VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI-590018

A Project Report on

“DEEPFAKE DETECTION”

SUBMITTED BY

NAME	USN
Harshit Sukumar Kallimani	2KL20CS025
Ketan Kumar Bagewadi	2KL20CS030
Prajwal Hampannavar	2KL20CS055
Sunanda Mange	2KL20CS106

UNDER THE GUIDANCE OF :-

Prof. GAMBHIR HALSE

Department of Computer Science and Engineering

KLE Dr. M.S. Sheshgiri College of

Engineering and Technology

Udayambag Belagavi-590008, Karnataka, India

2023-24

KLE Dr. M S SHESHGIRI
COLLEGE OF ENGINEERING & TECHNOLOGY,
BELAGAVI- 590008.

Department of Computer Science & Engineering

Certificate

This is to certify that the Project Report entitled **“DEEPFAKE DETECTION ”** carried out Harshit Sukumar Kallimani (2KL20CS025), Ketan Kumar Bagewadi (2KL20CS030), Prajwal Hampannavar (2KL20CS055), Sunanda Mange (2KL20CS106) are bonafide student of KLE Dr. M.S Sheshgiri College of Engineering & Technology Belagavi, in partial fulfillment for the award of Bachelor of Engineering in Computer Science branch of Technological University Belagavi, during the academic year 2023-2024.It is certified that all correction indicate have been incorporated in the report.

GUIDE	HOD	PRINCIPAL
Prof. Gambhir Halse	**Dr. Rajashri Khanai**	**Dr. S.F Patil**

EXTERNAL VIVA

Examiner: **Signature with date**

1.
2.

DECLARATION

We hereby declare that the Project work entitled **“DEEPFAKE DETECTION”** has been independently carried out by students mention below, under the guidance of **PROF. GAMBHIR HALSE** Department of Computer Science & Engineering, KLE Dr. M S Sheshgiri College of Engineering & Technology, Belagavi in partial fulfilment of the requirement for the award of degree of Bachelor of Engineering in Computer Science Engineering at Visvesvaraya Technological University Belagavi.

We further declare that no part of it has been submitted for the award, or diploma in any university or institute previously.

Place : Belagavi	Harshit Sukumar Kallimani – 2KL20CS025
Date : April 2024	Ketan Kumar Bagewadi – 2KL20CS030
	Prajwal Hampannavar – 2KL20CS055
	Sunanda Mange – 2KL20CS106

ACKNOWLEDGEMENT

Before we turn towards the project, we would like to add a few heartfelt words for the people who have been part of this project by supporting and encouraging us. In particular, we would like to take this opportunity to express our honour, respect, deep gratitude and genuine regards to our Guide, **PROF.GAMBHIR HALSE,** Professor, Department of Computer science for giving us all guidance required for our project apart from being a constant source of inspiration and motivation. We are grateful to **Dr. Rajashri Khanai**, Head of Computer Science & Engineering Department, for providing us the necessary help and encouragement whenever needed which has resulted in the success of this Project. We wish to express our sincere thanks to **Dr. S.F. Patil**, Principal of K.L.E Dr M.S Sheshgiri College of Engineering and Technology, Belgaum for providing a healthy environment in our college, which helped us in concentrating on our task. We owe special thanks to Our Parents for their moral support and warm wishes, and finally we would express our appreciation to all Our Friends for their support which helped us to complete this project successfully.

ABSTRACT

The growing computation power has made the deep learning algorithms so powerful that creating a indistinguishable human synthesized video popularly called asdeep fakes have became very simple. Scenarios where these realistic face swapped deep fakes are used to create political distress, fake terrorism events, revenge porn,blackmail peoples are easily envisioned. In this work, we describe a new deep learning-based method that can effectively distinguish AI-generated fake videos from real videos.Our method is capable of automatically detecting the replacement and reenactment deep fakes. We are trying to use Artificial Intelligence(AI) to fight Artificial Intelligence(AI). Our system uses a Res-Next Convolution neural network to extract the frame-level features and these features and further used to train the Long Short Term Memory(LSTM) based Recurrent Neural Network(RNN) to classify whether the video is subject to any kind of manipulation or not, i.e whether the video is deep fake or real video. To emulate the real time scenarios and make the model perform better on real time data, we evaluate our method on large amount of balanced and mixed data-set prepared by mixing the various available data-set like Face-Forensic++[1], Deepfake detection challenge[2], and Celeb-DF[3]. We also show how our system can achieve competitive result using very simple and robust approach.

Keywords:
Res-Next Convolution neural network.
Recurrent Neural Network (RNN).
Long Short Term Memory(LSTM).
Computer vision

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 Project Idea

In the world of ever growing Social media platforms, Deepfakes are considered as the major threat of the AI. There are many Scenarios where these realistic face swapped deepfakes are used to create political distress, fake terorism events, revenge porn, blackmail peoples are easily envisioned.Some of the examples are Brad Pitt, Angelina Jolie nude videos. It becomes very important to spot the difference between the deepfake and pristine video.

We are using AI to fight AI.Deepfakes are created using tools like FaceApp[11] and Face Swap [12], which using pre-trained neural networks like GAN or Auto encoders for these deepfakes creation. Our method uses a LSTM based artificial neural network to process the sequential temporal analysis of the video frames and pre-trained Res-Next CNN to extract the frame level features. ResNext Convolution neural network extracts the frame-level features and these features are further used to train the Long Short Term Memory based artificial Recurrent Neural Network to classify the video as Deepfake or real. To emulate the real time scenarios and make the model perform better on real time data, we trained our method with large amount of balanced and combination of various available dataset like FaceForensic++[1], Deepfake detection challenge[2], and Celeb-DF[3].Further to make the ready to use for the customers, we have developed a front end application where the user the user will upload the video.

1.2 Motivation of the Project

The increasing sophistication of mobile camera technology and the ever-growing reach of social media and media sharing portals have made the cre-ation and propagation of digital videos more convenient than ever before. Deep learning has given rise to technologies that would have been thought impossible only a handful of years ago. Modern generative models are one example of these, capable of synthesizing hyper realistic images, speech, music, and even video. These models have found use in a wide varities of applications, including making the world more accessible through text-to-speech, and helping generate training data for medical imaging.Like any trans-formative technology, this has created new challenges. So-called "deep fakes" produced by deep generative models that can manipulate video and audio clips. Since their first appearance in late 2017, many open-source deep fake generation methods and tools have emerged now, leading to a grow ing number of synthesized media clips. While many are likely intended to be humorous, others could be harmful to individuals and society. Until recently,the number of fake videos and their degrees of realism has been increasing due to availability of the editing tools, the high demand on domain expertise. Spreading of the Deep fakes over the social media platforms have become very common leading to spamming and peculating wrong information over the plat- form. Just imagine a deep fake of our prime minister declaring war against neighboring countries, or a Deep fake of reputed celebrity abusing the fans. These types of the deep fakes will be terrible, and lead to threatening, misleading of common people.

To overcome such a situation, Deep fake detection is very important. So, we describe a new deep learning-based method that can effectively distinguish AI- generated fake videos (Deep Fake Videos) from real videos. It’s incredibly important to develop technology that can spot fakes, so that the deep fakes can be identified and prevented from spreading over the internet

Chapter 2

LITERATURE SURVEY

In this comprehensive literature survey, we systematically reviewed 7 research papers addressing Deepfake Detection using Deep Learning.The paper “Deepfake Video Detection using Neural Networks” explores Face Warping Artifacts 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 describes a new method for detecting the deep-fakes 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 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. Our method is proposed to be trained on noiseless and real time datasets.Recurrent Neural Network (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 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 modelon large number of Realtime data Synthetic Portrait Videos using Biological Signals 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

detection of deepfake videos using deep learning models, specifically Xception and MobileNet. Deepfakes are synthetic media that replace a person in an image or video with someone else's likeness. The study focuses on the detection of deepfake videos generated by various platforms. Xception and MobileNet models were trained and evaluated using the FaceForensics++ dataset, achieving high detection accuracies ranging from 91-98%. A voting mechanism was developed to combine the outputs of the models for robust detection. The research also explored related works, dataset selection, training methodology, and results, including the impact of different deepfake generation methods on model performance. The study concluded by addressing future work and potential improvements in deepfake detection techniques.The research found that Xception and MobileNet models achieved high accuracy in detecting deepfake videos across different platforms. The voting mechanism provided a robust solution for detecting deepfake videos. The study emphasized the importance of considering inter-frame pattern correlations for video-oriented detection techniques and exploring the impact of different loss functions and optimizers on model performance. Additionally, the research outlined the need for user-friendly interfaces and the potential for further exploration of training models with specific facial features.

Abdulqader M. Almars et.al (2021) detection of deepfake videos using deep learning models. Deepfakes uses deep learning technology to manipulate images and videos of a person that humans cannot differentiatethem from the real one.Generative adversarial networks (GANs) are a form of deep neural network that has been commonly used to generate deep fake.Similar to GANs, FakeApp consists of the autoencoder which is used to construct latent features of the human face images and, the decoder which is used to re-extract the features for the human face imagesThis simple technique is powerful as it capable to produce extremely realistic fake videos that hard for people to differentiate from the real one.The architecture of VGGFace is improved by adding two layers called adversarial loss and perceptual lost.Those layers is added to autoencoder-decoder capture latent features of face images such as eye movements in order to produce more believable and realistic fake images.CycleGAN is a deepfake technique that extracts the characteristics of one image and produces another image with the same characteristics via the GAN architecture Deepfake image detection involves using deep learning techniques to identify manipulated images created by deepfake technology. Various methods have been proposed for detecting fake images, including neural network-based approaches that analyze statistical features of images and deep convolutional neural networks designed to detect fake image generation. These methods aim to distinguish between real and fake images, helping to address the increasing prevalence of deepfake content in social media and other platforms. Biological Signals Analysis: Detection models for fake videos use biological signals like eye blinking and heartbeat. One method employs a CNN with an RNN to spot fake face videos by focusing on eye blinking, proving effective in finding false images. Another model looks at the "heartbeat" of deepfakes, achieving a high 97.3% accuracy in spotting fake videos. Spatial and Temporal Features Analysis: convolutional neural network (CNN) for extracting frame features. These features undergo analysis in an LSTM layer, examining the temporal sequence for face manipulation between frames. The model's classification is done using a softmax function, distinguishing between real and fake videos.

MesoNet is an innovative neural network specifically engineered to combat the proliferation of deepfake videos, particularly in the context of social media platforms such as Instagram where content quality can be compromised due to compression and other factors. Unlike traditional deep learning architectures that might struggle with the nuances of low-quality, compressed content, MesoNet is finely tuned to excel in these challenging conditions.The architecture of MesoNet is meticulously crafted to address the unique obstacles presented by microscopic image noise analysis. It adopts an intermediate approach, striking a balance between complexity and effectiveness, with a deliberate focus on a limited number of layers to ensure efficient processing while maintaining accuracy.The network begins its processing pipeline with four layers of successive convolutions and pooling operations. This initial stage serves the crucial task of extracting intricate image features from the input data. Through a series of convolutions, the network identifies patterns and structures within the images, while pooling operations downsample the feature maps, reducing their dimensionality and retaining essential information.Following the convolutional layers, MesoNet incorporates a dense network with one hidden layer. This dense layer facilitates further abstraction of features extracted from the convolutional stage, enabling the network to capture higher-level representations of the input data.To introduce non-linearities and enhance regularization, Rectified Linear Unit (ReLU) activation functions and Batch Normalization are applied to the convolutional layers. ReLU activation functions ensure that the network can model complex relationships within the data, while Batch Normalization helps stabilize and accelerate the training process by normalizing the inputs to each layer.Moreover, MesoNet leverages Dropout within its fully-connected layers to fortify its robustness and promote generalization. Dropout randomly deactivates a portion of neurons during training, preventing overfitting and encouraging the network to learn more resilient and adaptable features.In summary, MesoNet represents a sophisticated fusion of convolutional and dense layers, strategically augmented with activation functions, normalization techniques, and dropout regularization. This carefully orchestrated architecture equips the network with the prowess to effectively detect deepfake videos, even amidst the challenges posed by low-quality, compressed content prevalent on social media platforms like Instagram.

