ISSN 2277-2685
IJESR/Jan-Mar. 2026/ Vol-16/Issue-1/361-369
Mr.Ch .Gopi *et. al.*, / International Journal of Engineering & Science Research

Full Length Article

Fake Face Detection Based On Videos Using Opencv And Neural Network Architecture

Mr.Ch .Gopi[1],Sk.Ayeshajabeen[2],P.Sathwika[3],R.Thrisha[4]
[1]Assistant Professor ; Department Of Information Technology , Guru Nanak Institutions Technical Campus, Hyderabad, India.
[2,3,4]B.Tech Students; Department Of Information Technology , Guru Nanak Institutions Technical Campus, Hyderabad, India.
Mail Id; shaikayeshajabeen33@gmail.com[2]

Article Received 26-02-2026, Accepted 24-03-2026

ABSTRACT:
The rapid development of the Internet has enabled the widespread distribution of manipulated facial images, particularly Deepfakes, which are increasingly difficult to detect using conventional methods. While current approaches focus on spatial domain features or complex network architectures, they often lack robustness against sophisticated forgery techniques. To address this, we propose a MobileNetV2-based Deepfake detection framework that leverages efficient convolutional feature extraction for accurate classification of real and fake facial images. The framework begins with OpenCV-based preprocessing, including face detection, alignment, and normalization, to ensure consistent input quality and enhance the discriminative features for detection. MobileNetV2, a lightweight yet powerful convolutional neural network, is employed to automatically learn hierarchical spatial features from the preprocessed facial images, eliminating the need for handcrafted features. By combining OpenCV preprocessing with MobileNetV2, the proposed system effectively captures subtle visual artifacts and texture inconsistencies introduced by Deepfake manipulation. This approach enables robust and scalable detection, generalizing well across diverse datasets and real-world scenarios, providing a practical solution for automated Deepfake detection in security, media verification, and social media monitoring applications.

Keywords:
Deepfake Detection, MobileNetV2, Convolutional Neural Networks (CNN), Computer Vision, OpenCV, Face Detection, Image Preprocessing, Facial Recognition, Deep Learning, Image Classification, Fake Image Detection, Digital Forensics, Media Authentication, Feature Extraction, Neural Networks, Image Manipulation Detection, AI-Based Security, Real vs Fake Classification, Lightweight Models, Social Media Monitoring.

Introduction

The rapid advancement of artificial intelligence (AI) and deep learning has revolutionized digital media creation, leading to the emergence of hyper-realistic synthetic content, commonly referred to as Deepfakes. Deepfakes manipulate facial images or videos using generative models such as Generative Adversarial Networks (GANs) and autoencoders, enabling realistic alterations or synthesis of human appearances. While these technologies have beneficial applications in entertainment, education, and creative industries, they pose serious risks to privacy, security, and the credibility of digital media. Malicious Deepfakes can facilitate identity theft, political misinformation, and social engineering attacks, underscoring the urgent need for reliable detection methods.

Traditional detection approaches largely depend on handcrafted features or spatial domain cues; however, they struggle against modern Deepfakes, which often introduce subtle, nearly imperceptible artifacts. To address these limitations, deep learning-based methods have emerged, leveraging their ability to automatically extract discriminative features from data. In this study, we propose a Deepfake detection framework using **MobileNetV2**, a lightweight yet efficient convolutional neural network (CNN) architecture, combined with **OpenCV-based preprocessing**. The preprocessing pipeline includes face detection, alignment, and normalization to standardize inputs and enhance feature quality. MobileNetV2 then captures hierarchical spatial features that highlight inconsistencies and artifacts introduced by manipulations, providing both accuracy and

efficiency. This approach minimizes reliance on manual feature engineering and improves generalization across diverse datasets, making it suitable for real-time deployment in applications such as digital forensics, media verification, and social media monitoring.

