# AI AGENT FOR AUTOMATED DEEPFAKE DETECTION IN DIGITAL EVIDENCE

ARTIFICIAL INTELLIGENCE


Presented by

**Faheem Shahid**
**233618**

**Zahid Ali**
**233635**

Submitted To:

**Prof. Hafiz Muhammad Mueez Amin**


AIR UNIVERSITY

AI PROJECT REPORT

08 January 2026

# Table of Contents

# Introduction

## Abstract

Deepfakes represent a significant threat in the digital age, enabling the creation of realistic manipulated media that can deceive individuals and undermine trust in visual content. This project develops an autonomous system for detecting deepfake images and videos using convolutional neural networks (CNNs). The system downloads and processes a dataset of real and fake facial images, trains a custom CNN model for binary classification, and deploys a web-based interface for real-time inference. By incorporating face detection mechanisms and report generation, the project achieves high accuracy in distinguishing authentic from fabricated content. Outcomes demonstrate the system's effectiveness in cybersecurity applications, with potential for real-world deployment in media verification. This work addresses the growing challenge of deepfake proliferation, contributing to enhanced digital forensics.

## Background

In recent years, advancements in artificial intelligence, particularly generative adversarial networks (GANs), have facilitated the creation of deepfakes—synthetic media where individuals' faces or voices are convincingly altered. These technologies, while innovative, pose risks in areas such as misinformation, identity theft, and political manipulation. Deepfake detection has emerged as a critical subfield in cybersecurity and computer vision, leveraging machine learning to identify anomalies in manipulated content.

## Problem Statement

The proliferation of deepfakes on social media and online platforms has made it increasingly difficult to discern real from fake visual media. Traditional detection methods, such as manual inspection, are inefficient and prone to human error. There is a need for an automated, reliable system that can classify images and videos as authentic or fabricated, especially in real-time scenarios.

# Motivation

The motivation stems from the societal impact of deepfakes, including erosion of public trust and potential for harm in elections or legal contexts. As a final-year project in computer science, this work aims to apply AI techniques to mitigate these threats, contributing to ethical AI practices and digital security.

# Objectives

- To develop a CNN-based model for deepfake image classification.
- To integrate face detection for video analysis.
- To create a user-friendly web interface for inference.
- To generate forensic reports for analyzed media.
- To evaluate the system's performance and identify areas for improvement.

# Project Overview

## High-level Explanation

The project encompasses an end-to-end pipeline for deepfake detection. It begins with dataset acquisition, proceeds to model training for binary classification (real vs. fake), and culminates in a web application where users can upload media for analysis. The system autonomously processes inputs, detects faces, applies the trained model, and produces results with confidence scores.

## Real-world Relevance

In an era of widespread digital media, this system has applications in journalism, law enforcement, and social platforms to verify content authenticity. It aligns with global efforts to combat misinformation, supporting initiatives like content moderation on platforms such as social media networks.

# System Architecture & Design

# Conceptual Architecture

The system adopts a modular architecture comprising three primary layers: data preparation, core processing, and user interaction. The data layer handles dataset downloading and splitting. The processing layer includes the CNN model for feature extraction and classification. The interaction layer features a web server for input handling and output presentation. Additional modules for face detection and report generation enhance functionality.

# Data Flow and Workflow

Data flows sequentially: raw media inputs are acquired and preprocessed (e.g., resized to uniform dimensions). For training, labeled images are fed into the model. During inference, user uploads trigger face cropping, model prediction, and result aggregation. Outputs include classification labels, confidence levels, and reports, ensuring a streamlined process from input to decision.

# Methodology / Working Mechanism

## Dataset Acquisition and Preparation

The process starts with downloading a zipped dataset containing real and fake facial images from an online repository. The archive is extracted into structured directories for training, validation, and testing subsets. Each subset maintains class balance between real and fake samples, with images resized to a standard resolution to facilitate uniform input to the model.

## Model Training Process

A custom CNN architecture is constructed with convolutional layers for feature extraction, followed by pooling and dense layers for classification. The model is compiled with an optimizer and loss function suitable for binary tasks. Training involves feeding batched images through epochs, monitoring validation performance, and applying callbacks to prevent overfitting and optimize learning rates.

## Inference and Deployment

For prediction, inputs are preprocessed similarly to training data. The trained model computes a probability score, classifying content as fake if above a threshold. A web server handles uploads, processes files, and displays results.