The project employs a strategic combination of techniques and methodologies to effectively detect deepfake videos, utilizing DenseNet169 as the backbone for its detection framework while incorporating indicators for face warping artifacts, a common hallmark of deepfake manipulation. A key principle of the project is the avoidance of training face-swapping algorithms, instead relying on the compatibility with existing ones to enhance detection capabilities.Data collection for model training involves a comprehensive approach, leveraging internet photos for positive examples of authentic content and introducing various types of noise to generate negative examples. This noise augmentation encompasses Gaussian blur, exponential blur, and Rayleigh blur, simulating the imperfections often present in manipulated images.Face extraction is executed utilizing the dlib package, a widely-used toolkit for facial recognition and manipulation tasks. To further diversify the training data and enhance robustness, random affine transformations and specific blur effects are applied to resized pictures, augmenting the dataset with a variety of potential scenarios and challenges.Evaluation of the model's performance is conducted on the Celeb-DF dataset, comprising 1,000 videos featuring high-quality face-swapping algorithms. Each video undergoes meticulous analysis, with the classification process extending to every frame. If as little as 1% of the frames within a video are identified as modified, the entire video is flagged as a deepfake, ensuring a stringent threshold for detection accuracy.The project extends its detection criteria beyond videos to encompass photos or frames featuring multiple faces, ensuring comprehensive coverage and detection capabilities across various mediums and scenarios.

Chapter 3

PROBLEM STATEMENT

3.1 Problem Statement

Convincing manipulations of digital images and videos have been demonstrated for several decades through the use of visual effects, 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. These so-called AI-synthesized media (popularly referred to as deep fakes).Creating the Deep Fakes using the Artificially intelligent tools are simple task. But, when it comes to detection of these Deep Fakes, it is major challenge. Already in the history there are many examples where the deepfakes are used as powerful way to create political tension[14], fake terrorism events, revenge porn, blackmail peoples etc.So it becomes very important to detect these 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 Identifying deepfake videos is tough because they look real, and the technology keeps improving. Detection faces challenges like realistic visuals, new creation methods, diverse content types, easy-to-use tools, and a lack of comprehensive training data. Deepfakes can trick detection systems, and ethical concerns limit certain approaches. Real-time detection is difficult, and the rapid spread of deepfakes on social media complicates control. Researchers are working on improving detection, but it's an ongoing challenge.

Developing effective deepfake detection software faces challenges due to the evolving sophistication of deepfake generation techniques. Current solutions often struggle to keep pace with the rapid advancements in deepfake technology, leading to a pressing need for more robust and adaptable detection algorithms. Additionally, the balance between accuracy and real-time processing speed remains a critical concern, as quick and reliable identification of deepfakes is crucial in various applications, including media forensics and online content moderation. Addressing these challenges requires innovative approaches to enhance detection accuracy, efficiency, and resilience against emerging deepfake strategies.

3.1.1 Goals and objectives

Goal and Objectives:

- Our project aims at discovering the distorted truth of the deep fakes.
- Our project will reduce the Abuses' and misleading of the common people on the world wide web.
- Our project will distinguish and classify the video as deepfake or pristine.
- Provide a easy to use system for used to upload the video and distinguish whether the video is real or fake. GHRCEM-Wagholi,Pune, Department of Computer Engineering 2019-2020 7 Deepfake Video Detection

3.1.2 Statement of scope

There are many tools available for creating the deep fakes, but for deep fake detection there is hardly any tool available. Our approach for detecting the deep fakes will be great contribution in avoiding the percolation of the deep fakes 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 deep fake detection's. Even big application like WhatsApp, Facebook can integrate this project with their application for easy pre-detection of deep fakes before sending to another user. A description of the software with Size of input, bounds on input, input validation, input dependency, i/o state diagram, Major inputs, and outputs are described without regard to implementation detail.

3.2 Major Constraints

- User: User of the application will be able detect the whether the uploaded video is fake or real, Along with the model confidence of the prediction.
- Prediction: The User will be able to see the playing video with the output on the face along with the confidence of the model.

- Easy and User-friendly User-Interface: Users seem to prefer a more simplified process of Deep Fake video detection. Hence, a straight forward and user-friendly interface is implemented.The UI contains a browse tab to select the video for processing. It reduces the complications and at the same time enrich the user experience.
- Cross-platform compatibility: with an ever-increasing target market, accessibility should be your main priority. By enabling a cross-platform compatibility feature, you can increase your reach to across different platforms. Being a server side application it will run on any device that has a web browser installed in it

3.3 Methodologies of Problem solving

3.3.1 Analysis

- Solution Requirement

  We analysed the problem statement and found the feasibility of the solution of the problem. We read different research paper as mentioned in 3.3. After checking the feasibility of the problem statement. The next step is the dataset gathering and analysis. We analysed the data set in different approach of training like negatively or positively trained i.e training the model with only fake or real video's but found that it may lead to addition of extra bias in the model leading to inaccurate predictions. So after doing 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 a good accuracy.

- Solution Constraints

  We analysed the solution in terms of cost,speed of processing,requirements,level of expertise, availability of equipment's.

Parameter Identified

1. Blinking of eyes
2. Teeth enchantment
3. Bigger distance for eyes
4. Moustaches
5. Double edges, eyes, ears, nose
6. Iris segmentation
7. Wrinkles on face
8. Inconsistent head pose
9. Face angle
10. Skin tone
11. Facial Expressions
12. Lighting
13. Different Pose
14. Double chins
15. Hairstyle
16. Higher cheek bones

3.3.2 Design

After research and analysis we developed the system architecture of the solution as mentioned in the Chapter 6. We decided the baseline architecture of the Model which includes the different layers and their numbers.

3.3.3 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 Graphic Processing Unit (GPU) and it is customize-able. Google Cloud Platform for training the final model on large number of data-set.

3.3.4 Evaluation

We evaluated our model with a large number of real time dataset which include YouTube videos dataset. Confusion Matrix approach is used to evaluate the accuracy of the trained model.

3.4 Outcome

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

3.5 Applications

Web based application 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 rea

Chapter 4

OBJECTIVES

Deep fake is a technique for human image synthesis based on neural network tools like GAN(Generative Adversarial Network) or Auto Encoders etc. These tools super impose target images onto source videos using a deep learning techniques and create a realistic looking deep fake video. These deep-fake video are so real that it becomes impossible to spot difference by the naked eyes. In this work, we describe a new deep learning-based method that can effectively distinguish AI-generated fake videos from real videos. We are using the limitation of the deep fake creation tools as a powerful way to distinguish between the pristine and deep fake videos. During the creation of the deep fake the current deep fake creation tools leaves some distinguishable artifacts in the frames which may not be visible to the human being but the trained neural networks can spot the changes. Deepfake creation tools leave distinctive artefacts in the resulting Deep Fake videos, and we show that they can be effectively captured by Res-Next Convolution Neural Networks. Our system uses a Res-Next Convolution Neural Networks to extract frame-level features. These features are then used to train a Long Short Term Memory(LSTM) based Recurrent Neural Network(RNN) to classify whether the video is subject to any kind of manipulation or not, i.e whether the video is deep fake or real video.

We proposed to evaluate our method against a large set of deep fake videos collected from multiple video websites. We are tried to make the deep fake detection model perform better on real time data. To achieve this we trained our model on combination of available data-sets.So that our model can learn the features from different kind of images. We extracted a adequate amount of videos from Face-Forensic++[1], Deepfake detection challenge[2], and Celeb-DF[3] data-sets. We also evaluated our model against the large amount of real time data like YouTube data-set to achieve competitive results in the real time scenario's

To combat misinformation and safeguard digital integrity, we have developed a cutting-edge deepfake detection application. This tool serves multiple critical functions: Firstly, it identifies and flags manipulated media content, effectively stemming the spread of deceptive narratives and false information. By doing so, it shields users from falling victim to misleading content and helps preserve the authenticity of digital media. Moreover, our app acts as a shield against reputational damage by swiftly detecting and neutralizing malicious deepfake videos targeting individuals, public figures, and organizations. By safeguarding reputations, it upholds the integrity of online discourse and protects against orchestrated smear campaigns. Additionally, our platform empowers users with knowledge and awareness, offering real-time alerts and informative resources to help them recognize deepfake threats. Through education and vigilance, we aim to foster a culture of discernment and trust in digital media consumption, ensuring that users can navigate the online landscape with confidence and clarity.

Creating the DF using the Artificially intelligent tools are simple task. But, when it comes to detection of these DF, it is 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. 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 able to detect the temporal inconsistencies between frames introduced by the DF creation tools.Expected result against a large set of fake videos collected from standard data set. We show how our system can be competitive result in this task results in using a simple architecture.

Chapter 5
REQUIREMENTS ANALYSIS

SOFTWARE REQUIREMENTS:

Platform :

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

HARDWARE REQUIREMENTS:

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

	Parameter	Minimum Requiremen
	Intel Xeon E5 2637	3.5 GHz
	RAM	16 GB
	Hard Disk	100 GB
	Graphic card	NVIDIA GeForce GTX Titan (12 GB RAM)

Table 5.1: Hardware Requirements

Chapter 6

SYSTEM DESIGN

6.1 Introduction

6.1.1 System Architecture

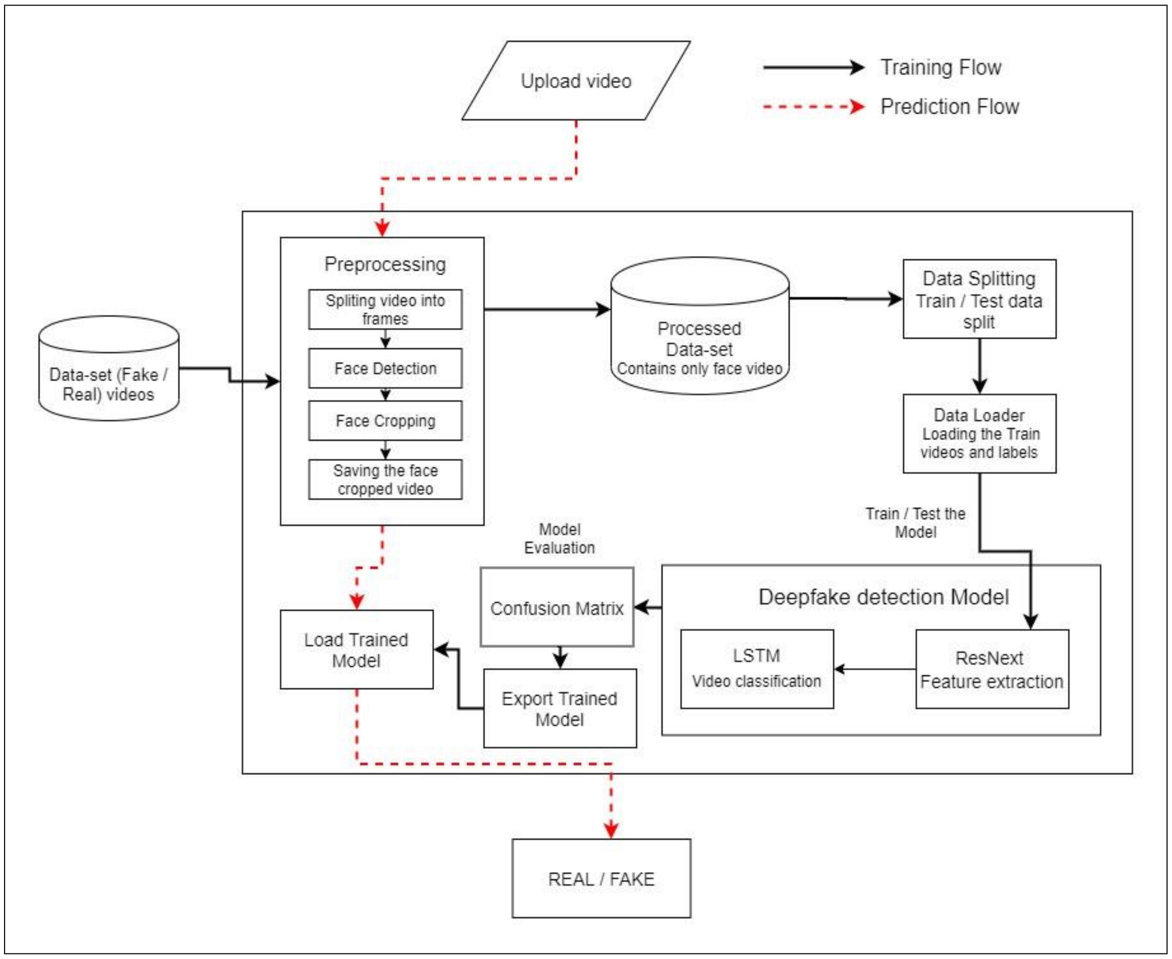

Upload video
Training Flow
Prediction Flow
Preprocessing
Spliting video into frames
Face Detection
Face Cropping
Saving the face cropped video
Processed Data-set Contains only face video
Data Splitting Train / Test data split
Data Loader Loading the Train videos and labels
Data-set (Fake / Real) videos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

Figure 6.1: System Architecture

In this system, we have trained our PyTorch deepfake detection model on equal number of real and fake videos in order to avoid the bias in the model. The system architecture of the model is showed in the figure. In the development phase, we have taken a dataset, preprocessed the dataset and created a new processed dataset which only includes the face cropped videos.

