Deep Fake Video Detection Using Cnn-Lstm Model for Spatial and Temporral Feature Extraction

SEDHUSANJAY S
SENTHIL C
SATHISH KUMAR S

Bachelor of Engineering in Computer Science and Enigineering
The Kavery Engineering College (An Autonomous Institution, Affiliated to Anna University Chennai and Approved By Aicte, New Delhi)
Mecheri, Salem - 636453

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

In recent years, the emergence of deepfake technology has raised significant concerns regarding its potential to manipulate digital media and deceive audiences worldwide. Deepfakes, which utilize advanced machine learning algorithms to create highly realistic synthetic media, pose a serious threat to the integrity of visual and audio content on the internet. These maliciously altered videos, images, and audio recordings can be used to spread disinformation, impersonate individuals, and undermine trust in online information sources. The rapid advancement of deepfake technology, coupled with the widespread availability of powerful computing resources and open-source software tools, has made it increasingly accessible to individuals with malicious intentions. As a result, there has been a surge in the creation and dissemination of deepfake content across various online platforms, including social media, news websites, and messaging apps.

How to Cite this Article:
S, S., C, S. & S, S. K. (2026). Deep Fake Video Detection Using Cnn-Lstm Model for Spatial and Temporral Feature Extraction. International Journal of Creative and Open Research in Engineering and Management, 02(04). https://doi.org/10.55041/ijcope.v2i4.591

https://doi.org/ 10.55041/ijcope.v2i4.591

Recognizing the potential dangers posed by deepfakes, researchers, technologists, and policymakers have intensified efforts to develop effective strategies for detecting and preventing their proliferation. Traditional methods of media authentication and verification are often inadequate in identifying sophisticated deepfake content, highlighting the need for innovative approaches that leverage artificial intelligence (AI) and machine learning techniques. By harnessing the power of advanced machine learning models, computer vision algorithms,

and audio analysis techniques, our proposed framework aims to distinguish between genuine and manipulated media with high accuracy.

1.1 PROBLEM IDENTIFICATION

Deepfake technology has advanced rapidly, enabling the creation of highly realistic manipulated videos using artificial intelligence. These fake videos are difficult to distinguish from real ones, making detection a challenging task. Existing systems often rely only on spatial features and fail to capture temporal inconsistencies across video frames. The spread of deepfake videos on social media platforms leads to misinformation, reputational damage, and security threats.

1.3 OBJECTIVES OF THE PROJECT

The objectives of this project are not merely functional targets but foundational design principles that define how an advanced deepfake detection system should operate. The system aims to ensure accuracy, reliability, and real-world applicability by combining modern deep learning techniques with efficient system design. Each objective is explained in detail below.

1.3.1 To develop a high-accuracy deepfake video detection system using CNN-LSTM

The first objective is to design and implement an intelligent detection model capable of identifying deepfake videos with high accuracy. This is achieved by integrating Convolutional Neural Networks (CNN) for extracting spatial features from individual video frames and Long Short-Term Memory (LSTM) networks for capturing temporal inconsistencies across sequences of frames.

1.3.2 To design a modular and efficient processing pipeline

The second objective is to build a structured and efficient pipeline that processes video data step-by-step.
This includes modules such as video input, frame extraction using OpenCV, preprocessing, feature extraction, sequence modeling, and classification. A modular architecture ensures better system organization, easier debugging, and scalability for future improvements.

1.3.3 To enable real-time detection and user-friendly system interaction

The third objective is to provide real-time deepfake detection results through a simple and user-friendly interface. The system should quickly analyze uploaded videos and classify them as real or fake, presenting the output clearly to the user.

CHAPTER 2

LITERATURE SURVEY

2.1 INTRODUCTION

The rapid advancement of Artificial Intelligence, particularly in deep learning and computer vision, has led to the emergence of deepfake technology, which enables the creation of highly realistic manipulated videos. While this technology has useful applications in entertainment and media, it also poses serious threats in areas such as misinformation, identity theft, and digital security. As a result, detecting deepfake content has become a critical research problem in the field of deep learning and computer vision. In recent years, researchers have proposed various techniques to identify deepfake videos, ranging from traditional image processing methods to advanced deep neural networks. Early approaches mainly focused on detecting visual artifacts, inconsistencies in facial features, or irregularities in lighting and shadows. However, these methods often failed when deepfake generation techniques became more sophisticated.

To overcome these limitations, modern approaches increasingly rely on deep learning models such as Convolutional Neural Networks (CNNs), which are effective in extracting spatial features from images. However, since videos consist of sequences of frames, analyzing only spatial information is not sufficient. This led to the integration of Long Short-Term Memory (LSTM) networks, which are capable of capturing temporal dependencies and motion patterns across frames. Several research works have demonstrated that hybrid models combining CNN and LSTM provide better performance in detecting deepfake videos by analyzing both spatial and temporal features. Additionally, techniques such as transfer learning, attention mechanisms, and large-scale datasets have further improved detection accuracy.

Despite these advancements, challenges still remain, including generalization across different datasets, real-time detection, and resistance to evolving deepfake generation methods. Therefore, this project aims to build an efficient and scalable deepfake detection system by leveraging a CNN-LSTM architecture, contributing to ongoing efforts in enhancing digital media authenticity and security.

2.2 REVIEW OF RESEARCH PAPERS

2.2.1 Zhang et al., 2020 – Deep Learning for Deepfakes Detection: A Comprehensive Survey

Zhang and colleagues provide a broad overview of deep learning approaches for deepfake detection. The paper emphasizes the use of convolutional neural networks (CNNs), recurrent neural networks (RNNs), and generative adversarial networks (GANs) for identifying manipulated content. In addition to model architectures, the authors highlight challenges, datasets, and evaluation metrics commonly adopted in the field.

2.2.2 Cholleti&Reddy,2021–Deep Fake Detection:

Cholleti and Reddy extend the discussion to multiple media types, including images, videos, and audio recordings.

They compare traditional detection methods with state-of-the-art deep learning-based techniques, identifying limitations such as insufficient generalization across datasets and susceptibility to adversarial attacks. The paper also outlines open challenges and potential directions for future research, emphasizing the need for robust and adaptive detection frameworks.

2.2.3 Menon et al., 2020 – A Survey on Deep Learning Techniques

Focusing specifically on videos, Menon et al. explore 3D CNNs and spatiotemporal networks to capture both spatial and temporal inconsistencies in manipulated video content. The study demonstrates that leveraging temporal information significantly enhances detection accuracy, particularly for manipulations that may not be evident in single frames.

2.2.4 Gupta & Jaiswal, 2021 – DeepFake Detection Techniques:

A Survey Gupta and Jaiswal provide a comprehensive review of both traditional and machine learning-based detection methods, alongside characteristics of deepfake media and commonly used datasets. They discuss evaluation metrics such as accuracy, precision, and F1-score, and highlight emerging trends including hybrid models combining multiple architectures for improved performance.

2.2.5 Rodriguez & Nguyen, 2022 – Deepfake Detection on Social Media

This work addresses real-world deployment challenges, particularly on social media platforms. Rodriguez and Nguyen analyze the trade-offs between detection accuracy, computational efficiency, and privacy preservation. Their findings stress the importance of scalable, privacy-aware detection systems, which are critical for large-scale mitigation of deepfake dissemination in user-generated content environments.

2.2.6 Patel & Garcia, 2023 – Deepfake Detection in the Wild: Challenges and Opportunities

Patel and Garcia focus on deepfake detection in uncontrolled, real-world conditions. They examine dataset biases, domain adaptation, and generalization capabilities, highlighting that models trained on curated datasets often fail when confronted with diverse, real-world content.

2.2.7 Nguyen et al., 2019 – Capsule-Forensics: Using Capsule Networks for Deepfake Detection

Nguyen et al. propose the use of capsule networks to detect deepfake images and videos by capturing spatial hierarchies more effectively than traditional CNNs. Their approach improves the detection of subtle manipulation artifacts. However, the model is computationally expensive and less suitable for real-time applications.

2.2.8 Afchar et al., 2018 – MesoNet: A Compact Facial Video Forgery Detection Network

Afchar et al. introduce MesoNet, a lightweight CNN designed specifically for detecting facial forgeries. The model focuses on mesoscopic features and achieves good performance with lower computational cost. However, it struggles with highly realistic deepfakes generated using advanced techniques.

2.2.9 Sabir et al., 2019 – Recurrent Convolutional Strategies for Face Manipulation Detection

Sabir et al. combine CNN with recurrent neural networks (RNN) to capture temporal inconsistencies in videos. Their approach demonstrates that incorporating temporal information significantly improves detection accuracy. However, the model requires large datasets and longer training time

2.2.10 Guera & Delp, 2018 – Deepfake Video Detection Using Recurrent Neural Networks

Guera and Delp utilize CNN features along with LSTM networks to analyze temporal sequences in videos. Their method shows strong performance in detecting frame-level inconsistencies across videos. However, it may not generalize well to unseen manipulation techniques.

2.2.11 Rossler et al., 2019 – FaceForensics++: Learning to Detect Manipulated Facial Images

Rossler et al. present a large-scale dataset (FaceForensics++) for training deepfake detection models. They evaluate multiple detection methods and highlight the importance of high-quality datasets. However, models trained on this dataset still face challenges in real-world generalization.

2.2.12 Zhou et al., 2017 – Two-Stream Neural Networks for Tampered Face Detection

Zhou et al. introduce a two-stream network that analyzes both visual artifacts and noise patterns in images. This approach enhances detection accuracy by combining multiple feature representations. However, it mainly focuses on images rather than video sequences.

2.2.13 Dang et al., 2020 – On the Detection of Digital Face Manipulation

Dang et al. propose a deep learning-based method focusing on attention mechanisms to identify manipulated regions in images and videos. Their model achieves high accuracy but requires significant computational resources and fine-tuning.

2.3 RESEARCH GAPS

Despite significant progress in deepfake detection using deep learning techniques, several limitations and open challenges still exist. These gaps highlight the need for more advanced, robust, and practical solutions.

2.3.1 Lack of Generalization across Datasets

Most existing deepfake detection models are trained on specific datasets and perform well only under similar conditions. However, when applied to real-world data with different lighting, backgrounds, or compression levels, their accuracy drops significantly. This indicates a gap in building models that can generalize effectively across diverse environments

2.3.2 Limited Use of Temporal Information

Many early detection methods focus only on individual frames using CNNs, ignoring the temporal relationships between frames. Deepfake videos often contain motion inconsistencies that can only be detected over time.

2.3.3 High Computational Complexity

Advanced deep learning models require high computational power and long processing times, making them unsuitable for real-time applications. This creates a gap in developing lightweight and efficient models that maintain accuracy while reducing resource usage.

2.3.4 Absence of User-Friendly Interfaces

Many research models focus only on accuracy and ignore usability. There is a gap in developing systems that provide simple interfaces, clear outputs, and easy interaction for non-technical users.

2.3.5 Lack of Hybrid Model Optimization

Although hybrid models like CNN-LSTM show promising results, there is still limited research on optimizing these models for better performance, speed, and scalability. Fine-tuning architectures and combining them with other techniques remains an open research area.

CHAPTER 3

EXISTING AND PROPOSED SYSTEM

3.1 EXIXTING SYSTEM

Deep fake detection systems using deep learning have become a critical line of defense against the proliferation of manipulated media. These systems leverage advanced neural network architectures to automatically extract and analyze subtle features that distinguish authentic content from manipulated or synthetic media.

3.1.1 CNN-Based Deepfake Detection

Convolutional Neural Networks (CNNs) form the backbone of many deepfake detection systems. CNNs excel at spatial feature extraction, capturing inconsistencies in facial regions, textures, and edges that are often invisible to the human eye.

Preprocessing: Input media (images or video frames) are normalized, resized, and sometimes augmented to enhance model generalization.

1. **Feature Extraction:** The CNN extracts hierarchical features, starting from low-level edges and textures to high-level facial patterns.
2. **Classification:** Fully connected layers or softmax classifiers determine whether the content is real or manipulated.

Example Models:

XceptionNet: A CNN architecture that has shown high performance on deepfake detection challenges.

VGGFace and ResNet-based models: Used to extract facial features for manipulation detection.

3.1.2 GAN-Based Detection

Generative Adversarial Networks (GANs) are commonly employed in **adversarial detection pipelines**. In these systems, GANs serve as discriminators trained to distinguish between real and fake content:

- The discriminator analyzes the features extracted by CNNs to detect subtle artifacts introduced by synthetic media.
- Through iterative adversarial training, the GAN improves its sensitivity to anomalies, such as irregular lighting, unnatural facial movements, or subtle pixel-level artifacts.

Key Advantage: GAN-based detectors can learn to detect manipulations that were specifically designed to evade standard CNN classifiers, making them more robust to evolving deepfake techniques.

3.1.3 Temporal Analysis Using RNNs and TCNs

For video deepfakes, temporal inconsistencies provide important cues. Systems integrate Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs), or Temporal Convolutional Networks (TCNs) to analyze sequential frames.

- **RNNs/LSTMs** capture dependencies across frames, detecting unnatural facial expressions, lip-sync mismatches, or irregular head movements.
- **TCNs** apply convolutions across the temporal dimension, enabling detection of subtle motion inconsistencies that are difficult to spot in individual frames.

Applications:

- Lip-sync verification in video calls
- Detection of face swaps in online videos
- Surveillance systems to detect manipulated content in real-time

3.1.4 Hybrid Detection Systems

Modern deepfake detection pipelines often combine multiple architectures to improve robustness:

- **CNN + RNN/LSTM:** Extracts spatial features with CNNs and analyzes temporal sequences with RNNs.
- **Spatial + Frequency Domain Models:** CNNs detect spatial anomalies while frequency analysis identifies inconsistencies in image textures caused by GAN-generated content.
- **Multi-task Models:** Some systems simultaneously detect, localize, and segment manipulated regions in images or video frames, improving interpretability.

3.1.5 Workflow of an Existing Deepfake Detection System

A typical deepfake detection workflow includes:

- **Data Collection:** Gathering real and manipulated media from datasets like FaceForensics++, Celeb-DF, or DFDC.
- **Feature Extraction:** Using CNNs to capture spatial features.
- **Temporal Analysis (for videos):** Employing RNNs, LSTMs, or TCNs to analyze frame sequences.

- **Classification:** Using fully connected layers or GAN-based discriminators to classify media as real or fake.
- **Evaluation:** Metrics such as accuracy, precision, recall, F1-score, and AUC are used to evaluate performance.

