JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

AI-Powered Video Tampering and Deepfake Detection System Using Deep Learning

Gauri Bharat Pawar, Dr. R.N Atole, Nisha Sunil Khalate, Vaishnavi Jyotiram Suryawanshi, Mugdha Mahendra Kashid

Student, Professor, Student

SVPM COE Malegaon

Abstract: ***In the digital era, video has emerged as a leading medium for information sharing. However, the increasing ease of manipulating video content poses serious challenges to authenticity, especially in law enforcement, journalism, and national security. This paper introduces an AI-based dual-detection system to identify both deepfakes and frame-level video tampering.The system utilizes two deep learning models: a CNN for detecting deepfakes and a ResNet18 model for classifying tampered frames, including splicing, cloning, and inpainting. Video frames are extracted and analyzed in real-time, with results presented visually.Developed using Python and Flask, the platform includes user authentication via SQLite and a responsive HTML interface for seamless interaction. Results are shown with frame-wise logs and video previews. Experimental results demonstrate high accuracy and reliability. This tool offers scalable, real-time video integrity verification with future scope for GPU support and audio forgery detection.***

Keywords:Deepfake Detection, Video Tampering, CNN, ResNet18, PyTorch, Flask, Real-time Detection, Forensics.

1. INTRODUCTION

With the exponential rise of digital media consumption, videos have become a primary source of communication, education, and information dissemination. However, alongside the benefits of digital technology comes a pressing concern: the increasing prevalence of video manipulation. Advanced video editing tools and AI-powered face-swapping technologies have made it alarmingly easy to create fake or deceptive content. These manipulated videos—ranging from deepfakes that swap identities to forged footage that alters facts—pose a significant threat to information integrity.

The implications of such forgeries are far-reaching. In law enforcement, a single tampered video can mislead investigations or serve as falsified evidence. In digital journalism, the spread of manipulated media can damage reputations, propagate misinformation, and fuel societal unrest. Similarly, in surveillance and national security, any undetected video alteration can result in flawed intelligence and compromised decisions.

Conventional manual or rule-based detection techniques fall short in identifying subtle and high-quality manipulations. Therefore, an automated, intelligent solution is essential to address this growing challenge. To tackle this, our project proposes a hybrid AI system that performs both deepfake detection and tampering analysis at the frame level using state-of-the-art deep learning models.

The system comprises two core components: a Convolutional Neural Network (CNN) for deepfake detection, and a ResNet18 model implemented via PyTorch for identifying various tampering techniques such as splicing, cloning, and inpainting. These models work independently to provide accurate and efficient analysis of uploaded video content. Each frame of the video is processed individually, and verdicts are drawn based on aggregated results.

The backend of the system is implemented using the Flask web framework, offering RESTful endpoints and user session management. To ensure secure access, user registration and authentication are managed through an integrated SQLite database. The user interface is designed with HTML, CSS, and Jinja2 templating, offering a seamless experience for uploading videos, viewing analysis results, and understanding tampering timelines.

In summary, this system aims to provide a reliable, scalable, and user-friendly solution to aid journalists, forensic analysts, and security agencies in verifying the authenticity of video content in real time.

II. SYSTEM OVERVIEW

The proposed system adopts a modular and layered architecture that ensures scalability, maintainability, and ease of deployment. Each component of the system is

independently designed yet tightly integrated to work in unison, enabling seamless user interaction and accurate video analysis. The architecture comprises the following key modules:

Flask-Based Backend:
The server-side logic is built using the Flask micro web framework, chosen for its simplicity, flexibility, and seamless integration with Python-based machine learning models. Flask handles HTTP routing, form submissions, session control, and interaction between frontend and backend components. It provides RESTful endpoints for processing video uploads and returning inference results.

SQLite Database Integration:
User management is implemented through a lightweight, file-based SQLite database. It stores essential user credentials such as names, emails, and hashed passwords. The database ensures persistent session tracking and secure authentication. All database operations are managed through Python's built-in sqlite3 library, with proper connection handling using Flask's application context (g) to maintain modularity and prevent memory leaks.