• Creating deepfake videos

To detect the deepfake videos it is very important to understand the creation process of the deepfake. Majority of the tools including the GAN and autoencoders takes 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 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 my removing the left-over traces by the deepfake creation model. Which result in creation of a deepfake looks realistic in nature. We have also used the same approach to detect the deepfakes. Deepfakes created using the pretrained neural networks models are very realistic that it is almost impossible to spot the difference by the naked eyes. But in reality, the deepfakes creation tools leaves some of the traces or artifacts in the video which may not be noticeable by the naked eyes. The motive of this paper to identify these unnoticeable traces and distinguishable artifacts of these videos and classified it as deepfake or real video.

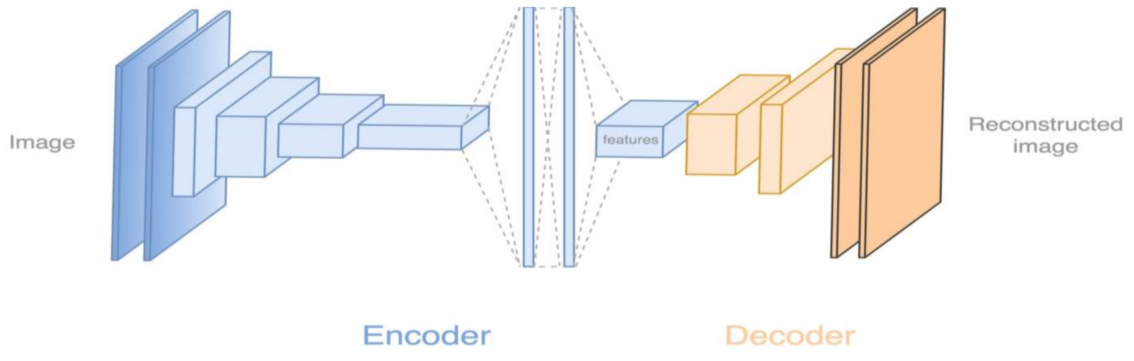


Figure 6.2: Deepfake generation

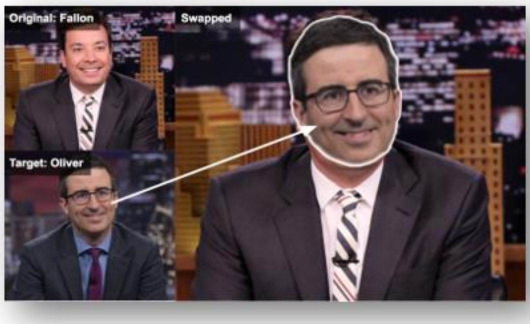


Figure 6.3: Face Swapped deepfake generation

Tools for deep fake creation.

1. Faceswap
2. Faceit
3. Deep Face Lab
4. Deepfake Capsule GAN
5. Large resolution face masked

6.2 Architectural Design

6.2.1 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]. Futher we have mixed the dataset the collected datasets and created our own new dataset, to accurate and real time detection on different kind of videos. To avoid the training bias of the model we have considered 50% Real and 50% fake videos.

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. After preprocessing of 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 consisting 3000 Real, 3000 fake videos and 6000 videos in total. Figure 2 depicts the distribution of the data-sets.

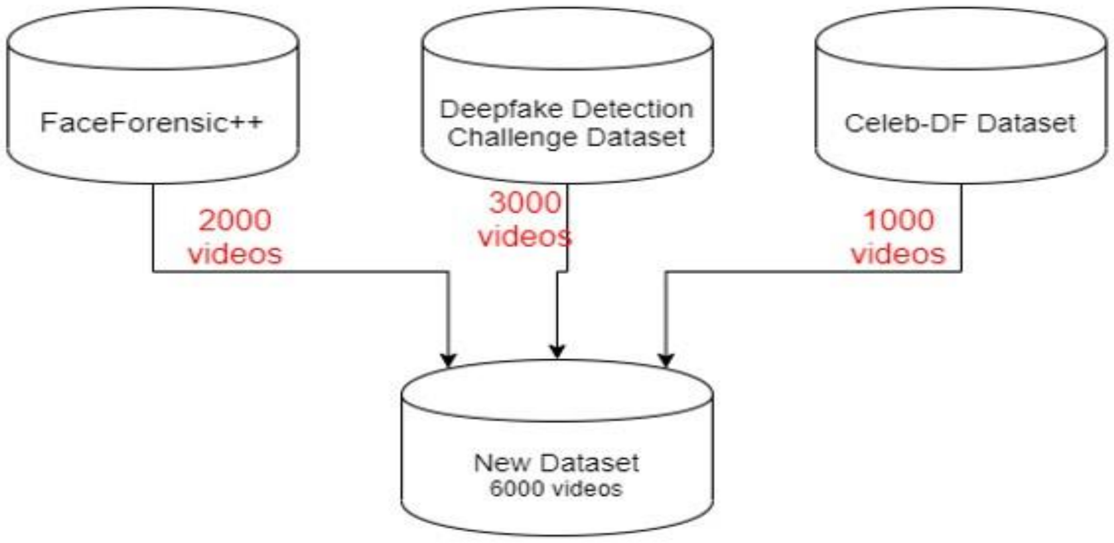


Figure 6.4: Dataset

6.2.2 Module 2 : Pre-processing

In this step, the videos are preprocessed and all the unrequired and noise is removed from videos. Only the required portion of the video i.e face is detected and cropped. The first steps 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 frame 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 creation of processed dataset containing face only videos. The frame that does not contain the face is ignored while preprocessing.

To maintain the uniformity of number of frames, we have selected a threshold value based on the mean of total frames count of each video. Another reason for selecting a threshold value is limited computation power. As a video of 10 second at 30 frames per second(fps) will have total 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 Graphic Processing Unit (GPU) computational power in 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 in the sequential manner i.e. first 150 frames and not randomly. The newly created video is saved at frame rate of 30 fps and resolution of 112 x 112.

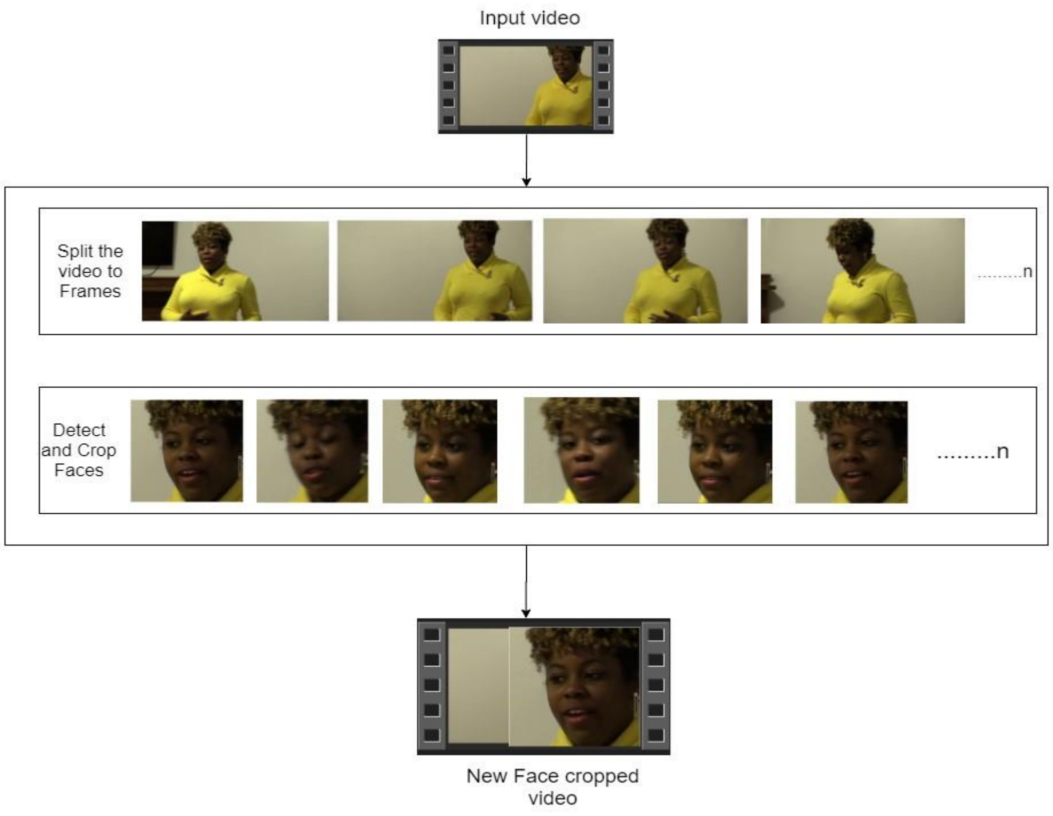


Figure 6.5: Pre-processing of video

6.2.3 Module 3: Data-set split

The dataset is split into train and test dataset 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.

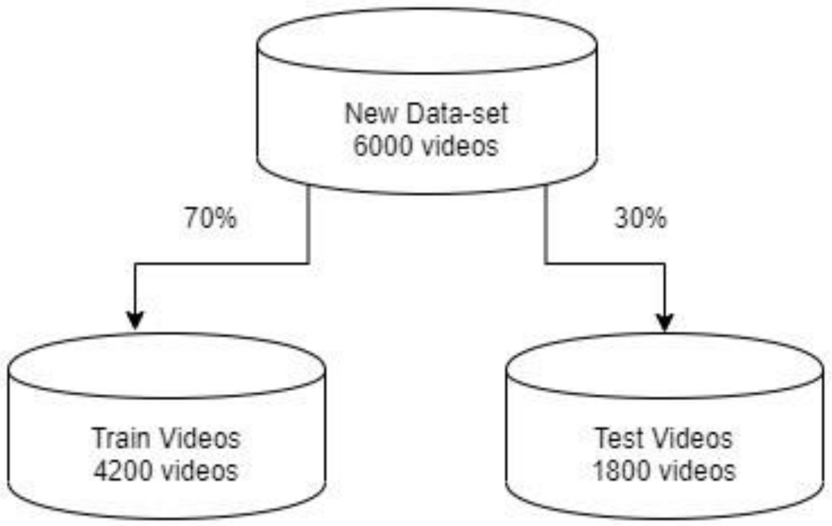


Figure 6.6: Train test split

6.2.4 Module 4: Model Architecture

Our model is a combination of CNN and RNN. We have used the Pre- trained ResNext CNN model to extract the features at frame level and based on the extracted features a LSTM network is trained to classify the video as deepfake or pristine. Using the Data Loader on 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 Residual CNN network optimized for high performance on deeper neural networks. For the experimental purpose we have used resnext50_32x4d model. We have used a ResNext of 50 layers and 32 x 4 dimensions.

