"Deep Fake Video Detection"

Design and Implementation of a System to Detect DeepFake Generated Videos

A project report of Phase-II submitted to Rajiv Gandhi Proudyogiki Vishwavidhyalaya, Bhopal towards partial fulfillment of the degree of Bachelor of Engineering in Computer Engineering

Guided By:
Mrs. Nikita Tiwari
Assistant Professor
Department of Computer Engg.

Ms. Neha Mehra
Assistant Professor
Department of Computer Engg.

Submitted By:
0801CS191053 - Mohnish Khanna
0801CS191063 - Priyabrata Saha
0801CS191066 - Priyanshu Sharma
0801CS191106 - Yogesh Dhangar

DEPARTMENT OF COMPUTER ENGINEERING SHRI G.S. INSTITUTE OF TECHNOLOGY AND SCIENCE, INDORE(M.P.) 2022-2023

SHRI G.S. INSTITUTE OF TECHNOLOGY AND SCIENCE, INDORE(M.P.)

RECOMMENDATION

The project report of Phase-II entitled "Deep Fake Video Detection": System to Detect DeepFake Videos was submitted by: **0801CS191053 - Mohnish Khanna, 0801CS191063 - Priyabrata Saha, 0801CS191066 - Priyanshu Sharma, 0801CS191106 -Yogesh Danger**, students of B.E. III year in the session 2022-2023, towards partial fulfillment of the degree of Bachelor of Engineering in Computer Engineering of Rajiv Gandhi Proudyogiki VishwaVidhyalaya, Bhopal is a satisfactory account of their work.

Mrs. Nikita Tiwari
Project Guide
Department of Computer Engg

Head of Department
Department of Computer Engg.

Ms. Neha Mehra
Project Co-Guide
Department of Computer Engg

Dean (Academics)

SHRI G.S. INSTITUTE OF TECHNOLOGY AND SCIENCE, INDORE(M.P.)

CERTIFICATE

The project report of Phase-II entitled "Deep Fake Video Detection": System to Detect DeepFake Videos submitted by: **0801CS191053 - Mohnish Khanna, 0801CS191063 - Priyabrata Saha, 0801CS191066 - Priyanshu Sharma, 0801CS191106 - Yogesh Dhangar, students** of B.E. IV year in the session 2022-2023, towards partial fulfillment of the degree of **Bachelor of Engineering in Computer Engineering** of Rajiv Gandhi Proudyogiki VishwaVidhyalaya, Bhopal is a satisfactory account of their work.

Internal Examiner **External Examiner**

Date:

ACKNOWLEDGEMENT

We express our profound sense of gratitude to our project guide **Mrs.Nikita Tiwari** and our co-guide **Ms. Neha Mehra** who had advised us to do the project work entitled "Deep Fake Video Detection": System to Detect DeepFake Videos. Their continuous support and motivation always made us deliver our best. Having such guidance has always been an amazing experience which is a valuable gift for an engineer to progress in his/her life.

We are also grateful to **Dr. Urjita Thakar**, Head, Department of Computer Engineering, and **Dr. Rakesh Saxena**, Director, S.G.S.I.T.S., Indore for providing us with numerous facilities and an academic environment during the study.

We sincerely wish to express, our gratefulness to all the members of staff of S.G.S.I.T.S. who have extended their cooperation at all times and have contributed in their way to developing the project.

The successful completion of a project is not an individual effort. It is an outcome of the cumulative number of people, each having their importance to the objective. We express love and respect towards our parents and all family members who are our strength in everything we do.

With a blend of gratitude, pleasure, and great satisfaction we convey our indebtedness to all those who have directly or indirectly contributed to the successful completion of our project work.

0801CS191053 - Mohnish Khanna ________________

0801CS191063 - Priyabrata Saha ________________

0801CS191066 - Priyanshu Sharma ________________

0801CS191106 - Yogesh Dhangar ________________

ABSTRACT

In recent months, free deep learning-based software tools have facilitated the creation of credible face exchanges in videos that leave few traces of manipulation, in what they are known as "DeepFake"(DF) videos. Manipulations of digital videos have been demonstrated for several decades through the good use of visual effects, recent advances in deep learning have led to a drastic increase in the realism of fake content and the accessibility in which it can be created. These so-called AI-synthesized media (popularly referred to as DF).

Creating the DF using the Artificially intelligent tools are simple task. But, when it comes to detection of these DF, it is a major challenge. Because training the algorithm to spot the DF is not simple. We have taken a step forward in detecting the DF using Convolutional Neural Network and Recurrent Neural Network.

The system uses a Convolutional Neural network (CNN) to extract features at the frame level. These features are used to train a recurrent neural network (RNN) which learns to classify if a video has been subject to manipulation or not and is able to detect the temporal inconsistencies between frames introduced by the DF creation tools.

The expected result against a large set of fake videos collected from the standard data set. We show how the proposed system can be competitive result in this task resulting by using a simple architecture.

Table of Contents

LIST OF FIGURES

Chapter 1

Introduction

In this chapter, there is a brief introduction about the project giving the need for the project, the problem being solved, objectives, and also the approach to be used.

1.1 Preamble

Machine learning (ML) has witnessed exponential growth throughout the past decade, and computer science has broadly embraced machine learning technology in various fields for many practical apps. However, the evolution of ML technology often causes new data protection and security challenges. With the exponential rise in online sites that intake and broadcast videos, the validity of the videos are in desperate need of testing. If any confusion emerges, there must be a solution for different innovative approaches to tackling them.

Deepfakes are videos or pictures altered to [1]look other than their original state with the help of Artificial Intelligence. The initial development of this technology served to fulfill the requirement of generating synthetic videos in the entertainment sector. For animation films and influential science fiction films, deepfake can provide realistic output. With more development in the field, happen the development of applications that allow people to use this face-swapping feature for humor purposes. Such applications were available for everyone to use and were not generating many authentic results. The same technology, when combined with deep learning turned out to be a massive outbreak. With the help of artificial intelligence, deepfakes are the most realistic doctored images and videos.

[1]

Big tech companies like Facebook and Google have put together researchers by arranging competitions to help identify the deep fakes and create a vast volume of datasets for the same. Google worked with paying and consenting actors to capture hundreds of videos over a year to create an extensive dataset and created thousands of deep-fakes from 1 such video utilizing open-source methods of deep-fake creation. With limited effort and simple equipment like smartphones, many new devices accessible on the Web have made it simpler than ever for anybody to create practical ”deepfakes.” Recent advancement in the development of deep-fake algorithms that generate distorted information has had harmful consequences for anonymity, protection, and mass communication.

Every day technical people are working on advancing technologies. Artificial intelligence is getting so powerful that every industry is using it in some way. Deepfake images may also be detrimental to facial recognition systems by interpreting an individual’s facial expressions and manipulating them on someone else’s image. Since their first appearance in late 2017 numerous open-source deepfake generation strategies have emerged, leading to a growing amount of synthesized media clippings.

1.2 Need of the Project

There are many tools available for creating the DF, but for DF detection there is hardly any tool available. The proposed approach for detecting the DF will be a great contribution to avoiding the percolation of the DF over the world wide web. We will be providing a web-based platform for the user to upload the video and classify it as fake or real. This project can be scaled up from developing a web-based platform to a browser plugin for automatic DF detections. Even big applications like WhatsApp, and Facebook can integrate this project with their application for easy pre-detection of DF before sending it to another user.

1.3 Problem Statement

Recent advances in deep learning have led to a dramatic increase in the realism of fake content and the accessibility in which it can be created. It becomes very important to detect the deepfake and avoid the percolation of deepfake through social media platforms. We have taken a step forward in detecting the deep fakes using LSTM based artificial Neural network.

1.4 Objectives

Goal and Objectives:
- The proposed project aims at discovering the distorted truth of the deep fakes.
- The proposed project will reduce the abuse and misleading of the common people on the world wide web.
- The proposed project will distinguish and classify the video as deepfake or pristine.
- Provide an easy-to-use system for users to upload the video and distinguish whether the video is real or fake.

1.6 Organization of the Report

The brief summary about the organization of report is given.

- Chapter 1: A brief introduction about the project giving the need for the project, the problem it is going to solve, objectives, and also the approach to be used.

- Chapter 2: It provides a background study of the project which includes Tools and Technologies.

- Chapter 3: This chapter describes the literature survey for the research paper published regarding the the topic.

- Chapter 4: This Chapter describes the detailed problem statement of the project work. It deals with a detailed analysis of the project with the functional requirements and non-functional requirements, system components, project model, risk identification, Usecase view.

- Chapter 5: This chapter focuses on design details comprising of the basic architecture of the system and working of various modules of the system. Along with this the algorithm designed has also discussed in this chapter.

- Chapter 6: This chapter contains the conclusion of the project and scope for future work along with references.

Chapter 2

Background Study

In this chapter, the background information of the project is presented. The first section contains a brief description of the main technology domain in which the proposed project falls, i.e., Deep Learning. The tools and technologies, different frameworks, and APIs required to build this project are specified in the next section. The last section of this chapter covers the existing application that has been developed in document creation and cleaning and used by people to perform academic day-to-day tasks.

2.1 Tools & Technologies

2.1.1 OpenCV

OpenCV is a cross-platform library using which users can develop real-time computer vision applications. It is a huge open-source library for computer vision, machine learning, and image processing. It supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. It is used to Identify image patterns and their various features such as vector space and perform mathematical operations on these features.
OpenCV was initially an Intel research initiative to advise CPU-intensive applications. It was officially launched in 1999. In the year 2006, its first major version, OpenCV 1.0 was released. In October 2009, the second major version, OpenCV 2 was released. In August 2012, OpenCV was taken by a nonprofit organization OpenCV.org.

Features of OpenCV Library:

- Read and write images
- Capture and save videos
- Process images (filter, transform)

- Perform feature detection
- Detect specific objects such as faces, eyes, and cars, in the videos or images.
- Analyze the video, i.e., estimate the motion in it, subtract the background, and track objects in it.

Below is a small demonstration of reading an image:

```
import cv2 image = cv2.imread(’image.png’)
```

Applications of OpenCV Library:

- Face Recognition • Automated inspection and surveillance
- Vehicles counting on highways along with their speeds
- Street view image stitching • Video/image search and retrieval
- Robot and driver-less car navigation and control
- Object Recognition
- Medical image analysis
- Movies – 3D structure from motion
- TV Channel's advertisement recognition

2.1.2 Google Colab

Google is quite aggressive in AI research. Over many years, Google developed an AI framework called TensorFlow and a development tool called Collaboratory. Today TensorFlow is open-sourced and since 2017, Google made Colaboratory free for public use. Colaboratory is now known as Google Colab or simply Colab.

Another attractive feature that Google offers to the developers is the use of GPU. Colab supports GPU and it is totally free. The reason for making it free for the public could be to make its software a standard in the academics for teaching machine learning and data science. It may also have a long-term perspective of building a customer base for Google Cloud APIs which are sold per-use basis.

Irrespective of the reasons, the introduction of Colab has eased the learning and development of machine learning applications.

As a programmer, you can perform the following using Google Colab.

