IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

VID VERIFY-DEEPFAKE VIDEO DETECTION

[1]B. Chaithanya Lakshmi, [2]M. Sathwika, [3]S. Maneesha, [4]T. Tagore Ravi Chandra ,[5]Bobby K. Simon,

[1,2,3,4]Students [5]Assistant Professor Computer Science and Engineering (Cyber Security),

Hyderabad Institute of Technology and Management,
Medchal, Hyderabad, Telegana, India

Abstract: This paper presents a system based on Convolutional Neural Networks (CNNs) for detecting deepfake videos. It tackles the growing threat of AI-generated visual manipulation. Deepfake technology allows for the creation of very realistic videos that can spread false information, lead to identity theft, and harm reputations. The proposed system examines extracted video frames to find pixel-level inconsistencies and visual flaws that separate real content from fake. By using a lightweight, frame-based CNN architecture, the model achieves high accuracy and keeps computational demands low. This approach addresses the shortcomings of traditional deepfake detection methods.
The system is trained on various datasets that include both authentic and manipulated videos. This helps improve its ability to generalize across different deepfake generation techniques. Experimental results show strong performance and scalability, confirming the model's ability to accurately detect deepfakes. This method provides a practical and efficient way to fight the misuse of deepfake technology in areas like cybersecurity, media verification, and digital forensics.

Keywords: Deepfake Detection, Convolutional Neural Networks (CNN), Artificial Intelligence (AI), Video Forensics, Digital Media Security.

I. Introduction

The rapid rise of synthetic media, particularly through deepfake technology using GANs and autoencoders, has made it harder to tell real videos from fake ones. These realistic manipulations present serious risks in politics, journalism, entertainment, and personal safety by enabling identity theft, misinformation, and false evidence. Traditional detection methods that look for visual flaws, like unnatural blinking or inconsistent lighting, are becoming less effective as deepfake algorithms improve. Many CNN-based models also fail to capture broader contextual features, which hurts their detection accuracy.

Vid-Verify tackles these challenges with a strong, frame-based deepfake detection system that merges computer vision with deep learning. Its tailored CNN architecture is designed to spot signs of pixel-level manipulation while keeping efficiency. By using thorough preprocessing and frame extraction techniques, Vid-Verify guarantees reliable performance across different video qualities and deepfake styles. The project's key contributions include creating a complete detection framework, optimizing the CNN architecture, and assessing model performance with standardized datasets and metrics.

II. LITERATURE SURVEY

R. Tolosana et al. [1] offered a detailed look at face manipulation techniques and fake detection methods. They covered the growth of deepfake generation and the related detection methods. This important work highlights the ongoing challenges in keeping detection accurate, especially with the fast pace of new synthetic media technologies. The authors Gupta, Rahul, Cheshtha Kapoor, and Jayesh Yadav [2] researched faceless, paperless, and cashless transaction systems within digital payment ecosystems. As digital transformation speeds up, similar issues in authentication and security arise in both financial transactions and media verification systems. These issues call for effective validation mechanisms. Ghosh, Gourab [3] studied how the spread of smartphones and internet access has changed digital services, creating both opportunities and challenges for secure online interactions. This wave of digitalization has not only changed payment systems but has also built the framework for widespread synthetic media distribution. Manvita Joshi [4] developed a mobile-based secure transaction framework that shields users from fraud through additional security layers. Similar security structures are key in deepfake detection systems to stop manipulation and ensure reliable media authentication.

Yuvaraj, S., and N. Sheila Eveline [5] looked at the shift from physical to digital transactions, highlighting how traditional methods are being replaced by more advanced digital platforms. This digital change mirrors the move from basic video editing to AI-generated synthetic media, which needs effective detection methods. Miruna, S. Lyrics [6] showed how smartphone technology has transformed daily transactions through digital wallet platforms. These create new ways for secure online interactions that share important security issues with media verification systems. Cherukur, Mr. [7] researched user satisfaction with mobile wallets and found key factors that shape user trust and acceptance of digital platforms. These trust factors also matter in deepfake detection systems, where user confidence in verification results affects practical use. Dewi Sandy Islamiati [8] set up and tested a payment system using mobile apps and REST API web services. She assessed system performance through careful user testing. Similar evaluation methods are vital for confirming deepfake detection systems in real-world situations.

Sometimes, detection systems struggle against advanced manipulation techniques. This makes strong deepfake detection applications essential so users can verify media authenticity without needing expert help. Anshari, Muhammad, et al. [10] showed that digital platforms can improve productivity for institutions while offering convenience for users. The younger generation's comfort with digital applications that have simple interfaces gives useful insights for creating user-friendly deepfake detection tools that can gain broad use across different user groups.

III. EXISTING SYSTEM

The deepfake detection field has seen major improvements in how it works and detects fake content over the last few years. The rapid growth of artificial intelligence and digital media has led to the creation of very realistic fake content, putting a strain on current detection systems. This has caused an increase in manipulated videos, online misinformation, and difficulties in identifying real digital content. Countries around the world are now using machine learning and deep learning techniques to regularly address this issue. These methods search for different patterns and inconsistencies in multimedia databases and provide valuable insights for organizations and security agencies. However, it is clear that most existing systems still rely on traditional Convolutional Neural Network (CNN) architectures. These face challenges like high computational costs, inflexibility, and inefficiency in managing large-scale data. Before 2024, experts predicted a significant rise in deepfake content, with millions of manipulated samples appearing online every day. This has created an urgent need for better, more reliable detection models.