Following, 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 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.

model also consists of 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 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 predication.

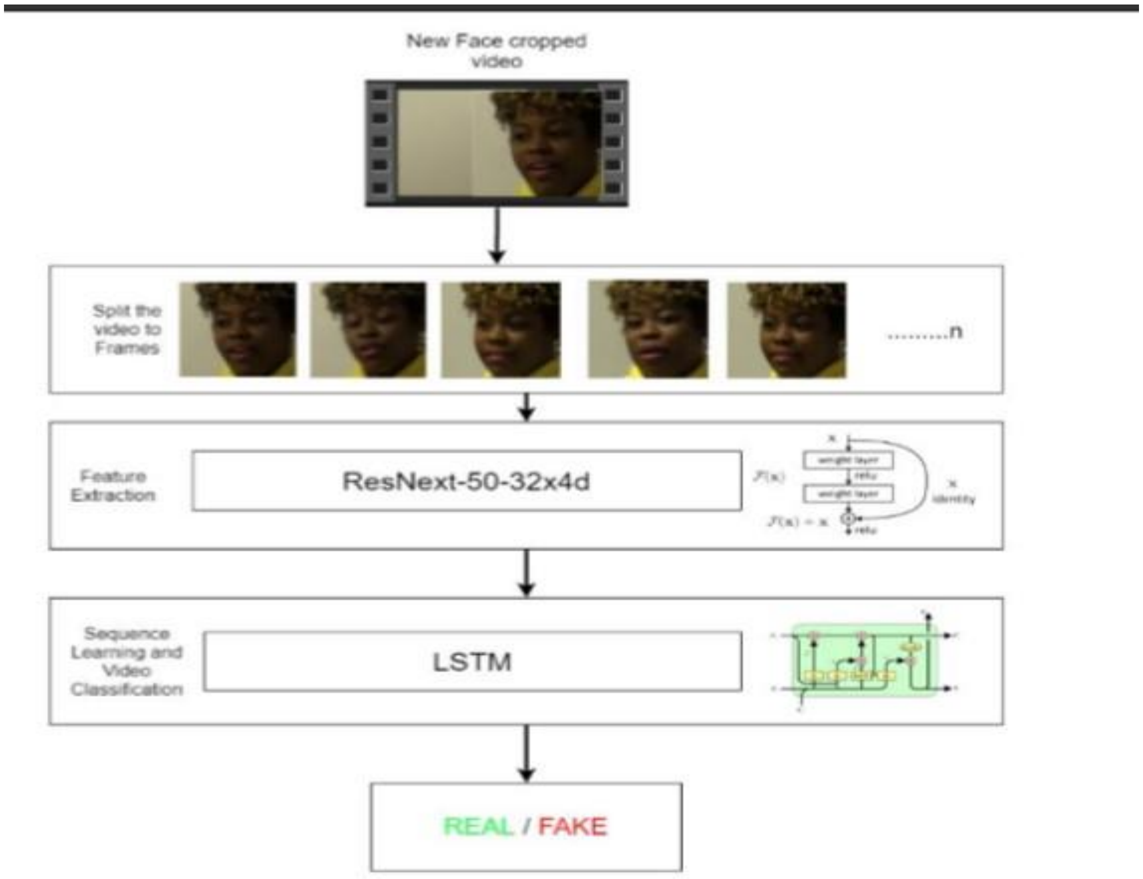


Figure 6.7: Overview of our model

6.2.5 Module 5: Hyper-parameter tuning

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

As this is a classification problem so to calculate the loss cross entropy approach is used.To use the available computation power properly the batch training is used. The batch size is taken of 4. Batch size of 4 is tested to be ideal size for training in our development environment. The User Interface for the application is developed using 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 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 7

METHODOLOGY

Introduction

7.1 Purpose and Scope of Document

This document lays out a project plan for the development of Deepfake video detection using neural network.The intended readers of this document are current and future developers working on Deepfake video detection using neural network and the sponsors of the project. The plan will include, but is not restricted to, a summary of the system functionality, the scope of the project from the perspective of the “Deepfake video detection” team (me and my mentors), use case diagram, Data flow diagram,activity diagram, functional and non- functional requirements, project risks and how those risks will be mitigated, the process by which we will develop the project, and metrics and measurements that will be recorded throughout the project.

7.1.2 Use Case View

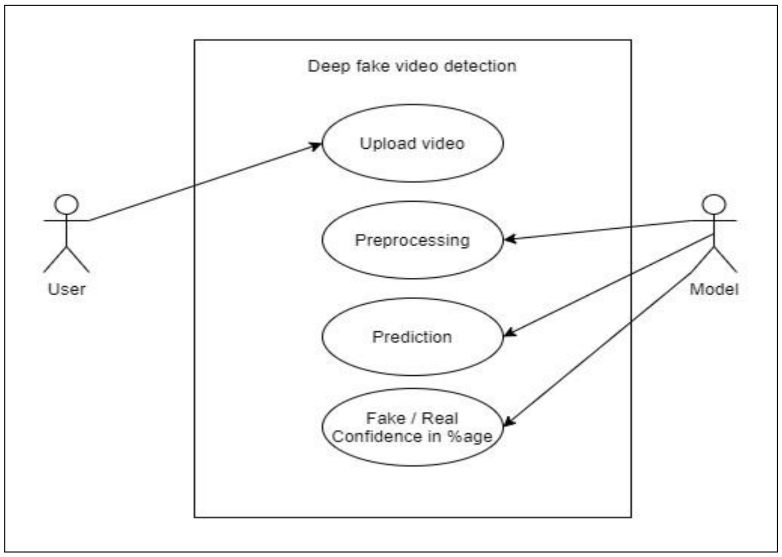


Figure 7.1: Use case diagram

7.2 Functional Model and Description

A description of each major software function, along with data flow (structured analysis) or class hierarchy (Analysis Class diagram with class description for object oriented system) is presented.

7.2.1 Data Flow Diagram

DFD Level-0

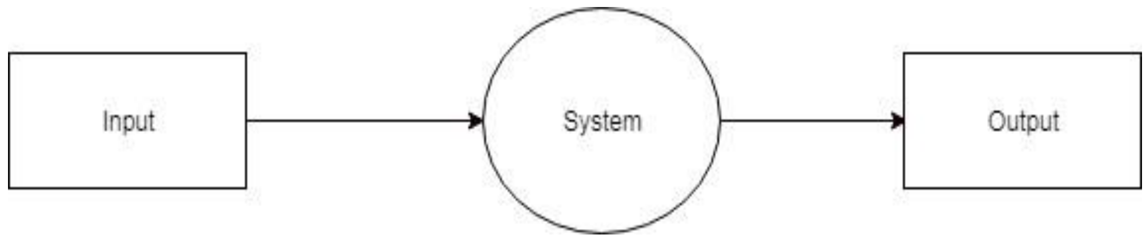


Figure 7.2: DFD Level 0

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

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

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

DFD Level-1

[1] DFD Level – 1 gives more in and out information of the system.

[2] Where system gives detailed information of the procedure taking place.

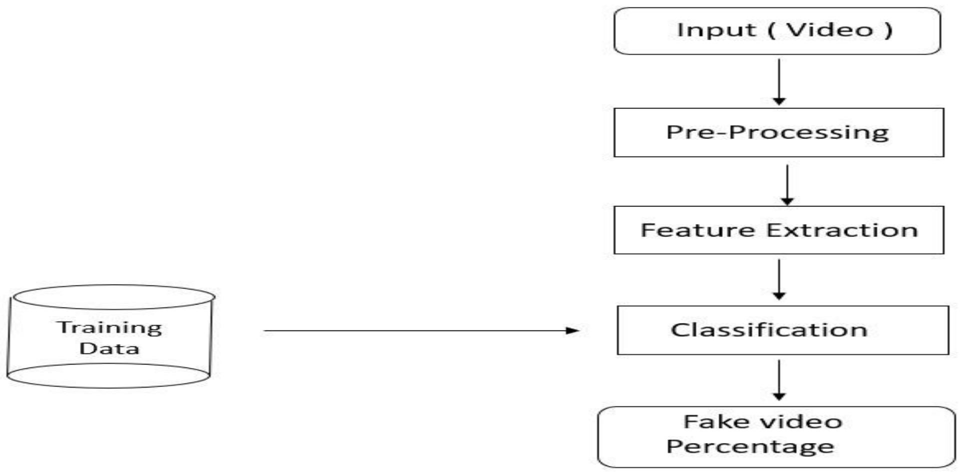


Figure 7.3: DFD Level 1

DFD Level-2

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

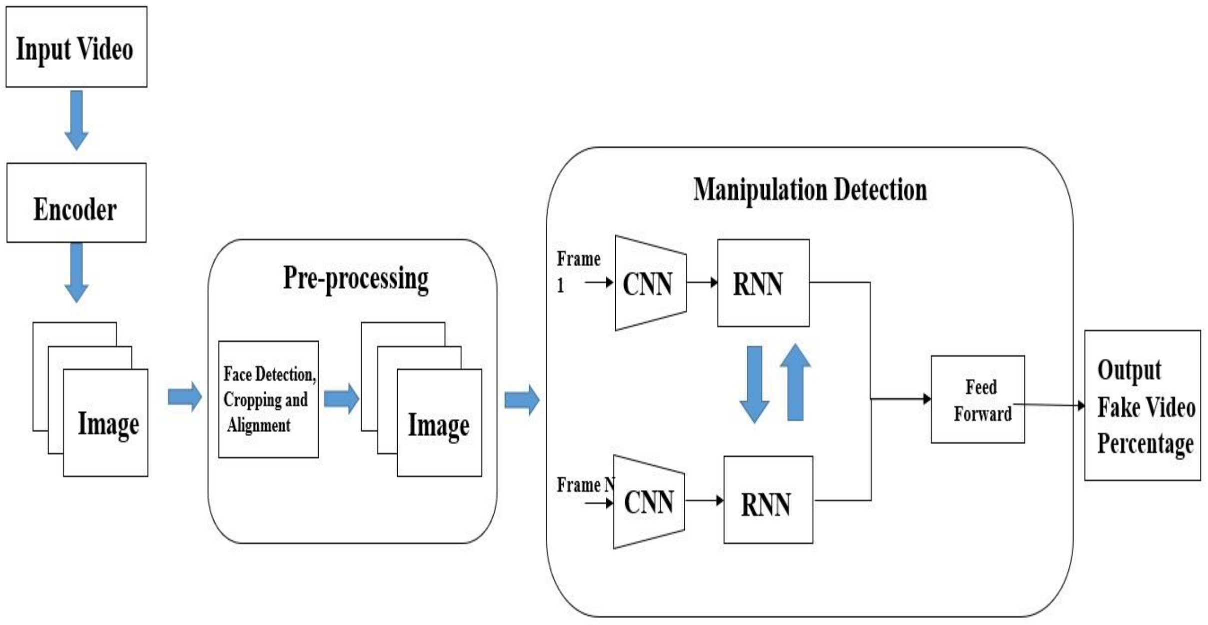


Figure 7.4: DFD Level 2

6.2.2 Activity Diagram:

Training Workflow:

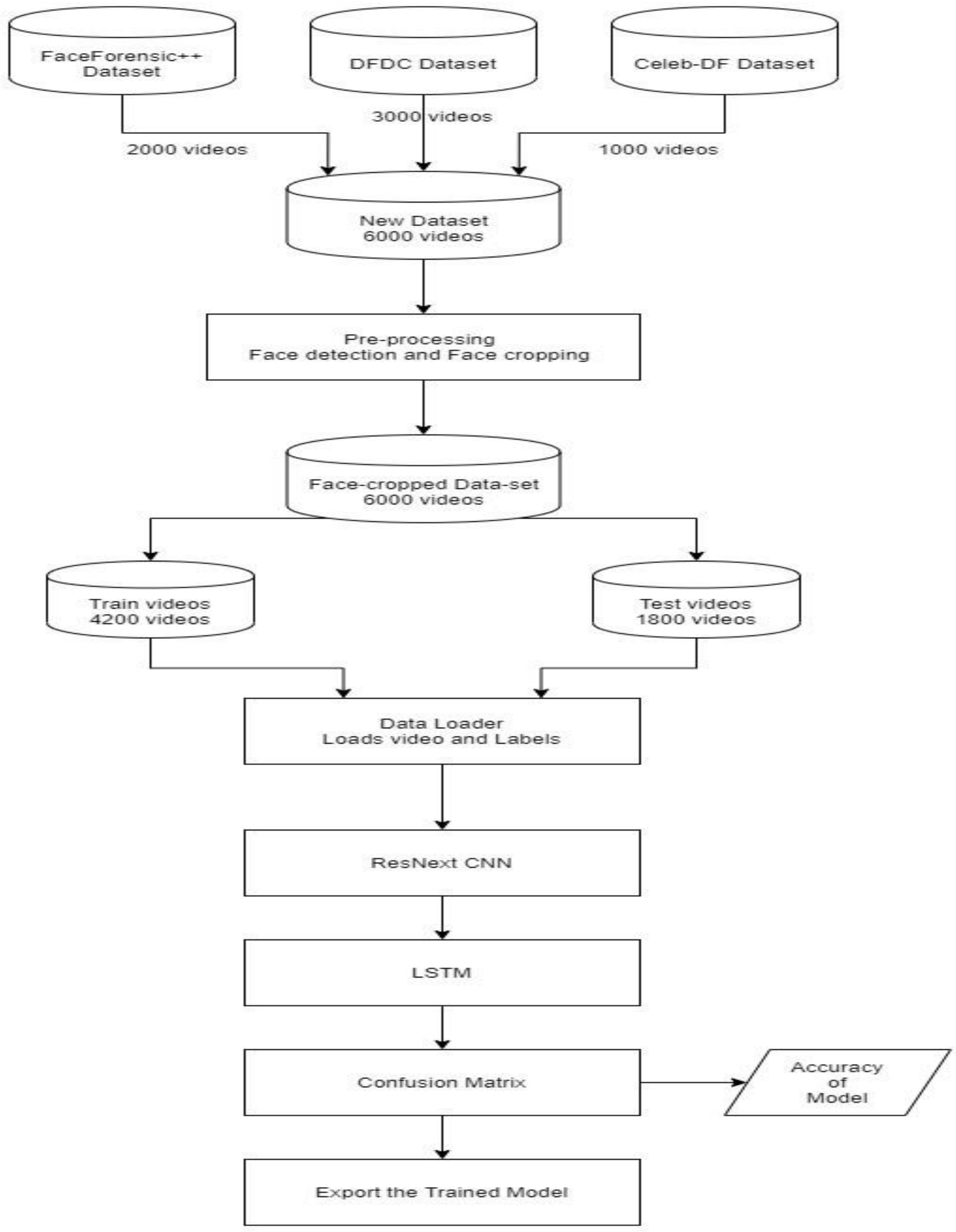


Figure 7.5: Training Workflow

Testing Workflow:

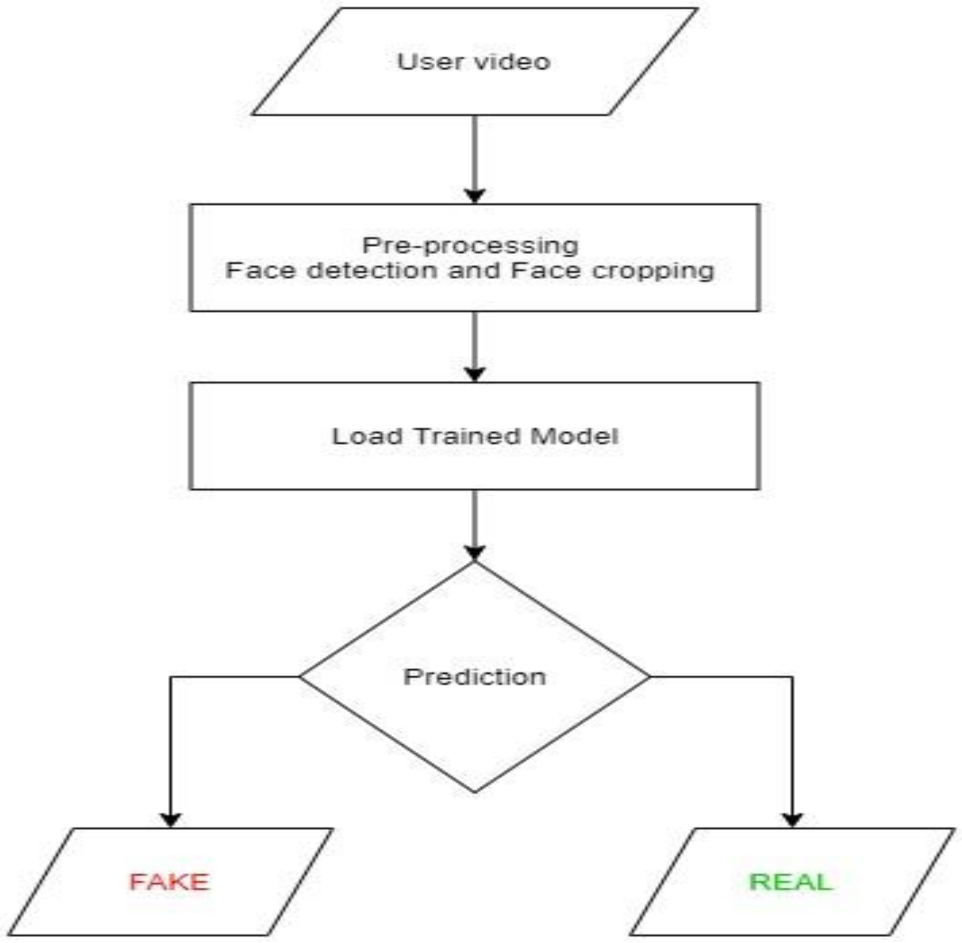


Figure 7.6: Testing Workflow

7.2.3 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 purpose.
- The design is versatile and user friendly.
- The application is fast, reliable and time saving.

- The system have universal adaptations.
- The system is compatible with future upgradation 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 secured from the human interventions, as the uploaded video is not are not able for 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 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 maintain the security and integrity of the video.
- SSL certification is made mandatory for Data security.

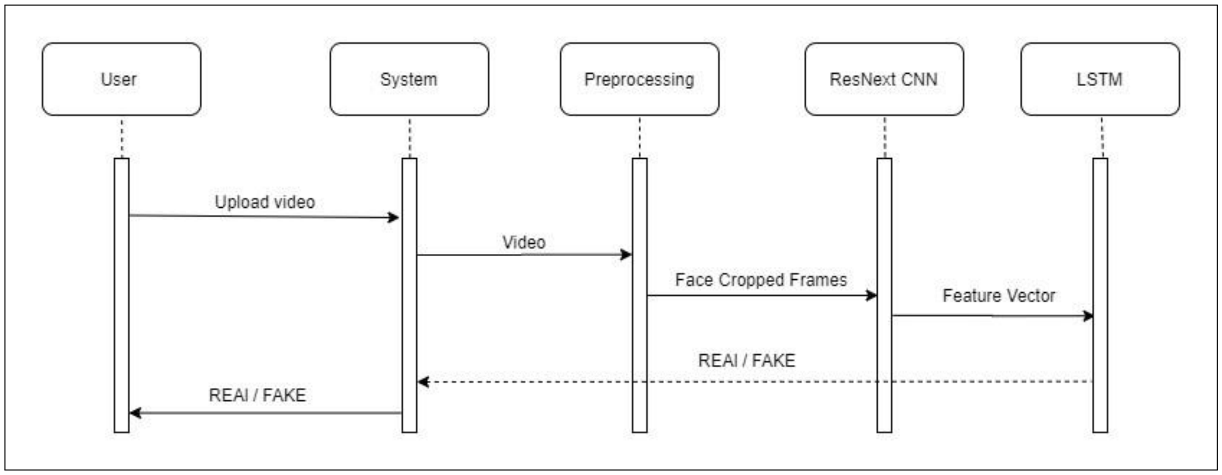


Figure 7.7: Sequence Diagram

Chapter 8
IMPLEMENTATION

8.1 Introduction

There are many examples where deepfake creation technology is used to mis lead the people on social media platform by sharing the false deepfake videos of he 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 [5]. 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 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[10] and Generative Adversarial Network (GAN) 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 [11,12] that generate deepfake videos are being extensively used to create fake celebrity pornographic videos or revenge porn [13]. Some of the examples are Brad Pitt and Angelina Jolie nude videos. The real-looking nature of the deepfake videos makes celebrities and other famous personalities the target of pornographic material, fake surveillance videos

8.2 Tools and Technologies Used

8.2.1 Planning

1. OpenProject

8.2.2UML Tools

1. draw.io

8.2.3 Programming Languages

1. Python3
2. JavaScript

8.2.4Programming Frameworks

1. PyTorch
2. Django

8.2.5IDE

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

8.2.6 Versioning Control

1. Git

8.2.7 Cloud Services

1. Google Cloud Platform

8.2.8 Application and web servers:

1. Google Cloud Engine

8.2.9 Libraries

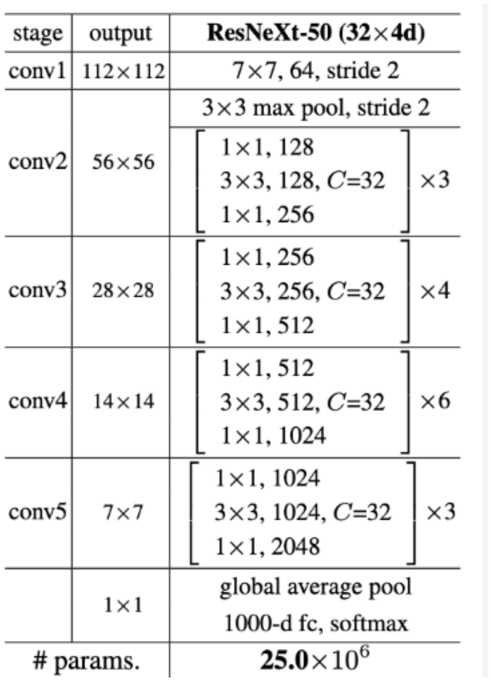
1. torch
2. torchvision
3. os
4. numpy
5. cv2
6. matplotlib
7. face_recognition
8. json
9. pandas
10. copy
11. glob
12. random
13. sklearn

8.2 Algorithm 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.

8.3.3 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()[22]. 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
		$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C{=}32 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3	28×28	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C{=}32 \\ 1\times1, 512 \end{bmatrix} \times 4$
conv4	14×14	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C{=}32 \\ 1\times1, 1024 \end{bmatrix} \times 6$
conv5	7×7	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, C{=}32 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	global average pool 1000-d fc, softmax
# params.		$\mathbf{25.0}\times10^6$

Figure 8.1: ResNext Architecture

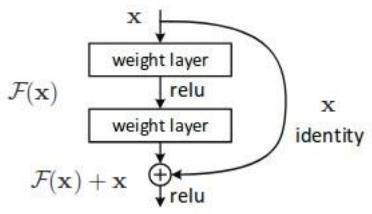

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 8 .2: ResNext Working

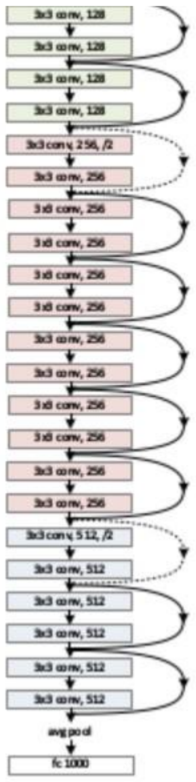


Figure 8.3: Overview of ResNext Architecture

- 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.

