DEEP FAKE DETECTOR USING CONVOLUTIONAL NEURAL NETWORKS

BACHELOR OF TECHNOLOGY IN

COMPUTER SCIENCE AND ENGINEERING - DATA SCIENCE

Submitted by

N. Sai Yudhi (22071A67A0)

ABSTRACT

Deepfake technology, which runs advanced AI, makes it extremely easy to create fake videos that look very realistic, representing the serious challenges of online truth and digital security. These AI-generated videos are hardly distinguished from real film materials to the human eye. To counter this, we have developed a robust recognition system that combines spatial characteristics extraction of res-nex text folding networks with time sequence modeling of long-term memory (LSTM) networks (long-term memory). This allows the system to analyze both individual video frames and their changes over time, effectively distinguishing the depth of the actual video. In training a comprehensive dataset using FaceForensics++, Celeb-DF, and Deepfake Detection Challenge, the model shows a powerful generalization of the real world. We also created a user-friendly web application that allows you to upload all your videos and instantly store feedback on their reliable reliability.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

In recent years, the digital landscape has undergone a dramatic transformation, with artificial intelligence (AI) playing a central role in revolutionizing information creation, sharing and consumption. One of the most controversial and fastest AI applications is the creation of DeepFakes-HyperRealistic, using AI-generated videos that simulate real human behavior and language with incredible accuracy.

Initially, deepfakes developed as an empirical demonstration of deep learning, becoming a technological phenomenon with important social, political and ethical implications. These synthetic videos are so visually persuasive that they frequently avoid human recognition, making it almost impossible to distinguish between actual content and invented content without special tools.

This growing realism is a serious threat to the authenticity of information, as it is a deep purpose for malicious purposes such as misinformation, political manipulation, revenge, identification, and defamation. Although technology for creating content is developed quickly, recognition mechanisms must keep up to ensure that the abuse of synthetic media does not affect public trust, security, or digital integrity. This project deals with deep learning solutions that can recognize deep-faf video through a hybrid architecture that combines spatial and temporal analysis.

Our approach uses two stage architectures for neural networks. First, we use a reconnect diagram diagram neural network (CNN) to extract rich spatial features from individual video frames. These frame-level representations record subtle visual anomalies and artifacts, such as inconsistent lighting, unnatural skin textures, and irregular facial movements, often introduced in the course of deeper generations. The extracted properties are then given a long short-term memory (LSTM) network, a variant of a recurrent neuronal network (RNN) that analyzes time dependence and patterns on frame sequences.

This combination allows the system to perform a thorough analysis of the visual and temporal

integrity of the video. These data records provide a variety of facial manipulation, environment and lighting conditions, allowing the model to learn generalized patterns of deep fuck artifacts. In addition to this training process, pre-processing techniques such as facial recognition, framework extraction and dissolution normalization were implemented to maintain consistency and improve identification accuracy.

Convolutional Neural community (CNN)

Introduction:

A Convolutional Neural network (CNN) is a deep mastering structure particularly designed for photograph and video processing. CNNs are particularly powerful for obligations on the aspect of photograph magnificence, facial reputation, and item detection because of their capability to routinely research spatial hierarchies of features.

Structure components:

Enter Layer: Accepts picture records (e.g., RGB pixel values).

Convolutional Layers: exercise filters to extract close by skills such as edges and textures.

Activation function: Introduces non-linearity into the network, usually the use of ReLU.

Pooling Layers: Downsample the function maps to lessen dimensionality and computational load.

Absolutely related Layers: Interpret the extracted capabilities and convey category consequences.

Softmax/Output Layer: Converts very last outputs into elegance opportunities.

Benefits:

Mechanically learns beneficial visible capabilities without guide intervention.Reduces the need for giant preprocessing.Nicely-relevant for detecting spatial artifacts and irregularities in image frames.

2. **ResNeXt**

ResNeXt is a deep convolutional community structure that builds upon the ResNet (Residual community) shape. It introduces the concept of cardinality, which refers back to the form of parallel transformation paths inside each block.

Key principles

Residual gaining knowledge of: Shortcut connections help in education deep networks thru mitigating vanishing gradients.

Grouped Convolutions:

Instead of a unmarried massive convolution, a couple of small convolutions are finished in parallel. Break up-remodel-Merge technique: The enter is cut up into multiple branches, each processed independently, and then merged.

Evaluation with ResNet:

ResNet specializes in increasing depth.ResNeXt will growth cardinality, providing a stability among intensity, width, and performance.

Advantages in Deepfake Detection:

Learns richer and extra complex features with efficient computation.better generalization and accuracy than elegant CNNs.specifically appropriate at detecting excessive-frequency anomalies in faux photographs.

3. **Prolonged short-term memory (LSTM):**

Introduction:

LSTM is a specialised sort of Recurrent Neural community (RNN) designed to have a look at prolonged-term dependencies in sequential records. it is broadly applied in time series evaluation, natural language processing, and video data processing.

Architecture additives

Reminiscence mobile: Shops relevant information throughout time steps.

Enter Gate: Makes a selection which values from the input need to be delivered to the memory.

Overlook Gate: Determines which data need to be discarded.

Output Gate: Comes to a decision what a part of the memory to output at each time step.

LSTM in series Processing:

LSTM networks system a chain of inputs (together with photograph features from video frames) and take a look at temporal patterns and dependencies. In deepfake detection, it may model how facial expressions and movements change over the years.

Blessings in Deepfake Detection:

Detects inconsistencies in movement or temporal transitions.Learns lengthy-range temporal dependencies.Appropriate for analyzing video sequences in which temporal coherence topics.

4. The use of CNN, ResNeXt, and LSTM for Deepfake Detection

Deepfake Detection

Workflow Preprocessing:

Extract person frames from the video.Come across and crop faces the use of face detection algorithms.Resize and normalize the cropped faces for version enter.Function Extraction using CNN or ResNeXt.Each frame is passed via a CNN or ResNeXt to extract spatial functions. Those features seize facial information, textures, and artifacts specific to manipulated content material.

Temporal Modeling with LSTM:

The collection of features from consecutive frames is exceeded into an LSTM.

CHAPTER 2
LITERATURE SURVEY / EXISTING WORK

The rise of Deepfake videos has become one of the most urgent challenges in the digital media era. The technology behind deepfakes, often operated by geese (a generically controversial network) and automated encoders, has progressed rapidly, but researchers have also worked to develop methods of identifying manipulated content that can reliably distinguish it from actual film material. This literature review checks the most important contributions in this field.

2.1 EXISTING WORK

Some of the existing models

1. **Face Warping Artifact Detection**

   One of the early endeavors in deepfake detection focused on recognizing anomalies created during the face substitution procedure. Li et al. devised an approach utilizing CNNs to identify distortions in the facial area—a common artifact that emerges when adjusting the size of generated faces. Their model contrasted the altered facial region with its surroundings to identify unnatural changes. While it performed well on certain datasets, its efficacy decreased when dealing with high-quality deepfakes that minimize artifacts.

2. **Eye Blink Detection**

   Other progressive technique changed into based totally at the statement that early deepfakes regularly did now not show off realistic eye blinking. Scientists leveraged an LRCN (long-time period Recurrent Convolutional community) to evaluate eye actions in video sequences. using this approach, motion pictures were categorised as faux if they lacked or showed inconsistent natural blinking patterns.

3. **Capsule Network-Based Detection**

   Nguyen et al. delivered the usage of tablet networks, that can model spatial relationships and pose information among facial features. Their approach helped to come across subtle inconsistencies in geometry and shape that deepfake generators often battle to copy correctly. Tablet networks had been especially useful in identifying forged or replay assaults.

4. **FaceForensics++**

   In addition to offering detection methods, Rossler et al. added one of the more extensively used datasets for schooling and benchmarking deepfake detection systems. They used XceptionNet to come across manipulations and showed that huge-scale supervised education improves detection overall performance.

5. **Deepfake Detection Challenge**

   To inspire the improvement of robust and scalable detection solutions, facebook released the DFDC dataset and a challenge. The dataset includes thousands of manipulated motion pictures featuring numerous actors and manipulation techniques, making it perfect for training popular- detection tools.

2.1.1 Drawbacks by previous models

- Lacks the ability to generalize across newer, high-resolution deepfakes where artifact traces are minimal or entirely corrected.
- Required high-resolution and well-lit videos to capture biological signals, which is not always practical
- Requires high computational resources for effective model training.

While numerous strategies have been proposed for detecting deepfakes—ranging from artifact detection and eye tracking to deep neural networks and biological sign analysis—each has its strengths and weaknesses. most early fashions centered either on spatial functions or short-time period styles, regularly failing in real-global, noisy scenarios.

This task builds on the strengths of those previous approaches with the aid of combining spatial feature extraction (thru ResNext CNN) with temporal sequence modeling (via LSTM RNN). It additionally addresses the era difficulty by using schooling on a combined dataset from FaceForensics++, DFDC, and celeb-DF, making sure exposure to a huge sort of deepfake generation techniques and video conditions

1. Popular RNN/LSTM with out visible function Extraction:

Description: Those models manner raw frame sequences or low-stage features without effective spatial encoding.

Drawbacks:

Inadequate characteristic representation: raw input lacks excessive-degree skills essential to encounter visible artifacts.immoderate Computational value: long video sequences could make schooling unstable and sluggish with out spatial pre-processing.

Overfitting hazard: LSTMs on my own, without CNN-based totally function input, commonly have a tendency to overfit because of restricted learning ability from raw inputs.

2. Single-Modality models

Description: a few fashions use nice seen or audio input, now not each.

Drawbacks:

Incomplete assessment: Deepfakes frequently involve mismatches in both video (visual) and audio

(voice) cues. Ignoring one modality reduces accuracy.

Less tough to Detect: Attackers can take advantage of the not noted modality (e.g., syncing faux audio with actual video).

Bad bypass-Dataset overall performance: Accuracy drops considerably whilst tested on exceptional datasets like FaceForensics++, film famous person-DF, or DFDC.Restrained real-international software program: those fashions are not strong sufficient for deployment in various or dynamic environments.

3. Insufficient Temporal Context in quick Clips :

Description: fashions educated on very short clips (2–three frames) may additionally pass over important patterns in longer sequences.Restricted potential: now not capable of capture complex styles in manipulated content.

Low Robustness: Extraordinarily touchy to modifications in lighting fixtures, selection, and video tremendous.

In cyber security context, we cannot avoid testing the windows that are connected to the system of the machine to find and fix the vulnerabilities surfaced beforehand. The above tests would help determine whether the security system is by current attacks or not as well as to what extent the system can foresee the attacks that could happen under the given operating environmental and security conditions. Cyber Security is a process which takes cyber evaluations of a system's ability to operate under various attacks, and then builds a model on what measures need to be taken for a successful defense. We are very confident that we will be able to solve the problem by revealing the level of possible harm within the named system. These consequences may be immediately seen post an attack or they might remain hidden until discovered after a life under threat, upon which purposeful screening of the comprehensive system will be executed to prove its safety.