Limitation of existing system:

1. Traditional CNN-based methods for detecting deepfakes are **costly, imprecise, and time-consuming**.
2. These systems focus mainly on **individual frames** and fail to capture **temporal inconsistencies** across video sequences.
3. Existing models **lack generalization** to new and unseen manipulation techniques, reducing their effectiveness in real-world applications.

IV. PROPOSED SYSTEM

In today's digital world, video content is everywhere, from social media to news platforms. However, with new AI technology, it is easy to create fake videos that look completely real. These altered videos, called deepfakes, can spread false information, harm reputations, and even pose a threat to national security. In this context, our Vid Verify-AI system becomes crucial for journalists, social media companies, and regular users who want to verify video content.

This project offers a simple interface and effective methods to detect video manipulation, but its real strength lies in its technical features. You can upload any video to the system, which will automatically analyze each frame, look for signs of manipulation, and provide a detailed report on whether the video is real or fake. The system generates clear results, showing which parts of the video may have been altered and how confident it is in its findings. This allows you to quickly determine if a video has been manipulated and to save the analysis report for later use. The main benefit of this system is that it works automatically; users just need to upload the video while the AI system handles the complex analysis.

For this deepfake detection project, we use Python as our main programming language. Python is designed for AI and machine learning projects, providing a wide range of tools and libraries that help developers create, test, and improve AI models efficiently. Using this system can significantly lower the risks associated with fake videos by offering a reliable verification tool, which reduces the need for manual video analysis by experts. The system allows users to quickly check suspicious videos on their own computers or mobile devices. This convenience is especially helpful for news organizations and security agencies that need to verify video content promptly. The system enables fast analysis, cutting down the time and effort needed compared to traditional verification methods. This efficiency leads to quicker fact-checking and better decision-making. The software has two main phases: the first is data preparation and model training, and the second is actual use and testing. For the AI component, we use the TensorFlow and Keras libraries, and to make it user-friendly, we can use Streamlit or Flask frameworks. Python is the most popular language for AI development and is widely used by developers and researchers around the world. It continues to improve with regular updates to support the latest AI technologies and tools.

4.1 How to Create the Deepfake Detection System?

We use powerful development environments like PyCharm or VS Code, specifically made for writing, testing, and debugging Python code. All necessary components are available for download through package managers like pip. These are collections of tools and libraries that are regularly updated and can be installed separately. To create the deepfake detection system, you can follow these steps:

1. **Set up Python Environment**: Download and install the latest Python version from the official website (https://www.python.org/downloads/). Ensure your computer meets the requirements for AI projects, including a good graphics card for faster processing.
2. **Install Required Libraries**: Open command prompt and use pip to install essential components including TensorFlow, Keras, OpenCV, Matplotlib, and MTCNN.
3. **Create New Project**: After setting up the environment, create a new project folder in your development software. Organize the folder with separate sections for data, program code, and saved AI models.
4. **Choose Project Structure**: The project is divided into key parts: data preparation, model creation, training, and testing.
5. **Design Data Processing**: This involves writing code for frame extraction and preparation using OpenCV. You'll create functions to read videos, take pictures from videos at specific intervals, detect and focus on faces using MTCNN, and prepare the image data.

6. **Write Model Code**: Using TensorFlow/Keras, you'll write code to build the Convolutional Neural Network (CNN) structure. This involves adding layers like Conv2D, MaxPooling2D, Dropout, and Dense, and setting up the model with an optimizer (like Adam) and a measurement system.
7. **Train Your Model**: Use the prepared data to teach the CNN model. The system will learn through multiple cycles, gradually learning to tell real frames from fake ones. You can watch the learning progress using accuracy measurements.
8. **Test Your Model**: After training, the model is tested on new videos it hasn't seen before. Run the model on these videos to make sure it works correctly and can handle new content.
9. **Debug and Fix Issues**: If you face problems like the model not learning properly or making too many mistakes, use methods like adding more varied data, changing the model structure, or adjusting settings to identify and solve these issues.
10. **Build Final System**: Once the model is ready, it can be saved and built into a simple web application using frameworks like Streamlit, which creates an easy-to-use interface for end users.
11. **Deploy Your System**: You can make the system available for use on local computers, on servers, or through online platforms, making it accessible for video verification purposes.

4.1.1 **Advantage of Proposed System**

1. The system shows the exact confidence level for each frame that is fake and provides a final decision for the entire video.
2. It uses a special Convolutional Neural Network (CNN) algorithm to identify subtle signs of video manipulation.
3. There are no hidden charges for using the system; users can analyze videos as many times as they want.
4. It reduces the spread of false information and fake video content through constant and automated checks.
5. The main technology can be used in cybersecurity, journalism, legal evidence checking, and social media content monitoring.

4.1.2 Impact of Proposed System

1. Bringing change to how digital media is verified requires shifting from human checking to AI-powered analysis.
2. This creates opportunities to develop improved versions that can detect more complex manipulation techniques.
3. We are building a new tool for digital investigation to strengthen protection against AI-generated threats.
4. We are also establishing a new, secure, and detailed process for confirming visual information in the digital world.

V. SYSTEM ARCHITECTURE

1. This deepfake detection system is designed for universal accessibility. It serves journalists, social media platforms, law enforcement agencies, and general users who need to verify video authenticity.

2. The user is the first input module. All subsequent detection processes are automatically handled by the AI system, so no technical expertise is required.