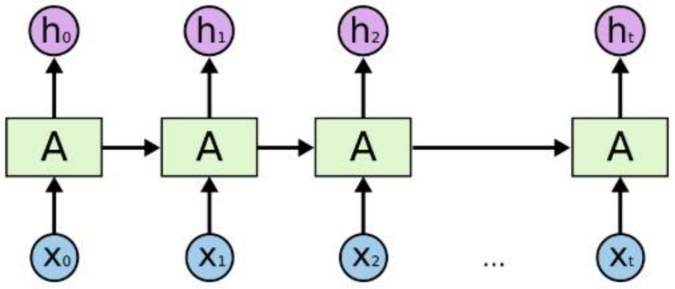


Figure 8.4: Overview of LSTM Architecture

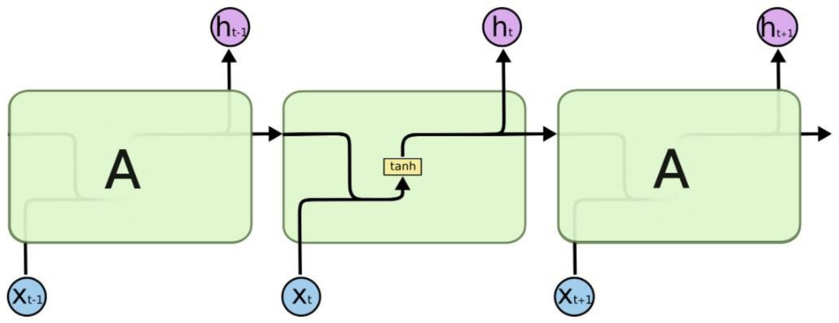


Figure 8.5: Internal LSTM Architecture

- 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.

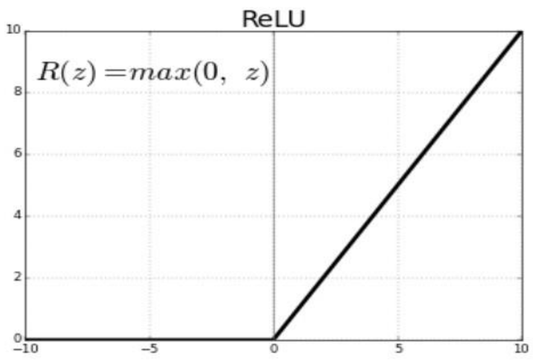


Figure 8.6: Relu Activation function

- 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.

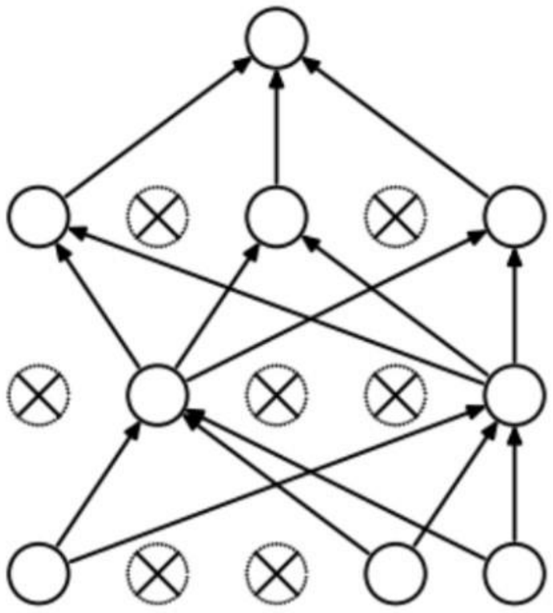


Figure 8.7: Dropout layer overview

- 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 model.

Model Training Details

- Train Test Split:The dataset is split into train and test dataset 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. Refer figure 7.6

- 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-5 (0.00001),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.

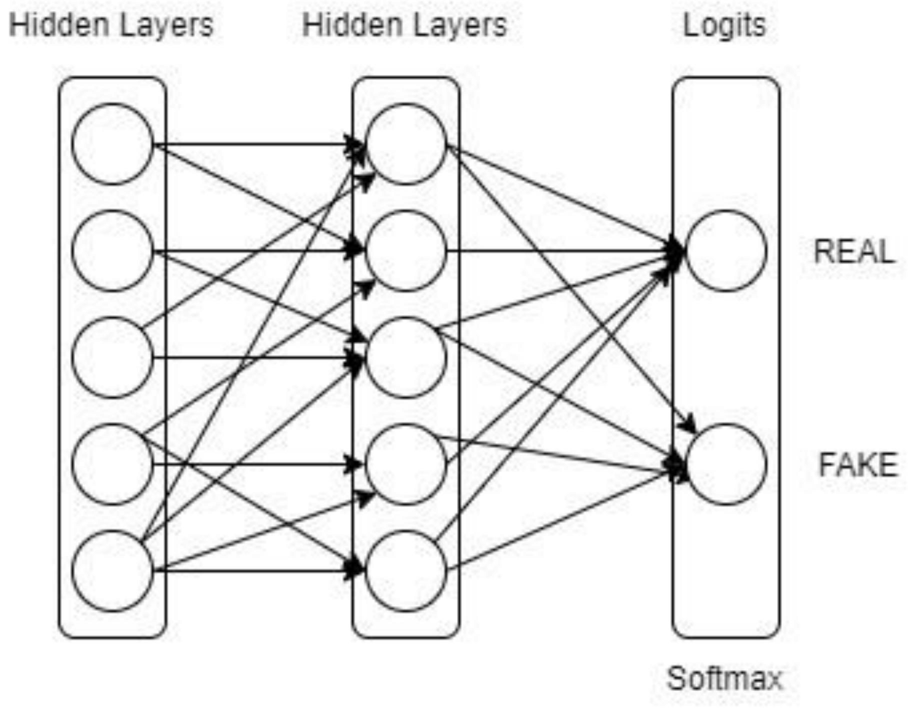


Figure 8.8: Softmax Layer

- 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.

Model Prediction Details

- The model is loaded in the application
- The new video for prediction is preprocessed(refer 8.3.2, 7.2.2) 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 9

TESTING

TESTING plays a vital role in the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. Once program code has been developed, testing begins.

1. **Unit Testing:**

Testing individual units or components of the code in isolation, Writing test cases for functions, classes, or modules to verify their behavior. To support this Python has it's built-in \`unittest\` framework or third-party libraries like \`pytest\`.

2. **Integration Testing:**

Testing the interactions between different components or modules of the system. Verifying that integrated parts of the system work together as expected.

3. **End-to-End Testing:**

Testing the entire system from end to end to ensure it behaves as expected. Simulating user interactions with the dashboard and verify the overall functionality. Selenium WebDriver for automated browser testing, or custom scripts to simulate user interactions.

4. **Performance Testing:**

Evaluate the performance of the code under various conditions, such as different input sizes or loads. Measuring execution time, memory usage, or other performance metrics for critical code paths. To Support this Python's \`timeit\` module for basic performance measurements, or more advanced profiling tools like \`cProfile\` or \`line_profiler\`.

5. **Acceptance Testing:**

Ensuring that the system meets the specified requirements and user expectations. Defining acceptance criteria and perform tests to verify that the system meets those criteria.

6. **Usability Testing:**

Evaluating the usability and user experience of the dashboard interface. User surveys, interviews, or usability testing tools.

7. **Security Testing:**

Identifying and mitigating security vulnerabilities in the codebase. Analyze the code for common security issues such as injection attacks, authentication flaws, or sensitive data exposure. Static code analysis tools like Bandit or manual code review by security experts.

9.1 Test Cases and Test Results

Case id	Test Case Description	Expected Result	Actual Result	Status
1	Upload a word file instead of video	Error message: Only video files allowed	Error message: Only video files allowed	Pass
2	Upload a 200MB video file	Error message: Max limit 100MB	Error message: Max limit 100MB	Pass
3	Upload a file without any faces	Error message:No faces detected. Cannot process the video.	Error message:No faces detected. Cannot process the video.	Pass
4	Videos with many faces	Fake / Real	Fake	Pass

5	Deepfake video	Fake	Fake	Pass
6	Enter /predict in URL	Redirect to /upload	Redirect to /upload	Pass
7	Press upload button without selecting video	Alert message: Please select video	Alert message: Please select video	Pass
8	Upload a Real video	Real	Real	Pass
9	Upload a face cropped real video	Real	Real	Pass
10	Upload a face cropped fake video	Fake	Fake	Pass

Table 9.1: Test Case Report

Chapter 10

APPENDIX

1. APPENDIX 1 (Details of Software/Tools used)

10.1 IDE: VISUAL STUDIO

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages and runtimes (such as C++, C#, Java, Python, PHP, Go, .NET).

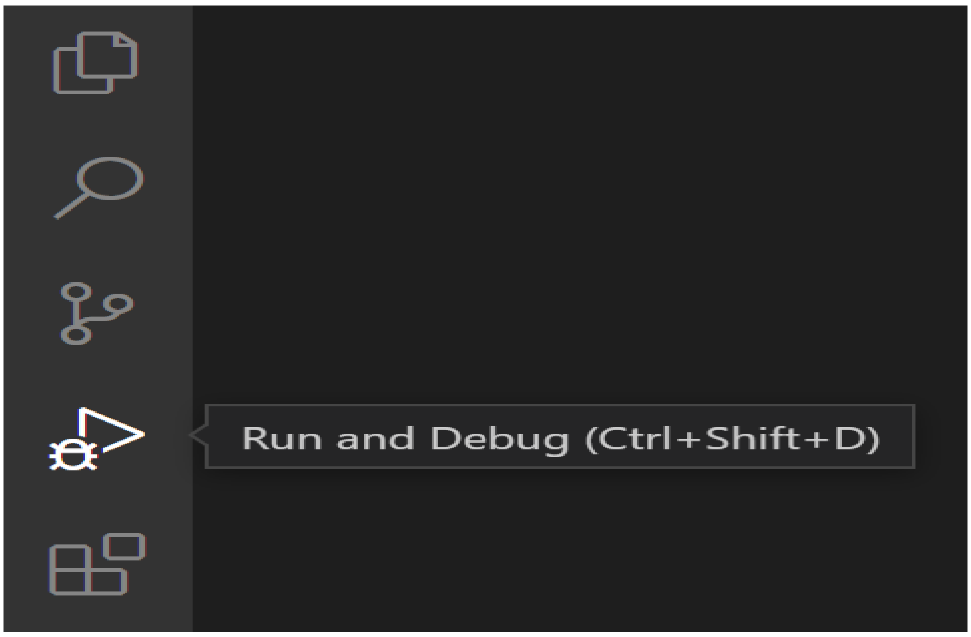
Start debugging

The following documentation is based on the built-in Node.js debugger, but most of the concepts and features are applicable to other debuggers as well.