CHAPTER 3

SOFTWARE REQUIREMENTS

3.1 Platform Requirements

- **Operating System**:
    - Windows 10 or higher
    - Ubuntu 18.04 or higher
- **Programming Language**:
    - Python 3.6 or above
- **Frameworks and Libraries**:
    - PyTorch 1.4 (for deep learning model implementation)
    - Django 3.0 (for backend web development)
    - OpenCV (for image and video preprocessing)

3.2 Non-functional Requirements

- **Security:** All uploaded videos should be encrypted during transmission (SSL).
- **Performance:** version inference should take no extra than 1 minute for a ten-second video.
- **Reliability:** The software ought to cope with invalid uploads gracefully with suitable error messages.
- **Scalability:** The device have to be scalable to handle more and more hazard fashions and users.
- **Compatibility:** The machine ought to paintings throughout principal browsers (Chrome, Firefox, edge) and it have to assist video formats like MP4, AVI, and MOV.

CHAPTER 4

SOFTWARE DESIGN

4.1 UML Diagrams

Unified Modeling Language (UML) is a standardized manner to visualize the design of a machine. in this task, the principle UML diagram serves as a blueprint that represents the architecture and workflow of the Deepfake Detection system using a hybrid ResNeXt CNN–LSTM model. The UML diagrams assist smash down the device's complexity into understandable visual models for improvement, conversation, and documentation.

4.1.1 Class Diagram

The class diagram models the core structure of the deepfake detection system by defining its main components and their relationships.

- VideoProcessor: In charge of taking frames out of the video that has been uploaded.
- Frame: frame is a single photo taken from the video. It incorporates photograph statistics on the pixel stage.
- ResNeXtCNN: This module extracts wealthy spatial functions from character frames the use of the ResNeXt Convolutional Neural network.
- **LSTM**: The Long Short-Term Memory network learns the temporal dynamics between frames by processing the feature sequence.
- **DeepfakeDetector**: Coordinates the detection process by analyzing video and producing results using CNN and LSTM.
- **ResultAnalyzer:** Determines if the video is authentic or manipulated (deepfake) based on the output from LSTM.

This diagram helps in understanding the object-oriented architecture and how each class contributes to the pipeline.

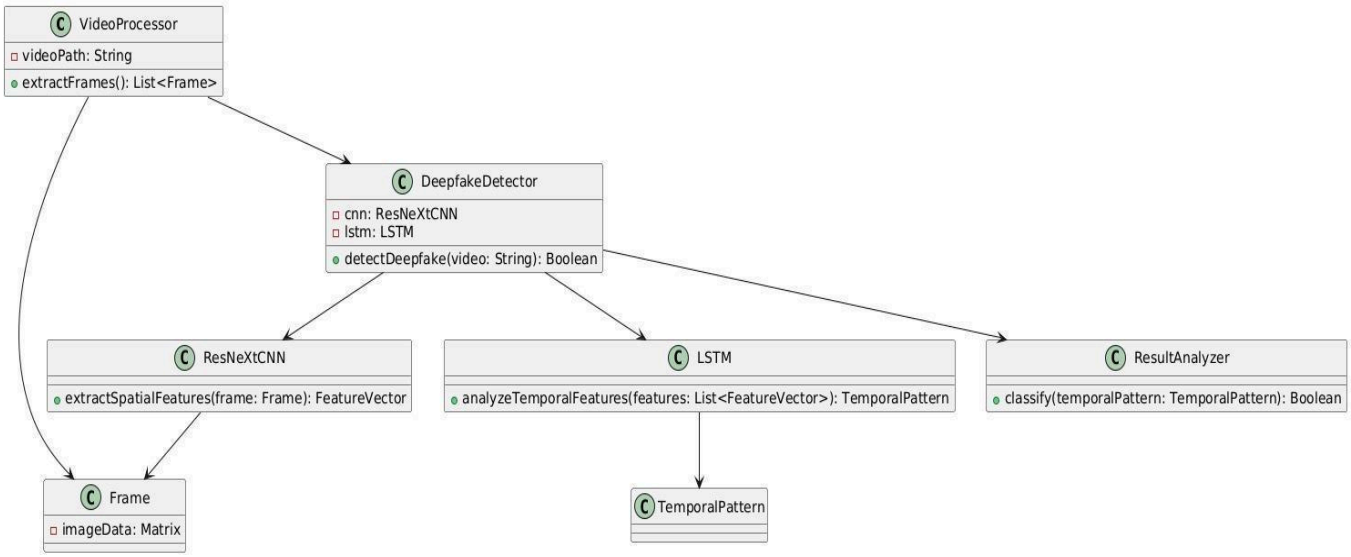


Fig 4.1.1. Class Diagram

4.1.2 Use Case Diagram

The use case diagram models the interaction among users (actors) and the device.

- **User**: A preferred user who uploads the video and views the detection result.
- **Admin**: A privileged person who can teach or first-rate-song the detection version the usage of labeled statistics.

Use instances:

- **Upload Video**: The user submits a video for evaluation.
- **Detect Deepfake:** The gadget techniques the video to determine if it's miles faux.
- **View results:** The person is proven the class end result.
- **Train model**: Admins can update the model to enhance accuracy the usage of new datasets.

This diagram defines the system functionality from the end-user perspective.

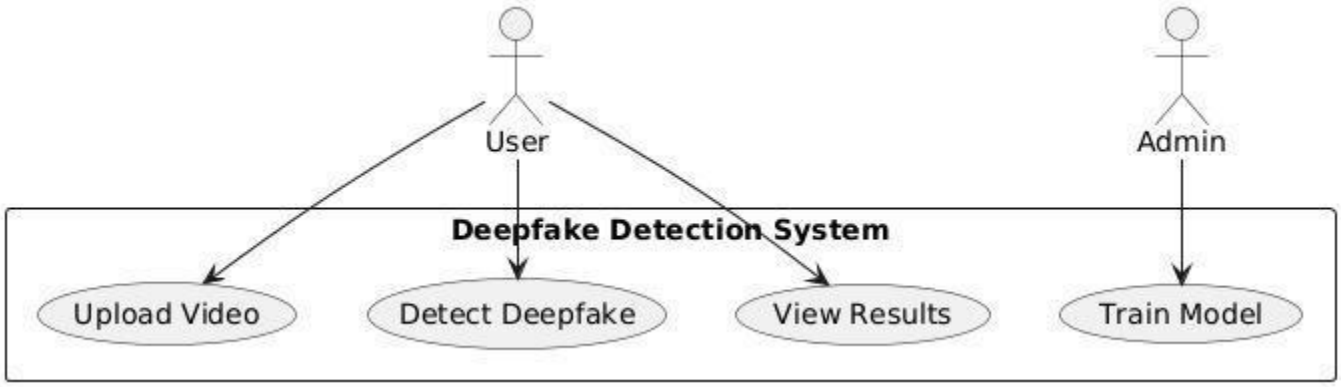


Fig 4.1.2. Use Case Diagram

4.1.3 Sequence Diagram

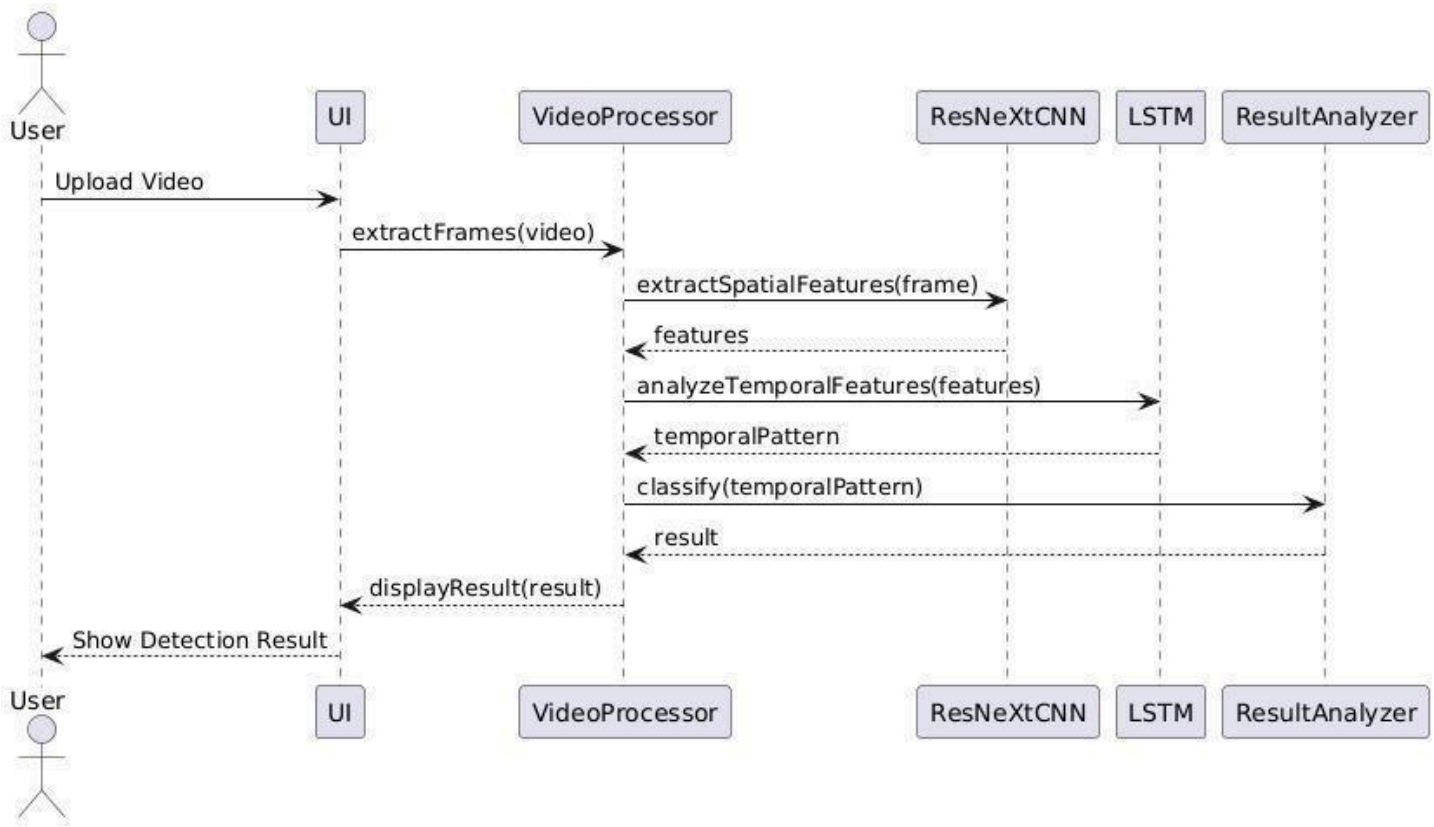
The sequence diagram fashions the dynamic conduct and data go with the flow for deepfake detection.Person uploads a video thru the interface.

- UI sends the video to VideoProcessor, which extracts person frames.
- Each frame is despatched to the ResNeXtCNN to extract spatial capabilities.
- The listing of extracted functions is exceeded to LSTM for temporal analysis.
- The LSTM returns a temporal pattern taking pictures movement consistency.
- ResultAnalyzer receives the pattern and classifies the video as real or fake.
- The final end result is displayed returned to the user through the UI.
- This diagram captures how the machine components collaborate in real-time to manner a videoand return a deepfake detection result.

Fig: 4.1.3. Sequence Diagram.

This diagram defines the system functionality from the end-user perspective.

Uses of UML in This Project:



1. **Machine design & making plans:**

   UML enables in designing the machine structure before actual implementation, saving time and decreasing layout errors.

2. **Group communication:**

   Presents a not unusual visible language for developers, testers, and stakeholders to discuss and recognize the gadget.