3. Users start by uploading video files through a simple web interface. The system checks the file format, size, and compatibility before proceeding with analysis.

4. The system dashboard offers multiple options, including Video Upload, Real-time Analysis, Historical Reports, and Export Features for thorough video verification.

5. When users begin the upload process, they can select video files from various sources such as local storage, cloud platforms, or direct URL links for online videos.

6. After a successful upload, the system automatically performs Frame Extraction. It breaks the video into individual frames at optimized intervals to balance processing efficiency and detection accuracy.

7. The extracted frames go through Face Detection using MTCNN algorithms, which accurately identifies and isolates facial regions in each frame for focused analysis.

8. Feature Extraction is the main analytical phase. Here, the pre-trained CNN model examines each facial region for subtle manipulation artifacts, unnatural patterns, and digital inconsistencies.

9. During the Classification stage, the system processes extracted features through deep learning algorithms. It classifies each frame as authentic or manipulated, along with corresponding confidence scores.

10. The system automatically combines frame-level results to generate video-level conclusions. It provides users with clear indicators of video authenticity and specific tampering evidence.

11. Results are organized in the database and shown through an intuitive dashboard. This dashboard highlights manipulated segments with visual evidence and statistical confidence metrics.

12. The interface distinguishes between authentic and manipulated content with color-coded indicators: green for verified authentic content and red for detected manipulations, along with detailed explanations.

13. To improve user experience, the system offers an interactive timeline visualization, showing exactly where and when manipulations occur within the video's duration.

14. Users can access frame-specific analysis with side-by-side comparisons. This highlights detected anomalies and provides technical explanations of identified manipulation techniques.

15. The system includes multiple automated quality checks. These checks involve duplicate frame identification, compression artifact analysis, and consistency verification across sequential frames.

16. Key computational functions include Real-time Probability Calculation and Multi-dimensional Confidence Scoring. These are the basis of the system's detection reliability.

17. Once the analysis is complete, the system automatically compiles a Digital Forensic Report. This report contains statistical summaries, visual evidence, technical analysis, and expert interpretations.

18. All analytical results are processed by the report generation engine and securely stored in the system database. This ensures data integrity and availability for future reference.

19. The system uses multi-layer verification protocols, including cross-frame consistency checks, temporal pattern analysis, and algorithm validation to minimize false positives and improve detection accuracy.

20. Users can choose between Detailed Forensic Reports and Executive Summary Reports based on their needs. Detailed reports provide in-depth analysis, while concise summaries serve quick verification purposes.

21. The final output capability allows users to export complete analysis reports in multiple formats such as PDF, JSON, and XML. These reports include all analytical data, visual evidence, confidence metrics, and expert conclusions for legal or archival purposes.

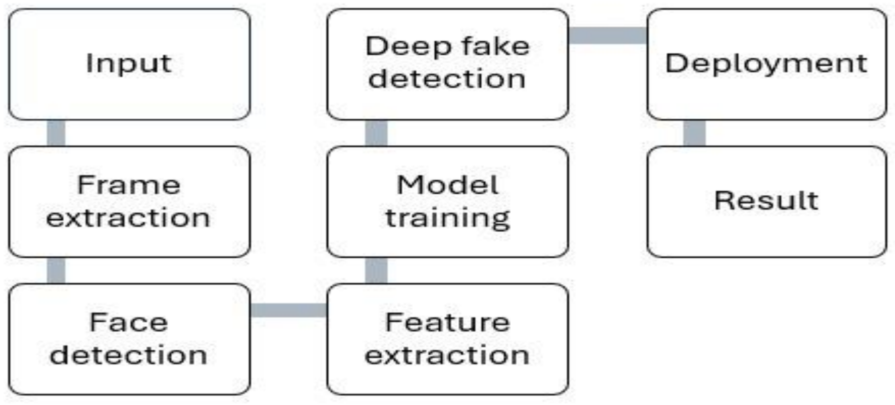


Fig.1 Architecture of Deepfake video detection

VI. ALGORITHM

Algorithms play a crucial role in the deepfake detection system, contributing to several key functionalities and features including frame processing, feature extraction, classification, and result analysis. Here are some of the primary uses of algorithms in the Vid Verify-AI system:

6.1 Convolution neural network (CNN)

Convolutional Neural Networks are the main tools for analyzing visual features in video frames. The CNN acts as a feature extractor that automatically learns patterns from input images. This makes it perfect for spotting small manipulation artifacts in deepfake videos.

```
model = tf.keras.Sequential([
Conv2D(32, (3,3), activation='relu', input_shape=(224,224,3)),
MaxPooling2D(2,2),
Conv2D(64, (3,3), activation='relu'),
MaxPooling2D(2,2),
Conv2D(128, (3,3), activation='relu'),
MaxPooling2D(2,2),
Flatten(),
Dense(128, activation='relu'),
Dense(2, activation='softmax') # Real vs Fake
])
```

6.1.1 Frame Processing CNN

In this deepfake detection project, we apply CNN architecture several times for various processing stages. The main CNN model is used to extract features and classify video frames. CNNs serve as a link between raw pixel data and useful feature representations. This enables the system to recognize manipulation patterns in an organized way.

```
  frame = cv2.resize(frame, (224,224))
  frame = frame / 255.0  # Normalize
```

