
```

The provided code performs the following tasks:

1. Defines a function called frame_extract(path) that takes a video file path as input and extracts frames from the video using OpenCV's VideoCapture and read functions. The function uses a generator to yield each frame as it is read.

2. Defines a function called create_face_videos(path_list, out_dir) that creates face videos from a list of video file paths. It takes a list of video file paths and an output directory as input.

3. Checks if any videos already exist in the output directory by counting the number of files with the ".mp4" extension.

4. Iterates over each video file path in the provided list and extracts frames from the videos.

5. For each frame, it performs the following steps:
   - Appends the frame to a list called frames.
   - Checks if the length of frames is equal to 4 (indicating that 4 frames have been collected).
   - Uses the face_recognition library to detect faces in the collected frames.
   - Resizes the face regions to a size of (112, 112) pixels.
   - Writes the resized face frames to an output video file using OpenCV's VideoWriter function.

The processed frames are saved as face videos in the specified output directory.
Overall, the code takes input video files, detects faces in the frames, crops and resizes the face regions, and saves the processed frames as face videos in the output directory.

```
def evaluate_model(model, dataloader):
    model.eval()
    running_corrects = 0
    all_preds = []
    all_labels = []

    with torch.no_grad():
        for inputs, labels in dataloader:
            inputs = inputs.cuda()
            labels = labels.cuda()

            _, outputs = model(inputs)
            _, preds = torch.max(outputs, 1)

            running_corrects += torch.sum(preds == labels.data)
            all_preds.extend(preds.cpu().numpy())
            all_labels.extend(labels.cpu().numpy())

    accuracy = running_corrects.double() / len(dataloader.dataset)
    print(f'Validation Accuracy: {accuracy:.4f}')
    return all_preds, all_labels

val_preds, val_labels = evaluate_model(model, val_loader)

# Compute confusion matrix
cm = confusion_matrix(val_labels, val_preds)
print('Confusion Matrix')
print(cm)

# Classification report
report = classification_report(val_labels, val_preds,
target_names=['REAL', 'FAKE'])
print('Classification Report')
print(report)

# Plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=['REAL',
'FAKE'], yticklabels=['REAL', 'FAKE'])

plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.title('Confusion Matrix')
plt.show()
```

The provided code defines a function called evaluate_model which takes a model, an image, and some optional parameters as input. Here's a breakdown of the code:

1. Function Definition: Define a function named evaluate_model that takes two arguments: model (the trained model) and dataloader (the validation data loader).

2. Evaluation Loop: Iterate through the validation data loader. For each batch:
    - Transfer inputs and labels to GPU.
    - Pass inputs through the model.
    - Apply softmax to get probabilities and obtain predictions.
    - Update running_corrects by adding the number of correct predictions in the batch.
    - Extend all_preds and all_labels lists with predictions and labels for later analysis.

3. Compute Accuracy: Calculate accuracy by dividing the total number of correct predictions (running_corrects) by the total number of samples in the dataset. Print the validation accuracy.

4. Return Predictions and Labels: Return all_preds (all predictions) and all_labels (all true labels) for further analysis.

5. Evaluation: Call the evaluate_model function with the trained model (model) and the validation data loader (val_loader). Store the returned predictions and labels in variables val_preds and val_labels.

6. Confusion Matrix and Classification Report: Compute the confusion matrix using confusion_matrix from sklearn.metrics. Print the confusion matrix. Generate a classification report using classification_report from sklearn.metrics. Print the classification report.

7. Plot Confusion Matrix: Plot the confusion matrix using sns.heatmap for better visualization. Customize the plot with labels and a title. Show the plot.

In summary, the code performs a prediction using a given model and image, visualizes the result with a heatmap overlay, and returns the predicted class and confidence if the confidence is above a threshold.

5. Results

5.1 Test results examples

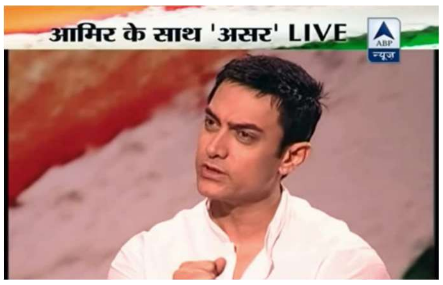
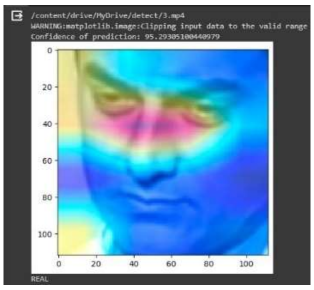


Fig 5.1.1: Real input example



```
/content/drive/MyDrive/detect/3.mp4
WARNING:matplotlib.image:Clipping input data to the valid range
Confidence of prediction: 95.29305100440979
```

Fig 5.1.2: Real prediction

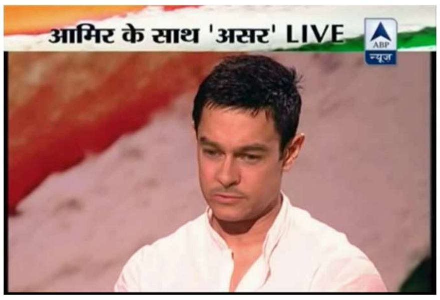
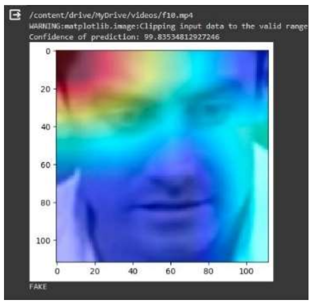


Fig 5.1.3: Fake input examples