- Write and execute code in Python
- Document your code that supports mathematical equations
- Create/Upload/Share notebooks
- Import/Save notebooks from/to Google Drive
- Import/Publish notebooks from GitHub
- Import external datasets e.g. from Kaggle
- Integrate PyTorch, TensorFlow, Keras, OpenCV
- Free Cloud service with free GPU

2.1.3 Pytorch

PyTorch is defined as an open-source machine learning library for Python. It is used for applications such as natural language processing. It is initially developed by the Facebook artificial-intelligence research group and Uber's Pyro software for probabilistic programming which is built on it.

Originally, PyTorch was developed by Hugh Perkins as a Python wrapper for the LusJIT based on the Torch framework. There are two PyTorch variants.

PyTorch redesigns and implements Torch in Python while sharing the same core C libraries for the backend code. PyTorch developers tuned this back-end code to run Python efficiently. They also kept the GPU-based hardware acceleration as well as the extensibility features that made Lua-based Torch.

The major features of PyTorch are mentioned below –

Easy Interface – PyTorch offers easy to use API; hence it is considered to be very simple to operate and runs on Python. The code execution in this framework is quite easy.

Python usage – This library is considered to be Pythonic which smoothly integrates with the Python data science stack. Thus, it can leverage all the services and functionalities offered by the Python environment.

Computational graphs – PyTorch provides an excellent platform that offers dynamic computational graphs. Thus a user can change them during runtime. This is highly useful when a developer has no idea of how much memory is required for creating a neural network model.

PyTorch is known for having three levels of abstraction as given below –

- Tensor – Imperative n-dimensional array which runs on GPU.

- Variable – Node in the computational graph. This stores data and gradient.
- Module – Neural network layer which will store state or learnable weights.
- It can be considered as a NumPy extension to GPUs.
- It allows the building of networks whose structure is dependent on computation itself.

2.1.4 Pandas

Pandas is an open-source Python library providing high-performance data manipulation and analysis tools using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.

In 2008, developer Wes McKinney started developing pandas when in need of high performance, flexible tool for analysis of data.

Prior to Pandas, Python was majorly used for data munging and preparation. It had very little contribution to data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyze.

Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, statistics, analytics, etc.

Key Features of Pandas

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets.
- Label-based slicing, indexing, and subsetting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High-performance merging and joining of data.
- Time Series functionality.

2.1.5 Data Collection

Collecting data for training the ML model is the basic step in the machine learning pipeline. The predictions made by ML systems can only be as good as the data on which they have been trained. Following are some of the problems that can arise in data collection:

- Inaccurate data. The collected data could be unrelated to the problem statement.
- Missing data. Sub-data could be missing. That could take the form of empty values in columns or missing images for some class of prediction.
- Data imbalance. Some classes or categories in the data may have a disproportionately high or low number of corresponding samples. As a result, they risk being under-represented in the model.
- Data bias. Depending on how the data, subjects, and labels themselves are chosen, the model could propagate inherent biases on gender, politics, age, or region, for example. Data bias is difficult to detect and remove.

Several techniques can be applied to address those problems:

- Pre-cleaned, freely available datasets. If the problem statement (for example, image classification, object recognition) aligns with a clean, pre-existing, properly formulated dataset, then take advantage of existing, open-source expertise.
- Web crawling and scraping. Automated tools, bots, and headless browsers can crawl and scrape websites for data.
- Private data. ML engineers can create their own data. This is helpful when the amount of data required to train the model is small and the problem statement is too specific to generalize over an open-source dataset.
- Custom data. Agencies can create or crowdsource the data for a fee.

2.1.6 Data Pre-Processing

Real-world raw data and images are often incomplete, inconsistent, and lacking in certain behaviors or trends. They are also likely to contain many errors. So, once collected, they are pre-processed into a format the machine learning algorithm can use for the model.

Pre-processing includes a number of techniques and actions:

- Data cleaning. These techniques, manual and automated, remove data incorrectly added or classified.

- Data imputations. Most ML frameworks include methods and APIs for balancing or filling in missing data. Techniques generally include imputing missing values with standard deviation, mean, median, and k-nearest neighbors (k-NN) of the data in the given field.
- Oversampling. Bias or imbalance in the dataset can be corrected by generating more observations/samples with methods like repetition, bootstrapping, or Synthetic Minority Over-Sampling Technique (SMOTE), and then adding them to the under-represented classes.
- Data integration. Combining multiple datasets to get a large corpus can overcome incompleteness in a single dataset.
- Data normalization. The size of a dataset affects the memory and processing required for iterations during training. Normalization reduces the size by reducing the order and magnitude of data.

2.1.7 Image Processing

Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be an image or characteristics/features associated with that image. Nowadays, image processing is among rapidly growing technologies.

Image processing basically includes the following three steps:

1. Importing the image via image acquisition tools.

2. Analyzing and manipulating the image.

3. Output in which result can be altered image or report that is based on image analysis.

There are two types of methods used for image processing[4] namely, analog and digital image processing. Analog image processing can be used for hard copies like printouts and photographs. They are compatible with human viewing and can be seen as brightness levels and colors. Image analysts use various fundamentals of interpretation while using these visual techniques. Digital image processing techniques help in manipulation of the digital images by using computers. A digital image is divided into a matrix or array of small picture elements called pixels. The three general phases that all types of data have to undergo while using digital techniques are pre-processing, enhancement, and display, information extraction.

The Following are the Phases of Image Processing:

1.Acquisition It could be as simple as being given an image that is in digital form. The main work involves:

a) Scaling b) Color conversion(RGB to Gray or vice-versa)

2.Image Enhancement It is amongst the simplest and most appealing in areas of Image Processing it is also used to extract some hidden details from an image and is subjective.

3.Image Restoration It also deals with appealing of an image but it is objective(Restoration is based on a mathematical or probabilistic model or image degradation).

4. Color Image Processing It deals with pseudocolor and full-color image processing color models are applicable to digital image processing.

5.Image Compression It involves developing some functions to perform this operation. It mainly deals with image size or resolution.

6. Morphological Processing It deals with tools for extracting image components that are useful in the representation description of shape.

7. Object Detection and Recognition It is a process that assigns a label to an object based on its descriptor.

2.1.8 Convolutional Neural Network

A convolutional neural network (CNN or ConvNet), is a network architecture for deep learning which learns directly from data, eliminating the need for manual feature extraction.

CNN's are particularly useful for finding patterns in images to recognize objects, faces, and scenes. They can also be quite effective for classifying non-image data such as audio, time series, and signal data.

Applications that call for object recognition and computer vision — such as self-driving vehicles and face-recognition applications — rely heavily on CNN's.

A convolutional neural network can have tens or hundreds of layers that each learns to detect different features of an image. Filters are applied to each training image at different resolutions, and the output of each convolved image is used as the input to the next layer. The filters can start as very simple features, such as brightness and edges, and increase in complexity to features that uniquely define the object.

Like other neural networks, a CNN is composed of an input layer, an output layer, and many hidden layers in between.

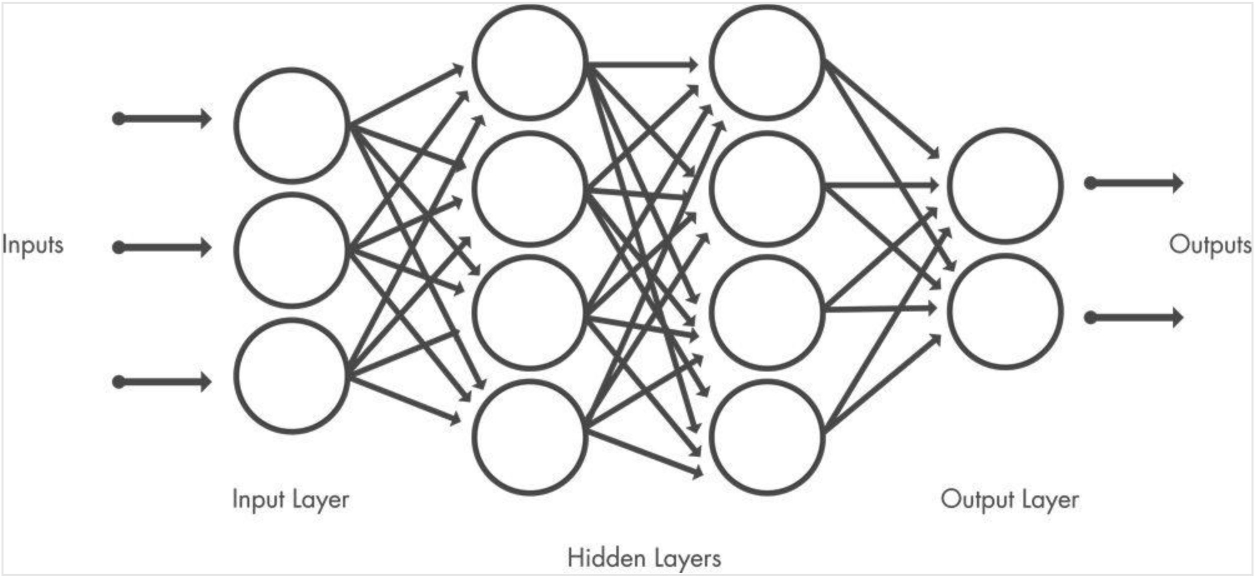


Fig 2.1 Typical CNN Model

These layers perform operations that alter the data with the intent of learning features specific to the data. Three of the most common layers are convolution, activation or ReLU, and pooling.

Convolution puts the input images through a set of convolutional filters, each of which activates certain features from the images.
Rectified linear unit (ReLU) allows for faster and more effective training by mapping negative values to zero and maintaining positive values. This is sometimes referred to as *activation* because only the activated features are carried forward into the next layer.
Pooling simplifies the output by performing nonlinear downsampling, reducing the number of parameters that the network needs to learn.

These operations are repeated over tens or hundreds of layers, with each layer learning to identify different features.

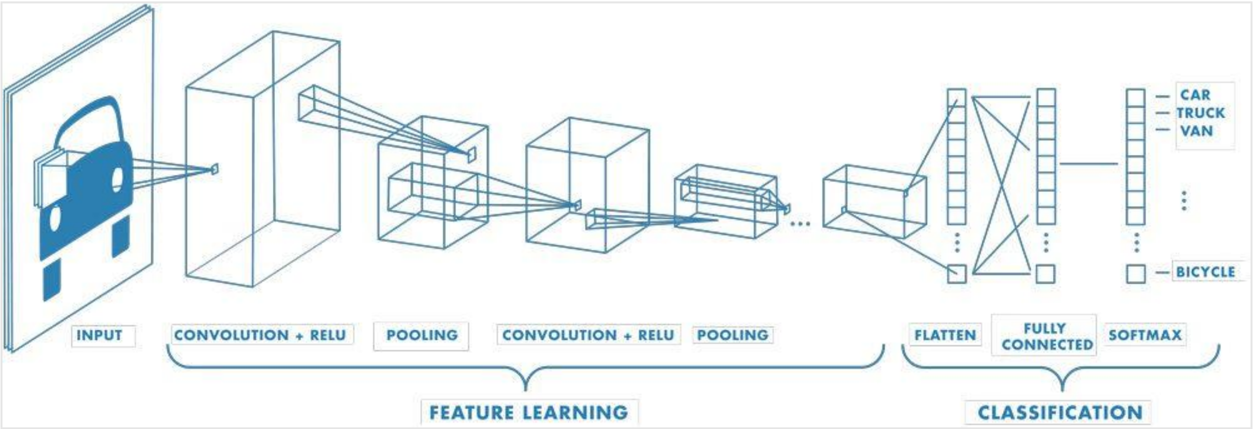


Fig 2.2 Example of a network with many convolutional layers. Filters are applied to each training image at different resolutions, and the output of each convolved image is used as the input to the next layer.

