A MAJOR PROJECT REPORT ON

DETECTION OF DEEP FAKES USING DEEPLEARNING

Submitted in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

In

Computer Science and Engineering

By

T. SAI KISHORE (20A81A05I5)

V.V. SRI SAI TEJASWI (20A81A05J1)

G.V.S. SIVA RAM (20A81A05E1)

SK. SUBHAN SAHEB (20A81A05I3)

K. VIKAS KUMAR (20A81A05F4)

Under the Esteemed Supervision of

D.ANJANI SUPUTRI DEVI **M.Tech, (Ph.D)**

Sr.Asst.Professor

Department of Computer Science and Engineering (Accredited by N.B.A.)

SRI VASAVI ENGINEERING COLLEGE(Autonomous)

Pedatadepalli, Tadepalligudem-534101, A.P.

2023-2024

SRI VASAVI ENGINEERING COLLEGE (Autonomous)

Department Of Computer Science and Engineering

Pedatadepalli, Tadepalligudem

Certificate

This is to certify that the Project Report entitled "Detection of Deep Fakes Using Deep Learning" submitted by T. Sai Kishore (20A81A05I5), V.V. Sri Sai Tejaswi (20A81A05J1), G.V.S. Siva Ram (20A81A05E1), Sk.Subhan Saheb (20A81A05I3), K. Vikas Kumar (20A81A05F4) for the award of the degree of Bachelor of Technology in the Department of Computer Science and Engineering during the academic year 2023-2024.

Name of Project Guide	**Head of the Department**
D. Anjani Suputhri Devi,M.Tech,(Ph.D)	Dr. D. Jaya Kumari,M.Tech, Ph.D
Sr.Asst.Professor	Professor & HOD

External Examiner

DECLARATION

We here by declare that the project report entitled “**Detection of Deep Fakes Using Deep Learning**" submitted by us to Sri Vasavi Engineering College (Autonomous), Tadepalligudem, affiliated to JNTUK Kakinada in partial fulfillment of the requirement for the award of the degree of B.Tech in Computer Science and Engineering is a record of Bonafide project work carried out by us under the guidanceof D. Anjani Suputri Devi, M.Tech,(Ph.D),Sr.Asst.Professor. We further declare that the work reported in this project has not been submitted and will not be submitted, either in partor in full, for the award of any other degree in this institute or any other institute or University.

Project Associates
T. SAI KISHORE (20A81A05I5)
V.V. SRI SAI TEJASWI (20A81A05J1)
G.V.S. SIVA RAM (20A81A05E1)
SK. SUBHAN SAHEB (20A81A05I3)
K. VIKAS KUMAR (20A81A05F4)

ACKNOWLEDGEMENT

First and foremost, we sincerely salute to our esteemed institute **SRI VASAVI ENGINEERING COLLEGE,** for giving us this golden opportunity tofulfill our warm dream to become an engineer. Our sincere gratitude to our projectguide **D. Anjani Suputri Devi** M.Tech,(Ph.D),Sr.Asst.Professor. Department of Computer Science and Engineering**,** for her timely cooperation and valuablesuggestions while carrying out this project.

We express our sincere thanks and heartful gratitude to **Dr. D. Jaya Kumari**, Professor & Head of the Department of Computer Science and Engineering, for permitting us to do our project.

We express our sincere thanks and heartful gratitude to **Dr. G.V.N.S.R. Ratnakara Rao**, Principal, for providing a favorable environment and supporting us during the development of this project.

Our special thanks to the management and all the teaching and non-teaching staff members, Department of Computer Science and Engineering, for their support and cooperation in various ways during our project work. It is our pleasure to acknowledge the help of all those respected individuals.

We would like to express our gratitude to our parents, friends who helped to complete this project.

Project Associates

T. SAI KISHORE (20A81A05I5)

V.V. SRI SAI TEJASWI (20A81A05J1)

G.V.S. SIVA RAM (20A81A05E1)

SK. SUBHAN SAHEB (20A81A05I3)

K. VIKAS KUMAR (20A81A05F4)

ABSTRACT

The growing computation power has made the deep learning algorithms so powerful that creating an indistinguishable human synthesized video popularly called as deep fakes have become very simple. Scenarios where this 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.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

INTRODUCTION

The rise of deepfake technology has emerged as a significant concern in India, posing threats to both public figures and ordinary citizens alike. Recent incidents have underscored the potential dangers of deepfakes, with instances such as an actress's face being superimposed onto another person's body and a cricket player falsely portrayed endorsing dubious gaming software. These occurrences have not only sparked outrage and concern but have also prompted formal inquiries by law enforcement agencies, such as the Delhi and Mumbai Police, highlighting the urgent need for stringent regulations and public awareness campaigns. The global landscape is also grappling with similar challenges, as evidenced by controversies surrounding digital interactions with political figures and media outlets. The difficulty in discerning genuine content from deepfake manipulations underscores the critical importance of ensuring the authenticity of online interactions. Efforts to address this issue must encompass not only regulatory measures but also improvements in detection technologies and social media platform policies. The persistence of deepfake incidents necessitates continuous vigilance and collaborative efforts to mitigate the risks associated with AI-generated content.

MOTIVATION

The motivation behind our project stems from the alarming proliferation of deep fakes across the internet, which poses a significant threat to the integrity of digital content and the trustworthiness of online information. With the exponential growth of deepfake technology, there is an urgent need for robust detection mechanisms to combat the spread of manipulated videos. Our project aims to address this pressing issue by providing a user-friendly platform for individuals to upload videos and determine whether they are authentic or deepfake-generated. By offering a practical solution for distinguishing between real and fake content, we seek to empower users to navigate the digital landscape with greater confidence and safeguard against the potential harms of misinformation and deception. Moreover, our initiative holds promise for scalability, with the potential to expand beyond a standalone platform to integrate seamlessly with popular communication applications like WhatsApp and Facebook, thereby enhancing the accessibility and effectiveness of deep fake detection measures on a global scale.

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 detections. 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.

PROJECT OUTLINE

Our project aims to develop a web application utilizing neural networks to detect deepfake videos. With the increasing threat of misinformation spread through manipulated media, our initiative provides a user-friendly tool to distinguish between authentic and fake videos. By curating a diverse dataset and training state-of-the-art neural network models, integrated with Django backend, users can easily upload videos for immediate authenticity analysis. Through rigorous testing and ongoing refinement, our platform offers a robust solution to combat the proliferation of deepfake content.

CHAPTER 2
LITERATURE SURVEY

LITERATURE SURVEY

Title of the Paper: Deepfake Detection Method Based on Cross-Domain Fusion

Publisher: Hindawi Security and Communication Networks Volume 2021.

Authors: Afchar, Darius

Methodology: Cross-domain fusion

Conclusion: We propose a new deepfake detection method based on cross-domain fusion. However, our method is less effective on Neural Textures dataset as it has less consideration on subtle features

Title of the Paper: Adversarial Perturbations Fool Deepfake Detectors.

Publisher: 2020 International Joint Conference on Neural Networks (IJCNN).

Authors: Gandhi, Apurva, and Shomik Jain

Methodology: Defenses against adversarial perturbations in deep fakes.

Conclusion: This work uses adversarial perturbations to enhance deepfake images and fool common deepfake detectors.

Title of the Paper: Face X-Ray for More General Face Forgery Detection.

Publisher: IEEE -2020.

Authors: Li, Lingzhi

Methodology: Face X-ray by using CNN technique.

Conclusion: This work introduces a novel face forgery evidence called "face X-ray," exploiting intrinsic discrepancies in blending boundaries often overlooked by advanced face manipulation detectors.

Title of the Paper: Exposing Deep Fakes Using Inconsistent Head Poses.

Publisher: ICASSP 2019 - 2019 IEEE International Conference on Acoustics,Speech and Signal Processing (ICASSP).

Authors: Yang, Xin, Yuezun Li, and Siwei Lyu**.**

Methodology: Head poses.

Conclusion: Our method capitalizes on the splicing errors in Deep Fakes by exploiting discrepancies in 3D head pose estimation. We conduct experiments validating this approach and develop an SVM classifier using features derived from these cues, evaluated on real and Deep Fake face images.

Title of the Paper: Deepfake Video Detection Using Recurrent Neural Networks.
Publisher: IEEE- 2018
Authors: Guera, David, and ¨ Edward J. Delp**.**
Methodology: Passed through CNN for feature extraction.
Conclusion: Our system uses a convolutional neural network (CNN) to extract frame-level features. These features are then used to train a recurrent neural network (RNN) that learns to classify if a video has been subject to manipulation or not.

CHAPTER 3
SYSTEM STUDY AND ANALYSIS

PROBLEM STATEMENT

With the rise of sophisticated deep fake technology, there is a growing concern about its potential misuse, ranging from spreading misinformation to creating fraudulent content. In response to this threat, there is a critical need for robust deep fake detection systems capable of accurately discerning manipulated videos from authentic ones. This project aims to address this challenge by employing a combination of ResNext and LSTM architectures for deep fake detection.

LIMITATIONS OF EXISTING SYSTEM

The Existing System uses a variety of methods, such as Recurrent Neural Networks, Adversarial Perturbations, and Cross-Domain Fusion, to identify deep fakes. These methods do, however, have several drawbacks that make it more difficult for them to accurately detect edited videos.

Cross-domain fusion methods sometimes have trouble generalizing across different datasets, especially when confronted with changes in camera angles, lighting, or facial expressions. When applied to movies from unexplored domains, their performance may deteriorate, perhaps resulting in false positives or negatives, even though they can attain respectable accuracy inside certain domains.

Additionally, Recurrent Neural Networks (RNNs) may have trouble with long-range dependencies and have disappearing or ballooning gradient issues during training, even though they are excellent at capturing temporal dependencies in sequential data. This may make it more difficult for them to recognize minute temporal irregularities that point to deepfake alterations, particularly in videos with intricate and dynamic material.

All things considered, the shortcomings of the current system highlight the need for innovative strategies that can overcome these obstacles and improve the precision, resilience, and scalability of deep fake detection systems in practical settings.

PROPOSED SYSTEM

Our proposed system for detecting deepfake videos employs a sophisticated blend of advanced deep learning techniques, specifically utilizing LSTM-based neural networks and a pre-trained Res-Next CNN. This combination enables us to effectively analyze sequential temporal patterns within video frames while extracting crucial features unique to each frame. Through extensive training on diverse datasets such as Face Forensics++, Celeb-DF, DFDC, and DeeperForensics-1.0, our model is optimized to handle real-time scenarios with robust performance. Additionally, we have developed a user-friendly front-end application to facilitate seamless interaction, allowing users to upload videos for assessment of authenticity, alongside the confidence level provided by our model. Furthermore, our system incorporates our own dataset, enhancing performance, training accuracy, and enabling real-time identification of fraudulent videos. While acknowledging the potential biases in categorization procedures and the limited scope of existing datasets, our system aims to address these limitations through continuous refinement and adaptation.

ADVANTAGES OF PROPOSED SYSTEM

The proposed system offers several advantages over existing deep fake detection methods by utilizing ResNext for feature extraction and LSTM for temporal analysis. The model can extract rich hierarchical features by utilizing ResNext CNNs, which improves its capacity to recognize subtle indications that point to deep false manipulation. By capturing long-range dependencies and subtle temporal irregularities across video frames, the integration of LSTM networks enables dynamic temporal analysis and enhances detection accuracy. The architecture of the system also fosters generalization across various datasets and domains and improves resistance to adversarial attacks, all the while preserving scalability and processing efficiency for real-time deployment. All things considered, this integrated technique presents a viable option for dependable and successful deep fake detection in a range of scenarios.

FUNCTIONAL REQUIREMENTS

Feature Extraction with ResNext

The system should be capable of extracting high-level features from input video frames using ResNext convolutional neural networks.

Temporal Analysis with LSTM

It should perform dynamic temporal analysis on video sequences using Long Short-Term Memory (LSTM) networks to capture long-range dependencies and temporal inconsistencies.