```
/content/drive/MyDrive/videos/f10.mp4
WARNING:matplotlib.image:Clipping input data to the valid range
Confidence of prediction: 99.83534812927246
```

Fig 5.1.4: Fake prediction

Using the input, the model applies a softmax function, computes logits, creates feature maps, and makes predictions. Additionally, it overlays a heatmap on the input image to visualize the prediction. The model and the uploaded image are inputs into the predict function. After running the picture through the model, it determines the prediction's confidence. It returns the predicted class and confidence as a list if the confidence is higher than a predetermined threshold.

Figures 5.1.2 and 5.1.4 are the model's anticipated outputs, and Figures 5.1.1 and 5.1.3 are the model's inputs. In the example above, the model accurately predicted the first input, which was real, and the second, which was a fake. This shows that the model is able to differentiate between authentic and fraudulent inputs and is generating accurate predictions in this particular situation. The model's functionality and its capacity to identify real from fake images based on inputs are demonstrated by the provided code and analysis.

5.2 Test analysis

The training and validation losses are calculated during the training process of a machine learning model. Here's a brief explanation of how these losses is computed:

1. **Training Loss:** For the training data, the training loss is a measurement of the difference between the expected and actual outputs. Usually, a loss function, like cross-entropy loss or mean squared error (MSE), is used to calculate it. Log loss, or cross-entropy loss, is a widely used technique in classification tasks, such as binary and multi-class classification. The difference between the true probability distribution of the target classes and the predicted probability distribution is quantified by cross-entropy loss. Throughout the optimization process (such as backpropagation and gradient descent), the model's parameters are modified in response to this loss in order to reduce error and enhance the model's performance on the training set.

In binary classification, cross-entropy loss is computed using this formula:

$$\text{Cross-Entropy Loss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i)]$$

And for multi-class classification:

$$\text{Cross-Entropy Loss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c} \cdot \log(p_{i,c})$$

where C is the number of classes, N is the number of samples, yi is the true label, and pi is the predicted probability.

1. **Validation Loss:** The second type of loss is called validation loss, which is calculated using a different validation dataset but is comparable to the training loss. This dataset is meant to be a benchmark for assessing how well the model generalizes to new data; it is not used for training the model. A measure of the model's performance on fresh examples and a tool for evaluating its generalizability are provided by the validation loss. It can be used to track the model's development during training and is computed using a loss function, such as cross-entropy loss. The model's parameter updates are not, however, directly impacted by the validation loss.

For the purpose of tracking and assessing a machine learning model's performance, training and validation losses are both crucial metrics. The model seeks to reduce these losses in order to increase prediction accuracy and enhance overall performance by improving its ability to generalize to new data.

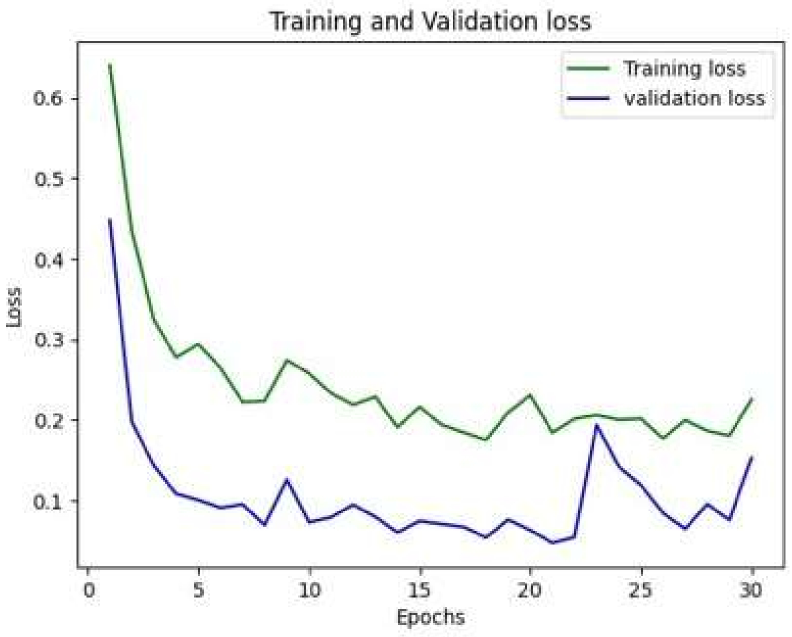


Fig 5.2.1 – Loss of the model

A machine learning model's performance is indicated by its average loss of 0.2417. It is possible to see that the model's training and validation losses get smaller over time from the provided image. The validation loss is represented by the green line, and the training loss is shown by the blue line. The graph indicates that as the losses consistently drop, the model is getting better. A particular point on the graph corresponds to the average loss value of 0.2417. Because of the relatively small loss, this value shows that the model is operating well. The graph shows the model's overall development and its capacity to minimize losses, which is a crucial component in maximizing the model's performance.

Accuracy: Accuracy is a frequently employed metric for assessing how well machine learning models perform. It calculates the percentage of the model's accurate predictions on a particular dataset. Comparing the model's predicted labels with the dataset's true labels yields the accuracy calculation.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

Put more simply, accuracy is the ratio of the number of accurate predictions to the total number of predictions the model made. It offers an overall indicator of the model's predictive accuracy.

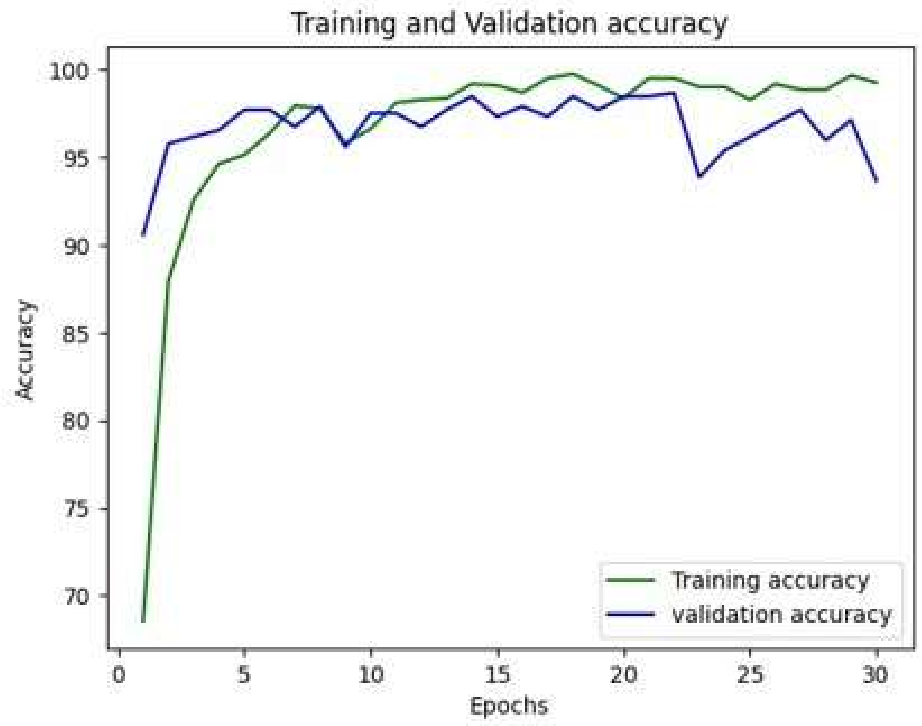


Fig 5.2.2 – Accuracy of the model

Our model achieves a maximum accuracy of 98.656%, and an average accuracy of 96.788 % is calculated. The accuracy values represent a machine learning model's performance in terms of its ability to generate accurate predictions. The model's ability to correctly classify or predict the target variable improves with increasing accuracy. The model's highest accuracy of 96.656 percent indicates that its predictions are very accurate. Furthermore, it can be concluded that the model performs well all throughout the dataset, with an average accuracy of 96.788 %. These accuracy values offer trustworthy insights into the model's predictive abilities and show how well the model works in producing accurate results.