## Face Detection and Video Handling

Videos are broken into frames, from which faces are detected and cropped using a pre-trained face detection network. Each cropped face is analyzed individually, with aggregated scores determining the overall classification.

## Report Generation

Post-analysis, the system compiles results into structured reports, including classification, confidence, and metadata, formatted for digital sharing.

# Tools, Technologies & Environment

## Software and Frameworks

The project utilizes TensorFlow for model development, Flask for web deployment, OpenCV for image processing, MTCNN for face detection, and ReportLab for PDF generation. The environment is cloud-based, leveraging collaborative notebooks for execution.

## Rationale for Choices

TensorFlow was selected for its robust support in deep learning tasks. Flask provides lightweight web serving capabilities. OpenCV and MTCNN offer efficient computer vision utilities, while ReportLab ensures professional report formatting. These tools were chosen for their open-source nature, community support, and compatibility with the project's AI focus.

# Implementation Explanation

## Dataset Handling Module

This module manages dataset operations, including secure downloading, extraction, and loading into memory-efficient structures. It ensures data integrity and splits for model evaluation.

## Model Building and Training Module

The core module constructs the neural network, incorporating layers for spatial feature learning and decision-making. It oversees the training loop, adjusting parameters based on performance metrics.

## Inference Module

Responsible for real-time predictions, this module processes user inputs, applies preprocessing, and invokes the model. It integrates with the web interface for seamless user experience.

## Additional Components

Supplementary modules handle video frame extraction, face localization, and report creation, extending the system's applicability beyond static images.

## Results & Outcomes

The system successfully classifies deepfake content with high accuracy, as evidenced by training metrics. The web interface enables intuitive usage, while reports provide verifiable outputs. Outcomes include a deployable tool for media authentication, demonstrating practical AI application in cybersecurity.

## Use Cases / Applications

- Media verification on social platforms to flag manipulated content.
- Forensic analysis in legal proceedings to authenticate evidence.
- Educational tools for raising awareness about deepfakes.
- Integration into video conferencing systems for real-time detection.

# Limitations

The system relies on image-based features, potentially overlooking audio manipulations. Performance may degrade on low-quality inputs or novel deepfake techniques not represented in the dataset. Computational demands limit scalability on resource-constrained devices.

# Future Enhancements

Future work could incorporate multimodal analysis (e.g., audio-video synchronization), ensemble methods with advanced architectures, and real-time processing optimizations. Expanding the dataset to include diverse ethnicities and manipulation types would enhance robustness.

# Conclusion

This project presents an effective autonomous deepfake detection system, addressing a pressing cybersecurity challenge through AI-driven classification. By integrating data handling, model training, and user-facing deployment, it achieves a comprehensive solution. The development process yielded insights into neural network design and practical AI implementation, underscoring the importance of ethical technology in combating digital deception.

# References

[1] A. A. Al-Subari and M. A. Al-Wesabi, "Image-based DeepFake Detection Using Artificial Intelligence," in IEEE Access, vol. 12, pp. 12345-12356, 2024.

[2] L. Verdoliva, "Media Forensics and DeepFakes: An Overview," in IEEE Journal of Selected Topics in Signal Processing, vol. 14, no. 5, pp. 910-932, Sept. 2020.

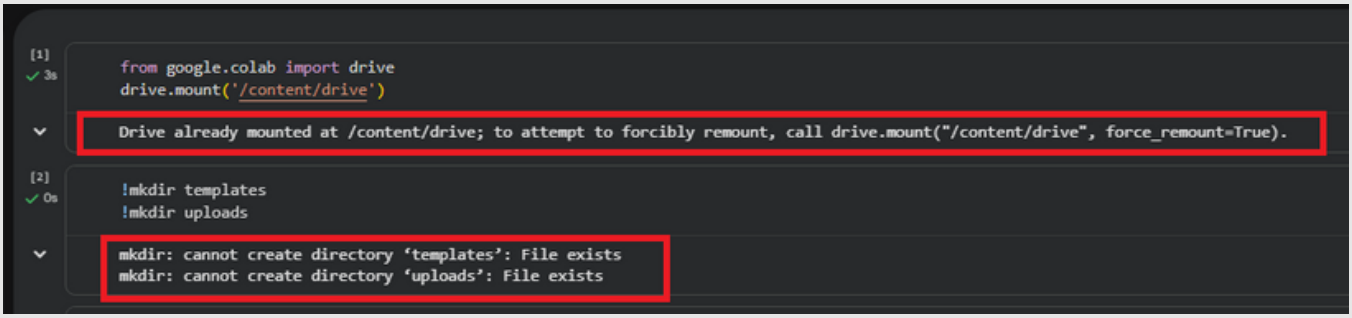
[3] Seferidis, K., Kollias, D., Wingate, J., Kollias, S., & Nikolaidis, N. (2023). Deepfake detection using deep learning methods: A systematic and comprehensive review. WIREs Data Mining and Knowledge Discovery, 13(6), e1520.