Deep Fake Classification

The system must classify videos as either authentic or deep fakes based on the features extracted by ResNext and the temporal patterns identified by LSTM.

Integration of ResNext and LSTM

It should seamlessly integrate the feature extraction capabilities of ResNext with the temporal analysis capabilities of LSTM to create a unified deep fake detection model.

NON-FUNCTIONAL REQUIREMENTS

Accuracy

The system should achieve high detection accuracy, minimizing false positives and false negatives to ensure reliable identification of deep fake content.

Robustness

It must demonstrate robustness against adversarial attacks and variations in input data, maintaining consistent performance across diverse datasets and challenging conditions.

Scalability

The system should be scalable to accommodate varying workloads and datasets, allowing for efficient processing of large volumes of video data without compromising performance.

Computational Efficiency

It should be computationally efficient, utilizing hardware resources effectively to enable real-time or near-real-time detection while minimizing processing time and resource consumption.

User Interface

The system should feature a user-friendly interface that allows users to easily input video data, view detection results, and interact with the system for configuration and analysis purposes, enhancing usability and accessibility.

SYSTEM REQUIREMENTS

SOFTWARE REQUIREMENTS

Software Resources Required Platform:

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

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

HARDWARE REQUIREMENTS

1. CPU: A modern CPU with multi-core(Intel Core i7 or AMD Ryzen 7 series)
2. Memory (RAM) : 16GB RAM or above
3. Storage : A high-performance GPU with CUDA-enabled capabilities such as NVIDIA GeForce GTX or RTX
4. Storage : A minimum of 100

CHAPTER 4

SYSTEM DESIGN

SYSTEM ARCHITECTURE

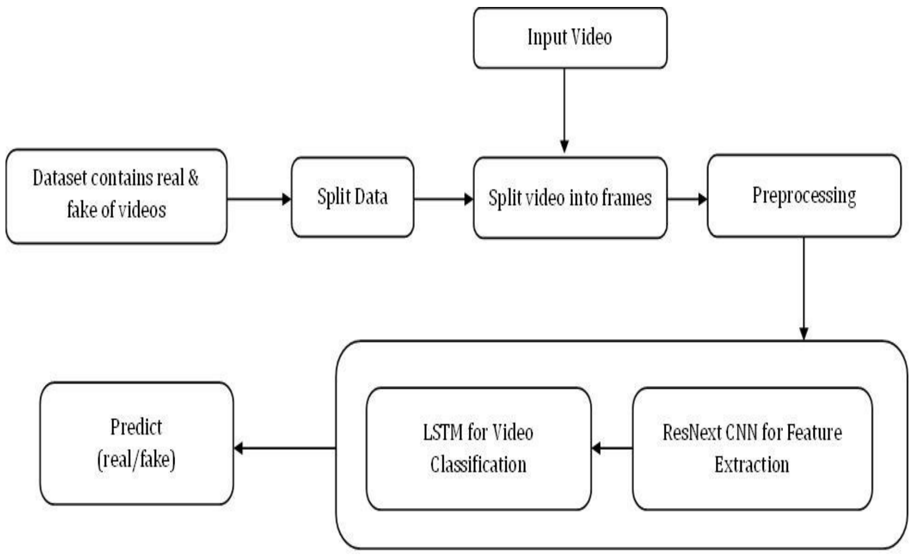

Input Video
Dataset contains real & fake of videos
Split Data
Split video into frames
Preprocessing
Predict (real/fake)
LSTM for Video Classification
ResNext CNN for Feature Extraction

Figure 1 : System Architecture

In the suggested framework, the utilization of Res-Next Convolutional Neural Network is employed for the extraction of frame-level features from videos. Subsequently, these features are inputted into an RNN (Recurrent Neural Network) based on LSTM (Long Short-Term Memory) architecture. This RNN is designed to discern the temporal relationships among individual frames, facilitating the classification of videos into the categories of either authentic or manipulated.

Gathering Data: We have combined data from several datasets, including the FaceForensics++, Celeb-DF, Deepfake Detection Challenge to create a new dataset.

Splitting of data: To ensure that our model is not biased, we trained it on an equal number of real and fraudulent videos. We took an existing dataset, preprocessed it, and produced a new processes dataset in the development phase that only contained the face-cropped videos.

Pre-processing: Dataset preprocessing comprises the breaking the video into frames. Identifying faces within each frame and precisely cropping the frame to focus solely on the detected face. To maintain uniformity in the dataset, the mean number of frames per video is calculated. Subsequently, a new dataset is created, containing face-cropped frames with this mean frame count. Frames lacking detected faces are excluded during the preprocessing stage.

Model: The architecture of the model consists of a single LSTM layer after resnext50_32x4d. The videos are divided into train and test sets by the Data Loader after they have been preprocessed and face cropped. During training and testing, the frames from the processed videos are divided into smaller batches. These batches facilitate efficient model training and evaluation.

Prediction: To make predictions, a fresh video is fed into the learned model. To incorporate the trained model's format, a fresh video is additionally preprocessed. Upon dividing the video into frames and cropping them to focus on faces, the face-cropped frames are directly fed to the trained model for identification. There is no local storage involved in this streamlined approach.

Output: The results, from the model will show if the video is real or a deepfake along with the level of certainty, in the model.

UML DIAGRAMS

A UML diagram shows the unified visual presentation of the UML (Unified Modelling Language) system intending to let developers or business owners understand, analyze, and undertake the structure and behaviors of their system. So far, the UML diagram has become one of the most common business process modelling tools, which is also highly significant to the development of object-oriented software. UML diagrams have many benefits for both software developers and businesspeople, and the most key advantages are:

Problem-Solving - Enterprises can improve their product quality and reduce cost especially for complex systems in large scale. Some other real-life problems including physical distribution or security can be solved.

Improve Productivity - By using the UML diagram, everyone in the team is on the same page and lots of time is saved down the line.

Easy to Understand - Since different roles are interested in different aspects of the system, the UML diagram offers non-professional developers, for example, stakeholders, designers, or business researchers, a clear and expressive presentation of requirements, functions and processes of their system.

USE CASE DIAGRAM:

A use case diagram is the primary form of system/software requirements for a new software program underdeveloped. Use cases specify the expected behavior, and not the exact method of making it happen. Use cases once specified can be denoted both textual and visual representation. A key concept of use case modelling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visiblesystem behavior.

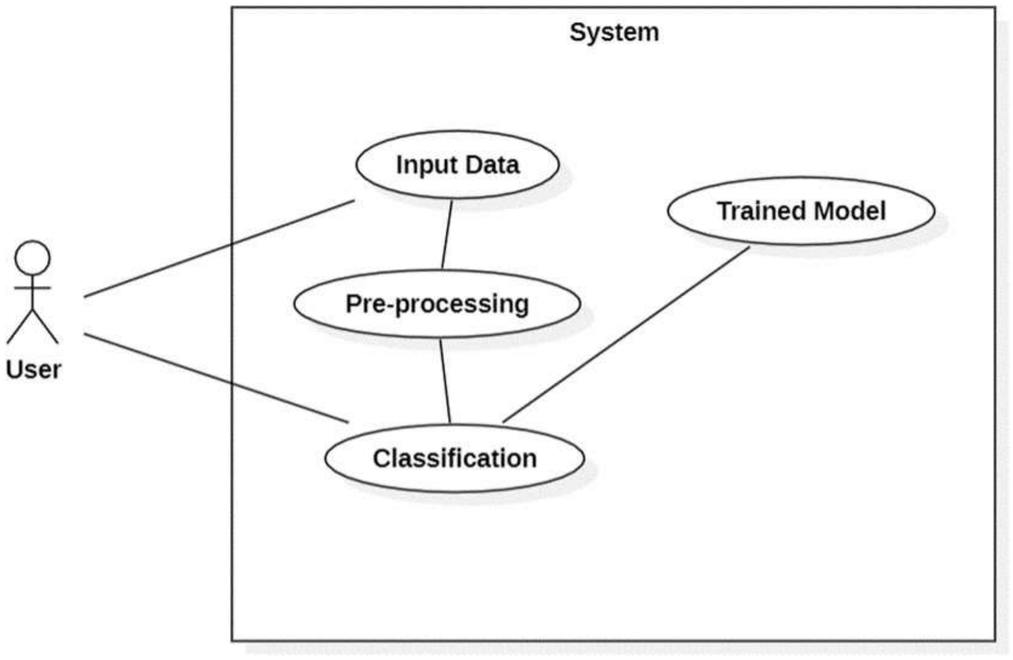


Figure 2: Usecase Diagram

CLASS DIAGRAM:

The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction. UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application, however class diagram is a bit different. It is the most popular UML diagram in the coder community.

The purpose of the class diagram can be summarized as –

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.

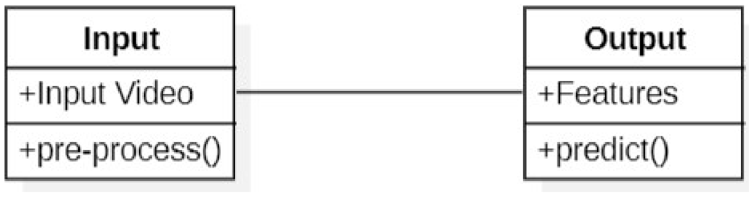


Figure 3: Class Diagram

SEQUENCE DIAGRAM:

To understand what a sequence diagram is, it's important to know the role of the Unified Modelling Language, better known as UML. UML is a modelling toolkit that guides the creation and notation of many types of diagrams, including behaviour diagrams, interaction diagrams, and structure diagrams.

A sequence diagram is a type of interaction diagram because it describes how—and in what order— a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process.

Sequence diagrams are sometimes known as event diagrams or event scenarios.

- To model high-level interaction among active objects within a system.
- To model interaction among objects inside a collaboration realizing a use case.
- It either model's generic interactions or some certain instances of interaction.

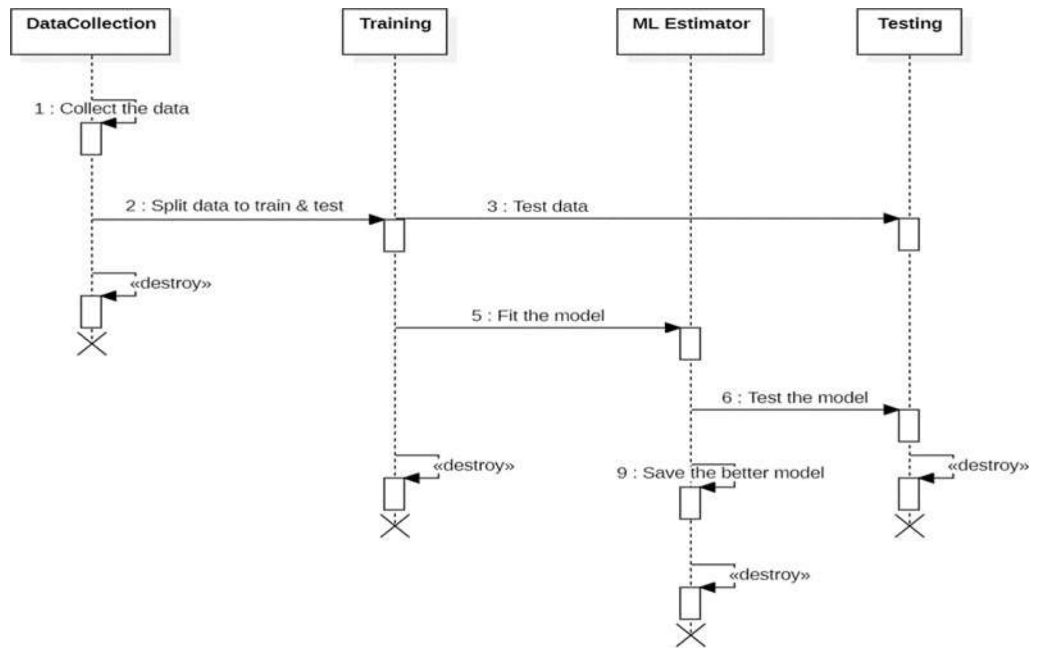


Figure 4: Sequence Diagram

ACTIVITY DIAGRAM:

A UML activity diagram helps to visualize a certain use case at a more detailed level. It is a behavioural diagram that illustrates the flow of activities through a system. UML activity diagrams can also be used to depict a flow of events in a business process. They can be used to examine business processes in order to identify its flow and requirements.