Confusion matrix: A confusion matrix is a table that's used to assess how well a classification model works. The model's predictions and their agreement with the actual class labels are broken down in great detail. To represent true positives (TP), false positives (FP), false negatives (FN), and true negatives (TN), the matrix is divided into four quadrants.

- True Positives (TP): These are the cases where the model accurately predicts a positive outcome. For instance, TP stands for the total number of instances that are correctly classified as belonging to the positive class in a binary classification problem.

- False Positives (FP): The instances in which the model predicts something negatively are called false positives. Stated differently, the model forecasts a positive class when the real class is negative.

- False Negatives (FN): False Negatives, or situations that the model predicts as negative but is actually incorrect. When the real class is positive, the model predicts the negative class.

- True Negatives (TN): These are the cases where the model accurately predicts a negative outcome. In a binary classification problem, for instance, TN stands for the total number of instances that were correctly assigned to the negative class.

Important information for assessing the model's performance and computing different assessment metrics can be found in the values in the confusion matrix.

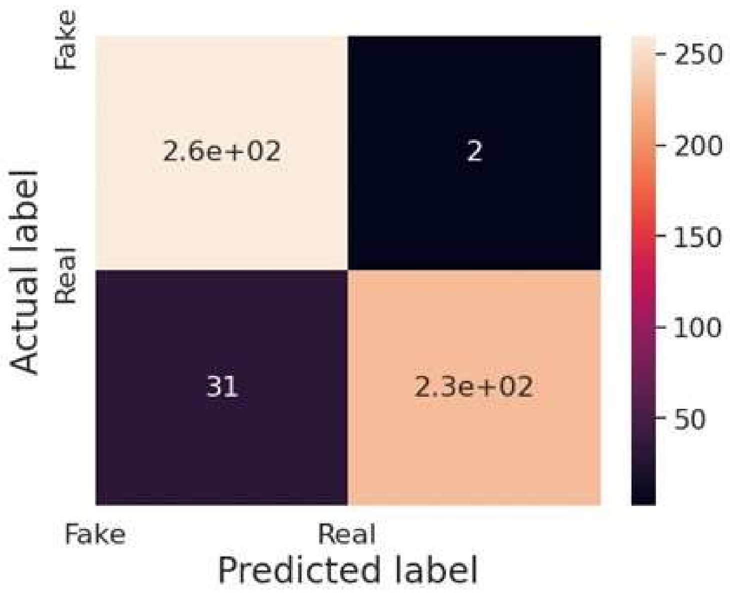


Fig 5.2.3 – Confusion matrix

[260   2]
[31 228]
True positive = 260
False positive = 2
False negative = 31
True negative = 228

Value for true positive is 260, which denotes the proportion of correctly predicted instances in the positive class—which, in this instance, might be the "Fake" class. The number of cases wrongly predicted as belonging to the positive class when in fact they belong to the negative class is represented by the false positive value of 2, which is 2. Thirty-one is the false negative value, which indicates the number of cases that were misclassified as positive when they should have been in the negative class. Ultimately, the number of instances correctly predicted as the negative class is indicated by the true negative value, which is 228, in this case.

Recall: Recall, also known as sensitivity, measures the model's ability to correctly identify all actual positive instances. It quantifies the proportion of true positive instances captured by the model among all actual positive instances in the dataset. A high recall value indicates a low false negative rate, meaning the model effectively captures positive instances without missing them. Recall is calculated using the formula:

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

Precision: The percentage of actual positive predictions among all of the model's positive predictions is known as precision. It is especially helpful when the cost of false positives is high because it shows how well the model can predict positive outcomes. A high precision value indicates a low false positive rate, which means the model correctly detects positive cases without misidentifying negative ones as positive. The formula is used to calculate precision:

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

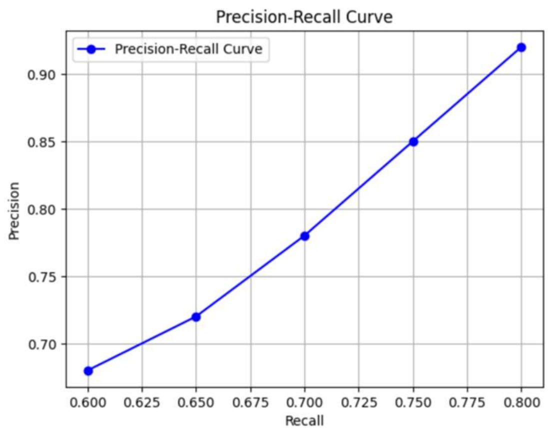


Fig 5.2.4: Precision – Recall curve

In the provided data, the precision value of 0.99 implies that 99% of the positive predictions made by the model are indeed correct, with only a small fraction being false positives.

With just a small percentage of positive predictions being false positives, the precision value of 0.99 in the data indicates that 99% of the positive predictions made by the model are in fact correct.

F1 Score: The F1 score offers a fair assessment of a model's performance and is calculated as the harmonic mean of recall and precision. It is especially helpful in situations where there is an imbalance between positive and negative occurrences or when the costs associated with false positives and false negatives are different. It takes into account both false positives and false negatives. The following formula is used to get the F1 score:

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

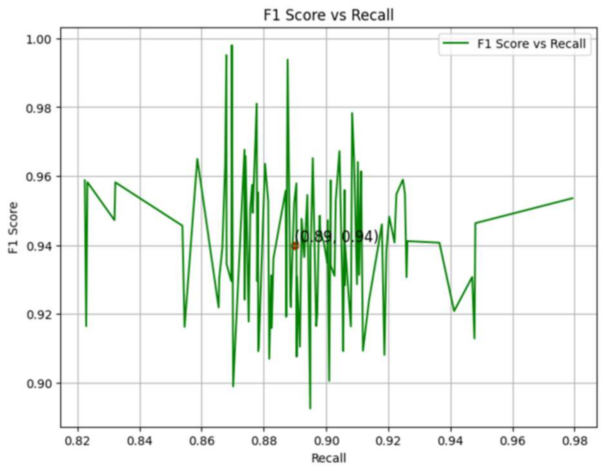


Fig 5.2.5: F1 score – Recall curve

In this instance, the F1 score comes out to 0.94 with precision and recall values of 0.99 and 0.89, respectively. This number shows a fair evaluation of the model's performance that takes into account both recall and precision.

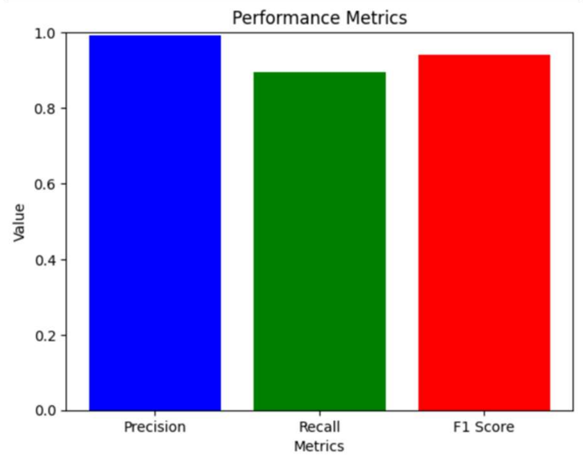


Fig 5.2.6 : Performance metrics

5.3 Prediction results

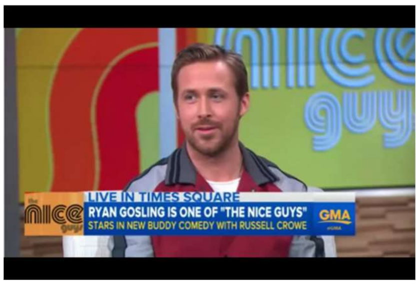

LIVE IN TIMES SQUARE
RYAN GOSLING IS ONE OF "THE NICE GUYS"
STARS IN NEW BUDDY COMEDY WITH RUSSELL CROWE
GMA

Fig 5.3.1: Input 1

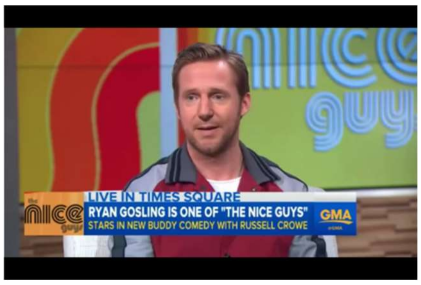
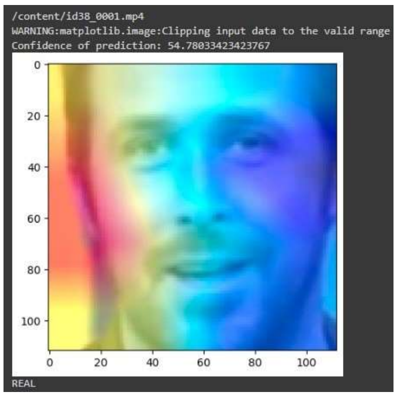
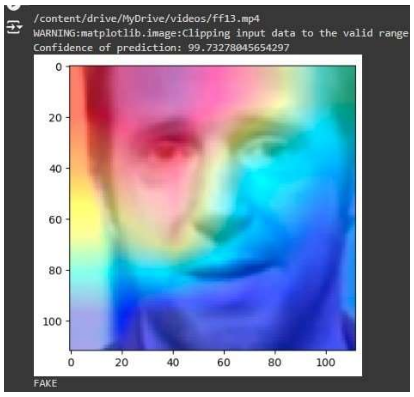

LIVE IN TIMES SQUARE
RYAN GOSLING IS ONE OF "THE NICE GUYS"
STARS IN NEW BUDDY COMEDY WITH RUSSELL CROWE
GMA

Fig 5.3.2: Input 2