3. **Requirement evaluation:**

   Use case diagrams clarify what functionalities the machine must help, supporting in better requirement accumulating.

4. **Code improvement steerage:**

   Elegance diagrams help in identifying key classes, methods, and interactions, making it less complicated to enforce smooth, modular code.

5. **Documentation & renovation:**

   UML serves as a protracted-term reference for destiny improvements, debugging, or scaling the system.

4.1.4 Workflow Diagram

Explanation for Report/Presentation

Workflow Diagram Explanation:

This diagram outlines the operational pipeline of the Deepfake Detection system:

1. **User Uploads Video**: The method begins off evolved while a person uploads a video for analysis.

2. **Frame Extraction**: The device extracts person frames from the video.

3. **Preprocessing**: each body is preprocessed (e.g., resized, normalized) to healthy the enter necessities of the CNN version.

4. **Spatial characteristic Extraction**: ResNeXt CNN extracts specific spatial functions from each body, identifying pixel-degree manipulations.

5. **Temporal feature evaluation:** The spatial functions are passed to an LSTM network to seize movement continuity and facial dynamics over the years.

6. **Class:** based on the found out temporal patterns, the machine classifies the video as both real or fake.

7. **Display effects**: The final classification end result is proven to the person.

This workflow ensures a sturdy pipeline for detecting both static and dynamic artifacts present in deepfake motion pictures.

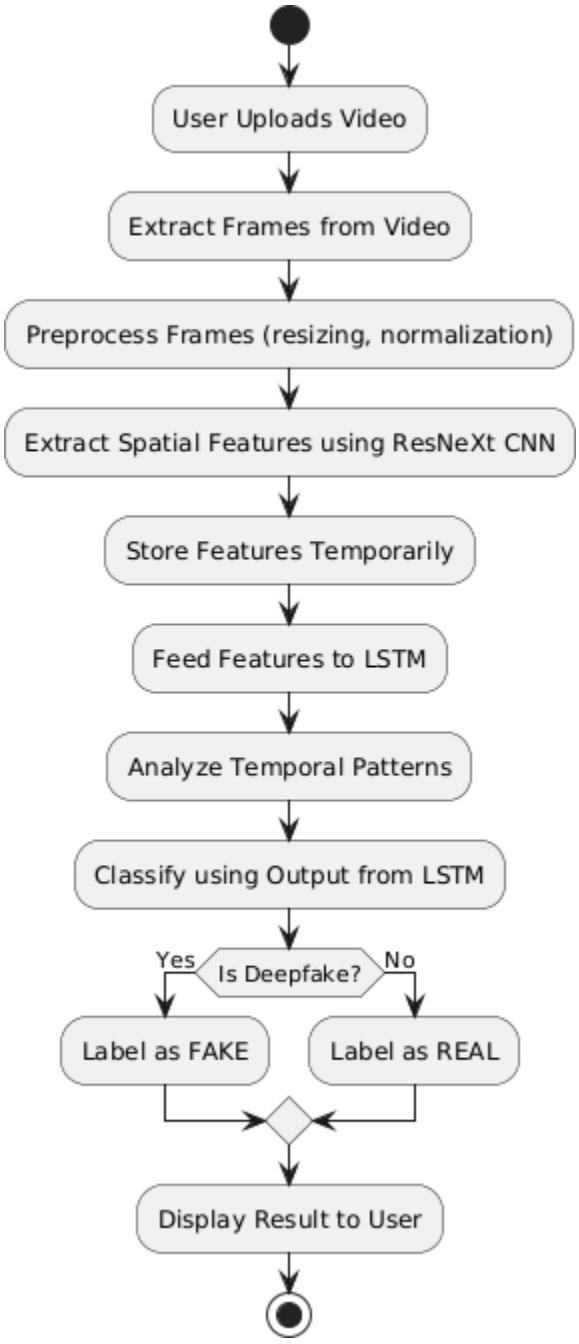


Fig 4.1.4. WorkFlow Diagram.

4.1.5 Component Diagram

Component Diagram

The UML aspect Diagram for the Deepfake Detection device gives a high-stage view of the system's modular architecture, illustrating how different functional units (components) engage to stumble on deepfakes in video content.

Key components and Their Roles:

User Interface:

Acts as the number one get right of entry to factor for users to upload videos and think about detection results. Passes the video to the frame Extractor and initiates the analysis.

Body Extractor:

Splits the uploaded video into character frames. Each body turns into an enter to the following steps inside the pipeline.

Preprocessing Module:

Systems statistics for time-series analysis .LSTM Module (long short-time period memory)Analyzes temporal dynamics across the series of frames. Detects motion-primarily based inconsistencies ordinary in deepfake motion pictures.

Temporal Analyzer:

Translates LSTM output to pick out styles through the years. Enables differentiate between real and pretend videos based on expression flow and continuity.

Type Module:

Makes the final decision with the aid of evaluating the output of the CNN and LSTM

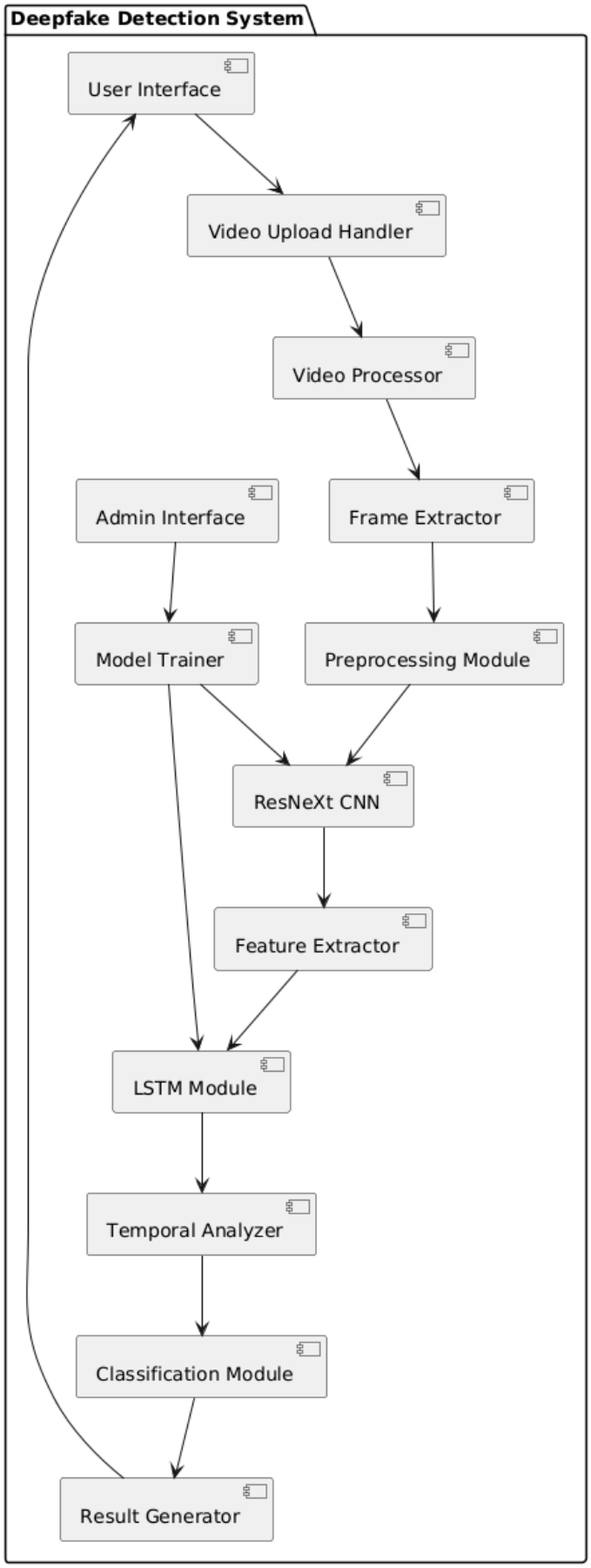


Fig 4.1.4. Component Diagram.

CHAPTER 5
PROPOSED SYSTEM

The proposed deepfake detection device is designed the use of a hybrid deep gaining knowledge of structure that combines spatial feature extraction with temporal sequence analysis. The methodology follows a multi-stage pipeline, integrating records preprocessing, deep function learning, sequence modeling, and prediction. the subsequent outlines every key issue of the proposed technique:

1. Information Preprocessing Pipeline

To make certain inputs for the neural network, the following preprocessing steps are applied to all movies in the dataset:

Face Detection and Alignment:

using MTCNN or OpenCV's Haar Cascade, faces are extracted from each frame and aligned to make certain consistency throughout the dataset.

Frame Extraction:

Movies are decomposed into frame sequences at constant durations (e.g., 25 fps), keeping a steady wide variety of frames in step with video clip.

Normalization:

Pixel values are normalized to a standard scale (e.g., [0, 1]) and resized to a fixed decision (e.g., 224x224) for compatibility with the CNN input.

Facts Augmentation:

To improve generalization, modifications inclusive of flipping, rotation, noise injection, and assessment adjustments are implemented at some stage in education.

2. Characteristic Extraction with ResNeXt (Spatial analysis)

The first degree of the model makes use of a pre-trained ResNeXt-50 or ResNeXt-one zero one convolutional neural network to extract wealthy spatial functions from individual frames:

Layer utilization:

capabilities are extracted from the penultimate convolutional block of ResNeXt to achieve excessive-dimensional representations (e.g., 2048-d feature vectors).

Transfer getting to know:

The network is initialized with ImageNet weights and fine-tuned on the deepfake dataset to capture project-unique visible anomalies.

Why ResNeXt?

ResNeXt's grouped convolutions and increased cardinality provide the capacity to research complex patterns along with unnatural textures, lighting mismatches, or boundary artifacts regularly found in solid frames.

3. Modeling with LSTM (Temporal analysis)

The spatial functions from every frame are sequentially fed into a long short-term reminiscence (LSTM) network to research temporal dynamics across movies:

LSTM Configuration:

enter size: equal to the characteristic vector measurement from ResNeXt (e.g., 2048)

Hidden length: 512–1024 gadgets

Layers: 1–2 stacked LSTM layers

Dropout: implemented to save you overfitting

Temporal functions Captured:

The LSTM identifies temporal irregularities, inclusive of:

Inconsistent facial motion or expression waft

Unnatural blinking or gaze path

Abrupt transitions or glitches across frames

4. Class Layer

The very last hidden kingdom of the LSTM is exceeded thru a completely connected (dense) layer, observed by means of a softmax or sigmoid activation (depending on whether or not binary or multi-elegance class is used):

Output:

Binary Label: real or fake

self belief score: chance of prediction

5. Education

method Loss

feature:

Binary cross-Entropy (for 2 instructions) or categorical go-Entropy (for multiple kinds of deepfakes)

Optimizer:

Adam optimizer with a getting to know rate scheduler

Early stopping:

Monitors validation loss to save you overfitting

Batch length & Epochs:

Normally sixteen–32 video samples according to batch, educated for 20–50 epochs relying on dataset size and convergence

6. Assessment Metrics

To assess version overall performance, the following metrics are used:

Accuracy: normal classification correctness

Precision and keep in mind: mainly critical for identifying fake content

F1-rating: Balances precision and keep in mind

ROC-AUC rating: For evaluating version discrimination capability

Confusion Matrix: Visualizes genuine vs fake predictions

CHAPTER 6

CODING

6.1 DATA AUGMENTATION FOR TRAINING STABILITY

This code snippet defines the load_data() feature, which hundreds and preprocesses video body statistics from a nearby dataset prepared into actual and faux folders. every subfolder includes a couple of video frame directories. The characteristic performs the subsequent operations :

