DEEPFAKE DETECTION FOR HUMAN FACE IMAGES AND VIDEOS

PROJECT REPORT

submitted by

ASMI FAISEL

PTA20CS018

JOSIN JOSE

PTA20CS035

NANDHANA A

PTA20CS043

V H PRANAV

PTA20CS062

to

the APJ Abdul Kalam Technological University

in partial fulfilment of the requirements for the award of the Degree

of

Bachelor of Technology

in

Computer Science and Engineering

Department of Computer Science & Engineering

College of Engineering Kallooppara

Pathanamthitta

May 2024

DECLARATION

We undersigned hereby declare that the project report **DEEPFAKE DETECTION FOR HUMAN FACE IMAGES AND VIDEOS** submitted for partial fulfilment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under supervision of Assistant Professor, **Ms. Nimitha Mary Mohan.** This submission represents our ideas in our own words and ideas or words of others have been included where we have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the university and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other university.

Place Kallooppara

Date: 29/04/2024

ASMI FAISEL

JOSIN JOSE

NANDHANA A

V H PRANAV

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

COLLEGE OF ENGINEERING KALLOOPPARA

PATHANAMTHITTA- 689603

CERTIFICATE

This is to certify that the project design report entitled ' **DEEPFAKE DETECTION FOR HUMAN FACE IMAGES AND VIDEOS** ' submitted by **Asmi Faisel** (PTA20CS018), **Josin Jose** (PTA20CS035), **Nandhana A** (PTA20CS043), **V H Pranav** (PTA20CS062) in partial fulfillment with the requirements of the award of the Degree of Bachelor of Technology in Computer Science and Engineering of APJ Abdul KalamTechnological University is a bonafide work carried out by them under our guidance and supervision. The report in any form has not been submitted to any other University or Institutefor any purpose.

Co-coordinator	**Guide**	**Head of Department & Coordinator**
Ms. Nimitha Mary Mohan Assistant Professor, Department of Computer Science & Engineering	**Ms. Nimitha Mary Mohan** Assistant Professor, Department of Computer Science & Engineering	**Dr. Renu George** Assistant Professor, Department of Computer Science & Engineering

ACKNOWLEDGEMENT

We take this opportunity to express our deepest sense of gratitude and sincere thanks to everyone who helped us to complete this work successfully. We express our sincere thanks to **Dr. Nisha Kuruvilla** (Principal) for the constant support and help. We also thank **Dr. Renu George** (Head of Department) Computer Science and Engineering, College of Engineering Kallooppara for providing us with all the necessary facilities and support. We would like to express our sincere gratitude to **Ms. Nimitha Mary Mohan** department of Computer Science and Engineering, College of Engineering Kallooppara for their support and cooperation. We would like to place on record our sincere gratitude to our project guide **Ms. Nimitha Mary Mohan**, Assistant Professor, Computer Science and Engineering, College of Engineering Kallooppara for the guidance and mentorship throughout the course. Finally, we thank our family, and friends who contributed to the successful fulfilment of this project work.

ASMI FAISEL
JOSIN JOSE
NANDHANA A
V H PRANAV

ABSTRACT

The growing computation power has made the deep learning algorithms so powerful that creating an indistinguishable human synthesized video popularly called as deepfakes have became very simple. The deepfake technology has been used to create realistic but fake videos and images of people, which can be used for malicious purposes such as spreading false information or defaming individuals. The goal of deepfake detection is to develop reliable methods for identifying and removing these manipulated media from circulation. Scenarios where these realistic face swapped deep fakes are used to create political distress, fake terrorism events, revenge-porn, blackmail peoples are easily envisioned. In response to the growing concern over deep fake technology's impact on media credibility, this project presents a comprehensive DFD system with a user-friendly graphical interface. Leveraging the Faceforensic++ dataset ,the system employs pretrained face detection models for precise face segmentation and custom deep learning models to extract distinctive features from segmented regions and bounding boxes. A Classification Network is trained to discern authentic from manipulated faces, achieving robust detection accuracy. Implemented within a Python Tkinter-based GUI, the system offers a practical tool for users to analyze and verify multimedia content, contributing to the mitigation of deep fake-related risks in the digital landscape.

Keywords:
DFD(deep fake detection)
Bounding boxes
Faceforensic++ dataset

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

DFD	Deep Fake Detection
CNN	Convolutional Neural Network
AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
CV	Computer Vision
GAN	Generative Adversarial Network

CHAPTER 1
INTRODUCTION

1.1 BACKGROUND

Deepfakes refer to the use of artificial intelligence techniques, particularly deep learning algorithms, to create highly realistic but fabricated media, such as videos, images, or audio, that depict events or individuals that never actually occurred or existed. The proliferation of deepfakes has raised concerns regarding the potential for misinformation, fraud, and the manipulation of public opinion. Deepfake detection involves the development of algorithms and techniques to identify and distinguish between genuine and manipulated content. The primary goal is to detect and mitigate the potential harm caused bydeepfakes by providing tools and methods to verify the authenticity of media. Deepfake detection techniques can be categorized into two main approaches: traditional methods and deep learning-based methods. Traditional methods focus on detecting anomalies, artifacts, and inconsistencies that may arise from the manipulation process. These techniques involve analyzing various visual or audio cues, such asfacial inconsistencies, unnatural movements, inconsistencies in lighting or shadows, and audio inconsistencies.

These methods often rely on domain-specific expertise and manually designed features. Deep learning- based methods have gained significant attention due to their ability to learn and extract features automatically from large datasets. Convolutional neural networks (CNNs) and recurrent neural networks(RNNs) are commonly used architectures for deepfake detection. These models are trained on large datasets of both genuine and manipulated media to learn patterns and features that can differentiate between them. The models can then classify new unseen media as genuine or manipulated based on the learned representations. Deepfake detection is an ongoing and evolving field, as deepfake techniques continue to advance and become more sophisticated. The arms race between deepfake creators and detection algorithms necessitates continuous research and development to improve detection accuracy and robustness. The importance of deepfake detection extends to various domains, including journalism,politics, entertainment, and online platforms.

Detecting and flagging deepfakes can help prevent the spread of misinformation, protect individuals fromreputation damage, and ensure the integrity of visual and audio evidence in critical contexts. As deepfaketechnology progresses, so does the research in deepfake detection. Advancements in deep learning architectures, improved datasets, adversarial approaches, and explainability methods are among the ongoing research efforts. Collaborations between researchers, industry experts, and policymakers are also crucial for developing effective strategies to combat the challenges posed by deepfakes. Overall, deepfake detection plays a vital role in combating the potential negative impacts of manipulated media. By developing and improving detection algorithms and techniques, it is possible to mitigate the risks associated with deepfakes and safeguard the integrity of digital content.

1.2 EXISTING SYSTEM

In the landscape of combating deepfakes, a diverse array of systems and methodologies has emerged, each offering unique strengths and challenges. Forensic analysis stands as a foundational pillar, involving meticulous examination of digital media to unearth telltale signs of manipulation. This approach scrutinizes images and videos for inconsistencies in lighting, shadows, or facial features, as well as anomalies that deviate from expected behaviors. However, while forensic analysis provides valuable insights, its reliance on human expertise and manual inspection limits scalability and efficiency.

In contrast, machine learning (ML) techniques have garnered significant attention for their potential to automate deepfake detection at scale. ML algorithms, particularly deep neural networks, are trained on extensive datasets comprising authentic and manipulated media to discern subtle patterns indicative of deepfake fabrication. By learning from diverse examples, these algorithms can generalize and identify nuanced cues that elude human perception. Nevertheless, challenges persist in ensuring the robustness and generalizability of ML models across different manipulation techniques, as well as addressing the adversarial strategies employed by deepfake creators to evade detection.

Another avenue of exploration lies in blockchain technology, which offers immutable ledger systems for verifying the authenticity and provenance of digital content. By timestamping media files and recording transaction histories on decentralized networks, blockchain platforms aim to establish a tamper-proof chain of custody, thereby enhancing trust and accountability in the digital domain. However, while blockchain holds promise for authenticating genuine content, its implementation complexities, scalability concerns, and user adoption hurdles pose significant

obstacles to widespread adoption.

1.3 PROBLEM STATEMENT

Deepfake had become popular due to the massive availability of images and videos in social contents. This is particularly important nowadays because the tools for making deepfakes are becoming more accessible, and social media sites will easily allowing people to distribute and share such fake contents. The goal of deepfake detection is to develop reliable methods for identifying and removing these manipulated media from circulation. Scenarios where these realistic face swapped deep fakes are used to create political distress, fake terrorism events, revenge-porn, blackmail peoples are easily envisioned.

1.4 OBJECTIVES

The objective of a deepfake detector is to identify and distinguish between genuine and manipulated media content created using deep learning techniques, such as deepfake algorithms. Deepfakes are synthetic media, including images, videos, or audio, that have been artificially altered or created using advanced machine learning models. The primary objectives of a deepfake detector can be summarized asfollows:

- Identification of Deepfakes: The primary objective is to accurately identify and detect deepfake content. Deepfake detection algorithms analyze various features, patterns, and artifacts within the media to determine if it has been manipulated or is authentic.
- Differentiation from Genuine Content: A deepfake detector aims to distinguish between deepfake content and genuine media. It compares the characteristics and inconsistencies within the analyzed content to known patterns of manipulation and abnormal behaviors.
- Advancement of Countermeasures: Deepfake detection contributes to the ongoing research and development of countermeasures against deepfake technology. By studying the methods used in creating deepfakes, detectors can identify weaknesses and vulnerabilities that can be addressed through improvedalgorithms and techniques.
- Promoting Trust and Authenticity: Deepfake detection aims to maintain trust and authenticity in digital media. By providing a means to identify and verify manipulated content, detectors help users make informed decisions and avoid potential deception or manipulation.

1.5 SCOPE

The future scope of deepfake detection involves several areas of research and development:
Enhanced deep learning models: Researchers are exploring more advanced deep learning architectures to improve detection accuracy. Techniques such as recurrent neural networks, attention mechanisms, and generative models may play a crucial role in developing more robust deepfake detection algorithms. 1.Adversarial approaches: Adversarial machine learning techniques can be employed to create robust detection algorithms that can withstand adversarial attacks. By training models against specifically designed deepfake attacks, it is possible to improve their resilience to manipulation.

2. Dataset improvements: The availability of large-scale, diverse and wellcurated datasets is crucial for training effective deepfake detection models. Efforts should continue to expand and improve datasets to encompass a wide range of deepfake variations and ensure better generalization.
3. Explainability and interpretability: Developing methods to explain the decisions made by deepfake detection algorithms is crucial for building trust and understanding their limitations. Research in interpretable AI can provide insights into the decision-making process of detection models.
4. Real-time and scalable solutions: As deepfake content can spread rapidly on social media platforms, developing real-time and scalable deepfake detection solutions is important for timely identification andmitigation of potential risks.
5. Collaboration and standardization: Collaboration between researchers, industry experts, and policymakers is necessary to address the challenges associated with deepfakes effectively. Standardization of evaluation metrics, benchmark datasets, and best practices will help in comparing andimproving deepfake detection methods.

In summary, deepfake detection is an ongoing and evolving field with significant challenges and opportunities. Continued research, innovation, and collaboration are vital to stay ahead of deepfake creators and protect individuals, organizations, and society from the potential harms of manipulated media.

CHAPTER 2
LITERATURE REVIEW

Literature was reviewed from various sources, like from research papers, publications books, existing bibliographic information, and recommendations by the project panel. These research papers has provided us sufficient amount of data for the survey.

2.1 DETECTING GAN GENERATED FAKE IMAGES USING CO-OCCURRENCE MATRICES

By Lakshmanan Nataraj; Mayachitra Inc., Santa Barbara, California, USA. The advent of Generative Adversarial Networks (GANs) has brought about completely novel ways of transforming and manipulating pixels in digital images. GAN based techniques such as Image-to-Image translations, DeepFakes, and other automated methods have become increasingly popular in creating fake images. This paper, proposes a novel approach to detect GAN generated fake images using a combination of co-occurrence matrices and deep learning [6]. This extract co- occurrence matrices on three color channels in the pixel domain and train a model using a deep convolutional neural network (CNN) framework. Experimental results on two diverse and challenging GAN datasets comprising more than 56,000 images based on unpaired image-to-image translations and facial attributes/expressions show that the approach is promising and achieves more than 99% classification accuracy in both datasets. Further, the approach also generalizes well and achieves good results when trained on one dataset and tested on the other. Recent advances in Machine Learning and Artificial Intelligence have made it tremendously easy to create and synthesize digital manipulations in images and videos [1].

2.2 DEEPFAKE VIDEO DETECTION USING RECURRENT NEURAL NETWORKS

By David Guera Edward J. Delp Video and Image Processing Laboratory (VIPER), Purdue University. This paper proposes a temporal-aware pipeline to automatically detect deepfake videos. This 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 [9]. Then evaluate the method against a large set of

deepfake videos collected from multiple video websites. Shows how the system can achieve competitive results in this task while using a simple architecture. The main contributions of this work are summarized as follows. First, a two-stage analysis composed of a CNN to extract features at the frame level followed by a temporally-aware RNN network to capture temporal inconsistencies between frames introduced by the face-swapping process. Second, have used a collection of 600 videos to evaluate the proposed method, with half of the videos being deepfakes collected from multiple video hosting websites. Third, show experimentally the effectiveness of the described approach, which allows use to detect if a suspect video is a deepfake manipulation with 94% more accuracy than a random detector baseline in a balanced setting[2].

2.3 EXPOSING DEEP FAKES USING INCONSISTENT HEAD POSES

By Xin Yang , Yuezun Li and Siwei Lyu University at Albany, State University of New York, USA. In this paper, we propose a new method to expose AI generated fake face images or videos (commonly known as the Deep Fakes). This method is based on the observations that Deep Fakes are created by splicing synthesized face region into the original image, and in doing so, introducing errors that can be revealed when 3D head poses are estimated from the face images. This perform experiments to demonstrate this phenomenon and further develop a classification method based on this cue. Using features based on this cue, an SVM classifier is evaluated using a set of real face images and Deep Fake. The errors in landmark locations may not be visible directly to human eyes, but can be revealed from head poses (i.e, head orientation and position) estimated from 2D landmarks in the real and faked parts of the face. Specifically, compare head poses estimated using all facial landmarks and those estimated using only the central region. The rationale is that the two estimated head poses will be close for the original face. Experimentally confirm the significant difference in the estimated head poses in Deep Fakes. Then use the difference in estimated head poses as a feature vector to train a simple SVM based classifier to differentiate original and Deep Fakes.Experiments on realistic Deep Fake videos demonstrate the effectiveness of the algorithm[3].

2.4 IN ICTU OCULI: EXPOSING AI CREATED FAKE VIDEOS BY DETECTING EYE BLINKING

By Yuezun Li, Ming-Ching Chang and Siwei Lyu University at Albany, State University of New York, USA. In this work, it describe a new method to expose fake face videos generated with deep neural network models. The method is based on detection of eye blinking in the videos, which is a

physiological signal that is not well presented in the synthesized fake videos. The method is evaluated over benchmarks of eye-blinking detection datasets and shows promising performance on detecting videos generated with DNN based software DeepFake .While traditional media forensic methods based on signal level cues (e.g, sensor noise, CFA interpolation and double JPEG compression), physical level evidence (e.g, lighting condition, shadow and reflection) or semantic level consistencies (e.g, consistency of meta-data) can be applied for this purpose, they are not sufficiently reliable or efficient for detecting DeepFake videos. This situation calls for novel detection techniques. In this work, we describe a method to expose DeepFake videos by detecting the lack of eye blinking of the synthesized faces . Most photos of a person that can be obtained online will not show them with their eyes closed, so this likelihood is even smaller in practice. Therefore, the lack of eye blinking is thus a telltale sign of a DeepFake video[4].

2.5 A NEW DEEP LEARNING-BASED METHODOLOGY FOR VIDEO DEEPFAKE DETECTION USING XGBOOST

By Aya Ismail, Marwa Elpeltagy ,Mervat S. Zaki and Kamal Eldahshan. This paper presents a new deepfake detection method: you only look once–convolutional neural network–extreme gradient boosting (YOLO-CNN-XGBoost). The YOLO face detector is employed to extract the face area from video frames, while the InceptionResNetV2 CNN is utilized to extract features from these faces. These features are fed into the XGBoost that works as a recognizer on the top level of the CNN network. The proposed method achieves 90.62% of an area under the receiver operating characteristic curve (AUC), 90.73% accuracy, 93.53% specificity, 85.39% sensitivity, 85.39% recall, 87.36% precision, and 86.36% F1-measure on the CelebDF-FaceForencics++ (c23) merged dataset. The experimental study confirms the superiority of the presented method as compared to the state-of-the-art methods..This paper introduces a new efficient architecture, YOLO-InceptionResNetV2-XGBoost (YIX), which discovers the visual discrepancies and artifacts within video frames and then judges whether a given video is real or a deepfake. The combination of these three methods is justified as follows: The YOLO detector proves its efficiency in object detection and face recognition systems over the state-of-the-art detectors since it has a good trade-off between performance and speed. Additionally, it is characterized by its ability to produce fewer false positives in the background ,thus improving the detection method performance[5].

