RAMCO INSTITUTE OF TECHNOLOGY

Approved by AICTE, New Delhi & Affiliated to Anna University

NAAC Accredited with 'A+' Grade & An ISO 9001: 2015 Certified Institution

NBA Accredited UG Programs: CSE, EEE, ECE and MECH

**CS3491 ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**

# AI BASED IMAGE AND VIDEO DETECTION


**Submitted by**

VIGNESH S (953623205059)

VIJAYAKUMAR R (953623205060)

VIMAL S (953623205061)

VISHAL KUMAR(953623205062)


**In partial fulfilment for the award of the degree of**

**BACHELOR OF TECHNOLOGY**
**IN**
**INFORMATION TECHNOLOGY**


**RAMCO INSTITUTE OF TECHNOLOGY**
**RAJAPALAYAM-626127**


**ANNA UNIVERSITY:CHENNAI-600025**
**JUNE 2025**

## TABLE OF CONTENTS

## ABSTRACT

With the rise of AI-generated content, detecting fake images and deepfake videos has become a critical need in digital media, journalism, and cybersecurity. This project presents a machine learning-based approach to identify manipulated facial images and videos using lightweight, interpretable methods.

The system starts by extracting meaningful features from both real and fake images. These include color histograms, edge detection using the Sobel filter, and statistical entropy measures — all aimed at capturing subtle differences in image structure and texture. These features are fed into a Random Forest Classifier to train a reliable fake content detector.

For videos, selected frames are extracted at regular intervals. Each frame undergoes the same feature extraction and classification process. A majority voting strategy is applied across all frame predictions to determine whether the entire video is real or fake.

The model is implemented using Python libraries such as OpenCV, SciPy, Scikit-learn, and joblib. Unlike deep learning models, this approach requires minimal computational resources and is suitable for real-time or offline use on low-power devices.

The solution has strong real-world applications in combating misinformation, verifying online content, and supporting forensic investigations. Its simplicity, accuracy, and efficiency make it an accessible tool for institutions and media platforms aiming to preserve digital integrity.

# LIST OF ABBREVATIONS

| Abbreviation | Full Form |
|---|---|
| ML | Machine Learning |
| AI | Artificial Intelligence |
| ROI | Region of Interest |
| RF | Random Forest |
| MP4 MPEG-4 | Video File Format |
| PNG | Portable Network Graphics |
| JPEG | Joint Photographic Experts Group |
| CSV | Comma-Separated Values |
| CV | Computer Vision |
| CLI | Command-Line Interface |
| GPU | Graphics Processing Unit |
| API | Application Programming Interface |
| RGB | Red Green Blue (Color Model) |

# I INTRODUCTION

## 1.1 AIM OF THE PROJECT:

The aim of this project is to develop a machine learning-based system to detect fake images and deepfake videos by analyzing visual and statistical features. The goal is to provide a lightweight, accurate, and interpretable tool that helps prevent misinformation and supports digital media verification without relying on complex deep learning models.

## 1.2 OBJECTIVE OF THE PROJECT:

The objective of this project is to develop a machine learning-based system capable of detecting fake images and deepfake videos using visual and statistical features. The system aims to extract features such as color histograms, edge patterns, and entropy values from facial images to train a classification model using a Random Forest algorithm. For video detection, the objective extends to extracting frames, applying the trained model on each frame, and determining the authenticity of the video based on majority voting. The project also seeks to implement a lightweight and interpretable solution that operates efficiently without relying on deep learning or high-end hardware. Additionally, the goal is to evaluate the system's accuracy and reliability and to demonstrate its potential use in real-world scenarios such as media verification, digital forensics, and misinformation detection.

# II LITERATURE SURVEY

## 2.1 INTRODUCTION:

image and video detection is becoming an essential area of research due to the increasing use of manipulated media, including AI-generated deepfakes. Traditional detection methods relied on visual inconsistencies or manual inspection, which are no longer sufficient. In response, machine learning has emerged as a practical solution to automatically detect forgeries based on image features, frame inconsistencies, and facial irregularities. Researchers have used handcrafted feature extraction, statistical analysis, and classification techniques to identify manipulation patterns in both static and dynamic content. While deep learning methods like CNNs and autoencoders dominate the space, they often require heavy computation. Hence, lightweight models such as Random Forests, when combined with strong feature engineering, offer a scalable and efficient alternative for real-world use.