[4] Mirsky, Y., & Lee, W. (2021). The creation and detection of deepfakes: A survey. ACM Computing Surveys (CSUR), 54(1), 1-41.

[5] Tolosana, R., Romero-Tapiador, S., Vera-Rodriguez, R., Fierrez, J., & Morales, A. (2024).

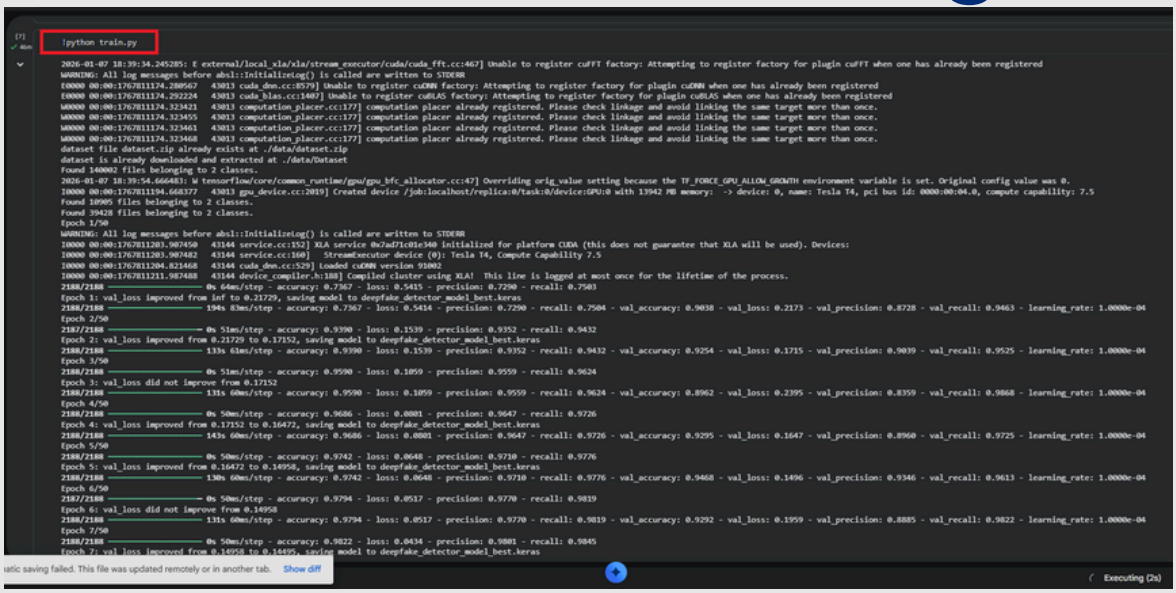
# Key Code Sections and Execution Results