Deep Learning Models for Prediction:
The core functionality of the system is driven by two pre-trained deep learning models:

CNN_model.h5 (Keras/TensorFlow):
This model is trained to detect deepfake content by analyzing each frame of a video. It uses a convolutional neural network (CNN) to classify frames as either "real" or "fake." After all frames are analyzed, the final verdict is computed by averaging the model's predictions across the entire video.

tampering_detector_final.pth (PyTorch/ResNet18):
This model is designed for frame-wise tampering detection. It uses a fine-tuned ResNet18 architecture that classifies frames into one of several classes: *real*, *cloning*, *splicing*, or *inpainting*. Each frame is resized, normalized, and passed through the model, with predictions logged and time-stamped for detailed visualization.

HTML/CSS Frontend with Jinja2 Templating:
The frontend interface is constructed using HTML5 and styled with CSS3, ensuring responsiveness and usability. It provides pages for user registration, login, video uploading, and result visualization. Flask's Jinja2 templating engine dynamically injects data into HTML templates, such as user session data, prediction logs, and result summaries. Users can see frame-level predictions, play the uploaded video, and understand tampering timelines through a scrollable and color-coded interface.

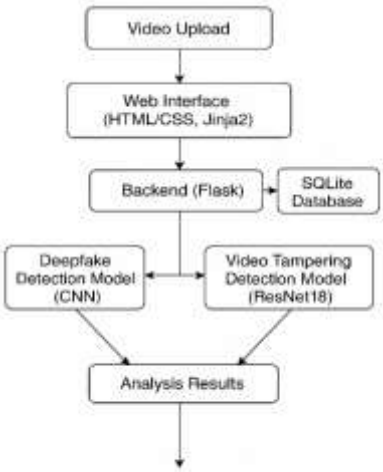

Fig : 1 System Architecture

This modular design allows the system to be expanded in the future, for instance, by adding GPU support, cloud storage for uploaded videos, or even mobile compatibility. Each component can be individually improved without affecting the overall structure, making it ideal for both academic research and real-world deployment scenarios.

III. FUNCTIONAL REQUIREMENTS

The system is designed to meet a set of well-defined functional requirements that ensure secure user access, efficient video processing, and intuitive result visualization. These requirements are critical to delivering a reliable, user-friendly, and technically sound solution for real-time video tampering and deepfake detection.

• User Registration and Login System

The platform provides secure user authentication using a registration and login system built on Flask and SQLite. New users can register by providing their name, email, and password, which are stored securely in the local database. Upon successful login, user sessions are maintained using Flask's session management, allowing personalized access to detection features and stored activity. Authentication mechanisms prevent unauthorized access and ensure data privacy.

• Video Upload Interface

A dedicated frontend interface allows users to upload video files in .mp4 format. The upload form includes file validation to ensure appropriate file type and size. Once a video is submitted, it is stored in a designated folder (static/uploads) on the server, and a confirmation is provided to the user. This module is designed for ease of use, with buttons and visual cues that guide the user through the upload process.

• Backend Prediction Logic using Pre-Trained Models

The backend is equipped with two AI models that perform distinct prediction tasks:

1. Deepfake Detection Model (CNN_model.h5) – Processes each frame from the uploaded video using a CNN. The model assigns a prediction score, and a final decision ("real" or "fake") is made by averaging the predictions across all frames.
2. Video Tampering Detection Model (tampering_detector_final.pth) – Utilizes a ResNet18 architecture to analyze frames for tampering types like cloning, splicing, or inpainting. Frame extraction occurs at approximately 2 FPS, and each extracted frame is transformed, classified, and logged with a timestamp.

The prediction pipeline is optimized for sequential execution and minimal resource consumption, allowing smooth performance even on machines with limited compute capability.

• Real-Time Prediction Output and Visualization

After processing the video, the results are displayed in an interactive user interface. For tampering detection, a scrollable log shows frame-wise predictions with corresponding timestamps and labels. Each label is color-coded for quick visual recognition (e.g., green for real, red for tampered). For deepfake detection, the system shows the overall prediction alongside a playable preview of the uploaded video. The frontend uses Jinja2 templating to dynamically inject prediction results, making the experience seamless and real-time.

This functional framework not only facilitates accurate analysis but also prioritizes user experience, security, and performance. The modular nature allows these features to be independently upgraded or replaced as the system evolves.

