www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 10, October 2025, pp : 754-763

e-ISSN : 2583-1062
Impact Factor : 7.001

# VISION GUARD: DEEPFAKE VIDEO DETECTION SYSTEM

**Pravin Kumar S[1], Subanesh B[2], Sahaya Selva Ritesh A[3], Yamini B[4]**

[1,2,3]Student, Department Of CSE AI & DS, Vels Institute Of Science Technology And Advanced Studies, Chennai, India.

[4]Assistant Professor, Department Of CSE AI & DS, Vels Institute Of Science Technology And Advanced Studies, Chennai, India.

E-Mail: subanesh3033@gmail.com

DOI: https://www.doi.org/10.58257/IJPREMS44271

## ABSTRACT

The proliferation of manipulated and AI- generated videos, commonly known as deepfakes, has created serious challenges in maintaining digital trust, authenticity, and security. These highly convincing falsified visuals can be exploited for misinformation, fraud, or defamation, making it crucial to develop automated systems capable of identifying such manipulated media. Vision Guard is an AI-powered system designed to detect and classify fake or tampered videos using advanced machine learning and deep learning algorithms. The system leverages convolutional neural networks (CNNs) to analyze spatial and temporal inconsistencies in facial expressions, textures, and motion patterns that typically distinguish real content from fake. A comprehensive dataset containing both authentic and manipulated videos is used to train the model, ensuring reliable detection across diverse environments. Once a video is uploaded, the system extracts key frames, performs feature analysis, and classifies the content as "Real" or "Fake." A Flask- based web application serves as the interface for real- time analysis and result visualization, allowing users to verify the authenticity of media effortlessly. The system can be integrated with social media platforms, digital forensic tools, and content moderation frameworks to ensure trust worthy visual communication. By providing an automated, accurate, and scalable detection mechanism, Vision Guard contributes significantly to combating the global spread of misinformation and enhancing digital media integrity.

**Keywords**: Vision Guard, Deepfake Detection, Machine Learning, Flask Application, Video, Forensics, Real-time.

## 1. INTRODUCTION

In today's rapidly evolving digital ecosystem, multimedia content has become one of the most powerful and influential forms of communication. The widespread availability of image and video editing tools, along with the advancement of deep learning algorithms, has given rise to highly realistic manipulated videos, commonly known as deep fakes. Deepfakes leverage sophisticated artificial intelligence techniques, particularly generative adversarial networks (GANs), to produce videos in which individuals appear to say or do things they never actually did. While these technologies were originally developed for entertainment and research purposes, their misuse has led to a surge in misinformation, identity theft, and social manipulation. This has created an urgent need for automated systems capable of detecting and preventing the spread of such maliciously altered visual content.

This project focuses on developing an intelligent system capable of automatically detecting manipulated videos using advanced visual analysis and machine learning techniques. Unlike traditional media verification methods that rely on manual inspection or metadata analysis, the proposed system examines the intrinsic characteristics of the video itself, including facial patterns, texture distortions, and motion inconsistencies. For example, while human eyes and expressions follow natural synchronization in genuine recordings, deep fakes often display subtle irregularities in blinking, lip movement, and lighting reflections that can be algorithmic ally identified.

Vision Guard is proposed as an advanced AI-based solution to address this challenge. The system is designed to analyze visual features and frame sequences to detect inconsistencies that reveal potential video tampering. By leveraging convolutional neural networks (CNNs) and deep learning architectures, Vision Guard is capable of identifying subtle discrepancies in facial expressions, lighting, and motion continuity that typically distinguish fake videos from authentic ones. The system processes uploaded videos in real time and classifies them as either Real or Fake with a high degree of accuracy. The outcome is presented through a user-friendly web interface developed using Flask, allowing users to upload, process, and view authenticity results instantly.

The project focuses not only on the detection accuracy but also on the practicality and scalability of the system. Vision Guard is designed to operate efficiently on standard computing devices without requiring high-end hardware. Through frame extraction, normalization, and feature optimization, the system ensures low latency and high responsiveness. Moreover, the platform architecture allows for easy integration into existing digital ecosystems such as social media

applications, content verification systems, and forensic tools. This flexibility makes Vision Guard suitable for diverse real-world scenarios, including journalism, cyber forensics, and public safety.

From a technical standpoint, the Vision Guard framework integrates multiple stages of video analysis. The process begins with preprocessing, where the uploaded video is decomposed into frames, and irrelevant data such as background noise or static segments are filtered out. These frames are then passed through feature extraction layers of the CNN, which capture key visual and temporal patterns. The model is trained on a large dataset of real and fake videos to learn the complex visual signatures associated with deepfake generation techniques. By using supervised learning, the model continuously improves its ability to classify unseen samples with higher precision. The decision layer outputs a binary classification, indicating whether the analyzed video is authentic or manipulated.The design philosophy of Vision Guard emphasizes both accuracy and ethical deployment.