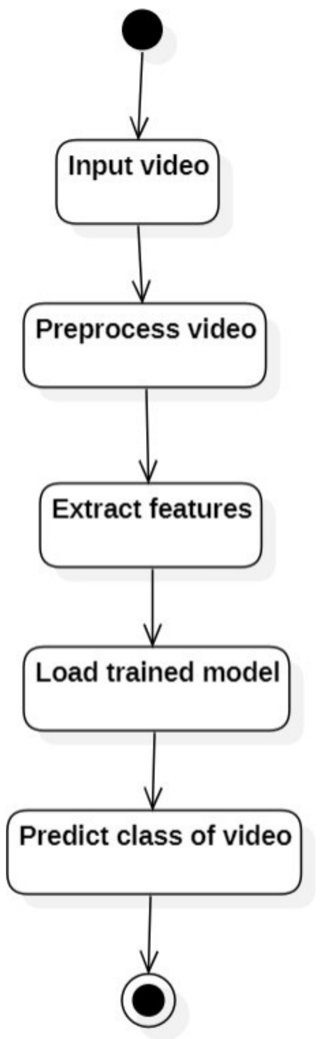


Figure 5 : Activity Diagram

CHAPTER 5
TECHNOLOGIES

PYTHON

Python is a high level, interpreted and general-purpose dynamic programming language that focuses on code readability. It has fewer steps when compared to Java and C. It was founded in 1991 by developer Guido Van Rossum. It is used in many organizations as it supports multiple programming paradigms. It also performs automatic memory management.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where asother languages use punctuation, and it has fewer syntactical constructions than other languages.

Python is a must for students and working professionals to become a great Software Engineer especially when they are working in Web Development Domain.

Advantages:

1. Presence of third-party modules
2. Extensive support libraries (NumPy for numerical calculations, Pandas for dataanalytics etc)
3. Open source and community development
4. Easy to learn
5. User-friendly data structures
6. High-level language
7. Dynamically typed language (No need to mention data type based on valueassigned, it takes data type)
8. Object-oriented language
9. Portable and Interactive
10. Portable across Operating systems

PYTHON LIBRARIES AND FRAMEWORKS

1. PyTorch
2. Django
3. NumPy
4. OpenCV or cv2
5. Face Recognition
6. Pandas
7. Torch

8. Torchvision
9. Matplotlib

PyTorch: PyTorch is an open-source deep learning framework developed primarily by Facebook's AI Research lab (FAIR). It provides a flexible and dynamic computational graph, allowing for intuitive model development and experimentation. With its Pythonic interface, PyTorch simplifies the process of building complex neural networks. Its automatic differentiation feature enables efficient gradient computation, crucial for training deep learning models. PyTorch supports both CPU and GPU computation, leveraging CUDA for accelerated training on NVIDIA GPUs. Its extensive ecosystem includes torchvision for computer vision tasks and torchtext for natural language processing, making it a popular choice for researchers and practitioners alike.

Django: Django is a high-level Python web framework known for its simplicity and versatility in building web applications. It follows the "batteries-included" philosophy, providing a wide range of features out-of-the-box, such as authentication, URL routing, and database management. With Django's built-in ORM (Object-Relational Mapper), developers can interact with databases using Python code rather than SQL queries directly. Its robust security features help developers build secure web applications effortlessly. Django's scalability and extensive documentation make it a popular choice for web development projects of any size.

NumPy: NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

OpenCV: OpenCV (Open-Source Computer Vision Library) is a powerful open-source computer vision and machine learning software library written in C++ and optimized for real-time applications. It provides a wide range of functionalities for image and video analysis, including object detection, tracking, and manipulation. OpenCV's extensive collection of algorithms makes it a popular choice for academic research, commercial applications, and hobbyist projects.

Face Recognition: Python's face recognition library is a popular tool for detecting and recognizing faces in images and videos. Built on top of dlib, it provides pre-trained models for face detection, recognition, and clustering. With a simple interface and robust performance, it's widely used for various applications, including security systems, biometrics, and augmented reality.

Pandas: Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.

torch: Python's Torch library, based on Lua's Torch framework, offers powerful capabilities for deep learning research and development. It's known for its flexibility, enabling easy experimentation with complex neural network architectures while providing efficient GPU acceleration for faster training.

torchvision: Torchvision is a Python package that provides datasets, models, and transforms for computer vision tasks within PyTorch. It simplifies the process of working with image data by offering standard datasets, pre-trained models, and image transformations for training neural networks.

matplotlib: Matplotlib is a widely-used Python library for creating static, interactive, and publication-quality visualizations with just a few lines of code. Its versatile functionality and easy integration make it a cornerstone tool for data visualization in various fields, including scientific research, data analysis, and machine learning.

CHAPTER – 6
IMPLEMENTATION

IMPLEMENTATION STEPS

DATASET DETAILS

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 col lected 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 consid ered 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 DFDCdataset 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 Celeb DF [3] dataset. Which makesourtotal dataset consisting 3000 Real, 3000 fake videos and 6000 videos in total.

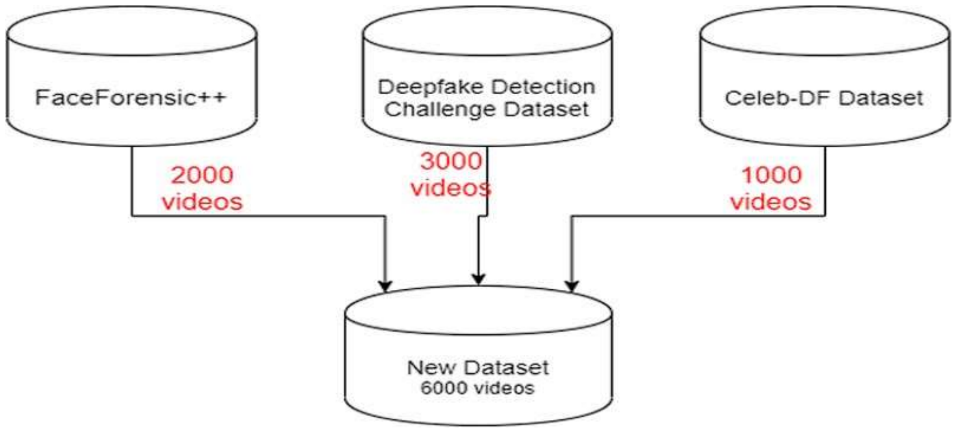


Figure 6: Dataset preparation

PREPROCESSING DETAILS

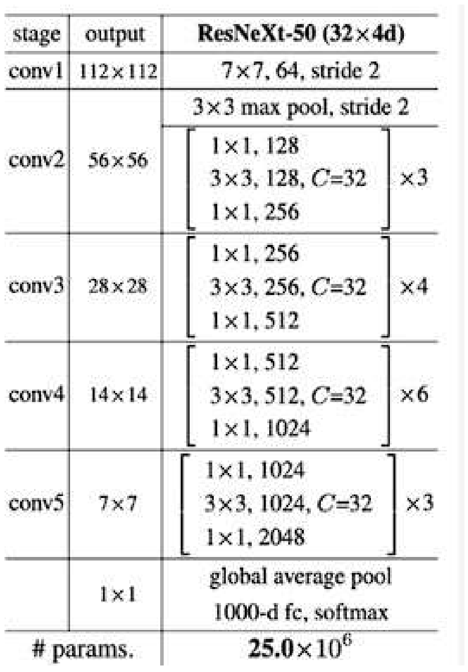
- Using glob, we imported all the videos in the directory in a python list.
- cv2.VideoCapture is used to read the videos and get the mean number of frames in each video.
- To maintain uniformity, based on mean a value 150 is selected as idea value for creating the new dataset.
- The video is split into frames and the frames are cropped on face location.

- The face cropped frames are again written to new video using VideoWriter.
- The newvideo is written at 30 frames per second and with the resolution of 112x 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.

MODEL DETAILS

The model consists of following layers:

ResNext CNN: The pre-trained model of Residual Convolution Neural Network is used. The model's 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.		**25.0**×10^6

Figure 7 : ResNext Architecture

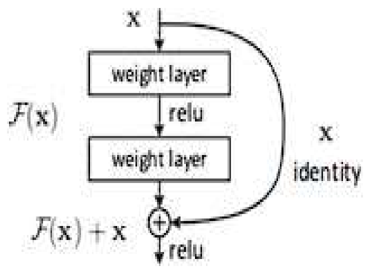


Figure 8 : ResNext working

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.

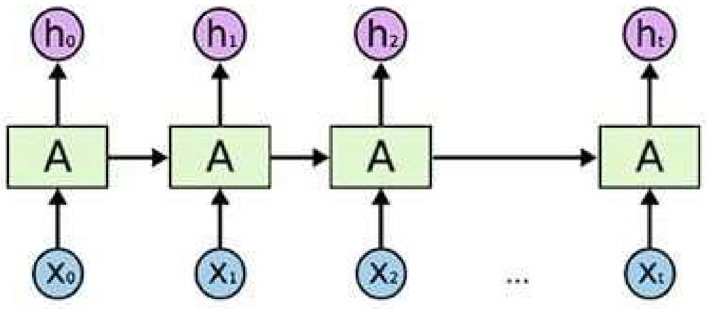


Figure 9: Overview of LSTM Architecture

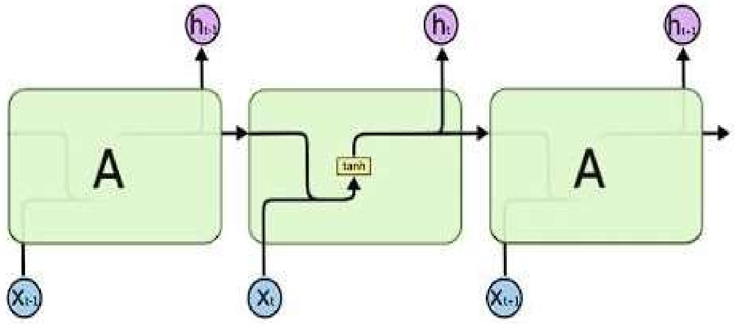


Figure 10: Internal LSTM Architecture

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.

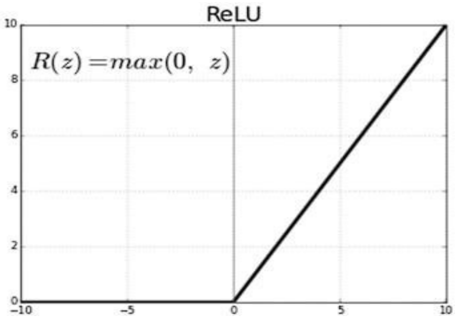


Figure 11: Relu Activation function

Dropout Layer: Dropout layer with the value of 0.4 is used to avoid over f itting 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.

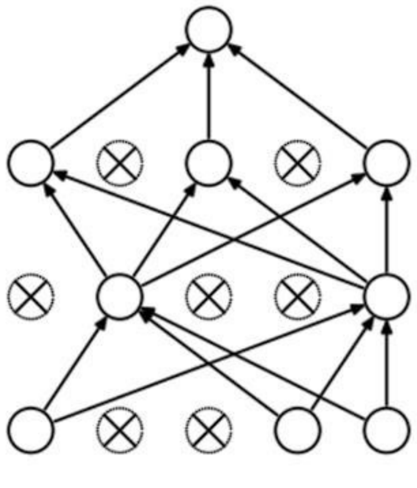


Figure 12: Dropout layer overview

Adaptive Average Pooling Layer: It is used to reduce variance, reduce com putation complexity and extract low level features from neighbourhood.2 di mensional 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 are 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-5 (0.00001), weight decay of 1e-3 (0.001) using the Adam optimizer.