2.1.9 Recurrent Neural Network (RNN)

A recurrent neural network (RNN) is a deep learning network structure that uses information from the past to improve the performance of the network on current and future inputs. What makes RNNs unique is that the network contains a hidden state and loops. The looping structure allows the network to store past information in the hidden state and operate on sequences.

These features of recurrent neural networks make them well suited for solving a variety of problems with sequential data of varying lengths such as

Natural language processing
Signal classification
Video analysis

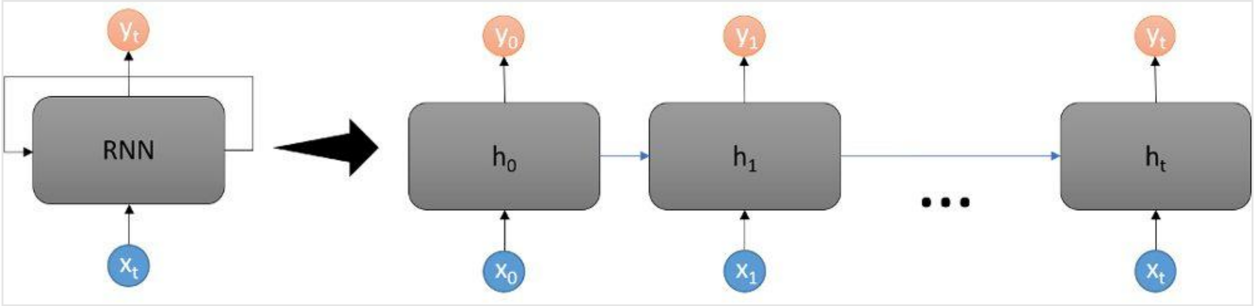


Fig 2.3 Unrolling a single cell of an RNN, which shows how information moves through the network for a data sequence. Inputs are acted on by the hidden state of the cell to produce the output, and the hidden state is passed to the next time step.

How does the RNN know how to apply the past information to the current input? The network has two sets of weights, one for the hidden state vector and one for the inputs. During training, the network learns weights for both the inputs and the hidden state. When implemented, the output is based on the current input, as well as the hidden state, which is based on previous inputs.

LSTM

In practice, simple RNNs experience a problem with learning longer-term dependencies. RNNs are commonly trained through backpropagation, where they can experience either a ‘vanishing’ or ‘exploding’ gradient problem. These problems cause the network weights to either become very small or very large, limiting the effectiveness of learning long-term relationships.

A special type of recurrent neural network that overcomes this issue is the long short-term memory (LSTM) network. LSTM networks use additional gates to control what information in the hidden cell makes it to the output and the next hidden state. This allows the network to more effectively learn long-term relationships in the data. LSTMs are a commonly implemented type of RNN.

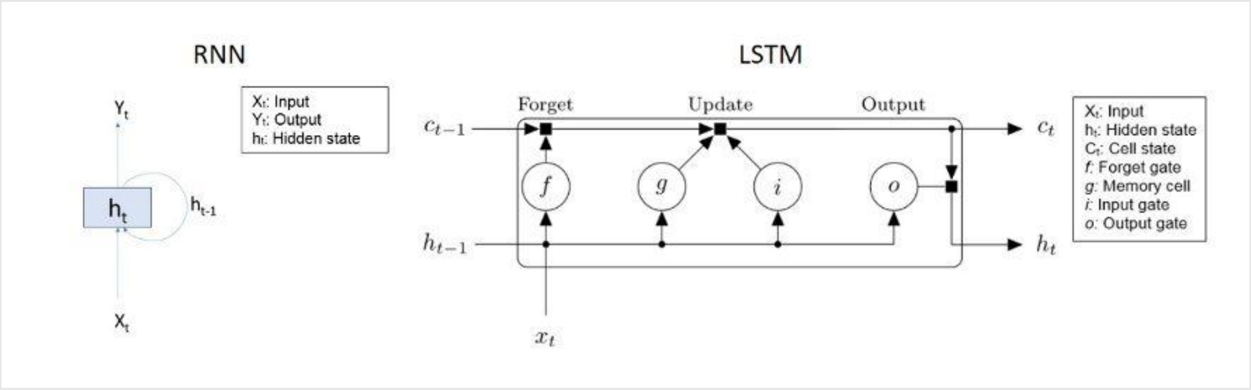


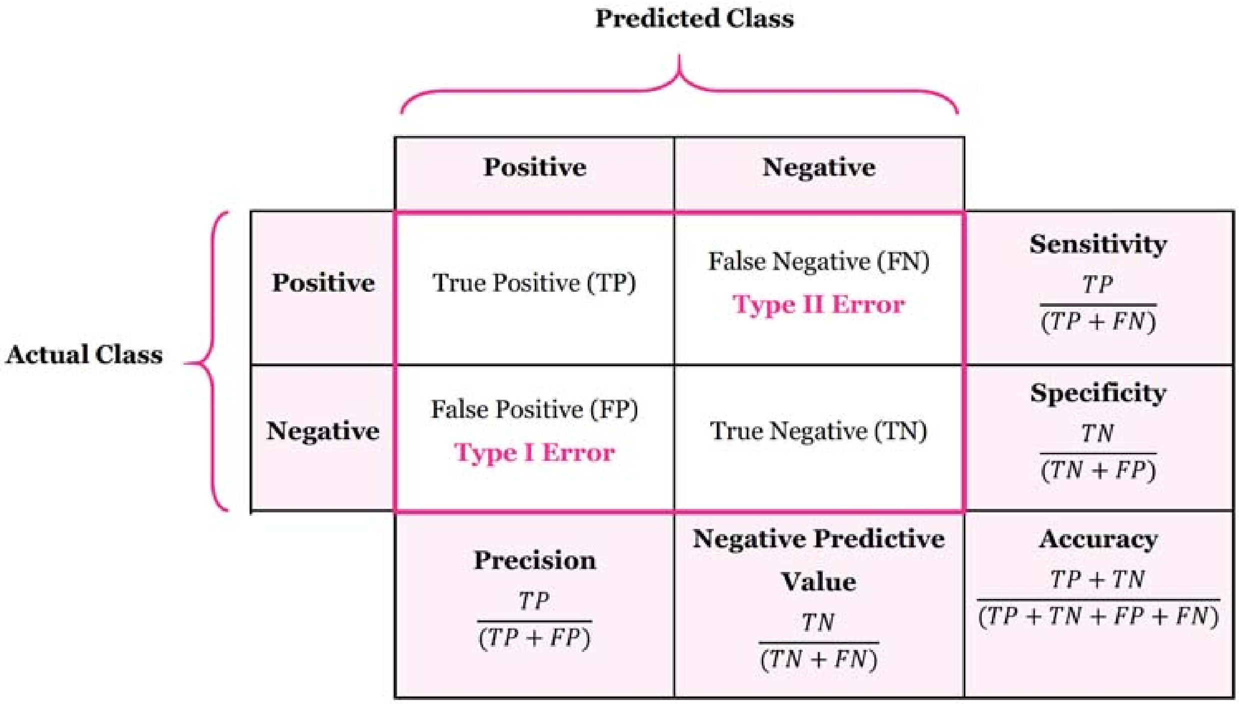
Fig 2.4 Comparison of RNN (left) and LSTM network (right)

2.1.10 Performace Evaluation

Confusion Matrix

A confusion matrix is a beneficial technique for illustrating a classifier model; This matrix will observe the relationship between the classified and the actual values. A matrix of ambiguity will summarise quickly how accurate a classification process is. In real or fake related problems in which categorization is binary, it is very relevant, because it predicts two factors positive or negative.

The confusion matrix has four sections: true positive, false positive, false negative, and true negative, as represented in Figure.



		Predicted Class		
		Positive	Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) Type II Error	Sensitivity $\frac{TP}{(TP + FN)}$
	Negative	False Positive (FP) Type I Error	True Negative (TN)	Specificity $\frac{TN}{(TN + FP)}$
		Precision $\frac{TP}{(TP + FP)}$	Negative Predictive Value $\frac{TN}{(TN + FN)}$	Accuracy $\frac{TP + TN}{(TP + TN + FP + FN)}$

Fig 2.5 Confusion Matrix

Train and Validation Accuracy and Loss

Accuracy is one parameter used to test models of classification. Imprecisely, accuracy is the proportion of observations that the process has got correct.

Some of the most significant issues in a machine learning algorithm will be overfitting. That is when the model suits well with the dataset, but it cannot make assumptions and make accurate assumptions about the general data.

This graph will represent the ratio between the accuracy observed when training the data, and the accuracy with the validation data.

Train and Validation Loss

Training a model involves learning (determining) positive values from designated instances with both the weights and the bias. A supervised deep learning model is shaped by looking at several instances and trying to come up with a solution that reduces the loss; this method is called computational risk minimization.

Loss is retribution for poor judgment. It is a measure that describes the forecasting of the model. If the forecasting of the approach is perfect, then the loss has to be 0; else the loss is higher.

Chapter 3

Literature Review

3.1 Existing Solutions

The idea of fake images or images with different person's faces is not new, but the recent technical advancements render it more accurate and believable. For the generation of deepfakes, the major roadblock is the quality of the output. Training a confrontational Deepfake model will contribute to a significant deterioration in the quality of the synthesized images Yang et al. (2020).

As with many other state-of-the-art approaches, the computing capacity it requires is far greater. The available models are often comparable when combining deepfakes into actual source videos. An approach, depending on the width of the videos to limit the storing of extracted facial features, may stand out Hashmi et al. (2020).

Deep forgery discriminator (DeepFD) Hsu et al. (2018) built to be a total one Convolutional Network that maps the features presented in the fake image to find unrealistic information. It is motivated by an utterly convolutional network Long et al. (2015). In the proposed classifier, two channels are added in the last layers, contributing to the learning target of this layer would continue to learn a vigorously active representation of impractical localization of data.

For the detection of deepfakes using neural network work on the extraction of facial features from the video at the level of frames. These can be improved with the addition of more layers to work on the quality of the video output. The main problem that is needed to tackle in this approach is the creation of a model to process a series recursively in a meaningful way G¨uera and Delp (2018a). Dynamics related to training can have a tremendous effect on the content of the resulting videos.

For high video quality deepfakes, the algorithm based on the measurements of visual consistency in terms of quality, the approach used in the area of display attack detection, turns out to be a better performing approach than algorithms that focused on the inconsistency in the video. Tests in Korshunov and Marcel (2018) show that

GAN-generated Deepfake videos are demanding both face recognition systems and current methods of deepfake detection.