Traditional media verification techniques rely on manual inspection, watermark validation, or metadata analysis. However, these methods are neither scalable nor reliable in identifying subtle manipulations that occur at the pixel or frame level. Deepfakes, in particular, are designed to be indistinguishable to the human eye, making manual detection almost impossible. The absence of automated verification tools has made social media platforms, news agencies, and individuals highly vulnerable to digital deception. Therefore, the development of an intelligent detection system that can automatically differentiate between real and fake videos is crucial for ensuring the authenticity of online media.

One of the core objectives of Vision Guard is to enhance digital trust by preventing the circulation of deceptive videos. In social media environments where content spreads rapidly, deepfakes can be used for propaganda, blackmail, or defamation, leading to significant personal and societal harm. By providing an automated mechanism for detecting such content, Vision Guard plays a key role in mitigating these risks. The system can serve as a first line of defense against visual misinformation, alerting users and administrators before manipulated videos gain traction. within the system environment, without external data transmission. This guarantees that users' personal media remains secure throughout the detection process. Additionally, the model is optimized to minimize false positives and negatives, which are critical factors for maintaining credibility and reliability in real-world use cases.

The innovation of Vision Guard lies in combining advanced deep learning models with practical usability through an accessible web platform. Users can interact with the system seamlessly, upload videos for verification, and obtain results supported by visual evidence. The inclusion of modules such as login authentication, gallery management, and result history further enhances the professional scope of the system. Through these integrated components, Vision Guard not only performs deepfake detection but also demonstrates the potential of AI-driven tools to promote responsible technology usage.

## 2. LITERATURE REVIEW

The rapid advancement of artificial intelligence and deep learning technologies has significantly influenced the field of multimedia analysis and content verification. In recent years, researchers have developed numerous methods to detect manipulated audio, images, and videos, with a particular focus on deepfake detection. Deepfakes utilize advanced neural network architectures such as Generative Adversarial Networks (GANs) and autoencoders to synthesize highly realistic facial expressions and motions, often making them indistinguishable from real footage to the human eye. Consequently, the need for automated detection systems capable of identifying subtle artifacts and inconsistencies in these manipulated videos has become an important area of research.

Early research in media forensics focused primarily on detecting inconsistencies in metadata or compression patterns. For instance, traditional forensic methods relied on identifying discrepancies in image noise levels, color tone variations, and encoding parameters to detect tampering. However, these techniques proved inadequate against modern deepfakes that maintain high visual quality even after multiple compression stages. As generative models evolved, researchers began exploring data-driven approaches that utilize machine learning for classification tasks.

One of the earliest deepfake detection methods involved the use of hand-crafted features and shallow machine learning models.Studies demonstrated that features such as eye blinking rate, lip synchronization, and head pose estimation could serve as indicators of video manipulation. Li et al. (2018) introduced a recurrent convolutional architecture that analyzed eye-blink frequency to detectsynthetic face videos, as most early deepfake algorithms failed to replicate natural blinking behavior. Similarly, Afchar et al. (2018) proposed MesoNet, a convolutional neural network specifically designed for deepfake detection using mesoscopic-level image features. These early approaches paved the way for applying convolutional neural networks (CNNs) to identify facial inconsistencies in generated videos. These studies highlighted the significance of usingdeep representations to capture subtle visual distortions and temporal inconsistencies caused by generative models.

As the field matured, researchers began leveraging deep learning architectures for higher accuracy and generalization. Rossler et al. (2019) developed the FaceForensics++ dataset, a large-scale benchmark that enabled researchers to train and test deepfake detection models under controlled conditions. They utilized XceptionNet, a deep CNN model, achieving superior performance in distinguishing authentic and fake content. Following this, Guarnera et al. (2020) explored the use of texture- based descriptors and frequency-domain features to detect manipulation artifacts invisible to the human eye.

Several hybrid models were also developed to improve detection robustness across diverse datasets. For example, Sabir et al. (2019) proposed a recurrent CNN framework that combines convolutional layers for spatial analysis with recurrent layers for temporal coherence detection. Their model effectively captured temporal dependencies between consecutive frames, reducing false positives caused by single-frame anomalies. Similarly, Yang et al. (2020) integrated attention mechanisms within CNN architectures to focus on discriminative facial regions, improving detection accuracy even under low-resolution and high-compression conditions.

In addition to model-centric advancements, numerous studies have emphasized the importance of diverse datasets and preprocessing techniques. Works such as Dolhansky et al. (2020) introduced the DeepFake Detection Challenge (DFDC) dataset, which includes a wide range of subjects, lighting conditions, and manipulation methods. These datasets allowed researchers to evaluate detection systems under realistic conditions. Preprocessing steps like face alignment, normalization, and region-of-interest extraction were also found to significantly influence model performance by focusing computational resources on relevant facial areas.

Despite these advances, several challenges persist in achieving reliable deepfake detection in real- world environments. One of the primary challenges lies in generalization — models trained on a specific dataset or manipulation technique often fail to perform effectively on unseen data. Furthermore, as generative models evolve, the quality of fake videos improves, making existing detection algorithms less effective. Another key issue is computational efficiency.