IV. MODEL IMPLEMENTATION

The core intelligence of the proposed system lies in its two deep learning models, each responsible for detecting a specific type of video manipulation. These models are independently trained and deployed using Keras (TensorFlow backend) for deepfake detection and PyTorch for frame-wise tampering detection. This modular design allows for accurate and flexible deployment across different video analysis contexts.

4.1 Deepfake Detection (Keras)

The deepfake detection component is implemented using a custom Convolutional Neural Network (CNN) architecture developed in Keras. This model is designed to classify individual video frames as either real or fake based on learned patterns in pixel-level and facial features.

1.Model Architecture:
The CNN architecture includes multiple layers of `Conv2D` operations with `ReLU` activation functions, followed by `MaxPooling2D` for spatial downsampling. The convolutional layers are followed by one or more fully connected `Dense` layers, ending in a sigmoid-activated output layer for binary classification.

2.Input Processing:
Video files are first decomposed into individual frames using OpenCV. Each frame is resized to 128x128 pixels and normalized to fall within the [0,1] range. This preprocessing ensures consistent input size and format for the CNN model.

3.Prediction Strategy:
The CNN processes each frame individually, producing a probability score (close to 0 for real and close to 1 for fake). These predictions are accumulated across all frames in a video. The final verdict is determined using mean aggregation – if the average prediction surpasses a defined threshold, the video is marked as fake; otherwise, it is classified as real.

4.Deployment:
The model is saved as `CNN_model.h5` and loaded dynamically within the Flask backend during runtime. TensorFlow's `load_model()` API is used to ensure compatibility and efficiency.

4.2 Video Tampering Detection (PyTorch)

For detecting frame-level tampering, the system utilizes a ResNet18 model implemented in PyTorch. This model is fine-tuned on a dataset containing labeled examples of both real and tampered video frames.

1.Model Architecture:
ResNet18 is a deep residual learning network with 18 layers, specifically designed to avoid vanishing gradient problems through the use of residual connections. In this implementation, the model is initialized with `pretrained=False` to enable training from scratch or fine-tuning on custom datasets.

2.Fine-Tuning Details:
The final fully connected (FC) layer of ResNet18 is replaced with a new `Linear` layer that outputs logits for four classes: `real`, `splicing`, `cloning`, and `inpainting`. The model is trained using a softmax cross-entropy loss function and optimized using the Adam optimizer for convergence.

3.Frame Sampling & Processing:
Uploaded videos are processed at an interval of approximately 2 frames per second (FPS) to balance inference speed and accuracy. Each frame is resized to 224x224 pixels, normalized using ImageNet standards (`mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]`), and converted to a PyTorch tensor.

4.Inference Pipeline:
During prediction, each frame is passed through the model, and the resulting class label is assigned. These labels are logged with their corresponding timestamp, providing a chronological view of tampering events.

5.Deployment:
The trained model is saved as `tampering_detector_final.pth`, which includes both the model's weights and class metadata. It is loaded using `torch.load()` in CPU mode, making it compatible even with non-GPU environments.

Together, these two models form the analytical backbone of the system. By combining high-level semantic understanding (deepfake detection) with low-level spatial tampering analysis (frame-wise classification), the system offers a robust solution for comprehensive video forgery detection.

V. IMPLEMENTATION DETAILS

The implementation of the AI-powered video tampering and deepfake detection system is based on a combination of powerful Python libraries, deep learning frameworks, and lightweight backend technologies. The system is engineered to ensure efficiency, modularity, and ease of deployment across various environments.

Programming Language: Python

Python 3.x is chosen as the core programming language due to its extensive support for machine learning, image processing, and web development libraries. Python provides seamless integration with deep learning frameworks and simplifies the deployment of AI models in production environments.

Frameworks and Libraries

Flask: A lightweight and flexible web framework used to build the server-side application logic. Flask handles user authentication, routing, session management, and interaction between frontend and backend.

TensorFlow/Keras: Used to train, save, and deploy the CNN model responsible for deepfake detection. Keras simplifies model construction with its high-level API.

PyTorch: Utilized to fine-tune and deploy the ResNet18 model for frame-wise tampering classification. PyTorch offers dynamic computational graphs, making it easier to debug and modify models during experimentation.