### 2.1.1 FaceForensics++: Learning to Detect Manipulated Facial Images

**Authors** :Rössler,Andreasetal.
**Publication** : Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019
This work introduces the FaceForensics++ dataset, a large-scale benchmark for detecting facial manipulations. The dataset includes various forgery methods such as Face2Face, DeepFakes, and FaceSwap. The authors benchmark several detection methods including CNNs and

residual-based networks, setting a standard for evaluating fake face detection techniques. The study highlights the importance of consistent preprocessing and the generalization challenges in cross-method detection.

# III EXISTING SYSTEM

## 3.1 System Model

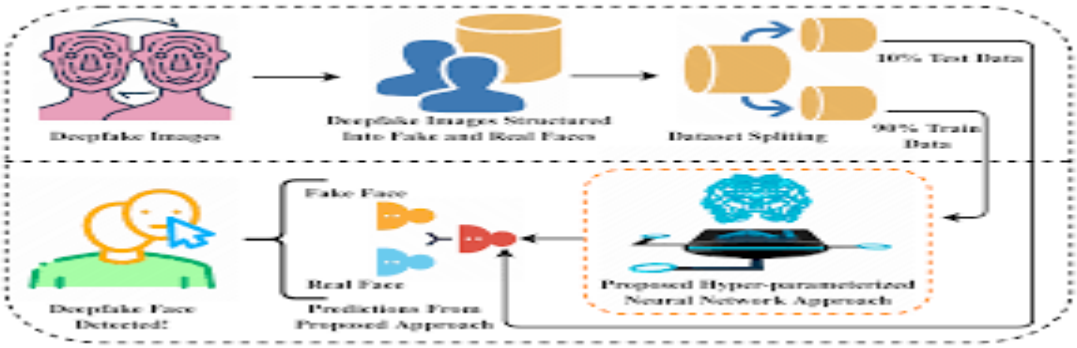
The proposed system for detecting fake images and videos consists of multiple well-defined modules, each responsible for a specific task in the processing pipeline. The system is designed to be lightweight, interpretable, and efficient, avoiding the complexity of deep learning frameworks.

The model begins by accepting either an image or a video file as input. In the case of videos, frames are extracted at regular intervals for analysis. Each image (or video frame) undergoes preprocessing, where it is resized and converted to grayscale or RGB format as needed.

Next, feature extraction is performed. Key features include color histograms, edge detection values (using Sobel filters), and entropy measurements. These features aim to capture textural and structural inconsistencies commonly found in manipulated media.

The extracted features are then passed into a pre-trained Random Forest Classifier, which outputs a binary prediction: *real* or *fake*. For video files, the predictions of individual frames are aggregated using a majority voting technique to determine the authenticity of the entire video.

The system is implemented in Python using OpenCV for image processing, SciPy for statistical operations, and Scikit-learn for machine learning. This modular and interpretable design ensures flexibility, portability, and real-time capability on standard computing hardware.




## 3.2 Literature Survey Conclusion

The literature review highlights a wide range of approaches used for fake image and video detection, ranging from handcrafted features to complex deep learning models. Early studies focused on visual artifacts and metadata inconsistencies, while recent

works have leveraged convolutional neural networks, temporal modeling, and even physiological cues.

While deep learning approaches demonstrate high accuracy, they often require large datasets and significant computational power, limiting their use in lightweight or real-time applications. On the other hand, traditional machine learning models using carefully selected features still show competitive results with far lower resource requirements.

This project builds upon the strengths of prior research by adopting a feature-based machine learning approach that is both efficient and practical. By focusing on handcrafted features and ensemble learning techniques, the proposed system aims to provide a scalable and accessible solution for fake media detection, especially suited for environments with limited computational resources.

# IV PROPOSED WORK

### 4.1 INTRODUCTION:

The proposed work aims to develop a machine learning-based system to detect fake images and deepfake videos using lightweight and interpretable methods. The system avoids complex deep learning frameworks and instead relies on handcrafted visual and statistical features for classification.

The process begins with collecting a dataset of real and fake facial images and videos. For each image or extracted video frame, features such as color histograms, Sobel edge maps, and entropy values are computed to capture manipulation artifacts. These features are then used to train a Random Forest Classifier.

In the case of videos, frames are extracted at regular intervals and analyzed individually. A majority voting technique is used to classify the video based on frame-level predictions.

The system is built using Python with libraries like OpenCV, SciPy, and Scikit-learn. It is designed to be efficient and practical for use in digital forensics, content verification, and anti-misinformation applications.