Many state-of-the-art detection systems require high-end GPUs and significant processing power, limiting their deployment on mobile or low- resource devices. Finally, ethical considerations, such as data privacy and consent, continue to be critical concerns when training on real human facial data.

To address these limitations, recent research has focused on combining spatial and temporal feature extraction to improve model generalization. Multi- stream architectures that analyze both visual frames and motion vectors have shown promising results in differentiating between authentic and manipulated content. Additionally, transfer learning techniques have been utilized to fine-tune pre-trained models on new datasets, reducing the dependency on large annotated datasets. Lightweight CNN models and quantization methods have also been introduced to make detection feasible on edge devices without compromising accuracy.

Building upon insights from prior research, Vision Guard introduces several innovations to enhance the performance and practicality of deepfake detection. Unlike traditional systems that rely solely on static facial features, Vision Guard integrates both spatial and temporal cues, analyzing frame sequences to detect unnatural transitions, motion mismatches, and visual inconsistencies. The system employs convolutional neural networks for frame-level analysis, supported by preprocessing modules that enhance clarity and normalize illumination variations. The inclusion of real and synthetic video datasets ensures diversity and robustness in training. Moreover, Vision Guard is implemented as a Flask-based web application, providing an accessible platform for real-time video analysis and classification.

By combining advanced feature extraction, efficient model design, and user-centric deployment, Vision Guard overcomes the limitations observed in earlier research. Its hybrid approach enables accurate detection even in compressed or noisy video samples, while the web-based interface allows seamless integration with various digital platforms. Thus, the project contributes to the growing field of media authenticity verification, bridging the gap between academic research and practical implementation.

## 3. METHODOLOGY

The methodology adopted for the development of Vision Guard follows a structured and modular approach to ensure accuracy, scalability, and real-time performance. The system integrates machine-learning-based classification with a Flask-based deployment framework, enabling users to upload video samples and obtain instant authenticity results. The workflow consists of five major phases: Dataset Collection, Preprocessing and Feature Extraction, Model Training and Evaluation, Web Deployment, and Result Generation with Alert Mechanism. Each phase is carefully designed to handle specific tasks while maintaining efficiency and consistency across the pipeline.

www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 10, October 2025, pp : 754-763

e-ISSN : 2583-1062
Impact Factor : 7.001

### 3.1 System Overview

The Vision Guard architecture is organized as a multistage pipeline that converts raw input videos into analyzed and classified output. It begins with collecting a balanced dataset of real and fake videos, followed by preprocessing steps such as frame extraction, normalization, and noise reduction. Feature extraction using convolutional neural networks (CNNs) captures spatial and temporal information. The extracted features are then passed through the classification layer, which predicts whether the content is authentic or manipulated. Finally, the Flask interface displays the result and stores the report in the database for reference.

The overall design emphasizes modularity, allowing each component to be developed, tested, and updated independently. This modular structure facilitates easy integration with other digital-forensics platforms and ensures that the system can evolve as deepfake-generation techniques become more advanced.

### 3.2 Dataset Collection

The foundation of any supervised-learning model is a diverse and well-labeled dataset. For Vision Guard, the dataset includes both real and fake video samples obtained from public deepfake repositories such as FaceForensics++, Celeb-DF, and custom-collected recordings.

Each video is labeled according to its authenticity.

The real category contains unaltered recordings with natural facial expressions, lighting conditions, and motion, while the fake category consists of manipulated videos produced using GAN-based synthesis techniques.

To ensure diversity, the dataset includes subjects of different age groups, ethnicities, and genders under varying illumination, camera quality, and pose conditions. A balanced distribution between the two classes helps the model learn representative patterns and prevents bias toward a specific category. Data augmentation techniques, including horizontal flipping, brightness variation, and Gaussian noise injection, are applied to enhance dataset variability and improve model generalization.

### 3.3 Preprocessing and Feature Extraction

Before model training, each video undergoes preprocessing to standardize its format and reduce computational complexity. The preprocessing phase includes:

1. **Frame Extraction:** Each video is decomposed into frames at a constant frame-rate (e.g., 30 fps). Extracted frames are stored as images for independent analysis.
2. **Face Detection and Cropping:** Using libraries such as MTCNN or OpenCV, the face region is localized and cropped from each frame to remove background noise.
3. **Normalization:** Pixel values are scaled between 0 and 1 to accelerate convergence during training.
4. **Resizing:** All images are resized to 224 × 224 pixels to maintain uniformity across the dataset.
5. **Noise Reduction and Color Correction:** Filters are applied to eliminate visual artifacts and enhance feature clarity.
6. Following preprocessing, **feature extraction** is carried out using a convolutional neural network.
7. The CNN learns to identify discriminative characteristics such as facial asymmetry, blending boundaries, eye-blink irregularities, and texture inconsistencies. These features are more informative than raw pixel values.