In Neural Texture Synthesis by Thies et al. (2019), 3D texture reconstruction is performed under imperfect geometry and generated at real-time speeds. It collects high-level encoding of the presence of the surface and the 3D world. It lets the network exploit the re-rendering of the source voter system on-target candidates with ease. Nirkin et al. (2019) suggested an approach that does not even allow that the goal actor is present inside the training corpus. Paper has used two additional failure functions step by step to failure of continuity and loss of mixing. Stepwise lack of continuity governs the transition of the image of source candidate to goal candidate, and the lack of restoration holds in check the accuracy of facial reconstruction. Poisson mixing tends to smoothly mix with all faces with the surrounding setting with the mind.

Face Warping Artifacts [15] used the approach to detect artifacts by comparing the generated face areas and their surrounding regions with a dedicated Convolutional Neural Network model. In this work there were two-fold of Face Artifacts. Their method is based on the observations that current deepfake algorithm can only generate images of limited resolutions, which are then needed to be further transformed to match the faces to be replaced in the source video. Their method has not considered the temporal analysis of the frames.

Detection by Eye Blinking [16] describes a new method for detecting the deepfakes by the eye blinking as a crucial parameter leading to classification of the videos as deepfake or pristine. The Long-term Recurrent Convolution Network (LRCN) was used for temporal analysis of the cropped frames of eye blinking. As today the deepfake generation algorithms have become so powerful that lack of eye blinking can not be the only clue for detection of the deepfakes. There must be certain other parameters must be considered for the detection of deepfakes like teeth enchantment, wrinkles on faces, wrong placement of eyebrows etc.

Capsule networks to detect forged images and videos [17] uses a method that uses a capsule network to detect forged, manipulated images and videos in different scenarios, like replay attack detection and computer-generated video detection.

In their method, they have used random noise in the training phase which is not a good option. Still the model performed beneficial in their dataset but may fail on real time data due to noise in training. The proposed method is proposed to be trained on noiseless and real time datasets.

Recurrent Neural Network [18] (RNN) for deepfake detection used the approach of using RNN for sequential processing of the frames along with ImageNet pre-trained model. Their process used the HOHO [19] dataset consisting of just 600 videos. Their dataset consists small number of videos and same type of videos, which may not perform very well on the real time data. We will be training out model on large number of Realtime data.

Synthetic Portrait Videos using Biological Signals [20] approach extract biological signals from facial regions on pristine and deepfake portrait video pairs. Applied transformations to compute the spatial coherence and temporal consistency, capture the signal characteristics in feature vector and photoplethysmography (PPG) maps, and further train a probabilistic Support Vector Machine (SVM) and a Convolutional Neural Network (CNN). Then, the average of authenticity probabilities is used to classify whether the video is a deepfake or a pristine.

Fake Catcher detects fake content with high accuracy, independent of the generator, content, resolution, and quality of the video. Due to lack of discriminator leading to the loss in their findings to preserve biological signals, formulating a differentiable loss function that follows the proposed signal processing steps is not straight forward process.

Chapter 4

Analysis

In this chapter, a detailed analysis of the project has been discussed.

4.1 Detailed Problem Statement

In order to expose the forgery in extremely detailed facial expressions, such details are to be investigated frame by frame in the production of a deepfake picture. The goal of this project is to create a deep learning model that is capable of recognizing deepfake videos.

To develop a web application that will let users upload a video from their local storage and the model should predict whether the video is deepfake or pristine. The application should perform all the steps from pre-processing to percentage truth evaluation upon the given video.

The application will be very useful for people to themselves detect whether the video that they have received is actually real or it is fake.

4.2.1 Solution Requirement

We analyzed the problem statement and found the feasibility of the solution of the problem. We read different research papers as mentioned in 2.2. After checking the feasibility of the problem statement. The next step is the dataset gathering and analysis. We analyzed the data set in the different approaches of training like negatively or positively trained i.e training the model with only fake or real video but found that it may lead to the addition of extra bias in the model leading to inaccurate predictions. So after doing a lot of research we found that the balanced training of the algorithm is the best way to avoid the bias and variance in the algorithm and get good accuracy.

4.2.2 Solution Constraints

We analyzed the solution in terms of cost, speed of processing, requirements, level of expertise, and availability of equipment.

4.2.3 Development

After analysis we decided to use the PyTorch framework along with python3 language for programming. PyTorch is chosen as it has good support to CUDA i.e Graphics Processing Unit (GPU) and it is customizable. Google Cloud Platform for training the final model on a large number of data-set.

4.2.4 Evaluation

We will evaluate the proposed model with a large number of real-time datasets. The confusion matrix approach is used to evaluate the accuracy of the trained model.

4.3 Outcome

The outcome of the solution is trained deepfake detection models that will help the users to check if the new video is deepfake or real.

4.4 Applications

Web-based applications will be used by the user to upload the video and submit the video for processing. The model will pre-process the video and predict whether the uploaded video is a deepfake or real video.

4.5 Hardware Resources Required

In this project, a computer with sufficient processing power is needed. This project requires too much processing power, due to the image and video batch processing.

• Client-side Requirements: Browser: Any Compatible browser device

Sr. No.	Parameter	Requirement
1	Intel I5	2.5 GHz
2	RAM	8 GB
3	Hard Disk	100 GB
4	Graphic Card	NVIDIA GeForce GTX

4.6 Software Resources Required

Platform:

1. Operating System: Windows 7+
2. Programming Language: Python 3.0
3. Framework: PyTorch 1.4, Django 3.0
4. Cloud platform: Google Cloud Platform
5. Libraries: OpenCV, Face-recognition

4.7 Project Model

We use the Spiral model As the Software development model focuses on the people doing the work, how they work together, and risk handling. We are using Spiral because It ensures changes can be made quicker and throughout the development process by having consistent evaluations to assess the product with the expected outcomes requested. As we developed the application in the various modules, the spiral model is best suited for this type of application. A Spiral approach provides a unique opportunity for clients to be involved throughout the project, from prioritizing features to iteration planning and review sessions to frequent algorithms containing new features. However, this also requires clients to understand that they are seeing a work in progress in exchange for this added benefit of transparency. As the proposed model consists of a lot of risks and the spiral model is capable of handling the risks that are the reason we are using the spiral model for product development.

4.8 Risk Identification

Before the training, we need to prepare thousands of images for both persons. We can take a shortcut and use a face detection library to scrape facial pictures from their videos. Spend significant time improving the quality of your facial pictures. It impacts your final result significantly.
1. Remove any picture frames that contain more than one person.
2. Make sure you have an abundance of video footage. Extract facial pictures containing different poses, face angles, and facial expressions.
3. Some resembling of both persons may help, like similar face shapes.

4.9 Usecase View

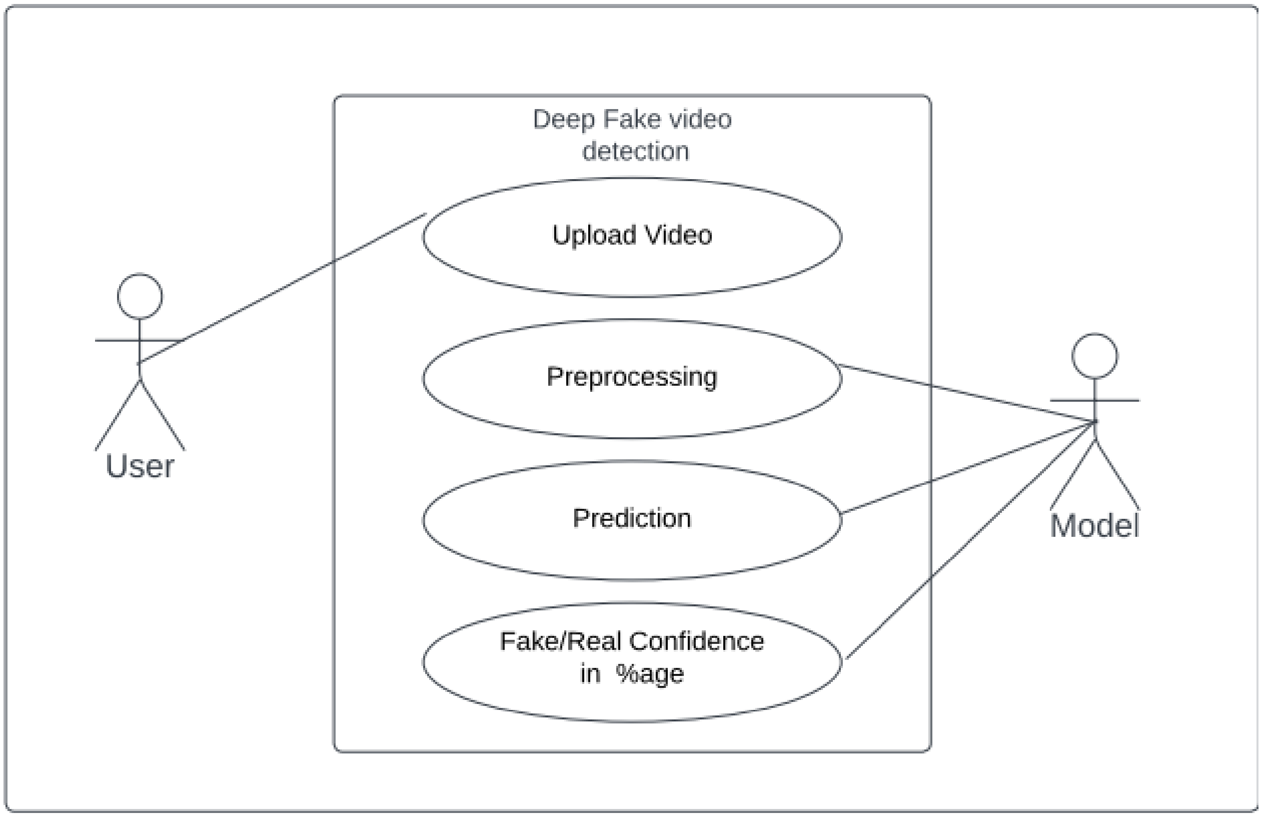


Fig 4.1 Usecase Diagram

4.10 Non-Functional Requirements

Performance Requirement

- The software should be efficiently designed so as to give reliable recognition of fake videos and so that it can be used for more pragmatic purposes.
- The design is versatile and user-friendly.
- The application is fast, reliable, and time-saving.
- The system has universal adaptations.
- The system is compatible with future up-gradation and easy integration.

Safety Requirement

- The Data integrity is preserved. Once the video is uploaded to the system. It is only processed by the algorithm. The videos are kept secure from the human interventions, as the uploaded video is not are not able to human manipulation.
- To extent the safety of the videos uploaded by the user will be deleted after 30 min from the server.

Security Requirement

- While uploading the video, the video will be encrypted using a certain symmetric encryption algorithm. On the server also the video is in encrypted format only. The video is only decrypted from preprocessing till we get the output. After getting the output the video is again encrypted.
- This cryptography will help in maintaining the security and integrity of the video.
- SSL certification is made mandatory for Data security.