```
/content/id38_0001.mp4
WARNING:matplotlib.image:Clipping input data to the valid range
Confidence of prediction: 54.78033423423767
```

Fig 5.3.3: Output 1



```
/content/drive/MyDrive/videos/ff13.mp4
WARNING:matplotlib.image:Clipping input data to the valid range
Confidence of prediction: 99.73278045654297
```

Fig 5.3.4: Output 2

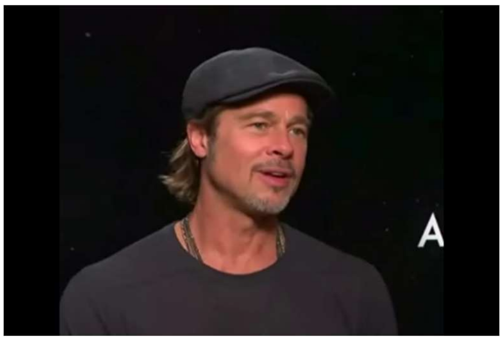


Fig 5.3.5: Input 3

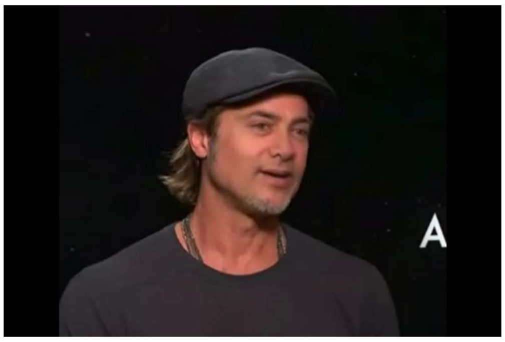
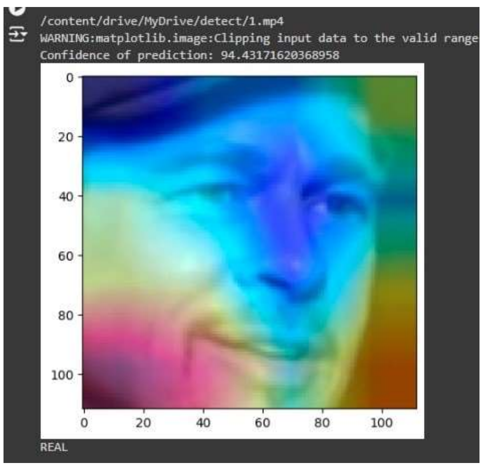
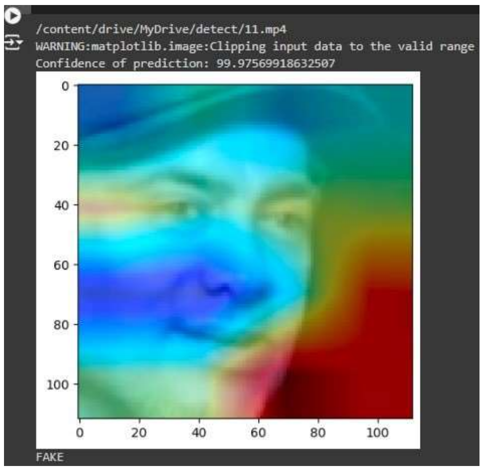


Fig 5.3.6: Input 4



```
/content/drive/MyDrive/detect/1.mp4
WARNING:matplotlib.image:Clipping input data to the valid range
Confidence of prediction: 94.43171620368958
```

Fig 5.3.7: Output 3



```
/content/drive/MyDrive/detect/11.mp4
WARNING:matplotlib.image:Clipping input data to the valid range
Confidence of prediction: 99.97569918632507
```

Fig 5.3.8: Output 4

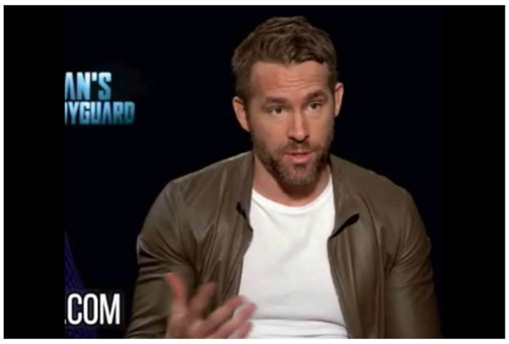


Fig 5.3.9: Input 5

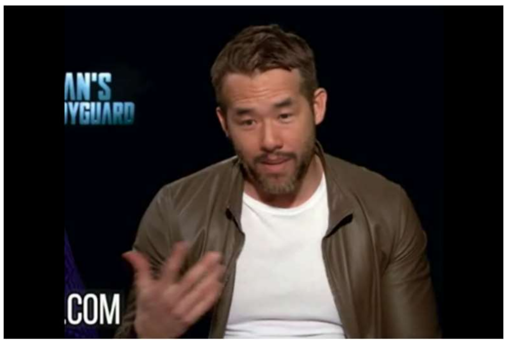
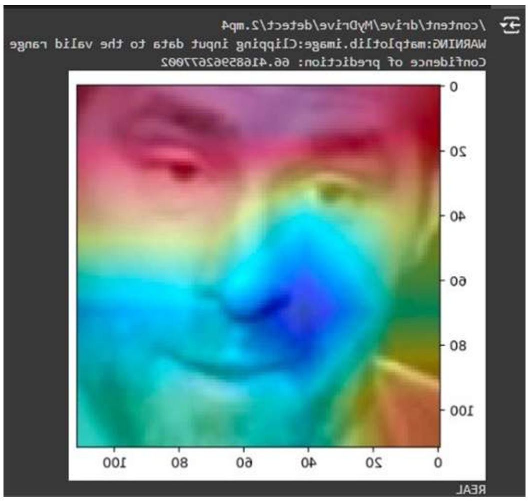
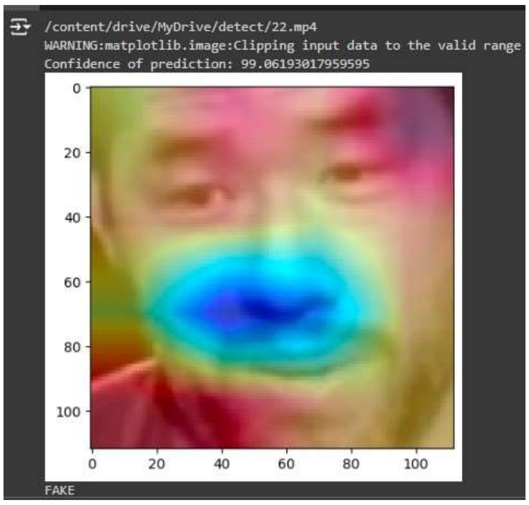


Fig 5.3.10: Input 6