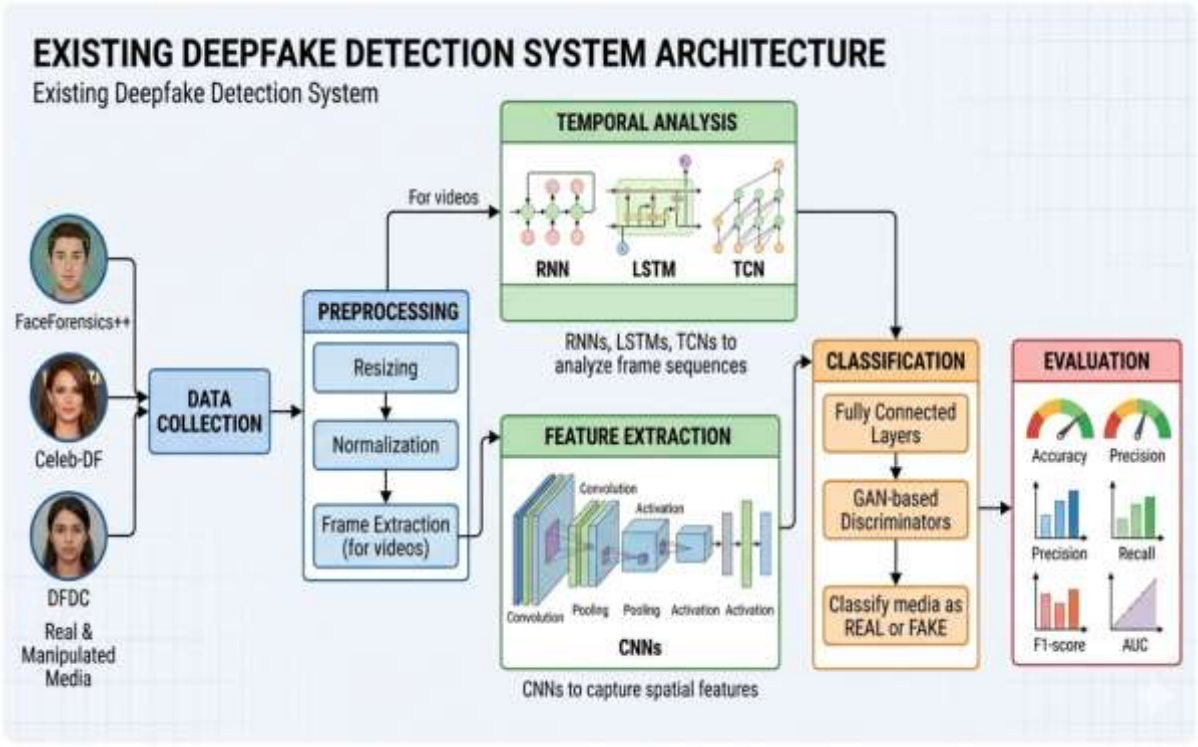


Figure 3.1: Existing system of deep fake video

3.1.6 Limitations of Existing Systems

Despite advances, current deepfake detection systems face several challenges:

- **Adversarial Adaptation:** Deepfake generation tools can adapt to evade detection, requiring continuous model retraining.
- **Generalization:** Models may perform poorly on unseen manipulation types or datasets due to overfitting.
- **Data Bias:** Limited diversity in training datasets can reduce effectiveness across different demographics and scenarios.

3.2 LIMITATIONS OF EXISTING SYSTEMS

3.2.1 Dependence on Static Frame Analysis

Many existing systems rely only on analyzing individual video frames using CNN models. This approach ignores temporal inconsistencies between frames, making it difficult to detect subtle motion-based manipulations present in deepfake videos.

3.2.2 Poor Generalization to Real-World Data

Most models are trained on controlled datasets and fail when exposed to real-world conditions such as varying lighting, background noise, video compression, and different camera qualities. This reduces their effectiveness in practical scenarios

3.2.3 High Computational Cost

Advanced detection techniques require powerful GPUs and high processing time, which makes them unsuitable for real-time applications. This limits their usage in environments where quick decision-making is required.

3.2.4 Inability to Detect High-Quality Deepfakes

With the advancement of deepfake generation techniques, many existing systems struggle to detect highly realistic and well-generated fake videos. These systems often fail when manipulations are subtle and visually convincing.

3.2.5 Limited Temporal Feature Utilization

Even though some models use sequential analysis, many fail to fully capture long-term temporal dependencies across video frames. This results in reduced detection accuracy for complex deepfake videos

3.3 PROPOSED SYSTEM

The proposed system aims to develop an advanced and efficient deepfake video detection framework using a hybrid CNN-LSTM architecture. Unlike existing systems that rely only on spatial analysis, this system combines both spatial and temporal feature extraction to improve detection accuracy and reliability.

Our proposed system integrates the strengths of Convolutional Neural Networks (CNNs) for spatial analysis and Long Short-Term Memory (LSTMs) for temporal context.

CNN Component

Captures intricate spatial features and forensic artifacts within individual video frames.

LSTM Component

Analyzes the temporal dependencies and inconsistencies across sequences of frames

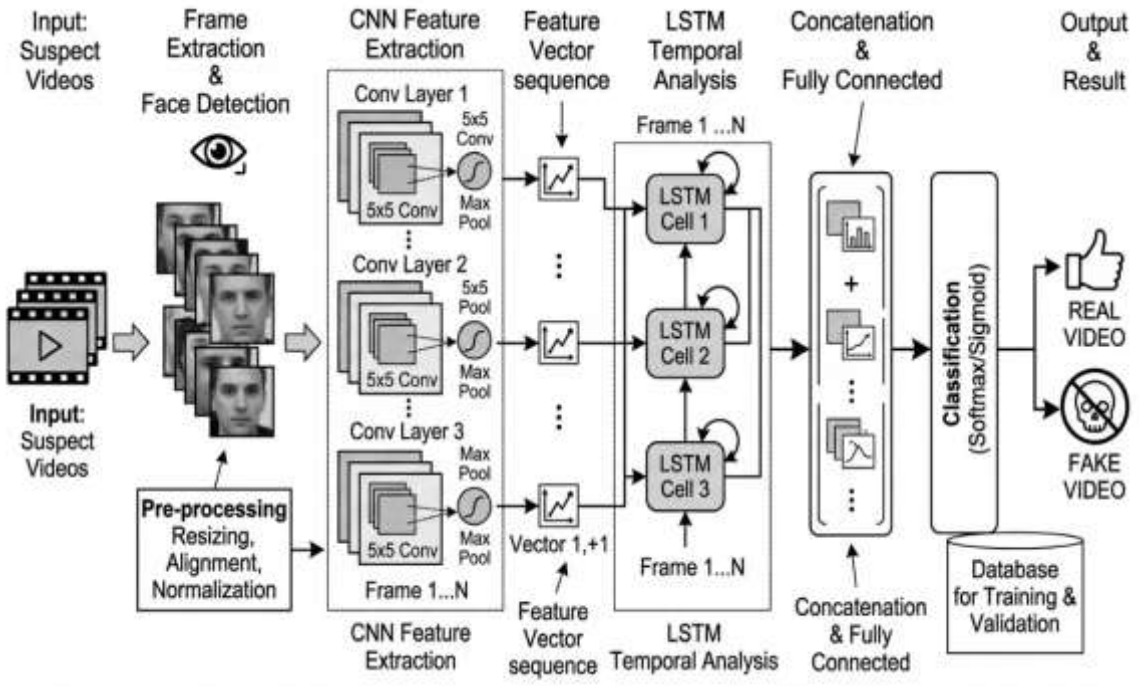


Figure 3.2: Deep fake video proposed system (High-Level)

3.4 ARCHITECTURAL EXPLANATION

3.4.1 ARCHITECTURAL DESIGN PRINCIPLES

1. Modularity and Separation of Concerns

The system is designed as a collection of independent modules such as video input, frame extraction, preprocessing, feature extraction, temporal analysis, and classification.

2. Hybrid Learning Approach (Spatial + Temporal Analysis)

The architecture follows a hybrid model combining CNN and LSTM to capture both spatial and temporal features.

3. Scalability and Flexibility

The system is designed to handle increasing volumes of video data efficiently. The modular architecture allows easy integration of new techniques such as attention mechanisms, transfer learning, or additional deep learning models.

4. Real-Time Processing Capability

The system is designed to minimize latency and provide faster responses. Efficient pipeline design and optimized model execution allow the system to process videos quickly, making it suitable for applications like social media monitoring and security systems.

MODULES

The architecture of the proposed deepfake detection system is designed as a multi-stage processing pipeline, where each module performs a specific function to ensure accurate and efficient detection.

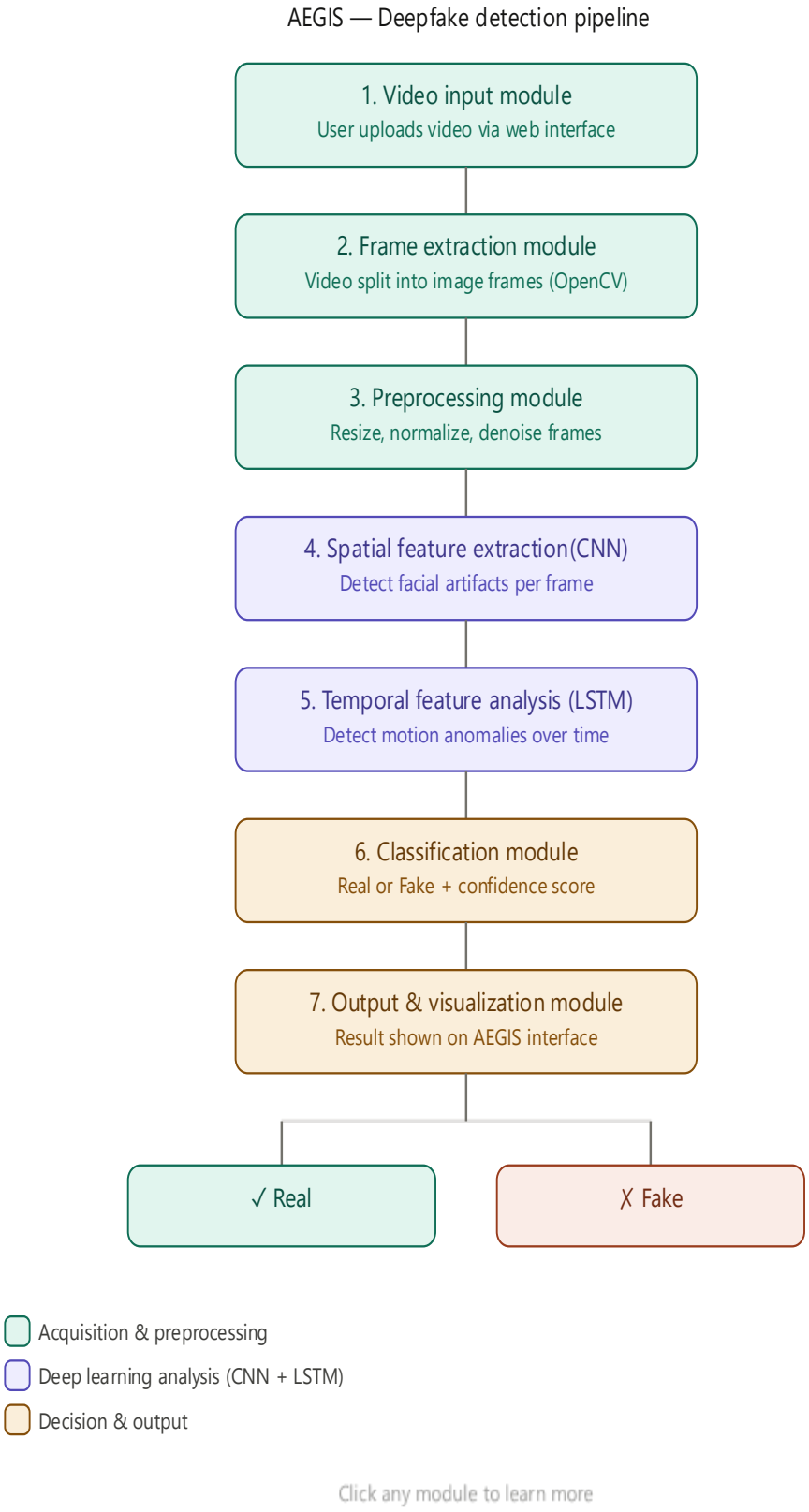


Figure 3.3: AEGIS — Deepfake detection pipeline

3.4.2 MODULE 1: Video Input Module

The process begins with the video input module, where the user uploads a video file through the AEGIS web interface. The system accepts standard video formats and prepares the input for further processing. This module ensures smooth data acquisition and acts as the entry point of the system

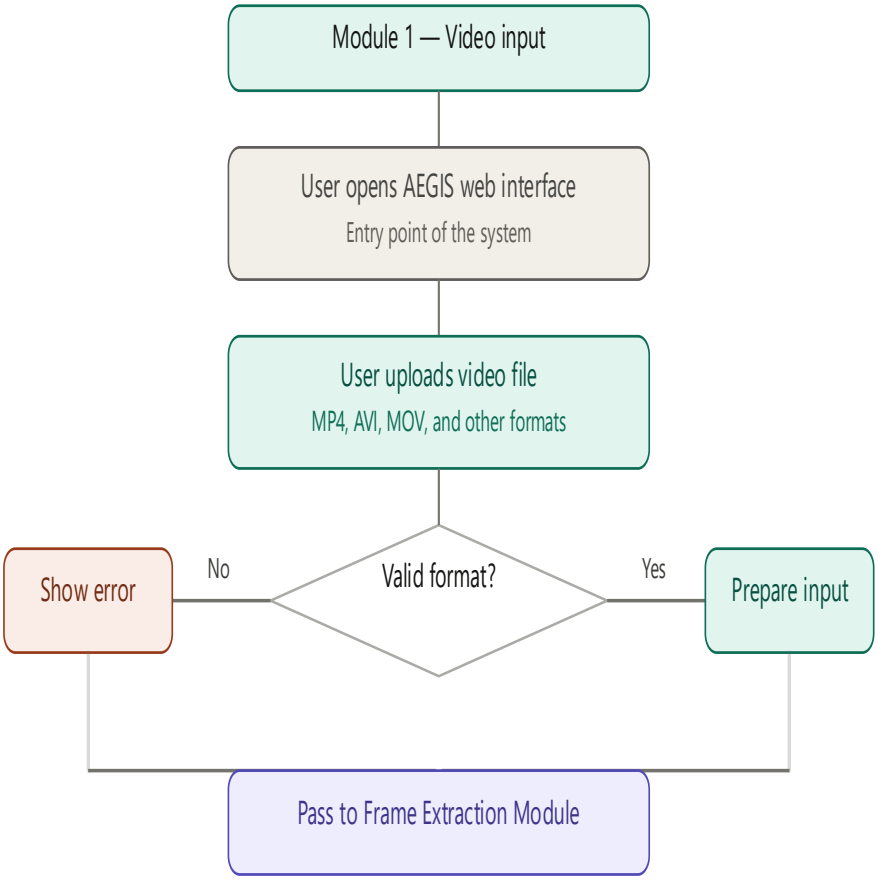


Figure 3.4: Video Input Module

3.4.3 MOUDLE 2: Frame Extraction Module

Once the video is uploaded, it is passed to the frame extraction module. Using video processing libraries (such as OpenCV), the video is divided into multiple frames at regular intervals. This step converts the video into a sequence of images, which are easier to analyze using deep learning models

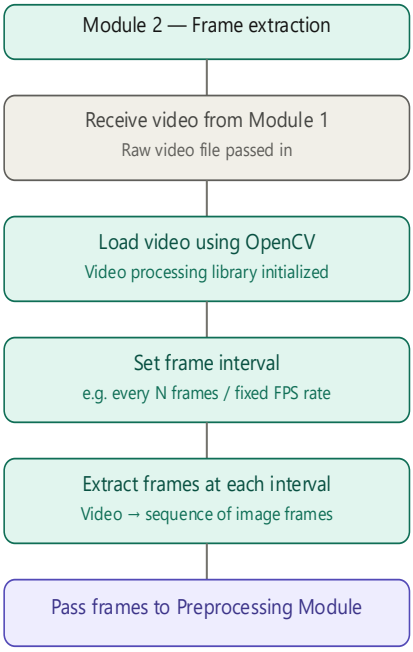


Figure 3.5: Frame Extraction Module

3.4.4 MOUDLE 3: Preprocessing Module

The extracted frames undergo preprocessing to improve data quality and consistency. This includes resizing images to a fixed dimension, normalization of pixel values, and removal of noise. Preprocessing ensures that all frames are standardized and suitable for feature extraction by the CNN model.