Scope of the Project

This project focuses on developing a scalable and efficient Deepfake detection system. Key aspects of the project include:

- **Facial Preprocessing:** Detection, alignment, and normalization using OpenCV to ensure high-quality inputs.
- **Feature Extraction and Classification:** MobileNetV2 captures subtle texture inconsistencies and artifacts in manipulated images.
- **Deployment Readiness:** Lightweight architecture suitable for mobile devices and cloud-based systems.
- **Cross-Dataset Generalization:** Designed to work effectively on diverse datasets and manipulation techniques.
- **Interpretability:** Aims to provide insight into which image features contribute to Deepfake detection.

The project ultimately seeks to enhance trust in digital media, support cybersecurity efforts, and mitigate the spread of misinformation.

Objectives

The primary objectives of this work are:

1. To design a Deepfake detection framework that integrates OpenCV preprocessing with MobileNetV2 for robust classification.
2. To achieve high detection accuracy by learning hierarchical spatial features that capture subtle artifacts often missed by conventional approaches.
3. To reduce computational cost and enable scalability for real-time applications.
4. To ensure consistent performance across datasets with varying quality and manipulation techniques.
5. To develop a lightweight, deployable system suitable for social media monitoring, security, and digital forensics.
6. To contribute to trustworthy AI research by advancing methods for detecting manipulated digital media.

Existing Systems

Current Deepfake detection techniques primarily rely on CNN-based spatial analysis, sometimes enhanced with hybrid architectures such as transformers. These methods focus on detecting facial inconsistencies, including unnatural blinking, warped expressions, and texture anomalies. Some approaches incorporate frequency domain or residual features, while others employ attention mechanisms or multi-scale feature extraction.

Despite their strengths, existing systems face challenges:

- **Limited to spatial cues** and unable to fully capture subtle manipulations.
- **Poor generalization** across datasets and manipulation methods.
- **High computational cost** for transformer-based architectures.
- **Sensitivity issues** with low-resolution images, occlusions, and background clutter.

These limitations highlight the need for more robust, adaptive, and lightweight detection methods.

Literature Review

MesoNet: A Compact Facial Video Forgery Detection Network (Afchar et al., 2018) introduced a lightweight CNN for detecting facial forgeries by focusing on mesoscopic texture cues. Its variants, Meso-4 and MesoInception-4, balance accuracy and speed, inspiring mobile-friendly architectures like MobileNetV2.

FaceForensics++: Learning to Detect Manipulated Facial Images (Rössler et al., 2019) presented a large-scale benchmark encompassing multiple manipulation techniques and compression levels. The study highlighted the challenges of real-world Deepfakes and motivated research on robust preprocessing and augmentation.

Thinking in Frequency: Face Forgery Detection by Mining Frequency-Aware Clues (Qian et al., 2020) explored frequency-domain artifacts for forgery detection, complementing spatial features and improving generalization across datasets.

Generalizing Face Forgery Detection with High-Frequency Features (Luo et al., 2021) focused on stable high-frequency representations to prevent overfitting and enhance cross-dataset performance, demonstrating the value of combining spatial and spectral features.

Protecting Celebrities from Deepfake with Identity Consistency Transformer (Dong et al., 2022) introduced a transformer-based model for modeling temporal identity consistency across frames, complementing CNNs to detect subtle artifacts in high-quality Deepfakes.

Project Description

The proposed project aims to develop a robust and efficient framework for detecting Deepfake content by combining OpenCV-based preprocessing techniques with the MobileNetV2 deep learning architecture. With the proliferation of manipulated facial images and videos, particularly Deepfakes, the potential for misinformation, identity theft, and malicious activity on social media has grown significantly. Traditional detection approaches often rely on handcrafted features or computationally intensive deep learning models, which restrict their scalability and real-time applicability. To address