```
/content/drive/MyDrive/detect/2.mp4
WARNING:matplotlib.image:Clipping input data to the valid range
Confidence of prediction: 66.41685962677002
```

Fig 5.3.11: Output 5



```
/content/drive/MyDrive/detect/22.mp4
WARNING:matplotlib.image:Clipping input data to the valid range
Confidence of prediction: 99.06193017959595
```

Fig 5.3.12: Output 6

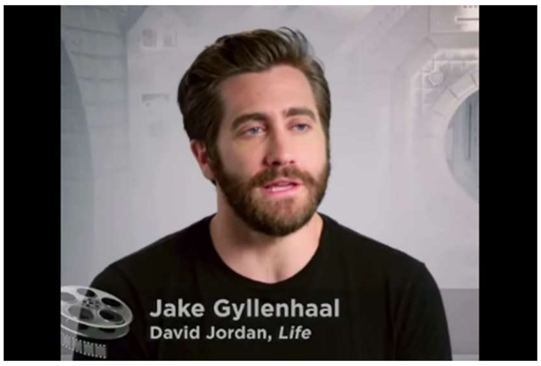


Fig 5.3.13: Input 7

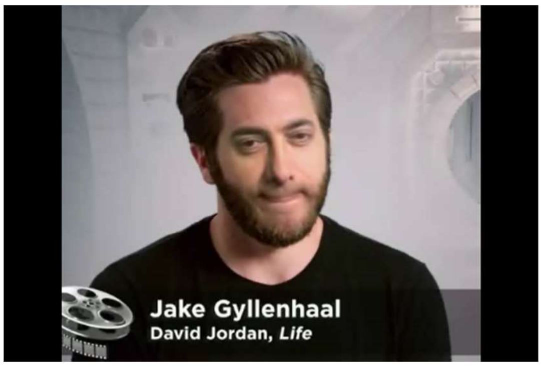
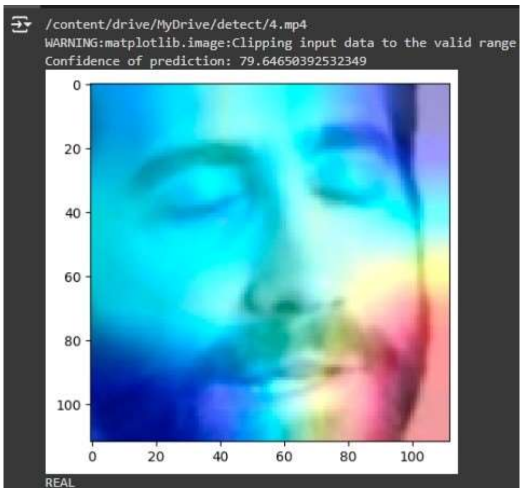


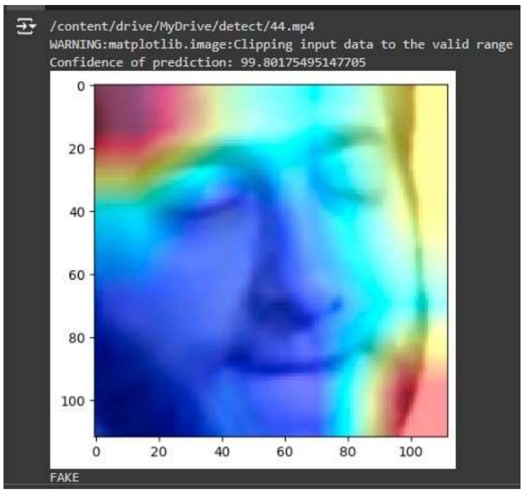
Fig 5.3.14: Input 8



```
/content/drive/MyDrive/detect/4.mp4
WARNING:matplotlib.image:Clipping input data to the valid range
Confidence of prediction: 79.64650392532349
```

```
REAL
```

Fig 5.3.15: Output 7



```
/content/drive/MyDrive/detect/44.mp4
WARNING:matplotlib.image:Clipping input data to the valid range
Confidence of prediction: 99.80175495147705
```

```
FAKE
```

Fig 5.3.16: Output 8

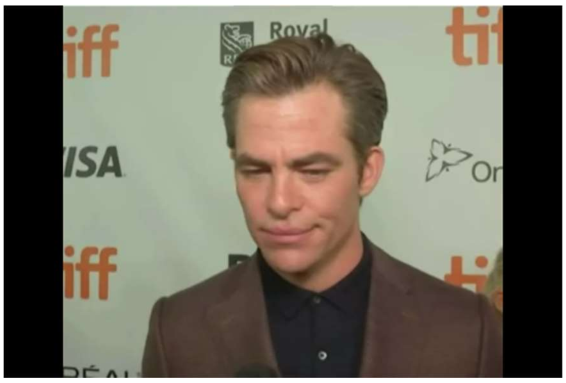


Fig 5.3.17: Input 9

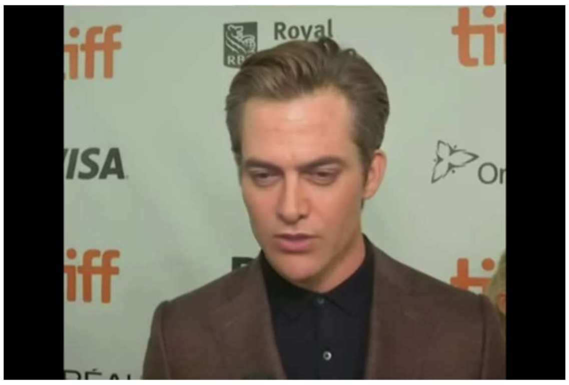
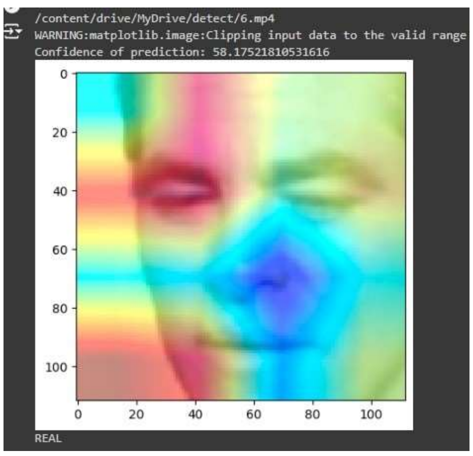
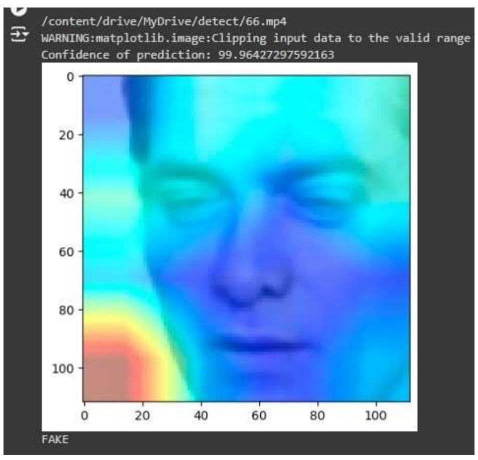


Fig 5.3.18: Input 10



```
/content/drive/MyDrive/detect/6.mp4
WARNING:matplotlib.image:Clipping input data to the valid range
Confidence of prediction: 58.17521810531616
```

Fig 5.3.19: Output 9