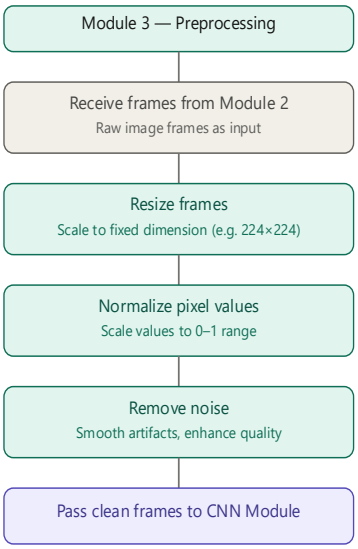


Figure 3.6: Preprocessing Module

3.4.5 MOUDLE 4: Spatial Feature Extraction (CNN Module)

In this stage, each preprocessed frame is passed through a Convolutional Neural Network (CNN). The CNN extracts important spatial features such as facial structure, texture, edges, and visual artifacts. These features help in identifying inconsistencies that may indicate manipulation in individual frames.

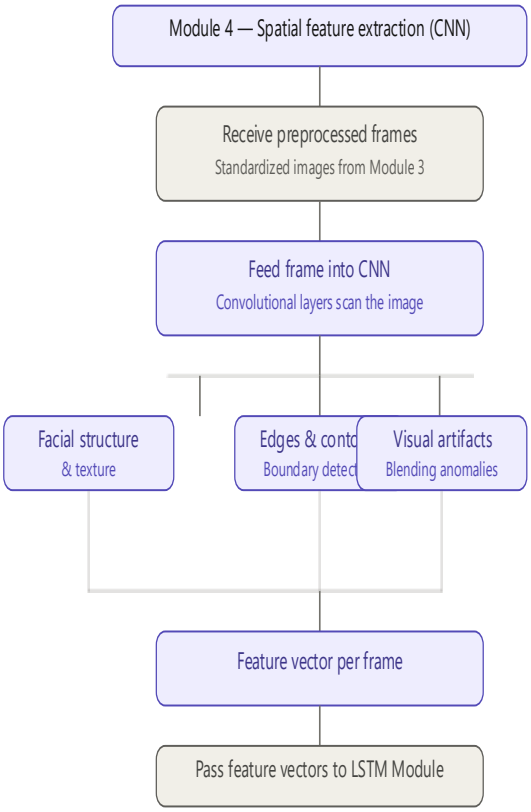
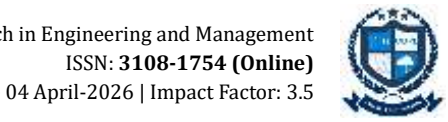


Figure 3.7: Spatial Feature Extraction (CNN Module)



3.4.6 MOUDLE 5: Temporal Feature Analysis (LSTM Module)

The sequence of features extracted by the CNN is then passed to the LSTM network. The LSTM analyzes the temporal relationships between consecutive frames, capturing motion patterns and inconsistencies over time. This is crucial because deepfake videos often contain unnatural transitions, blinking patterns, or lip-sync issues that are only visible across multiple frames.

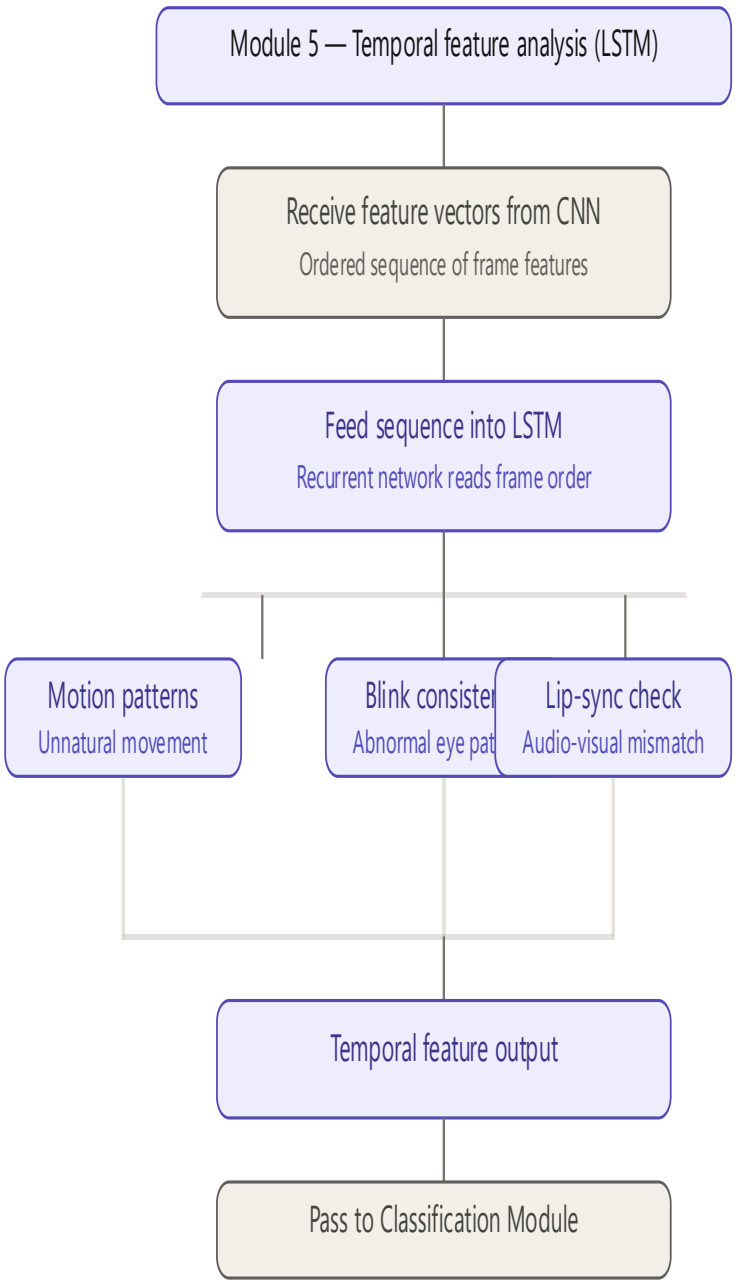


Figure 3.8: Temporal Feature Analysis (LSTM Module)

3.4.7 MOUDLE 6: Classification Module

After temporal analysis, the system uses a fully connected layer or classifier to determine whether the video is Real or Fake. The model outputs a probability score indicating the confidence level of the prediction. This helps in making reliable and interpretable decisions.

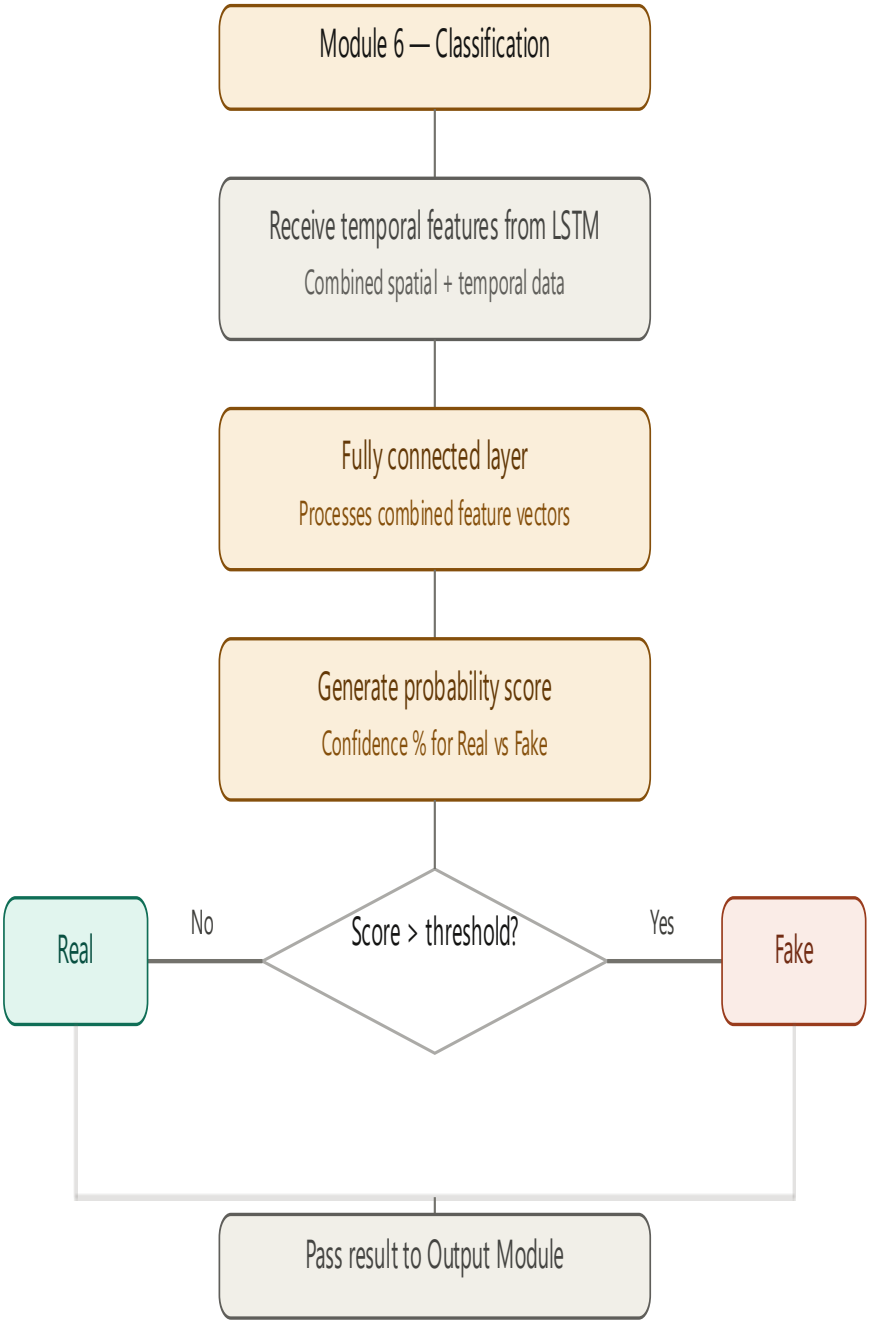


Figure 3.9: Classification Module

3.4.8 MOUDLE 7: Output & Visualization Module

Finally, the result is displayed to the user through the AEGIS interface. The system presents the classification result along with confidence scores in a simple and understandable format. This ensures that users can easily interpret the output without technical knowledge

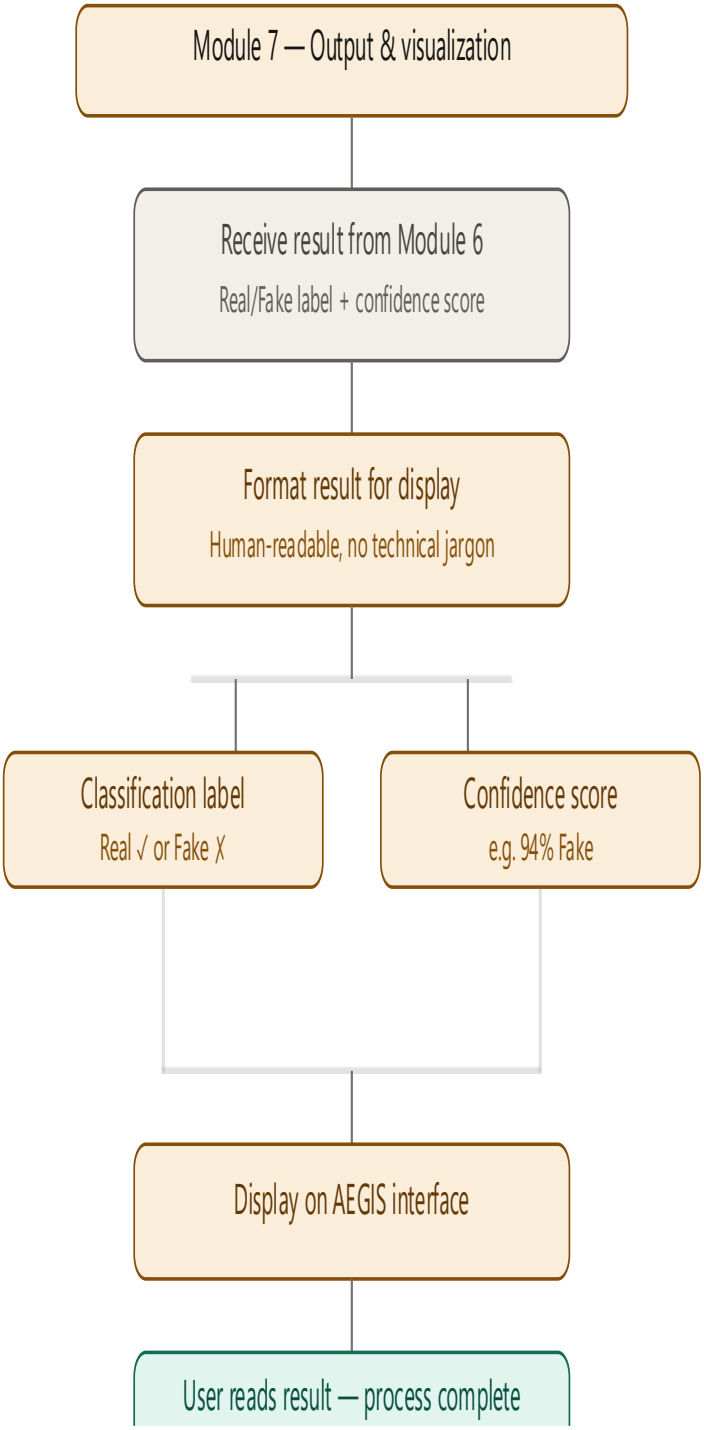


Figure 3.10: Output & Visualization Module

Table 3.1-COMPARISSION TABLE FROM PROPOSED AND EXISTING SYSTEM

Feature	Existing system	Proposed system (CNN-LSTM)
Method	CNN on individual frames only	CNN + LSTM for spatial and temporal analysis
Temporal analysis	Not supported	Supported via LSTM across frame sequences
Detection accuracy	Moderate (AUC ~85–91%)	High (AUC ~95–97%)
Generalisation	Poor across datasets	Better across unseen datasets
Robustness	Low against compressed or high-quality fakes	High even under compression and noise