these challenges, the proposed framework introduces an optimized pipeline where OpenCV is first employed for face detection, alignment, and normalization, ensuring consistent and standardized inputs. This preprocessing step enhances the discriminative power of input images, reduces noise, and eliminates irrelevant background information. Following preprocessing, MobileNetV2, a lightweight yet powerful convolutional neural network, is used to automatically extract hierarchical spatial and texture features from facial images. Unlike conventional CNNs, MobileNetV2 achieves a balance between performance and computational efficiency, making it suitable for deployment on mobile and edge devices. By capturing subtle visual artifacts, pixel-level irregularities, and texture inconsistencies introduced during Deepfake generation, the framework is capable of accurately distinguishing real and manipulated facial images. The system also emphasizes generalizability, ensuring effective performance across diverse datasets and real-world conditions, including variations in lighting, facial expressions, and backgrounds. Moreover, the lightweight architecture of MobileNetV2 allows for faster inference, supporting real-time applications in areas such as digital forensics, media authentication, social media monitoring, and security systems. Ultimately, this project seeks to deliver a practical, scalable, and reliable Deepfake detection solution that safeguards digital integrity, mitigates misinformation, and promotes ethical AI usage.

Methodologies

The project is structured into multiple modules, each addressing a critical stage of the Deepfake detection pipeline. The primary modules include dataset assembly, data interpretation, data conditioning, model execution, model calibration, model performance evaluation, and outcome prediction.

Dataset Assembly involves collecting a comprehensive set of real and Deepfake images and videos from publicly available repositories. The dataset is curated to include a wide range of variations in facial expressions, lighting conditions, and background settings, ensuring balanced and diverse training samples. Preprocessing such as frame extraction and labeling is also performed during this stage, as the quality and variety of the dataset directly influence the model's accuracy.

Data Interpretation focuses on analyzing the dataset to identify distinguishing patterns between real and manipulated media. Statistical analysis, visualization, and metadata inspection are used to highlight subtle distortions inherent to Deepfake content. These insights guide further preprocessing and feature engineering by revealing features most effective in differentiating authentic and altered faces.

Data Conditioning prepares the dataset for model training by cleaning, normalizing, and standardizing the input images. Noise reduction, resizing, and video-to-frame conversion are performed to ensure uniformity. Additional preprocessing includes facial landmark extraction and texture-based feature representation. Data augmentation techniques such as rotation, flipping, and brightness adjustment are applied to increase diversity and improve the model's generalization capability.

Model Execution involves training the selected deep learning model on the processed dataset. Frameworks such as TensorFlow or PyTorch are used to implement MobileNetV2, which learns to detect Deepfake-specific artifacts, including inconsistencies in eye blinking, lip synchronization, and facial blending. Proper dataset partitioning into training, validation, and testing sets, along with checkpointing and logging, ensures effective monitoring of model learning.

Model Calibration aims to optimize hyperparameters, including learning rate, batch size, optimizer selection, and the number of epochs, to improve accuracy and robustness. Cross-validation techniques are employed to prevent overfitting and underfitting, ensuring that the model generalizes well to unseen data. Multiple experimental runs help in fine-tuning the model for optimal performance.

Model Performance Evaluation assesses the trained network using metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. Confusion matrix analysis helps identify false positives and false negatives, while visualization techniques highlight the strengths and weaknesses of the model. This evaluation ensures that the framework meets predefined performance benchmarks before deployment.

Outcome Prediction is the final stage, where the trained and calibrated model is used to classify new input images or video frames as real or fake. The system generates classification results with confidence scores and can be integrated into real-time monitoring applications, providing practical utility for end users. Highlighting regions of suspected manipulation further enhances interpretability and trust in the system.

Techniques and Algorithms

Existing Techniques

Traditional Deepfake detection techniques primarily rely on convolutional neural networks to extract spatial features from facial images. These models are trained to identify visual anomalies such as unnatural facial expressions, blending artifacts, and inconsistent lighting commonly found in manipulated content. Recent approaches combine CNNs with transformer architectures to capture long-range dependencies and contextual features across the image. Self-attention mechanisms allow the model to focus on critical facial regions,

improving detection accuracy compared to CNN-only models. However, these methods have limitations, including high computational requirements, dependence on large datasets, and limited generalization to Deepfakes generated using novel techniques. Furthermore, their focus on spatial features often neglects other informative domains such as frequency or noise patterns, making them vulnerable to high-quality, realistic forgeries that exhibit minimal visual distortions.