### 3.4 Model Training and Evaluation

The extracted features are used to train the classification model responsible for detecting fake content. Vision Guard utilizes a Convolutional Neural Network (CNN) architecture comprising convolutional, pooling, and fully connected layers. The CNN captures both low-level spatial features and high-level semantic patterns.

1. **Input Layer:** Accepts preprocessed frames of size 224×224×3.
2. **Convolutional Blocks:** Extract hierarchical spatial patterns using 3×3 kernels followed by ReLU activation.
3. **Pooling Layers:** Downsample feature maps to reduce dimensionality while retaining essential information.
4. **Dropout Layers:** Prevent overfitting by randomly disabling neurons during training.
5. **Fully Connected Layer:** Flattens the features into a one-dimensional vector.
6. **Output Layer:** Uses a sigmoid or softmax function to classify frames as Real (1) or Fake (0).

The model is trained using the binary **cross-** entropy loss function and optimized with the Adam optimizer for efficient gradient convergence. The dataset is split into training (70 %), validation (20 %), and testing (10 %) subsets. Evaluation metrics include accuracy, precision, recall, F1-score**,** and confusion matrix analysis. During experimentation, early stopping and learning-rate scheduling are employed to avoid overfitting and improve stability.

www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 10, October 2025, pp : 754-763

e-ISSN : 2583-1062
Impact Factor : 7.001

To further enhance reliability, ensemble techniques combining CNN outputs with Support Vector Machine (SVM) classifiers are explored. This hybrid configuration leverages the deep CNN's ability to extract complex features and the SVM's robustness in decision-boundary formation, leading to improved classification accuracy on challenging video samples.

### 3.5 Web Deployment and Integration

Once trained and validated, the model is integrated into a web application using the Flask framework. Flask provides a lightweight yet powerful environment for deploying machine- learning models as interactive applications.

1. **Frontend Interface:** Developed using HTML, CSS, and JavaScript, it allows users to log in, upload videos, and view analysis results.
2. **Backend Server:** Handles model loading, frame extraction, and prediction tasks using Python libraries such as TensorFlow, OpenCV, and NumPy
3. **Database and Storage:** Stores user information, uploaded videos, and classification logs using SQLite.

When a user uploads a video, the backend automatically extracts frames, processes them through the trained CNN model, and aggregates the predictions. The system computes an overall authenticity score by averaging frame-wise probabilities. If the score exceeds a predefined threshold (e.g., 0.6), the video is classified as Real; otherwise, it is labeled Fake.

To improve user experience, Flask renders results dynamically on the result page, displaying the classification label, confidence percentage, and time of analysis. The gallery page maintains a history of previously analyzed videos, while secure login ensures restricted access to authorized users only.

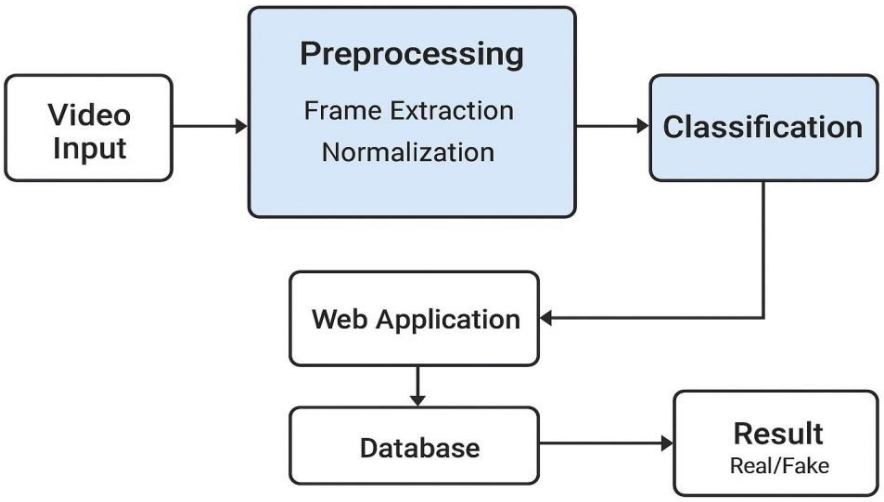
### 3.6 Result Generation and Alert Mechanism

After classification, Vision Guard generates a detailed report summarizing the outcome. The report includes file name, upload time, prediction label, and confidence level. In advanced configurations, the system can automatically trigger alerts or send email notifications when fake videos are detected in sensitive datasets.

## 4. IMPLEMENTATION

The implementation of Vision Guard is carried out through a structured and systematic approach to ensure the seamless integration of machine learning models with a user-friendly web interface. This section explains the key phases of implementation, including the setup of the working environment, data handling, model development, user interface creation, and real-time testing.

Each component of the implementation is designed to ensure optimal accuracy, stability, and scalability, thereby making the system effective in real-world scenarios.




### 4.1 System Setup and Environment Configuration

The development of Vision Guard was performed using the Python programming language due to its extensive support for machine learning and image processing libraries. The model and web application were implemented in the Flask framework, which allows the seamless integration of backend logic and frontend design.