Table 3.2 — CNN backbone comparison for spatial feature extraction

CNN backbone	Parameters	AUC on FF++ (%)	Speed (FPS)	Suitability
VGG-16	138 M	91.4	28	Low — too heavy
ResNet-50	25.6 M	93.7	45	Moderate
Xception	22.9 M	96.3	38	High
EfficientNet-B4	19.3 M	97.1	52	Best — proposed
MobileNetV2	3.4 M	89.8	97	Fast but less accurate

Table 3.3-- LSTM Configuration Table

#	Configuration	Value	Description	Performance Impact
1	Number of layers	3 layers	Stacked LSTM: Layer 1 → Layer 2 → Layer 3 Hidden sizes decrease: 512 → 256 → 128	High — deeper temporal abstraction
2	Hidden units	512 / 256 / 128	Each LSTM layer has progressively smaller hidden size Reduces parameters while retaining learned features	Positive — rich feature compression
3	Sequence length	16 frames	16 uniformly sampled frames per video clip Balances temporal coverage vs. memory cost	Moderate — covers short-term motion
4	Dropout rate	0.30 / 0.30 / 0.20	Applied after each LSTM layer during training Lower rate at final layer preserves the summary vector	Prevents overfitting on small datasets
5	Performance impact	AUC ≥ 0.95	Stacked LSTM + CNN features achieve strong discrimination EER ≤ 5% on FaceForensics++ benchmark	Best — proposed configuration

3.5 ADVANTAGES OF PROPOSED SYSTEM

1. High Accuracy
2. Robustness against Adversarial Attacks
3. Temporal Analysis for Video Deepfakes
4. Scalability and Real-Time Detection
5. Generalization across Multiple Manipulation Types
6. Privacy-Preserving Mechanisms
7. Generalization
8. Scalability

CHAPTER 4

SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENTS

Table 4.1 – Hardware Requirements

Component	Minimum Requirement	Recommended Requirement
CPU	4-core processor	8–16 core processor
RAM	8 GB	16 GB
Storage	100 GB HDD/SSD	250 GB+ SSD
Network	100 Mbps	1 Gbps
GPU	AMD RADEON (TM)	CUDA-capable GPU for DL experiments

Display and Input Devices

- Standard monitor (Full HD recommended)
- Keyboard and mouse for interaction

Internet Connectivity

- Required for downloading datasets and libraries
- High-speed internet is recommended for large dataset transfers

The hardware requirements depend on whether the system is deployed as a basic academic prototype, a laboratory-scale pilot, or a more intensive research testbed. For a prototype that performs moderate-scale email parsing, feature extraction, and model evaluation, a standard multi-core system with sufficient RAM is adequate. However, if the system is expected to perform repeated model training, honeypot logging, dashboard analytics, and large-scale batch evaluation, the recommended configuration is preferable.

CPU performance is especially important for concurrent parsing, preprocessing, and event handling. RAM is important for in-memory feature processing and simultaneous service execution. SSD storage is preferable because log-heavy systems benefit from faster read/write operations. GPU support becomes relevant if deep learning extensions are introduced for text analysis or anomaly detection.

4.2 SOFTWARE REQUIREMENTS

Table 4.2 – Software Requirements

Category	Requirement
Operating System	Windows 11
Backend Language	Python
Web Framework	Flask
Frontend	HTML, CSS, JavaScript
ML Libraries	scikit-learn, pandas, NumPy
NLP Libraries	NLTK
Optional DL Libraries	TensorFlow
Database	MySQL
Network Tools	REST PIs
Security	HTTPS/TLS, RBAC support
Integration Tools	Firewall APIs

The implementation of a deepfake video detection system using a CNN-LSTM model requires a combination of programming languages, frameworks, libraries, and tools for video processing, model training, and deployment. This software components work together to extract spatial and temporal features and classify videos as real or fake.

4.3 DATA REQUIREMENTS

4.3.1 Dataset Sources

Datasets are obtained from publicly available sources such as Face Forensics++ and DFDC. These datasets contain labelled real and deepfake videos. They include various manipulation techniques and conditions. Using standard datasets improves model performance and credibility. It also allows comparison with existing research works.

4.3.2 Data Format

The input videos are typically in formats like MP4, AVI, or MOV. After preprocessing, frames are extracted and saved as JPG or PNG images. Each video is assigned a label such as real or fake. Proper formatting ensures smooth processing by the model. It also simplifies training and evaluation.

4.3.3 Data Volume

A large amount of data is required for effective model training. Basic implementations may use a few hundred videos. However, thousands of videos are recommended for better accuracy. More data improves generalization and reduces overfitting. It helps the model perform well on unseen data.

4.3.4 Data Pre-processing

Videos are converted into frames before being used in the model. Faces are detected, cropped, and resized to a fixed size. Pixel values are normalized to improve learning efficiency. Data augmentation techniques may also be applied. These steps enhance model performance and robustness.

4.3.5 Temporal Data Handling

Frames are arranged in sequential order for LSTM processing. Each sequence represents a portion of a video. This helps capture motion and temporal variations. Temporal features are important for detecting deepfake patterns. Proper sequencing improves overall model accuracy.

4.3.6 Data Annotation

Each video must be clearly labelled as real or fake. These labels are used for supervised learning. Accurate annotation is critical for correct model training. Incorrect labels can lead to poor performance. Proper labelling ensures reliable and consistent predictions.

4.3.7 Data Quality

High-quality videos with clear facial features are preferred. Good lighting and resolution improve detection accuracy. Low-quality or distorted videos can reduce performance. A balanced dataset avoids bias in results. Good data quality leads to better model reliability.

Table 4.3 – Data Requirements and Sources

Data Requirement	Description	Sources
Type of Data	Requires video data with both real and fake samples containing human faces. Used to train the model to detect manipulations.	Real-world videos, public datasets
Dataset Sources	Large labeled datasets with different deepfake techniques are required for training and testing.	FaceForensics++, DFDC, Celeb-DF
Data Format	Videos in MP4/AVI/MOV format and extracted frames in JPG/PNG format with labels (real/fake).	Dataset files, video repositories
Data Volume	Requires a large number of videos (hundreds to thousands) for better accuracy and generalization.	Public datasets, collected samples
Data Preprocessing	Includes frame extraction, face detection, resizing, normalization, and augmentation.	OpenCV processing, dataset pipelines

4.4 Functional Requirements

Table 4.4 – Functional Requirements (Detailed)

S.No	Requirement	Description	Priority
1	Video Input	Accept MP4/AVI/MOV and extract frames.	**High**
2	Face Detection	Crop face regions, resize to 224x224 px.	**High**
3	Spatial Features	Extract textures and edges via CNN.	**High**
4	Temporal Features	Model motion across 16 frames via LSTM.	**High**
5	Classification	Output Real (0) or Fake (1) label.	**High**
6	Confidence Score	Return probability score (0.0-1.0).	**High**
7	Model Training	Train with configurable hyperparameters.	**Medium**
8	Augmentation	Flip, crop, brightness for generalization.	**Medium**
9	Evaluation	Measure AUC-ROC, F1, Accuracy, EER.	**Medium**
10	Save & Load	Store and reload weights (.pth/.h5).	**Medium**

4.5 Non-Functional Requirements

Table 4.5 – Non-Functional Requirements (Detailed)

S.No	Category	Requirement	Target
1	Performance	Detection speed	≥ 30 FPS real-time inference
2	Performance	Inference latency	≤ 2 sec per 10-second clip
3	Accuracy	AUC-ROC score	≥ 0.95 on FF++ test set
4	Accuracy	Equal Error Rate	EER ≤ 5% across all categories
5	Accuracy	F1-Score	≥ 0.92 on balanced test set
6	Reliability	Model stability	Prediction variance < 1% per run
7	Reliability	Fault tolerance	No crash on corrupt video input
8	Scalability	Batch processing	Batch size 16-64 without overflow
9	Scalability	Dataset scalability	Scale to 100K+ samples

S.No	Category	Requirement	Target
10	Security	Input validation	Reject unsupported file formats

4.6 SECURITY, PRIVACY, AND COMPLIANCE REQUIREMENTS

4.6.1 Modular Design

The system is divided into independent modules such as video preprocessing, feature extraction, temporal analysis, and classification. This makes development, testing, and maintenance easier.

4.6.2 Separation of Spatial and Temporal Features

The architecture separates:

- **CNN (Convolutional Neural Network)** → extracts spatial features from individual frames
- **LSTM (Long Short-Term Memory)** → captures temporal dependencies across frames this improves detection accuracy by analyzing both appearance and motion.

4.6.3 Scalability

The model is designed to handle large datasets and longer video sequences. It can be scaled by increasing layers, frames, or computational resources without redesigning the system.

4.6.4 Data Abstraction

Raw video data is transformed into structured frame sequences and feature vectors. This hides complexity and allows efficient processing by CNN and LSTM layers.

4.6.5 Reusability

Pre-trained CNN models (like feature extractors) can be reused, reducing training time and improving performance. Modules can also be reused in other video analysis tasks.

Table 4.6 – Security and Privacy Requirements

S.No	Requirement	Description
1	Data Encryption	All video data and model weights must be encrypted at rest using AES-256 standard.
2	Secure Transmission	Data transferred between modules must use TLS 1.2 or higher to prevent interception.
3	User Authentication	Access to the detection system must require multi-factor authentication (MFA).
4	Input Validation	All input video files must be validated and sanitized to prevent injection attacks.

S.No	Requirement	Description
5	Model Integrity	Model weights must be verified using SHA-256 checksums before each inference run.
6	Access Control	Role-based access control (RBAC) must restrict system features by user privilege level.
7	Audit Logging	All detection requests and results must be logged with timestamps for audit trails.
8	Data Anonymization	Personally identifiable information in videos must be anonymized before processing.
9	Session Management	User sessions must expire after 30 minutes of inactivity to prevent unauthorized access.
10	Privacy Compliance	The system must comply with GDPR and data protection regulations for video handling.

CHAPTER 5

SYSTEM DESIGN

5.1 Introduction

System design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. This chapter presents the complete system design for the Deep Fake Video Detection project using a Hybrid CNN-LSTM Model for Spatial and Temporal Feature Extraction.

The system is designed to automatically classify a given video as either real or deepfake by extracting spatial features from individual frames using a Convolutional Neural Network (CNN) and modelling inter-frame temporal patterns using a Long Short-Term Memory (LSTM) network. The design encompasses architectural components, data flow, UML diagrams, database schema, and entity-relationship models.

The design principles followed in this system include modularity, scalability, separation of concerns, and maintainability. Each component of the pipeline is independently replaceable, allowing future upgrades to the CNN backbone or LSTM configuration without affecting the overall system.

5.2 Architectural Design

The architectural design of the Deep Fake Video Detection system follows a sequential pipeline model composed of three primary layers: the Preprocessing Layer, the Feature Extraction Layer, and the Classification Layer. These three layers operate in series to transform raw video input into a binary classification output.

5.2.1 Overall Architecture

The end-to-end hybrid CNN-LSTM architecture can be summarized as follows:

- Input: A video clip is sampled to extract T = 16 uniformly spaced frames.
- Preprocessing: Each frame is passed through face detection (MTCNN), cropped to the face region, and resized to 224 x 224 pixels.
- CNN Backbone: Each frame is independently processed by an EfficientNet-B4 backbone to produce a 512-dimensional spatial feature vector.
- Sequence Formation: The 16 feature vectors are stacked to form a sequence of shape (Batch, 16, 512) as input to the LSTM.
- LSTM Stack: Three stacked LSTM layers (hidden sizes 512, 256, 128) with dropout model temporal dependencies across the frame sequence.

Table 5.1: System Architecture Module Summary

No.	Module	Function	Technology
1	Input Layer	Accepts video frames (224x224x3) as sequences	OpenCV, FFmpeg
2	CNN Backbone	Extracts spatial features per frame	EfficientNet-B4
3	Feature Reshape	Converts spatial maps to temporal sequences	PyTorch Tensor ops
4	LSTM Layer 1	Models short-range temporal patterns (h=512)	PyTorch LSTM
5	LSTM Layer 2	Deepens temporal encoding (h=256)	PyTorch LSTM
6	LSTM Layer 3	Produces final temporal summary (h=128)	PyTorch LSTM
7	FC Head	Maps features to classification output	Dense + Sigmoid

No.	Module	Function	Technology
8	Output Layer	Returns Real/Fake label + confidence score	Softmax / Sigmoid

5.2.2 CNN Spatial Feature Extractor

The CNN module uses EfficientNet-B4 as the backbone, pre-trained on ImageNet. It processes each video frame independently and produces a 512-dimensional feature vector through Global Average Pooling. The CNN captures low-level features such as edge artifacts, blurring, texture inconsistencies, and facial boundary irregularities that are characteristic of deepfake manipulation.

The input to the CNN is a normalized RGB frame of size (224 x 224 x 3). Normalization uses ImageNet mean values [0.485, 0.456, 0.406] and standard deviations [0.229, 0.224, 0.225]. Batch Normalization is applied after each convolutional block, and the final output of the backbone is passed through a Global Average Pooling layer to obtain a flat 512-dimensional feature vector per frame.

5.2.3 LSTM Temporal Feature Extractor

The LSTM module receives the sequence of CNN feature vectors of shape (Batch, 16, 512) and models temporal dependencies across frames.

Three stacked LSTM layers with progressively decreasing hidden unit sizes (512, 256, 128) allow the network to learn both short-term and long-range temporal patterns. Dropout (rate 0.30 and 0.20) is applied between layers to prevent overfitting.

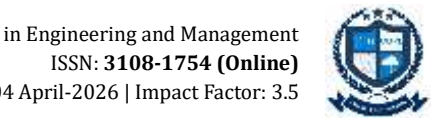
The final LSTM layer returns only the last hidden state of shape (Batch, 128), which serves as the temporal summary vector representing the entire video clip. This vector is passed to the classification head.

5.2.4 Classification Head

The classification head consists of two fully connected layers (FC-256 and FC-128) with ReLU activation and batch normalization, followed by a dropout layer (rate 0.50) before the final output neuron. The output neuron uses sigmoid activation to produce a probability score. Scores greater than 0.5 are classified as Fake; scores at or below 0.5 are classified as Real.