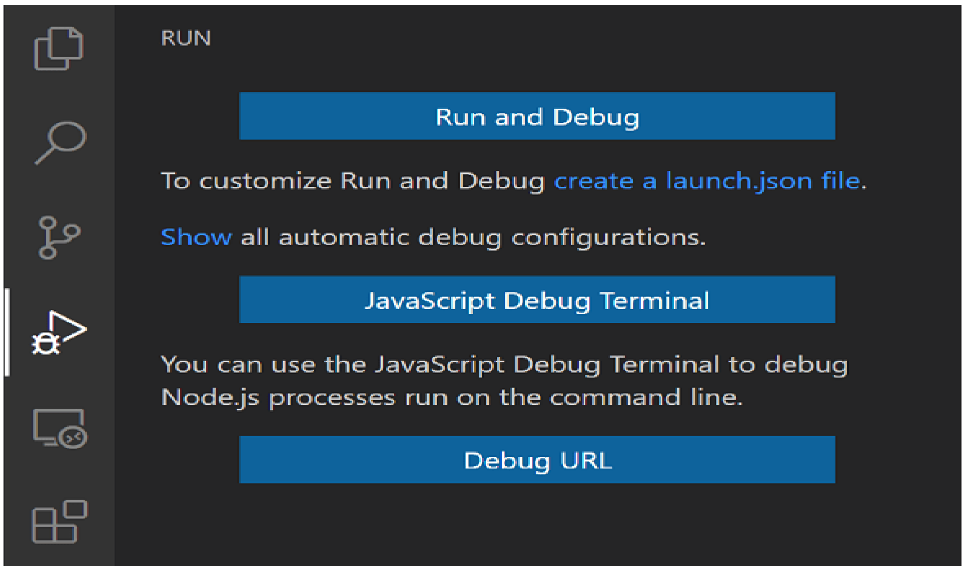
It is helpful to first create a sample Node.js application before reading about debugging. You can follow the Node.js walkthrough to install Node.js and create a simple "Hello World" JavaScript application (app.js). Once you have a simple application set up, this page will take you through VS Code debugging features.

Run and Debug view

To bring up the **Run and Debug** view, select the **Run and Debug** icon in the **Activity Bar** on the side of VS Code. You can also use the keyboard shortcut Ctrl+Shift+D.



The **Run and Debug** view displays all information related to running and debugging and has a top bar with debugging commands and configuration settings. If running and debugging is not yet configured (no `launch.json` has been created), VS Code shows the Run start view.


RUN

Run and Debug

To customize Run and Debug create a launch.json file.

Show all automatic debug configurations.

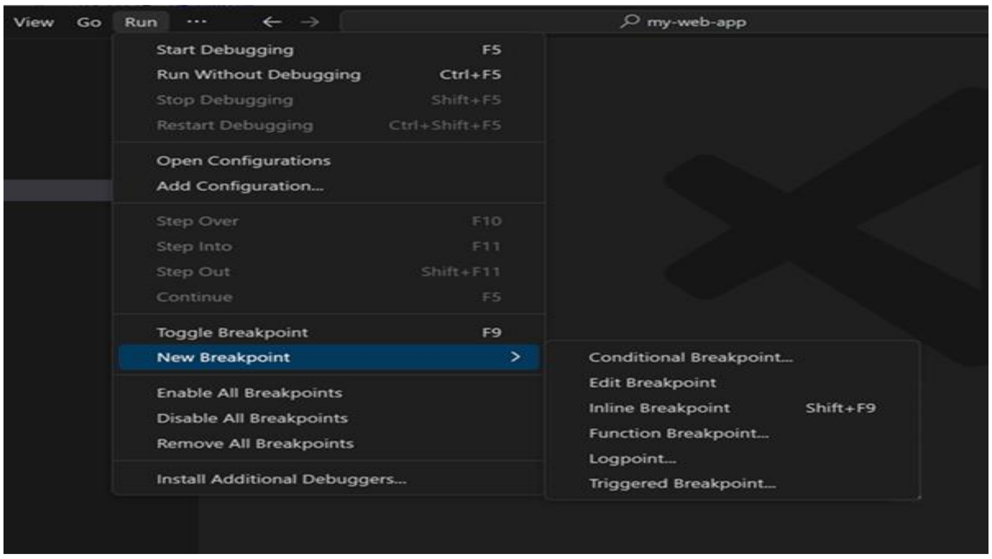
JavaScript Debug Terminal

You can use the JavaScript Debug Terminal to debug Node.js processes run on the command line.

Debug URL

Run menu

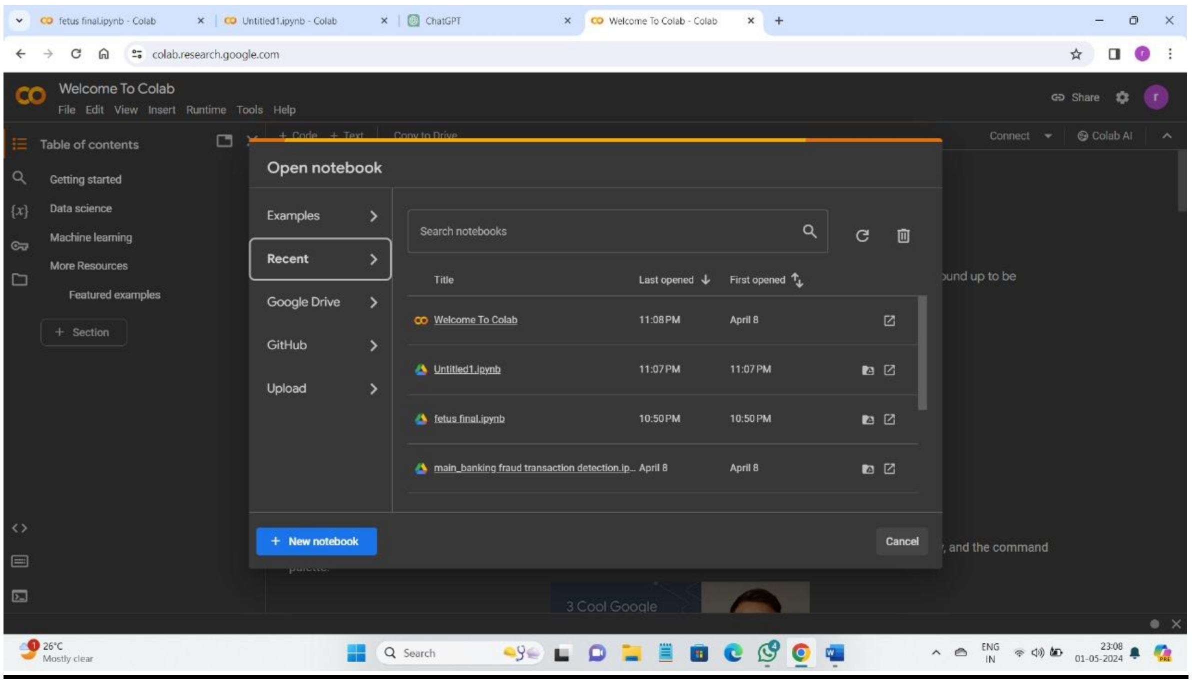
The top-level **Run** menu has the most common run and debug commands:



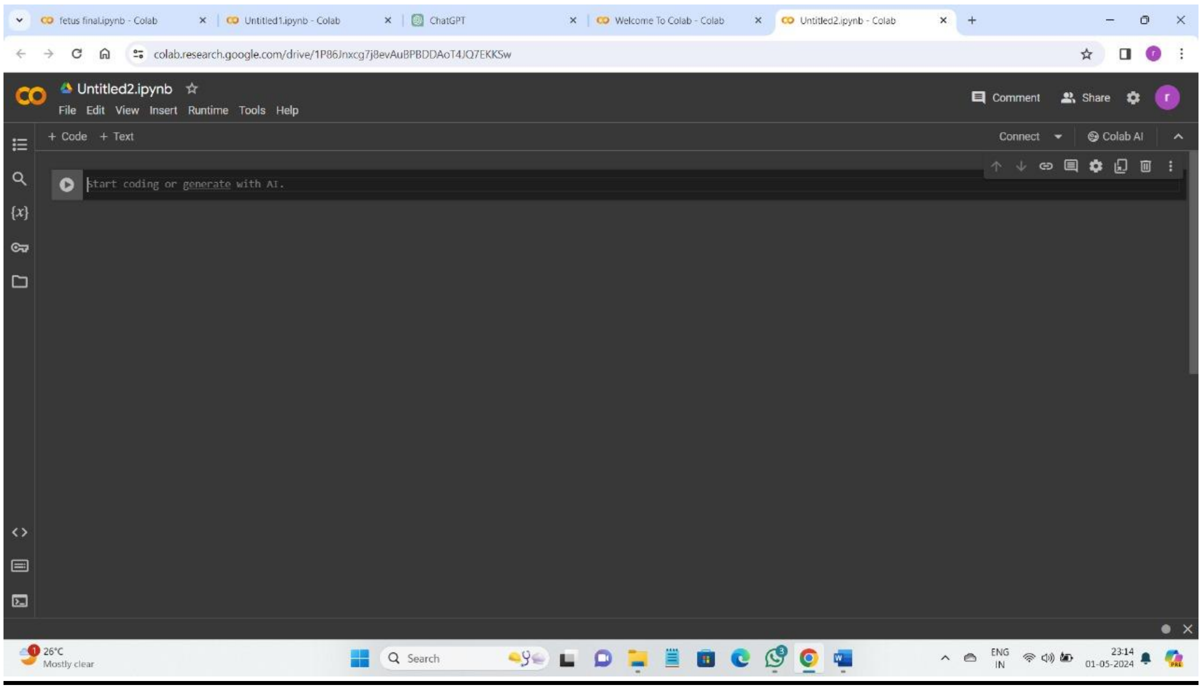
Launch configurations

To run or debug a simple app in VS Code, select **Run and Debug** on the Debug start view or press F5 and VS Code will try to run your currently active file.

10.2 GOOGLE COLAB



Title	Last opened	First opened
Welcome To Colab	11:08 PM	April 8
Untitled1.ipynb	11:07 PM	11:07 PM
fetus final.ipynb	10:50 PM	10:50 PM
main_banking fraud transaction detection.ip...	April 8	April 8



```
start coding or generate with AI.
```

How to get started with Google Colab?

1. Go to Google AI Studio and log in with your Google account.
2. Create an API key.
3. Use a quickstart for Python, or call the REST API using curl.
4. Explore use cases
5. Create a marketing campaign
6. Analyze audio recordings
7. Use System instructions in chat

What is Colab?

Colab, or "Colaboratory", allows you to write and execute Python in your browser, with

Zero configuration required
Access to GPUs free of charge
Easy sharing
Whether you're a student, a data scientist or an AI researcher, Colab can make your work easier. Watch Introduction to Colab to learn more, or just get started below!

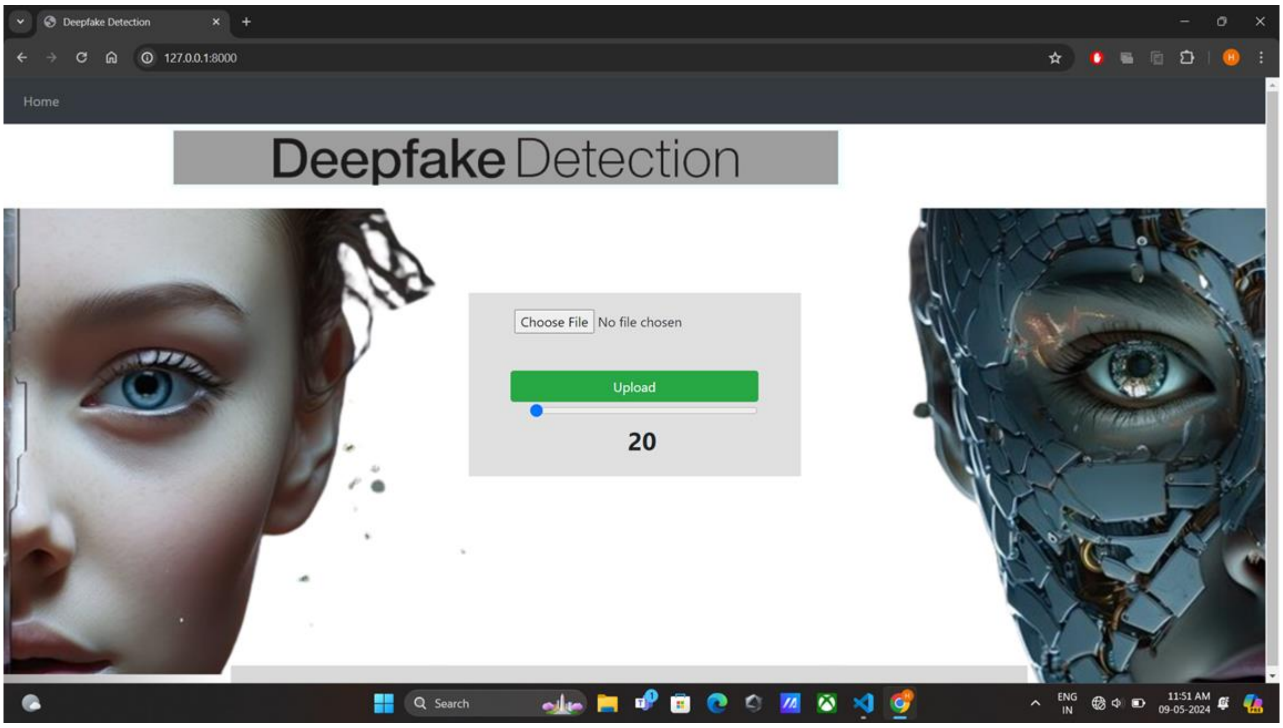
Getting started
The document you are reading is not a static web page, but an interactive environment called a Colab notebook that lets you write and execute code.