.

2.6 FACEFORENSICS++: LEARNING TO DETECT MANIPULATED FACIAL IMAGES

By Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies Matthias. This paper examines the realism of state-of-the-art image manipulations, and how difficult it is to detect them, either automatically or by humans. To standardize the evaluation of detection methods, this proposes an automated benchmark for facial manipulation detection. In particular, the benchmark is based on DeepFakes, Face2Face, FaceSwap and NeuralTextures as prominent representatives for facial manipulations at random compression level and size. The benchmark is publicly available and contains a hidden test set as well as a database of over 1.8 million manipulated images. This dataset is over an order of magnitude larger than comparable, publicly available, forgery datasets. Based on this data, performed a thorough analysis of data- driven forgery detectors. This shows that the use of additional domain specific knowledge improves forgery detection to unprecedented accuracy, even in the presence of strong compression, and clearly outperforms human observer. In this work, it can automatically and reliably detect such manipulations, and thereby outperform human observers by a significant margin. This leverage recent advances in deep learning, in particular, the ability to learn extremely powerful image features with convolutional neural networks (CNNs). It tackles the detection problem by training a neural network in a supervised fashion. To the end, it generates a large-scale dataset of manipulations based on the classical computer graphics-based methods Face2Face and FaceSwap as well as the learning based approaches DeepFakes[6].

CHAPTER 3
SYSTEM ANALYSIS

3.1 EXPECTED SYSTEM REQUIREMENTS

Expected system requirements encompass both hardware and software components to ensure optimal performance and functionality. Hardware requirements include a robust processor such as an Intel Core i5 or i7, a minimum of 8GB RAM for handling large datasets and deep learning operations, a dedicated GPU with CUDA support for accelerated computations, at least 500GB SSD storage for storing datasets and models, and standard input devices like a mouse and keyboard. On the software front, compatibility with operating systems like Windows 10, macOS, or Linux is crucial, along with Python version 3.x for coding and execution. Essential Python libraries such as TensorFlow for deep learning, OpenCV for image processing, NumPy for numerical computations, and tkinter for GUI development are required. Development environments like PyCharm , along with supplementary tools such as Anaconda for managing Python environments, contribute to efficient project execution. These system requirements collectively create an optimal environment for developing and running a deep fake detection system with a user-friendly interface.

3.2 HARDWARE REQUIREMENTS

Table 3.2 Hardware Requirements

Components	Minimum Requirements
Processor	i5 or i7
RAM	8GB
Storage	500GB SSD
Graphics Processing Unit (GPU):	NVIDIA GeForce with CUDA support

3.3 SOFTWARE REQUIREMENTS

Table 3.3 Software Requirements

Specification	Components
Tool	Python IDLE (VSCode, Jupyter Notebook)
Python	version3
Python Libraries	TensorFlow, OpenCV, NumPy, tkinter
Supplementary Tool	Anaconda
Operating system	Windows 10

CHAPTER 4

METHODOLOGY

4.1 PROPOSED SYSTEM

Deepfake detector is trying to provide a sophisticated solution for the problem of determining deepfakes. Even though there exists some tools to solve this problem none of them really takes all myriad factors together to create a solution. Hence deepfake detector will provide a result that'll be the best in an overall perspective and that is the main objective of the product, i.e. to make sure that all factors are considered while determining the deepfakes. It becomes very important to spot the difference between the deepfake and pristine video. We are using AI to fight AI.

The proposed system utilizes a customized deep learning architecture consisting of three interconnected networks: a face network, a context recognition network, and a bounding box network. Each network focuses on extracting specific features from segmented facial regions, context surrounding the face, and bounding boxes around the detected face, respectively. The extracted features are then concatenated and fed into a classification network that determines the authenticity of the input video or image. The project leverages the publicly available Faceforensic++ dataset for training and testing the model. To enhance accuracy, the system implements various preprocessing techniques like face detection, segmentation, and resizing. Additionally, the project utilizes Python Tkinter to develop a user-friendly graphical interface for uploading and analyzing video and image content.

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

4.1.1 Functionality

Deepfake detector is trying to provide a sophisticated solution for the problem of determining deepfakes. Even though there exists some tools to solve this problem none of them really takes all myriad factors together to create a solution. Hence deepfake detector will provide a result that'll be the best in an overall perspective and that is the main objective of the product, i.e. to make sure that all factors are considered while determining the deepfakes

Some major functionalities are

• Media Analysis: Deepfake detectors analyze various types of media such as images, videos to identify signs of manipulation or synthetic generation.

• Feature Extraction: Deepfake detectors often use machine learning algorithms to extract relevant features or patterns from the media being analyzed.

• Model Training: Deepfake detectors are often trained using large datasets of both real and synthetic media to develop machine learning models that can accurately identify deepfakes.

• Classification: Deepfake detectors typically use classification algorithms to determine if a given media is genuine or a deepfake.

The primary functionalities of deepfake detection systems involve a combination of techniques and approaches to scrutinize media content for anomalies or inconsistencies that may indicate manipulation. These functionalities typically revolve around two key aspects: face detection and model training. Face Detection involves loading a pretrained face detection model, performing face detection, then the image is cropped with bounding box technique. Then face and context segmentation is performed and image is resized. The dataset is further split into training and testing set. Model training involves the development and refinement of algorithms and machine learning models that are trained on datasets containing both authentic and manipulated media. These models learn to differentiate between genuine and manipulated content by recognizing subtle discrepancies in patterns, textures, or contextual elements. Training these models involves using deep learning architectures, such as Convolutional Neural Networks (CNNs) optimizing them to detect deepfake characteristics. Continuous learning and improvement of these models are imperative to stay ahead of evolving deepfake creation techniques and to enhance the accuracy and reliability of detection mechanisms.

a) Face Network Model

- Create Customized Face Network model
- Input segmented faces and get its feature vectors

b) Context Recognition Network model

- Create Customized Context Recognition Network Model
- Input segmented contexts and get its feature vectors

c) Bounding Box Network model

- Create Customized Bounding Box Network Model
- Input segmented bounding boxes and get its feature vectors

d)Perform Concatenation of outputs from the 3 Networks

e) Classification Network

-Create Customized Classification Network Model

- Training the model
- Calculate Accuracy of trained model
- Save Trained model

4.1.2 Stakeholders

Stakeholders in deep fake detection include users, developers, and regulatory authorities, each with unique roles and interests in ensuring the accuracy and integrity of digital media.

(i) Developers:

Developers are responsible for creating and maintaining the deepfake detection application. Their primary goal is to build a reliable tool that accurately identifies deepfake content while ensuring scalability, usability, and performance. Challenges include staying updated on evolving deepfake technology and navigating ethical considerations surrounding privacy and freedom of expression.

(ii) Users:

Users of a deepfake detection application come from diverse sectors such as journalism, law enforcement, and social media. They seek a user-friendly and reliable tool to protect themselves from misinformation, fraud, or defamation associated with manipulated media.

(iii) Regulatory Authorities:

Regulatory authorities are concerned with the societal impacts of deepfake technology, including misinformation and fraud. They may seek to regulate deepfake detection apps to ensure compliance with legal and ethical standards. Collaboration between developers and regulators is vital to address technical capabilities, legal compliance, and policy development surrounding deepfake detection.

CHAPTER 5
SYSTEM DESIGN

5.1 ARCHITECTURE

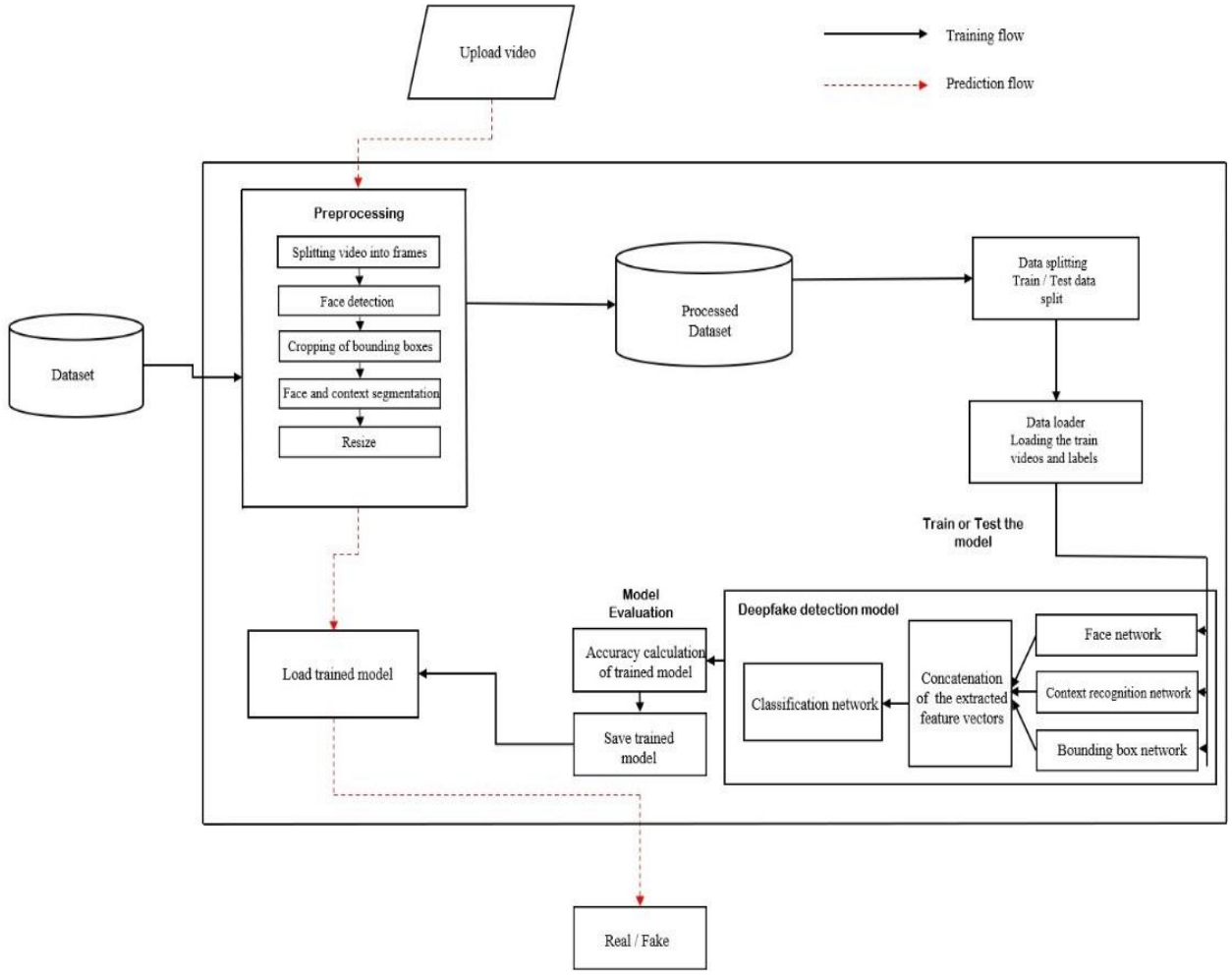


Fig 5.1 Architecture Diagram

The system architecture of the model is shown in the figure. In the development phase, we have taken adataset, preprocessed the dataset and created a new processed dataset which only includes the face cropped videos.

Module 1 : Dataset Gathering

For making the model efficient for real time prediction. We have gathered the data from different available data-sets like FaceForensic++. To avoid the training bias of the model we have considered 50%Real and 50% fake videos.

Module 2 : Preprocessing

In this step, the videos are preprocessed and all the unrequired and noise is removed from videos. Only the required portion of the video i.e face is detected and cropped. The first steps in the preprocessing of the video is to split the video into frames. After splitting the video into frames the face is detected in eachof the frame and the frame is cropped along the face using bounding box. Later the face and context is segmented and resized.

Module 3: Data splitting

The data is split into training and testing sets. The train and test split is a balanced split i.e 50% of the real and 50% of fake videos in each split.

Module 4: Data loading

Data is loaded using a data loader and dataset is loaded and prepared for training or testing. The loader ensures efficient handling of data to train or test the data

Module 5: Deepfake detection model

Three main networks are used feature extraction namely, Face network, context recognition network andbounding box network. The resulted extracted features and concatenated and fed to classification network.

Module 6: Model evaluation

Accuracy of trained model is calculated and the trained model is saved it then load the trained model andis used to detect whether deepfake or real.

5.2 DIAGRAMS

5.2.1 DATA FLOW DIAGRAM

A data flow diagram (DFD) is a graphical or visual representation using a standardized set of symbols and notations to describe a business's operations through data movement. They are often elements of a formal methodology such as Structured Systems Analysis and Design Method (SSADM).

(i) Zeroth Level DFD

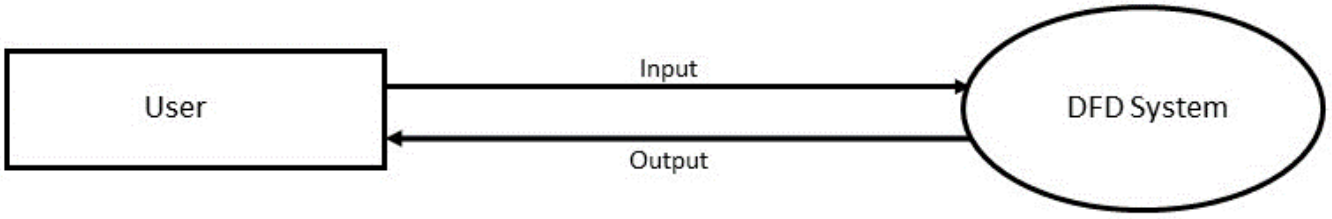


Fig 5.2 Zeroth Level DFD

In this zeroth level DFD, the "Deepfake Detection system" is the central component. It interacts with the user through the input and output.