```
/content/drive/MyDrive/detect/66.mp4
WARNING:matplotlib.image:Clipping input data to the valid range
Confidence of prediction: 99.96427297592163
```

Fig 5.3.20: Output 10

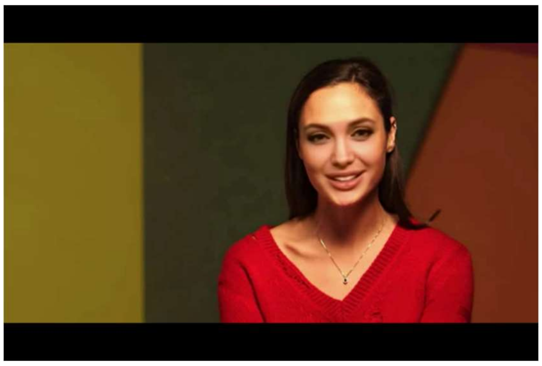


Fig 5.3.21: Input 11

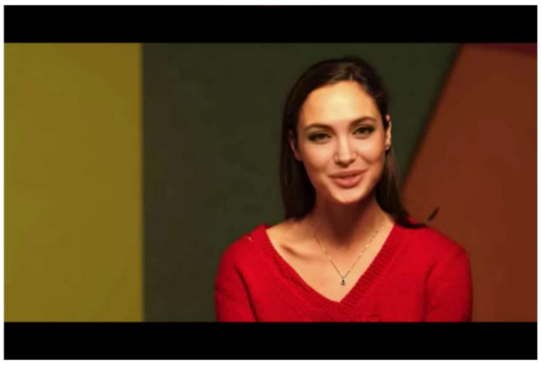
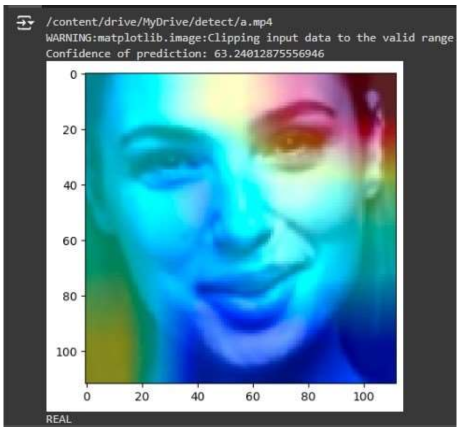
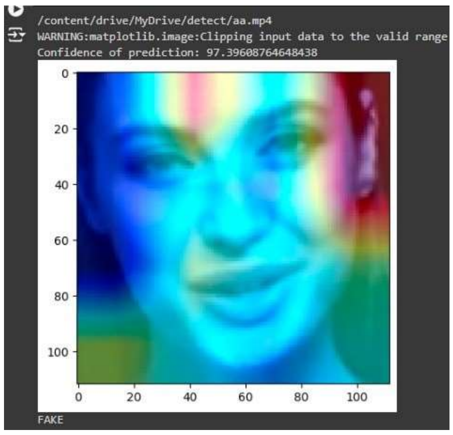


Fig 5.3.22: Input 12



```
/content/drive/MyDrive/detect/a.mp4
WARNING:matplotlib.image:Clipping input data to the valid range
Confidence of prediction: 63.24012875556946
```

Fig 5.3.23: Output 11



```
/content/drive/MyDrive/detect/aa.mp4
WARNING:matplotlib.image:Clipping input data to the valid range
Confidence of prediction: 97.39608764648438
```

Fig 5.3.24: Output 12

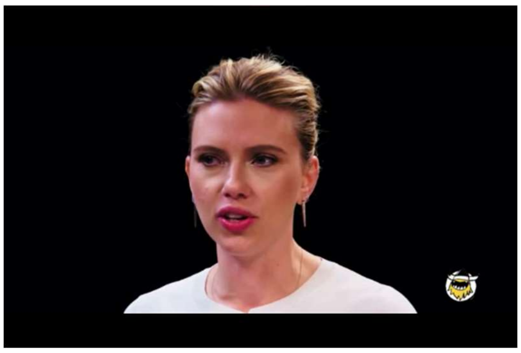


Fig 5.3.25: Input 13

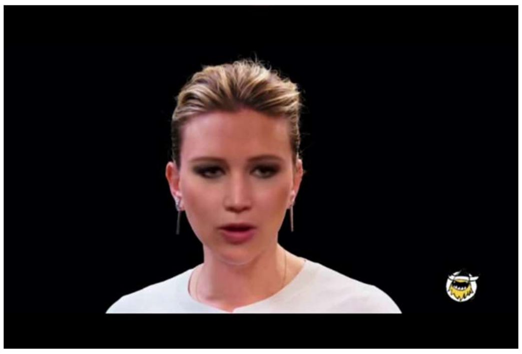
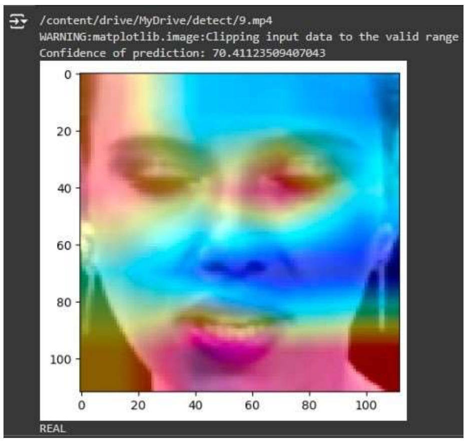
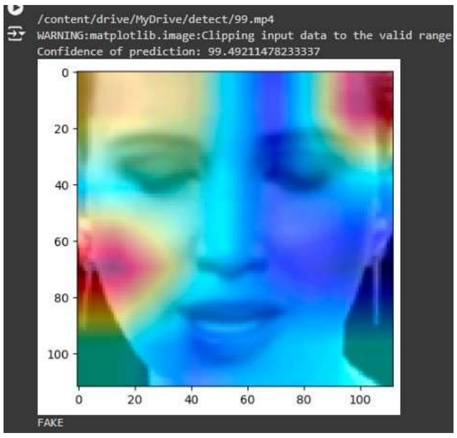


Fig 5.3.26: Input 14



```
/content/drive/MyDrive/detect/9.mp4
WARNING:matplotlib.image:Clipping input data to the valid range
Confidence of prediction: 70.41123509407043
```

Fig 5.3.27: Output 13