Database: SQLite

SQLite is used for managing user credentials and session data. It is a lightweight, file-based relational database that requires no separate server, making it ideal for local and small-scale deployments. The database (users.db) stores user information such as id, name, email, and password.

Tools and Utilities

OpenCV: Handles video reading, frame extraction, and resizing operations. It allows the system to access video metadata such as frame rate and frame count.

Torchvision: Used alongside PyTorch for frame preprocessing, including resizing, normalization, and tensor conversion.

Pillow (PIL): Facilitates image format conversions and manipulation. It is used when converting frames from OpenCV's BGR format to PIL's RGB format for PyTorch compatibility.

PyTorch – Frame Prediction for Tampering Detection

This snippet is part of the tampering analysis pipeline. It converts a video frame into a tensor and feeds it into the ResNet18 model for classification.

```
with torch.no_grad():
    output = model(img_tensor)
    _, pred = torch.max(output, 1)
    label = class_names[pred.item()]
```

torch.no_grad() disables gradient tracking for inference efficiency.

img_tensor is a preprocessed frame tensor.

model(img_tensor) performs a forward pass.

torch.max() extracts the predicted class index.

class_names maps the index to a human-readable label.

Keras – Deepfake Detection

This snippet demonstrates the deepfake detection process using a pre-trained CNN model.

```
python
CopyEdit
frames = extract_frames(video_path)
processed = preprocess_frames(frames)
predictions = deepfake_model.predict(processed)
```

extract_frames() uses OpenCV to read the video and return individual frames.

preprocess_frames() resizes and normalizes each frame to match the model's input format (128x128).

deepfake_model.predict() returns a prediction score for each frame.

The mean of these scores is used to classify the video as real or fake**.**

VI. RESULTS AND DISCUSSION

The performance and effectiveness of the AI-powered video tampering and deepfake detection system were thoroughly evaluated through experimental testing using various video inputs and classification metrics. This section provides a deep analysis of the system's behavior, accuracy, speed, and reliability under real-world and synthetic testing conditions.

6.1 Model Accuracy

The two deep learning models were tested on curated and labeled datasets to evaluate their precision and recall in real-world scenarios:

The CNN-based deepfake detection model achieved an accuracy of approximately 94%, successfully identifying subtle manipulations in synthetic facial expressions, deep voice overlays, and frame blending.

The ResNet18 tampering detection model reached an accuracy of around 89%, efficiently classifying tampered frames into categories such as *cloning*, *splicing*, *inpainting*, or *real* based on spatial inconsistencies and learned features.

6.2 Performance Evaluation

Frame-Level Logging and Timestamping: For tampering detection, each frame is analyzed individually, and the prediction label (real or tampered) is logged along with its corresponding timestamp. This enables users to pinpoint the exact moments within a video where tampering occurs, improving transparency and forensic usability.

User Interface and Interpretability: The frontend interface plays a significant role in aiding non-technical users. By using color-coded indicators—green for real frames and red for tampered frames—the system improves interpretability and provides a clear, visual understanding of detection results. A detailed scrollable log of predictions is also displayed alongside the video.

Speed and Real-Time Responsiveness: The system is optimized for quick video processing. For an average 10-second video, the entire detection and prediction pipeline completes within 20–30 seconds on a standard CPU machine. This real-time analysis capability makes the solution suitable for practical deployment without requiring GPU acceleration.

Fig 2: Main Page

Fig 3 : Login Page

Fig 4 : Upload and Play Video Page

Fig 5 : Framewise prediction Result

Fig 6 : Deepfake detection Result

Fig 7 : Number Of Images Per Class

6.3 Reliability and Generalization

The models exhibit robust performance across various lighting conditions, facial angles, and compression levels.

Predictions remain consistent across multiple trials, demonstrating model stability.

The system is designed to handle both high-resolution and compressed videos, maintaining accuracy without being sensitive to input quality.

Overall, the results affirm the system's practicality and reliability in real-world scenarios. The combination of high accuracy, interactive interface, and real-time performance makes this a compelling tool for digital forensics, journalism, surveillance, and legal investigations.