(ii) First Level DFD

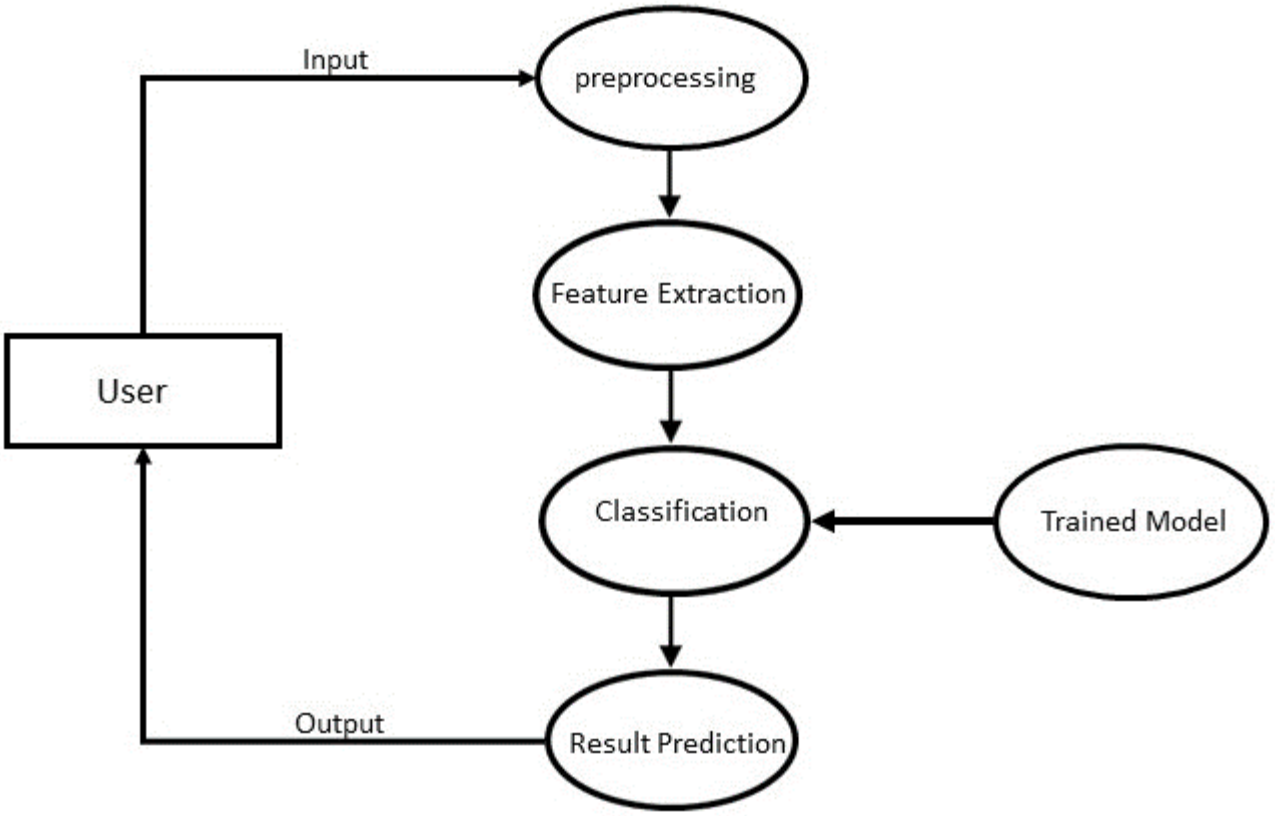


Fig 5.3 First Level DFD

(iii) Second Level DFD

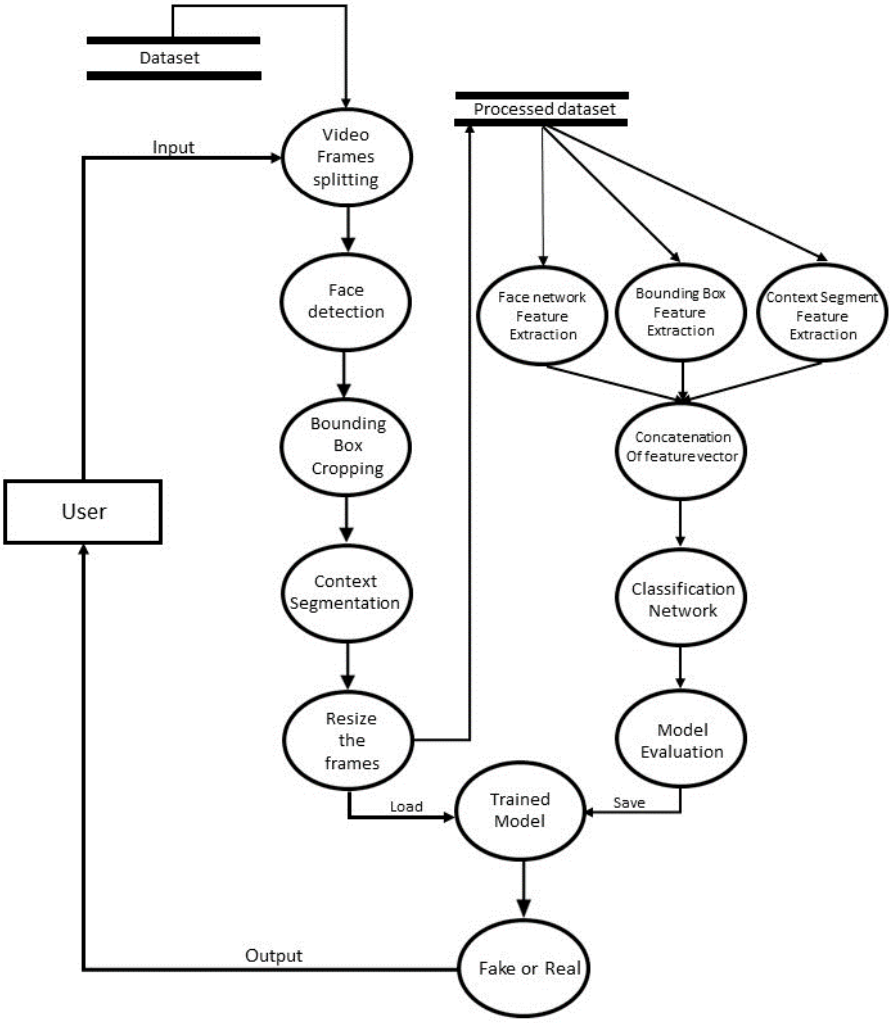


Fig 5.4 Second Level DFD

5.2.2 UML DIAGRAMS

Unified Modeling Language (UML) diagrams serve as a visual representation of a system's structure, behavior, interactions, and architecture. They help in understanding, designing, and communicating complex systems by providing different types of diagrams, such as sequence diagrams, use case diagrams.

(i) Use Case Diagram

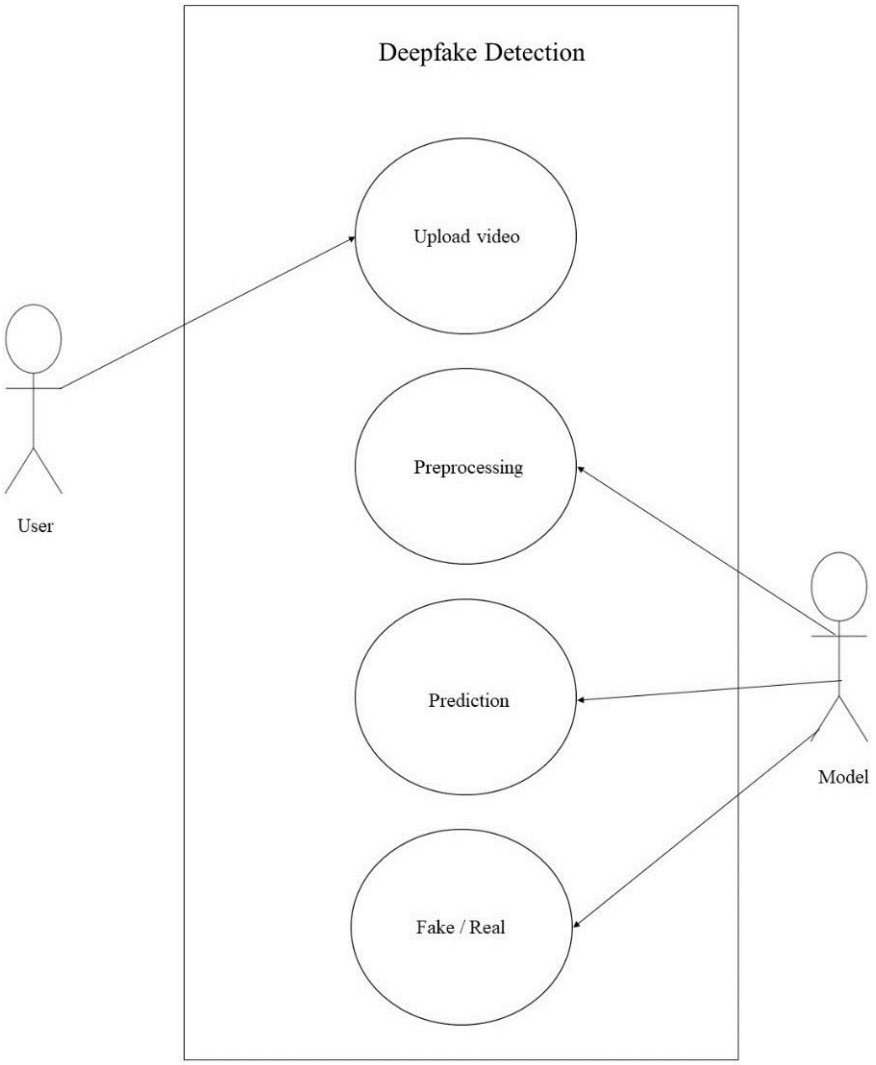


Fig 5.5 Use Case Diagram

(ii) Sequence diagram

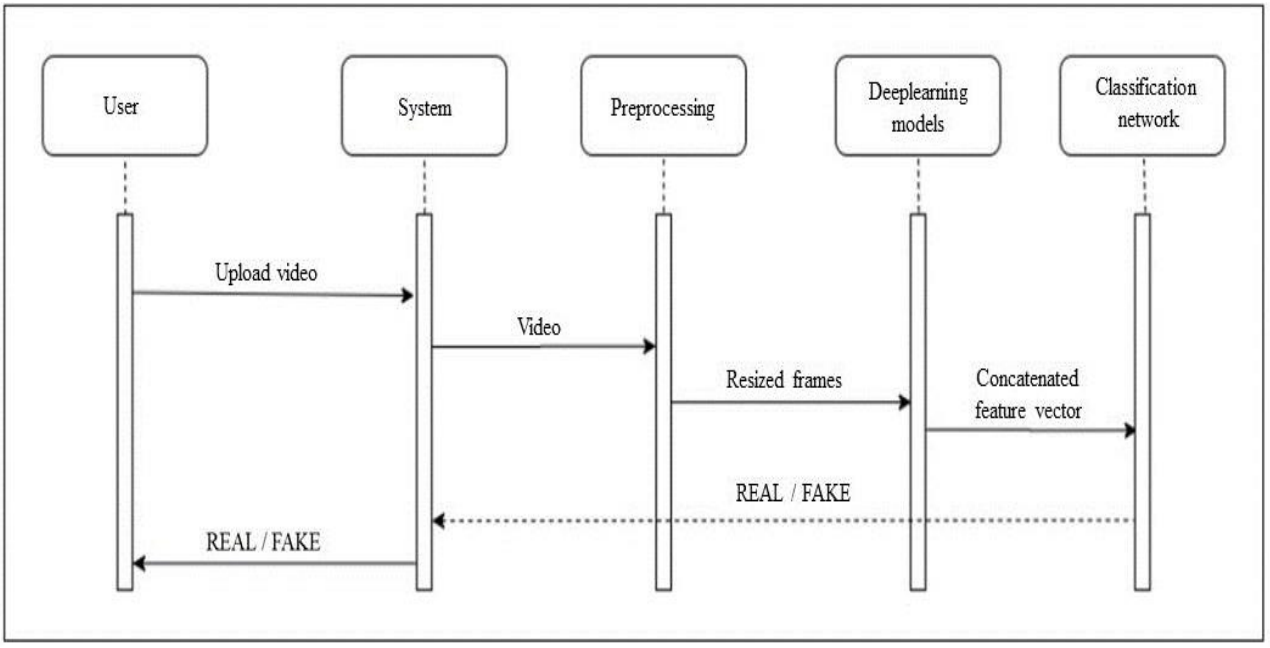


Fig 5.6 Sequence Diagram

(iii) Activity Diagram

Training workflow:

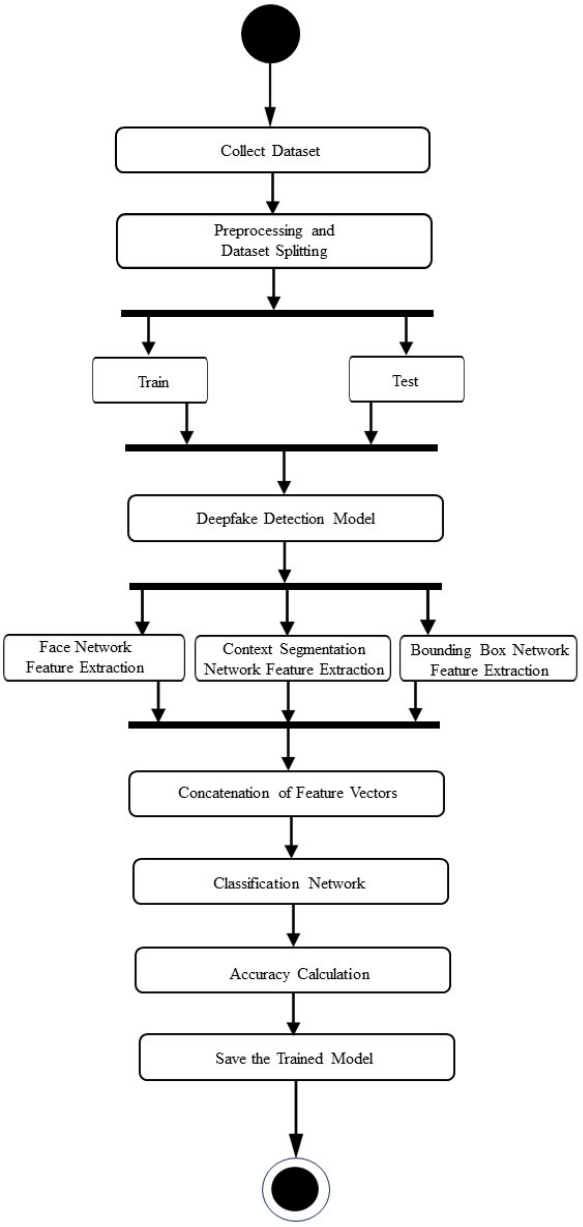


Fig. 5.7 Training workflow diagram

Testing Workflow:

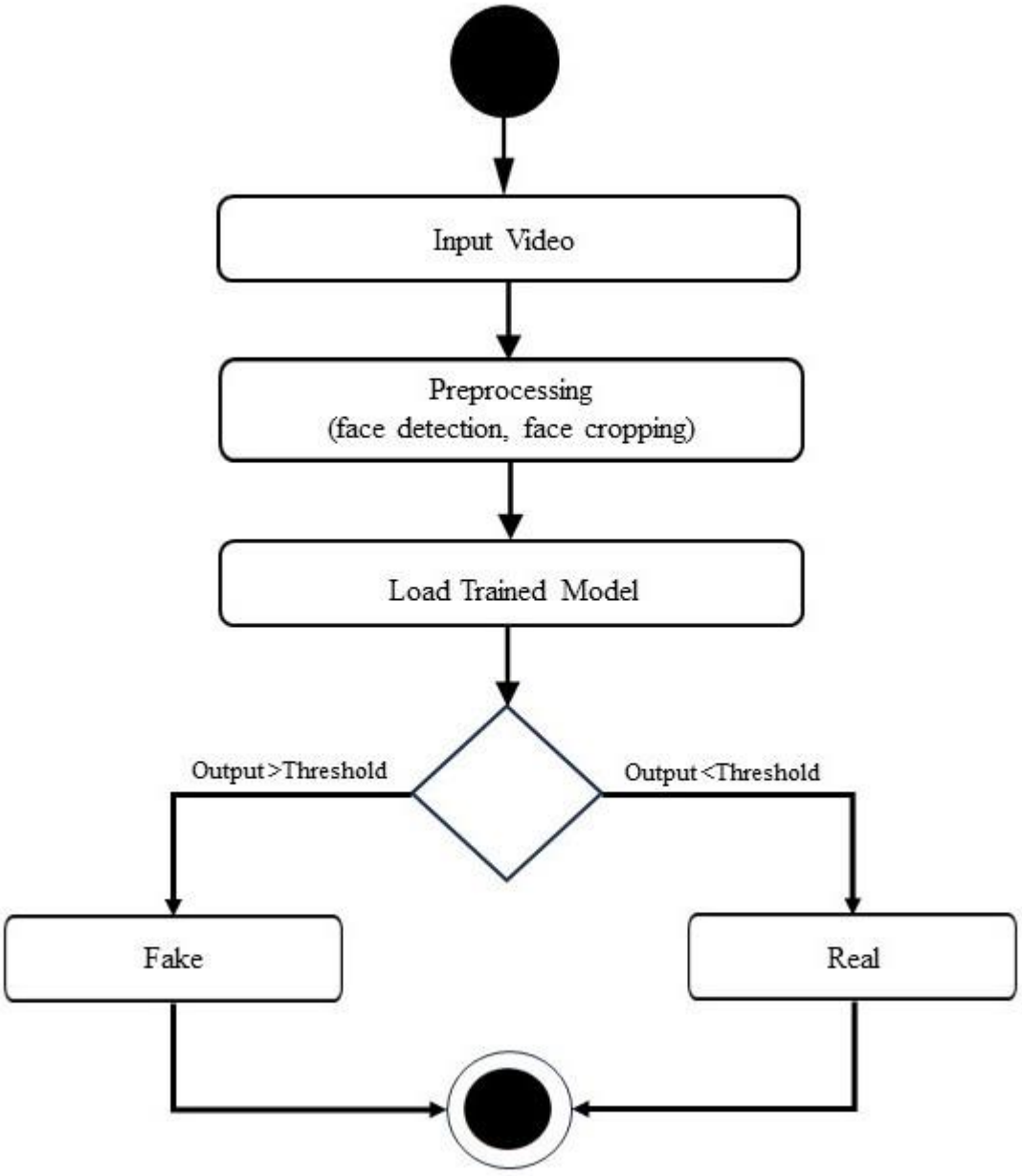


Fig. 5.8 testing workflow diagram

CHAPTER 6

SYSTEM IMPLEMENTATION

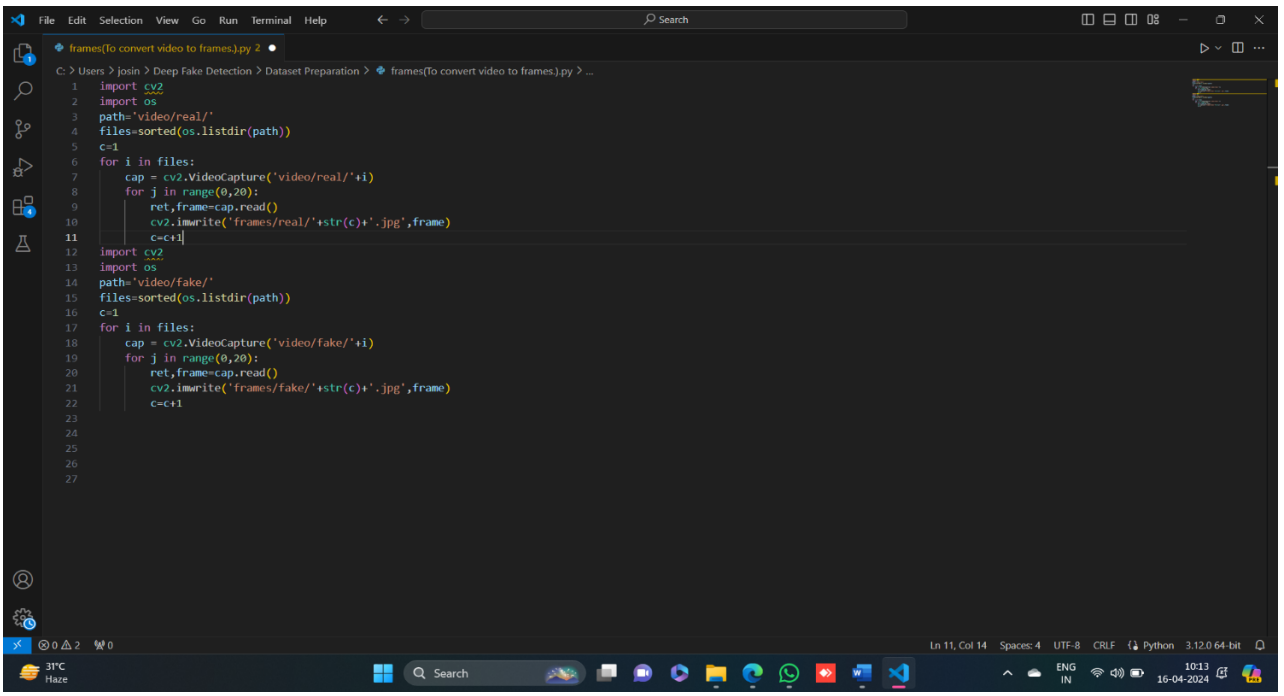
6.1 DATASET LOADING

For making the model efficient for real time prediction, we have gathered the FaceForensic++ dataset which is publicly available will be used. It contains videos and images manipulated using deepfakes, face2face and faceswap methods.