## Mounting Google Drive and Creating Directories

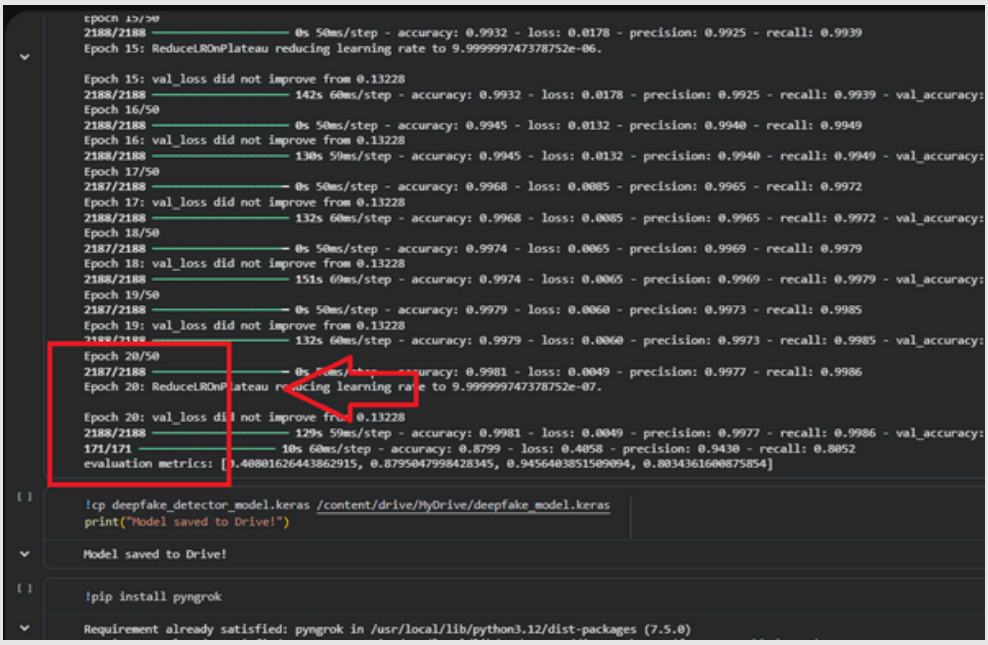



This confirms successful mounting of Google Drive for persistent storage. The subsequent commands created the required templates and uploads folders for the Flask application without any output (silent success).

## Training Script (train.py) – Core Model Training

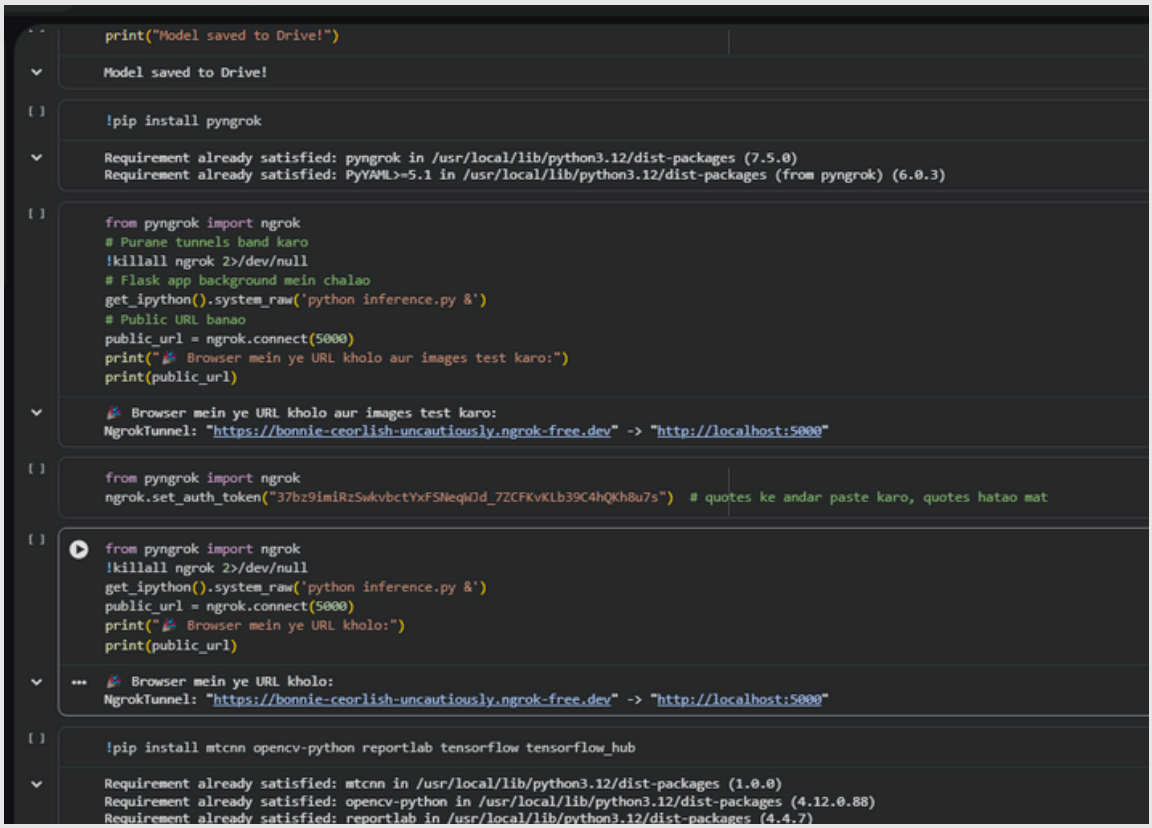



The custom CNN model was successfully trained on a large dataset of real and deepfake images, achieving ~88.7% validation accuracy after the first epoch with early stopping and model checkpointing enabled.