Chapter 5

Design

This chapter specifies various diagrams related to the project. It includes System Architecture, Activity Diagram, Sequence Diagram, Collaboration, and Data Flow Diagrams(level 0, level 1, and level2)

5.1 System Architecture

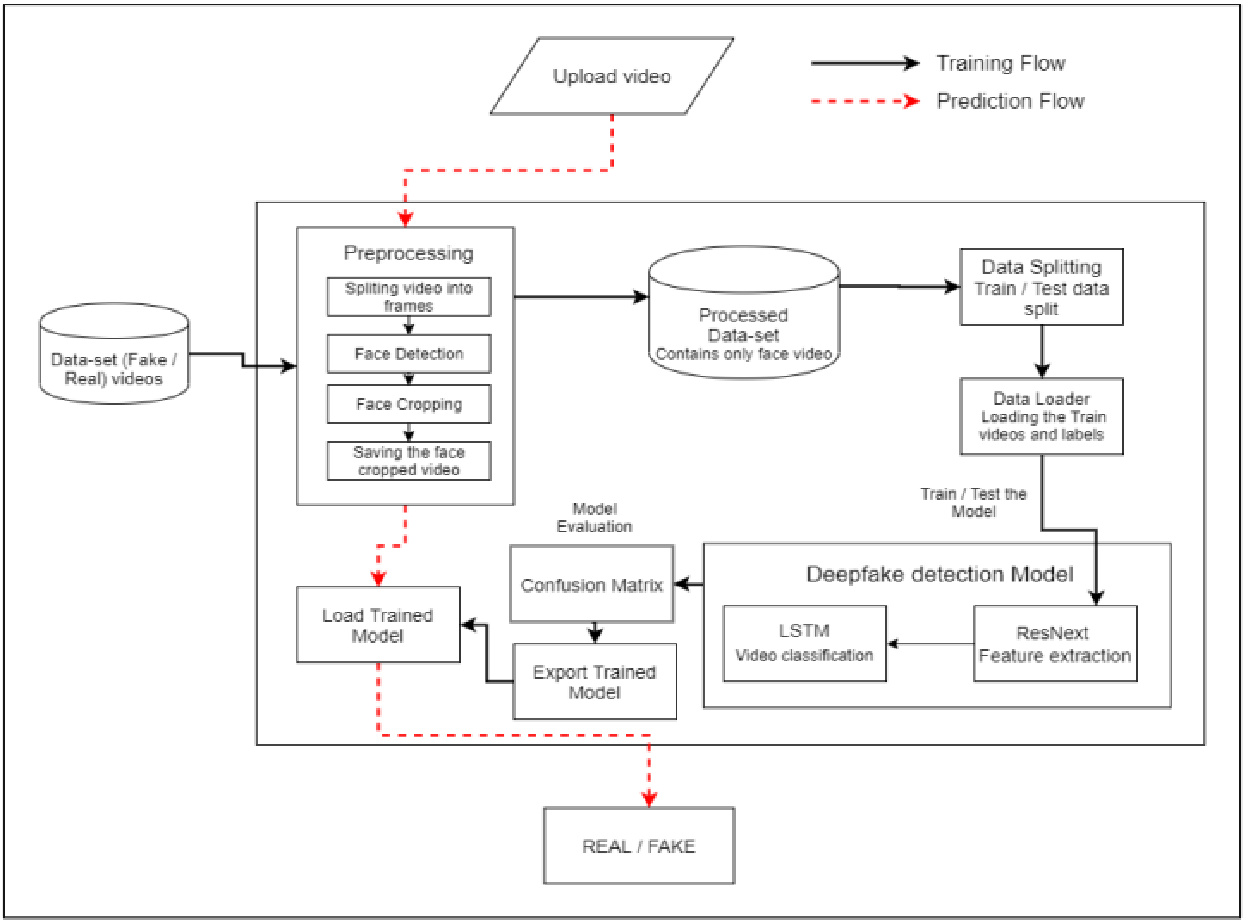


Fig. 5.1 System Architecture

In this system, we will trained the proposed PyTorch deepfake detection model on an equal number of real and fake videos in order to avoid bias in the model. The system architecture of the model is shown in the figure. In the development phase, we will take a dataset, preprocesse the dataset, and create a new processed dataset that only includes the face-cropped videos.

Creating deepfake videos

To detect deepfake videos it is very important to understand the creation process of the deepfake. The majority of the tools including the GAN and autoencoders take a source image and target video as input. These tools split the video into frames, detect the face in the video and replace the source face with the target face on each frame. Then the replaced frames are then combined using different pre-trained models. These models also enhance the quality of video by removing the left-over traces by the deepfake creation model. Which results in the creation of a deepfake that looks realistic in nature. We will also used the same approach to detect deepfakes. Deepfakes created using the pre-trained neural network models are very realistic that it is almost impossible to spot the difference with the naked eyes. But in reality, the deepfakes creation tools leave some of the traces or artifacts in the video which may not be noticeable by the naked eyes. The motive of this paper is to identify these unnoticeable traces and distinguishable artifacts of these videos and classified them as deepfake or real videos.

5.2 Design Diagrams

In this section Activity diagram, Sequence diagram, Collaboration diagram, and Data Flow diagram have been drawn for a better understanding of the system

5.2.1 Data Flow Diagram

DFD Level-0

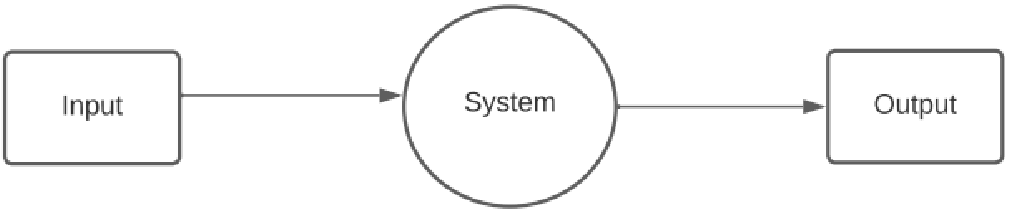


Fig. 5.2 DFD Level-0

DFD level – 0 indicates the basic flow of data in the system. In this System, Input is given equal importance as that Output.

- Input: Here input to the system is uploading video.
- System: In the system, it shows all the details of the Video.
- Output: The output of this system is whether it shows the fake video or not.

Hence, the data flow diagram indicates the visualization of the system with its input and output flow.

DFD Level-1

[1] DFD Level – 1 gives more in and out information about the system.
[2] Where the system gives detailed information about the procedure taking place.

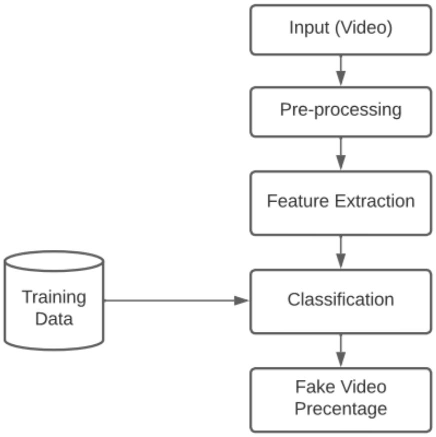


Fig 5.3 DFD Level-1

DFD Level-2

[1] DFD level-2 enhances the functionality used by user etc.

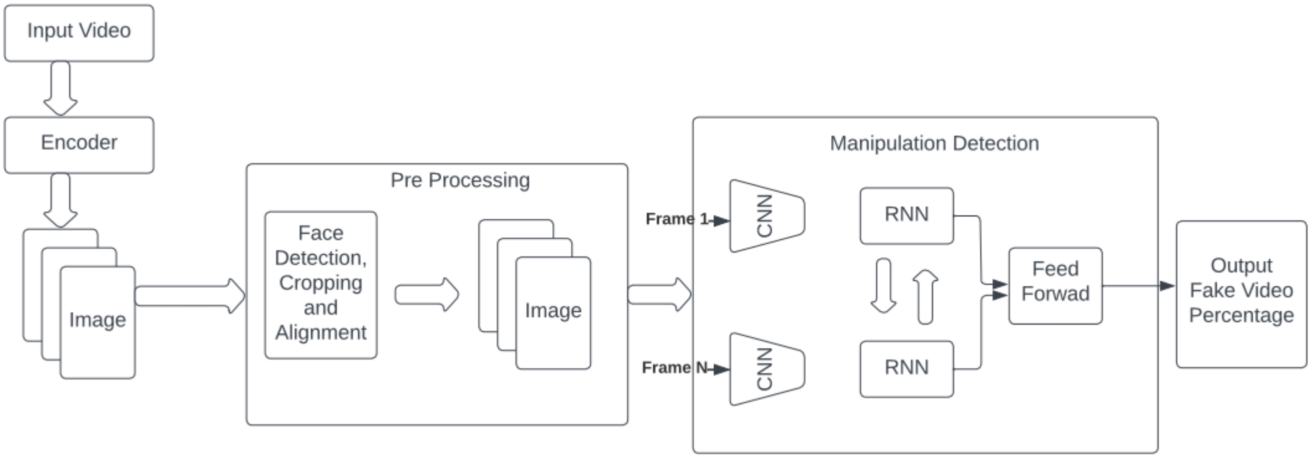


Fig 5.4 DFD Level-2

5.2.2 Activity Diagram

Training Workflow:

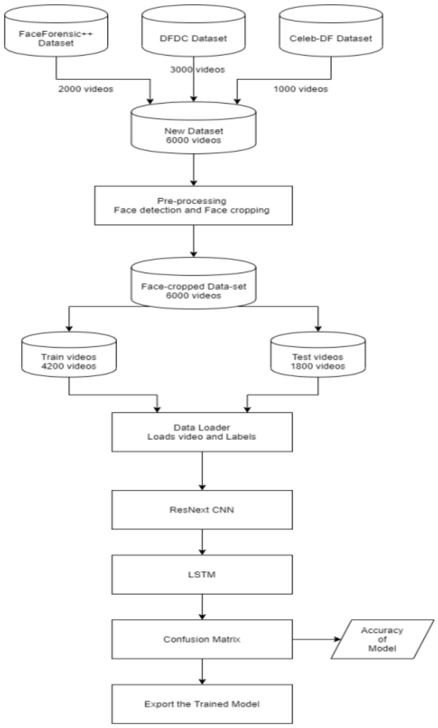


Fig 5.5 Activity Diagram: Training Workflow

Testing Workflow:

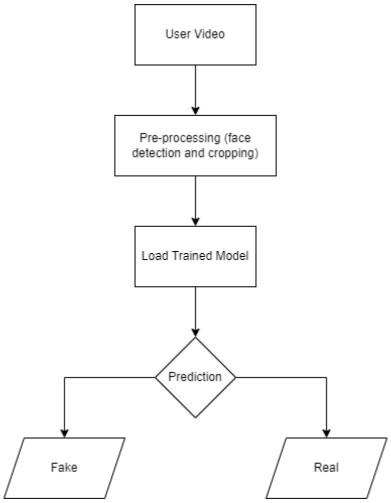


Fig 5.6 Activity Diagram: Testing Workflow

5.2.3 Sequence Diagram

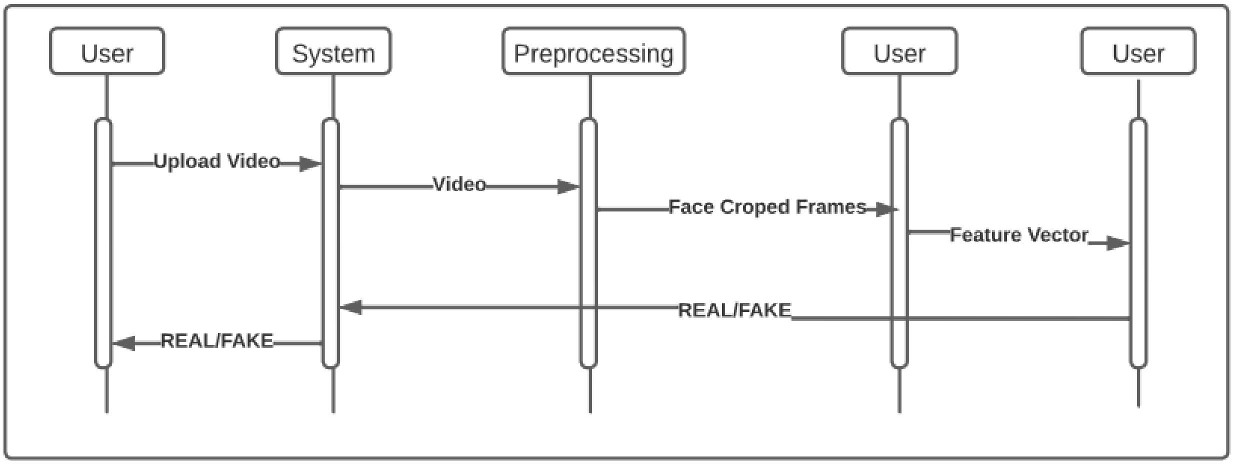


Fig 5.7 Sequence Diagram

5.3 Architectural Design

5.3.1 Module 1: Data-set Gathering

For making the model efficient for real-time prediction. We have gathered the data from different available data-sets like FaceForensic++(FF)[1], Deepfake detection challenge(DFDC)[2], and Celeb-DF[3]. Further, we have mixed the dataset and the collected datasets and created our own new dataset, for accurate and real-time detection of different kinds of videos. To avoid the training bias of the model we have considered 50% Real and 50% fake videos. The deep fake detection challenge (DFDC) dataset [3] consist of certain audio alerted video, as audio deepfake are out of scope for this paper. We preprocessed the DFDC dataset and removed the audio altered videos from the dataset by running a python script

t. After preprocessing the DFDC dataset, we have taken 1500 Real and 1500 Fake videos from the DFDC dataset. 1000 Real and 1000 Fake videos from the FaceForensic++(FF)[1] dataset and 500 Real and 500 Fake videos from the CelebDF[3] dataset. Which makes our total dataset consists of 3000 Real, 3000 fake videos, and 6000 videos in total. Figure 2 depicts the distribution of the data sets.

5.3.2 Module 2: Pre-processing

In this step, the videos are preprocessed and all the unrequired noise is removed from the videos. Only the required portion of the video i.e face is detected and cropped. The first step in the preprocessing of the video is to split the video into frames. After splitting the video into frames the face is detected in each of the frames and the frame is cropped along the face. Later the cropped frame is again converted to a new video by combining each frame of the video. The process is followed for each video which leads to the creation of a processed dataset containing face-only videos. The frame that does not contain the face is ignored while preprocessing. To maintain the uniformity of the number of frames, we select a threshold value based on the mean of the total frames count of each video. Another reason for selecting a threshold value is limited computation power. A video of 10 seconds at 30 frames per second(fps) will have a total of 300 frames and it is computationally very difficult to process the 300 frames at a single time in the experimental environment. So, based on our Graphics Processing Unit (GPU) computational power in the experimental environment we have selected 150 frames as the threshold value. While saving the frames to the new dataset we have only saved the first 150 frames of the video to the new video. To demonstrate the proper use of Long Short-Term Memory (LSTM) we have considered the frames the sequential i.e. first 150 frames and not randomly. The newly created video is saved at a frame rate of 30 fps and a resolution of 112 x 112.

5.3.3 Module 3: Data-set split

The dataset is split into train and test datasets with a ratio of 70% train videos (4,200) and 30% (1,800) test videos. The train and test split is a balanced split i.e 50% of the real and 50% of fake videos in each split.

5.3.4 Module 4: Model Architecture

The proposed model is a combination of CNN and RNN. We will be using the Pre-trained ResNext CNN model to extract the features at frame level and based on the extracted features the LSTM network is trained to classify the video as deepfake or pristine. Using the Data Loader on the training split of videos the labels of the videos are loaded and fitted into the model for training.

ResNext:

Instead of writing the code from scratch, we used the pre-trained model of ResNext for feature extraction. ResNext is a Residual CNN network optimized for high performance on deeper neural networks. For the experimental purpose, we will be using the resnext50_32x4d model. We have used a ResNext of 50 layers and 32 x 4 dimensions.

Following this, we will be fine-tuning the network by adding extra required layers and selecting a proper learning rate to properly converge the gradient descent of the model. The 2048-dimensional feature vectors after the last pooling layers of ResNext is used as the sequential LSTM input.

LSTM for Sequence Processing:

2048-dimensional feature vectors are fitted as the input to the LSTM. We will be using 1 LSTM layer with 2048 latent dimensions and 2048 hidden layers along with a 0.4 chance of dropout, which is capable to do achieve our objective. LSTM is used to process the frames in a sequential manner so that the temporal analysis of the video can be made, by comparing the frame at ‘t’ second with the frame of ‘t-n’ seconds. Where n can be any number of frames before t. The model also consists of the Leaky Relu activation function. A linear layer of 2048 input features and 2 output features are used to make the model capable of learning the average rate of correlation between eh input and output. An adaptive average polling layer with the output parameter 1 is used in the model. Which gives the target output size of the image of the form H x W. For sequential processing of the frames, a Sequential Layer is used. The batch size of 4 is used to perform the batch training. A SoftMax layer is used to get the confidence of the model during prediction.

5.3.5 Module 5: Hyper-parameter tuning

It is the process of choosing the perfect hyper-parameters for achieving maximum accuracy. After reiterating the many times on the model. The best hyper-parameters for our dataset will be chosen. To enable the adaptive learning rate Adam[21] optimizer with the model parameters is used. The learning rate will be tuned to 1e-5 (0.00001) to achieve a better global minimum of gradient descent. The weight decay used will be 1e-3.

As this is a classification problem so to calculate the loss, the cross-entropy approach is used. To use the available computation power properly batch training is used. The batch size is taken off 4. A batch size of 4 is tested to be the ideal size for training in our development environment.

The User Interface for the application will be developed using the Django framework. Django is used to enable the scalability of the application in the future.

The first page of the User interface i.e index.html contains a tab to browse and upload the video. The uploaded video is then passed to the model and a prediction is made by the model. The model returns the output whether the video is real or fake along with the confidence of the model. The output is rendered in the predict.html on the face of the playing video.

Chapter 6

Project Implementation

6.1 Introduction

There are many examples where deepfake creation technology is used to mislead the people on social media platform by sharing the false deepfake videos of the famous personalities like Mark Zuckerberg Eve of House A.I. Hearing, Donald Trump's Breaking Bad series where he was introduces as James McGill, Barack Obama's public service announcement and many more. These types of deepfakes creates a huge panic among the normal people, which arises the need to spot these deepfakes accurately so that they can be distinguished from the real videos.

Latest advances in the technology have changed the field of video manipulation. The advances in the modern open source deep learning frameworks like TensorFlow, Keras, PyTorch along with cheap access to the high computation power has driven the paradigm shift. The Conventional autoencoders and Generative Adversarial Network pretrained models have made the tampering of the realistic videos and images very easy. Moreover, access to these pretrained models through the smartphones and desktop applications like FaceApp and Face Swap has made the deepfake creation a childish thing. These applications generate a highly realistic synthesized transformation of faces in real videos. These apps also provide the user with more functionalities like changing the face hair style, gender, age and other attributes. These apps also allow the user to create a very high quality and indistinguishable deepfakes. Although some malignant deepfake videos exist, but till now they remain a minority. So far, the released tools that generate deepfake videos are being extensively used to create fake celebrity pornographic videos or revenge porn. Some of the examples are Brad Pitt, Angelina Jolie nude videos. The real looking nature of the deepfake videos makes the celebraties and other famous personalities the target of pornographic material, fake surveillance videos, fake news and malicious hoaxes. The Deepfakes are very much popular in creating the political tension. Due to which it becomes very important to detect the deepfake videos and avoid the percolation of the deepfakes on the social media platforms.

6.2 Tools and Technologies Used

6.2.1 Planning

1. OpenProject

6.2.2 UML Tools

1. draw.io

6.2.3 Programming Languages

1. Python3

6.2.4 Programming Frameworks

1. pyTorch

6.2.5 IDE

1. Google Colab
2. Jupyter Notebook
3. Visual Studio Code

6.2.6 Versioning Control

1. Git

6.2.7 Libraries

1. Torch
2. Torchvision
3. Os
4. Numpy
5. Cv2
6. Matpolib
7. Face_recognition
8. Json
9. Pandas
10. Copy
11. Glob
12. Random

13. sklearn

6.3 Algorithm Details

6.3.1 Dataset Details

Refer 5.3.1

6.3.2 Preprocessing Details

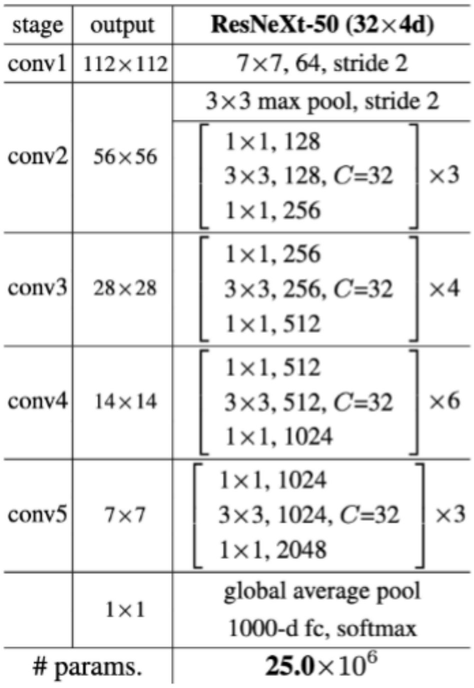
- Using glob we imported all the videos in the directory in a python list.
- cv2.VideoCapture is used to read the videos and get the mean number of frames in each video.
- To maintain uniformity, based on mean a value 150 is selected as idea value for creating the new dataset.
- The video is split into frames and the frames are cropped on face location.
- The face cropped frames are again written to new video using VideoWriter.
- The new video is written at 30 frames per second and with the resolution of 112 x 112 pixels in the mp4 format.
- Instead of selecting the random videos, to make the proper use of LSTM for temporal sequence analysis the first 150 frames are written to the new video.

```
def create_face_videos(path_list,out_dir):
  already_present_count =  glob.glob(out_dir+'*.mp4')
  print("No of videos already present " , len(already_present_count))
  for path in tqdm(path_list):
    out_path = os.path.join(out_dir,path.split('/')[-1])
    file_exists = glob.glob(out_path)
    if(len(file_exists) != 0):
      print("File Already exists: " , out_path)
      continue
    frames = []
    flag = 0
    face_all = []
    frames1 = []
    out = cv2.VideoWriter(out_path,cv2.VideoWriter_fourcc('M','J','P','G'), 30, 
(112,112))
    for idx,frame in enumerate(frame_extract(path)):
      #if(idx % 3 == 0):
      if(idx <= 150):
        frames.append(frame)
        if(len(frames) == 4):
          faces = face_recognition.batch_face_locations(frames)
          for i,face in enumerate(faces):
            if(len(face) != 0):
              top,right,bottom,left = face[0]
            try:
              out.write(cv2.resize(frames[i][top:bottom,left:right,:],(112,112)))
            except:
              pass
          frames = []
    try:
      del top,right,bottom,left
    except:
      pass
    out.release()
```