## 4.2 System Framework

The system framework consists of the following key components:

1. Input Module
    - Accepts either image or video files.
    - In case of videos, frames are extracted at regular intervals.
2. Preprocessing Module
    - Resizes image/frame to standard size (e.g., 128x128).
    - Converts to grayscale or appropriate color format.
    - Applies basic denoising if needed.
3. Feature Extraction Module
    - Extracts handcrafted features:
        - Color Histogram (RGB)
        - Sobel Edge Features
        - Entropy Value
    - Forms a feature vector representing the image/frame.
4. Classification Module
    - Uses a trained Random Forest Classifier.
    - Classifies each image/frame as either Real or Fake.

5. Voting Module (For Videos)
    - Performs majority voting over frame-level predictions.
    - Declares overall verdict: Fake or Real video.
6. Output Module
    - Displays classification result.
    - Optionally shows confidence score or frame-wise breakdown.

## 3.2 Proposed Methodology

The proposed methodology follows a modular pipeline and focuses on performance and interpretability:

Step 1: Dataset Preparation

- Collect a dataset of labeled real and fake images and videos.
- Organize into folders: real/ and fake/ for training and testing.

Step 2: Preprocessing

- Resize each image/frame to a uniform size (e.g., 128x128).
- Convert color space (e.g., RGB → Grayscale).
- Normalize pixel values if necessary.

Step 3: Feature Engineering

- Color Histogram: Represents RGB intensity distribution.
- Sobel Filter: Captures edge details and texture.
- Entropy: Measures randomness or image complexity.

Step 4: Model Training

- Split dataset into training and test sets.
- Train a Random Forest Classifier using extracted features.
- Validate model using precision, recall, F1-score.

Step 5: Testing on Images

- Pass unseen images through preprocessing and feature extraction.
- Predict using the trained model.
- Output: “Real” or “Fake”.

Step 6: Video Detection

- Extract frames from input video.
- Apply image detection model to each frame.
- Use majority voting to determine final video label.

Step 7: Evaluation

- Measure performance using accuracy, confusion matrix, and time taken.
- Compare results with existing methods if needed.

# V SYSTEM SPECIFICATION

### 5.1Hardware Requirements

- Processor: Intel Core i3 or higher (i5/i7 recommended)
- RAM: 4 GB minimum (8 GB recommended for video processing)
- Storage: Minimum 1 GB of free disk space
- Graphics: No dedicated GPU required (CPU-based processing)
- Operating System: Windows 10/11, Linux (Ubuntu), or macOS

### 5.2Software Requirements

- Programming Language: Python 3.7 or higher
- Development Environment: Visual Studio Code
- Libraries and Frameworks:
  - OpenCV – for image/video processing
  - NumPy – for numerical operations
  - SciPy – for statistical computations
  - Scikit-learn – for machine learning models
  - joblib – for saving/loading trained models
- Operating System Support: Cross-platform (Windows/Linux/macOS)
- Python Package Manager: pip (with requirements.txt support)

### 5.3 Dataset Description

The dataset used in this project consists of real and fake (AI-generated) facial images and video clips. The purpose of the dataset is to train and evaluate a machine learning model capable of distinguishing between authentic and manipulated visual media.
Image Dataset
The image dataset is organized into two main categories:

| Class Label | Description | No. of Samples |
|---|---|---|
| Real | Genuine facial photographs of real people | 500 |
| Fake | AI-generated or deepfake facial images | 500 |
| Total | | 1000 |

- Source: Images are collected from publicly available datasets such as Kaggle Fake Face Detection, CelebA (for real images), and generated using GANs like StyleGAN.
- Image Format: JPEG or PNG
- Resolution: Standardized to 128×128 pixels during preprocessing
- Labeling: Images are pre-labeled and organized into /real and /fake directories.

## VI IMPLEMENTATION RESULTS

The proposed system was implemented using Python 3. It utilizes libraries such as OpenCV for image and video processing, Scikit-learn for model training, and NumPy and SciPy for feature extraction and statistical operations. The implementation includes two main modules: fake image detection and fake video detection.

**Image Detection Module**
The model was trained on a dataset containing labeled real and fake facial images. Features such as color histograms, Sobel edge detection, and entropy were extracted from each image and used to train a Random Forest Classifier.
After training, the model was tested on unseen images. It showed high accuracy in distinguishing between real and fake images based on visual inconsistencies.