```
/content/drive/MyDrive/detect/99.mp4
WARNING:matplotlib.image:Clipping input data to the valid range
Confidence of prediction: 99.49211478233337
```

Fig 5.3.28: Output 14

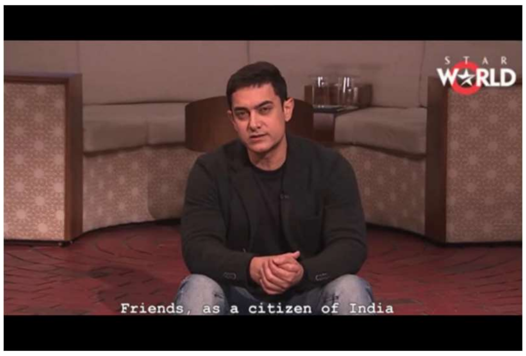


Fig 5.3.29: Input 15

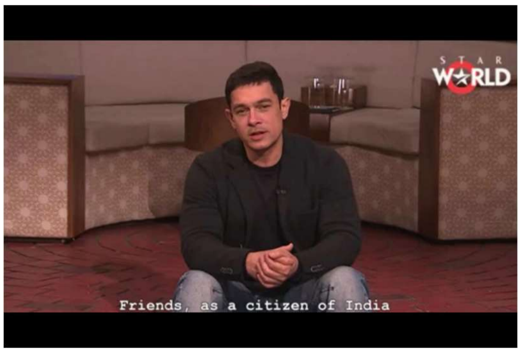
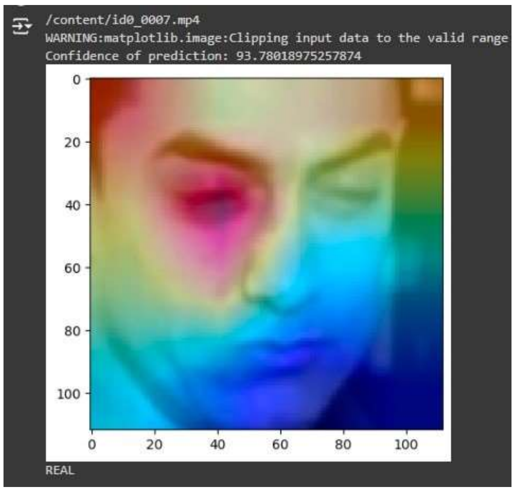


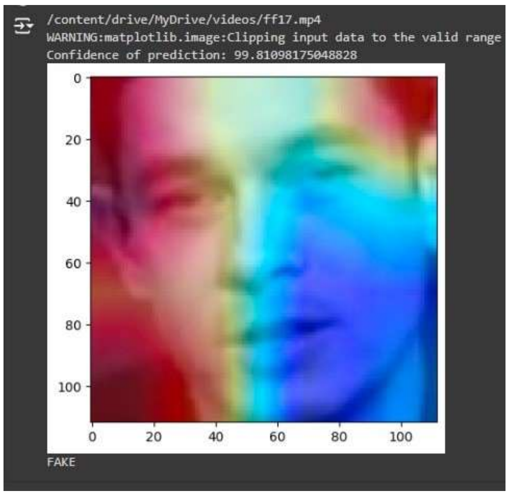
Fig 5.3.30: Input 16



```
/content/id0_0007.mp4
WARNING:matplotlib.image:Clipping input data to the valid range
Confidence of prediction: 93.78018975257874
```

```
REAL
```

Fig 5.3.31: Output 15



```
/content/drive/MyDrive/videos/ff17.mp4
WARNING:matplotlib.image:Clipping input data to the valid range
Confidence of prediction: 99.81098175048828
```

```
FAKE
```

Fig 5.3.32: Output 16

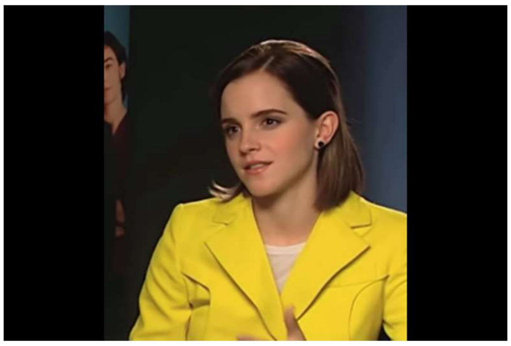


Fig 5.3.33: Input 17

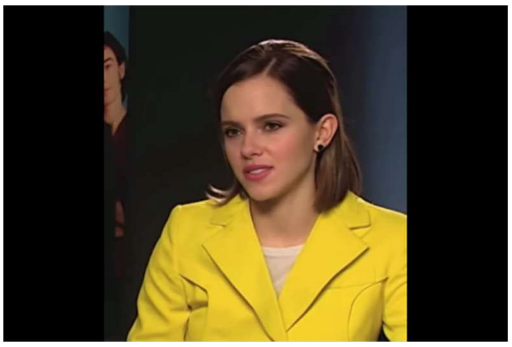
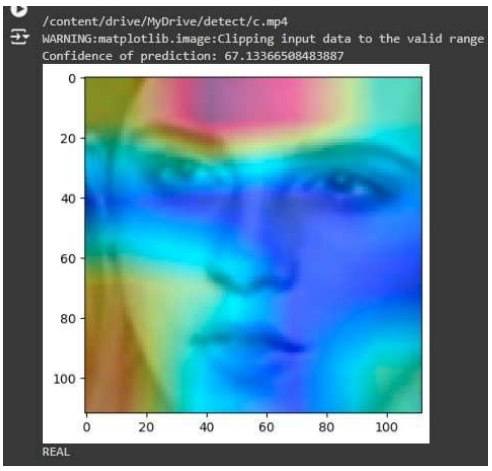


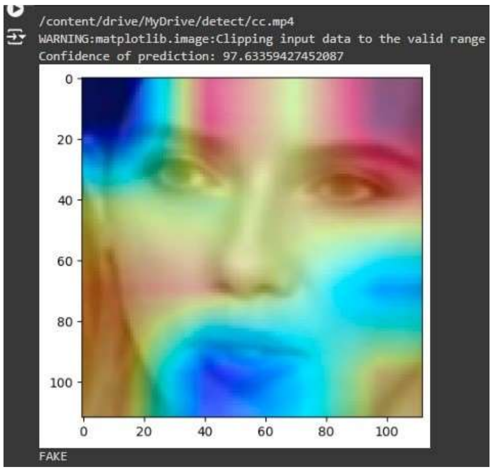
Fig 5.3.34: Input 18



```
/content/drive/MyDrive/detect/c.mp4
WARNING:matplotlib.image:Clipping input data to the valid range
Confidence of prediction: 67.13366508483887
```

```
REAL
```

Fig 5.3.35: Output 17



```
/content/drive/MyDrive/detect/cc.mp4
WARNING:matplotlib.image:Clipping input data to the valid range
Confidence of prediction: 97.63359427452087
```