6.3.3 Model Details

The model consists of following layers:

- **ResNext CNN** : The pre-trained model of Residual Convolution Neural Network is used. The model name is resnext50_32x4d(). This model consists of 50 layers and 32 x 4 dimensions. Figure shows the detailed implementation of model.



stage	output	**ResNeXt-50 (32×4d)**
conv1	112×112	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2 [1×1, 128 3×3, 128, C=32 1×1, 256] ×3
conv3	28×28	[1×1, 256 3×3, 256, C=32 1×1, 512] ×4
conv4	14×14	[1×1, 512 3×3, 512, C=32 1×1, 1024] ×6
conv5	7×7	[1×1, 1024 3×3, 1024, C=32 1×1, 2048] ×3
	1×1	global average pool 1000-d fc, softmax
# params.		**25.0**×10^6

Figure 6.1: ResNext Architecture

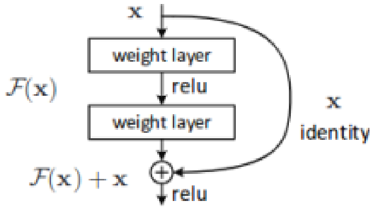

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 6.2: ResNext Working

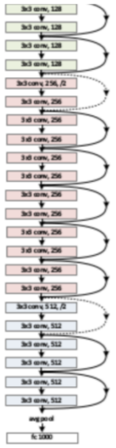


Figure 6.3: Overview of ResNext Architecture

- **Sequential Layer :** Sequential is a container of Modules that can be stacked together and run at the same time. Sequential layer is used to store feature vector returned by the ResNext model in a ordered way. So that it can be passed to the LSTM sequentially.

- **LSTM Layer :** LSTM is used for sequence processing and spot the temporal change between the frames.2048-dimensional feature vectors is fitted as the input to the LSTM. We are using 1 LSTM layer with 2048 latent dimensions and 2048 hidden layers along with 0.4 chance of dropout, which is capable to do achieve our objective. LSTM is used to process the frames in a sequential manner so that the temporal analysis of the video can be made, by comparing the frame at ‘t’ second with the frame of ‘t-n’ seconds. Where n can be any number of frames before t.

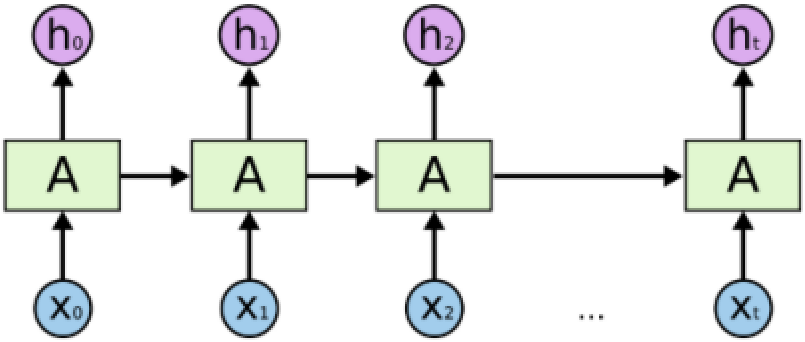


Figure 6.4: Overview of LSTM Architecture

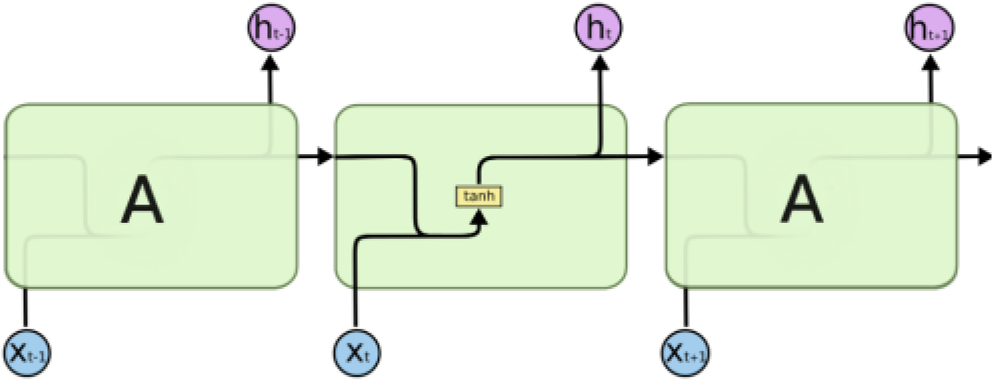


Figure 6.5: Internal LSTM Architecture

- **ReLU:** A Rectified Linear Unit is activation function that has output 0 if the input is less than 0, and raw output otherwise. That is, if the input is greater than 0, the output is equal to the input. The operation of ReLU is closer to the way our biological neurons work. ReLU is non-linear and has the advantage of not having any backpropagation errors unlike the sigmoid function, also for larger Neural Networks, the speed of building models based off on ReLU is very fast.

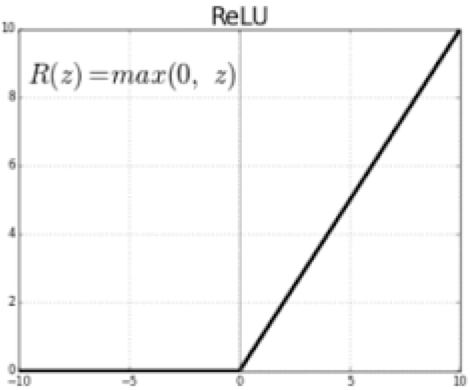


Figure 6.6: Relu Activation function

- **Dropout Layer :** Dropout layer with the value of 0.4 is used to avoid overfitting in the model and it can help a model generalize by randomly setting the output for a given neuron to 0. In setting the output to 0, the cost function becomes more sensitive to neighbouring neurons changing the way the weights will be updated during the process of backpropagation.

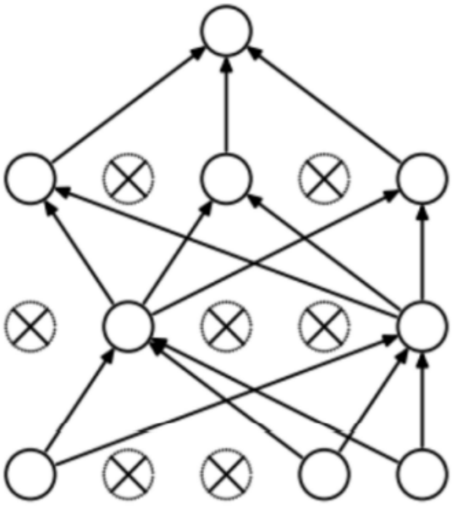


Figure 6.7: Dropout layer overview

- **Adaptive Average Pooling Layer :** It is used To reduce variance, reduce computation complexity and extract low level features from neighbourhood.2 dimensional Adaptive Average Pooling Layer is used in the mode

```
class Model(nn.Module):
    def __init__(self, num_classes,latent_dim= 2048, lstm_layers=1 , hidden_dim =
2048, bidirectional = False):
        super(Model, self).__init__()
        model = models.resnext50_32x4d(pretrained = True) #Residual Network CNN
        self.model = nn.Sequential(*list(model.children())[:-2])
        self.lstm = nn.LSTM(latent_dim,hidden_dim, lstm_layers,  bidirectional)
        self.relu = nn.LeakyReLU()
        self.dp = nn.Dropout2d(0.2)
        self.linear1 = nn.Linear(2048,num_classes)
        self.avgpool = nn.AdaptiveAvgPool2d(1)
    def forward(self, x):
        batch_size,seq_length, c, h, w = x.shape
        x = x.view(batch_size * seq_length, c, h, w)
        fmap = self.model(x)
        x = self.avgpool(fmap)
        x = x.view(batch_size,seq_length,2048)
        x_lstm,_ = self.lstm(x,None)
        return fmap,self.dp(self.linear1(torch.mean(x_lstm,dim = 1)))
```

6.3.4 ModelTraining Details

- **Train Test Split:** The dataset is split into train and test dataset with a ratio of 80% train videos and 20% test videos. The train and test split is a balanced split i.e 50% of the real and 50% of fake videos in each split.
- **Data Loader:** It is used to load the videos and their labels with a batch size of 4.
- Training: The training is done for 20 epochs with a learning rate of 1e-6 (0.000001),weight decay of 1e-3 (0.001) using the Adam optimizer.
- **Adam optimizer:** To enable the adaptive learning rate Adam optimizer with the model parameters is used.
- **Cross Entropy:** To calculate the loss function Cross Entropy approach is used because we are training a classification problem.
- **Softmax Layer:** A Softmax function is a type of squashing function. Squashing functions limit the output of the function into the range 0 to 1. This allows the output to be interpreted directly as a probability. Similarly, softmax functions are multi-class sigmoids, meaning they are used in determining probability of multiple classes at once. Since the outputs of a softmax function can be interpreted as a probability (i.e.they must sum to 1), a softmax layer is typically the final layer used in neural network functions. It is important to note that a softmax layer must have the same number of nodes as the output later. In our case softmax layer has two output nodes i.e REAL or FAKE, also Softmax layer provide us the confidence(probability) of prediction.

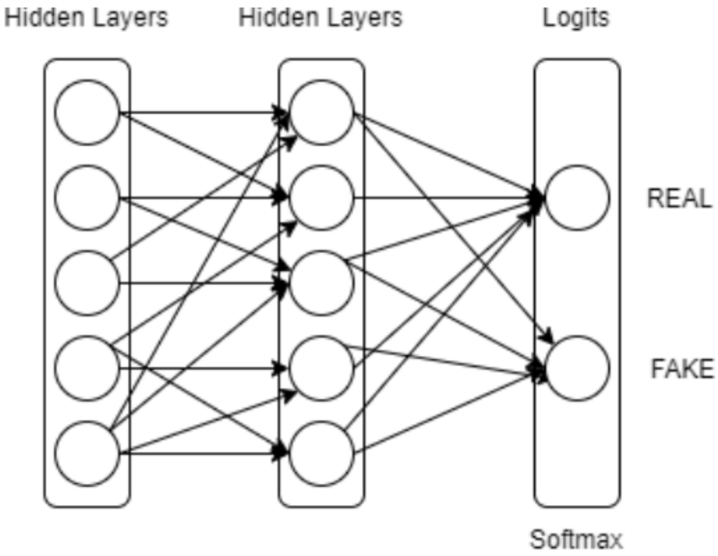


Figure 6.8: Softmax Layer

- **Confusion Matrix:** A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. The confusion matrix shows the ways in which your classification model is confused when it makes predictions. It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made. Confusion matrix is used to evaluate our model and calculate the accuracy.
- **Export Model:** After the model is trained, we have exported the model. So that it can be used for prediction on real time data.

