Deep Learning for Media Authentication and Fake Content Detection

1st Vinayak Rajput
Department of Computer Science,
Bangalore Institute of Technology,
Bangalore, India
vk.ra.org@gmail.com

2nd Sudeep Patil
Department of Computer Science,
Bangalore Institute of Technology,
Bangalore, India
sudeeppatill56@gmail.com

3rd Thushar D M
Department of Computer Science,
Bangalore Institute of Technology,
Bangalore, India
tushardm123@gmail.com

4th S Ashwin Reddy
Department of Computer Science,
Bangalore Institute of Technology,
Bangalore, India
sashwinreddy17@gmail.com

5th Prof. Nikitha Gowda K S
Assistant Professor,
Department of Computer Science,
Bangalore Institute of Technology,
Bangalore, India
nikithagowdaks@gmail.com

***Abstract*—In today's digital era, the widespread creation of manipulated media-ranging from subtly altered images to highly convincing deepfake videos presents serious risks to privacy, security, journalism, and public trust. Traditional forensic methods often struggle to detect these increasingly sophisticated forgeries. As a result, there is growing interest in using deep learning techniques for media authentication and fake content detection. This paper examines the role of deep neural networks, particularly convolutional neural networks (CNNs) and recurrent neural networks (RNNs), in identifying synthetic and tampered media. We explore architectures such as autoencoders, generative adversarial networks (GANs), and transformers, which play a key role in both generating and detecting fake content. Key challenges include limitations in available datasets, the ability to generalize to new types of manipulation, and the need for real-time detection. We also discuss widely used benchmark datasets, including FaceForensics++ and the DeepFake Detection Challenge (DFDC), and evaluate model performance using metrics like accuracy, precision, recall, and AUC. This study emphasizes the potential of deep learning as a crucial tool in combating digital misinformation, while also highlighting the need for AI solutions that are robust, interpretable, and ethically sound. To overcome the "black-box" nature of traditional AI, the framework incorporates LIME (Local Interpretable Model-agnostic Explanations), which provides a "Right to Explanation" by generating visual heatmaps that highlight the specific regions of an image such as unnatural eye reflections or blurred facial boundaries that led to a "Fake" verdict. Finally, to ensure the forensic integrity of these results, the system integrates a Local Ethereum Blockchain (Ganache); by recording the unique SHA-256 cryptographic hash of every analyzed file and its corresponding detection results on an immutable ledger, the system creates a tamper-proof "Chain of Custody" that prevents unauthorized data alteration and fosters absolute transparency for journalists, legal experts, and content moderators.**

***Index Terms*—Deep Learning, CNNs, RNNs, GANs, Transformers, Xception, LSTM, Blockchain, XAI**

I. Introduction

The rise of manipulated digital media has become a critical challenge in today's information landscape. With the advent of powerful generative technologies, particularly Generative Adversarial Networks (GANs), creating fake images and videos that appear indistinguishable from real ones has become increasingly effortless. These synthetic media, commonly known as deepfakes, can serve harmless purposes such as entertainment but pose severe risks when used to spread misinformation, manipulate public opinion, or perpetrate fraud. As these threats continue to evolve, the demand for robust and reliable methods to authenticate media and detect forgeries has grown substantially.

Deep learning has emerged as one of the most effective approaches in this field, owing to its ability to learn complex spatial and temporal patterns. Convolutional Neural Networks (CNNs) are widely employed in image-based forgery detection, identifying subtle inconsistencies in texture, lighting, and structure caused by manipulation. For video analysis, more advanced architectures such as Recurrent Neural Networks (RNNs) and transformers capture temporal anomalies, including unnatural facial expressions or mismatched audio-visuals. These models are typically trained on extensive datasets of authentic and manipulated media, such as FaceForensics++ and the DeepFake Detection Challenge (DFDC), to learn discriminative features that distinguish real content from synthetic fabrications.

To address these challenges, researchers are increasingly exploring hybrid modelling approaches, explainable AI techniques, and standardized benchmarks for evaluation. There is also growing awareness of the ethical considerations surrounding detection technologies, emphasizing the need to balance combating misinformation with protecting privacy and preventing misuse. Ultimately, deep learning stands at the forefront of efforts to safeguard the integrity of digital media, offering powerful tools to defend truth and trust in an age of pervasive synthetic content.

The proposed system addresses the critical need for a transparent and tamper-proof media verification pipeline. By

combining advanced neural networks with blockchain technology and interpretability tools, the framework ensures not just detection, but also the accountability and clarity of the results. Deep learning models are often viewed as ”black boxes.” To overcome this, the system incorporates XAI via LIME. Visual Evidence: When a media file is flagged as ”Fake,” LIME generates a Heatmap that is overlaid on the original image. Region-Specific Highlighting: These heatmaps highlight the specific areas (such as the mouth, eyes, or background edges) that contributed most to the model’s ”fake” prediction. To handle large-scale media verification efficiently, the system implements a Query-First Optimization strategy. Avoid Redundant Computation: Before initiating the resource-heavy deep learning analysis, the system first generates a unique cryptographic hash of the uploaded file and queries the Blockchain. Instant Verification: If the file has been analysed previously, the system retrieves the existing result directly from the ledger. This mechanism significantly reduces computational overhead, saves energy, and provides near-instant response times for previously seen content.