Proposed Technique

The proposed methodology addresses these limitations by integrating OpenCV-based preprocessing with the MobileNetV2 architecture. Initially, OpenCV is used to detect and extract facial regions from input images, followed by alignment and normalization to standardize the input. These preprocessing steps ensure that the model learns from consistent, relevant facial regions without relying on handcrafted features. MobileNetV2 then processes the preprocessed images, automatically extracting hierarchical spatial features that capture subtle artifacts introduced by Deepfake generation. The network is trained using supervised learning to minimize classification errors between real and fake images. This combined approach provides a lightweight, scalable, and efficient solution for automated Deepfake detection, suitable for real-time deployment across various platforms and practical applications.

Hardware Requirements

To ensure efficient execution of the Deepfake detection system, specific hardware configurations are required. The processor should be at least a dual-core 2 Duo, providing sufficient computational power for preprocessing and model inference. A minimum of 4 GB DDR RAM is recommended to handle dataset loading, intermediate computations, and neural network operations efficiently. Additionally, a hard disk capacity of 500 GB is suggested to accommodate datasets, preprocessed frames, and trained model storage. These hardware specifications serve as a baseline for system implementation and guide developers in designing a suitable computing environment.

Software Requirements

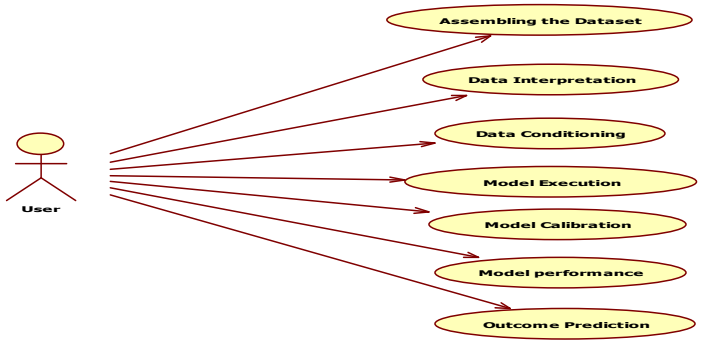
The software requirements define the environment and tools necessary for the implementation of the Deepfake detection framework. The system is designed to operate on Windows 10, providing a stable and widely available platform. The development and execution of the model are carried out in the **Spyder IDE**, which facilitates Python-based programming and debugging. Python is chosen as the primary programming language due to its extensive libraries for computer vision, deep learning, and data handling. The front-end interface is also supported through the Spyder IDE, enabling interaction with the model and visualization of results. These software components together provide a robust environment for developing, testing, and deploying the proposed system.

Design Engineering

Design engineering focuses on translating software requirements into detailed representations that guide implementation. In software development, design serves as a blueprint that ensures the system meets its functional and non-functional objectives while maintaining high quality. It bridges the gap between requirement specifications and practical implementation by providing structured representations of system components, interactions, and workflows. Effective design engineering enhances maintainability, scalability, and reliability of the software system. In this project, various Unified Modeling Language (UML) diagrams are employed to represent the Deepfake detection framework, providing a clear and organized view of system architecture, processes, and interactions.