1. **Programming Language:** Python 3.9
2. **Framework:** Flask (for web integration)
3. **Libraries:** TensorFlow, Keras, NumPy, OpenCV, Matplotlib, Scikit-learn

4. **Database:** SQLite
5. **Development Tools:** Visual Studio Code and Jupyter Notebook
6. **Operating System:** Windows 10
7. **Hardware Specifications:** Intel i7 Processor, 16 GB RAM, NVIDIA GPU for model training

A virtual environment was created using venv to manage dependencies and ensure project reproducibility. The folder structure was organized to include directories for model scripts, datasets, templates, static assets, and uploaded files. This modular organization simplified debugging and deployment.

### 4.2 Dataset Preprocessing and Frame Generation

The first implementation step involved preparing the dataset. The video dataset containing both real and fake samples was imported into the working environment.

Each video was split into frames using OpenCV's VideoCapture() function. The extracted frames were then resized to a consistent resolution of **224 × 224 pixels** and normalized.Each processed frame was labeled and stored in separate folders — /real/ and /fake/ — ensuring a balanced dataset for model training. Noise reduction filters and image enhancements were applied to improve clarity. Additionally, the dataset was split into training, validation, and test sets in a 70:20:10 ratio to ensure unbiased model evaluation.

### 4.3 Model Development

The deep learning model forms the heart of Vision Guard. The system employs a Convolutional Neural Network (CNN) architecture due to its proven ability to analyze spatical and temporal video patterns. The model was developed and trained using the TensorFlow and Keras libraries. The CNN architecture used includes the following components:

1. **Input Layer:** Accepts RGB image frames.
2. **Convolutional Layers:** Extract low- and high- level spatial features such as edges, color transitions, and textures.
3. **Activation Function:** ReLU (Rectified Linear Unit) introduces non-linearity.
4. **Pooling Layers:** Perform dimensionality reduction and retain key information.
5. **Dropout Layers:** Prevent overfitting during training.
6. **Fully Connected Layers:** Aggregate features for final decision-making.
7. **Output Layer:** Uses a Softmax activation to classify frames as Real or Fake.

The model is compiled using binary **cross- entropy loss** and optimized with Adam **optimizer**. The batch size and learning rate were tuned through experimentation for better convergence. During training, model checkpoints were used to save the best-performing weights automatically.

Once the model achieved satisfactory accuracy on validation data, it was exported as a .h5 file (e.g., model.h5) for integration into the Flask application.

### 4.4 Flask Web Application Development

Once the deep learning model was finalized, the next stage focused on developing the Flask-based web interface to allow real-time video analysis. Flask was chosen for its simplicity, scalability, and compatibility with machine learning models. The application comprises the following key modules:

1. **Login Page:** Allows users to sign in securely before accessing the upload feature.
2. **Upload Page:** Enables users to browse and upload video files for authenticity verification.
3. **Result Page:** Displays classification results with confidence scores (Real or Fake).
4. **Gallery Page:** Stores the history of previous uploads and results for reference.

The Flask backend script (app.py) handles routing, file upload management, and model inference. Uploaded videos are stored temporarily in the uploads/ folder, where the backend extracts frames using OpenCV. Each frame is processed through the CNN model, and the final classification result is determined by aggregating frame-level predictions. The output is then sent back to the frontend as a formatted result page.

Static assets like CSS and JavaScript were included to improve the visual appearance and responsiveness of the interface. The entire web app is lightweight and can be hosted on both local and cloud servers.

### 4.5 Database Integration

The **SQLite database** is used to maintain a record of user accounts, uploaded videos, and classification results. The database schema includes three main tables:

www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 10, October 2025, pp : 754-763

e-ISSN : 2583-1062
Impact Factor : 7.001

1. users – stores login credentials.
2. uploads – tracks file names, timestamps, and file paths.
3. Results – logs classification output and confidence percentage.

This integration ensures persistent storage and enables administrators to review previous analyses for auditing or system improvement. SQLAlchemy ORM (Object Relational Mapper) was used for database operations, providing a smooth interface between Python objects and database tables.

### 4.6 Testing and Result Visualization

To verify functionality, the system was tested using both genuine and manipulated videos. Each video was analyzed for frame-level inconsistencies. When a fake video was detected, the system displayed the result page indicating "Fake" with a red marker, whereas authentic videos were labeled "Real" in green.

1. **Login Page:** The initial access page that verifies user credentials.
2. **Upload Page:** Where users submit videos for authenticity checking.
3. **Result Page – Real:** Displays a confirmation for authentic videos.
4. **Result Page – Fake:** Indicates manipulated or AI - generated content.
5. **Gallery Page:** Lists previously analyzed videos and their results.

These visual components were developed using HTML and CSS templates under Flask's templates/ directory, ensuring a professional layout consistent with modern web standards.