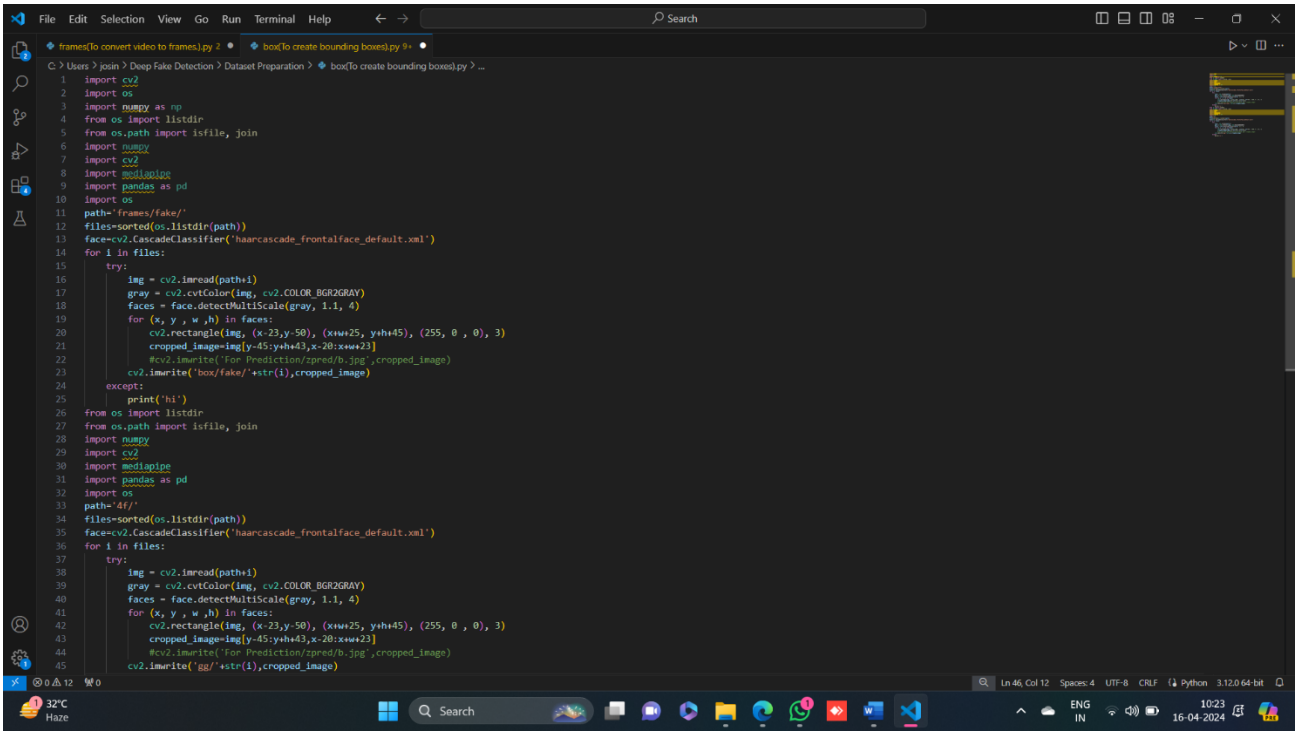
6.2 DATASET PREPROCESSING

In this step, the videos are preprocessed and all the unrequired and noise is removed from videos. Only the required portion of the video i.e face is detected and cropped. The first steps in the preprocessing of the video is to split the video into frames. After splitting the video into frames the face is detected in each of the frame and the frame is cropped along the face using bounding box. Later the face and context is segmented and resized.

6.2.1 Extraction of frames:

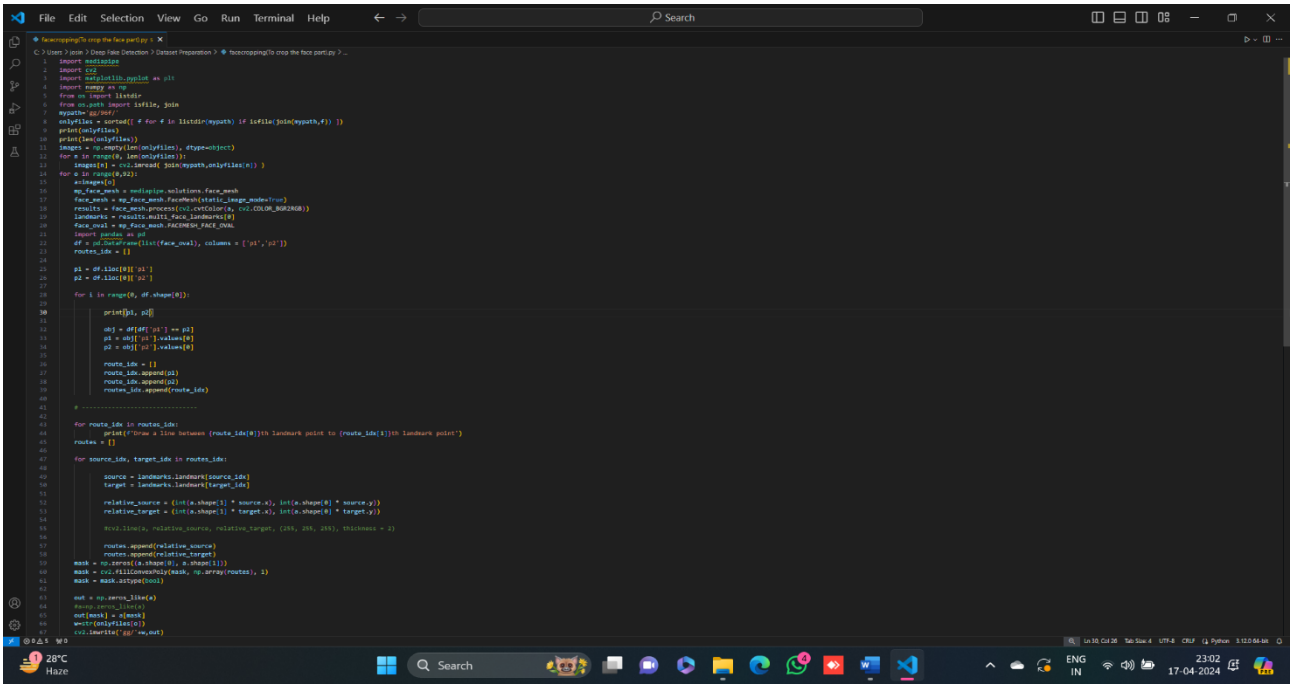


6.2.2 Bounding Box Face Cropping



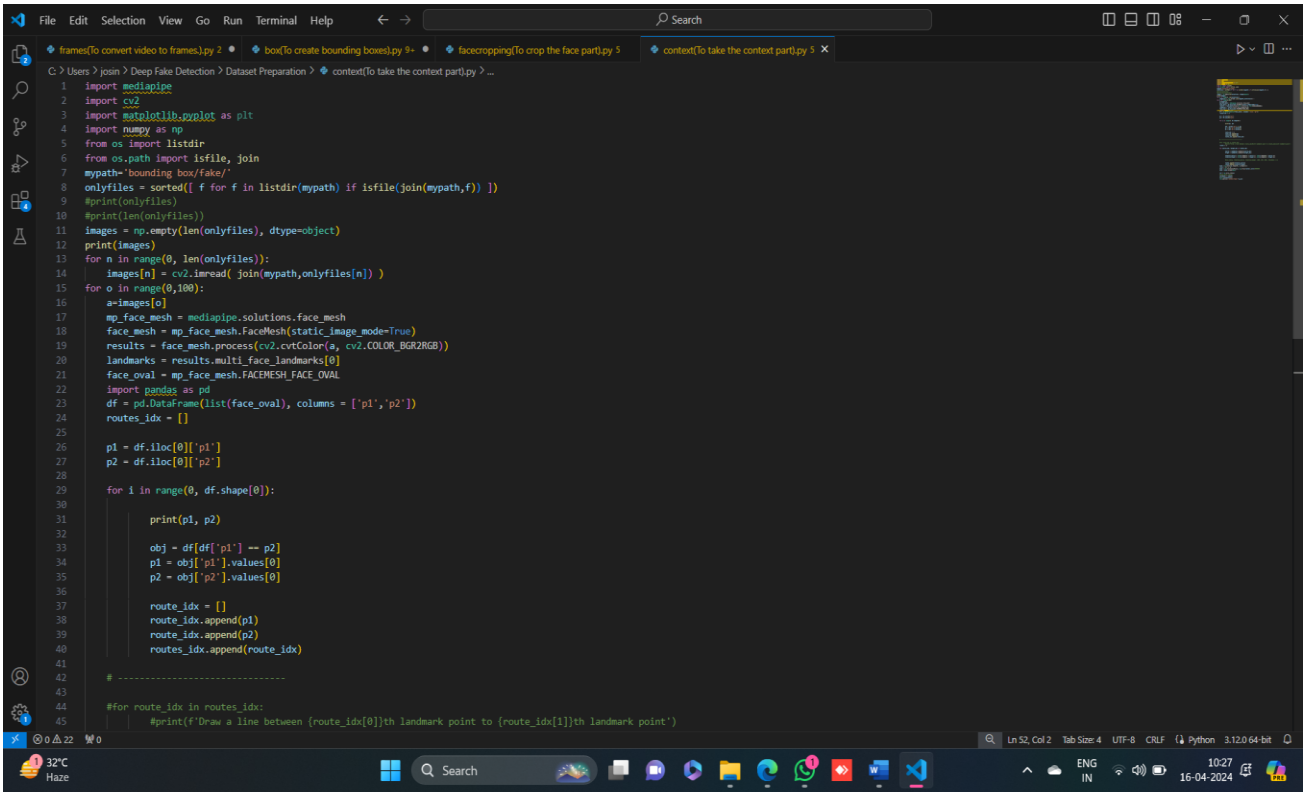
```
import cv2
import os
import numpy as np
from os import listdir
from os.path import isfile, join
import numpy
import cv2
import mediapipe
import pandas as pd
import os
path='frames/fake/'
files=sorted(os.listdir(path))
face=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
for i in files:
    try:
        img = cv2.imread(path+i)
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        faces = face.detectMultiScale(gray, 1.1, 4)
        for (x, y , w ,h) in faces:
            cv2.rectangle(img, (x-23,y-50), (x+w+25, y+h+45), (255, 0 , 0), 3)
            cropped_image=img[y-45:y+h+43,x-20:x+w+23]
            #cv2.imwrite('For Prediction/zpred/b.jpg',cropped_image)
        cv2.imwrite('box/fake/'+str(i),cropped_image)
    except:
        print('hi')
from os import listdir
from os.path import isfile, join
import numpy
import cv2
import mediapipe
import pandas as pd
import os
path='4f/'
files=sorted(os.listdir(path))
face=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
for i in files:
    try:
        img = cv2.imread(path+i)
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        faces = face.detectMultiScale(gray, 1.1, 4)
        for (x, y , w ,h) in faces:
            cv2.rectangle(img, (x-23,y-50), (x+w+25, y+h+45), (255, 0 , 0), 3)
            cropped_image=img[y-45:y+h+43,x-20:x+w+23]
            #cv2.imwrite('For Prediction/zpred/b.jpg',cropped_image)
        cv2.imwrite('gg/'+str(i),cropped_image)
```

6.2.3 Face Segmentation

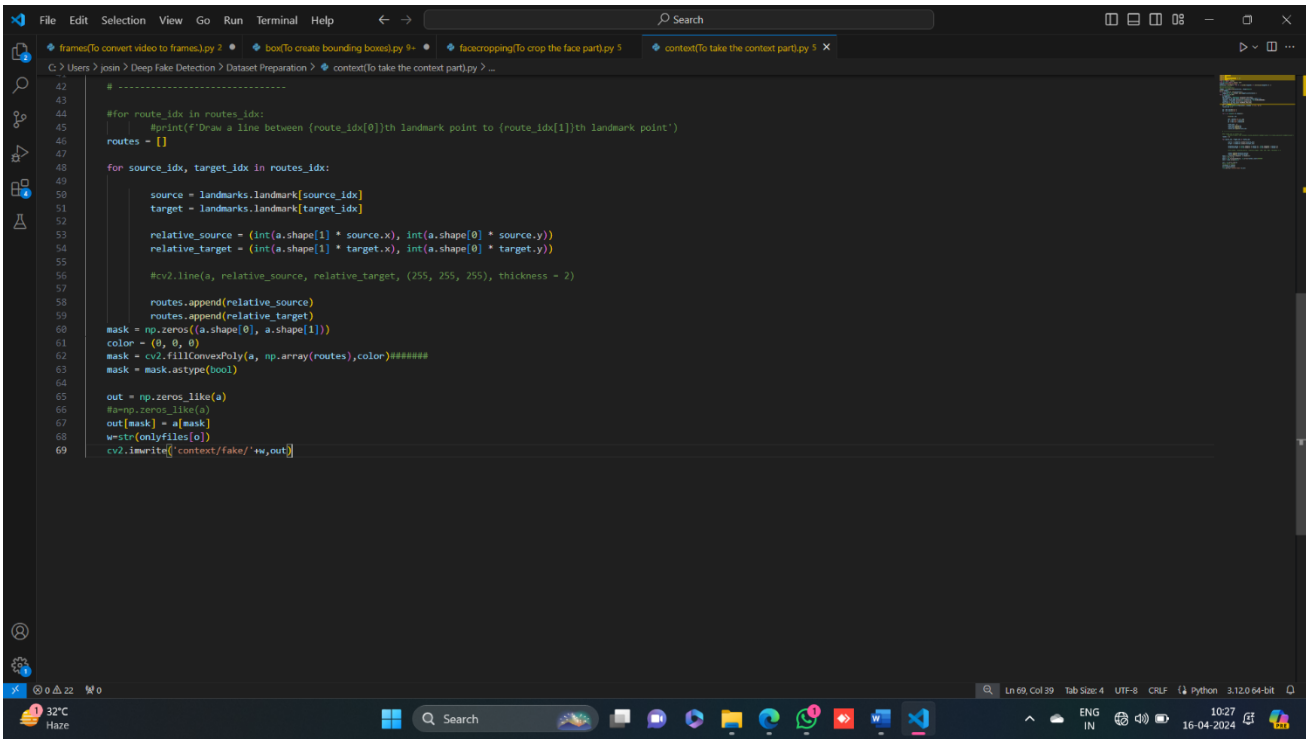


```
import mediapipe
import cv2
import matplotlib.pyplot as plt
import numpy as np
from os import listdir
from os.path import isfile, join
mypath='gg/001/'
onlyfiles = sorted([ f for f in listdir(mypath) if isfile(join(mypath,f)) ])
print(onlyfiles)
print(len(onlyfiles))
images = np.empty(len(onlyfiles), dtype=object)
for n in range(0, len(onlyfiles)):
    images[n] = cv2.imread( join(mypath,onlyfiles[n]) )
for a in range(0,92):
    a=images[a]
    mp_face_mesh = mediapipe.solutions.face_mesh
    face_mesh = mp_face_mesh.FaceMesh(static_image_mode=True)
    results = face_mesh.process(cv2.cvtColor(a, cv2.COLOR_BGR2RGB))
    landmarks = results.multi_face_landmarks[0]
    face_oval = mp_face_mesh.FACEMESH_FACE_OVAL
    import pandas as pd
    df = pd.DataFrame(list(face_oval), columns = ['p1','p2'])
    routes_idx = []

    p1 = df.iloc[0]['p1']
    p2 = df.iloc[0]['p2']

    for i in range(0, df.shape[0]):

        print(p1, p2)

        obj = df[df['p1'] == p2]
        p1 = obj['p1'].values[0]
        p2 = obj['p2'].values[0]

        route_idx = []
        route_idx.append(p1)
        route_idx.append(p2)
        routes_idx.append(route_idx)

    # -------------------------------

    for route_idx in routes_idx:
        print(f"Draw a line between {route_idx[0]}th landmark point to {route_idx[1]}th landmark point")
    routes = []

    for source_idx, target_idx in routes_idx:

        source = landmarks.landmark[source_idx]
        target = landmarks.landmark[target_idx]

        relative_source = (int(a.shape[1] * source.x), int(a.shape[0] * source.y))
        relative_target = (int(a.shape[1] * target.x), int(a.shape[0] * target.y))

        #cv2.line(a, relative_source, relative_target, (255, 255, 255), thickness = 2)

        routes.append(relative_source)
        routes.append(relative_target)
    mask = np.zeros((a.shape[0], a.shape[1]))
    mask = cv2.fillConvexPoly(mask, np.array(routes), 1)
    mask = mask.astype(bool)

    out = np.zeros_like(a)
    #print(zeros_like(a))
    out[mask] = a[mask]
    w=str(onlyfiles[a])
    cv2.imwrite('gg/'+w,out)
```