- Iterates through the real and fake classes, assigning labels hence (0 for real, 1 for fake).
- Limits the wide variety of frames processed in line with video using the MAX_FRAMES_PER_VIDEO constant.
- Reads and resizes every picture body to a set duration (IMG_SIZE × IMG_SIZE).
- Handles any exceptions eventually of photo loading.
- Converts the photograph and label lists into NumPy arrays and one-warm encodes the labels.

backend > prepare_dataset.py > ...

```python
import os
import cv2
import numpy as np
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import ImageDataGenerator

IMG_SIZE = 128  # Reduced from 224 to 128
MAX_FRAMES_PER_VIDEO = 10  # Limit number of frames per video
dataset_dir = "dataset_frames"

def load_data():
    X, y = [], []
    for category in ['real', 'fake']:
        category_path = os.path.join(dataset_dir, category)
        label = 0 if category == 'real' else 1  # 0 for real, 1 for fake
        for video_folder in os.listdir(category_path):
            folder_path = os.path.join(category_path, video_folder)
            if not os.path.isdir(folder_path):
                continue

            frame_files = sorted(os.listdir(folder_path))[:MAX_FRAMES_PER_VIDEO]  # Limit frames
            for img_file in frame_files:
                img_path = os.path.join(folder_path, img_file)
                try:
                    img = cv2.imread(img_path)
                    img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
                    X.append(img)
                    y.append(label)
                except Exception as e:
                    print(f"Error loading {img_path}: {e}")

    X = np.array(X, dtype="float32") / 255.0
    y = to_categorical(y, 2)
    return train_test_split(X, y, test_size=0.2, random_state=42)
```

Fig 6.1.1. Code Snippet of Data Loading and Preprocessing for Deepfake Detection

The first segment of the testing manner inside the Deepfake Video Detector involves the augmentation of the schooling dataset to improve the model's generalization. The gadget leverages Keras' ImageDataGenerator to use random alterations to the input photographs, thereby artificially growing the dataset length and variety.

backend > prepare_dataset.py > ...

```python
def augment_data(X_train):
    # Augmenting the data using ImageDataGenerator
    datagen = ImageDataGenerator(
        rotation_range=40,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest'
    )
    datagen.fit(X_train)
    return datagen

if __name__ == "__main__":
    X_train, X_test, y_train, y_test = load_data()
    np.save("X_train.npy", X_train)
    np.save("X_test.npy", X_test)
    np.save("y_train.npy", y_train)
    np.save("y_test.npy", y_test)
    print("✅ Dataset prepared and saved successfully.")

    # Data Augmentation
    augment_data(X_train)

```

Fig: 6.1.2. Code Snippet of Data Loading and Preprocessing for Deepfake Detection

6.2 Frame Processing

```python
import cv2
import os

def extract_frames_from_video(video_path, output_dir):
    # Read the video file
    cap = cv2.VideoCapture(video_path)

    # Get frame rate of the video
    fps = cap.get(cv2.CAP_PROP_FPS)

    # Ensure the output directory exists
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    frame_count = 0
    while True:
        ret, frame = cap.read()
        if not ret:
            break

        # Resize to 224x224
        frame = cv2.resize(frame, (224, 224))

        # Save the frame as an image
        frame_filename = os.path.join(output_dir, f"frame_{frame_count}.jpg")
        cv2.imwrite(frame_filename, frame)

        frame_count += 1

    # Release the video capture object
    cap.release()
    print(f"Extracted {frame_count} frames from {video_path}")

# Example usage: extract frames for all real and fake videos
videos_dir = "dataset_frames"  # Root directory where real/fake videos are stored
for category in ['real', 'fake']:
    video_dir = os.path.join(videos_dir, category)
    for video_name in os.listdir(video_dir):
        if video_name.endswith('.mp4'):
            video_path = os.path.join(video_dir, video_name)
            output_dir = os.path.join(videos_dir, category, video_name.split('.')[0])  # Create a folder for each video
            extract_frames_from_video(video_path, output_dir)

```

Fig 6.2.1. Code Snippet of adding frame processing.

6.3 DATA AUGMENTATION FOR TRAINING STABILITY

Programming Language:

- Python – extensively used in device studying and computer imaginative and prescient b because of its rich surroundings.

Deep studying Frameworks:

- PyTorch / TensorFlow – Used for building and education ResNeXt CNN and LSTM fashions.

- Keras (if using TensorFlow backend) for version abstraction.

Preprocessing & facts coping with:

- OpenCV – For video body extraction and picture processing.
- NumPy / Pandas – For coping with arrays and tabular statistics.

Version components:

- ResNeXt CNN – For spatial feature extraction from every person.

- LSTM (prolonged quick-term reminiscence) – For taking pictures temporal dependencies among frames.

video_detection.py | detection.py | app.py | train_model.py

backend > app.py > ...

```python
from flask import Flask, request, jsonify
from flask_cors import CORS
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import numpy as np
import os
import tempfile
import cv2

# Initialize Flask app
app = Flask(__name__)
CORS(app)  # Enable CORS for frontend communication

# Load the model once at startup
MODEL_PATH = "best_model.h5"
IMG_SIZE = 128
THRESHOLD = 0.5
model = load_model(MODEL_PATH)

# Image Prediction Endpoint
@app.route('/predict', methods=['POST'])
def predict_image():
    file = request.files.get('file')
    if not file:
        return jsonify({'error': 'No file uploaded'}), 400

    try:
        # Save uploaded image temporarily
        with tempfile.NamedTemporaryFile(delete=False, suffix=".jpg") as temp:
            file.save(temp.name)
            image_path = temp.name

        # Preprocess the image
        img = load_img(image_path, target_size=(IMG_SIZE, IMG_SIZE))
        img_array = img_to_array(img)
        img_array = np.expand_dims(img_array, axis=0)
        img_array = img_array / 255.0
```

Fig 6.3.1. Code Snippet of adding Tech stack

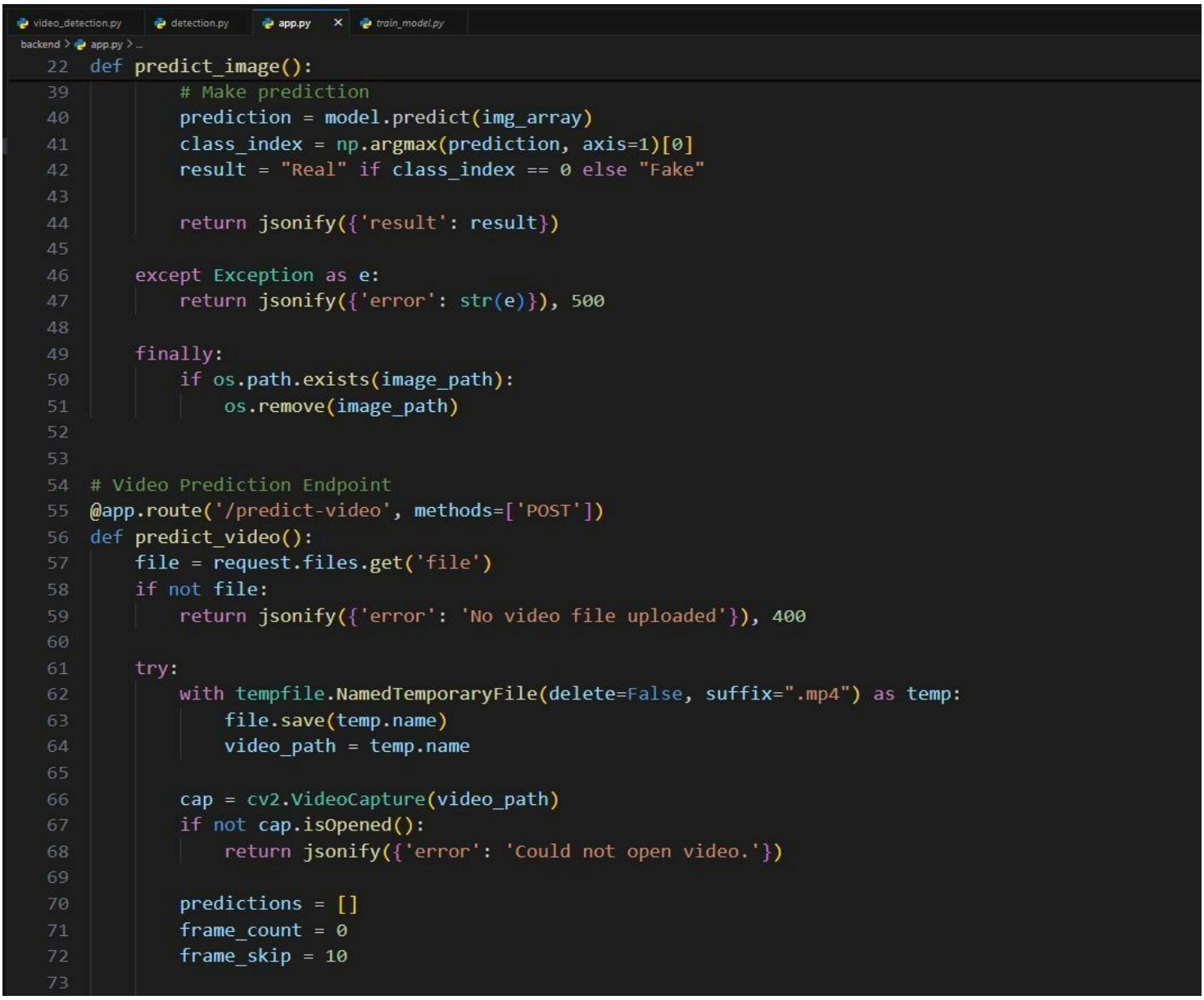


Fig 6.3.2. Code Snippet for Image Prediction Endpoint

```
        while cap.isOpened():
            ret, frame = cap.read()
            if not ret:
                break

            if frame_count % frame_skip == 0:
                resized_frame = cv2.resize(frame, (IMG_SIZE, IMG_SIZE))
                normalized_frame = resized_frame / 255.0
                input_frame = np.expand_dims(normalized_frame, axis=0)
                pred = model.predict(input_frame)[0][0]
                predictions.append(pred)

            frame_count += 1

        cap.release()

        if predictions:
            avg_pred = np.mean(predictions)
            result = "FAKE" if avg_pred < THRESHOLD else "REAL"
            return jsonify({"result": result, "confidence": round(float(avg_pred), 2)})
        else:
            return jsonify({'error': 'No frames processed.'})

    except Exception as e:
        return jsonify({'error': str(e)}), 500

    finally:
        if os.path.exists(video_path):
            os.remove(video_path)


if __name__ == '__main__':
    app.run(debug=True)

```

Fig 6.3.3. Code Snippet for Video Prediction Endpoint

CHAPTER 7
TESTING

In cyber security context, we cannot avoid testing the windows that are connected to the system of the machine to find and fix the vulnerabilities surfaced beforehand. The above tests would help determine whether the security system is by current attacks or not as well as to what extent the system can foresee the attacks that could happen under the given operating environmental and security conditions. Cyber Security is a process which takes cyber evaluations of a system's ability to operate under various attacks, and then builds a model on what measures need to be taken for a successful defense. We are very confident that we will be able to solve the problem by revealing the level of possible harm within the named system. These consequences may be immediately seen post an attack or they might remain hidden until discovered after a life under threat, upon which purposeful screening of the comprehensive system will be executed to prove its safety.