The results were analyzed both quantitatively— through performance metrics such as accuracy, precision, and recall—and qualitatively—by examining visual outputs and user interactions. This section discusses the obtained results, user interface functionality, and the system's comparative effectiveness.

### 4.7 Optimization and Enhancements Performance

1. **Batch Processing:** Videos are processed frame by frame in batches to reduce memory usage.
2. **GPU Utilization:** TensorFlow GPU support accelerates training and inference speed.
3. **Threaded Processing:** Flask's asynchronous threading reduces request delays.
4. **Threshold Calibration:** Adaptive thresholds minimize false positives in noisy videos.
5. **Caching:** Frequently accessed models and resources are cached to reduce load time.

## 5. RESULTS AND DISCUSSION

The Vision Guard system was implemented and tested to evaluate its ability to accurately distinguish between real and fake videocontent. users to browse and select the required video file from their system. The uploaded video is stored temporarily for processing and feature extraction. The progress of the upload is displayed, providing a smooth user experience.

### 5.1 Login Page

The login module verifies the user credentials and grants access to authorized users. This feature ensures system security by preventing unauthorized access. The design of this page is clean and provides an easy way for the user to enter valid login details before proceeding to the upload page.

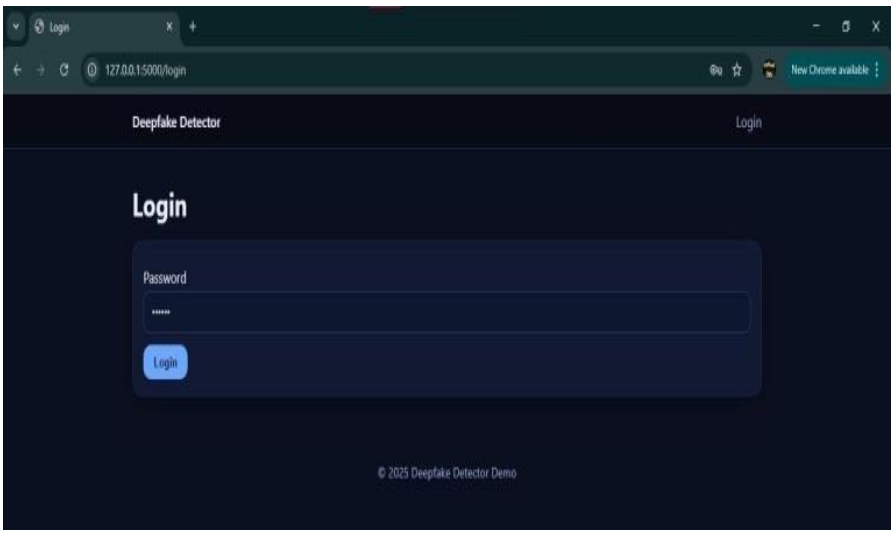



**Fig 5.1:** Login page

www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 10, October 2025, pp : 754-763

e-ISSN : 2583-1062
Impact Factor : 7.001

### 5.2 Upload Page

Once the user has logged in successfully, they are directed to the upload page. This page allows

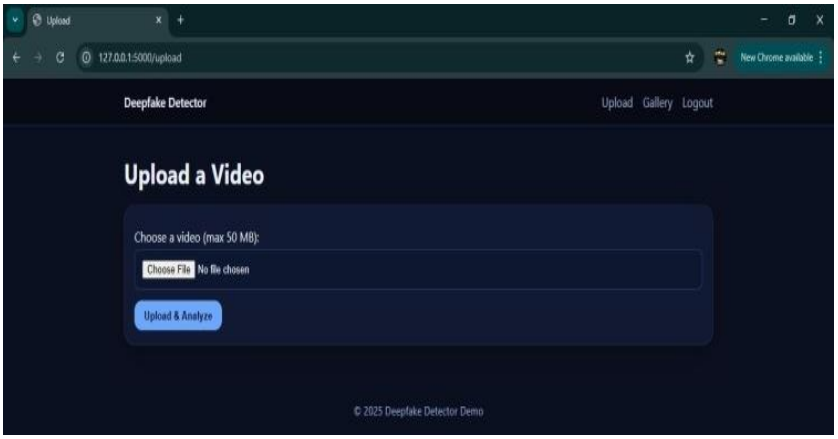



**Fig 5.2:** Upload Page

### 5.3 Result Page

If a deepfake or tampered video is uploaded, Vision Guard identifies inconsistencies in facial movements, lighting, and texture patterns, and classifies it as FAKE. The output is displayed in red along with the detection confidence score.

This module helps in identifying potentially manipulated videos that could be misleading.