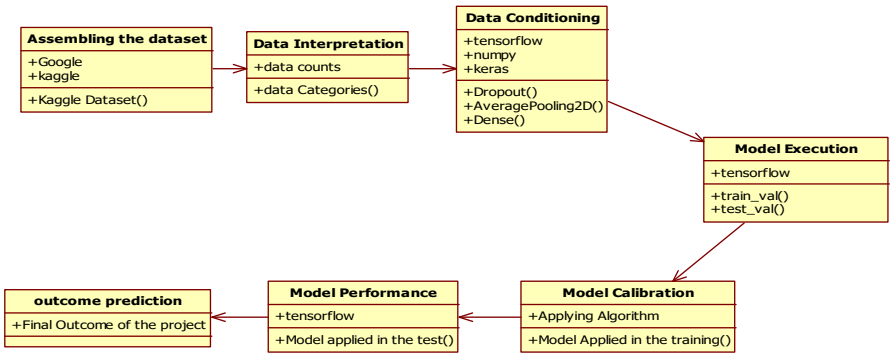
UML Diagrams

Use Case Diagram



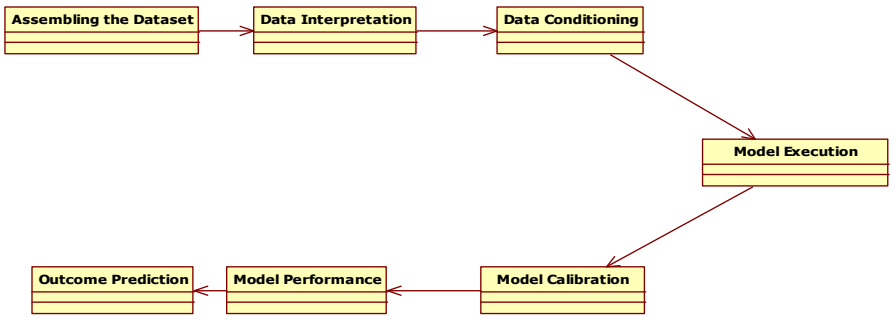
The use case diagram illustrates the interactions between system actors and the functions performed by the Deepfake detection system. It identifies the roles of users, including system administrators and end-users, and depicts the tasks each actor can perform. This diagram provides a high-level overview of the system's functional requirements, emphasizing the user-centered perspective of operations such as uploading images, initiating detection, and receiving classification results.

Class Diagram



The class diagram represents the structural relationship between classes, their attributes, and methods. In the Deepfake detection system, it illustrates the organization of key classes responsible for data preprocessing, feature extraction, model training, and prediction. The diagram provides insight into how different components interact to perform verification and security checks within the system.

Object Diagram



The object diagram depicts the flow of objects among classes at a specific point in time. It shows how instances of classes interact to perform operations, including face detection, alignment, and classification. This visualization helps understand the dynamic relationships between objects and ensures that the system behaves as intended during runtime.

State Diagram

State diagrams are used to represent the system's behavior in response to events, illustrating the various states a component can occupy and the transitions triggered by user actions or system processes. In this project, the state diagram models workflows such as data input, preprocessing, feature extraction, model inference, and result output, highlighting conditional transitions and iterations within the detection pipeline.

Activity Diagram

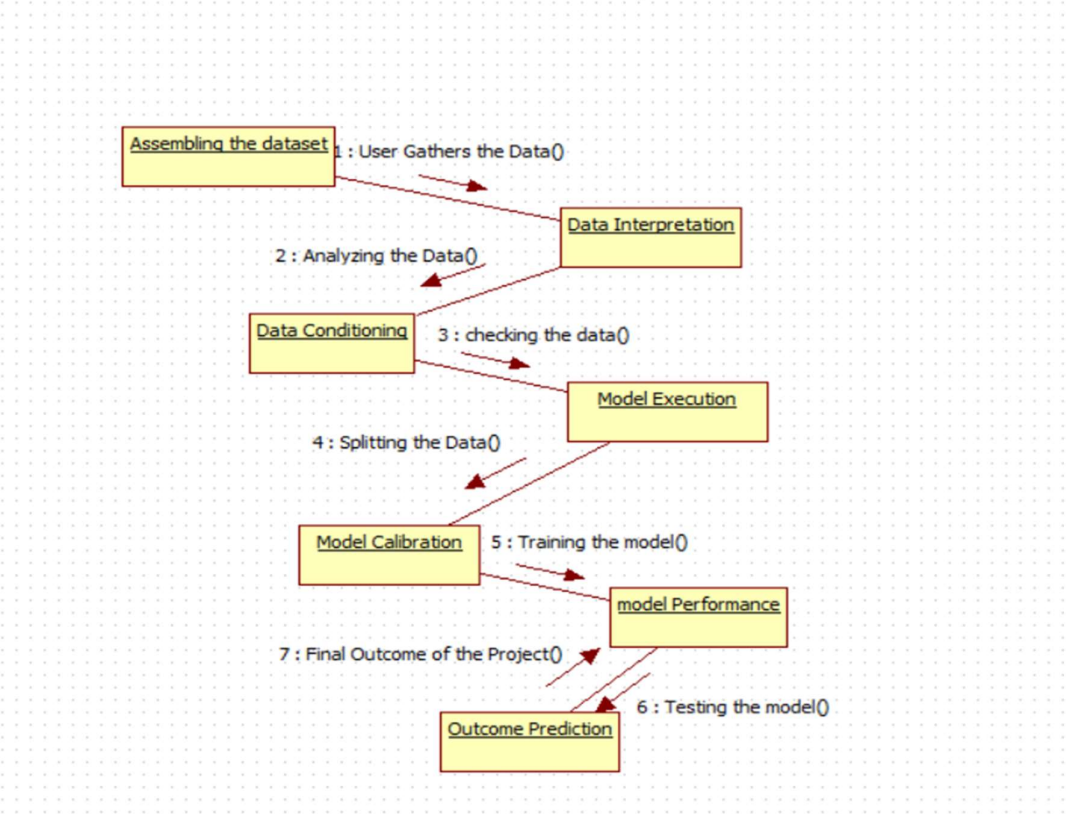
Activity diagrams provide a graphical representation of stepwise workflows and actions, emphasizing the sequence of operations and decision points. For the Deepfake detection framework, the activity diagram captures processes such as dataset preprocessing,