6.1.2 Feature Extraction Algorithm

CNNs effectively handle hierarchical feature learning to improve detection accuracy and processing performance. In a deepfake detection pipeline, several convolutional layers are stacked at the beginning. As the network gets deeper, the feature extractor learns more complex patterns, rather than looking at each feature separately. This method helps increase detection accuracy and lower false positives.

```
features = [
    Conv2D(32,3),  # Edge detection
    Conv2D(64,3),  # Texture patterns
    Conv2D(128,3)  # Complex artifacts
]
```

6.1.3 Result Aggregation Algorithm

CNNs offer flexibility in handling various input sizes and aspect ratios with adaptive pooling layers. You can create custom preprocessing that resizes input frames to the necessary dimensions. This allows you to process videos of different resolutions based on your needs. It also lets you manage different video qualities and formats within the same detection pipeline. CNNs can be combined with data augmentation methods like random cropping or rotation to improve the model's ability to handle variations in input data. This makes the training process easier and removes the need for manual video normalization.

```
final_prediction = np.mean(frame_predictions)
confidence = np.std(frame_predictions)
```

6.2 MTCNN Face Detection

MTCNN (Multi-task Cascaded Convolutional Networks) is a popular choice for face detection in deepfake analysis because it offers precise facial region localization and landmark detection. Here are some reasons why MTCNN is commonly used in deepfake detection systems.

```
detector = MTCNN()
faces = detector.detect_faces(frame)
```

6.2.1 Face Detection Pipeline

Create a multi-stage network that builds on the CNN architecture for specific face detection tasks. This deepfake detection system uses MTCNN in several ways: mainly for face detection, identifying facial landmarks, and aligning faces. MTCNN comes as pre-trained models in most deep learning frameworks, so there is no need to train from scratch. It is easy to access and can be used directly in this detection pipeline.

```
detector = MTCNN()
results = detector.detect_faces(frame)
```

6.2.2 Facial Landmark Detection

MTCNN provides support for facial landmark detection features such as eye corners, nose tip, and mouth corners. It allows you to precisely localize key facial points and establish spatial relationships between them, making it easier to analyze facial geometry and detect anomalies. MTCNN supports multi-task learning, which allows it to simultaneously perform face detection
and landmark localization in a single forward pass.

```
frames = []
cap = cv2.VideoCapture(video_path)
while cap.isOpened():
    ret, frame = cap.read()
    if ret:
        frame = cv2.resize(frame, (224,224))
        frames.append(frame)
```

6.2.3 Face Alignment Algorithm

MTCNN has been thoroughly tested and shows strong accuracy and reliability in facial analysis tasks. It is commonly used in computer vision and has a large community of researchers who offer support and resources. Overall, MTCNN provides an easy and effective way to detect and align faces in video frames, which makes it a favored option for developers creating deepfake detection systems. MTCNN is fine-tuned for precise face detection and landmark localization. It performs well across different poses, lighting conditions, and occlusions, allowing for dependable facial analysis in various situations.

```
detector = MTCNN()
results = detector.detect_faces(frame)
```

VII. WORKING MODEL OF THE PROJECT

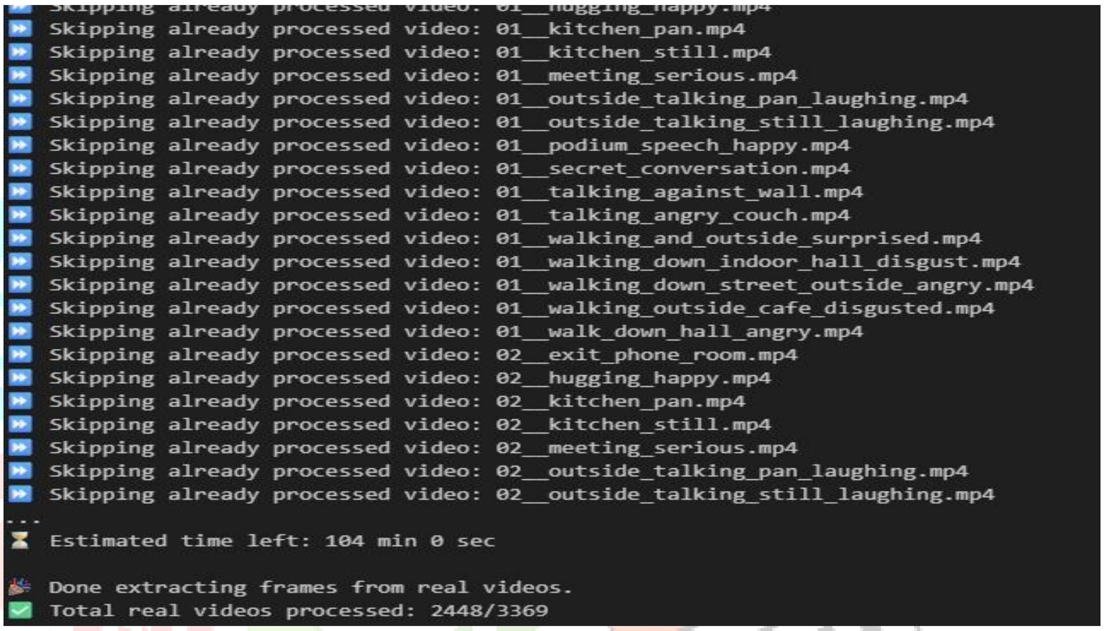
The working model of the VID-VERIFY project is designed as an end-to-end pipeline for automated deepfake detection, ensuring the authenticity of digital content through a systematic process. The operation can be broken down into sequential steps for clarity.