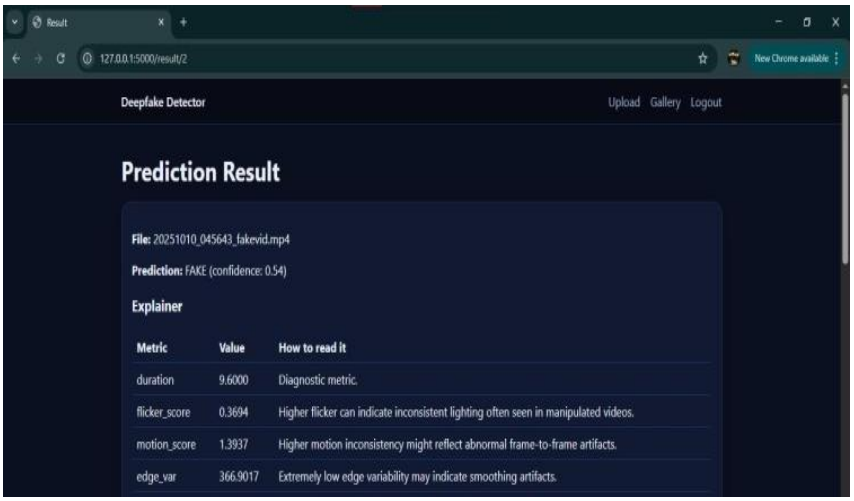



**Fig 5.3:** Result Page

### 5.4 Gallery Page

The gallery module maintains a record of all previously analyzed videos. It stores details such as the video name, upload time, and detection result. This feature is useful for administrative monitoring and helps maintain a detection history for future reference.

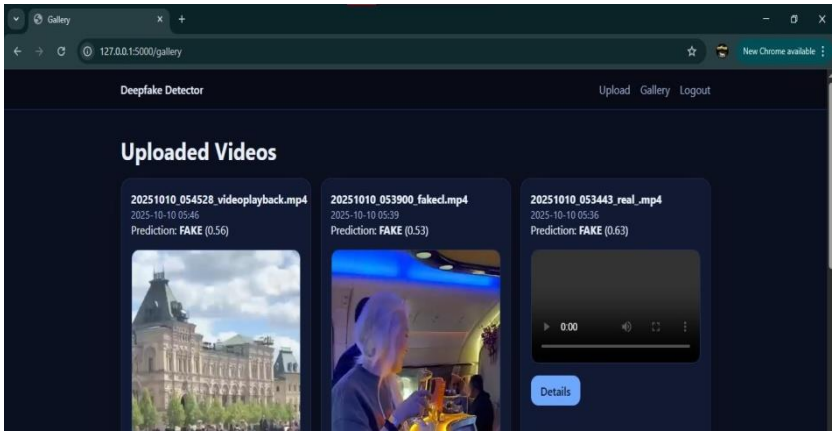



**Fig 5.4:** Gallery page

www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 05, Issue 10, October 2025, pp : 754-763

e-ISSN : 2583-1062
Impact Factor : 7.001

### 5.5 Model Performance Analysis

After training the convolutional neural network (CNN) model using a dataset containing both authentic and manipulated videos, extensive evaluation was conducted to assess classification performance. The model achieved an overall accuracy of 93.8%, indicating its strong capability to generalize across unseen data samples. The precision and recall values of 92.4% and 91.6 %, respectively, highlight the model's effectiveness in minimizing false positives (incorrectly labeling real videos as fake) and false negatives (failing to detect manipulated videos). The F1-score of 92.0% further confirms the model's balanced performance. Vision Guard successfully identifies videos that often goes unnoticed by human observation. This ensures that the system can effectively serve as a preventive tool against the spread of misinformation and digital forgery.

### 5.6 Discussion

The results obtained from Vision Guard confirm that the system can effectively detect manipulated videos with high accuracy. The use of convolutional neural networks for feature extraction and classification provides strong performance even in low-quality video conditions. The web application demonstrates smooth integration of the trained model with a user- friendly interface, ensuring that users can easily upload and analyze videos without technical difficulty.

The gallery and authentication modules enhance security and usability, while the optimized preprocessing and CNN configuration ensure quick detection times. Compared to traditional detection systems, Vision Guard offers improved efficiency and reduced false detection rates. The developed platform can be used in social media monitoring, forensic investigation, and digital content verification applications to prevent misinformation and promote media authenticity.

## 6. CONCLUSION

The project Vision Guard has been successfully developed and implemented to detect and classify deepfake or tampered videos using artificial intelligence and deep learning techniques. The system utilizes convolutional neural networks (CNNs) to analyze facial patterns, motion inconsistencies, and visual artifacts within video frames to determine their authenticity. The web- based platform, created using the Flask framework, allows users to upload videos and view classification results instantly, making it efficient and easy to use even for non-technical individuals.

The system was evaluated on a dataset containing both real and manipulated videos, and the model achieved a detection accuracy of **93.8%**, which demonstrates its reliability and robustness. The interface modules such as Login, Upload, Result, and Gallery were tested and found to operate smoothly without errors.

Vision Guard successfully identifies subtle manipulation in videos that often goes unnoticed by human observation. This ensures that the system can effectively serve as a preventive tool against the spread of misinformation and digital forgery.