6.2.4 Context segmentation



```python
import mediapipe
import cv2
import matplotlib.pyplot as plt
import numpy as np
from os import listdir
from os.path import isfile, join
mypath='bounding box/fake/'
onlyfiles = sorted([ f for f in listdir(mypath) if isfile(join(mypath,f)) ])
#print(onlyfiles)
#print(len(onlyfiles))
images = np.empty(len(onlyfiles), dtype=object)
print(images)
for n in range(0, len(onlyfiles)):
    images[n] = cv2.imread( join(mypath,onlyfiles[n]) )
for o in range(0,100):
    a=images[o]
    mp_face_mesh = mediapipe.solutions.face_mesh
    face_mesh = mp_face_mesh.FaceMesh(static_image_mode=True)
    results = face_mesh.process(cv2.cvtColor(a, cv2.COLOR_BGR2RGB))
    landmarks = results.multi_face_landmarks[0]
    face_oval = mp_face_mesh.FACEMESH_FACE_OVAL
    import pandas as pd
    df = pd.DataFrame(list(face_oval), columns = ['p1','p2'])
    routes_idx = []

    p1 = df.iloc[0]['p1']
    p2 = df.iloc[0]['p2']

    for i in range(0, df.shape[0]):

            print(p1, p2)

            obj = df[df['p1'] == p2]
            p1 = obj['p1'].values[0]
            p2 = obj['p2'].values[0]

            route_idx = []
            route_idx.append(p1)
            route_idx.append(p2)
            routes_idx.append(route_idx)

    # -------------------------------

    #for route_idx in routes_idx:
            #print(f'Draw a line between {route_idx[0]}th landmark point to {route_idx[1]}th landmark point')
```



```python
    # -------------------------------

    #for route_idx in routes_idx:
            #print(f'Draw a line between {route_idx[0]}th landmark point to {route_idx[1]}th landmark point')
    routes = []

    for source_idx, target_idx in routes_idx:

            source = landmarks.landmark[source_idx]
            target = landmarks.landmark[target_idx]

            relative_source = (int(a.shape[1] * source.x), int(a.shape[0] * source.y))
            relative_target = (int(a.shape[1] * target.x), int(a.shape[0] * target.y))

            #cv2.line(a, relative_source, relative_target, (255, 255, 255), thickness = 2)

            routes.append(relative_source)
            routes.append(relative_target)
    mask = np.zeros((a.shape[0], a.shape[1]))
    color = (0, 0, 0)
    mask = cv2.fillConvexPoly(a, np.array(routes),color)#######
    mask = mask.astype(bool)

    out = np.zeros_like(a)
    #a=np.zeros_like(a)
    out[mask] = a[mask]
    w=str(onlyfiles[o])
    cv2.imwrite('context/fake/'+w,out)
```

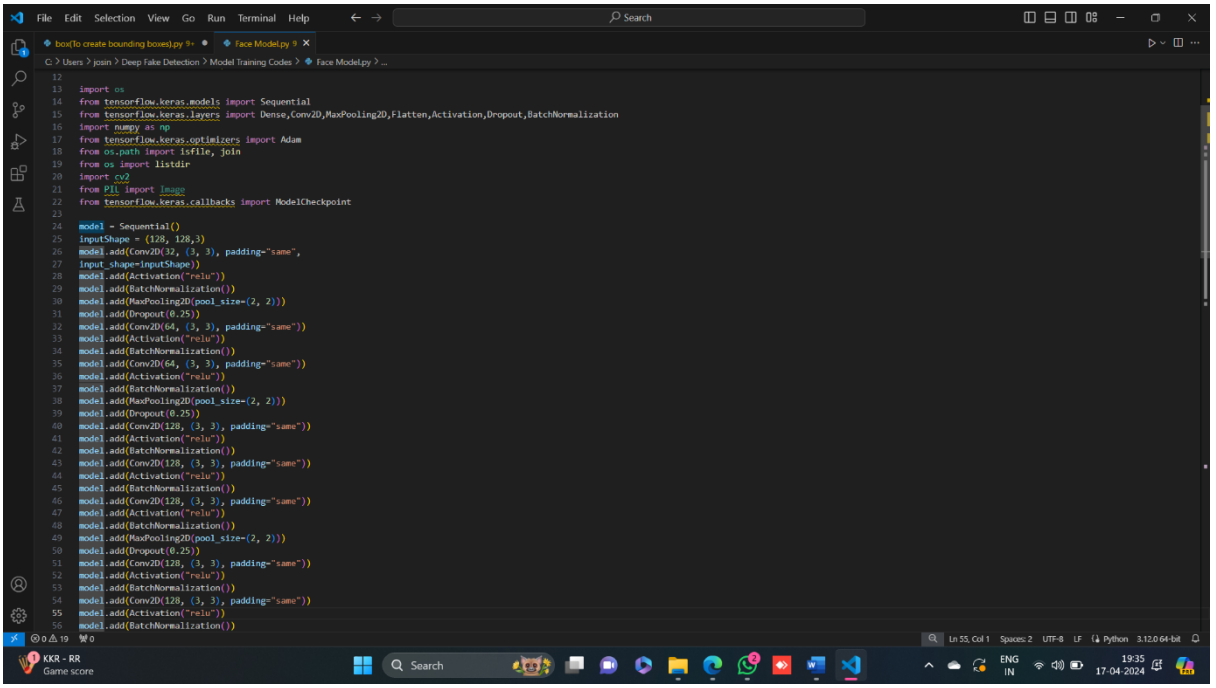
6.3 DEEPFAKE DETECTION MODEL

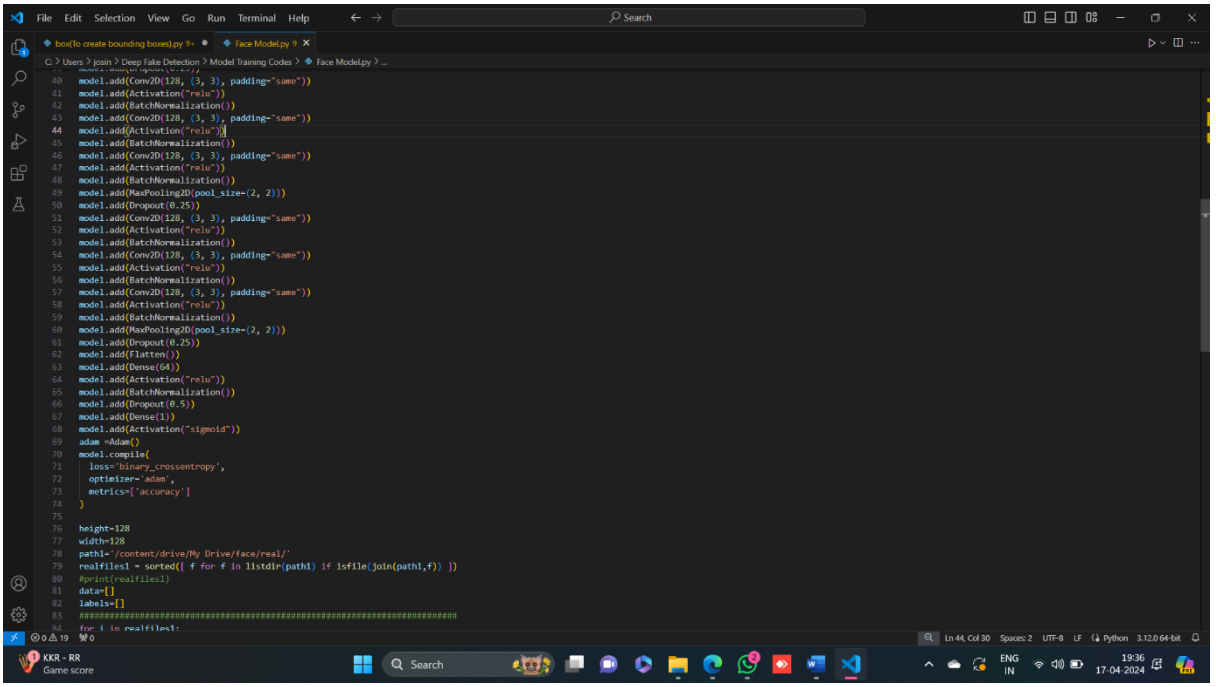
Three customized networks are used for feature extraction namely, Face network, context recognition network and bounding box network. The resulted extracted features and concatenated and fed to classification network.

1. Face Network
2. Context Recognition Network
3. Bounding Box Analysis Network
4. Classification Network

6.3.1 Face Network Model

This model is designed to scrutinize extracted facial regions, extracting features related to facial characteristics, textures, and potential inconsistencies. Its primary objective is to generate feature vectors representing the nuanced details of the face.

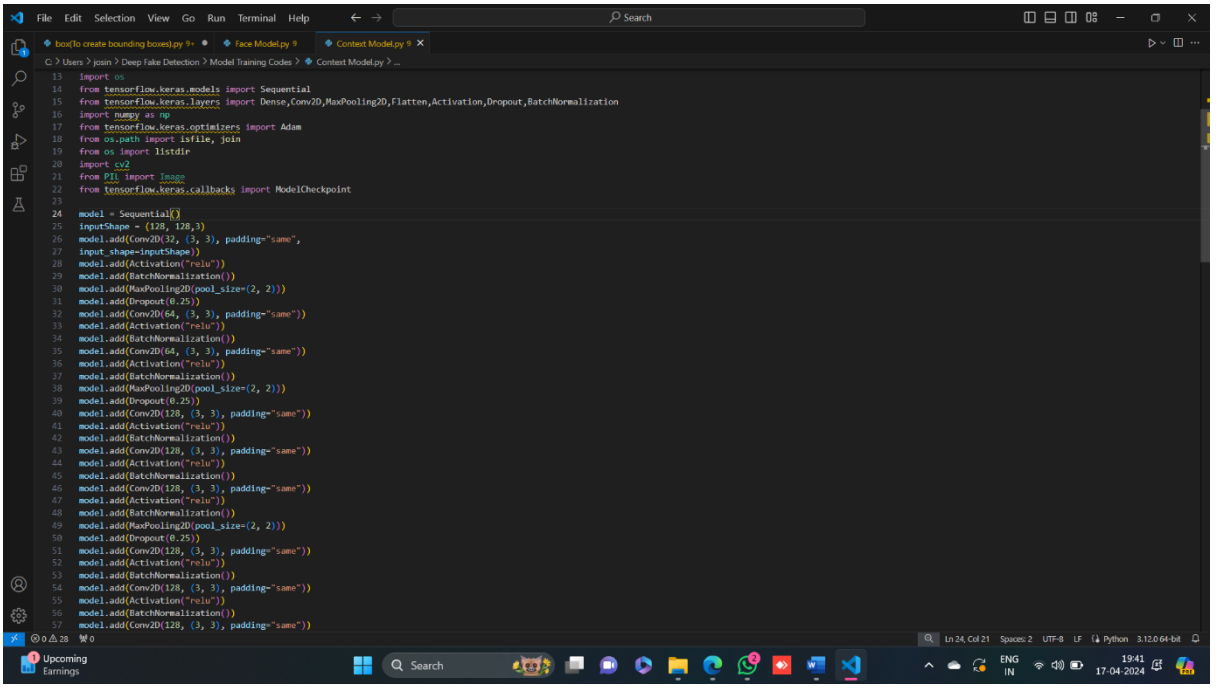




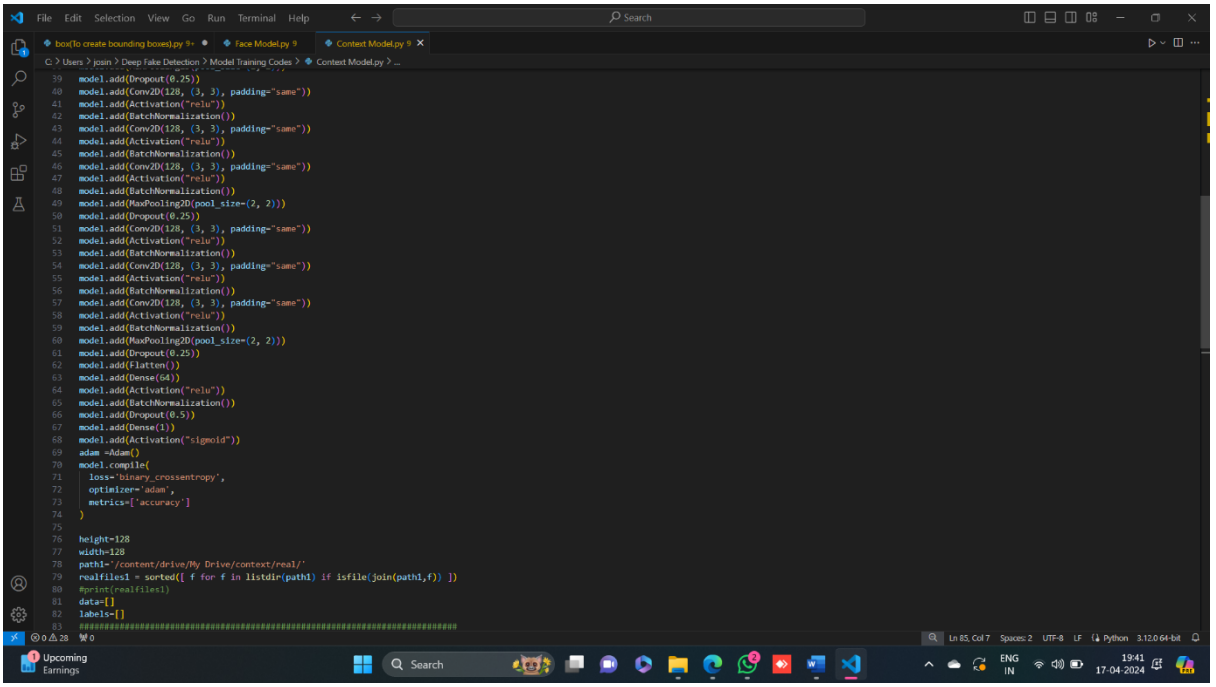
```
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(64))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation("sigmoid"))
adam =Adam()
model.compile(
  loss='binary_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)

height=128
width=128
path1='/content/drive/My Drive/face/real/'
realfiles1 = sorted([ f for f in listdir(path1) if isfile(join(path1,f)) ])
#print(realfiles1)
data=[]
labels=[]
##########################################################################
for i in realfiles1:
```

6.3.2 Context Network Model

Tasked with analyzing the segmented background context, this model focuses on identifying inconsistencies or unnatural patterns that may indicate manipulation. By extracting feature vectors, it aims to capture the essence of the background and detect any alterations or anomalies.