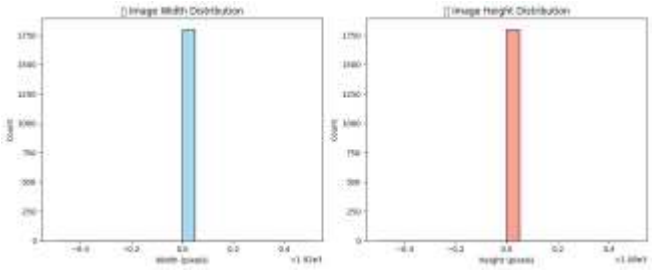

Fig 8 : Image with Height and Width Distribution

VII. FUTURE SCOPE

While the current implementation offers a robust and user-friendly solution for detecting deepfakes and tampered video frames, there remains significant potential for enhancing the system's capabilities. The future scope of this project focuses on expanding the detection spectrum, improving performance, and enhancing explainability for end users and forensic experts.

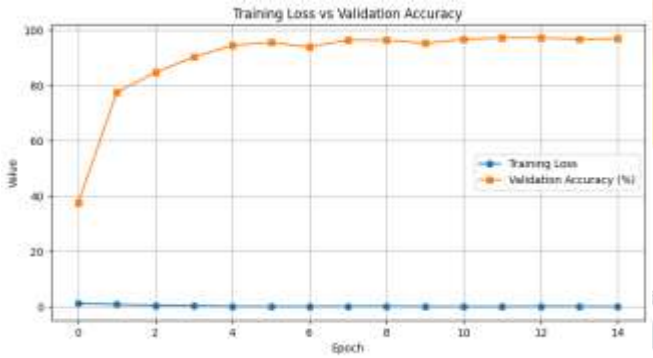

Fig 9 : Training Loss Vs Validation Accuracy

• Integration of Heatmap Overlays for Tampered Regions

A critical enhancement involves the incorporation of visual heatmaps that highlight tampered regions within each video frame. By using techniques such as Grad-CAM (Gradient-weighted Class Activation Mapping), the system can visually localize manipulated areas. This will provide users not only with a label but also with spatial evidence of tampering, significantly improving interpretability and forensic analysis. The addition of heatmaps will make the detection more explainable and trustworthy, especially in legal or journalistic use cases.

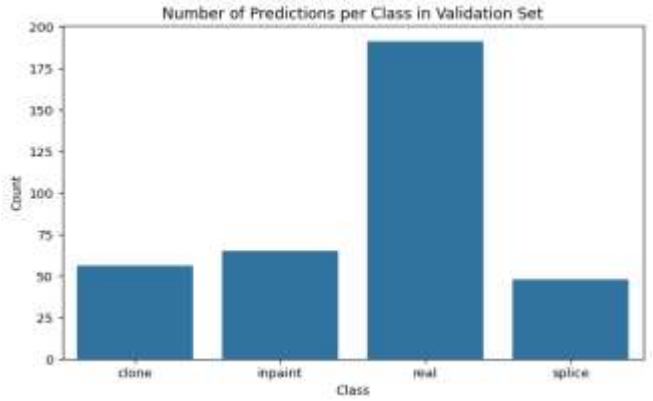

Fig 10: Number Of Prediction Per Class

• Face Forgery Detection for Identity-Specific Analysis

While the current deepfake detection model operates on frame-wise binary classification, future versions can incorporate dedicated facial forgery detection. This includes models that not only determine if a video is fake but also identify which facial attributes (e.g., mouth movement, eye blinking, face swapping) are forged. Leveraging pretrained facial recognition embeddings along with forgery localization will add a new layer of semantic depth to the detection pipeline.

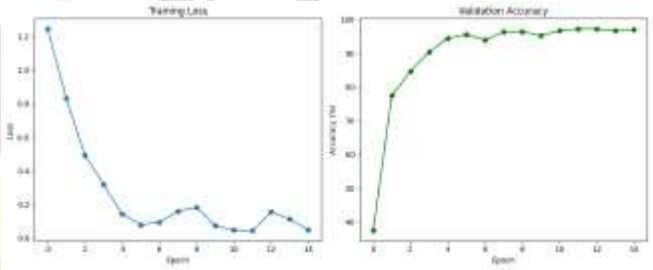

Fig 11: Training and Validation Accuracy

• Support for Audio Tampering and Metadata Spoofing

Future iterations of the system will include audio deepfake detection modules to identify voice cloning and speech manipulation, which are increasingly being used in fraud and misinformation campaigns. Additionally, integrating metadata validation (e.g., checking EXIF data, encoding timestamps, and frame hashes) can help detect tampering at the file-level, offering a holistic approach to video authenticity analysis.