Adam optimizer [21]: 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. Squash ing 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 Soft max layer provide us the confidence(probability) of prediction.

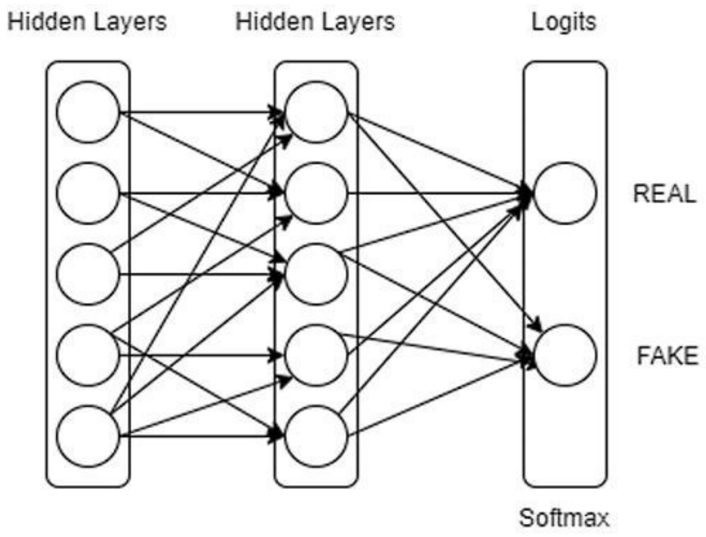


Figure 13: Softmax Layer

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 canbe 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

CODE MODULE:

```
from django.shortcuts import render, redirect
import torch
import torchvision
from torchvision import transforms, models
from torch.utils.data import DataLoader
from torch.utils.data.dataset import Dataset
import os
import numpy as np
import cv2
import matplotlib.pyplot as plt
import face_recognition
from torch.autograd import Variable
import time
import sys
from torch import nn
import json
import glob
import copy
import shutil
```

```
from PIL import Image as pImage
import time
from django.conf import settings
from .forms import VideoUploadForm
import datetime
from random import choice

from django.http import JsonResponse
from django.views.decorators.csrf import csrf_exempt

index_template_name = 'index.html'
predict_template_name = 'predict.html'

im_size = 112
mean = [0.485, 0.456, 0.406]
std = [0.229, 0.224, 0.225]
sm = nn.Softmax()
inv_normalize = transforms.Normalize(
    mean=-1*np.divide(mean, std), std=np.divide([1, 1, 1], std))

train_transforms = transforms.Compose([
    transforms.ToPILImage(),
    transforms.Resize((im_size, im_size)),
    transforms.ToTensor(),
    transforms.Normalize(mean, std)])



class Model(nn.Module):
    def __init__(self, num_classes, latent_dim=2048, lstm_layers=1, idden_dim=2048,
bidirectional=False):
        super(Model, self).__init__()
        model = models.resnext50_32x4d(pretrained=True)
        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.Dropout(0.4)
        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(x_lstm[:, -1, :]))



class validation_dataset(Dataset):
    def __init__(self, video_names, sequence_length=60, transform=None):
        self.video_names = video_names
        self.transform = transform
        self.count = sequence_length

    def __len__(self):
        return len(self.video_names)

    def __getitem__(self, idx):
        video_path = self.video_names[idx]
        frames = []
        a = int(100/self.count)
        first_frame = np.random.randint(0, a)
        for i, frame in enumerate(self.frame_extract(video_path)):
            # if(i % a == first_frame):
            faces = face_recognition.face_locations(frame)
            try:
                top, right, bottom, left = faces[0]
                frame = frame[top:bottom, left:right, :]
            except:
                pass
            frames.append(self.transform(frame))
            if (len(frames) == self.count):
                break
        """
        for i,frame in enumerate(self.frame_extract(video_path)):
            if(i % a == first_frame):
                frames.append(self.transform(frame))
        """
        # if(len(frames)<self.count):
        #   for i in range(self.count-len(frames)):
        #         frames.append(self.transform(frame))
        # print("no of frames", self.count)
        frames = torch.stack(frames)
        frames = frames[:self.count]
        return frames.unsqueeze(0)

    def frame_extract(self, path):
        vidObj = cv2.VideoCapture(path)
        success = 1
        while success:
            success, image = vidObj.read()
            if success:
                yield image

def im_convert(tensor, video_file_name):
    """ Display a tensor as an image. """
    image = tensor.to("cpu").clone().detach()
    image = image.squeeze()
    image = inv_normalize(image)
    image = image.numpy()
    image = image.transpose(1, 2, 0)
    image = image.clip(0, 1)
    # This image is not used
    # cv2.imwrite(os.path.join(settings.PROJECT_DIR, 'uploaded_images',
video_file_name+'_convert_2.png'),image*255)
    return image



def im_plot(tensor):
    image = tensor.cpu().numpy().transpose(1, 2, 0)
    b, g, r = cv2.split(image)
    image = cv2.merge((r, g, b))
    image = image*[0.22803, 0.22145, 0.216989] + [0.43216, 0.394666, 0.37645]
    image = image*255.0
    plt.imshow(image.astype(int))
    plt.show()


def predict(model, img, path='./', video_file_name=""):
    fmap, logits = model(img.to('cpu'))
    img = im_convert(img[:, -1, :, :, :], video_file_name)
    params = list(model.parameters())
    weight_softmax = model.linear1.weight.detach().cpu().numpy()
    logits = sm(logits)
    _, prediction = torch.max(logits, 1)
    confidence = logits[:, int(prediction.item())].item()*100
    print('confidence of prediction:',
          logits[:, int(prediction.item())].item()*100)
    return [int(prediction.item()), confidence]

def plot_heat_map(i, model, img, path='./', video_file_name=''):
    fmap, logits = model(img.to('cpu'))
    params = list(model.parameters())
    weight_softmax = model.linear1.weight.detach().cpu().numpy()
    logits = sm(logits)
    _, prediction = torch.max(logits, 1)
    idx = np.argmax(logits.detach().cpu().numpy())
    bz, nc, h, w = fmap.shape
    # out = np.dot(fmap[-1].detach().cpu().numpy().reshape((nc,
h*w)).T,weight_softmax[idx,:].T)
    out = np.dot(fmap[i].detach().cpu().numpy().reshape(
        (nc, h*w)).T, weight_softmax[idx, :].T)
    predict = out.reshape(h, w)
    predict = predict - np.min(predict)
    predict_img = predict / np.max(predict)
    predict_img = np.uint8(255*predict_img)
    out = cv2.resize(predict_img, (im_size, im_size))
    heatmap = cv2.applyColorMap(out, cv2.COLORMAP_JET)
    img = im_convert(img[:, -1, :, :, :], video_file_name)
    result = heatmap * 0.5 + img*0.8*255
    # Saving heatmap - Start
    heatmap_name = video_file_name+"_heatmap_"+str(i)+".png"
    image_name = os.path.join(settings.PROJECT_DIR,
                              'uploaded_images', heatmap_name)
    cv2.imwrite(image_name, result)
    # Saving heatmap - End
    result1 = heatmap * 0.5/255 + img*0.8
    r, g, b = cv2.split(result1)
    result1 = cv2.merge((r, g, b))
    return image_name

# Model Selection
def get_accurate_model(sequence_length):
    model_name = []
    sequence_model = []
    final_model = ""
    list_models = glob.glob(os.path.join(
        settings.PROJECT_DIR, "models", "*.pt"))

    for i in list_models:
        model_name.append(i.split("\\")[-1])

    for i in model_name:
        try:
            seq = i.split("_")[3]
            if (int(seq) == sequence_length):
                sequence_model.append(i)
        except:
            pass

    if len(sequence_model) > 1:
        accuracy = []
        for i in sequence_model:
            acc = i.split("_")[1]
            accuracy.append(acc)
        max_index = accuracy.index(max(accuracy))
        final_model = sequence_model[max_index]
    else:
        final_model = sequence_model[0]
    return final_model



ALLOWED_VIDEO_EXTENSIONS = set(
    ['mp4', 'gif', 'webm', 'avi', '3gp', 'wmv', 'flv', 'mkv'])

def allowed_video_file(filename):
    # print("filename" ,filename.rsplit('.',1)[1].lower())
    if (filename.rsplit('.', 1)[1].lower() in ALLOWED_VIDEO_EXTENSIONS):
        return True
    else:
        return False

def index(request):
    if request.method == 'GET':
        video_upload_form = VideoUploadForm()
        if 'file_name' in request.session:
            del request.session['file_name']
        if 'preprocessed_images' in request.session:
            del request.session['preprocessed_images']
        if 'faces_cropped_images' in request.session:
            del request.session['faces_cropped_images']
        return render(request, index_template_name, {"form": video_upload_form})
    else:
        video_upload_form = VideoUploadForm(request.POST, request.FILES)
        if video_upload_form.is_valid():
            video_file = video_upload_form.cleaned_data['upload_video_file']
            video_file_ext = video_file.name.split('.')[-1]
            sequence_length = video_upload_form.cleaned_data['sequence_length']
            video_content_type = video_file.content_type.split('/')[0]
            if video_content_type in settings.CONTENT_TYPES:
```

```
            if video_file.size > int(settings.MAX_UPLOAD_SIZE):
                video_upload_form.add_error(
                    "upload_video_file", "Maximum file size 100 MB")
                return render(request, index_template_name, {"form": video_upload_form})

        if sequence_length <= 0:
            video_upload_form.add_error(
                "sequence_length", "Sequence Length must be greater than 0")
            return render(request, index_template_name, {"form": video_upload_form})

        if allowed_video_file(video_file.name) == False:
            video_upload_form.add_error(
                "upload_video_file", "Only video files are allowed ")
            return render(request, index_template_name, {"form": video_upload_form})

        saved_video_file = 'uploaded_file_' + \
            str(int(time.time()))+"."+video_file_ext
        if settings.DEBUG:
            with open(os.path.join(settings.PROJECT_DIR, 'uploaded_videos',
saved_video_file), 'wb') as vFile:
                shutil.copyfileobj(video_file, vFile)
            request.session['file_name'] = os.path.join(
                settings.PROJECT_DIR, 'uploaded_videos', saved_video_file)
        else:
            with open(os.path.join(settings.PROJECT_DIR, 'uploaded_videos', 'app',
'uploaded_videos', saved_video_file), 'wb') as vFile:
                shutil.copyfileobj(video_file, vFile)
            request.session['file_name'] = os.path.join(
                settings.PROJECT_DIR, 'uploaded_videos', 'app', 'uploaded_videos',
saved_video_file)
        request.session['sequence_length'] = sequence_length
        return redirect('ml_app:predict')
    else:
        return render(request, index_template_name, {"form": video_upload_form})

def result(request):
    preprocessed_images = request.session["preprocessed_images"]
    heatmap_images = request.session["heatmap_images"]
    faces_cropped_images = request.session["faces_cropped_images"]
    models_location = request.session["models_location"]
    output = request.session["output"]
    confidence = request.session["confidence"]
    accuracy = request.session["accuracy"]
    decimal = request.session["decimal"]
    video_file_name = request.session["original_video"]
    return render(request, predict_template_name, {'preprocessed_images':
preprocessed_images, 'heatmap_images': heatmap_images, "faces_cropped_images":
faces_cropped_images, "original_video": video_file_name, "models_location":
models_location, "output": output, "confidence": confidence, "accuracy": accuracy,
"decimal": decimal})

def predict_page(request):
    if request.method == "GET":
        if 'file_name' not in request.session:
            return redirect("ml_app:home")
        if 'file_name' in request.session:
            video_file = request.session['file_name']
        if 'sequence_length' in request.session:
            sequence_length = request.session['sequence_length']
        path_to_videos = [video_file]
        video_file_name = video_file.split('\\')[-1]
        if settings.DEBUG == False:
            production_video_name = video_file_name.split('/')[3:]
            production_video_name = '/'.join([str(elem)
                                              for elem in production_video_name])
            print("Production file name", production_video_name)
        video_file_name_only = video_file_name.split('.')[0]
        video_dataset = validation_dataset(
            path_to_videos, sequence_length=sequence_length, transform=train_transforms)
        model = Model(2).cpu()

        model_name = os.path.join(
            settings.PROJECT_DIR, 'models', get_accurate_model(sequence_length))
        models_location = os.path.join(settings.PROJECT_DIR, 'models')
```

```
path_to_model = os.path.join(settings.PROJECT_DIR, model_name)
model.load_state_dict(torch.load(
    path_to_model, map_location=torch.device('cpu')))
model.eval()
start_time = time.time()
# Start: Displaying preprocessing images
print("<=== | Started Videos Splitting | ===>")
preprocessed_images = []
faces_cropped_images = []
cap = cv2.VideoCapture(video_file)

frames = []
while (cap.isOpened()):
    ret, frame = cap.read()
    if ret == True:
        frames.append(frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break
cap.release()

for i in range(1, sequence_length+1):
    frame = frames[i]
    image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    img = pImage.fromarray(image, 'RGB')
    image_name = video_file_name_only+"_preprocessed_"+str(i)+'.png'
    if settings.DEBUG:
        image_path = os.path.join(
            settings.PROJECT_DIR, 'uploaded_images', image_name)
    else:
        print("image_name", image_name)
        image_path = "/home/app/staticfiles" + image_name
    img.save(image_path)
    preprocessed_images.append(image_name)
print("<=== | Videos Splitting Done | ===>")
print("--- %s seconds ---" % (time.time() - start_time))
```

```
    # End: Displaying preprocessing images
    # Start: Displaying Faces Cropped Images
    print("<=== | Started Face Cropping Each Frame | ===>")
    padding = 40
    faces_found = 0
    for i in range(1, sequence_length+1):
        frame = frames[i]
        #fig, ax = plt.subplots(1,1, figsize=(5, 5))
        face_locations = face_recognition.face_locations(frame)
        if len(face_locations) == 0:
            continue
        top, right, bottom, left = face_locations[0]
        frame_face = frame[top-padding:bottom +
                          padding, left-padding:right+padding]
        image = cv2.cvtColor(frame_face, cv2.COLOR_BGR2RGB)

        img = pImage.fromarray(image, 'RGB')
        image_name = video_file_name_only+"_cropped_faces_"+str(i)+'.png'
        if settings.DEBUG:
            image_path = os.path.join(
                settings.PROJECT_DIR, 'uploaded_images',
video_file_name_only+"_cropped_faces_"+str(i)+'.png')
        else:
            image_path = "/home/app/staticfiles" + image_name
        img.save(image_path)
        faces_found = faces_found + 1
        faces_cropped_images.append(image_name)
    print("<=== | Face Cropping Each Frame Done | ===>")
    print("--- %s seconds ---" % (time.time() - start_time))

    # No face is detected
    if faces_found == 0:
        return render(request, predict_template_name, {"no_faces": True})

    # End: Displaying Faces Cropped Images
    try:
        heatmap_images = []
        for i in range(0, len(path_to_videos)):
            output = ""
            print("<=== | Started Predicition | ===>")
            prediction = predict(
                model, video_dataset[i], './', video_file_name_only)
            confidence = round(prediction[1], 1)
            print("<=== |  Predicition Done | ===>")
            # print("<=== | Heat map creation started | ===>")
            # for j in range(0, sequence_length):
            #     heatmap_images.append(plot_heat_map(j, model, video_dataset[i], './', 
video_file_name_only))
            if prediction[0] == 1:
                output = "REAL"
            else:
                output = "FAKE"
            print("Prediction : ",
                  prediction[0], "==", output, "Confidence : ", confidence)
            print("--- %s seconds ---" % (time.time() - start_time))
        request.session["preprocessed_images"] = preprocessed_images
        request.session["heatmap_images"] = heatmap_images
        request.session["faces_cropped_images"] = faces_cropped_images
        request.session["models_location"] = models_location
        request.session["output"] = output
        request.session["confidence"] = confidence
        request.session["accuracy"] = model_name.split("\\")[-1][6:8]
        request.session["decimal"] = model_name.split("\\")[-1][13:15]
        if settings.DEBUG:
            request.session["original_video"] = video_file_name
            # return render(request, predict_template_name, {'preprocessed_images': 
preprocessed_images, 'heatmap_images': heatmap_images, "faces_cropped_images": 
faces_cropped_images, "original_video": video_file_name, "models_location": 
models_location, "output": output, "confidence": confidence, "accuracy": 
model_name.split("\\")[-1][6:8], "decimal": model_name.split("\\")[-1][13:15]})
        else:
            request.session["original_video"] = production_video_name
            # return render(request, predict_template_name, {'preprocessed_images': 
preprocessed_images, 'heatmap_images': heatmap_images, "faces_cropped_images":
```

```
faces_cropped_images, "original_video": production_video_name, "models_location":
models_location, "output": output, "confidence": confidence, "accuracy":
model_name.split("\\")[-1][6:8], "decimal": model_name.split("\\")[-1][13:15]})
            return redirect("ml_app:result")
        except Exception as e:
            print("Execption :", e.args)
            return render(request, 'cuda_full.html')

def about(request):
    return render(request, "about.html")

def handler404(request, exception):
    return render(request, '404.html', status=404)

@csrf_exempt
def video_upload_api(request):
    if request.method == 'POST':
        file_name =
f"uploaded_video_{datetime.datetime.now().strftime('%Y%m%d_%H%M%S')}.mp4"
        with open(f"uploaded_videos/{file_name}", "wb") as f:
            f.write(request.body)
        request.session['file_name'] = os.path.join(
            settings.PROJECT_DIR, 'uploaded_videos', file_name)
        request.session["sequence_length"] = 20
        result = run_deep_fake_detection(request)
        return JsonResponse({'result': result})
    else:
        return JsonResponse({'error': 'Invalid request'}, status=400)

def run_deep_fake_detection(request):
    if 'file_name' in request.session:
        video_file = request.session['file_name']
    if 'sequence_length' in request.session:
        sequence_length = request.session['sequence_length']
    else:
        sequence_length = choice([20, 40, 60, 80, 100])
    path_to_videos = [video_file]
    video_file_name = video_file.split('\\')[-1]
    if settings.DEBUG == False:
        production_video_name = video_file_name.split('/')[3:]
        production_video_name = '/'.join([str(elem)
                                          for elem in production_video_name])
    video_file_name_only = video_file_name.split('.')[0]
    video_dataset = validation_dataset(
        path_to_videos, sequence_length=sequence_length, transform=train_transforms)
    model = Model(2).cpu()

    model_name = os.path.join(settings.PROJECT_DIR, 'models',
get_accurate_model(sequence_length))
    models_location = os.path.join(settings.PROJECT_DIR, 'models')
    path_to_model = os.path.join(settings.PROJECT_DIR, model_name)
    model.load_state_dict(torch.load(
        path_to_model, map_location=torch.device('cpu')))
    model.eval()
    start_time = time.time()
    # Start: Displaying preprocessing images
    preprocessed_images = []
    faces_cropped_images = []
    cap = cv2.VideoCapture(video_file)

    frames = []
    while (cap.isOpened()):
        ret, frame = cap.read()
        if ret == True:
            frames.append(frame)
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
        else:
            break
    cap.release()

    for i in range(1, sequence_length+1):
        frame = frames[i]
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        img = pImage.fromarray(image, 'RGB')
        image_name = video_file_name_only+"_preprocessed_"+str(i)+'.png'
        if settings.DEBUG:
            image_path = os.path.join(
                settings.PROJECT_DIR, 'uploaded_images', image_name)
        else:
            image_path = "/home/app/staticfiles" + image_name
        img.save(image_path)
        preprocessed_images.append(image_name)
    # End: Displaying preprocessing images
    # Start: Displaying Faces Cropped Images
    padding = 40
    faces_found = 0
    for i in range(1, sequence_length+1):
        frame = frames[i]
        #fig, ax = plt.subplots(1,1, figsize=(5, 5))
        face_locations = face_recognition.face_locations(frame)
        if len(face_locations) == 0:
            continue
        top, right, bottom, left = face_locations[0]
        frame_face = frame[top-padding:bottom +
                          padding, left-padding:right+padding]
        image = cv2.cvtColor(frame_face, cv2.COLOR_BGR2RGB)

        img = pImage.fromarray(image, 'RGB')
        image_name = video_file_name_only+"_cropped_faces_"+str(i)+'.png'
        if settings.DEBUG:
            image_path = os.path.join(
                settings.PROJECT_DIR, 'uploaded_images',
video_file_name_only+"_cropped_faces_"+str(i)+'.png')
        else:
            image_path = "/home/app/staticfiles" + image_name
        img.save(image_path)
        faces_found = faces_found + 1
        faces_cropped_images.append(image_name)
    # No face is detected
    if faces_found == 0:
        return render(request, predict_template_name, {"no_faces": True})

    # End: Displaying Faces Cropped Images
    try:
        for i in range(0, len(path_to_videos)):
            prediction = predict(
                model, video_dataset[i], './', video_file_name_only)
            confidence = round(prediction[1], 1)
        return {'output': 'REAL' if prediction[0] == 1 else 'FAKE', 'confidence': confidence,
'prediction': prediction[0], 'Number of frames used': sequence_length}
    except Exception as e:
        return {"Exception": e}

def get_data(request):
    is_splitted = request.session["is_video_splitted"] if "is_video_splitted" in request.session
else None
    data = {"is_splitted": is_splitted}
    return JsonResponse(data)
```