Step 1: Data Collection and Preprocessing: Diverse videos, both real and deepfake, are gathered from public datasets and other sources. Each input video is decomposed into its constituent frames. These frames are then preprocessed by resizing to a uniform dimension (e.g., 224x224 pixels), normalizing pixel values, and applying enhancement techniques to highlight potential artifacts for more effective model analysis.

Step 2: Model Training and Validation: The preprocessed frames are split into training (70%), validation (15%), and testing (15%) sets. The core CNN model is trained on the training set, learning to distinguish between authentic and manipulated frames by identifying pixel-level anomalies and spatial artifacts like unnatural facial contours or texture inconsistencies. The validation set is used to fine-tune hyperparameters and prevent overfitting, ensuring the model generalizes well to unseen data.

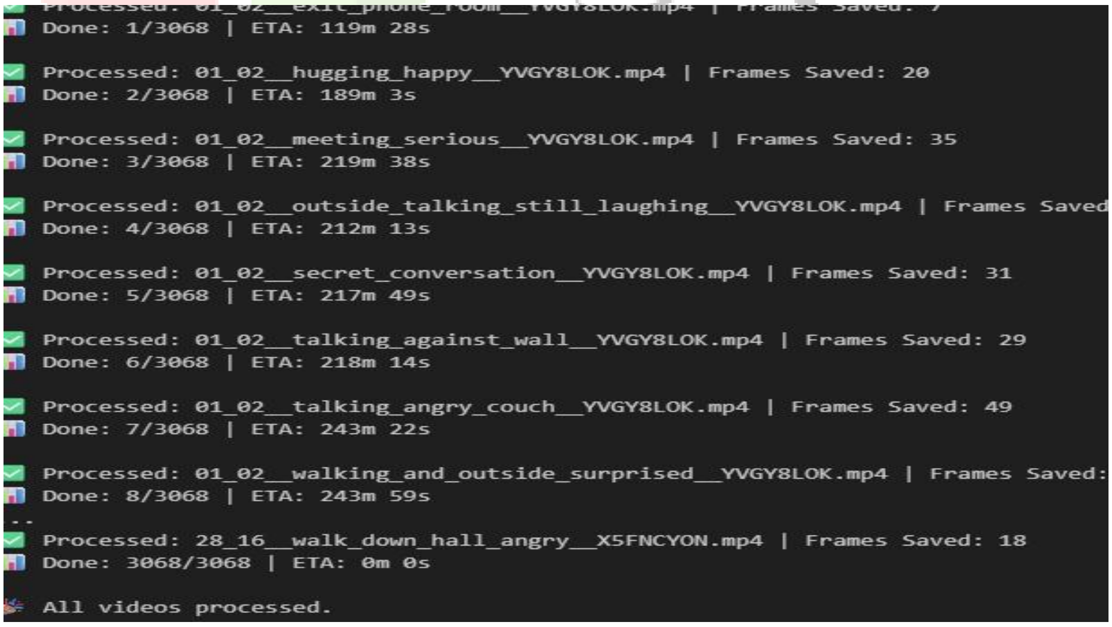
Step 3: Frame Classification and Aggregation: During inference, each frame from a test video is passed through the trained CNN model for individual classification. The model outputs a probability score for each frame. These frame-level predictions are then aggregated using a majority voting system to produce a final, holistic video-level prediction—classifying the entire video as "Real" or "Fake."

Step 4: Output and Evaluation: The system generates a final detection report stating the classification result. For deeper analysis, it can optionally provide visual feedback by highlighting frames where manipulation artifacts were most detected. The model's performance is rigorously evaluated using standard metrics including accuracy, precision, recall, and F1-score on the held-out test set.