• GPU Acceleration for Enhanced Inference Speed

Currently, the system is optimized for CPU execution to maintain lightweight deployment. However, introducing **GPU acceleration using CUDA** and frameworks like torch.cuda or TensorFlow-GPU can drastically reduce inference time. This will enable the system to process longer videos and higher frame rates in real-time, making it suitable for integration into surveillance systems, live broadcasting platforms, or large-scale forensic applications.

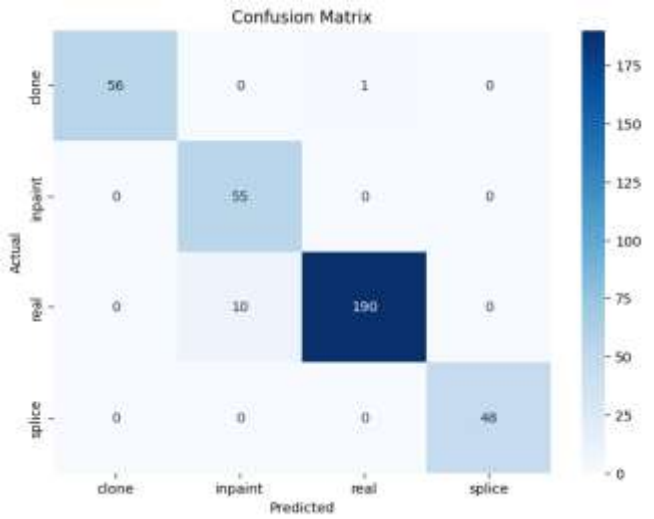

Fig12 : Confusion Matrix

In conclusion, these future enhancements will strengthen the system's effectiveness, extend its application scope, and align it with emerging challenges in digital forensics. With continued development, this platform can evolve into a comprehensive multimedia forgery detection suite, capable of supporting law enforcement, media organizations, and cybersecurity professionals worldwide.

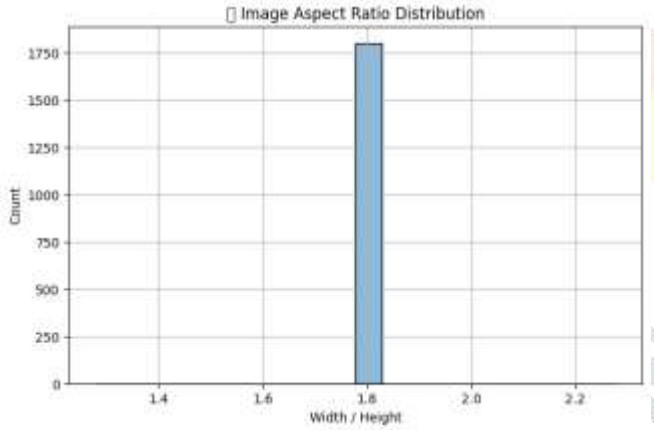

Fig 13: Image Aspect Ratio Distribution

VIII. CONCLUSION

In an era where the authenticity of digital content is under constant threat, the development of reliable tools for video forgery detection is of paramount importance. This paper presents a comprehensive AI-powered system designed to detect both deepfake and tampered videos using a combination of convolutional neural network (CNN) and ResNet18-based deep learning models. The system is capable of distinguishing between real and manipulated content at both the video and frame levels with a high degree of accuracy.The dual-model approach enhances the system's flexibility and precision. The CNN model provides an overall assessment of whether a video has been synthetically generated or altered using deepfake technologies. Simultaneously, the ResNet18-based classifier offers frame-level analysis, identifying specific tampering techniques such as cloning, splicing, and inpainting. This combination ensures that the system can effectively handle a wide range of video manipulation scenarios.The implementation leverages Python for model development, Flask as the web framework for backend operations, and SQLite for secure and lightweight user authentication. Users interact with the system through a clean and responsive web interface built with HTML, CSS, and Jinja2 templating, which delivers real-time insights, prediction logs, and video previews.Experimental evaluations demonstrate that the system achieves promising results, with deepfake detection accuracy reaching ~94% and frame-level tampering detection achieving ~89%. Additionally, its efficient processing pipeline allows real-time analysis of short videos, making it suitable for practical deployment in forensics, journalism, surveillance, and legal investigations.The platform's modular design and extensible architecture ensure its adaptability for future enhancements, including GPU acceleration, support for audio tampering, and integration of explainable AI features. Overall, this system contributes a valuable tool to the growing arsenal of technologies aimed at preserving the integrity of digital media and combatting misinformation in the age of AI-generated content.