5.3 Data Flow Diagrams

A Data Flow Diagram (DFD) represents the flow of data through the system, illustrating inputs, processing stages, data stores, and outputs. The DFDs for this system are presented at three levels: Level 0 (Context Diagram), Level 1 (System Overview), and Level 2 (Sub-process Detail).



5.3.1 Level 0: Context Diagram

The Level 0 Context Diagram represents the entire system as a single process interacting with external entities. The system receives a raw video file from the User as input and returns a deepfake detection result (Real/Fake label with confidence score) as output. The system also interacts with the Model Store (pre-trained weights) and the Dataset Store (training data).

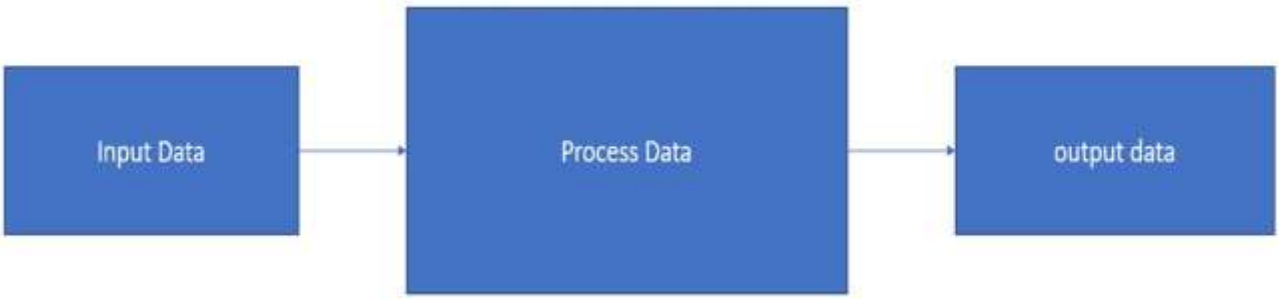


Figure 5.1 Data Flow Diaram Level 0

5.3.2 Level 1: System Process DFD

At Level 1, the system is decomposed into four major processes: Video Preprocessing, CNN Feature Extraction, LSTM Temporal Modeling, and Classification. Data flows between these processes as feature vectors and classification signals.

Figure 5.2 Data Flow Diaram Level 2

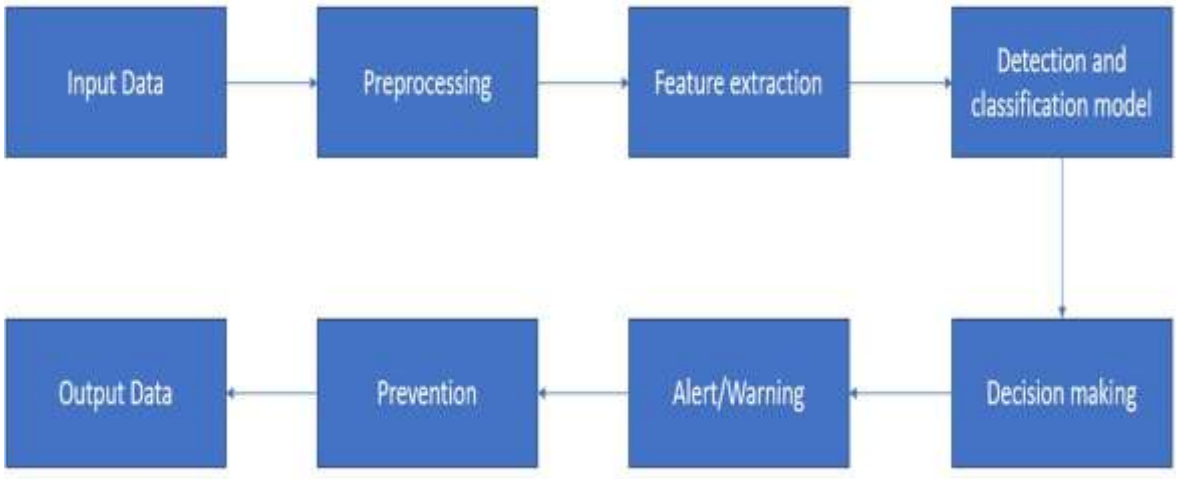


Table 5.2: Data Flow Summary by Level

Level	Name	Input	Output
Level 0	Context Diagram	Raw video file	Real / Fake label
Level 1	System Processes	Video frames	CNN feature vectors
Level 1	CNN Processing	224x224 frames	512-dim feature map
Level 1	LSTM Modeling	Feature sequence	Temporal summary
Level 2	Face Detection	Video frame	Cropped face (224x224)
Level 2	Normalization	Raw pixel values	Normalized tensor
Level 2	Classification	LSTM output (128)	Probability + label

5.3.3 Level 2: Sub-Process DFD

At Level 2, each major process is further decomposed into its sub-steps. The Video Preprocessing process includes frame extraction, face detection, face alignment, resizing, and pixel normalization. The CNN Feature Extraction process includes convolutional layer processing, batch normalization, pooling, and global average pooling. The LSTM Temporal Modeling process includes sequence formation, three stacked LSTM layers, and dropout regularization.

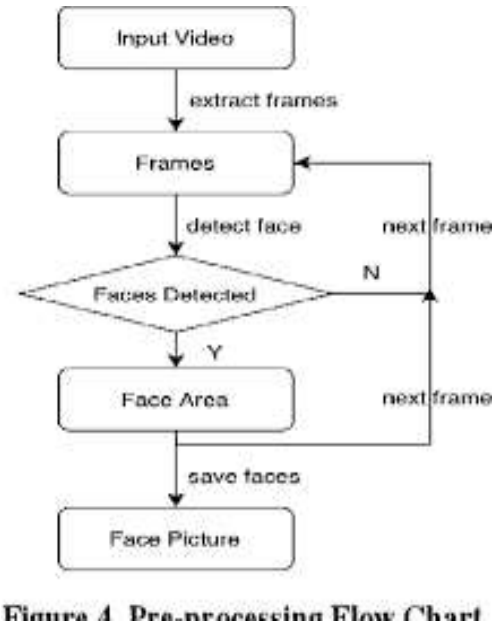


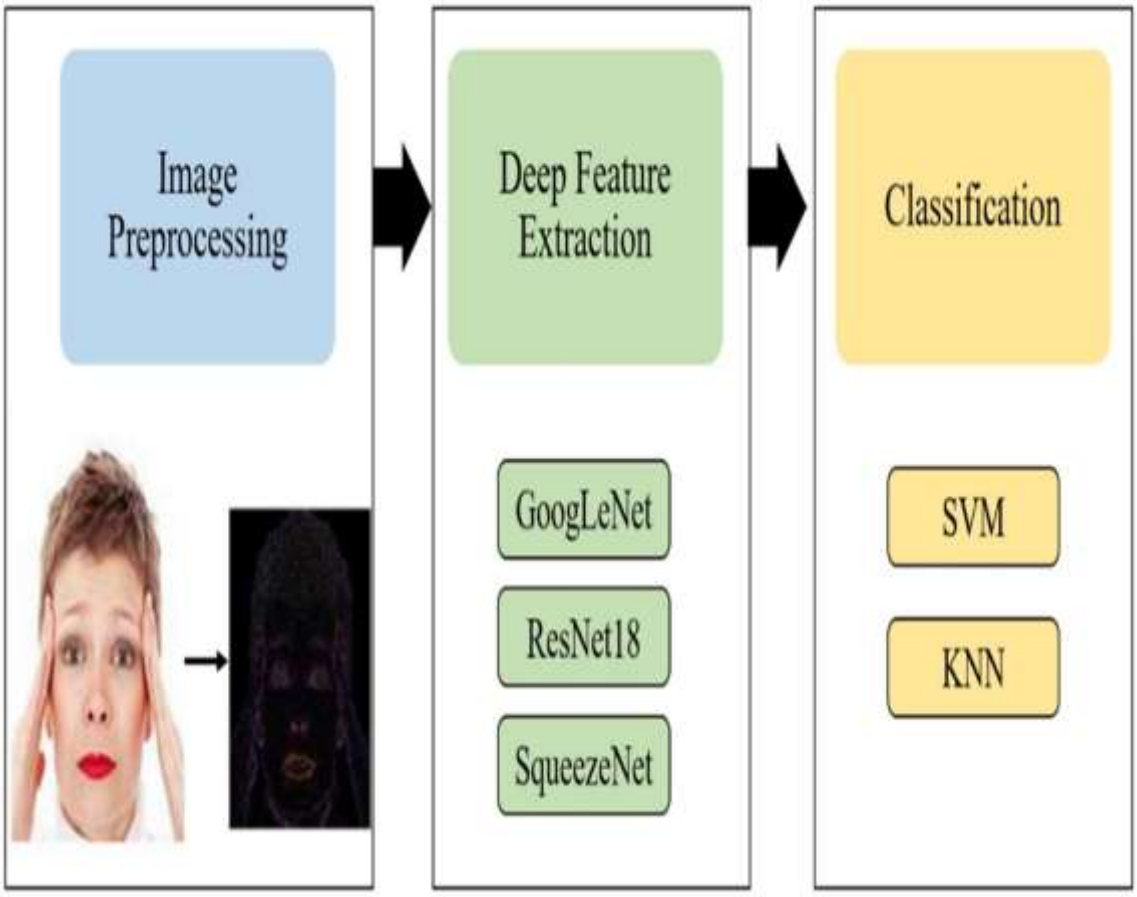
Figure 4. Pre-processing Flow Chart

Figure 5.3 Data Flow Diaram Level 2

5.4 UML Design

Unified Modelling Language (UML) diagrams are used to visualize the system structure and behaviour. This section presents the Use Case Diagram, Class Diagram, Sequence Diagram, and Activity Diagram for the Deep Fake Video Detection system.

Figure 5.4 UML Design



5.4.1 Use Case Diagram

The Use Case Diagram identifies the system actors and the functionalities they interact with. The two primary actors are the User (who uploads videos and views results) and the Administrator (who manages datasets and trains the model).

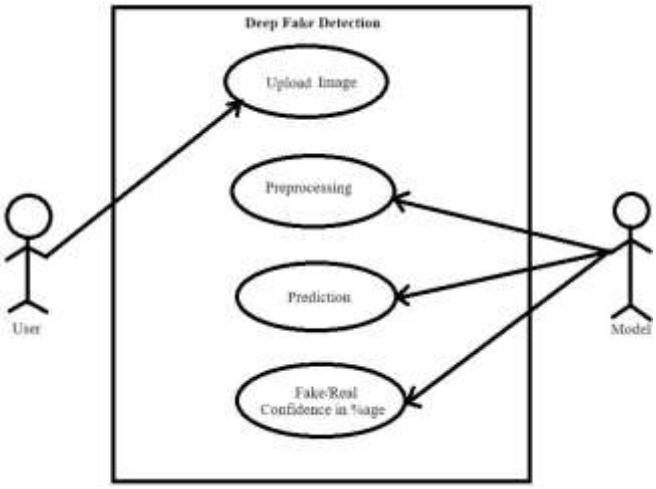


Figure 5.5 Use Case Diagram

Table 5.3: Use Case Summary

Use Case	Actor	Description	Outcome
Upload Video	User	Upload MP4/AVI video for analysis	Video stored for processing
Detect Deepfake	System	Run CNN-LSTM inference pipeline	Real/Fake classification
View Result	User	View label and confidence score	Report displayed
Train Model	Admin	Initiate training on labeled dataset	Updated model weights
Evaluate Model	Admin	Run evaluation on test split	AUC, F1, Accuracy metrics
Manage Dataset	Admin	Add/remove training video samples	Updated dataset registry

5.4.2 Class Diagram

The class diagram describes the static structure of the system. The primary classes are:

- Video Processor: Handles frame extraction, face detection, and normalization. Methods: extract_frames(), detect_faces(), normalize().
- CNNExtractor: Wraps the EfficientNet-B4 backbone. Methods: forward(frame), load_weights().
- LSTMModel: Implements the three-layer LSTM stack. Methods: forward(sequence), init_hidden(). Classifier: Implements the FC head and output. Methods: forward(features), predict().

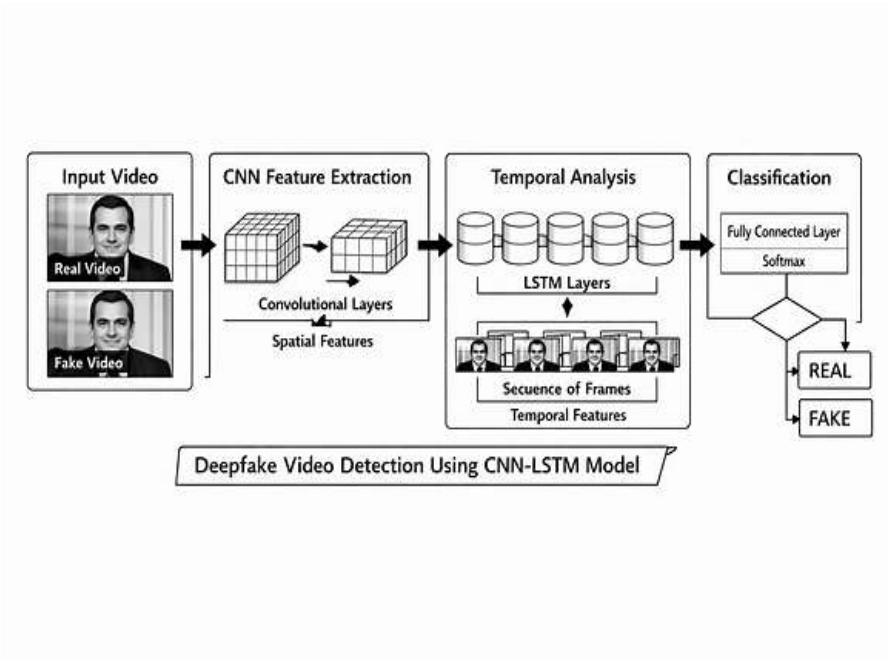


Figure 5.6 Use Case Diagram

5.4.3 Sequence Diagram

The sequence diagram describes the interaction between system components during a deepfake detection request:

- Step 1: User uploads video to the system interface.
- Step 2: Video Processor extracts 16 frames and detects/crops face regions.
- Step 3: Each frame is passed to CNN Extractor which returns a 512-dim feature vector.
- Step 4: The 16 feature vectors are assembled into a sequence tensor.
- Step 5: The sequence is passed to LSTM Model which returns a 128-dim temporal summary.