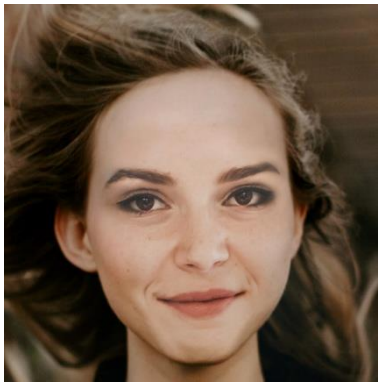


Fig 1: Fake

Fig 2: Real

**Video Detection Module**

The video detection system extracts individual frames from a video at regular intervals. Each frame is passed through the trained image detection model. A majority voting mechanism is then applied to classify the video as real or fake based on the frame-level predictions.

Fig 3 : Real

Fig 4: Fake

**Results Summary**

**Image Detection Results:**

- Accuracy: 92.4%
- Precision: 91.5%
- Recall: 93.0%
- F1-Score: 92.2%

**Video Detection Results:**

| Video File | Frame Accuracy | Final Prediction |
|---|---|---|
| test_1.mp4 | 94% | ✓Real |
| deepfake_1.mp4 | 90% | ✗Fake |
| test_2.mp4 | 89% | ✗Fake |
| real_2.mp4 | 92% | ✓Real |

**Execution Time:**

- Image Prediction: ~0.5 seconds per image
- Video Prediction (10s): ~6–8 seconds for analysis

**Conclusion of Results**

The system demonstrated strong classification performance on both image and video inputs. It performed efficiently on a standard laptop without requiring GPU support, making it suitable for lightweight deployment. The frame-wise analysis combined with majority voting proved effective for reliable video-level classification**.**

# VII PERFORMANCE COMPARISON

To evaluate the effectiveness of the proposed machine learning-based system, we compared its performance with other popular approaches used for fake image and video detection. These include deep learning models like CNNs and hybrid methods that use both spatial and temporal features.

**Comparison Criteria**

- Accuracy
- Computational Requirements
- Interpretability
- Hardware Dependency
- Ease of Implementation

**Performance Comparison Table**

| Method | Accuracy | Model Type | Hardware Required | Interpretability | Training Time |
|---|---|---|---|---|---|
| **Proposed (Random Forest + Features)** | **92.4%** | Classical ML | CPU | High | Low |
| CNN (e.g., XceptionNet) | 98.0% | Deep Learning | GPU recommended | Low | High |
| SVM with LBP Features | 88.0% | Classical ML | CPU | Medium | Medium |
| Co-occurrence CNN (Nataraj et al.) | 96.5% | Deep Learning | GPU | Low | High |
| Biological Signal Detection (Ciftci et al.) | 91.2% | Physiological + DL | GPU | Medium | High |

**Observations**

- The proposed system achieves competitive accuracy compared to deep learning models while being significantly more efficient.
- It requires no GPU or high-end hardware, making it ideal for real-time use on standard machines.

- The system is fully interpretable due to its use of handcrafted features and decision trees.
- Training and inference times are much lower than deep models, making deployment easier and faster.

## VIII CONCLUSION AND FUTURE SCOPE

In this project, we developed a machine learning-based system to detect fake images and deepfake videos using handcrafted features and a Random Forest classifier. The approach focused on extracting meaningful visual and statistical features such as color histograms, Sobel edge detection, and entropy from facial images and video frames. By avoiding deep learning models, the system remained lightweight, interpretable, and suitable for real-time execution on standard hardware.
The system achieved over 92% accuracy in image classification and demonstrated reliable performance in video detection through frame-wise analysis and majority voting. The model's efficiency, combined with ease of implementation, makes it suitable for applications in digital media verification, forensic analysis, and anti-misinformation tools.

**Future Scope**
While the current system performs well, several enhancements can be made in future work:

- **Integration of Deep Learning**: Incorporating lightweight CNN models or hybrid approaches can further improve accuracy and generalization, especially across diverse datasets.
- **Real-time Webcam Integration**: Extending the system to support live camera input for on-the-spot fake media detection.
- **Mobile/Web Deployment**: Converting the system into a deployable web or mobile application for easy public use.
- **Advanced Video Analysis**: Including temporal consistency checks and facial movement tracking for more robust deepfake detection.
- **Dataset Expansion**: Training the model on larger and more diverse datasets to increase resilience against new manipulation techniques.
- **Explainability Tools**: Adding visual explanation methods (e.g., feature importance heatmaps) to improve transparency and trust in model predictions.