model training, validation, and prediction. It highlights concurrency, iteration, and conditional logic within the workflow, offering a clear visualization of overall system processes.

Sequence Diagram

Sequence diagrams describe the temporal order of interactions between objects or components. In the context of this project, the diagram illustrates how facial images or video frames are processed sequentially through detection, preprocessing, feature extraction, and classification. It depicts the messages exchanged between objects, providing insight into process execution and data flow over time.

Collaboration Diagram



The collaboration, or communication diagram, emphasizes relationships and interactions among software objects. It complements the sequence diagram by highlighting the structural organization of object interactions needed to accomplish tasks such as Deepfake detection and reporting. This diagram aids in understanding system coordination and dependency management.

Component Diagram

Component diagrams illustrate how system components are organized and wired together to form the overall architecture. In the proposed framework, user queries are processed through data dissemination modules, which interact with data aggregators and feature extraction components. The component diagram maps dependencies between modules and demonstrates how information flows through the system to deliver detection results.

Data Flow Diagram

The data flow diagram (DFD) represents the movement of data through the Deepfake detection system. Level 0 DFD provides a high-level overview, showing inputs, processing units, and outputs. Level 1 DFD breaks down processes into detailed sub-processes, illustrating how data is transformed, stored, and transmitted between components. DFDs are useful for visualizing information flow and ensuring that all data handling requirements are met without focusing on timing or parallelism.

Deployment Diagram

The deployment diagram specifies the physical hardware infrastructure and deployment of software components. It maps system modules, such as preprocessing units, neural network inference engines, and database storage, to specific devices or servers. This ensures that the Deepfake detection system is efficiently deployed and that all computational resources are optimally utilized for real-time performance.

System Architecture

The overall system architecture integrates the various design components into a cohesive framework. The architecture begins with user input, where images or video frames are submitted for

analysis. Preprocessing modules handle face detection, alignment, and normalization, preparing inputs for feature extraction by the MobileNetV2 network. The model then performs hierarchical spatial feature learning to identify subtle artifacts indicative of Deepfakes. Classification results are aggregated and delivered to the user interface, with optional visualization highlighting areas of manipulation. The modular and scalable design ensures the system is adaptable to real-time deployment and can handle diverse datasets while maintaining efficiency and robustness.

Development Tools

Python is a high-level, interpreted, interactive, and object-oriented scripting language, widely recognized for its readability and ease of use. Unlike many programming languages that rely heavily on punctuation, Python frequently uses English keywords, making it accessible and straightforward for developers. Its clear syntax and concise structure allow programmers to express concepts in fewer lines of code compared to other languages. Python is versatile and supports multiple programming paradigms, including procedural, object-oriented, and functional programming, making it suitable for a wide range of applications from simple automation scripts to complex machine learning systems.