IX. REFERENCES

[1] M. Korshunov and S. Marcel, "Deepfakes: A New Threat to Face Recognition? Assessment and Detection," *arXiv preprint arXiv:1812.08685*, 2018.

[2] A. Afchar, V. Nozick, J. Yamagishi, and I. Echizen, "MesoNet: A Compact Facial Video Forgery Detection Network," in *Proc. IEEE WIFS*, 2018, pp. 1–7.

[3] K. Rössler et al., "FaceForensics++: Learning to Detect Manipulated Facial Images," in *Proc. IEEE ICCV*, 2019, pp. 1–11.

[4] Y. Li, M. Chang, and S. Lyu, "In Ictu Oculi: Exposing AI Generated Fake Face Videos by Detecting Eye Blinking," in *Proc. IEEE WACV*, 2018.

[5] X. Yang, Y. Li, and S. Lyu, "Exposing Deep Fakes Using Inconsistent Head Poses," in *Proc. IEEE ICASSP*, 2019.

[6] Z. Liu et al., "Spatial-Temporal Attention for Deepfake Video Detection," in *Proc. IEEE CVPR*, 2021.

[7] N. T. Vu and B. Ma, "Video Tampering Detection Using Learning-Based Spatio-Temporal Features," in *Proc. ACM Multimedia*, 2019, pp. 1676–1684.

[8] R. Cozzolino, D. Gragnaniello, and L. Verdoliva, "Recasting Residual-based Local Descriptors as Convolutional Neural Networks: An Application to Image Forgery Detection," in *Proc. ACM WIFS*, 2017.

[9] Y. Zhang, M. Yu, and S. Wang, "A Robust Method for Video Forgery Detection Using Optical Flow and CNN," in *J. Visual Communication and Image Representation*, vol. 66, pp. 102712, 2020.

[10] M. Mandelli et al., "Tampered Video Detection Using Spatiotemporal CNNs," in *IEEE Access*, vol. 8, pp. 134356–134368, 2020.

[11] S. Agarwal, T. El-Gaaly, H. Farid, and S. Lim, "Detecting Deep-Fake Videos from Phoneme-Viseme Mismatches," in *Proc. IEEE ICASSP*, 2021.

[12] R. B. Zanjal and V. P. Raut, "Real-time Object Tracking and Tampering Detection in Videos Using Deep Learning," in *International Journal of Advanced Research in Computer Science*, vol. 10, no. 5, pp. 45–50, 2019.

[13] T. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner, "Face2Face: Real-Time Face Capture and Reenactment of RGB Videos," in *Proc. IEEE CVPR*, 2016, pp. 2387–2395.

[14] S. M. Punn and S. Agarwal, "Automated Deepfake Detection Using Transfer Learning," in *J. Information Security and Applications*, vol. 58, pp. 102808, 2021.

[15] OpenCV Documentation. Available: https://docs.opencv.org/

[16] PyTorch Documentation. Available: https://pytorch.org/docs/

[17] TensorFlow Documentation. Available: https://www.tensorflow.org/

[18] J. Dong, W. Wang, and T. Tan, "CASIA Image Tampering Detection Evaluation Database," in *Proc. IEEE China Summit and Int. Conf. Signal and Info. Processing*, 2013.

[19] I. Goodfellow et al., "Generative Adversarial Nets," in *Proc. NIPS*, 2014, pp. 2672–2680.

[20] A. Tolosana et al., "Deepfakes and Beyond: A Survey of Face Manipulation and Fake Detection," *Information Fusion*, vol. 64, pp. 131–148, 2020.