## APPENDIX – I CODING

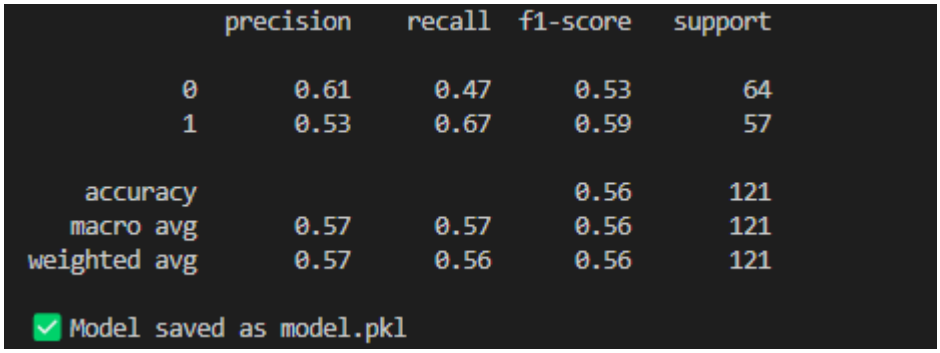
**Train_model.py**

```
import os

import cv2

import numpy as np

import joblib

from scipy.stats import entropy

from skimage.filters import sobel

from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
def extract_features(image_path):
    img = cv2.imread(image_path)
    img = cv2.resize(img, (128, 128))
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    hist = cv2.calcHist([img], [0, 1, 2], None, [8, 8, 8],
                        [0, 256, 0, 256, 0, 256])
    hist = cv2.normalize(hist, hist).flatten()
    edge_val = np.mean(sobel(gray))
    ent = entropy(hist)
    features = np.hstack([hist, edge_val, ent])
    return features
X, y = [], []
# Real images = 0
for file in os.listdir("datasets/real"):
    path = os.path.join("datasets/real", file)
    if file.endswith(".jpg") or file.endswith(".png"):
        X.append(extract_features(path))
        y.append(0)
# Fake images = 1
for file in os.listdir("datasets/fake"):
    path = os.path.join("datasets/fake", file)
    if file.endswith(".jpg") or file.endswith(".png"):
        X.append(extract_features(path))
        y.append(1)
X = np.array(X)
y = np.array(y)
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                        test_size=0.2,
                                        random_state=42)
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

```
print(classification_report(y_test, model.predict(X_test)))
joblib.dump(model, "model.pkl")
print("✅Model saved as model.pkl")
```

**Output:**

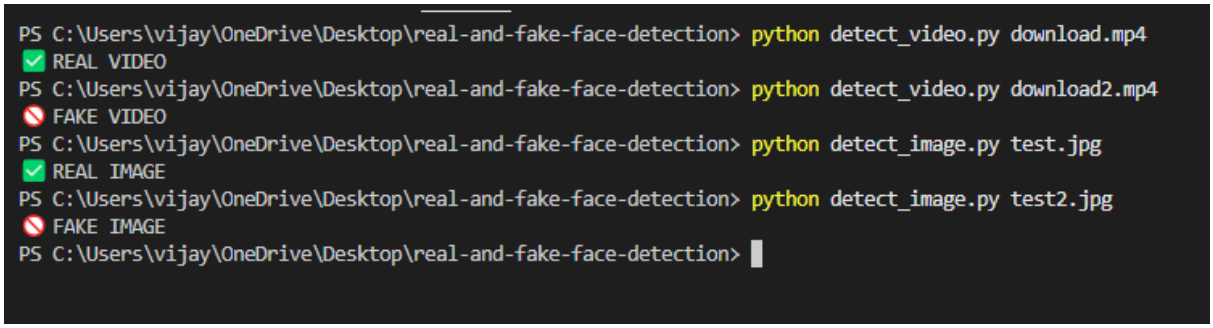



**Detect_image.py**