6.3.5 Model Prediction Details

- The model is loaded in the application.
- The new video for prediction is preprocessed and passed to the loaded model for prediction.
- The trained model performs the prediction and return if the video is a real or fake along with the confidence of the prediction.

Chapter 7

Testing and Performance

7.1 Test Cases and Test Results

Case ID	Test Case Description	Expected Result	Actual Result	Status
1	Deepfake Video	Fake	Fake	Pass
2	Upload a real video	Real	Real	Pass
3	Upload a face cropped real video	Real	Real	Pass
4	Upload a face cropped fake video	Fake	Fake	Pass
5	Videos with many faces	Fake/Real	Fake	Pass

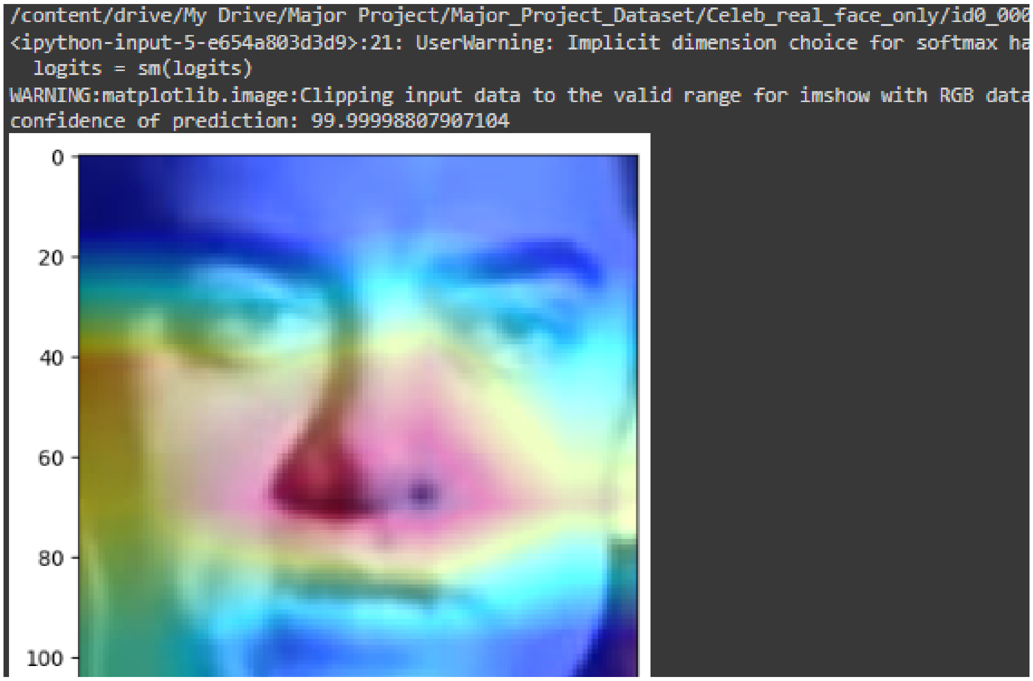
Table 7.1: Test Case Report

7.2 Model results

Model Name	Sequence Length	Accuracry
model_10_frames	10	83.22072072072072
model_20_frames	20	86.03603603603604
model_40_frames	40	89.1891891891892
model_60_frames	60	91.8271085934174

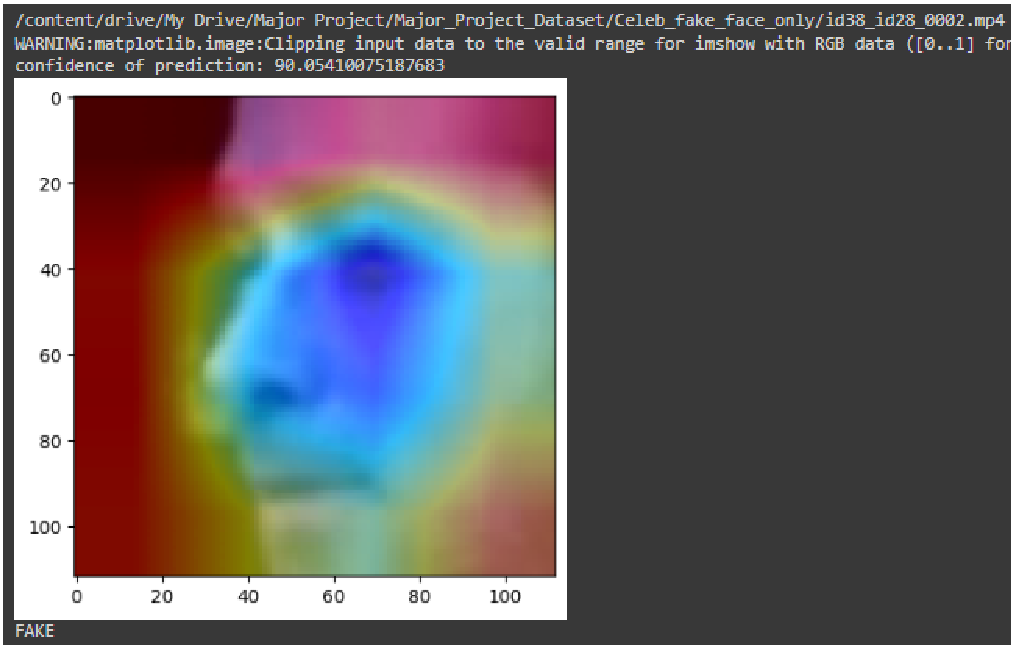
Table 7.2: Trained Model Results

7.3 Screen shots



```
/content/drive/My Drive/Major Project/Major_Project_Dataset/Celeb_real_face_only/id0_000
<ipython-input-5-e654a803d3d9>:21: UserWarning: Implicit dimension choice for softmax ha
  logits = sm(logits)
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data
confidence of prediction: 99.99998807907104
```

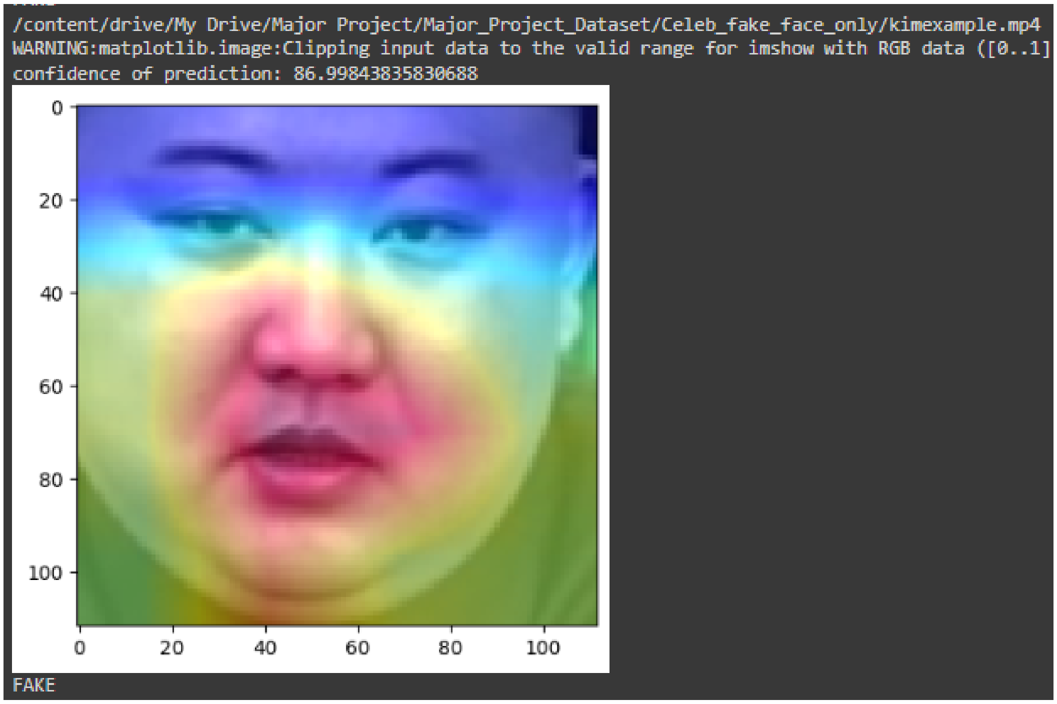
Figure 7.1: A real video predicted as real



```
/content/drive/My Drive/Major Project/Major_Project_Dataset/Celeb_fake_face_only/id38_id28_0002.mp4
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for
confidence of prediction: 90.05410075187683
```

```
FAKE
```

Figure 7.2: A fake video predicted as fake



```
/content/drive/My Drive/Major Project/Major_Project_Dataset/Celeb_fake_face_only/kimexample.mp4
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1]
confidence of prediction: 86.99843835830688
```

```
FAKE
```

Figure 7.3: A famous deepfake video predicted as fake

Chapter 8

Conclusion

8.1 Conclusion

We presented a neural network-based approach to classify the video as deep fake or real, The proposed method is capable of predicting the output by processing 1 second of video (10 frames per second) with a good accuracy. We implement the model by using pre-trained ResNext CNN model to extract the frame level features and LSTM for temporal sequence processing to spot the changes between the t and t-1 frame. The proposed model can process the video in the frame sequence of 10,20,40,60,80,100.

8.2 Future Scopes

There is always a scope for enhancements in any developed system, especially when the project build using latest trending technology and has a good scope in future.

- Web based platform can be upscaled to a browser plugin for ease of access to the user.
- Currently only Face Deep Fakes are being detected by the algorithm, but the algorithm can be enhanced in detecting full body deep fakes.

Chapter 9

References

8.1 References

[1] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies,Matthias Nießner, “FaceForensics++: Learning to Detect Manipulated Facial Images” in arXiv:1901.08971.

[2]Deepfake detection challenge dataset : https://www.kaggle.com/c/deepfake-detectionchallenge/data Accessed on 26 March, 2020

[3] 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

[4] Deepfake Video of Mark Zuckerberg Goes Viral on Eve of House A.I. Hearing : https://fortune.com/2019/06/12/deepfake-mark-zuckerberg/ Accessed on 26 March, 2020

[5] 10 deepfake examples that terrified and amused the internet : https://www.creativebloq.com/features/deepfake-examples Accessed on 26 March, 2020

[6] TensorFlow: https://www.tensorflow.org/ (Accessed on 26 March, 2020)

[7] Keras: https://keras.io/ (Accessed on 26 March, 2020)

[8] PyTorch : https://pytorch.org/ (Accessed on 26 March, 2020)

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

[10] J. Thies et al. Face2Face: Real-time face capture and reenactment of rgb videos. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2387–2395, June 2016. Las Vegas, NV.

[11] Face app: https://www.faceapp.com/ (Accessed on 26 March, 2020)

[12] Face Swap : https://faceswaponline.com/ (Accessed on 26 March, 2020)

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

[14] 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