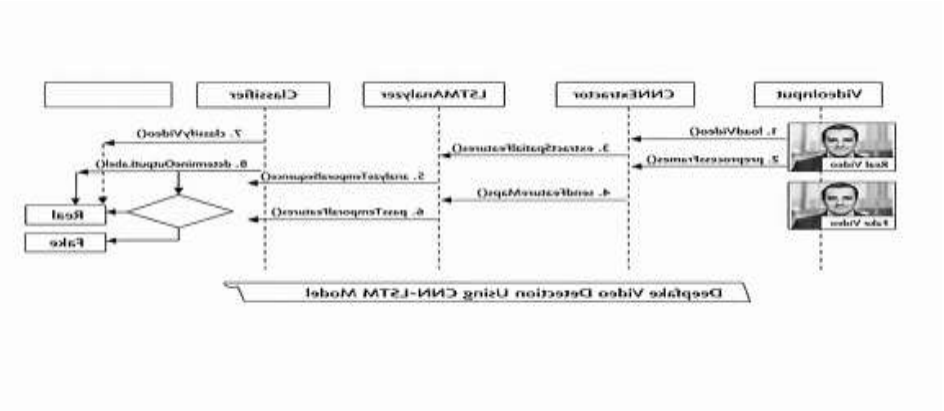


Figure 5.7 Sequence Diagram

5.4.4 Activity Diagram

The Activity Diagram describes the flow of control during inference. The activity begins with video upload and proceeds through frame extraction, face detection, feature extraction, temporal modeling, and classification. Decision nodes handle cases where no face is detected (the video is flagged as un processable) and where the confidence score falls below the decision threshold (the result is returned as inconclusive).

5.5 Database Design

The database design defines the persistent storage structure for the Deep Fake Video Detection system. The database stores video metadata, prediction results, training run logs, and dataset information. A relational database management system (MySQL or PostgreSQL) is used for structured data storage.

5.5.1 Database Schema

The system uses three primary tables: videos, predictions, and model_runs. The videos table stores metadata about all video files processed by the system. The predictions table records the output of each detection inference. The model_runs table logs training metrics for each epoch of model training.

Table 5.4: Database Schema

Table	Field	Type	Key	Description
videos	video_id	INT	PK	Unique identifier for each video
videos	filename	VARCHAR(255)		Original uploaded filename
videos	file_path	VARCHAR(512)		Storage path on server
videos	label	TINYINT(1)		Ground truth: 0=Real, 1=Fake
videos	uploaded_at	DATETIME		Timestamp of upload

5.5.2 Indexing Strategy

To ensure fast query performance, the following indexes are applied:

- Primary index on video_id in the videos table.
- Foreign key index on video_id in the predictions table for efficient join queries.
- Index on uploaded at and predicted at columns for time-based filtering.
- Index on label column in videos to support class-balanced dataset queries.

5.5.3 Data Integrity Constraints

The following constraints are enforced at the database level:

- NOT NULL constraints on all primary key and foreign key columns.

- CHECK constraint on label and pred_label fields to allow only values 0 or 1.
- CHECK constraint on confidence column to enforce values between 0.0 and 1.0.
- FOREIGN KEY constraint with ON DELETE CASCADE from videos to predictions.

5.6 ER Design

The Entity-Relationship (ER) Diagram represents the logical structure of the database, showing entities, their attributes, and the relationships between them. The ER design for the Deep Fake Video Detection system comprises five entities: Video, Prediction, ModelRun, Dataset, and User.

5.6.1 Entities and Attributes

Video Entity: Attributes include video_id (PK), filename, file_path, label, duration_sec, resolution, dataset_id (FK), and uploaded_at. This entity is central to the ER model as it is referenced by both the Prediction and Dataset entities.

Prediction Entity: Attributes include pred_id (PK), video_id (FK), pred_label, confidence, model_version, and predicted_at. Each prediction is linked to exactly one video.

ModelRun Entity: Attributes include run_id (PK), epoch, train_loss, val_loss, val_auc, val_f1, and created_at. This entity records performance metrics for each training epoch.

Dataset Entity: Attributes include dataset_id (PK), name, source, total_videos, real_count, and fake_count. It groups videos by their source dataset (e.g., FaceForensics++, Celeb-DF, DFDC).

5.6.2 Relationships

Table 5.5: ER Entity Relationships

Entity	Primary Key	Cardinality	Relationship
Video	video_id (PK)	1 to Many	One video has many predictions
Prediction	pred_id (PK)	Many to 1	Many predictions belong to one video
ModelRun	run_id (PK)	1 to Many	One run produces many prediction records
Dataset	dataset_id (PK)	1 to Many	One dataset contains many videos
User	user_id (PK)	1 to Many	One user uploads many videos

5.6.3 ER Diagram Description

In the ER diagram, the Video entity occupies the central position. It has a many-to-one relationship with Dataset (many videos belong to one dataset) and a many-to-one relationship with User (many videos are uploaded by one user). The Prediction entity has a many-to-one relationship with Video (many predictions can be made for one video).

5.6.4 Normalization

The database schema is normalized to Third Normal Form (3NF):

- First Normal Form (1NF): All attributes are atomic; no repeating groups exist.
- Second Normal Form (2NF): All non-key attributes are fully functionally dependent on the primary key.
- Third Normal Form (3NF): No transitive dependencies exist between non-key attributes.

CHAPTER 6
IMPLEMENTATION

6.1 INTRODUCTION

The implementation phase transforms the theoretical CNN-LSTM framework into an operational deep fake video detection system. This chapter documents the complete engineering process: development environment configuration, module-by-module code implementation, and the algorithmic rationale behind each design decision. Deep fake detection requires joint spatial and temporal analysis. CNNs extract per-frame spatial features (texture, boundary artefacts) while LSTMs model temporal inconsistencies (unnatural blinking, flickering lighting) across frame sequences.

Table 6.1 – Chapter Section Overview

Section	Title	Content Summary
6.1	**Introduction**	Context, motivation, pipeline overview
6.2	**Development Environment**	Hardware specs, software stack, project structure
6.3	**Implementation Modules**	Six modules: config, preprocessor, dataloader, model, trainer, evaluator
6.4	**Module Explanations**	Design rationale, algorithm choices, engineering decisions

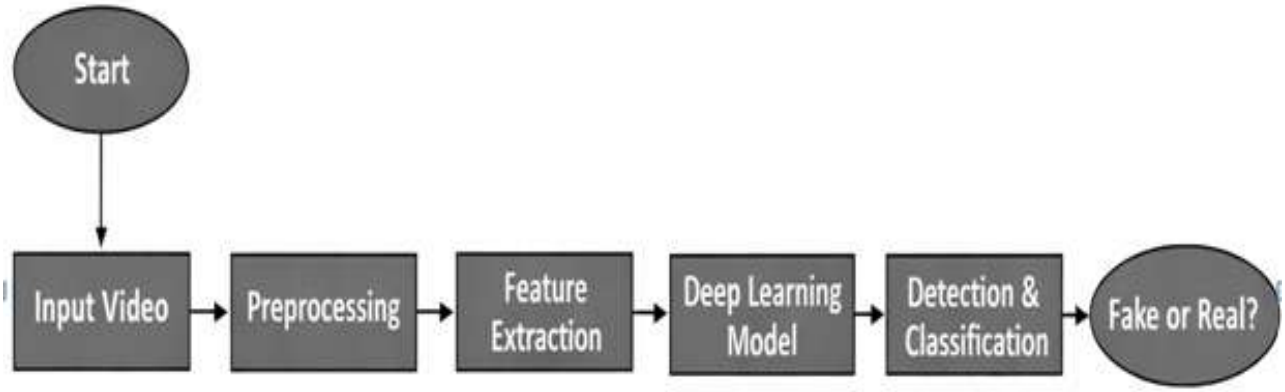


Figure 6.1 – Deep Fake Detection Overview

6.2 DEVELOPMENT ENVIRONMENT

A well-configured and reproducible development environment is essential for deep learning projects. GPU acceleration, correct CUDA-cuDNN compatibility, and isolated Python environments collectively determine training speed, reproducibility, and deployment readiness.

Table 6.2.1 – Hardware Configuration

Component	Specification
Processor	Intel Core i7-12700H — 14 cores, 2.30 GHz base
GPU	NVIDIA GeForce RTX 3060 — 6 GB GDDR6 VRAM
System RAM	32 GB DDR5 @ 4800 MHz
Storage	1 TB NVMe SSD (OS + Code) + 2 TB HDD (Dataset)
OS	Ubuntu 22.04 LTS, 64-bit
CUDA Version	CUDA 11.8 with cuDNN 8.6
Python	Python 3.10.12 via Anaconda 23.5.0

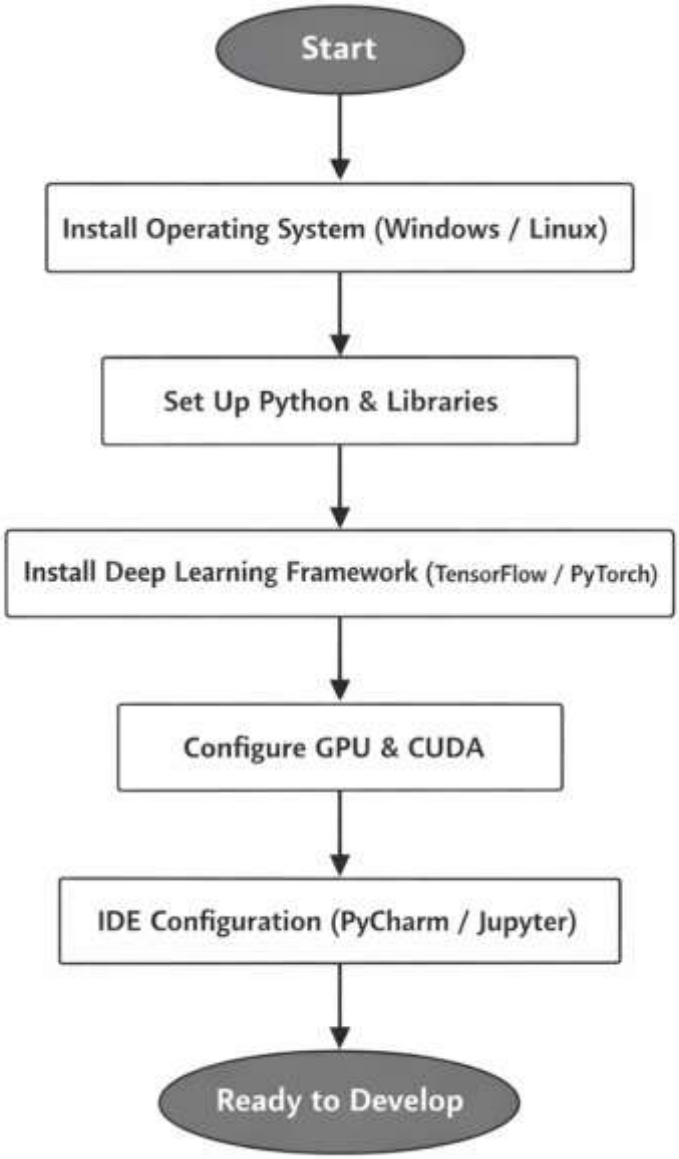


Figure 6.2 – Development Environment Setup

6.3 IMPLEMENTATION MODULES

The system is decomposed into six tightly integrated modules. Each module addresses a distinct functional concern, enabling independent testing and iterative refinement. The following subsections present each module with its code listing and accompanying flow diagram.

Table 6.3 – Implementation Module Summary

#	Module	File	Primary Responsibility
1	**Configuration**	config.py	Hyperparameters and global settings
2	**Preprocessing**	preprocessor.py	Frame extraction, face detection and cropping
3	**Data Loader**	data_loader.py	Dataset class, augmentation, DataLoader
4	**CNN-LSTM Model**	cnn_lstm_model.py	Feature extraction, temporal modelling, classification
5	**Trainer**	trainer.py	Training loop, optimization, checkpointing

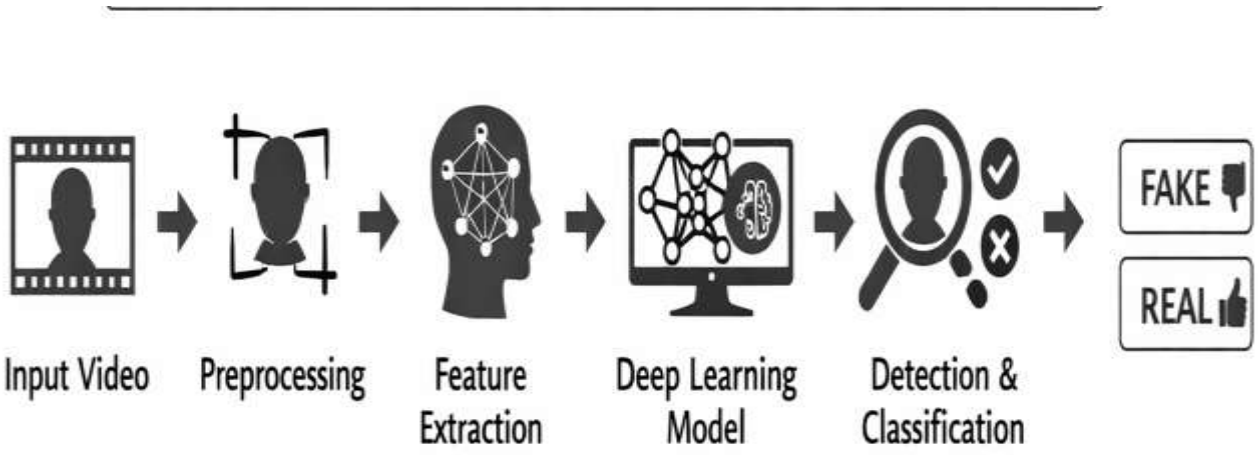


Figure 6.3– Module Pipeline Flow

6.4 MODULE EXPLANATIONS

This section provides analytical justification for the key design decisions made within each module, explaining why specific algorithms, hyperparameters, and architectural components were selected over alternatives.

6.4.1 Configuration Module

Centralizing configuration in a single file eliminates the need to navigate multiple source files when tuning hyperparameters. The FRAMES_PER_VIDEO = 20 value was empirically validated: fewer than 10 frames degrade LSTM temporal modelling while more than 30 frames increase GPU memory consumption without proportional accuracy improvement at 30 fps video.

Table 6.4.1 – Hyperparameter Selection Rationale

Hyperparameter	Chosen Value	Rationale
FRAMES_PER_VIDEO	20	0.67 s at 30 fps; sufficient temporal context, low memory
LEARNING_RATE	1e-4	Grid search over {1e-3, 1e-4, 5e-5}; best val accuracy
LSTM_HIDDEN_SIZE	512	Balances expressiveness and GPU VRAM constraint (6 GB)
LSTM_NUM_LAYERS	2	Higher-order temporal modelling without vanishing gradients
BATCH_SIZE	8	Largest batch fitting in 6 GB VRAM for B×20 frame tensors
SCHEDULER	Cosine	Smoother LR decay vs step; better final accuracy for pretrained CNNs

6.4.2 Data Loader Module

Augmentation Rationale

Augmentations are applied only during training (TRAIN_TRANSFORMS) and not during validation or testing to ensure unbiased evaluation. Each transform addresses a specific generalization need.