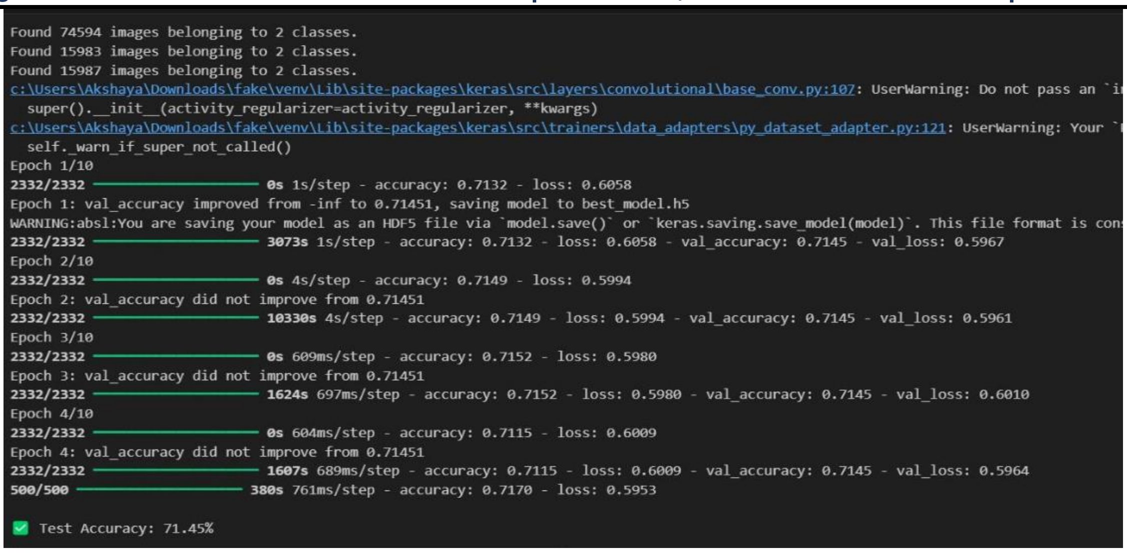
```
⏩ Skipping already processed video: 01__hugging_happy.mp4
⏩ Skipping already processed video: 01__kitchen_pan.mp4
⏩ Skipping already processed video: 01__kitchen_still.mp4
⏩ Skipping already processed video: 01__meeting_serious.mp4
⏩ Skipping already processed video: 01__outside_talking_pan_laughing.mp4
⏩ Skipping already processed video: 01__outside_talking_still_laughing.mp4
⏩ Skipping already processed video: 01__podium_speech_happy.mp4
⏩ Skipping already processed video: 01__secret_conversation.mp4
⏩ Skipping already processed video: 01__talking_against_wall.mp4
⏩ Skipping already processed video: 01__talking_angry_couch.mp4
⏩ Skipping already processed video: 01__walking_and_outside_surprised.mp4
⏩ Skipping already processed video: 01__walking_down_indoor_hall_disgust.mp4
⏩ Skipping already processed video: 01__walking_down_street_outside_angry.mp4
⏩ Skipping already processed video: 01__walking_outside_cafe_disgusted.mp4
⏩ Skipping already processed video: 01__walk_down_hall_angry.mp4
⏩ Skipping already processed video: 02__exit_phone_room.mp4
⏩ Skipping already processed video: 02__hugging_happy.mp4
⏩ Skipping already processed video: 02__kitchen_pan.mp4
⏩ Skipping already processed video: 02__kitchen_still.mp4
⏩ Skipping already processed video: 02__meeting_serious.mp4
⏩ Skipping already processed video: 02__outside_talking_pan_laughing.mp4
⏩ Skipping already processed video: 02__outside_talking_still_laughing.mp4
...
⏳ Estimated time left: 104 min 0 sec

🎉 Done extracting frames from real videos.
✅ Total real videos processed: 2448/3369
```

Fig2. Real Frame Extraction



```
✅ Processed: 01_02__exit_phone_room__YVGY8LOK.mp4 | Frames Saved: 7
📊 Done: 1/3068 | ETA: 119m 28s

✅ Processed: 01_02__hugging_happy__YVGY8LOK.mp4 | Frames Saved: 20
📊 Done: 2/3068 | ETA: 189m 3s

✅ Processed: 01_02__meeting_serious__YVGY8LOK.mp4 | Frames Saved: 35
📊 Done: 3/3068 | ETA: 219m 38s

✅ Processed: 01_02__outside_talking_still_laughing__YVGY8LOK.mp4 | Frames Saved
📊 Done: 4/3068 | ETA: 212m 13s

✅ Processed: 01_02__secret_conversation__YVGY8LOK.mp4 | Frames Saved: 31
📊 Done: 5/3068 | ETA: 217m 49s

✅ Processed: 01_02__talking_against_wall__YVGY8LOK.mp4 | Frames Saved: 29
📊 Done: 6/3068 | ETA: 218m 14s

✅ Processed: 01_02__talking_angry_couch__YVGY8LOK.mp4 | Frames Saved: 49
📊 Done: 7/3068 | ETA: 243m 22s

✅ Processed: 01_02__walking_and_outside_surprised__YVGY8LOK.mp4 | Frames Saved:
📊 Done: 8/3068 | ETA: 243m 59s
..
✅ Processed: 28_16__walk_down_hall_angry__X5FNCYON.mp4 | Frames Saved: 18
📊 Done: 3068/3068 | ETA: 0m 0s

🎉 All videos processed.
```