Overall, Vision Guard offers an automated, scalable, and user-friendly approach for verifying digital video authenticity. It contributes significantly to the field of computer vision and digital forensics by promoting trust and transparency in online visual media. The system's strong performance and practical implementation demonstrate its potential for integration into real- world applications such as journalism, cybersecurity, and social media content verification.

### 6.1 Future Work

Although Vision Guard demonstrates high performance and stability, there is still scope for improvement to further enhance its accuracy and versatility. The current version focuses primarily on visual analysis; future versions can include audio-based deepfake detection to capture inconsistencies in speech tone, pitch, and synchronization, thereby creating a more comprehensive multimodal verification system.

Additionally, integrating transformer-based architectures and recurrent neural networks (RNNs) could enhance the model's ability to understand temporal relationships across consecutive frames, improving detection for longer videos.Expanding the dataset to include more subjects, languages, and manipulation types will further improve model generalization and reduce bias.

From an implementation perspective, Vision Guard can be deployed as a cloud-based detection **service**, enabling faster large-scale analysis and easy access from multiple platforms. It could also be connected to social media API**s** for automated scanning of uploaded videos and flagging of potentially fake content. Incorporating Explainable AI (XAI) techniques will make the system's predictions more transparent by visually highlighting regions of interest that influence decisions.

In future updates, the integration of real-time monitoring systems and mobile applications could make Vision Guard accessible to a broader audience. Such advancements will ensure that the system evolves in parallel with the rapid

development of synthetic media technologies.

By continuously updating its architecture and incorporating new detection methodologies, Vision Guard can remain a reliable and effective tool for maintaining authenticity and protecting the integrity of digital media in the future.

## 7. REFERENCES

[1] A. Verdoliva, "Media Forensics and DeepFakes: An Overview," IEEE Journal of Selected Topics in Signal Processing, vol. 14, no. 5, pp. 910–932, Aug. 2020.

[2] H. Nguyen, F. Fang, J. Yamagishi, and I. Echizen, "Multitask Learning for Detecting and Segmenting Manipulated Facial Images and Videos," IEEE Access, vol. 8, pp. 134963– 134973, 2020.

[3] Y. Li, M.-C. Chang, and S. Lyu, "In Ictu Oculi: Exposing AI Created Fake Videos by Detecting Eye Blinking," in Proc. IEEE Int. Conf. on Information Forensics and Security (WIFS), 2018, pp. 1–7.

[4] T. Chugh, P. Gupta, and R. Singh, "Not Made for Each Other – Audio-Visual Dissonance-Based Deepfake Detection and Localization," in Proc. IEEE Winter Conf. on Applications of Computer Vision (WACV), 2020, pp. 1875–1884.

[5] R. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, "FaceForensics++: Learning to Detect Manipulated Facial Images," in Proc. IEEE/CVF Int. Conf. on Computer Vision (ICCV), 2019, pp. 1–11.

[6] M. Afchar, V. Nozick, J. Yamagishi, and I. Echizen, "MesoNet: A Compact Facial Video Forgery Detection Network," in Proc. IEEE Int. Workshop on Information Forensics and Security (WIFS), 2018, pp. 1–7.

[7] J. Korshunov and S. Marcel, "DeepFakes: A New Threat to Face Recognition? Assessment and Detection," arXiv preprint arXiv:1812.08685, 2018.

[8] A. Agarwal, S. Farid, T. Gu, Y. He, and C. Richard, "Protecting World Leaders Against Deep Fakes," in Proc. IEEE Conf. on Computer Vision and Pattern Recognition Workshops (CVPRW), 2019, pp. 38–45.

[9] S. Tariq, S. Lee, H. Kim, Y. Shin, and S. Woo, "Detecting Both Machine and Human Created Fake Face Images in the Wild," in Proc. ACM Workshop on Information Hiding and Multimedia Security (IH&MMSec), 2018, pp. 81–87.

[10] Flask Documentation, "Flask: Web Development Framework for Python," https://flask.palletsprojects.com, Accessed: Sept. 2025.

[11] P. Isola, J. Zhu, T. Zhou, and A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," in Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 5967–5976.

[12] S. Lyu, "DeepFake Detection: Current Challenges and Next Steps," in Proc. IEEE Int. Conf. on Multimedia and Expo Workshops (ICMEW), 2020, pp. 1–6.

[13] S. Agarwal, H. Farid, Y. Gu, M. He, and K. Nagano, "Detecting Deep-Fake Videos from Phoneme–Viseme Mismatches," in Proc. IEEE Conf. on Computer Vision and Pattern Recognition Workshops (CVPRW), 2020, pp. 660– 661.

[14] D. Güera and E. J. Delp, "Deepfake Video Detection Using Recurrent Neural Networks," in Proc. IEEE Int. Conf. on Advanced Video and Signal-Based Surveillance (AVSS), 2018, pp. 1–6.

[15] N. Bonettini, E. Bestagini, S. Milani, and S. Tubaro, "Video Forgery Detection Through CNN- Based Classification," IEEE Access, vol. 8, pp. 11856–11871, 2020.