CHAPTER – 7
TESTING

TESTING

System testing involves user training system testing and successful running of the developed proposed system. The user tests the developed system and changes aremade according to their needs. The testing phase involves the testing of developedsystem using various kinds of data.

An elaborate testing of data is prepared and the system is tested using the test data.While testing, errors are noted and the corrections are made. The corrections are also noted for the future use. The users are trained to operate the developed system.

Testing involves operation of a system or application under controlled conditionsand evaluating the results. The controlled conditions should include both normal and abnormal conditions. Testing should intentionally attempt to make things go wrong to determine if things happen when they shouldn't or things don't happen when they should .it is oriented to 'detection'.

System testing is the stage of implementation that is aimed at ensuring that the system works accurately before live operation commences. Testing is vital to the success of the system. System testing makes logical assumption that if all the parts of the system are correct, then the goal will be successfully achieved. A series of testing are done for the proposed system before the system is ready for the user.

TESTING METHODOLOGIES

There are two major types of testing. They are

1. White Box Testing
2. Black Box Testing

WHITE BOX TESTING

White box testing (also known as clear box testing, glass box testing, and transparent box testing and structural testing) is a method of testing software that tests internal structures are working of an application, an opposed to its functionality. In white box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tests are choosing inputs to exercise paths through the code and determine outputs.

While white box testing can be applied at the unit, integration and system level of the software testing process, it is usually done at the unit level. It can test paths within a unit, path between units during integration, and between sub systems during system level tests.

This method of test design can uncover many of our problems it might not detect unimplemented parts of the specification or missing requirements.