```
FAKE
```

Fig 5.3.36: Output 18

6. Conclusions and Suggestions for Future Work

6.1 Conclusion

The principal aim of this project was to create a deep learning technique that could accurately differentiate artificial intelligence (AI)-generated false videos, or deepfakes, from authentic videos. The increasing possibility of deepfake technology being misused for nefarious purposes, like inciting political unrest, fabricating terrorist incidents, or extorting money, makes this task extremely crucial.

We suggested a novel strategy that makes use of a combination of recurrent neural networks (RNN) and convolutional neural networks (CNN) to overcome this difficulty. In particular, we extracted frame-level features from video data using the robust ResNeXt architecture—a CNN variant. Then, using these features, an RNN called a Long Short-Term Memory (LSTM) network was trained to determine whether the video material was real or fake. The successful temporal modeling of video sequences made possible by the combination of these two architectures captured the temporal and spatial characteristics essential for identifying deepfakes.

Our system was rigorously evaluated using a comprehensive dataset compiled from multiple existing deepfake detection datasets. This dataset was designed to be large, balanced, and representative of real-world scenarios, thereby enhancing the robustness and generalizability of our model. Through extensive experimentation, we demonstrated that our approach achieved competitive results in detecting both replacement (where a face is swapped with another) and reenactment (where facial expressions are altered) deepfakes.

Addressing the growing threat posed by deepfake technology has advanced significantly with the successful development of this deepfake detection system. Our project contributes to the larger endeavor to counteract the malicious use of deepfakes by offering a dependable technique for differentiating between real and fake videos. Spreading false information can have dire consequences in a number of domains, including politics, the media, and personal privacy. These are just a few of the profound implications of this.

In conclusion, this project underscores the potential of leveraging advanced AI techniques to tackle the challenges associated with deepfake detection. Our approach not only demonstrates technical efficacy but also highlights the importance of interdisciplinary efforts in safeguarding digital authenticity.

6.1 Future work

This deepfake detection project's future scope will undoubtedly surpass its current limitations. One of the main areas of growth will be the creation of an online tool that lets users submit and check media content for deepfakes. This platform would give people and organizations an easy-to-use interface to quickly confirm the legitimacy of image and video files. By making verification quick and easy, such a tool would be invaluable in preventing the spread of manipulated media.

Integrating our deepfake detection system directly into social media platforms represents a crucial next step. By embedding the detection system within the content moderation pipelines of these platforms, we can facilitate real-time fact-checking of published media. This integration would help ensure that any AI-generated or manipulated content is identified and flagged before it has the opportunity to be widely disseminated, thus combating the spread of misinformation and deception more effectively.

Extending the detection system's reach to handle body-based deepfakes is another exciting avenue for future research. The capacity to identify alterations of the complete human form, rather than just the face, will be more crucial as deepfake generation technology advances. A more complete response to the deepfake threat will come from the development of techniques to recognize AI-generated modifications of gestures, body movements, and other physical characteristics.

To further enhance the system's robustness, we should explore advanced data augmentation techniques. These techniques could include simulating various challenging environmental conditions, such as different lighting scenarios, low-resolution inputs, and noisy backgrounds. Evaluating and optimizing the model's performance under these conditions will ensure its practical applicability in real-world settings, where video quality and environmental factors can vary widely.

There is a need to address potential biases within the model, particularly those that may affect different demographic groups disproportionately. Investigating and mitigating these biases will be crucial in ensuring that the deepfake detection system is fair and unbiased. This involves analyzing fairness metrics and implementing strategies to correct any identified disparities in model performance across different demographic segments.

By pursuing these future developments, the deepfake detection project can evolve into a versatile and integrated platform that empowers individuals, organizations, and social media platforms to effectively combat the growing challenge of manipulated and synthetic media. This comprehensive approach will be crucial in preserving trust, authenticity, and transparency in the digital landscape.

References

[1] Rossler, Cozzolino, D, Verdoliva, L., Riess, Thies, J & Nießner, M (2019). "FaceForensics++: Learning to Detect Manipulated Facial Images". Findings of the IEEE International Conference on Computer Vision (ICCV).

[2] Li, Y., Chang, M. C., & Lyu, S. (2018). "In ictu oculi: Exposing AI generated fake face videos by detecting eye blinking." In Proceedings of the IEEE International Workshop on Information Forensics and Security (WIFS).

[3] Hsu, C. W., Chang, C. S., Lin, Y. Y., & Wei, S. H. (2018). "A two-stream neural network for tampered facial image detection." In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

[4] Bayar, B., & Stamm, M. C. (2016). "A deep learning approach to universal image manipulation detection using a new convolutional layer." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW).

[5] Afchar, Nozick, Yamagishi, J, & Echizen, I. (2018). "MesoNet- a Compact Facial Forgery Detection Network". Findings of European Conference on Computer Vision (ECCV).

[6] Gazi Hasin Ishrak, Zalish Mahmud, MD. Zami Al Zunaed Farabe, Tahera Khanom Tinni, Tanzim Reza and Mohammad Zavid Parvez "Explainable Deepfake Video Detection using Convolutional Neural Network and CapsuleNet".

[7] Aarti Karandikar, Vedita Deshpande, Sanjana Singh, Sayali Nagbhidkar, Saurabh Agrawal.(2020)"Deepfake Video Detection Using Convolutional Neural Network" In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).

[8] India Today: https://www.indiatoday.in/india/story/pm-modi-deepfakes-videos-photos-smart-india-hackathon-rashmika-mandanna-kajol-2478140-2023-12-20

[9] The Economic Times: https://economictimes.indiatimes.com/industry/tech/hong-kong-mnc-suffers-25-6-million-loss-in-deepfake-scam/articleshow/107465111.cms?from=mdr

[10] Outlook India: https://business.outlookindia.com/technology/deepfake-elections-how-indian-politicians-are-using-ai-manipulated-media-to-malign-opponents#:~:text=Election%20propaganda%20in%20India%20has,constituency%20hundreds%20of%20miles%20away.

[11] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus

Thies,Matthias Nießner, “FaceForensics++: Learning to Detect Manipulated Facial Images” in arXiv:1901.08971.

[12] Deepfake detection challenge dataset : https://www.kaggle.com/c/deepfake-detection challenge/

[13] Yuezun Li , Xin Yang , Pu Sun , Honggang Qi and Siwei Lyu “Celeb-DF: A Large-scale Challenging Dataset for DeepFake Forensics” in arXiv:1909.12962

[14] Deepfake Video of Mark Zuckerberg Goes Viral on Eve of House A.I. Hearing : https://fortune.com/2019/06/12/deepfake-mark-zuckerberg/

[15] 10 deepfake examples that terrified and amused the internet : https://www.creativebloq.com/features/deepfake-examples

[16] TensorFlow: https://www.tensorflow.org/

[17] Keras: https://keras.io/

[18] PyTorch : https://pytorch.org/

[19] G. Antipov, M. Baccouche, and J.-L. Dugelay. Face aging with conditional generative adversarial networks. arXiv:1702.01983, Feb. 2017

[20] J. Thies et al. Face2Face: Real-time face capture and re-enactment of RGB videos. Findings of the IEEE Conference on the Computer Vision and Pattern Recognition, pages 2387–2395, June 2016. Las Vegas, NV.

[21] Face app: https://www.faceapp.com/
[22] Face Swap : https://faceswaponline.com/

[23] Deepfakes, Revenge Porn, And The Impact On Women : https://www.forbes.com/sites/chenxiwang/2019/11/01/deepfakes-revenge-porn-and the-impact-on-women/

[24] The rise of the deepfake and the threat to democracy : https://www.theguardian.com/technology/ng-interactive/2019/jun/22/the-rise-of the-deepfake-and-the-threat-to-democracy

[25] Yuezun Li, Siwei Lyu, “ExposingDF Videos By Detecting Face Warping Arti facts,” in arXiv:1811.00656v3.

[26] Yuezun, Ming-Ching Chang and Siwei Lyu- “Exposing AI Created Fake Videos by Detecting Eye Blinking” in arXiv:1806.02877v2.

[27] Huy Nguyen , Junichi Yamagishi and Isao Echizen- “ Using capsule net works to detect forged images and videos ” in arXiv:1810.11215.

[28] D. Güera and J. Delp- "Deepfake Detection of Videos Using RNN," 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance, Auckland, New Zealand, 2018.

[29] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1–8, June 2008. Anchorage, AK

[30] Umur Aybars Ciftci, ˙Ilke Demir, Lijun Yin “Detection of Synthetic Portrait Videos using Biological Signals” in arXiv:1901.02212v2

[31] Thanh Thi Nguyena , Quoc Viet Hung Nguyenb , Dung Tien Nguyena , Duc Thanh Nguyena , Thien Huynh-Thec , Saeid Nahavandid , Thanh Tam Nguyene , Quoc-Viet Phamf , Cuong M. Nguyeng “Deep Learning for Deepfakes Creation and Detection: A Survey” arXiv:1909.11573v5 [cs.CV] 11 Aug 2022.

[32] Arash Heidari, Nima Jafari Navimipour, Hasan Dag, Mehmet Unal (October 2023). “Deepfake detection using deep learning methods: A systematic and comprehensive review”.

Bill of Material:

Items	Price
Laptop	₹70000/-
Google Colab	₹760/-

Table 7.1.1: Bill of Material