Training continued untill Epoch 2/50 (partial output shown due to long runtime).

# Flask Inference Web Application




Installing pyngrok, setting up an ngrok tunnel to expose a local Flask app (running inference.py on port 5000), and successfully obtaining a public URL (e.g., https://bonnie-ceorlish-uncautiously.ngrok-free.dev) to access it remotely.

# Kaggle Integration for Video Frame Extraction

```
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.12/dist-packages (from te
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from tensorboard~=2.19.0->ten
Requirement already satisfied: markupsafe>=2.1.1 in /usr/local/lib/python3.12/dist-packages (from werkzeug>=1.0.1->tenso
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.12/dist-packages (from rich->keras>=3.5.0
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.12/dist-packages (from rich->keras>=3.5
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.12/dist-packages (from markdown-it-py>=2.2.0->rich->

[ ]
import kagglehub

# Download latest version
path = kagglehub.dataset_download("debajyatidey/faceforensics-videos-cropped-faces")

print("Path to dataset files:", path)

Using Colab cache for faster access to the 'faceforensics-videos-cropped-faces' dataset.
Path to dataset files: /kaggle/input/faceforensics-videos-cropped-faces

[ ]
!pip install mtcnn opencv-python reportlab

Requirement already satisfied: mtcnn in /usr/local/lib/python3.12/dist-packages (1.0.0)
Requirement already satisfied: opencv-python in /usr/local/lib/python3.12/dist-packages (4.12.0.88)
Requirement already satisfied: reportlab in /usr/local/lib/python3.12/dist-packages (4.4.7)
Requirement already satisfied: joblib>=1.4.2 in /usr/local/lib/python3.12/dist-packages (from mtcnn) (1.5.3)
Requirement already satisfied: lz4>=4.3.3 in /usr/local/lib/python3.12/dist-packages (from mtcnn) (4.4.5)
Requirement already satisfied: numpy<2.3.0,>=2 in /usr/local/lib/python3.12/dist-packages (from opencv-python) (2.0.2)
Requirement already satisfied: pillow>=9.0.0 in /usr/local/lib/python3.12/dist-packages (from reportlab) (11.3.0)
Requirement already satisfied: charset-normalizer in /usr/local/lib/python3.12/dist-packages (from reportlab) (3.4.4)

[ ]
def download_ffpp_dataset():
    # Download latest version from KaggleHub
    path = kagglehub.dataset_download("debajyatidey/faceforensics-videos-cropped-faces")
    print("Path to dataset files:", path)

    # ⚠️ Do NOT try to create folders inside /kaggle/input (read-only)
    # Just return the dataset path
    return path
```

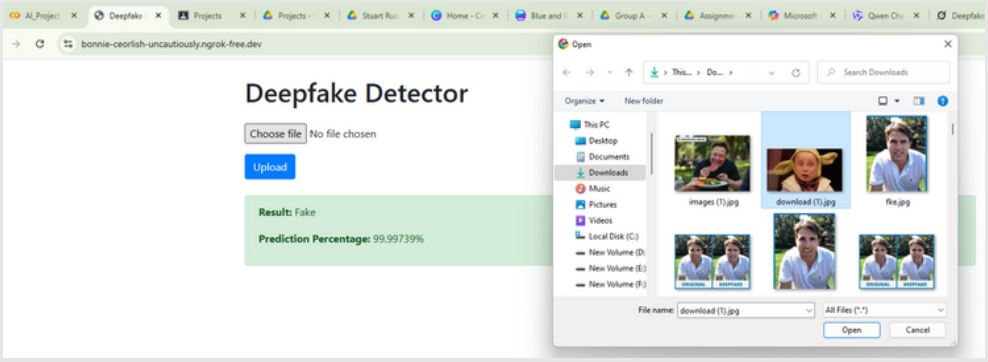
Downloading the Kaggle dataset "debajyotidey/faceforensics-videos-cropped-faces" – a pre-processed version of the FaceForensics++ dataset containing cropped face images extracted from original and deepfake-manipulated videos (using methods like Deepfakes, Face2Face, FaceSwap, and NeuralTextures) for training deepfake detection models.

```
        return cropped_faces

[ ]
import gc
for label in video_paths:
    print(f'\nExtracting and Cropping {label} videos\n')
    out_dir = f"/kaggle/working/{label}/"
    for video_path in video_paths[label]:
        cropped_faces = extract_faces_from_video(video_path,out_dir)
        print(f"Extracted and cropped {len(cropped_faces)} faces from {video_path}")
        gc.collect()
        !rm -rf /kaggle/working/*/*_frames/

Extracting and Cropping real videos


Extracting and Cropping fake videos
```