White box test design techniques include:

- Control flow testing
- Data flow testing
- Branch testing
- Statement coverage
- Decision coverage

BLACK BOX TESTING

Black box testing is a method of software testing that examines the functionality of an (see white box testing). This method of test can be applied to virtually every level of software testing: unit, integration, system and acceptance. It typically compares most if not all higher-level testing but can also dominate unit testing as well. The following are the types of testing:

1. Unit Testing
2. Integration Testing
3. Validation Testing
4. Verification Testing
5. User Acceptance Testing

Unit Testing : Unit testing focuses verification efforts and the smallest unit of the software design, the module. This is also known as “module testing”. The modules are tested separately, this testing was carried out during programming stage itself, in this testing each module is found to be working satisfactory as regards to the expected output from the module.

Integration Testing: Data can be lost across an interface: one module can have adverse efforts on another. Integration testing is the systematic testing for construction of program structure, while at the same time conducting test to uncover errors associated with in the interface. A correction is difficult because the isolation of the cause is complicated by the cast expanse of the entire program. Thus, in the integration testing step , all the errors uncovered are corrected for the next testing step.

Validation Testing: At the conclusion of integration testing, software is completely assembled as a package, interfacing errors have been uncovered and corrected and a final series of software tests begins validation test has been conducted of the two possible conditions exists. One is the function or performance characteristics confirmed to specifications and are accepted and the other is deviation from specifications in uncovered and eficiency list is created.

Verification Testing: Verification is a fundamental concept in software design. This is the bridge between customer requirements and an implementation that specifies those requirements. This is verifiable if it can be demonstrated that the testing will result in an implementation that satisfies the customer requirements.

User Acceptance Testing : User acceptance testing of a system is the key factor of the success of the nay system. The system under study is tested for the user acceptance by constantly keeping in touch with their susceptive system users at any time of developing and making changes whenever required.

TEST CASES AND TEST RESULTS

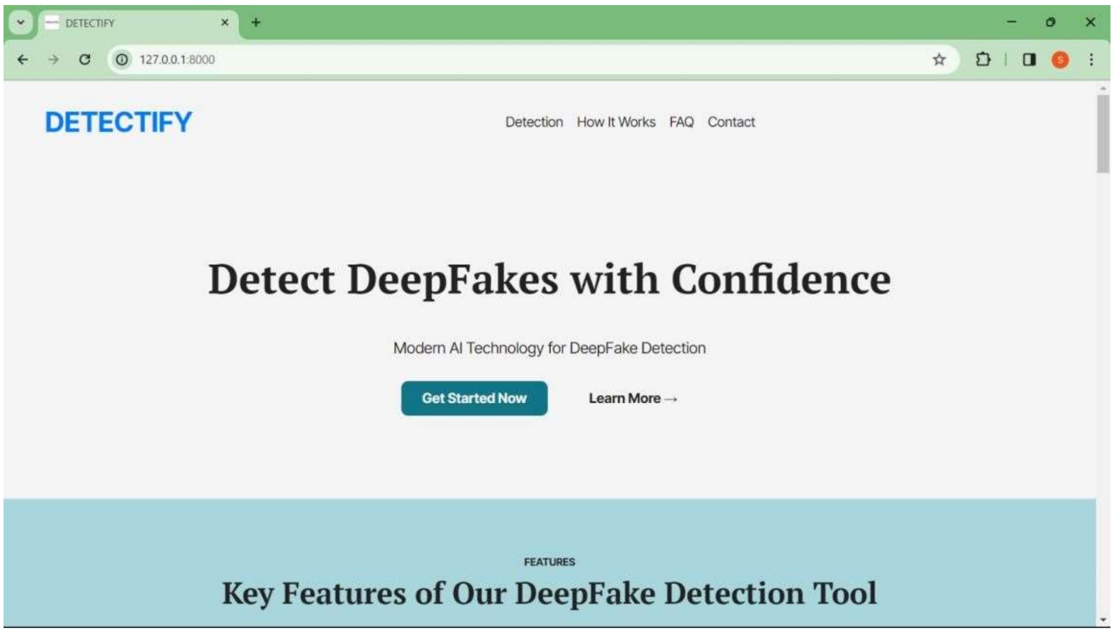
Test Cases

Case ID	Test Case Description	Expected Result	Actual Result	Status
1	Upload the video inputs of various formats (e.g. MP4, AVI, etc.)	Real/Fake	Real/Fake	Pass
2	Upload corrupted or invalid video files	Error Message: Only video files allowed	Error Message: Only video files allowed	Pass
3	Upload the files with video of different resolution and frame rates	If frames less than100 and resolution greater than 200MB an error message is raised else the output will be Real/Fake	If frames less than 100 and resolution greater than 200MB an error message is raised else the output will be Real/Fake	Pass
4	Upload video with multiple faces	Real/Fake	Fake	Pass
5	Upload video with no faces	An error message will be shown on the screen	An error message shown on the scree	Pass
6	Upload a real video	Real	Real	Pass
7	Upload a fake video	Fake	Fake	Pass

CHAPTER – 8
RESULTS AND DISCUSSION

RESULT:

Home Page:



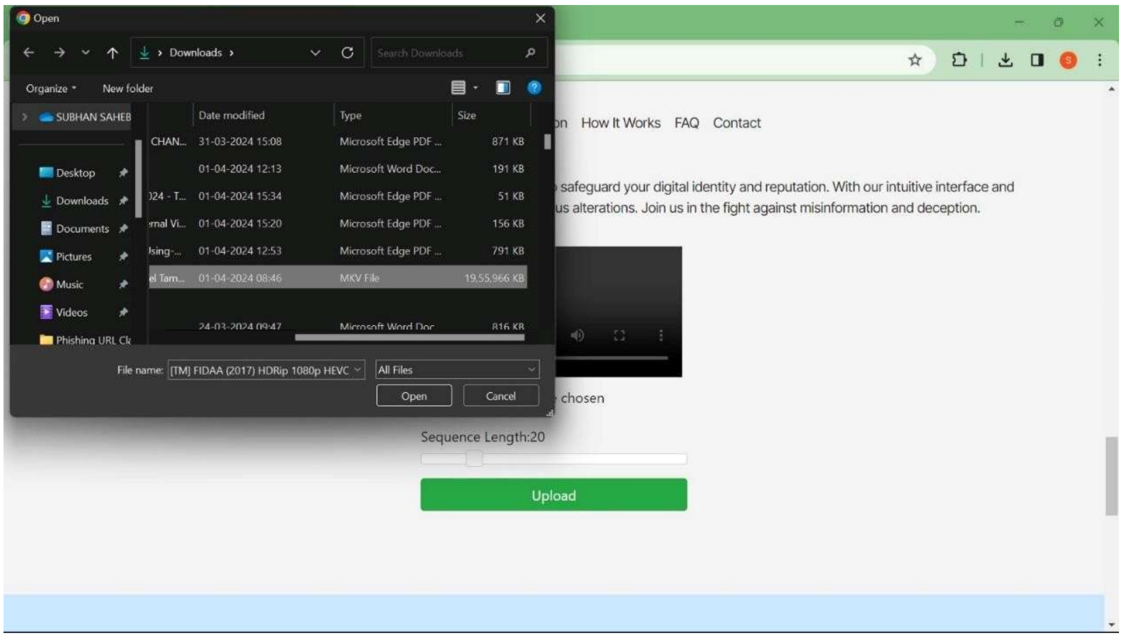
Detect DeepFakes with Confidence

Modern AI Technology for DeepFake Detection

Key Features of Our DeepFake Detection Tool

Figure 14: Application's home page

File Upload Area:



	Date modified	Type	Size
CHAN...	31-03-2024 15:08	Microsoft Edge PDF ...	871 KB
	01-04-2024 12:13	Microsoft Word Doc...	191 KB
)24 - T...	01-04-2024 15:34	Microsoft Edge PDF ...	51 KB
rnal Vi...	01-04-2024 15:20	Microsoft Edge PDF ...	156 KB
Ising-...	01-04-2024 12:53	Microsoft Edge PDF ...	791 KB
el Tam...	01-04-2024 08:46	MKV File	19,55,966 KB
	24-03-2024 09:47	Microsoft Word Doc	816 KB

safeguard your digital identity and reputation. With our intuitive interface and us alterations. Join us in the fight against misinformation and deception.

Figure 15: The page that appears after selecting get started now & uploading File