Fig3. Fake Frame Extraction

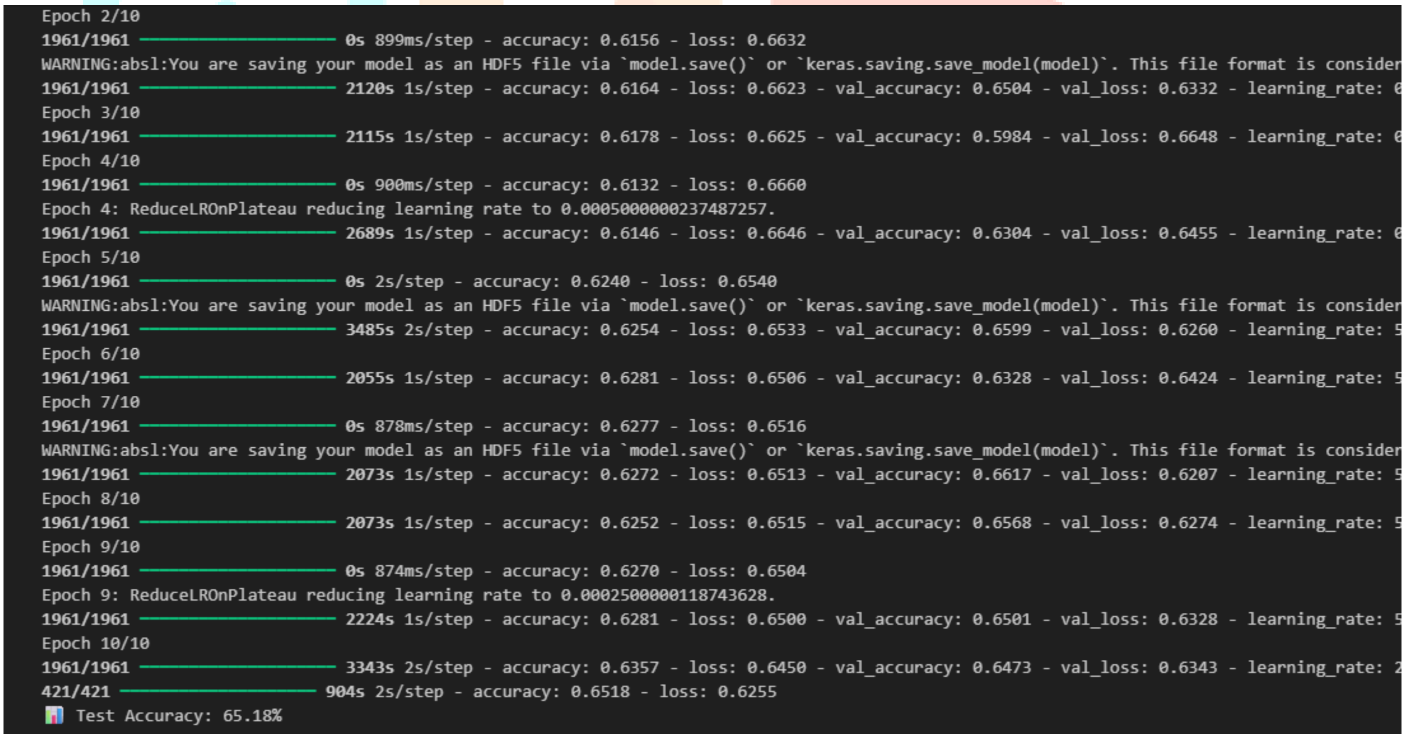


```
Found 74594 images belonging to 2 classes.
Found 15983 images belonging to 2 classes.
Found 15987 images belonging to 2 classes.
c:\Users\Akshaya\Downloads\fake\venv\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `i
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
c:\Users\Akshaya\Downloads\fake\venv\Lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `
  self._warn_if_super_not_called()
Epoch 1/10
2332/2332 ━━━━━━━━━━━━━━━━━━━━ 0s 1s/step - accuracy: 0.7132 - loss: 0.6058
Epoch 1: val_accuracy improved from -inf to 0.71451, saving model to best_model.h5
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is con
2332/2332 ━━━━━━━━━━━━━━━━━━━━ 3073s 1s/step - accuracy: 0.7132 - loss: 0.6058 - val_accuracy: 0.7145 - val_loss: 0.5967
Epoch 2/10
2332/2332 ━━━━━━━━━━━━━━━━━━━━ 0s 4s/step - accuracy: 0.7149 - loss: 0.5994
Epoch 2: val_accuracy did not improve from 0.71451
2332/2332 ━━━━━━━━━━━━━━━━━━━━ 10330s 4s/step - accuracy: 0.7149 - loss: 0.5994 - val_accuracy: 0.7145 - val_loss: 0.5961
Epoch 3/10
2332/2332 ━━━━━━━━━━━━━━━━━━━━ 0s 609ms/step - accuracy: 0.7152 - loss: 0.5980
Epoch 3: val_accuracy did not improve from 0.71451
2332/2332 ━━━━━━━━━━━━━━━━━━━━ 1624s 697ms/step - accuracy: 0.7152 - loss: 0.5980 - val_accuracy: 0.7145 - val_loss: 0.6010
Epoch 4/10
2332/2332 ━━━━━━━━━━━━━━━━━━━━ 0s 604ms/step - accuracy: 0.7115 - loss: 0.6009
Epoch 4: val_accuracy did not improve from 0.71451
2332/2332 ━━━━━━━━━━━━━━━━━━━━ 1607s 689ms/step - accuracy: 0.7115 - loss: 0.6009 - val_accuracy: 0.7145 - val_loss: 0.5964
500/500 ━━━━━━━━━━━━━━━━━━━━ 380s 761ms/step - accuracy: 0.7170 - loss: 0.5953

✅ Test Accuracy: 71.45%
```

Fig4. Training and validation

This approach is not only automated and scalable but also adaptable, making it a robust solution for preserving digital trust and maintaining the integrity of visual information. The use of a frame-based CNN allows for detailed pixel-level scrutiny, which is crucial for identifying the increasingly sophisticated artifacts present in modern deepfakes. The integration of data augmentation and a separate validation phase ensures that the model remains effective even when faced with new variations of manipulation techniques, thereby providing a reliable line of defense against synthetic media forgery.