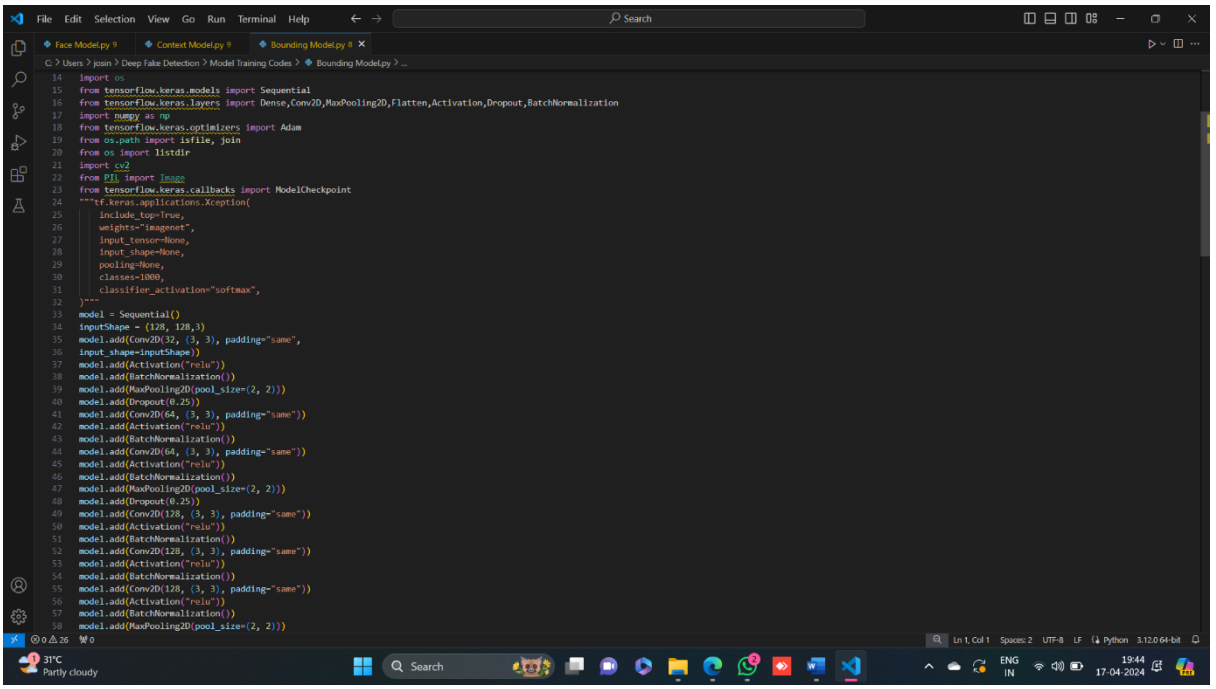
```
import os
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Conv2D,MaxPooling2D,Flatten,Activation,Dropout,BatchNormalization
import numpy as np
from tensorflow.keras.optimizers import Adam
from os.path import isfile, join
from os import listdir
import cv2
from PIL import Image
from tensorflow.keras.callbacks import ModelCheckpoint

model = Sequential()
inputShape = (128, 128,3)
model.add(Conv2D(32, (3, 3), padding="same",
input_shape=inputShape))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), padding="same"))
```



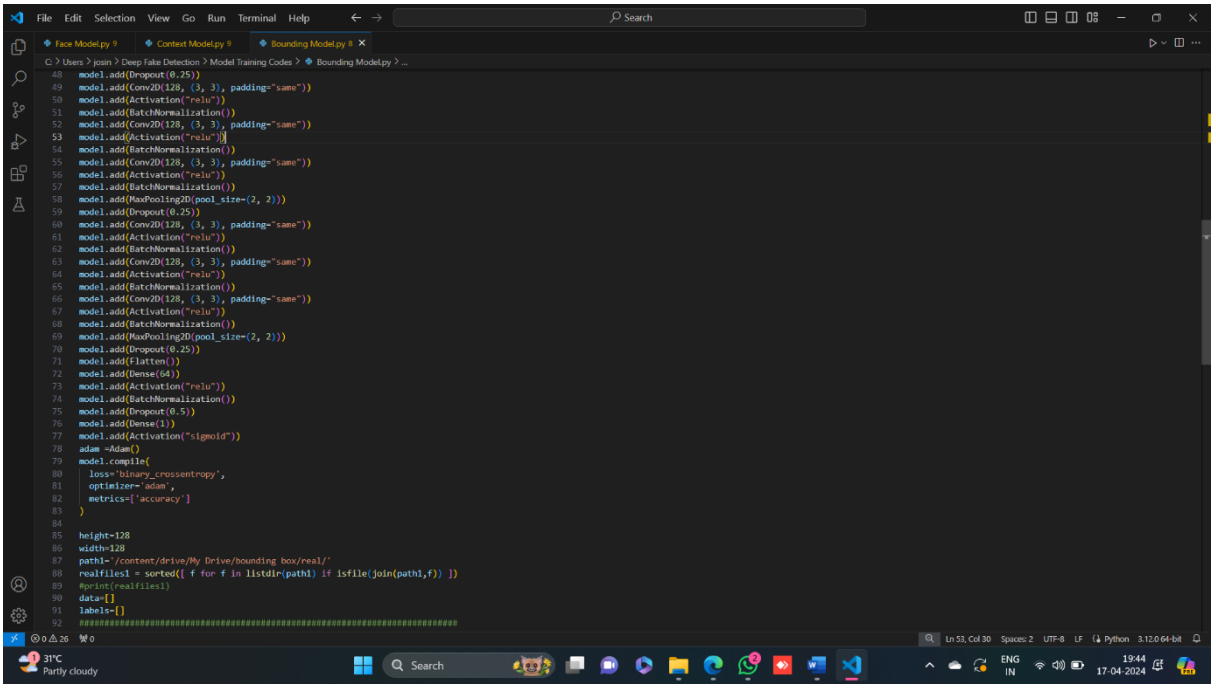
```
model.add(Dropout(0.25))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(64))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation("sigmoid"))
adam =Adam()
model.compile(
  loss='binary_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)

height=128
width=128
path1='/content/drive/My Drive/context/real/'
realfiles1 = sorted([ f for f in listdir(path1) if isfile(join(path1,f)) ])
#print(realfiles1)
data=[]
labels=[]
```

6.3.3 Bounding Box Network Model

Dedicated to analyzing the extracted bounding boxes, this model learns to identify potential discrepancies in size, position, or other attributes that might suggest manipulation. Its feature vectors encapsulate key information regarding the bounding boxes, aiding in the detection of any irregularities.



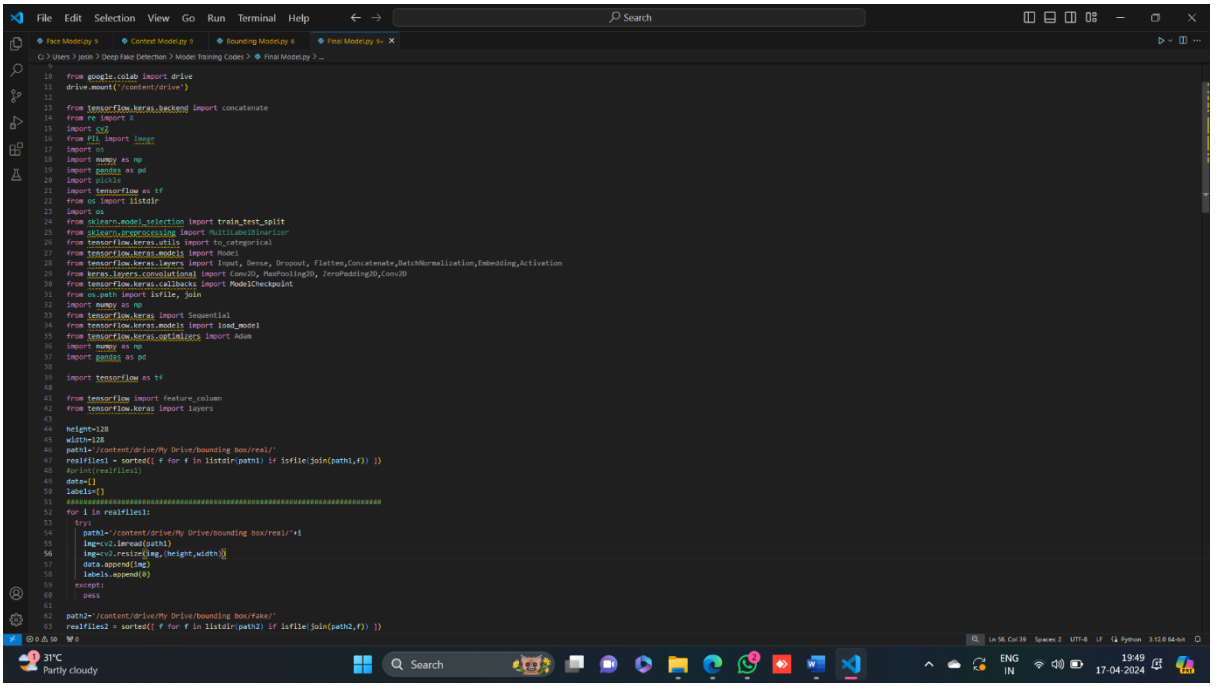
```
import os
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Conv2D,MaxPooling2D,Flatten,Activation,Dropout,BatchNormalization
import numpy as np
from tensorflow.keras.optimizers import Adam
from os.path import isfile, join
from os import listdir
import cv2
from PIL import Image
from tensorflow.keras.callbacks import ModelCheckpoint
"""tf.keras.applications.Xception(
    include_top=True,
    weights="imagenet",
    input_tensor=None,
    input_shape=None,
    pooling=None,
    classes=1000,
    classifier_activation="softmax",
)"""
model = Sequential()
inputShape = (128, 128,3)
model.add(Conv2D(32, (3, 3), padding="same",
input_shape=inputShape))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
```



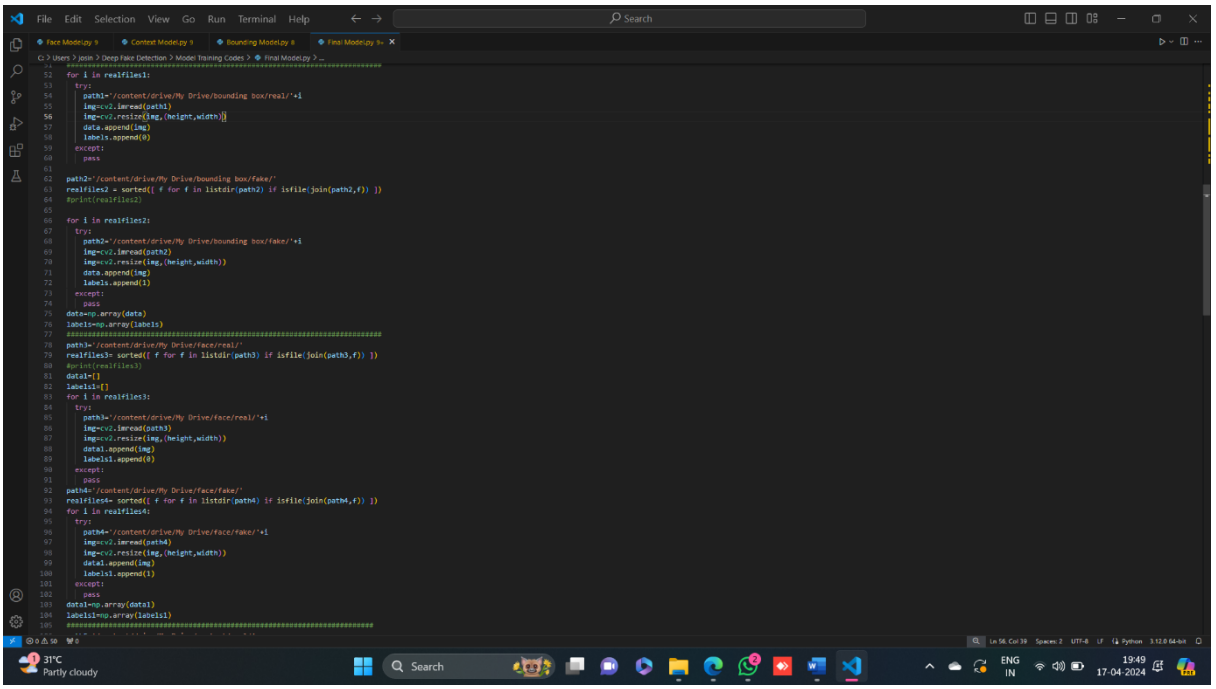
```
model.add(Dropout(0.25))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(64))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation("sigmoid"))
adam =Adam()
model.compile(
  loss='binary_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)

height=128
width=128
path1='/content/drive/My Drive/bounding box/real/'
realfiles1 = sorted([ f for f in listdir(path1) if isfile(join(path1,f)) ])
#print(realfiles1)
data=[]
labels=[]
```

6.3.4 Classification Model

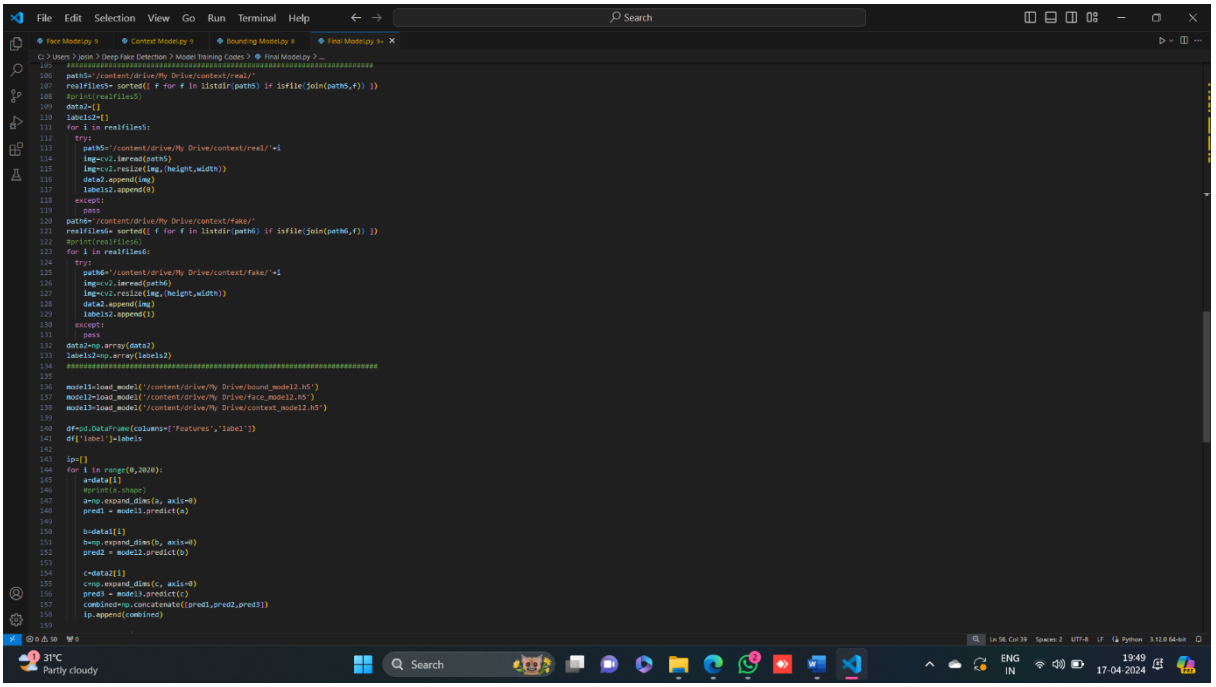
This final network integrates the outputs from the Facial Analysis Network, Context Recognition Network, and Bounding Box Analysis Network. These outputs, represented as concatenated feature vectors, serve as inputs for the classification task. By leveraging the collective information extracted from facial regions, background context, and bounding boxes, this network aims to classify images based on their authenticity or detect potential manipulations.