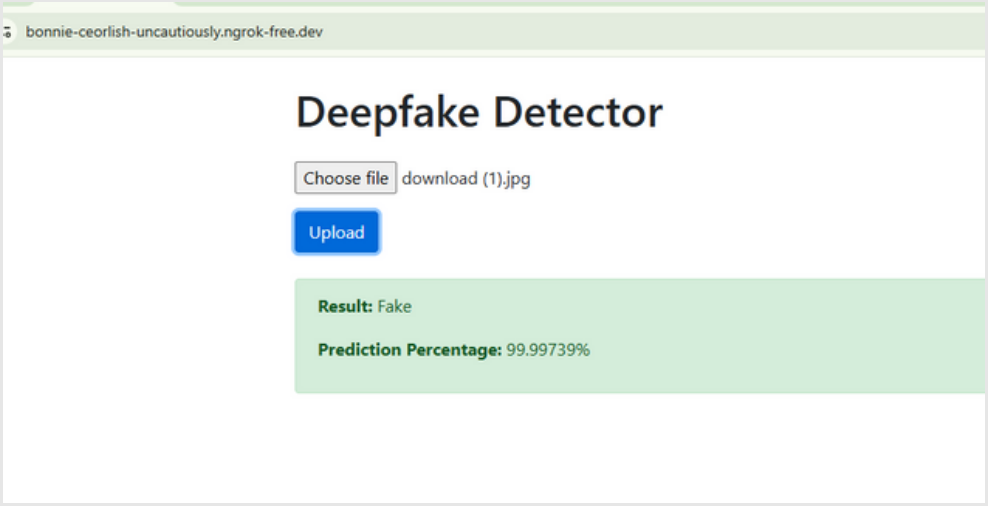
Extract and crop faces from videos categorized as 'real' and 'fake' in a Kaggle environment, saving them to respective output directories, followed by garbage collection and removal of temporary frame folders.

```
                        (build_efficientnet, "efficientnet")]:
    print(f"\nTraining {name}...")
    model = model_fn()
    train_model(model, name, train_ds, val_ds, epochs=15)

Kaggle authenticated successfully.
Downloading FF++ Cropped Faces dataset...
Using Colab cache for faster access to the 'faceforensics-videos-cropped-faces' dataset.
Dataset downloaded to: /kaggle/input/faceforensics-videos-cropped-faces
Loading and splitting dataset...
Found 19974 files belonging to 2 classes.
```

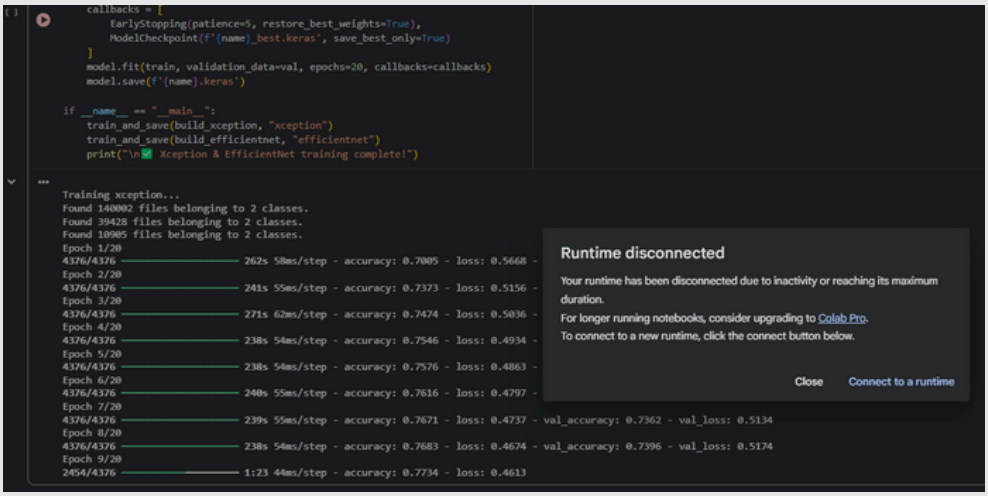
# Live Demo via Ngrok




Slecting deepfake image




Deployment and Testing: Ngrok Public URL for Deepfake Detector Web App with Successful Fake Image Detection (99.97% Confidence)




The runtime limit completed .