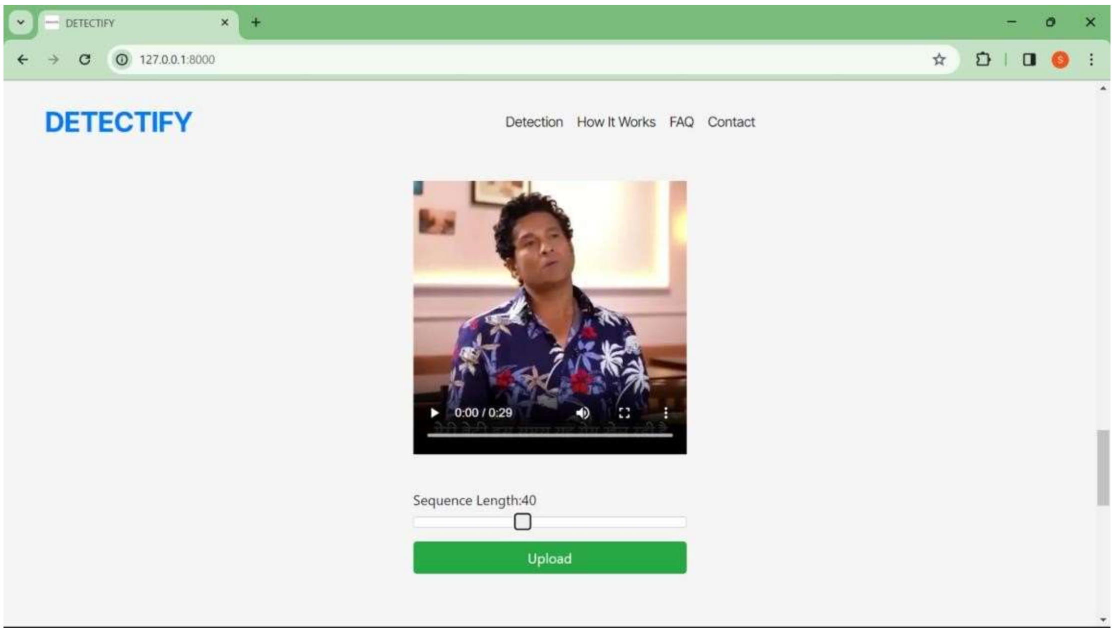


Figure 16: user uploaded video

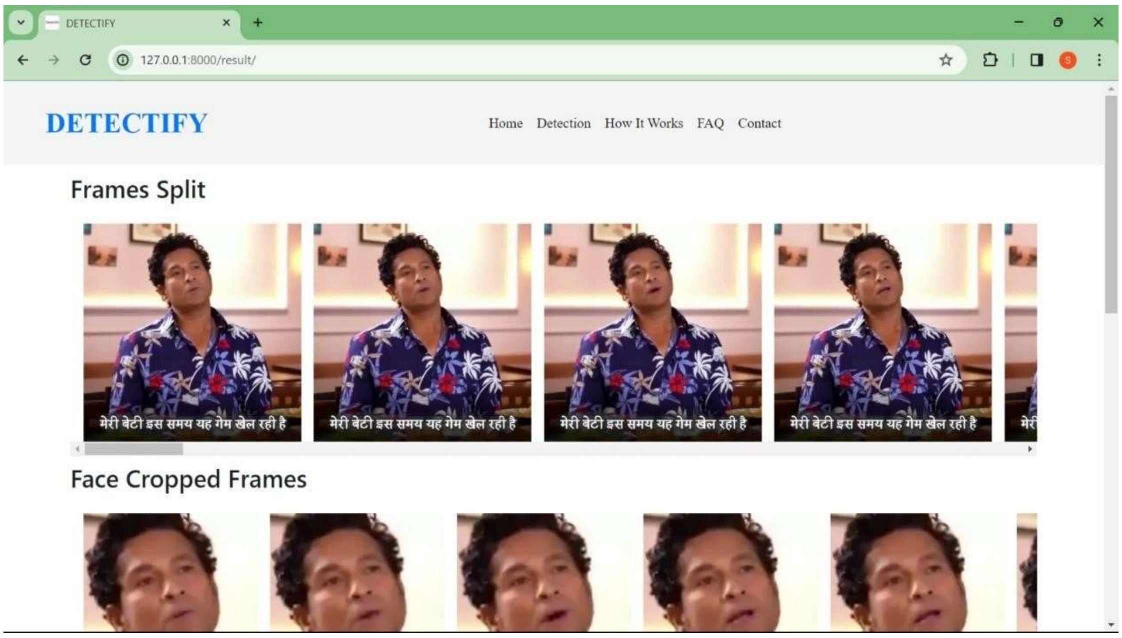
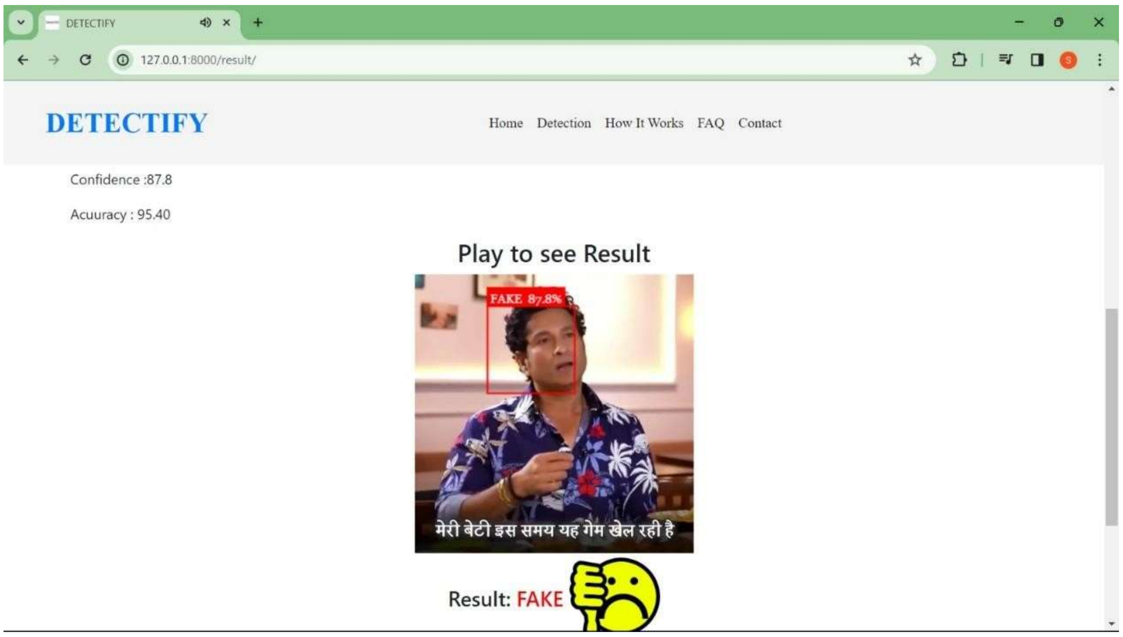


Figure 17: Video divided into frames based on the length of the sequence


DETECTIFY
127.0.0.1:8000/result/
DETECTIFY
Home Detection How It Works FAQ Contact
FAKE 87.8%
मेरी बेटी इस समय यह गेम खेल रही है

Confidence :87.8

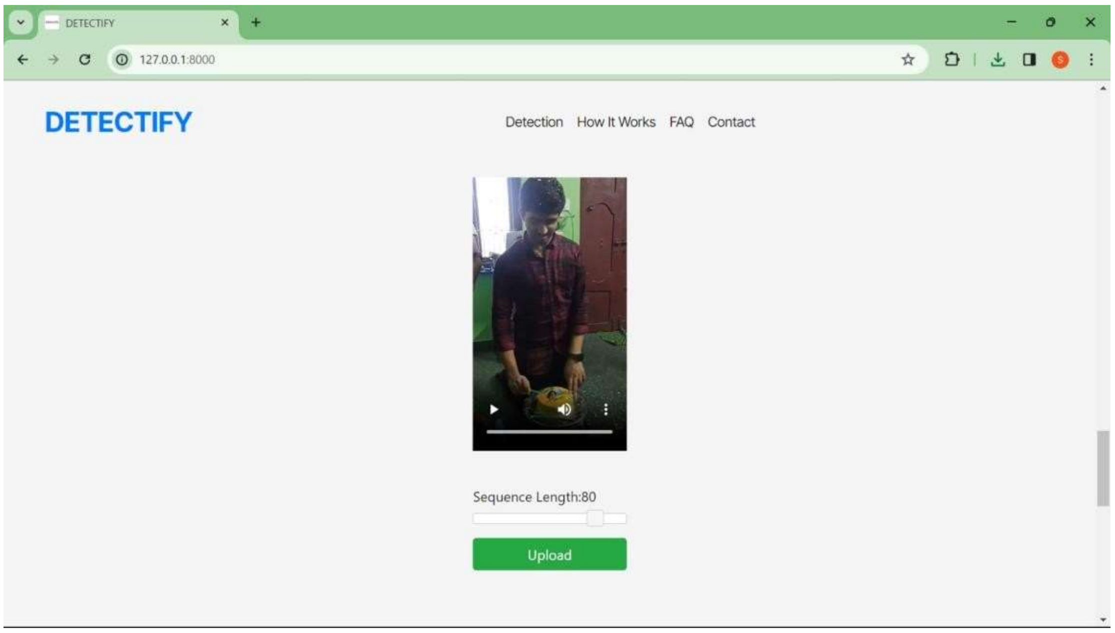
Acuuracy : 95.40

Play to see Result

Result: FAKE

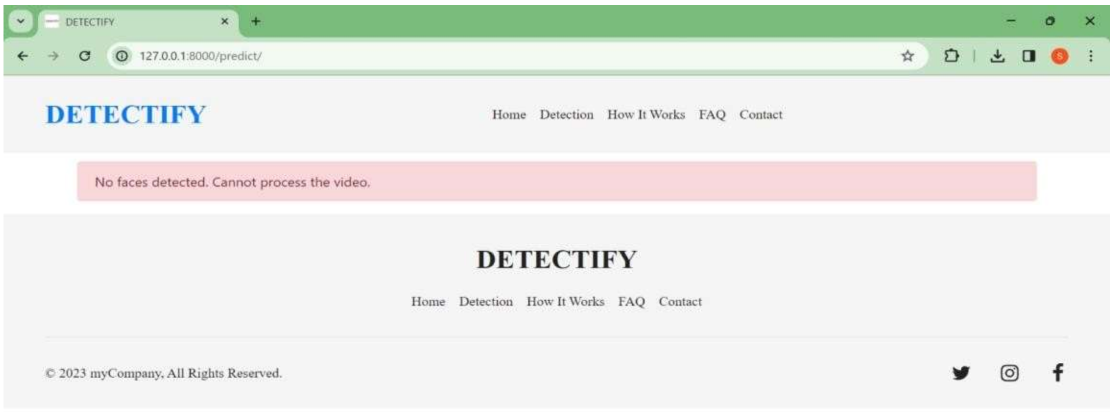
Figure 18: Final result along with a percentage to indicate whether the video is real or not.

Testcase2 : Where the face is not detected in the video



Sequence Length:80

Figure 19: uploaded video-2


DETECTIFY

127.0.0.1:8000/predict/

DETECTIFY

Home Detection How It Works FAQ Contact

No faces detected. Cannot process the video.

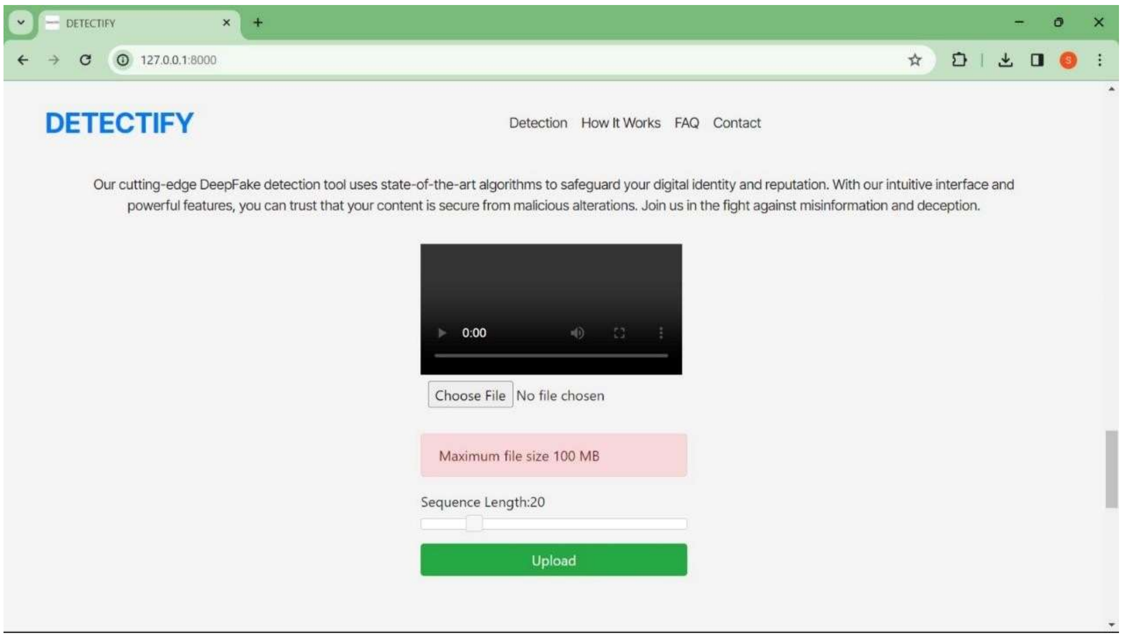
DETECTIFY

Home Detection How It Works FAQ Contact

© 2023 myCompany, All Rights Reserved.

Figure 20: output showing no faces detected

Testcase3: Where the file size is exceeded(100MB file)


DETECTIFY

127.0.0.1:8000

DETECTIFY

Detection How It Works FAQ Contact

Our cutting-edge DeepFake detection tool uses state-of-the-art algorithms to safeguard your digital identity and reputation. With our intuitive interface and powerful features, you can trust that your content is secure from malicious alterations. Join us in the fight against misinformation and deception.

Choose File No file chosen

Maximum file size 100 MB

Sequence Length:20

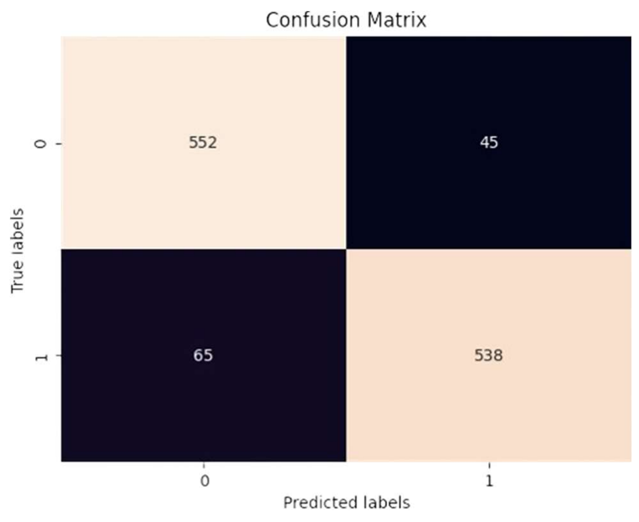
Upload

Figure 21: output showing that video won't be accepeted if size greater than 100MB

Results and Discussion:

The performance of Deep Fake Detection system can be evaluated using the mostly used machine learning or deep learning metric known as confusion matrix and accuracy score. These are helpful to evaluate and enhance performance of the deep fake detection model. Since it is a classification technique these metrics can provide, a stable measuring scale.