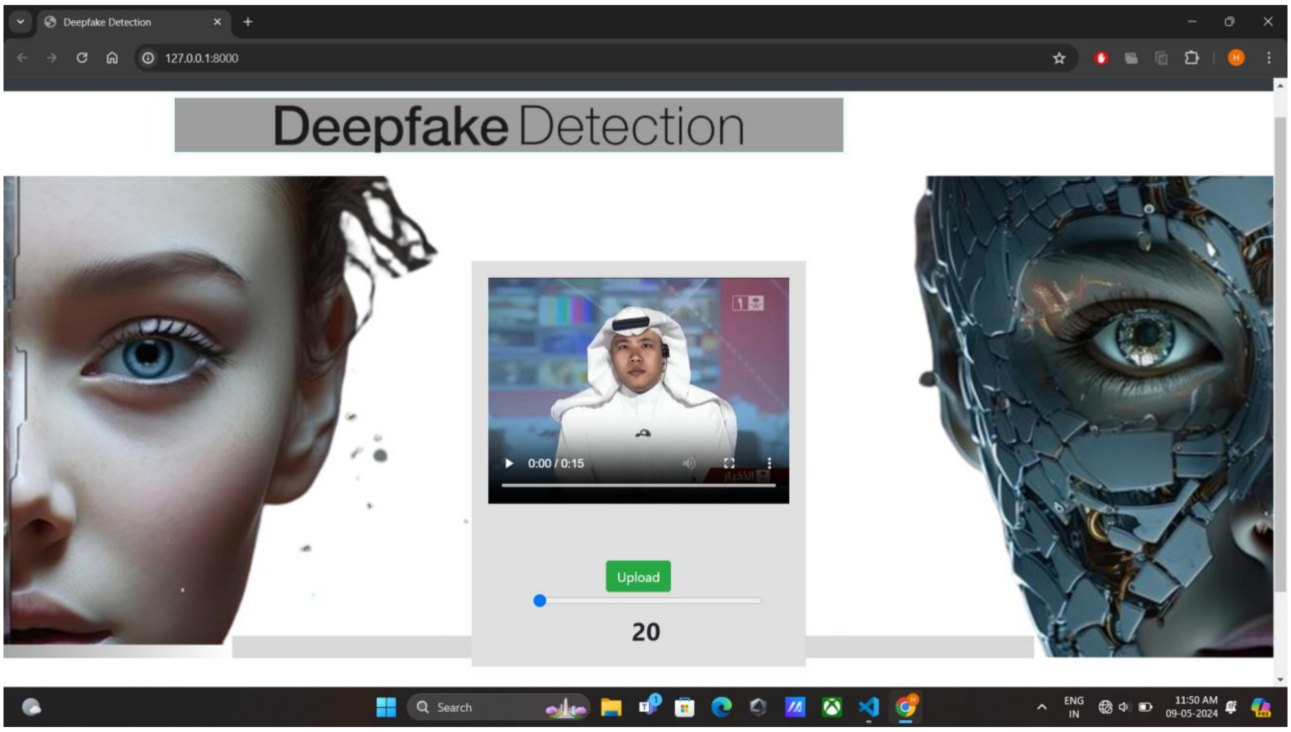
For example, here is a code cell with a short Python script that computes a value, stores it in a variable, and prints the result:

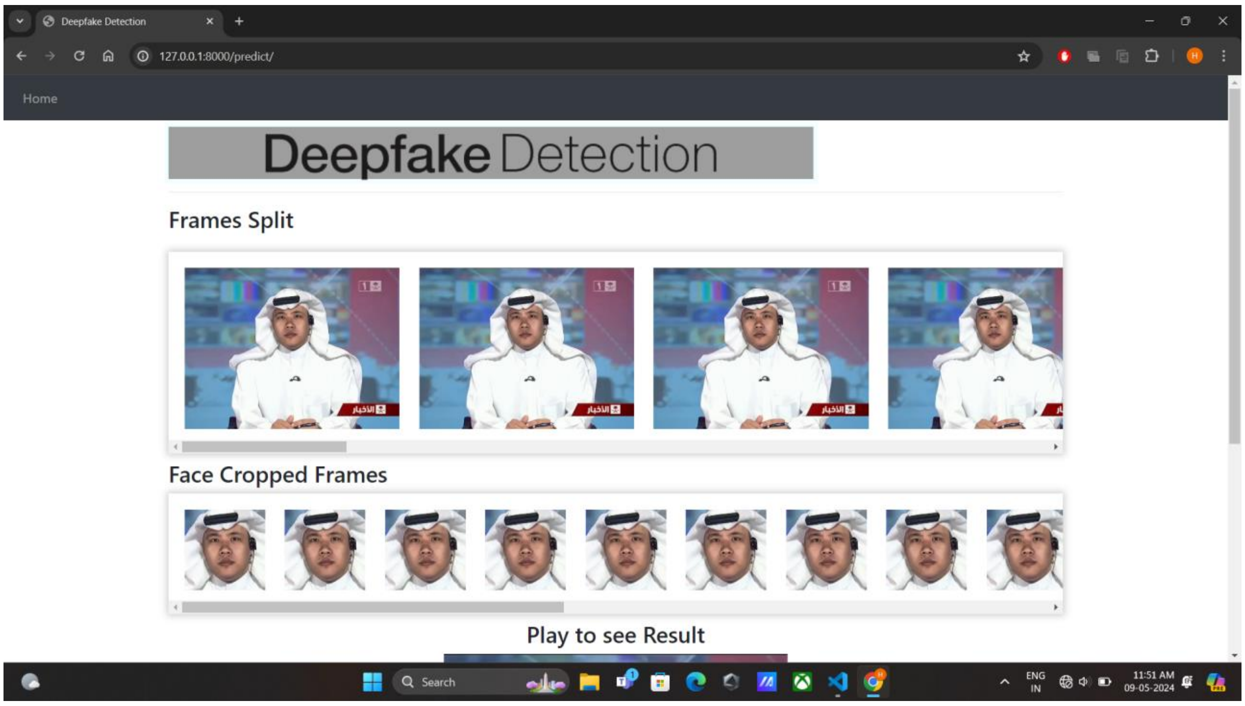
To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter". To edit the code, just click the cell and start editing. Variables that you define in one cell can later be used in other cells.

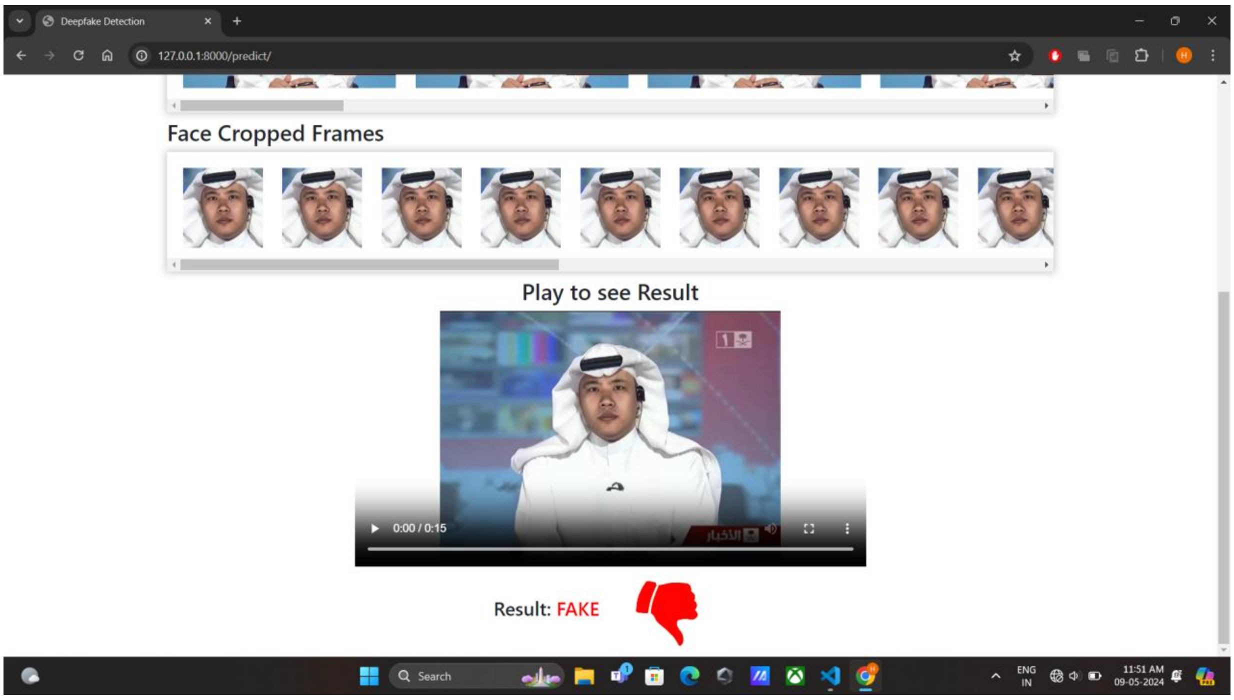
Chapter 11

Result






Deepfake Detection
127.0.0.1:8000/predict/
Home
Deepfake Detection
Frames Split
Face Cropped Frames
Play to see Result


Deepfake Detection
127.0.0.1:8000/predict/
Face Cropped Frames
Play to see Result
0:00 / 0:15
Result: FAKE

Chapter 12

Conclusion and Future Scope

12.1 Conclusion

We presented a neural network-based approach to classify the video as deep fake or real, along with the confidence of proposed model. Our method is capable of predicting the output by processing 1 second of video (10 frames per second) with a good accuracy. We implemented 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. Our model can process the video in the frame sequence of 10,20,40,60,80,100.

12.2 Future Scope

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

REFERENCES

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

[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: ALarge-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 : 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.

[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.

[14] The rise of the deepfake and the threat to democracy the-deepfake-and-the-threat-To democracy(2020).

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

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

[17] Huy H. Nguyen , Junichi Yamagishi, and Isao Echizen “ Using capsule networks to detect forged images and videos ” in arXiv:1810.11215.

[18] D. Güera and E. J. Delp, "Deepfake Video Detection Using Recurrent Neural Networks," 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Auckland, New Zealand, 2018, pp. 1-6.

[19] 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.

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

[21] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. arXiv:1412.6980, Dec. 2014.

[22] ResNext Model : https://pytorch.org/hub/pytorch_vision_resnext/ accessed on 06 April 2020.

[23] https://www.geeksforgeeks.org/software-engineering-cocomo-model/ Accessed on 15 April 2020.

[24] Deepfake video detection http://www.ijsrd.com/articles/IJSRDV8I10860.pdf.

[25] International Journal for Scientific Research and Development http://ijsrd.com/.