7.1 TESTING TYPES

7.1.1 MANUAL TESTING

White Box Testing: In this test environment we essentially know the way the system is supposed to function leveraging the open-ended nature of the program to better understand how different components interact in their process. This approach entails running the individual patterns along with the lines and the respective branches since the ultimate goal is the confirmation of its correctness. In White Box Testing, however, it is the developers who execute the testing. Developers need to know the programming languages and code structures, before being able to perform this kind of testing.

Black Box Testing: In contrast to White Box Testing, Black Box Testing is implemented like an isolated black box where testers are unable to know any internal details. Tester's shift focuses on the fundamental steps involved in a given business process and critically assess whether
appropriate input processes have been successfully accomplished to meet the desired outputs. Black Box Testing is mainly user-centered and does not have the program code familiarity requirement, therefore it is aimed, in most cases, at the testers without programming expertise.

7.1.2 AUTOMATED TESTING

White Box Testing: Automated White Box Testing is the process of applying automated tools that are designed to work based on the internal structure of the software systems. Through this process, the internal structure is analyzed to detect issues like code coverage and adhesion to the coding standards. These tools can therefore automatically generate test cases based on the structure and logic of the coded to ensure the code for each code path is fully attempted.

Black Box Testing: In the case of Automated Black Box Testing we are using some tools that mimic the user behaviors directly interacting with the software. These tools typically perform automated execution of the previously put in tests, validating outputs against the anticipated results and follow-up with generating reports. Such technology is extremely beneficial for regression testing, which is the process of checking that any changes didn't cause problems in the code and thus automated Black Box Testing can certainly save time and energy of test developers.

7.2 SOFTWARE TESTING METHODS

7.2.1 BLACK BOX TESTING

Black box testing is a kind of software testing that looks at a program's functionality without looking at its core coding or structure. A customer-stated requirements specification serves as the main source of black box testing data. This method, as seen in Fig. 7.2.1, involves the tester choosing a function, providing an input value to assess its functionality, and determining whether the function is producing the desired output or not. The function passes testing if it generates accurate output; if not, it fails. After reporting the outcome to the development team,the test team moves on to testing the subsequent function. If serious issues are found after all features have been tested, the development team is notified so that they can address them.

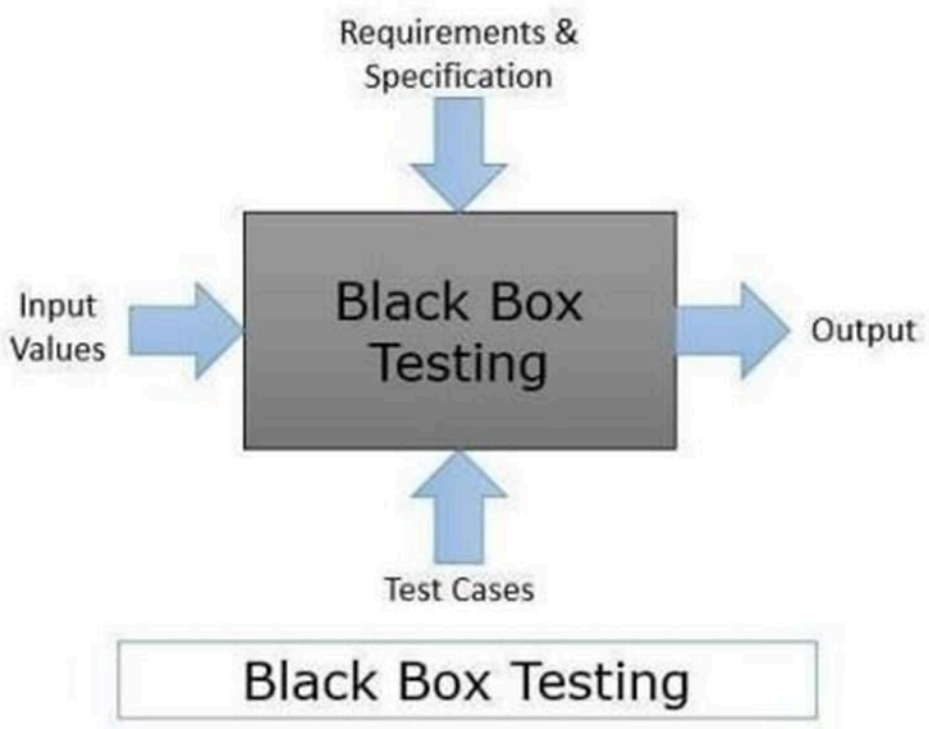


Fig 7.2.1 Black Box Testing

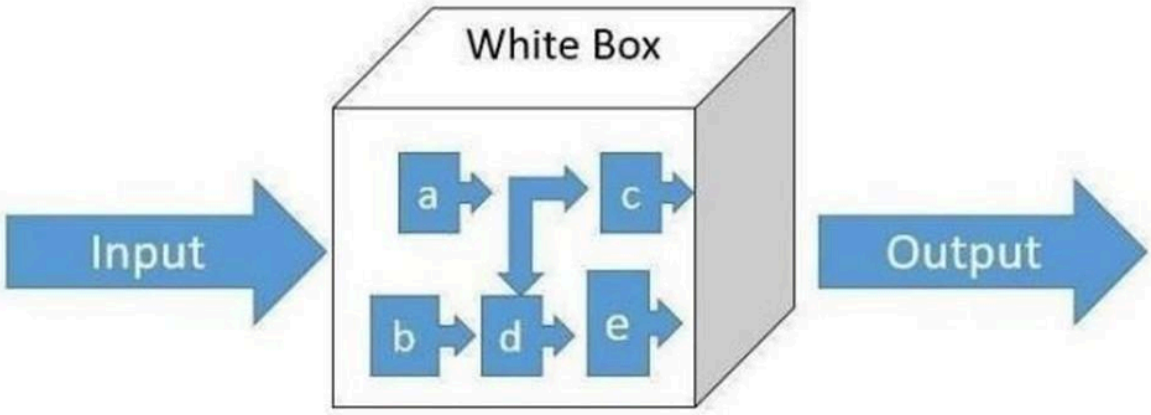
7.2.2 GRAY BOX TESTING

A software testing technique called "grey box" testing allows users to test an application whileknowing only a portion of its internal workings. Because testing procedures are carried out at the functionality level as in black box testing and access to internal coding is required for the formulation of test cases, it is a **hybrid of white box and black box testing**. Web system context-specific faults are frequently found through grey box testing. For instance, if the testerfinds a flaw during testing, he fixes the code and runs the test again in real time. To boost testingcoverage, it focuses on every layer of any complicated software system. It provides the capability to evaluate the internal code structure in addition to the display layer. It is typically employed in penetration and integration testing. Black box and white box testing are combinedin this testing method. When conducting black box testing, the tester is blind to the code. Theyare in possession of data regarding the intended outcome for the supplied input. When doing white box testing, the tester is fully conversant with the code. Grey box testers have some, butnot all, code expertise. An overview of grey box testing is provided in Fig 7.2.2.

Fig 7.2.2 Grey Box Testing

7.2.3 WHITE BOX TESTING

Using a technique known as "**white box testing**," testers can examine and validate a software system's internal operations, including its code, infrastructure, and system integrations. An integral component of automated build procedures in a contemporary Continuous Integration/Continuous Delivery (CI/CD) development pipeline is white box testing. Static Application Security Testing (SAST), an approach that automatically verifies source code or binaries and offers input on errors and potential vulnerabilities, is frequently discussed in relation to white box testing. White box testing is a type of testing where test data is derived from the logic and code of the programme by looking at its structure. Clear box testing, open box testing, rationale driven testing, path driven



testing, and structural testing are additional terms for glass box testing**.** Fig. 7.3 gives the flow of white box testing

Fig 7.2.3. White Box Testing

7.3 TESTING LEVELS

7.3.1 NON-FUNCTIONAL TESTING

Non-functional testing involves the assessment of the dependability, performance, and accountability of the software from the point of view of user needs. The main purpose of the assessment is measuring performance as opposed to functionality based on specific KPIs. Functional testing is different from non-functional because the latter looks at what resides beyond functionality and is vital since customer satisfaction is difficult to attain.

7.3.1.1 PERFORMANCE TESTING

Performance testing is a tool for testing how a system works under given loads because it does not look for finding bugs in the software. Through using different types of evaluation against specific criteria, it helps developers get diagnostic insights and solve inefficiency issues efficiently.

7.3.1.2 STRESS TESTING

Stress Testing ensures the robustness and adaptability of software platforms working in very stressful

environments with high system load. The main purpose of it is to evaluate the software robustness and error handling competences, as a result software stability even in hostile conditions.

7.3.1.3 SECURITY TESTING

Security Testing offers solutions for system vulnerabilities to keep data and properties secure from any intrusions.It identifies vulnerabilities and gaps in security and shields information or company's reputation against unlawful access or cyberattacks.

7.3.1.4 PORTABILITY TESTING

The portability testing techniques provide information about the appropriateness of a software or app for the transfer between diffeent systems be it hardware, software, or operating environments. It defines how gracefully an application can be embedded into other systems unconditionally and aligned the non-functional requirement of portability.

7.3.1.5 USABILITY TESTING

Usability Testing, in short, User Experience (UX) Testing, is a process of evaluating the user-friendliness and handy nature of the software applications through the user's point of view or through the end-users. This approach keeps target end-users at the center of the development process, consequently discovering usability errors early in the development, which also sheds some light on the customers' expectations during the initial design phase of the software development life cycle (SDLC).

7.3.2 FUNCTIONAL TESTING

Functional Testing is used to ensure that an application's functionality is suitable for the requirements.

Black-box testing gives the assurance that every function operates as it must, thus the application's functional nature is confirmed. Functional Testing is an indispensable tool for ensuring that an application accomplishes its intended purposes.

White Box Testing: In White Box Testing, part of Functional Testing, we examine the inner structure and the logic of the software, make sure every function works as it should. Testers review code paths and logic flows substantiation that the software processes correctly.

Black Box Testing: Besides Functional Testing applies also Black Box Testing and the software is checked from the view of a user. Testers pay attention to inputs and outputs, denying access to the internal processes and only making sure that the software acts according to given conditions.

7.3.2.1 INTEGRATION TESTING

Integration testing is the process of evaluating the modules or components after they have beenintegrated to ensure that they function as intended, that is, to test the modules that function wellseparately and do not have problems when combined. Testing the interfaces between the units/modules is the primary purpose or objective of thistesting. First, the different modules are examined separately. Following unit testing, the modules are combined one at a time until all of them are integrated. This allows us to verify whether the requirements are implemented appropriately and examine combinational behavior.