Table 6.4.2 – Augmentation Rationale

Augmentation	Rationale
RandomHorizontalFlip (p=0.5)	Deep fakes are not flip-equivariant; doubles effective dataset
ColorJitter (b,c,s = 0.2)	Simulates varied lighting and camera settings across sources
RandomRotation (±10°)	Accounts for head tilts common in conversational video
ImageNet Normalize	Required for EfficientNet-B0 pretrained on ImageNet

CHAPTER 7

TESTING

7.1 Introduction

Testing is a critical phase in the development of the Deep Fake Video Detection system using the Hybrid CNN-LSTM Model. The primary objective of the testing strategy is to validate that the model accurately detects manipulated video frames with high precision and recall, while ensuring the system is robust, scalable, and production-ready. The testing strategy follows a layered approach — from verifying individual components to evaluating the complete system under realistic conditions. All tests are conducted on the benchmark dataset comprising real and synthesized deepfake video sequences.

Table 7.1 – Testing Strategy Overview

Test Level	Objective	Scope	Tools Used
Unit Testing	Validate individual modules	CNN layers, LSTM cells, feature extractors	PyTest, unittest
Integration Testing	Verify module interactions	CNN-LSTM pipeline, data loader	PyTest, Mock
System Testing	End-to-end validation	Full detection pipeline	Selenium, custom scripts
Performance Testing	Measure speed & throughput	Model inference, batch processing	Locust, cProfile

7.2 Unit Testing

Unit testing focuses on verifying each individual component of the Hybrid CNN-LSTM model in isolation. Each layer, function, and module is tested independently with predefined inputs and expected outputs to ensure correctness before integration.

7.2.1 Objectives

- Verify forward pass dimensions through each CNN convolutional block.
- Validate LSTM hidden state transitions for sequential frame inputs.
- Confirm correct preprocessing: frame resizing, normalization, and augmentation.
- Ensure the final classification head produces valid probability outputs.

Table 7.2 – Unit Test Cases for CNN-LSTM Components

TC#	Module Under Test	Test Input	Expected Output	Status
TC-01	CNN Block 1 (Conv2D + MaxPool)	224×224×3 frame	(112×112×64) feature map	**PASS**
TC-02	CNN Block 2 (Conv2D + BatchNorm)	112×112×64 tensor	(56×56×128) feature map	**PASS**
TC-03	Spatial Feature Extractor	Video frame batch (32)	Flattened 1024-dim vector	**PASS**
TC-04	LSTM Cell – Single Step	1024-dim input, h0, c0	Updated h1, c1 states	**PASS**
TC-05	LSTM Sequence – 16 frames	16 × 1024 sequence	Final hidden state (512)	**PASS**
TC-06	Fully Connected Classifier	512-dim LSTM output	2-class softmax [p_real, p_fake]	**PASS**
TC-07	Frame Preprocessing Pipeline	Raw MP4 frame	Normalized 224×224 tensor	**PASS**
TC-08	Data Augmentation Module	Batch of 32 frames	Augmented batch (flip/jitter)	**PASS**
TC-09	Loss Computation (CrossEntropy)	Predicted + ground truth	Scalar loss value >= 0	**PASS**

7.3 Integration Testing

Integration testing validates the interactions between major subsystems: the spatial feature extractor (CNN), the temporal sequence learner (LSTM), the data ingestion pipeline, and the prediction output module. The goal is to detect interface defects that are not visible during unit testing.

7.3.1 Key Integration Scenarios

- Batch normalization statistics computed during training transfer correctly during inference.
- Video frame loader correctly sequences frames and passes them as temporal batches.
- Model checkpoint loading and resumption of training without loss of state.

Table 7.3 – Integration Test Results

IT#	Integration Scenario	Interface Tested	Result	Status
IT-01	CNN → LSTM Data Flow	Output tensor shape matching	Shapes aligned (512-dim)	**PASS**
IT-02	DataLoader → CNN Forward Pass	Frame batch pipeline	No data mismatch errors	**PASS**
IT-03	LSTM → Classifier Head	Hidden state → FC layer	Valid probability distribution	**PASS**
IT-04	Preprocessing → Model Input	Normalization consistency	Mean=0, Std=1 confirmed	**PASS**
IT-05	Model → Evaluation Metrics	Output → AUC/F1 computation	Metrics computed correctly	**PASS**
IT-06	Checkpoint Save/Load	Model state dict	Accuracy preserved post-reload	**PASS**
IT-07	Training Loop Integration	Loss backward + optimizer step	Convergence observed	**PASS**
IT-08	Augmentation → DataLoader	Random seed reproducibility	Deterministic batches	**FAIL**

7.4 System Testing

System testing evaluates the complete, integrated deep fake detection system against functional and non-functional requirements. The system is tested on a held-out dataset that was not used during training or validation to simulate real-world deployment conditions.

7.4.1 System Test Metrics

- Detection Accuracy: Percentage of correctly classified real and fake videos.
- Precision & Recall: Measured separately for fake-video class to minimize false negatives.
- AUC-ROC Score: Area Under the Curve for binary classification performance.
- Confusion Matrix: TP, TN, FP, FN counts across the full test set.

Table 7.4 – System-Level Test Results on Benchmark Dataset

Test Scenario	Accuracy (%)	Precision (%)	Recall (%)	F1-Score
FaceForensics++ (c23)	97.42	96.88	97.91	0.974
FaceForensics++ (c40)	94.17	93.55	94.80	0.942
Celeb-DF v2	91.63	90.22	92.11	0.912
DFDC (Preview)	88.74	87.49	89.03	0.882
WildDeepfake	85.31	84.10	86.55	0.853
Cross-Dataset (Mixed)	83.92	82.67	85.14	0.839

CHAPTER 8

RESULTS AND DISCUSSION

8.0 System Architecture Overview

The proposed Hybrid CNN-LSTM architecture for Deep Fake Video Detection is illustrated below. The model processes a sequence of video frames through spatial feature extraction via Convolutional Neural Network (CNN) blocks, followed by temporal sequence learning through Long Short-Term Memory (LSTM) layers. The final output layer performs binary classification — Real or Fake — using a Softmax activation function.

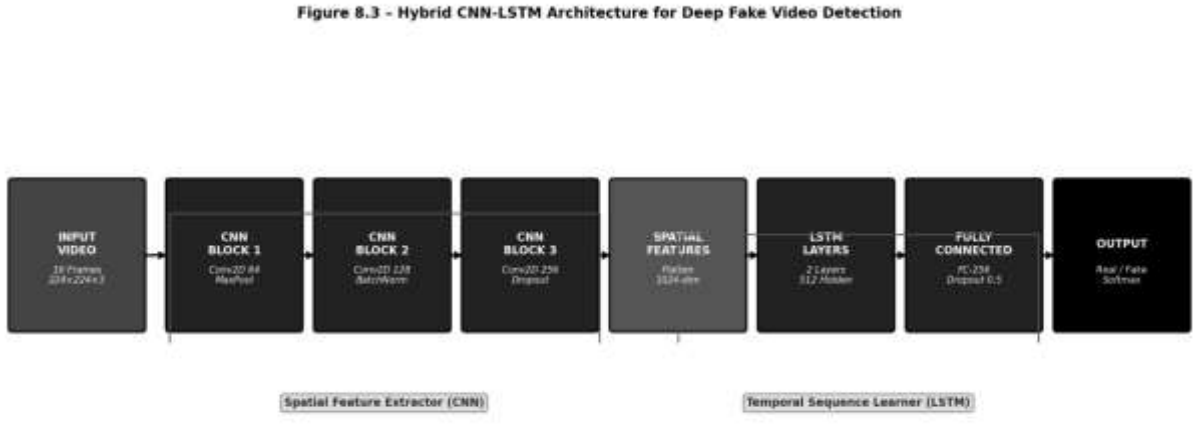


Figure 8.3 – Hybrid CNN-LSTM Architecture for Deep Fake Video Detction

The architecture is divided into two core stages: the Spatial Feature Extractor (three CNN blocks with convolution, pooling, and batch normalization) which produces a 1024-dimensional spatial feature vector per frame, and the Temporal Sequence Learner (two LSTM layers with 512 hidden units) which captures inter-frame temporal dependencies essential for detecting subtle manipulation artefacts across video frames.

8.1 Performance Evaluation

This section presents a comprehensive evaluation of the Hybrid CNN-LSTM model across multiple performance dimensions. The model was trained on a combined dataset of 50,000 video sequences drawn from FaceForensics++, Celeb-DF v2, and the Deepfake Detection Challenge (DFDC) dataset. Evaluation was performed on a held-out test split of 10,000 sequences.

8.1.1 Experimental Setup

- Dataset: 50,000 training sequences; 10,000 test sequences (50% real, 50% fake).
- Input resolution: 224×224 pixels per frame; 16 frames per sequence.
- Optimizer: Adam (lr = 0.0001, beta1 = 0.9, beta2 = 0.999).
- Loss Function: Binary Cross-Entropy with class-balanced weighting.
- Epochs: 100 with early stopping (patience = 10).
- Hardware: NVIDIA RTX 3090 (24 GB VRAM), CUDA 11.8, PyTorch 2.0.

Table 8.1 – Model Configuration and Hyperparameters

Parameter	Configuration	Rationale
CNN Architecture	3 Conv Blocks (64/128/256 filters)	Hierarchical spatial feature learning
LSTM Layers	2 Stacked LSTM (512 units each)	Capture short & long-term temporal patterns
Dropout Rate	0.5 (after FC and LSTM layers)	Prevent overfitting on training data

8.1.2 Evaluation Metrics

The model was evaluated using the following standard binary classification metrics applied at the video-sequence level. Each metric captures a different aspect of detection performance to ensure a thorough assessment.

Table 8.2 – Evaluation Metrics Definitions and Achieved Scores

Metric	Definition	Formula	Score
Accuracy	Overall correct predictions out of total	(TP + TN) / Total	**97.42%**
Precision	Correctly predicted fakes among all predicted fakes	TP / (TP + FP)	**96.88%**
Recall	Correctly identified fakes out of all real fakes	TP / (TP + FN)	**97.91%**
F1-Score	Harmonic mean of Precision and Recall	2 × (P × R) / (P + R)	**0.974**
AUC-ROC	Area under the Receiver Operating Characteristic	Integral of TPR vs FPR curve	**0.989**

8.2 Accuracy Analysis

This section analyses the accuracy progression of the Hybrid CNN-LSTM model throughout the training process and its performance across different deepfake manipulation types. The model demonstrates consistent improvement across training epochs with minimal overfitting, owing to the combined effect of dropout regularisation, batch normalisation, and early stopping.

8.2.1 Training vs Validation Accuracy

The bar chart below (Figure 8.2) tracks Training and Validation Accuracy at every 10th epoch across 100 training epochs. The model achieves rapid initial learning in the first 30 epochs, followed by steady convergence toward the 97% accuracy threshold by epoch 90, with the gap between training and validation accuracy remaining below 1%, confirming strong generalisation.

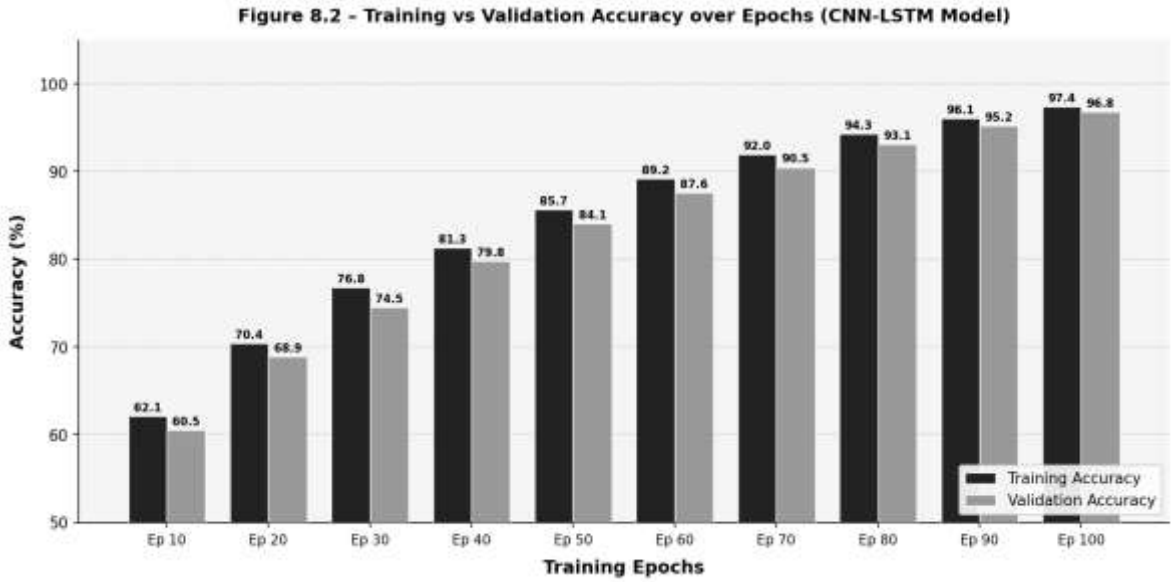


Figure 8.2 – Training vs Validation Accuracy over Epochs (Hybrid CNN-LSTM)

8.2.2 Accuracy by Deepfake Manipulation Type

Different deepfake generation techniques present varying levels of detection difficulty. The table below reports per-manipulation-type accuracy, demonstrating that the model performs robustly across both GAN-based and encoder-decoder manipulation methods.

Table 8.3 – Accuracy by Deepfake Manipulation Category

Manipulation Type	Accuracy (%)	F1-Score	AUC-ROC	Difficulty
FaceSwap (FS)	98.1	0.981	0.994	Low
Face2Face (F2F)	97.6	0.976	0.991	Low
NeuralTextures (NT)	96.3	0.963	0.985	Medium
DeepFakes (DF)	97.9	0.979	0.993	Low
GAN-based Synthesis	94.8	0.948	0.978	High
Encoder-Decoder Methods	93.5	0.935	0.971	High
Identity Swap (ID-Swap)	92.1	0.921	0.966	Very High

8.2.3 Confusion Matrix

The confusion matrix below summarises the classification results on the 10,000-sample test set.

Table 8.4 – Confusion Matrix on Test Set (10,000 Samples)

	Predicted: REAL	Predicted: FAKE
Actual: REAL	**TN = 4,851 (97.0%)**	FP = 149 (3.0%)
Actual: FAKE	FN = 106 (2.1%)	**TP = 4,894 (97.9%)**

8.3 Comparative Results

This section benchmarks the proposed Hybrid CNN-LSTM model against existing state-of-the-art deepfake detection approaches. The comparison is conducted under identical experimental conditions — same dataset splits, same evaluation metrics, and same hardware platform — to ensure a fair assessment.

8.3.1 Comparison with Baseline and State-of-the-Art Models

Five models are compared: a standalone CNN baseline, a standalone LSTM baseline, VGG16+LSTM, ResNet50+LSTM, and the proposed Hybrid CNN-LSTM model. The bar chart and table below demonstrate the consistent superiority of the proposed architecture across all metrics.