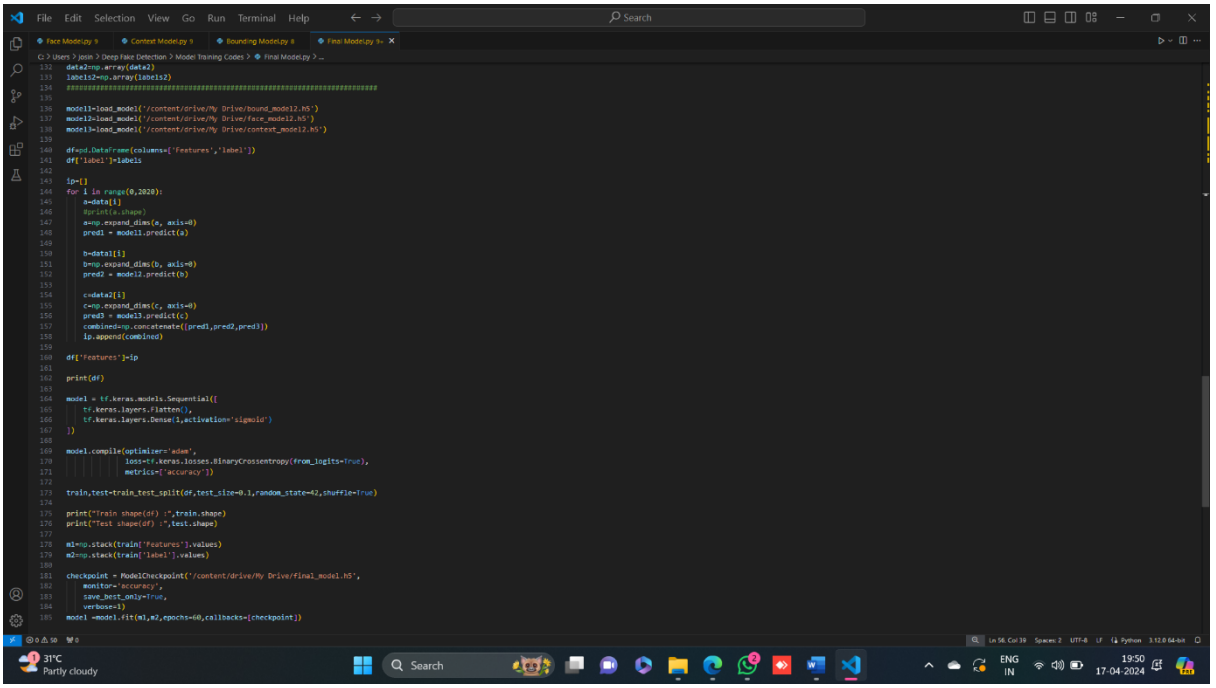


```
from google.colab import drive
drive.mount('/content/drive')

from tensorflow.keras.backend import concatenate
from re import X
import cv2
from PIL import Image
import os
import numpy as np
import pandas as pd
import pickle
import tensorflow as tf
from os import listdir
import os
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MultiLabelBinarizer
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense, Dropout, Flatten,Concatenate,BatchNormalization,Embedding,Activation
from keras.layers.convolutional import Conv2D, MaxPooling2D, ZeroPadding2D,Conv2D
from tensorflow.keras.callbacks import ModelCheckpoint
from os.path import isfile, join
import numpy as np
from tensorflow.keras import Sequential
from tensorflow.keras.models import load_model
from tensorflow.keras.optimizers import Adam
import numpy as np
import pandas as pd

import tensorflow as tf

from tensorflow import feature_column
from tensorflow.keras import layers

height=128
width=128
path1='/content/drive/My Drive/bounding box/real/'
realfiles1 = sorted([ f for f in listdir(path1) if isfile(join(path1,f)) ])
#print(realfiles1)
data=[]
labels=[]
#####################################################################
for i in realfiles1:
  try:
    path1='/content/drive/My Drive/bounding box/real/'+i
    img=cv2.imread(path1)
    img=cv2.resize(img,(height,width))
    data.append(img)
    labels.append(0)
  except:
    pass

path2='/content/drive/My Drive/bounding box/fake/'
realfiles2 = sorted([ f for f in listdir(path2) if isfile(join(path2,f)) ])
```



```
#####################################################################
for i in realfiles1:
  try:
    path1='/content/drive/My Drive/bounding box/real/'+i
    img=cv2.imread(path1)
    img=cv2.resize(img,(height,width))
    data.append(img)
    labels.append(0)
  except:
    pass

path2='/content/drive/My Drive/bounding box/fake/'
realfiles2 = sorted([ f for f in listdir(path2) if isfile(join(path2,f)) ])
#print(realfiles2)

for i in realfiles2:
  try:
    path2='/content/drive/My Drive/bounding box/fake/'+i
    img=cv2.imread(path2)
    img=cv2.resize(img,(height,width))
    data.append(img)
    labels.append(1)
  except:
    pass
data=np.array(data)
labels=np.array(labels)
#####################################################################
path3='/content/drive/My Drive/face/real/'
realfiles3= sorted([ f for f in listdir(path3) if isfile(join(path3,f)) ])
#print(realfiles3)
data1=[]
labels1=[]
for i in realfiles3:
  try:
    path3='/content/drive/My Drive/face/real/'+i
    img=cv2.imread(path3)
    img=cv2.resize(img,(height,width))
    data1.append(img)
    labels1.append(0)
  except:
    pass
path4='/content/drive/My Drive/face/fake/'
realfiles4= sorted([ f for f in listdir(path4) if isfile(join(path4,f)) ])
for i in realfiles4:
  try:
    path4='/content/drive/My Drive/face/fake/'+i
    img=cv2.imread(path4)
    img=cv2.resize(img,(height,width))
    data1.append(img)
    labels1.append(1)
  except:
    pass
data1=np.array(data1)
labels1=np.array(labels1)
#####################################################################
```



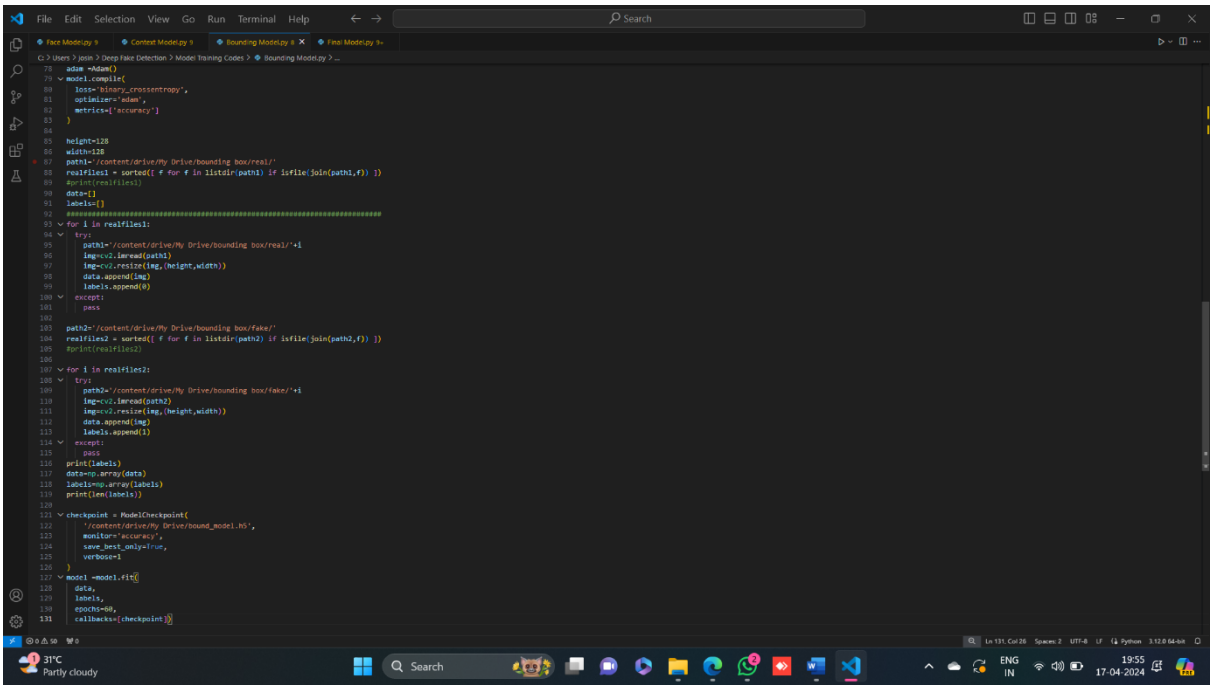


6.4 TRAINING PROCESS

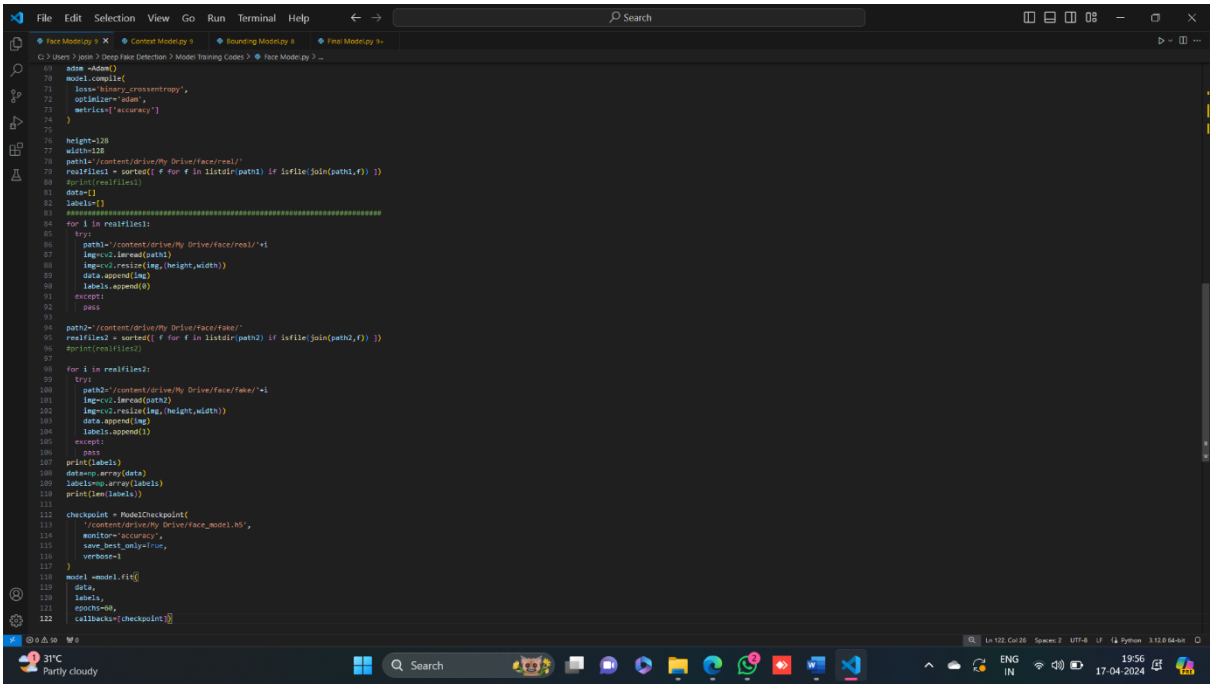
The training process begins with the provision of training data and labels, setting the number of epochs to 60. Each epoch represents a full iteration through the entire training dataset, during which the model adjusts its parameters to minimize the loss function, refining its ability to classify images accurately. At the end of each epoch, the model's performance on the training data is assessed. To ensure the preservation of the best-performing model, a Model Checkpoint callback is employed. This callback monitors the

model's accuracy and saves its weights whenever an improvement is detected compared to the previous best model. Once training concludes, the final trained model is saved for future use or deployment.

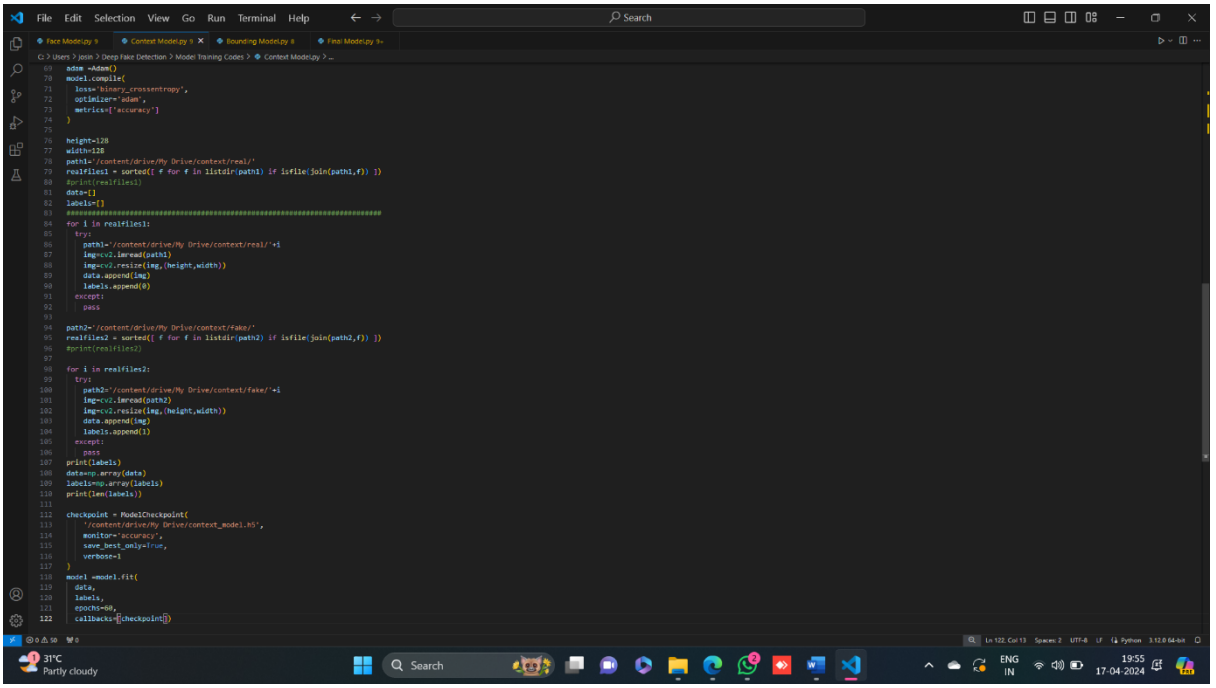
6.4.1 Bounding Box Network Training



6.4.2 Face Network Training

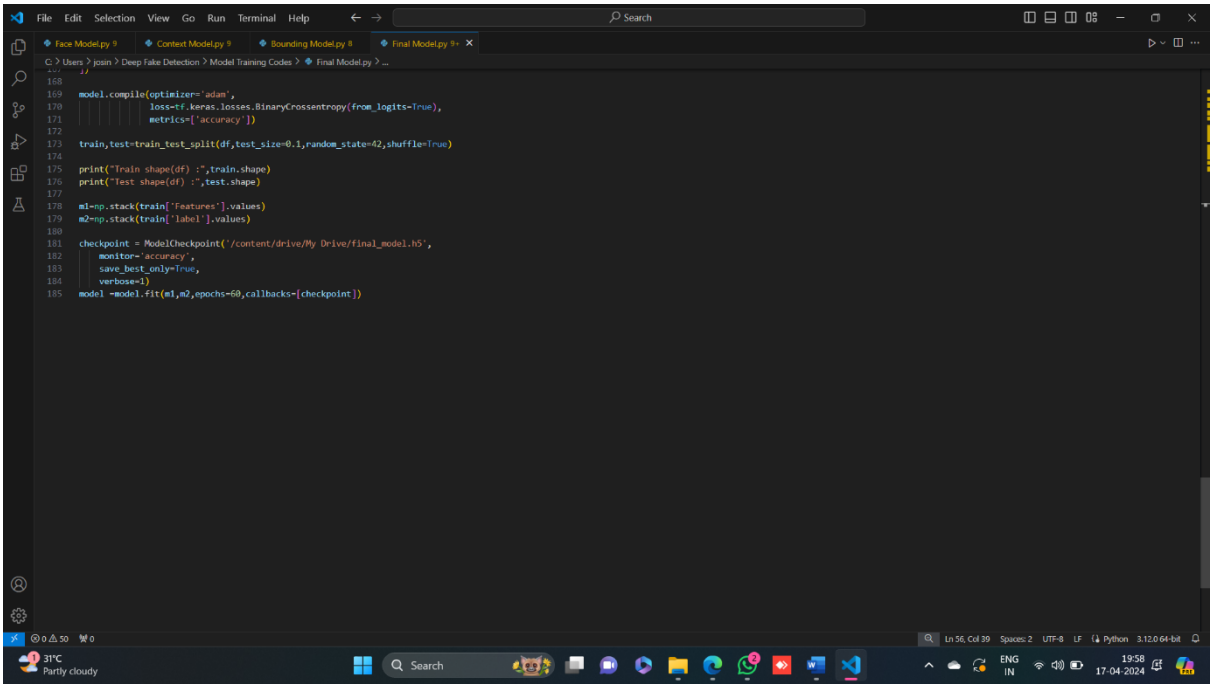


6.4.3 Context Network Training



```
adam =Adam()
model.compile(
  loss='binary_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)

height=128
width=128
path1='/content/drive/My Drive/context/real/'
realfiles1 = sorted([ f for f in listdir(path1) if isfile(join(path1,f)) ])
#print(realfiles1)
data=[]
labels=[]
##########################################################################
for i in realfiles1:
  try:
    path1='/content/drive/My Drive/context/real/'+i
    img=cv2.imread(path1)
    img=cv2.resize(img,(height,width))
    data.append(img)
    labels.append(0)
  except:
    pass

path2='/content/drive/My Drive/context/Fake/'
realfiles2 = sorted([ f for f in listdir(path2) if isfile(join(path2,f)) ])
#print(realfiles2)

for i in realfiles2:
  try:
    path2='/content/drive/My Drive/context/fake/'+i
    img=cv2.imread(path2)
    img=cv2.resize(img,(height,width))
    data.append(img)
    labels.append(1)
  except:
    pass
print(labels)
data=np.array(data)
labels=np.array(labels)
print(len(labels))

checkpoint = ModelCheckpoint(
    '/content/drive/My Drive/context_model.h5',
    monitor='accuracy',
    save_best_only=True,
    verbose=1
)
model =model.fit(
  data,
  labels,
  epochs=60,
  callbacks=[checkpoint])
```

6.4.4 Classification Network Training



```
model.compile(optimizer='adam',
              loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              metrics=['accuracy'])

train,test=train_test_split(df,test_size=0.1,random_state=42,shuffle=True)

print("Train shape(df) :",train.shape)
print("Test shape(df) :",test.shape)

m1=np.stack(train['Features'].values)
m2=np.stack(train['label'].values)

checkpoint = ModelCheckpoint('/content/drive/My Drive/final_model.h5',
    monitor='accuracy',
    save_best_only=True,
    verbose=1)
model =model.fit(m1,m2,epochs=60,callbacks=[checkpoint])
```

6.5 PREDICTION PROCESS

The prediction process begins with receiving either a video or image input, followed by the initialization and loading of a pretrained face detection model to identify faces within the content. After loading the trained classification model, frames are read from videos, and faces are detected using the pretrained model. Bounding box regions are then cropped around the detected faces for further analysis. These regions undergo segmentation into facial and background contexts, with respective feature extraction performed by loading the Bounding Box, Facial Analysis, and Context Recognition Network models. The extracted feature vectors from these networks are concatenated into a single vector, which is used as input for the trained classification model to predict the authenticity of the content. Finally, the results, indicating whether the content is deemed fake or real, are presented to the user for review.