7.3.2.2 REGRESSION TESTING

Regression testing is a kind of software testing used to verify that recent changes to the programor code have not negatively impacted already-existing features. Regression testing is just selecting some or all of the previously run test cases and running them again to make sure the features still function as intended.
The purpose of this testing is to make sure that updates to thecode don't negatively impact already-existing features.\

7.4 TEST CASES:

7.4.1 BLACKBOX TESTING:

Test Case ID	Test Scenario	Input	Expected Output	Pass/Fail
BB-01	Upload a clean real video	Real video file	"Real" label with high confidence	Pass
BB-02	Upload a clean fake video	Deepfake video file	"Fake" label with high confidence	Pass
BB-03	Upload a video with poor resolution	Low-res video (144p)	System still classifies correctly	Pass
BB-04	Upload a non-video file	.txt file	Show error "Unsupported file format"	Fail
BB-05	Upload a very short video (1-2 frames)	1-second video	Either "Insufficient data" or valid classification	Pass
BB-06	Upload video with no face	Video showing only scenery	Display error or "Real" with low confidence	Pass

7.4.2 WHITE BOX TESTING:

Test Case ID	Test Scenario	Test Detail	Expected Result	Actual Output / Pass/Fail
WB-01	Unit test for extract_frames()	Input a 10-sec video	Returns correct number of frames	Pass
WB-02	Unit test for preprocess_frame()	Input a raw frame	Output is (224,224,3), normalized	Pass
WB-03	Test sigmoid output range	Manually pass dummy frame	Output $\in$ (0,1)	Pass
WB-04	Test prediction consistency	Feed same frame 10×	Consistent outputs	Pass
WB-05	Model loading test	Call load_model()	Model loads successfully	Pass
WB-06	Exception handling on invalid input	Empty input to predict()	Graceful error handling	Fail
WB-07	Integration test: full pipeline	Real video through UI → model	Full detection without errors	Pass

7.4.3 GREY BOX TESTING:

Test Case ID	Test Scenario	Input	Expected Behavior (based on model design)	Actual Output / Pass/Fail
GB-01	Test on synthetic frames with known GAN artifacts	GAN-generated face video	Label as "Fake" due to learned features	Pass
GB-02	Check frame extraction consistency	Real video with 30 frames	Returns 30 preprocessed frames	Pass
GB-03	Model threshold boundary test	Prediction ~0.49/0.51	Classification changes around threshold (0.5)	Pass
GB-04	Check prediction consistency across frames	Similar consecutive frames	Similar confidence levels	Pass
GB-05	Partial occlusion in face	Partially hidden face video	Classify based on visible features	Pass
GB-06	Compressed/degraded video input	Compressed deepfake video	Slight confidence drop but correct label	Fail

CHAPTER 8

OUTPUT SCREENS/ RESULTS

8.1 OUTPUT SCREENS

Dashboard Page: The dashboard serves due to the fact the treasured hub of the threat modeling tool, supplying clients with a streamlined assessment of their ongoing duties, cutting-edge-day interest, and risk tests. This intuitive layout promotes situational focus and quick get right of entry to to critical information, empowering customers to control chance models efficaciously. It exemplifies the device's self-discipline to person-focused design and operational transparency.

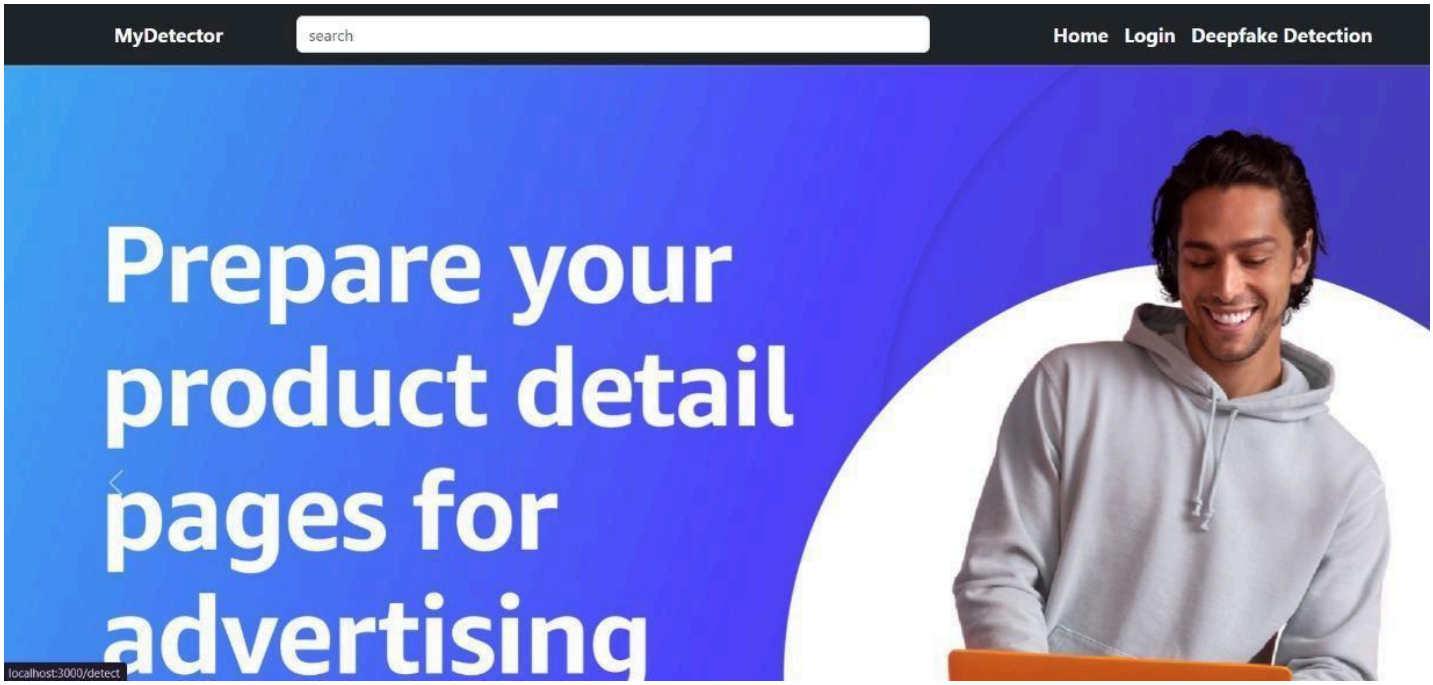


Fig 8.1.1. Dashboard Screen

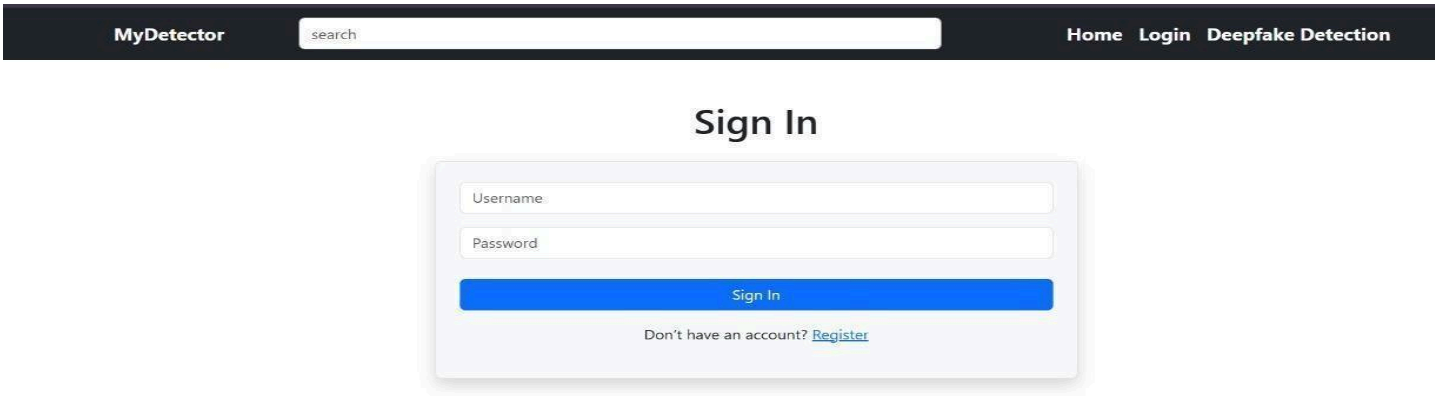


Fig 8.1.2. Sign-in page

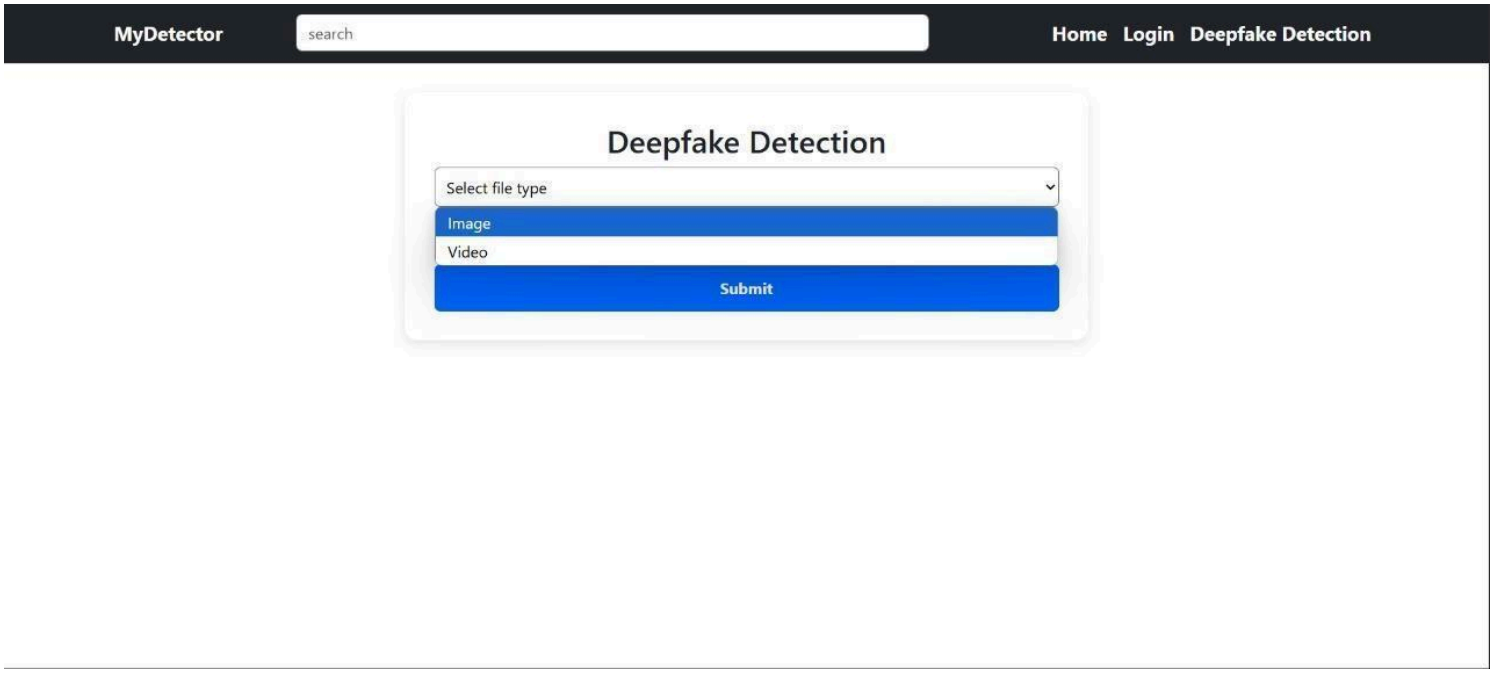


Fig 8.1.3. Deepfake Detection page

Deepfake Detection Page: The creation page enables users to initiate a new threat model by defining system components, data flows, and potential threats. With drag-and-drop capabilities and built-in templates, it simplifies complex modeling tasks. This feature supports secure design practices by encouraging systematic threat identification during the early stages of development.