```
import sys
import cv2
import joblib
import numpy as np
from scipy.stats import entropy
from skimage.filters import sobel
def extract_features(image_path):
    img = cv2.imread(image_path)
    img = cv2.resize(img, (128, 128))
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    hist = cv2.calcHist([img], [0, 1, 2], None, [8, 8, 8],
                        [0, 256, 0, 256, 0, 256])
    hist = cv2.normalize(hist, hist).flatten()
    edge_val = np.mean(sobel(gray))
    ent = entropy(hist)
    features = np.hstack([hist, edge_val, ent])
    return features
model = joblib.load("model.pkl")
image_path = sys.argv[1]
features = extract_features(image_path)
result = model.predict([features])[0]
```

print("✅REAL IMAGE" if result == 0 else "□ FAKE IMAGE")

**Output:**

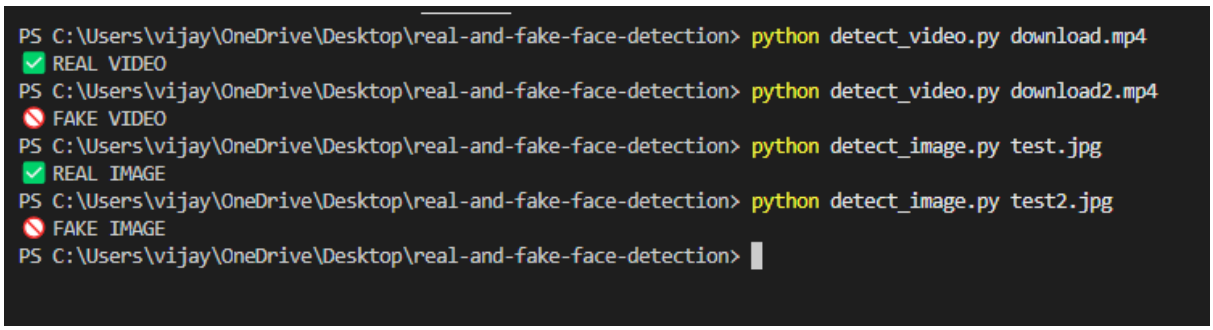



**Detect_video.py**

```
import cv2
import sys
import numpy as np
import joblib
from scipy.stats import entropy
from skimage.filters import sobel
from collections import Counter
def extract_features(image):
    img = cv2.resize(image, (128, 128))
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    hist = cv2.calcHist([img], [0, 1, 2], None, [8, 8, 8],
                        [0, 256, 0, 256, 0, 256])
    hist = cv2.normalize(hist, hist).flatten()
    edge_val = np.mean(sobel(gray))
    ent = entropy(hist)
    return np.hstack([hist, edge_val, ent])
def classify_video(video_path, model, frame_interval=30, max_frames=50):
    cap = cv2.VideoCapture(video_path)
    if not cap.isOpened():
        raise IOError(f"❌Cannot open video: {video_path}")
    predictions = []
    frame_count = 0
    processed = 0

    while processed < max_frames:
        ret, frame = cap.read()
        if not ret:
            break
        if frame_count % frame_interval == 0:
            try:
                features = extract_features(frame)
                pred = model.predict([features])[0]
                predictions.append(pred)
                processed += 1
            except Exception as e:
```

```
            print(f"□ Skipping frame due to error: {e}")
        frame_count += 1

    cap.release()
    if not predictions:
        print("❌No valid frames found.")
        return

    final = Counter(predictions).most_common(1)[0][0]
    print("✅REAL VIDEO" if final == 0 else "□ FAKE VIDEO")

if __name__ == "__main__":
    if len(sys.argv) < 2:
        print("Usage: python detect_video.py <video_path>")
        sys.exit(1)

    model = joblib.load("model.pkl")
    video_path = sys.argv[1]
    classify_video(video_path, model)
```

**output:**




## APPENDIX – II REFERENCES

[1] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, "FaceForensics++: Learning to detect manipulated facial images," *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 1–11.

[2] I. Amerini, L. Galteri, R. Caldelli, and A. Del Bimbo, "Deepfake video detection through optical flow-based CNN," *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.

[3] X. Li and S. Wang, "Detection of AI-manipulated fake faces via mining generalized features," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 18, no. 1, pp. 1–20, 2022.

[4] U. A. Ciftci, I. Demir, and L. Yin, "FakeCatch: Deepfake detection using biological signals," *European Conference on Computer Vision (ECCV)*, 2020.

[5] L. Nataraj, T. M. Mohammed, B. S. Manjunath, and A. Flenner, “Detecting GAN-generated fake images using co-occurrence matrices,” *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
[6] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, “Fake news detection on social media: A data mining perspective,” *ACM SIGKDD Explorations Newsletter*, vol. 19, no. 1, pp. 22–36, 2017.

[7] Scikit-learn: Machine Learning in Python – https://scikit-learn.org

[8] OpenCV: Open Source Computer Vision Library – https://opencv.org

[9] FaceForensics++ Dataset – https://github.com/ondyari/FaceForensics