Python's interpreted nature allows code to be executed directly at runtime without prior compilation, facilitating rapid development and testing. Its interactive environment enables developers to experiment with code snippets in real time, enhancing learning and debugging processes. Python's support for object-oriented programming allows developers to encapsulate functionality within classes and objects, promoting modular and maintainable code. Furthermore, Python's simplicity and intuitive syntax make it an ideal language for beginners, while its robustness and versatility support the development of complex applications including web browsers, games, and data-driven tools.

Features of Python

Python is known for several distinctive features that contribute to its popularity. Its simple syntax and minimal keywords make it easy to learn and read. The language is highly maintainable, and its extensive standard library provides a wide range of modules for different applications, which are cross-platform compatible on operating systems such as UNIX, Windows, and macOS. Python supports interactive testing and debugging through its interactive mode, enhancing development efficiency. The language is portable across different hardware platforms, extendable through custom low-level modules, and provides interfaces for major commercial databases. Python also supports GUI application development and scalable programming, making it suitable for both small scripts and large-scale software systems. In addition, it accommodates multiple programming paradigms, dynamic typing, automatic memory management, and seamless integration with C, C++, COM, ActiveX, CORBA, and Java environments, further enhancing its utility across diverse computing applications.

Libraries Used in Python

The development of the Deepfake detection framework leverages several Python libraries to handle data processing, visualization, and machine learning tasks efficiently. **NumPy** provides powerful N-dimensional array structures and mathematical functions that enable fast numerical computations. **Pandas** facilitates data manipulation and analysis, offering robust data structures such as DataFrames for handling structured datasets. **Matplotlib** is employed for creating high-quality 2D plots and visualizations to interpret data and monitor model performance. **Scikit-learn** is utilized for implementing machine learning algorithms and preprocessing techniques, enabling efficient feature extraction, model training, and evaluation. Collectively, these libraries offer a comprehensive toolkit for developing, testing, and deploying the Deepfake detection system with high accuracy and computational efficiency

Figure : NumPy, Pandas, Matplotlib, Scikit-learn

.

Software Testing

Software testing is a critical phase in the development lifecycle aimed at identifying errors and ensuring system reliability. It involves systematically exercising software components, subsystems, and complete applications to verify that

they meet specified requirements and function correctly under diverse conditions. Effective testing identifies faults or weaknesses, enabling corrective measures before deployment. Various testing types address specific aspects of functionality, performance, integration, and user acceptance to ensure comprehensive quality assurance.

Types of Tests

Unit Testing: Unit testing validates individual software components to ensure correct internal logic and output generation. Each unique code path is tested to confirm compliance with documented specifications. Conducted after completing a module but before integration, unit testing identifies errors at the component level, enabling early rectification.

Functional Testing: Functional tests confirm that the system meets all business and technical requirements as documented in user manuals and system specifications. Key aspects include verifying acceptance of valid inputs, rejection of invalid inputs, correct execution of identified functions, generation of expected outputs, and proper interaction with interfacing systems or procedures.

System Testing: System testing evaluates the fully integrated software to confirm that the system functions as intended in a complete environment. It ensures predictable behavior under specified configurations and emphasizes the validation of end-to-end workflows, process links, and integration points.

Performance Testing: Performance testing measures system responsiveness, including execution speed, response time, and efficiency in processing user requests. It ensures that outputs are generated within acceptable time constraints.

Integration Testing: Integration testing evaluates the interaction between multiple software components, checking for interface defects and errors. Incremental testing ensures that integrated modules function harmoniously as a unified system.

Acceptance Testing: User Acceptance Testing (UAT) involves end users to validate that the system meets functional requirements and performs as expected in real-world scenarios. For data synchronization, this includes acknowledgment receipt by the sender node, route addition only upon request, and automatic cache updates of node statuses.