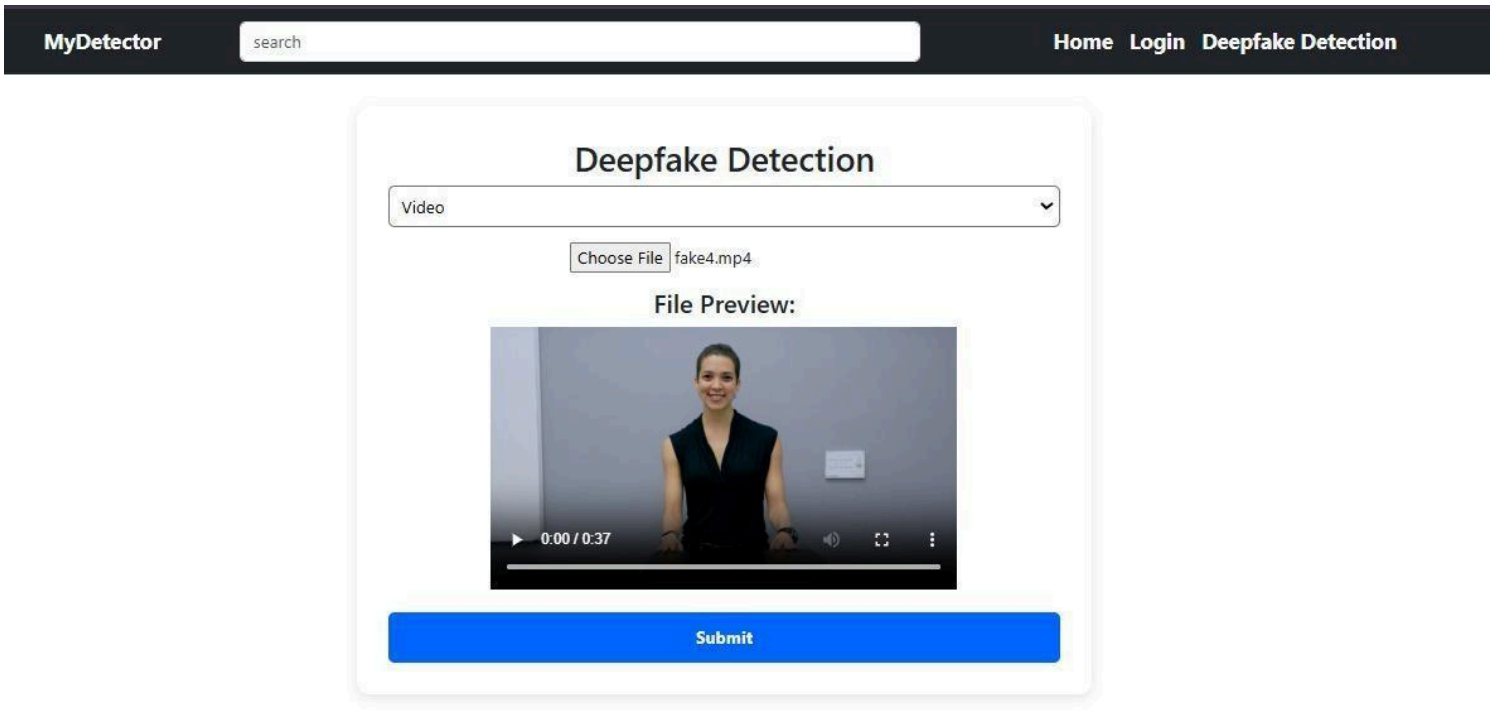


Fig 8.1.4. Uploading video for detection

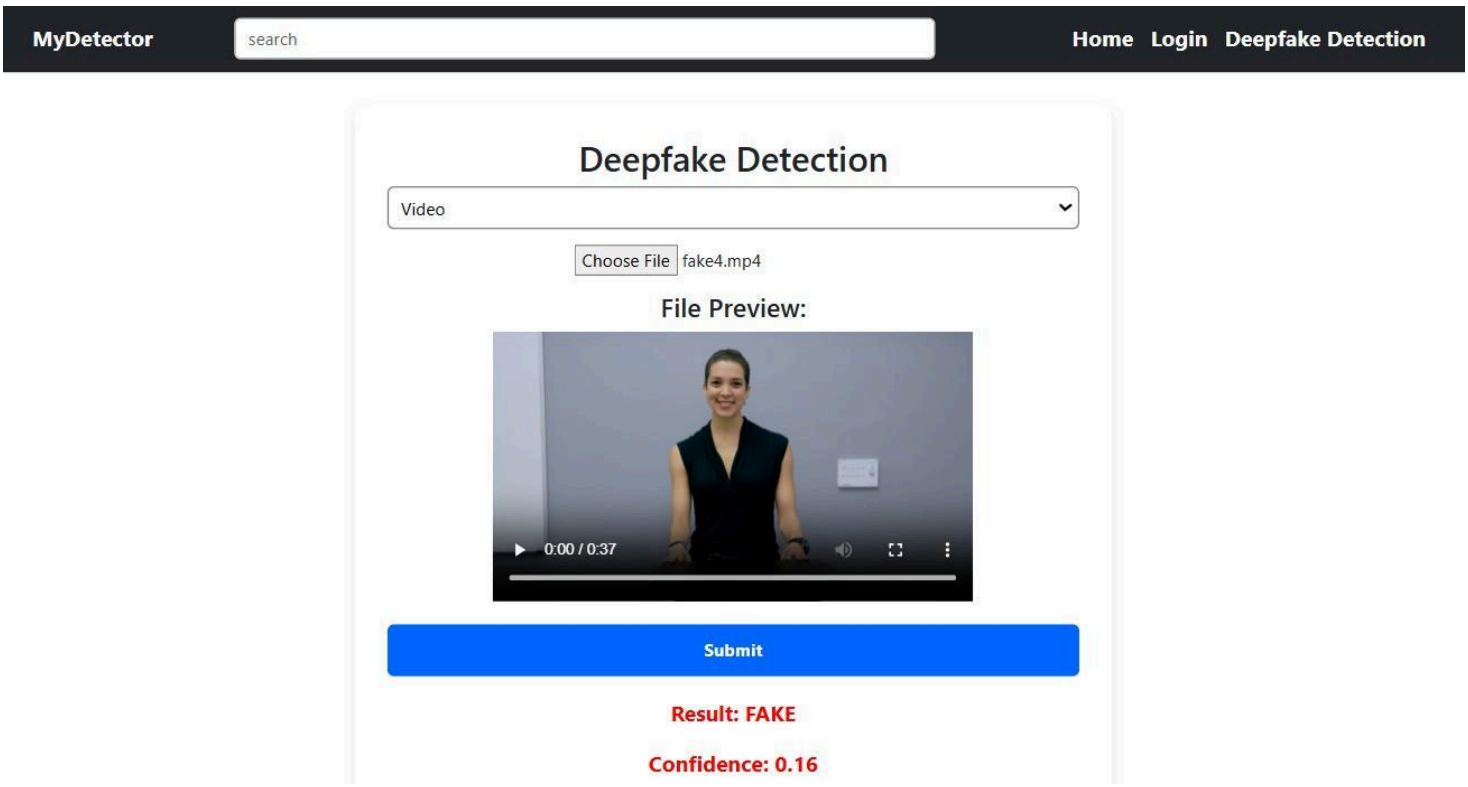


Fig 8.1.5. Detecting video as Fake

8.2 RESULTS

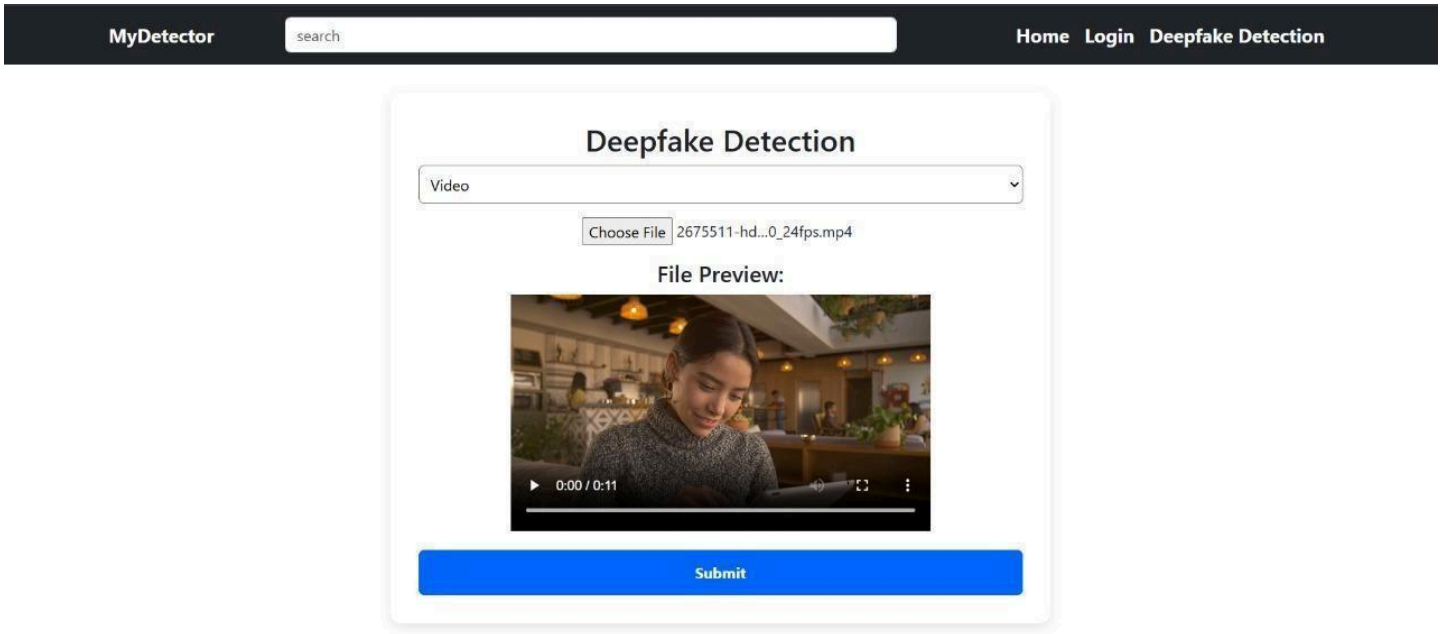


Fig 8.1.6. Uploading video for detection

Report Generation Page: The web page allows the automated advent of whole hazard evaluations based totally on the client's model. It includes sections for recognized threats, mitigation techniques, and compliance tests. This ensures traceability and facilitates choice-making, making the tool an asset for every developers and protection analysts in preserving device integrity.