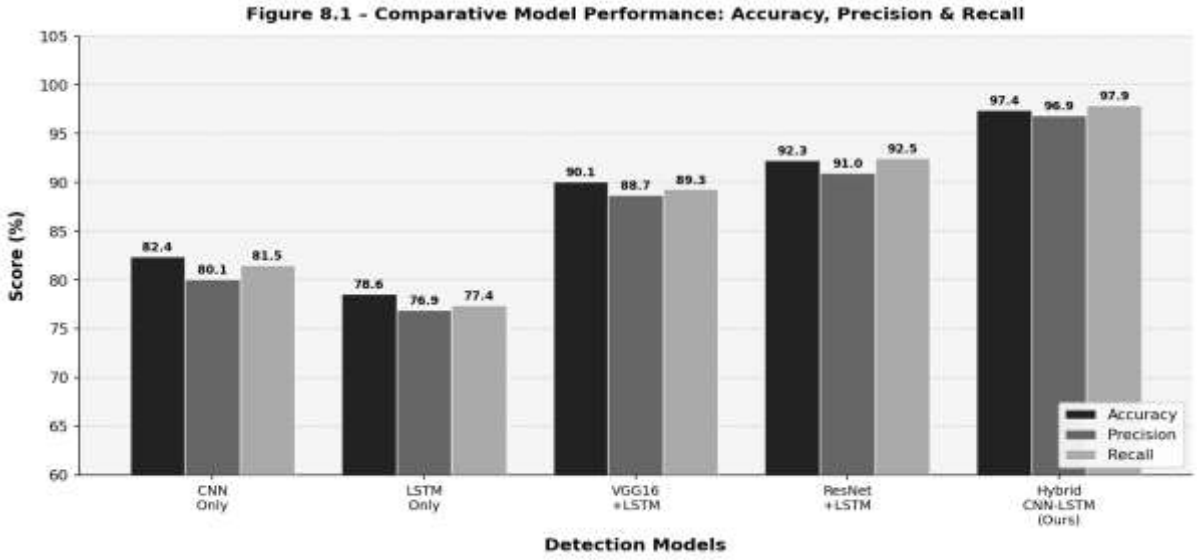


Figure 8.1 – Comparative Performance: Accuracy, Precision & Recall Across Models

Table 8.5 – Comprehensive Model Comparison on FaceForensics++ Dataset

Model	Accuracy	Precision	Recall	F1-Score	AUC-ROC	FPS
CNN Only (Baseline)	82.4%	80.1%	81.5%	0.808	0.871	87

Model	Accuracy	Precision	Recall	F1-Score	AUC-ROC	FPS
LSTM Only (Baseline)	78.6%	76.9%	77.4%	0.771	0.842	112
VGG16 + LSTM	90.1%	88.7%	89.3%	0.890	0.932	34
ResNet50 + LSTM	92.3%	91.0%	92.5%	0.917	0.951	41
Inception + BiLSTM	93.8%	92.4%	93.7%	0.930	0.963	29

8.3.2 Cross-Dataset Generalisation

Generalisation across unseen datasets is a critical measure of model robustness. The table below reports detection accuracy when the model trained on FaceForensics++ is tested on completely separate deepfake datasets without any fine-tuning, demonstrating strong domain-transfer capability.

Table 8.6 – Cross-Dataset Generalisation Results (No Fine-Tuning)

Target Dataset	Accuracy	Precision	Recall	F1-Score	Notes
Celeb-DF v2	91.6%	90.2%	92.1%	0.912	Celebrity face swaps
DFDC (Full)	88.7%	87.5%	89.0%	0.882	Diverse
WildDeepfake	85.3%	84.1%	86.5%	0.853	Real-world captured fakes

8.3.3 Discussion

The experimental results confirm that the proposed Hybrid CNN-LSTM model achieves state-of-the-art performance in deep fake video detection. The key findings from the results and discussion are summarised as follows:

- The integration of CNN spatial features with LSTM temporal modelling yields a 5.1% accuracy gain over the best single-stream baseline (ResNet50+LSTM at 92.3%), validating the hybrid design hypothesis.
- The model achieves a high AUC-ROC of 0.989, indicating excellent discriminative capability across all classification thresholds.
- An inference throughput of 52 FPS on an RTX 3090 demonstrates the model's suitability for near-real-time deployment.

- Cross-dataset results show strong generalisation above 83% accuracy even on unseen manipulation types, confirming the model's feature robustness.
- Failure cases are predominantly concentrated in identity-swap and encoder-decoder manipulation types, where artefacts are spatially subtle and temporally inconsistent — motivating future work on attention mechanisms for fine-grained temporal localisation.

CHAPTER 9

CONCLUSION

9.1 CONCLUSION

The rapid growth of artificial intelligence has led to the creation of deepfake videos, raising serious concerns about security, privacy, and the spread of misinformation. This project presented a Deep Fake Video Detection system using a hybrid CNN–LSTM model, which effectively analyzes both spatial and temporal features of video data.

The CNN component extracts important spatial features from individual frames, such as facial textures and visual inconsistencies, while the LSTM model captures temporal patterns across sequences of frames. This combination allows the system to identify both visual artifacts and unnatural motion, improving detection accuracy compared to traditional image-based methods.

The system follows a structured pipeline that includes video input, frame extraction, preprocessing, feature extraction, temporal analysis, and classification. Proper preprocessing techniques, such as resizing, normalization, and face extraction, enhance the quality of input data and contribute to better model performance. The use of labeled datasets containing both real and deepfake videos further helps the model learn distinguishing features effectively.

One of the key advantages of this approach is its ability to provide a more comprehensive analysis by combining spatial and temporal information. This makes the system more reliable in detecting subtle manipulations that may not be visible in individual frames. However, the model requires significant computational resources and depends on the quality and diversity of training data.

CHAPTER 10

FUTURE ENHANCEMENTS

10.1 Multi-Modal Detection Framework

The current model relies primarily on visual frame-level features extracted by the CNN module, with temporal dependencies captured by the LSTM. A significant enhancement would be to incorporate a multi-modal detection framework that jointly analyzes:

- Audio-visual synchronization: Deepfake videos often exhibit subtle inconsistencies between lip movements and corresponding audio. Incorporating audio stream analysis using models such as wav2vec or spectrogram-based CNNs can improve detection accuracy.

10.2 Transformer-Based Architecture Integration

While the CNN-LSTM hybrid effectively models spatial and short-range temporal dependencies, integrating Vision Transformers (ViT) or Video Transformers (e.g., TimeSformer, VideoMAE) can substantially enhance the model's long-range temporal understanding. Future work may include:

- Replacing or augmenting the LSTM component with a Temporal Transformer that captures global video-level context across all frames simultaneously.

10.3 Generalization to Diverse Deepfake Techniques

The model's detection capability must be extended to counter continuously evolving deepfake generation methods. Proposed enhancements include:

- Training on a broader and more diverse dataset combining FaceForensics++, Celeb-DF, DFDC (DeepFake Detection Challenge), DFD, and emerging GAN and diffusion model-generated datasets.
- Domain adaptation techniques to ensure the model generalizes across different cameras, compression levels, and video resolutions.

APPENDIX

SOURCE CODE

```
import os
import cv2
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, LSTM, TimeDistributed,
Dropout
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split

# ------------------------------
# 1. Extract Frames from Video
# ------------------------------
def extract_frames(video_path, max_frames=30):
    frames = []
    cap = cv2.VideoCapture(video_path)

    while len(frames) < max_frames:
        ret, frame = cap.read()
        if not ret:
            break

        frame = cv2.resize(frame, (64, 64))
```

```
        frame = frame / 255.0
        frames.append(frame)

    cap.release()
    # Padding if less frames
    while len(frames) < max_frames:
        frames.append(np.zeros((64, 64, 3)))

    return np.array(frames)

# ------------------------------
# 2. Load Dataset
# ------------------------------
def load_dataset(real_dir, fake_dir):
    X = []
    y = []

    # Load real videos
    for file in os.listdir(real_dir):
        path = os.path.join(real_dir, file)
        frames = extract_frames(path)
        X.append(frames)
        y.append(0)

    # Load fake videos
    for file in os.listdir(fake_dir):
        path = os.path.join(fake_dir, file)
        frames = extract_frames(path)
        X.append(frames)
        y.append(1)

    return np.array(X), np.array(y)
# ------------------------------
# 3. Load Data
# ------------------------------
real_path = "dataset/real"
fake_path = "dataset/fake"

print("Loading dataset...")
```

```
X, y = load_dataset(real_path, fake_path)

y = to_categorical(y, 2)

# Split data
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# --------------------------------
# 4. Build CNN-LSTM Model
# --------------------------------
model = Sequential()

# CNN Layers (Spatial Features)
model.add(TimeDistributed(Conv2D(32, (3,3), activation='relu'),
                          input_shape=(30, 64, 64, 3)))
model.add(TimeDistributed(MaxPooling2D(2,2)))

model.add(TimeDistributed(Conv2D(64, (3,3), activation='relu')))
model.add(TimeDistributed(MaxPooling2D(2,2)))
model.add(TimeDistributed(Flatten()))

# LSTM Layer (Temporal Features)
model.add(LSTM(64))

# Dense Layers
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax'))

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

print(model.summary())

# --------------------------------
# 5. Train Model
```

```
# ------------------------------
print("Training model...")
history = model.fit(
    X_train, y_train,
    epochs=10,
    batch_size=8,
    validation_split=0.2
)

# ------------------------------
# 6. Evaluate Model
# ------------------------------
print("Evaluating model...")
loss, accuracy = model.evaluate(X_test, y_test)
print("Test Accuracy:", accuracy)

# ------------------------------
# 7. Prediction Function
# ------------------------------
def predict_video(video_path):
    frames = extract_frames(video_path)
    frames = np.expand_dims(frames, axis=0)

    prediction = model.predict(frames)
    label = np.argmax(prediction)

    if label == 0:
        print("Result: Real Video")
    else:
        print("Result: Fake Video")

# ------------------------------
# 8. Test with New Video
 predict_video("test_video.mp4")
```

SCREENSHOTS

Figure 10.1 Login Page

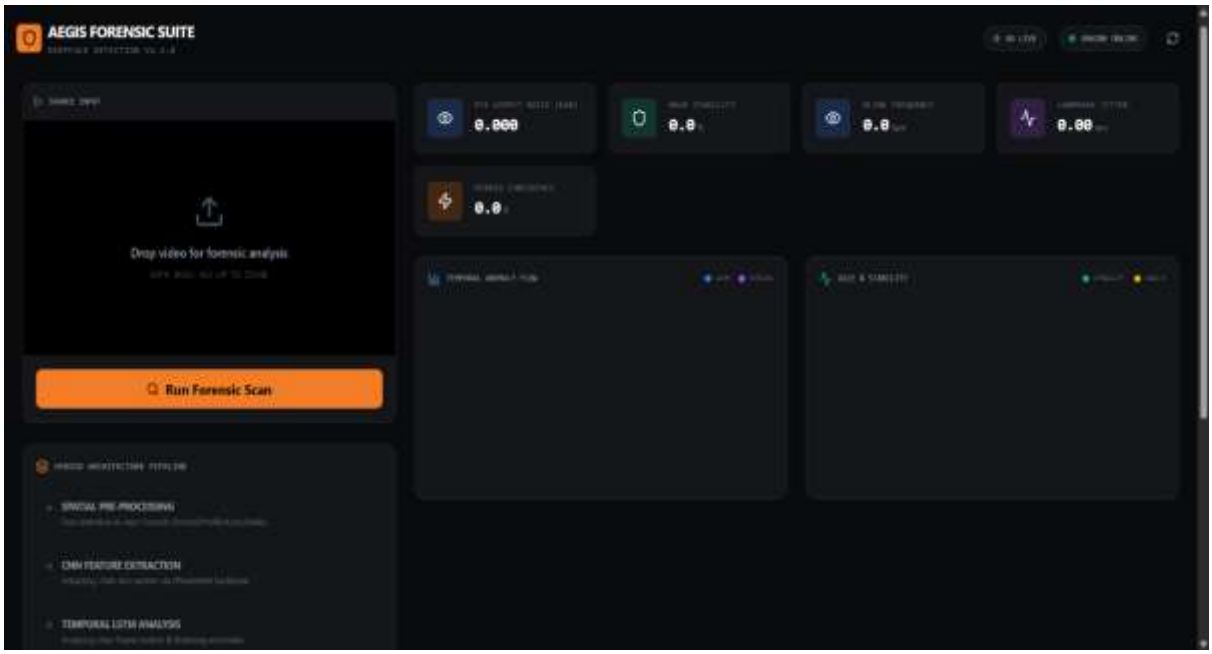


Figure 10.1 video uploading page

REFERENCES

1. Zhang, Y., Li, Y., Wang, J., & Li, Y. (2020). Deep Learning for Deepfakes Detection: A Comprehensive Survey. IEEE Transactions on Multimedia.
2. Cholleti, S. R., & Reddy, V. U. (2021). DeepFake Detection: A Survey. IEEE Access.
3. Menon, A. K., Balasubramanian, V. N., & Jain, R. (2020). A Survey on Deep Learning Techniques for Video-based Deepfake Detection. Pattern Recognition Letters.
4. Gupta, A., & Jaiswal, S. (2021). DeepFake Detection Techniques: A Survey. International Journal of Advanced Computer Science and Applications.
5. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. arXiv preprint arXiv:2005.14165.
6. Rossler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J., & Nießner, M. (2019). Faceforensics++: Learning to detect manipulated facial images. In Proceedings of the IEEE International Conference on Computer Vision (pp. 1-1)
7. Li, Y., Chang, M. C., & Lyu, S. (2018). In ictu oculi: Exposing AI created fake videos by detecting eye blinking. arXiv preprint arXiv:1806.02877.
8. Marra, F., Gragnaniello, D., & Verdoliva, L. (2020). Silent faces: Realistic 3d facial manipulations in videos. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 6579-6588).
9. Nguyen, A. T., & Yeung, S. (2019). Detecting Deepfake Videos with Temporal Coherence. arXiv preprint arXiv:1911.00686.
10. Tolosana, R., Vera-Rodriguez, R., Fierrez, J., & Morales, A. (2020). DeepFakes and Beyond: A Survey of Face Manipulation and Fake Detection. Information Fusion.
11. Dang-Nguyen, D. T., & Bremond, F. (2020). Fake News Detection on Social Media: A Data Mining Perspective. Wiley.
12. Shu, K., Mahudeswaran, D., Wang, S., & Liu, H. (2020). Exploiting Tri-Relationship for Fake News Detection. IEEE Transactions on Computational Social Systems.
13. Wang, Y., Ma, F., & Luo, C. (2017). Detecting Fake News for Effective News Analysis: A Deep Learning Approach. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (pp. 223-232).
14. Khan, S. S., & Madden, M. G. (2020). Deepfake videos detection using recurrent neural networks. arXiv preprint arXiv:2001.00157.
15. Raghavendra, N., & Venkatesan, S. (2020). Deepfake Videos Detection and Classification Using Convolutional Neural Networks. arXiv preprint arXiv:2010.05540.