Confusion Matrix: A confusion matrix is a table that is often used to evaluate the performance of a classification model. It summarizes the performance of a classification algorithm by presenting a count of the number of correct and incorrect predictions made by the model on a set of data for each class. In a binary classification problem, the confusion matrix typically consists of four values: true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). True positives are the instances that were correctly predicted as positive, false positives are the instances that were incorrectly predicted as positive, true negatives are the instances that were correctly predicted as negative, and false negatives are the instances that were incorrectly predicted as negative. These values can be used to calculate various performance metrics such as accuracy, precision, recall, and F1 score, which provide insights into the overall performance of the classification model.



Accuracy Score: Accuracy measures the overall correctness of the model's predictionsacross all classes.

$$Accuracy\,Score = \frac{TP + TN}{TP + TN + FP + FN} = \frac{552 + 538}{552 + 538 + 65 + 45} \approx 0.9083$$

Precision: Precision focuses on the accuracy of positive predictions made by the model.

$$Precision = \frac{TP}{TP + FP} = \frac{552}{552 + 45} \approx 0.9245$$

Recall (Sensitivity): Recall assesses the model's ability to capture all positive instances.

$$Recall = \frac{TP}{TP + FN} = \frac{552}{552 + 65} \approx 0.8947$$

Specificity: Specificity measures the model's ability to avoid false alarms or false positives.

$$Specificity = \frac{TN}{TN + FP} = \frac{538}{538 + 45} \approx 0.9224$$

F1 Score: F1 Score is the harmonic mean of precision and recall, providing a balanced assessment of the model's performance.

$$F1\ Score = 2 * \frac{Precision*Recall}{Precision+Recall} = 2 * \frac{0.9245*0.8947}{0.9245+0.8947} \approx 0.9093$$

According to all these values in here, our model is working good, efficient and effective. Like anything has no boundaries, it also can be improved much more efficiently and much more effectively by feeding and training across real-time videos from public social media accounts and getting surveys on this model will be helpful to get real-time results. But it needs much more computing units and much time to and also human interaction to cross check the results. There are always somethings will be left out for future enhancements and some flaws can be pointed in existing methodologies, all these will be discussed in the following chapter.

CHAPTER – 9
CONCLUSION AND FUTURE SCOPE

9.1 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 modelto extract the frame level features and LSTM for temporal sequence process ing 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.

9.2 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.

CHAPTER – 10
REFERENCES

REFERENCES:

[1]. Fang Sun, 1 Niuniu Zhang, 1 Pan Xu,1 and Zengren Song2 “Deepfake Detection Method Based on Cross-Domain Fusion”

[2]. Gandhi, Apurva, and Shomik Jain Adversarial Perturbations Fool Deepfake Detectors

[3]. Lingzhi Li, Jianmin Bao, Ting Zhang, Hao Yang, Dong Chen, Fang Wen, BainingGuo; “Face X-Ray for More General Face Forgery Detection”

[4]. Xin Yang; Yuezun Li; Siwei Lyu “Exposing Deep Fakes Using Inconsistent Head Poses”

[5]. David Güera; Edward J. Delp “Deepfake Video Detection Using Recurrent Neural Networks”

[6] https://ai.meta.com/blog/deepfake-detection-challenge/

[7] M. M. El-Gayar, Mohamed Abouhawwash,,S. S. Askar & Sara Sweidan “A novel approach for detecting deep fake videos using graph neural network”

[8] F. Song, X. Tan, X. Liu, and S. Chen, “Eyes closeness detection from still images with multi-scale histograms of principal oriented gradients,” Pattern Recognition, vol. 47, no. 9, pp. 2825–2838, 2014.

[9] D. E. King, “Dlib-ml: A machine learning toolkit,”JMLR, vol. 10, pp. 1755–1758, 2009.

[10] N.-T. Do, I.-S. Na, and S.-H. Kim, “Forensics face detection from gans using convolutional neural network,” ISITC, vol. 2018, pp. 376–379, 2018.

[11] X. Xuan, B. Peng, W. Wang, and J. Dong, “On the generalization of gan image forensics,” in Chinese conference on biometric recognition. Springer, 2019, pp. 134–141.

[12] P. Yang, R. Ni, and Y. Zhao, “Recapture image forensics based on laplacian convolutional neural networks,” in International Workshop on Digital Watermarking. Springer, 2016, pp. 119–128.

[13] B. Bayar and M. C. Stamm, “A deep learning approach to universal image manipulation detection using a new convolutional layer,” in Proceedings of the 4th ACM workshop on information hiding and multimedia security, 2016, pp. 5–10.

[14] T. Zhao, X. Xu, M. Xu, H. Ding, Y. Xiong, and W. Xia, “Learning self-consistency for deepfake detection,” in Proceedings of the IEEE/CVF international conference on computer vision, 2021, pp. 15023–15033.

[15] D. Afchar, V. Nozick, J. Yamagishi, and I. Echizen, 2018 IEEE international workshop on information forensics and security (WIFS). IEEE, 2018, pp. 1–7.

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

[17] Yuezun Li, Ming-Ching Chang and Siwei Lyu "Exposing AI Created Fake Videos by Detecting Eye Blinking" in arxiv.

[18] Huy H. Nguyen, Junichi Yamagishi, and Isao Echizen "Using capsule networks to detect forged images and videos".

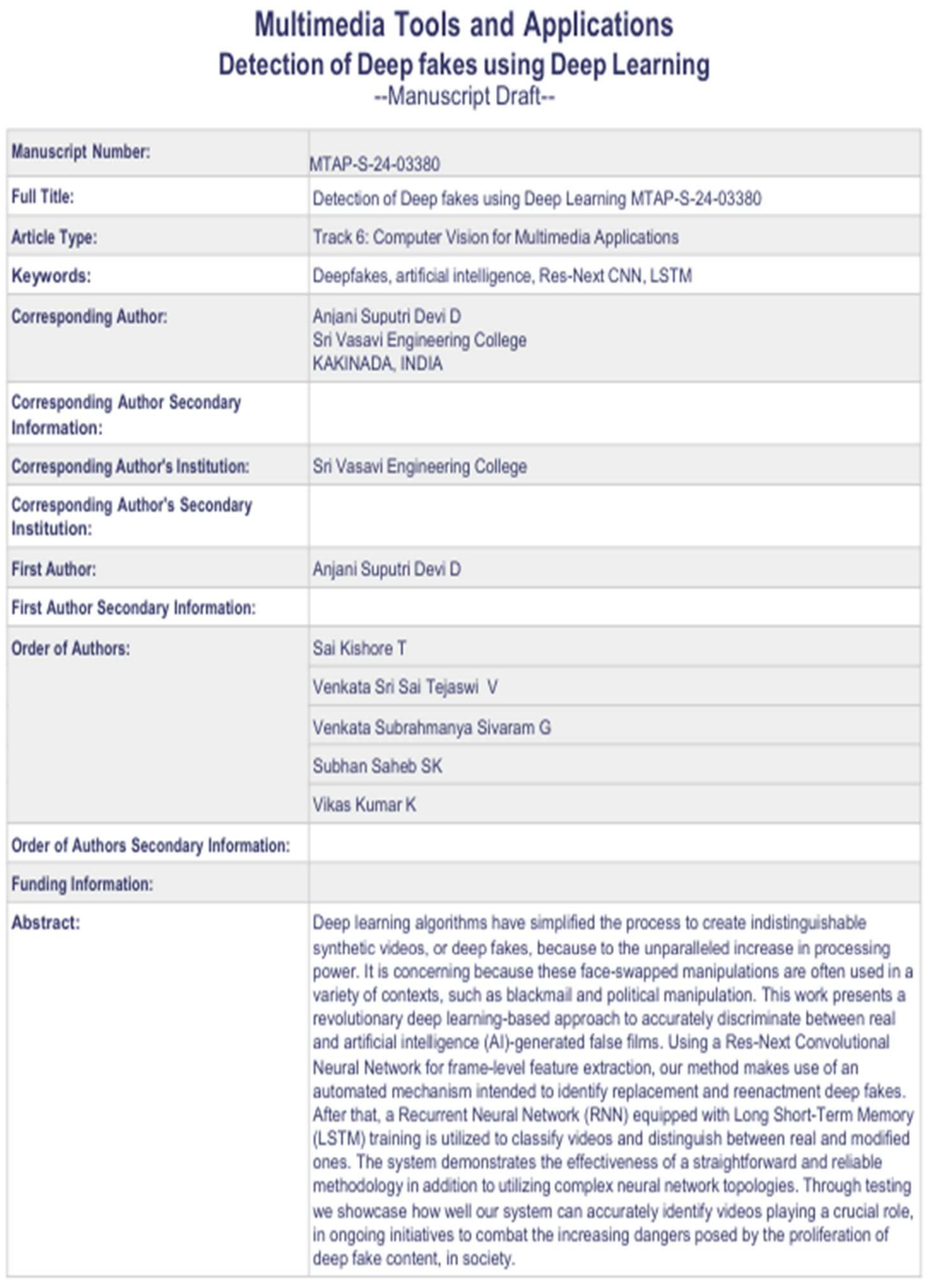
[19] Hyeongwoo Kim, Pablo Garrido, Ayush Tewari and Weipeng Xu "Deep Video Portraits" in arXiv:1901.02212v2.

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

[21] https://twitter.com/madhuriadnal/status/1749734395068739967

[22] https://twitter.com/sachin_rt/status/1746794062961950824?ref_src=twsrc%5Etfw%7Ctwcamp%5Etweetembed%7Ctwterm%5E1746794062961950824%7Ctwgr%5Eda7b6f40036aa90f57da6a01f8fd1d5bdab81712%7Ctwcon%5Es1_&ref_url=https%3A%2F%2Fwww.wionews.com%2Fsports%2Fsachin-tendulkars-deepfake-video-goes-viral-he-says-such-rampant-misuse-of-technology-is-disturbing680191

PROOF OF COMMUNICATION TO JOURNAL REGARDING MAIN PROJECT



Multimedia Tools and Applications

Detection of Deep fakes using Deep Learning

--Manuscript Draft--

Manuscript Number:	MTAP-S-24-03380
Full Title:	Detection of Deep fakes using Deep Learning MTAP-S-24-03380
Article Type:	Track 6: Computer Vision for Multimedia Applications
Keywords:	Deepfakes, artificial intelligence, Res-Next CNN, LSTM
Corresponding Author:	Anjani Suputri Devi D Sri Vasavi Engineering College KAKINADA, INDIA
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	Sri Vasavi Engineering College
Corresponding Author's Secondary Institution:	
First Author:	Anjani Suputri Devi D
First Author Secondary Information:	
Order of Authors:	Sai Kishore T
	Venkata Sri Sai Tejaswi V
	Venkata Subrahmanya Sivaram G
	Subhan Saheb SK
	Vikas Kumar K
Order of Authors Secondary Information:	
Funding Information:	
Abstract:	Deep learning algorithms have simplified the process to create indistinguishable synthetic videos, or deep fakes, because to the unparalleled increase in processing power. It is concerning because these face-swapped manipulations are often used in a variety of contexts, such as blackmail and political manipulation. This work presents a revolutionary deep learning-based approach to accurately discriminate between real and artificial intelligence (AI)-generated false films. Using a Res-Next Convolutional Neural Network for frame-level feature extraction, our method makes use of an automated mechanism intended to identify replacement and reenactment deep fakes. After that, a Recurrent Neural Network (RNN) equipped with Long Short-Term Memory (LSTM) training is utilized to classify videos and distinguish between real and modified ones. The system demonstrates the effectiveness of a straightforward and reliable methodology in addition to utilizing complex neural network topologies. Through testing we showcase how well our system can accurately identify videos playing a crucial role, in ongoing initiatives to combat the increasing dangers posed by the proliferation of deep fake content, in society.

Powered by Editorial Manager® and ProduXion Manager® from Aries Systems Corporation