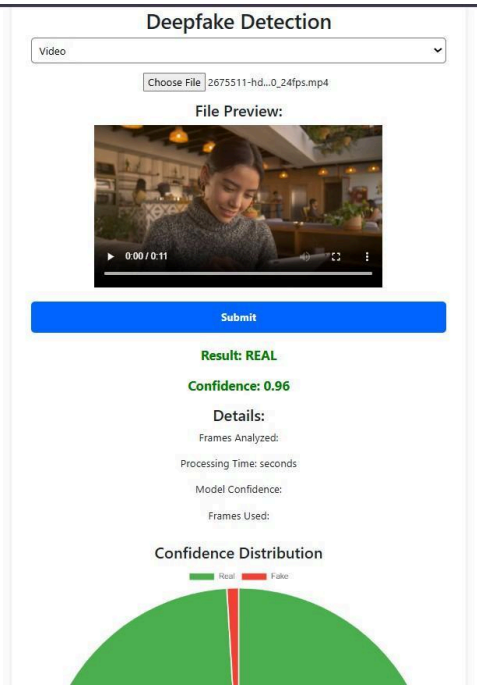


Fig 8.1.7. Report Generation Page

CHAPTER 9
CONCLUSIONS AND FURTHER WORK

9.1 CONCLUSION

The increasing realism and accessibility of deepfake era present a critical hazard to digital authenticity, public believe, and private protection. As synthetic media continues to conform, the significance of powerful deepfake detection systems becomes extra urgent than ever. on this challenge, we addressed this undertaking by using growing a sturdy deep studying- primarily based answer that could accurately distinguish among actual and manipulated motion pictures.

Our method combines the spatial function extraction abilties of the ResNeXt Convolutional Neural community with the temporal sequence modeling power of long brief-term reminiscence (LSTM) networks. This twin-model architecture allows the gadget to analyze each person frames and body-to-body dynamics, making it extra reliable than models that rely solely on one element.

To ensure the version's effectiveness and generalization, we trained it on a massive and various dataset compiled from FaceForensics++, superstar-DF, and the Deepfake Detection venture. The device finished high accuracy and demonstrated robust ability in detecting unique styles of deepfake videos, consisting of the ones generated using advanced techniques.

Further to the detection version, we evolved a person-friendly internet utility that lets in users to add movies and acquire actual-time predictions at the side of self belief scores. This realistic implementation showcases how artificial intelligence can be used to counter the very threats it allows create.

Newly, this challenge proves that deep studying, whilst implemented thoughtfully, can play a essential function within the fight towards misinformation and artificial media manipulation. Our gadget lays a strong foundation for destiny art work on this location, with room to increase its abilties in addition.

9.2 SUMMARY

In current years, the proliferation of deepfake movies—synthetically generated media created the use of advanced synthetic intelligence—has raised severe worries regarding incorrect information, virtual deception, and on-line protection. these movies, specifically the ones utilising practical face-swapping techniques, have come to be an increasing number of convincing, making it tough for human observers and conventional detection methods to differentiate among proper and manipulated content. Addressing this growing risk necessitates the improvement of robust, sensible detection structures that could analyze visible and temporal patterns in video facts.

This venture presents a deep gaining knowledge of-based totally approach for the reliable detection of deepfake videos, combining both spatial characteristic extraction and temporal collection evaluation. The middle structure of the proposed device is a hybrid neural community model that utilizes:

A pre-skilled ResNeXt convolutional neural community (CNN) to extract high-quality-grained, high-level spatial functions from every video body. those functions capture visual cues including facial structure, texture inconsistencies, and lights anomalies which can be regularly subtly altered in deepfake content.

A long quick-term reminiscence (LSTM) network that methods the collection of extracted body-level functions to study temporal inconsistencies, which include unnatural transitions in facial expressions, abnormal eye blinking patterns, or mismatches in head actions that may not be visible in individual frames however emerge as obvious over the years.

To make certain the version's generalizability and robustness, a complete and various education dataset became curated with the aid of integrating more than one publicly to be had deepfake datasets, consisting of:

FaceForensics++ – a benchmark dataset widely used for deepfake research, containing

manipulated films with varying stages of compression and artifacts,movie star-DF (v2) – offering 400 deepfake videos that reflect real-global manipulation strategies,

DFDC (Deepfake Detection project) – presenting a big and diverse dataset of both real and manipulated films created with exclusive generation strategies.

via combining these datasets, we created a balanced and representative education set that exposes the version to a extensive range of deepfake types and actual-world situations, thereby enhancing its detection accuracy and resilience to novel manipulations.

Similarly to the deep getting to know version, we evolved an internet-primarily based application using Django, designed to make deepfake detection available to a broader target market. The software permits customers to upload video documents directly via a browser interface, where the system tactics the video and returns a prediction label ("actual" or "fake") together with a confidence rating indicating the version's actuality. The platform is intended for researchers, content reviewers, reporters, and most of the people who require a quick, easy-to-use device for video authenticity verification.

This undertaking now not simplest contributes to the field of virtual forensics but also provides a realistic, consumer-oriented method to fight the malicious use of deepfake era. by means of leveraging spatial-temporal inconsistencies which are often invisible to the bare eye, our system gives a dependable technique for identifying synthetic content material and restoring trust in visible media.

9.3 FURTHER WORK

While this project has correctly tested a dependable technique for detecting deepfake motion pictures the use of deep gaining knowledge of, there are several regions that provide capacity for improvement and future improvement:

- **Real time Detection:** presently, the version procedures movies after they are uploaded. future enhancements should encompass real-time deepfake detection in live

video streams, making the gadget suitable for integration into video conferencing apps or stay broadcasting structures.

- **Full-body Deepfake Detection**: the existing device focuses mostly on facial deepfakes. increasing the model to locate complete-frame manipulations—including altered body actions or gestures—could growth its effectiveness in opposition to greater advanced synthetic media.

- **Light-weight model for cell and part gadgets**: to enhance accessibility, a compressed model of the version can be developed for deployment on cellular gadgets or facet computing structures, allowing offline and coffee-latency detection.

- **Integration with Social Media systems:** collaborating with structures like facebook, Instagram, or WhatsApp may want to help enforce pre-upload scanning tools to mechanically flag or block deepfake content before it spreads

- **Improved Dataset collection:** The excellent and diversity of training records significantly impact detection accuracy. within the future, we aim to encompass extra datasets that include numerous demographics, lighting situations, languages, and deepfake strategies to further generalize the model

- **Browser plugin development:** growing a browser extension that robotically tests the authenticity of movies in actual time even as browsing could provide users with immediate signals on potential deepfakes.

CHAPTER 10

REFERENCES

References

[1]. Anonymous. (2023). *Deepfake Detection: A Comparative Analysis*. arXiv preprint arXiv:2308.03471. Retrieved from https://arxiv.org/abs/2308.03471

[2]. Anonymous. (2023). *DeepfakeBench: A Comprehensive Benchmark of Deepfake Detection*. arXiv preprint arXiv:2307.01426. Retrieved from https://arxiv.org/abs/2307.01426

[3]. Anonymous. (2023). *Deepfake Detection with Deep Learning: CNNs vs. Transformers*. arXiv preprint arXiv:2304.03698. Retrieved from https://arxiv.org/abs/2304.03698

[4]. Oravec, J., Macháč, D., & Oravec, M. (2023). *Comparative Analysis of Deepfake Detection Models on Diverse GAN-Generated Images*. International Journal of Electrical and Computer Engineering Systems (IJECES), 14(1), 1-10. Retrieved from https://ijeces.ferit.hr/index.php/ijeces/article/view/3417

[5]. Anonymous. (2022). *Deepfake Detection: A Comprehensive Survey from the Reliability Perspective*. arXiv preprint arXiv:2211.10881. Retrieved from https://web3.arxiv.org/abs/2211.10881

[6]. Wang, T., Liao, X., Chow, K. P., Lin, X., & Wang, Y. (2022). *Deepfake Detection: A Comprehensive Survey from the Reliability Perspective*. arXiv preprint arXiv:2211.10881. Retrieved from https://web3.arxiv.org/abs/2211.10881

[7]. Abirami, M. S., & Vijayakumar, V. (2024). *Comparative Study of Deep Learning Techniques for DeepFake Video Detection*. Journal of Information and Communication

Technology.Retrievedfrom
https://www.sciencedirect.com/science/article/pii/S2405959524001218

[8]. Gupta, G., Raja, K., Gupta, M., Jan, T., Whiteside, S. T., & Prasad, M. (2024). *A Comprehensive Review of DeepFake Detection Using Advanced Machine Learning and Fusion Methods*. Electronics, 13(1), 95. Retrieved from https://www.mdpi.com/2079-9292/13/1/95

[9]. Ashani, Z. N., Che Ilias, I. S., Ng, K. Y., Ariffin, M. R. K., Jarno, A. D., & Zamri, N. Z. (2024). *Comparative Analysis of Deepfake Image Detection Method Using VGG16, VGG19, and ResNet50*. Applied Sciences and Engineering Technology Journal, 4(1), 45–52.Retrievedfrom https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/article/view/4925+

Plagiarism report

Report_RA0

ORIGINALITY REPORT

11%	6%	2%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.coursehero.com Internet Source	1%
2	Submitted to Griffith College Dublin Student Paper	1%
3	Submitted to South Birmingham College Student Paper	1%
4	Submitted to Taylor's Education Group Student Paper	1%
5	Submitted to Higher Education Commission Pakistan Student Paper	1%
6	Submitted to University of Wales Institute, Cardiff Student Paper	1%
7	core.ac.uk Internet Source	<1%
8	Submitted to The University of Texas at Arlington Student Paper	<1%
9	Submitted to University of Hertfordshire Student Paper	<1%
10	www.mdpi.com Internet Source	<1%
11	p.pdfhall.com Internet Source	<1%
12	Submitted to Manipal University Student Paper	<1%
13	Submitted to Vishwashanti Gurukul Student Paper	<1%

14	Submitted to CSU, San Jose State University Student Paper	<1 %
15	Ashfaque Ahmed. "Software Testing as a Service", Auerbach Publications, 2019 Publication	<1 %
16	john.cs.olemiss.edu Internet Source	<1 %
17	www.browserstack.com Internet Source	<1 %
18	de.slideshare.net Internet Source	<1 %
19	www.iuemag.com Internet Source	<1 %
20	publications.eai.eu Internet Source	<1 %
21	Submitted to Gitam University Student Paper	<1 %
22	Submitted to University of Greenwich Student Paper	<1 %
23	Mohamed E. Fayad, Huascar A. Sanchez, Srikanth G.K. Hegde, Anshu Basia, Ashka Vakil. "Software Patterns, Knowledge Maps, and Domain Analysis", Auerbach Publications, 2019 Publication	<1 %
24	dspace.uiu.ac.bd Internet Source	<1 %
25	ojs.ijemd.com Internet Source	<1 %
26	thequickadvisor.com Internet Source	<1 %
27	Gunel, Can. "Effectiveness of Using Clustering for Test Case Prioritization.", Izmir Institute of	<1 %

28	Lingfeng Wang, K.C. Tan. "Software testing for safety critical applications", IEEE Instrumentation & Measurement Magazine, 2005 Publication	<1 %
29	arxiv.org Internet Source	<1 %
30	irigs.iiu.edu.pk:64447 Internet Source	<1 %
31	ore.exeter.ac.uk Internet Source	<1 %
32	www.bartleby.com Internet Source	<1 %
33	www.ranorex.com Internet Source	<1 %
34	www.researchgate.net Internet Source	<1 %
35	Pankaj Jalote. "A Concise Introduction to Software Engineering", Springer Science and Business Media LLC, 2025 Publication	<1 %

Exclude quotes	Off	Exclude matches	< 2 words
Exclude bibliography	On		

AI Check

*% detected as AI

AI detection includes the possibility of false positives. Although some text in this submission is likely AI generated, scores below the 20% threshold are not surfaced because they have a higher likelihood of false positives.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Disclaimer
Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (it may misidentify writing that is likely AI generated as AI generated and AI paraphrased or likely AI generated and AI paraphrased writing as only AI generated) so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.

SHOW AND TELL