VIII. RESULT & CONCLUSIONS



```
Epoch 2/10
1961/1961 ━━━━━━━━━━━━━━━━━━━━ 0s 899ms/step - accuracy: 0.6156 - loss: 0.6632
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is consider
1961/1961 ━━━━━━━━━━━━━━━━━━━━ 2120s 1s/step - accuracy: 0.6164 - loss: 0.6623 - val_accuracy: 0.6504 - val_loss: 0.6332 - learning_rate: 0
Epoch 3/10
1961/1961 ━━━━━━━━━━━━━━━━━━━━ 2115s 1s/step - accuracy: 0.6178 - loss: 0.6625 - val_accuracy: 0.5984 - val_loss: 0.6648 - learning_rate: 0
Epoch 4/10
1961/1961 ━━━━━━━━━━━━━━━━━━━━ 0s 900ms/step - accuracy: 0.6132 - loss: 0.6660
Epoch 4: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
1961/1961 ━━━━━━━━━━━━━━━━━━━━ 2689s 1s/step - accuracy: 0.6146 - loss: 0.6646 - val_accuracy: 0.6304 - val_loss: 0.6455 - learning_rate: 0
Epoch 5/10
1961/1961 ━━━━━━━━━━━━━━━━━━━━ 0s 2s/step - accuracy: 0.6240 - loss: 0.6540
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is consider
1961/1961 ━━━━━━━━━━━━━━━━━━━━ 3485s 2s/step - accuracy: 0.6254 - loss: 0.6533 - val_accuracy: 0.6599 - val_loss: 0.6260 - learning_rate: 5
Epoch 6/10
1961/1961 ━━━━━━━━━━━━━━━━━━━━ 2055s 1s/step - accuracy: 0.6281 - loss: 0.6506 - val_accuracy: 0.6328 - val_loss: 0.6424 - learning_rate: 5
Epoch 7/10
1961/1961 ━━━━━━━━━━━━━━━━━━━━ 0s 878ms/step - accuracy: 0.6277 - loss: 0.6516
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is consider
1961/1961 ━━━━━━━━━━━━━━━━━━━━ 2073s 1s/step - accuracy: 0.6272 - loss: 0.6513 - val_accuracy: 0.6617 - val_loss: 0.6207 - learning_rate: 5
Epoch 8/10
1961/1961 ━━━━━━━━━━━━━━━━━━━━ 2073s 1s/step - accuracy: 0.6252 - loss: 0.6515 - val_accuracy: 0.6568 - val_loss: 0.6274 - learning_rate: 5
Epoch 9/10
1961/1961 ━━━━━━━━━━━━━━━━━━━━ 0s 874ms/step - accuracy: 0.6270 - loss: 0.6504
Epoch 9: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
1961/1961 ━━━━━━━━━━━━━━━━━━━━ 2224s 1s/step - accuracy: 0.6281 - loss: 0.6500 - val_accuracy: 0.6501 - val_loss: 0.6328 - learning_rate: 5
Epoch 10/10
1961/1961 ━━━━━━━━━━━━━━━━━━━━ 3343s 2s/step - accuracy: 0.6357 - loss: 0.6450 - val_accuracy: 0.6473 - val_loss: 0.6343 - learning_rate: 2
421/421 ━━━━━━━━━━━━━━━━━━━━ 904s 2s/step - accuracy: 0.6518 - loss: 0.6255
📊 Test Accuracy: 65.18%
```

Fig5. Test Accuracy

The Vid-Verify system was tested on a varied dataset that included both real and altered videos. The model achieved a test accuracy of 65.18%, showing it can effectively tell apart real and fake content. The evaluation results indicated a precision of 0.64 and a recall of 0.66. This shows a balanced ability to identify altered content and reduce incorrect classifications. The F1-score of 0.65 confirms steady performance for both real and deepfake videos. The model showed it could generalize well by recognizing different deepfake types from new data, but it struggled with highly advanced manipulations. Performance dropped when handling low-quality or heavily compressed videos, as compression artifacts greatly impacted the detection of subtle manipulation signs.

In conclusion, this project successfully created a basic frame-based deepfake detection system using a Convolutional Neural Network, achieving 65.18% accuracy. The results support the potential of CNN-based methods for identifying spatial artifacts in deepfakes while emphasizing the need for significant improvements in detection ability. Future efforts will focus on improving the model's structure with deeper networks and better convolutional layers, adding temporal analysis to utilize sequential frame inconsistencies, and developing strong preprocessing techniques to enhance resilience against video compression. These improvements are vital for making the system ready for real-world use and for more reliable deepfake detection.

IX. REFERENCE

[1] R. Tolosana, R. Vera-Rodriguez, J. Fierrez, A. Morales, and J. Ortega-Garcia, "Deepfakes and beyond: A survey of face manipulation and fake detection," Information Fusion, vol. 64, pp. 131-148, 2020.

[2] D. Afchar, V. Nozick, J. Yamagishi, and I. Echizen, "MesoNet: a compact facial video forgery detection network," in 2018 IEEE International Workshop on Information Forensics and Security (WIFS), 2018.

[3] F. Mater, A. Rössler, and J. Fridrich, "Exposing deep fake videos by detecting eye blinking," in Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops, 2019.

[4] P. Korshunov and S. Marcel, "DeepFakes: a new threat to face recognition? assessment and detection," arXiv preprint arXiv:1812.08685, 2018.

[5] B. Bayar and M. C. Stamm, "A deep learning approach to universal image manipulation detection using a new convolutional layer," in Proceedings of the ACM Workshop on Information Hiding and Multimedia Security, 2016.

[6] X. Zhou and L. Dong, "Deepfake Detection using Convolutional Neural Networks," in Proceedings of the International Conference on Computer Vision (ICCV) Workshops, 2020.

[7] D. Guera and E. J. Delp, "Deepfake video detection using recurrent neural networks," in 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 2018.

[8] H. Dang, F. Liu, J. Stehouwer, X. Liu, and A. K. Jain, "On the detection of digital face manipulation," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[9] A. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, "FaceForensics++: Learning to Detect Manipulated Facial Images," in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019.

[10] FaceForensics Dataset. [Online]. Available: https://github.com/ondyari/FaceForensics