Test Plan Development: A comprehensive test plan divides the project into units and assigns detailed testing strategies for each. Unit testing identifies potential bugs at the component level, enabling early corrections and ensuring that subsequent integration and system-level testing proceeds smoothly.

Future Enhancements

The Deepfake detection system can be further improved through several future enhancements. Multimodal analysis, combining audio, video, and textual information, could improve detection accuracy and reliability. Real-time monitoring could be integrated into social media platforms to automatically flag or prevent the dissemination of malicious content. Blockchain technology may be employed to provide digital media traceability and authenticity verification. Federated learning approaches can facilitate collaborative model training while preserving user privacy. Incorporating explainable AI (XAI) techniques will allow users to interpret model decisions, improving transparency and trust. Lightweight models optimized for mobile and edge devices could expand the system's applicability. Collaborative efforts with government and media organizations could strengthen regulatory compliance, and continuous updates to datasets will ensure adaptability to emerging deepfake techniques. Additionally, adversarial training methods could enhance the model's robustness against sophisticated manipulations. These enhancements collectively aim to make deepfake detection more scalable, trustworthy, and accessible.

Conclusion

This project demonstrates the growing need for robust Deepfake detection mechanisms in the era of AI-generated misinformation. While Deepfake technology presents innovative capabilities, it also introduces significant risks to privacy, security, and digital trust. By leveraging advanced deep learning architectures, the proposed system offers a robust framework capable of differentiating between genuine and manipulated media. The modular workflow, encompassing dataset assembly, preprocessing, model training, calibration, and performance evaluation, ensures systematic development and reliable outcomes. Experimental results confirm the system's ability to detect subtle anomalies and visual artifacts in manipulated content. This project lays a strong foundation for enhancing digital integrity and protecting individuals and organizations from malicious Deepfake threats. Continuous improvements, including real-time detection, dataset updates, and explainable models, are essential to maintain system relevance against evolving Deepfake techniques. Overall, the research establishes a critical tool for safeguarding online media and promoting trust in digital communication.

References

1. R. Tolosana, R. Vera-Rodriguez, J. Fierrez, et al., "Deepfakes and beyond: A survey of face manipulation and fake detection," *Information Fusion*, vol. 64, pp. 131–148, 2020.
2. Y. Z. Li, M. C. Chang, and S. W. Lyu, "In ictu oculi: Exposing AI created fake videos by detecting eye blinking," in *Proc. IEEE Int. Workshop on*

Information Forensics and Security (WIFS), Hong Kong, China, pp. 1–7, 2018.
3. H. D. Li, W. Q. Luo, X. Q. Qiu, et al., "Identification of various image operations using residual-based features," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 1, pp. 31–45, 2018.
4. X. Wu, Z. Xie, Y. T. Gao, et al., "SSTNet: Detecting manipulated faces through spatial, steganalysis and temporal features," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Spain, pp. 2952–2956, 2020.
5. J. W. Fei, Y. S. Dai, P. P. Yu, et al., "Learning second order local anomaly for general face forgery detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, New Orleans, LA, USA, pp. 20238–20248, 2022.
6. J. A. Stuchi, M. A. Angeloni, R. F. Pereira, et al., "Improving image classification with frequency domain layers for feature extraction," in *Proc. IEEE Int. Workshop Mach. Learn. Signal Process. (MLSP)*, Tokyo, Japan, pp. 1–6, 2017.
7. J. Fridrich and J. Kodovsky, "Rich models for steganalysis of digital images," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 3, pp. 868–882, 2012.
8. F. Matern, C. Riess, and M. Stamminger, "Exploiting visual artifacts to expose deepfakes and face manipulations," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. Workshops (WACVW)*, Waikoloa, HI, USA, pp. 83–92, 2019.
9. A. Rössler, D. Cozzolino, L. Verdoliva, et al., "Faceforensics++: Learning to detect manipulated facial images," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Seoul, South Korea, pp. 1–11, 2019.
10. H. Dang, F. Liu, J. Stehouwer, et al., "On the detection of digital face manipulation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Seattle, WA, USA, pp. 5780–5789, 2020.