CHAPTER 7

TESTING

The testing process involves evaluating the performance of the classification model on unseen data to assess its ability to generalize and make accurate predictions. Here's an overview of the testing process:

7.1 Data Preparation

Prepare a separate dataset (test_data) containing unseen samples that the model has not been exposed to during training. This dataset should include both input features (predictions from the pre-trained models) and corresponding labels.

7.2 Feature Extraction

Similar to the training process, extract features from the input data using the pre-trained models (model1, model2, and model3). These features will serve as input to the ensemble model for prediction.

7.3 Prediction

Combine the extracted features from the pre-trained models and feed them into the ensemble model. Use the predict method of the ensemble model to generate predictions for the unseen samples.

7.4 Evaluation

Compare the predicted labels with the ground truth labels from the testing dataset to evaluate the performance of the ensemble model. Accuracy score of the model on test_data is evaluated.

7.5 Result Analysis

Analyze the model's performance and identify any areas for improvement. This analysis involve examining misclassified samples, understanding the model's strengths and weaknesses, and considering potential adjustments to the model or the dataset.

CHAPTER 8

RESULTS

8.1 Home Window

Fig.8.1 Home Window

8.2 Uploading file

Step1: Click ‘SELECT IMAGE’ button or ‘SELECT VIDEO’ button correspondingly.

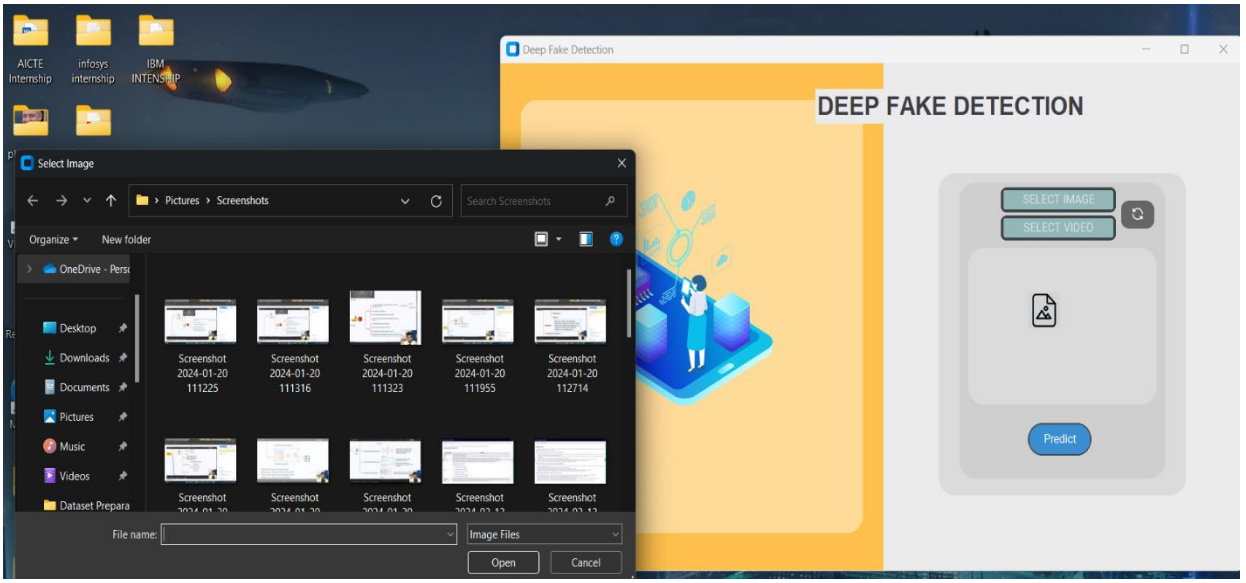


Fig 8.2 File Selection

Step 2: Select the file from the pop up window and click the ‘Predict’ button.

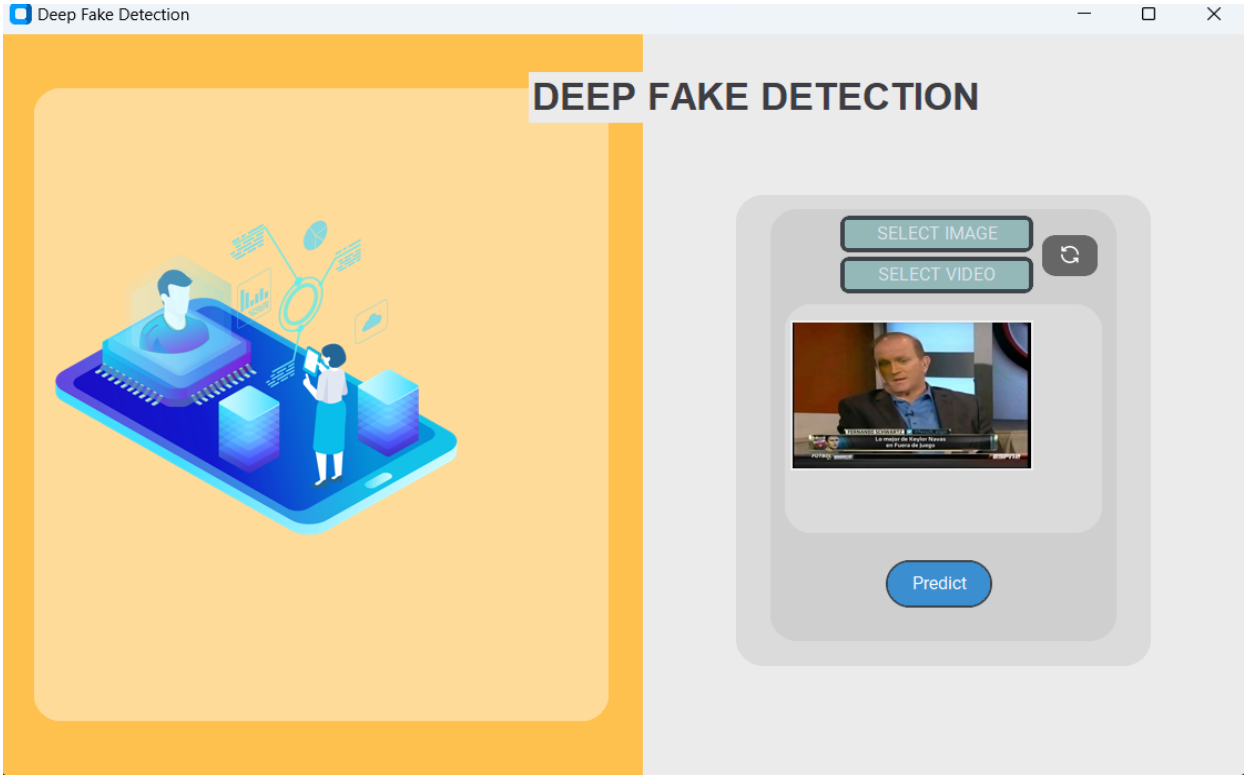


Fig 8.3 File Upload

8.3 Prediction Results

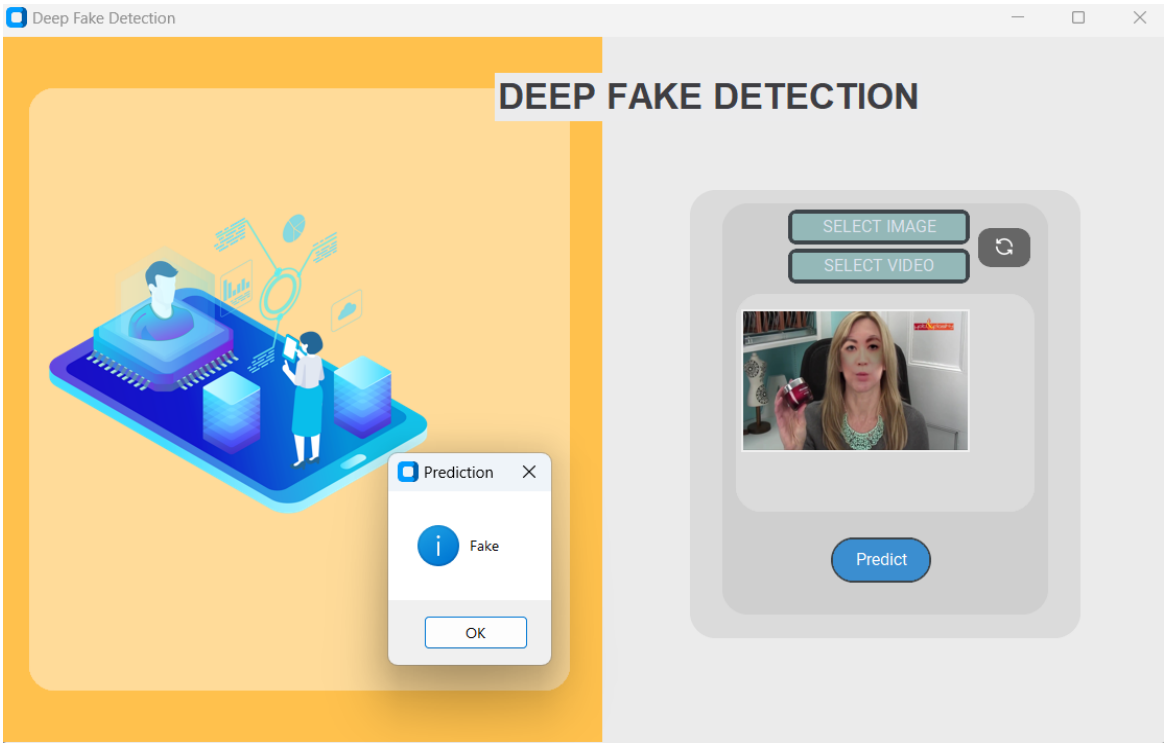


Fig.8.4 Fake Prediction

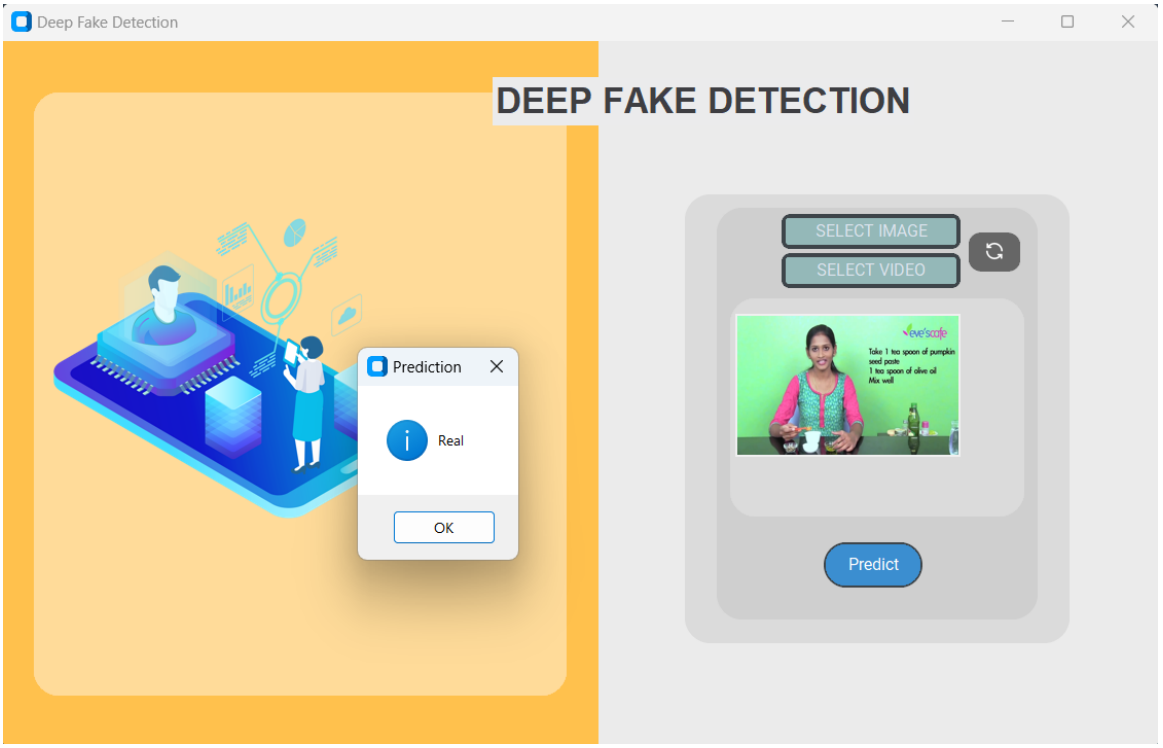


Fig.8.5 Real Prediction

CHAPTER 9
CONCLUSION

In conclusion, our project marks a significant milestone in the ongoing battle against digital misinformation, particularly in the realm of image and video manipulation detection. Through the integration of state-of-the-art techniques such as face detection, context segmentation, and bounding box analysis, we've developed a comprehensive and robust approach. Our method goes beyond surface-level inspection, delving deep into the intricate details of the media content to extract rich feature vectors that encapsulate various aspects of authenticity. By employing a multistage prediction process, we ensure thorough scrutiny and comprehensive assessment, enhancing the accuracy and reliability of our model. The strength of our approach lies in its ability to discern subtle nuances and identify even the most sophisticated manipulation techniques. Through meticulous feature extraction and classification, our model effectively distinguishes between authentic and manipulated media, providing valuable insights into potential instances of tampering. The concatenation of these techniques ensures a holistic understanding of the content, empowering users to make informed judgments about its trustworthiness. In conclusion, our project represents not only a technological achievement but also a commitment to the values of transparency, integrity, and truth. By leveraging the power of advanced technologies and collaborative partnerships, we stand poised to make a meaningful impact in the ongoing battle against digital misinformation, ensuring a safer and more reliable digital future for all.

REFERENCES

[1] Detecting GAN generated Fake Images using Co-occurrence Matrices Lakshmanan Nataraj; Mayachitra Inc., Santa Barbara, California, USAhttps://doi.org/10.2352/ISSN.2470-1173.2019.5.MWSF-532 © 2019, Society for Imaging Science and Technology

[2] Deepfake video detection using recurrent neural networks. In Proceedings of the 2020 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Auckland, New Zealand, 27–30 November 2020

[3] Exposing Deepfakes using inconsistent head poses. arXiv 2020, arXiv:1811.00656.
Ismail, A.; Elpeltagy, M.; S. Zaki, M.; Eldahshan, K. A New Deep Learning-Based Methodology for Video Deepfake Detection Using XGBoost. Sensors 2021, 21, 5413. https://doi.org/10.3390/s21165413

[4] In ictu oculi: Exposing ai created fake videos by detecting eye blinking. In Proceedings of the 2019 IEEE International Workshop on Information Forensics and Security (WIFS), Hong Kong, China, 11– 13 December 2019

[5] Elpeltagy, M.; S. Zaki, M.; Eldahshan, K. A New Deep Learning-Based Methodology for Video Deepfake Detection Using XGBoost. Sensors 2021, 21, 5413. https://doi.org/10.3390/s21165413

[6] FaceForensics++: Learning to Detect Manipulated Facial Images Andreas Rossler Davide Cozzolino Luisa Verdoliva Christian Riess Justus Thies Matthias Nießner 2019 IEEE/CVF International Conferenceon Computer Vision (ICCV)

[7]Methods of Deepfake Detection Based on Machine Learning Artem A. Maksutov1, Viacheslav O. Morozov, Aleksander A. Lavrenov, Alexander S. Smirnov 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)

[8] Deepfake Detection through Deep Learning Deng Pan, Lixian Sun, Rui Wang, Xingjian Zhang, Richard O. Sinnott 2020 IEEE/ACM International Conference on Big Data Computing, Applications andTechnologies (BDCAT)

[9] L. Verdoliva, ‘‘Media forensics and DeepFakes: An overview,’’ IEEE J. Sel. Topics Signal Process., vol.14, no. 5, pp. 910–932, Aug. 2020.

[10] K. Fukushima, ‘‘Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,’’ Biol. Cybern., vol. 36, no. 4, pp. 193–202, Apr. 1980.

[11] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard and L. Jackel, ‘‘Handwritten digit recognition with a back-propagation network,’’ in Proc. Adv. Neural Inf. Process. Syst., vol. 2, 1989, pp. 396–404.

[12] Y. LeCun, L. Bottou, Y. Bengio ‘‘Gradient-based learning applied to document recognition,’’ Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[13] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai,and T. Chen,‘‘Recent advances in convolutional neural networks,’’ Pattern Recognit., vol. 77, pp. 354–377, May 2018.

[14] S. Dong, P. Wang, and K. Abbas, ‘‘A survey on deep learning and its applications,’’ Comput. Sci. Rev.,vol. 40, May 2021, Art. no. 100379. 18772

[15] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M.Bernstein, and A. C. Berg, ‘‘ImageNet large scale visual recognition challenge,’’ Int. J. Comput. Vis., vol.115, no. 3, pp. 211– 252, Dec. 2015.

[16] M. D. Zeiler and R. Fergus, ‘‘Visualizing and understanding convolutional networks,’’ in Proc. Eur. Conf. Comput. Vis., Cham, Switzerland: Springer, 2014, pp. 818–833.

[17] K. Simonyan and A. Zisserman, ‘‘Very deep convolutional networks for large-scale image recognition,2014, arXiv:1409.1556.

[18] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, Going deeper with convolutions,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit.(CVPR), Jun. 2015