II. Literature Survey

Several studies have explored the application of deep learning technology to enhance security in media systems. This section reviews key contributions and identifies existing gaps. Njood ALShariah et al. [1] explored the detection of fake images specifically on Instagram using various machine learning and deep learning models such as CNNs, VGG variants, and ResNet. The study utilized feature extraction techniques including noise pattern and color inconsistencies to identify tampered content. The model achieved an accuracy of 97% and highlighted the effectiveness of deep learning models in social media contents. Md Shoel Ranan et al. [2] conducted a comprehensive review of deepfake detection techniques using machine learning, deep learning, and hybrid approaches. The study provided a detailed comparison of detection strategies, model architectures, and datasets like FaceForensics++, Celeb-DF, and DFDC.

Asad Malik et al. [3] presented a broad survey of deepfake generation and detection, particularly focusing on images and videos involving human faces. The paper examines domain generalization challenges and outlines the ethical and societal implications of synthetic media. V Venkata Reddy et al. [4] proposed a hybrid technique for fake image detection using a fusion of traditional image processing (texture-based analysis) and deep learning (CNN) approaches, with a focus on facial image manipulation. Raidah S Khudever et al. [5] created a platform for MCQ assessments using Ethereum, IPFS and HashiCorp Vault for key management. The system’s limitation to multiple-choice questions and its reliance on a centralized service (Pinata for IPFS) were notable drawbacks. Similarly, Peter Edwards et al. [6] delivered a comprehensive review of deepfake technologies, covering their creation, detection methodologies, and dataset limitations. It also suggests future directions in deepfake detection research.

III. Proposed System Architecture

The System Architecture of our project illustrates how the various components-ranging from user interaction to deep learning and blockchain interact to verify media. It is structured into four distinct layers: the User Interaction Layer, the Core Detection Engine, the Explainability Module, and the Blockchain Integrity Layer.

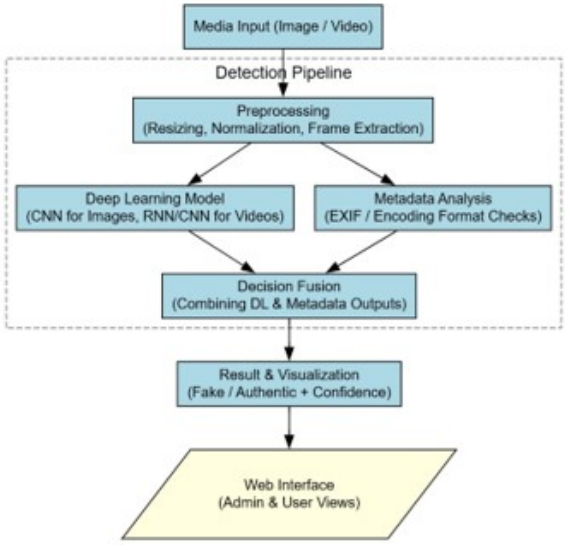

Fig. 1: High-level architecture of the MEDIA-Authentication System.

The Blockchain Integrity and Verification Module acts as the decentralized backbone of the framework, ensuring that media authentication is both permanent and tamper-proof. Utilizing a Solidity Smart Contract (DetectionLogger.sol) deployed on a local Ganache Ethereum testnet, the system establishes a mathematically verifiable ”Chain of Custody” for every file analyzed. Central to this module is Hash-Based Integrity, which generates a unique SHA-256 cryptographic fingerprint of the input media; this hash serves as a primary key, ensuring that even a single-pixel alteration to the file would result in a completely different identifier. To optimize performance, the system implements Query-First Logic, which instantly checks the ledger for an existing hash before initiating the resource-intensive AI models. If a match is found, the previous results are retrieved in milliseconds, significantly reducing computational overhead.

The Explainable AI (XAI) Module serves as the system’s transparency layer by demystifying the ”black-box” decisions of the deep learning models. This module utilizes Local Interpretable Model-agnostic Explanations (LIME) to perform a granular forensic analysis of the input media. Upon execution, the engine first performs Superpixel Segmentation, grouping pixels into semantically meaningful patches based on color and texture similarities. It then perturbs the image by selectively hiding these superpixels 1,000 times, observing how each change shifts the model’s confidence in its ”Real” or ”Fake”

classification. By assigning importance weights to these regions, the system generates a Visual Heatmap Overlay using the Viridis colormap. This high-contrast visualization pinpoint exactly which facial regions such as unnatural blending around the mouth or inconsistent shadows near the eyes most heavily influenced the detection of a deepfake.

IV. Methodology

This project employs a dual-model approach to address the multifaceted challenge of fake media detection. We developed two distinct models: one for analyzing spatial-temporal patterns in videos and another for analyzing spatial artifacts in static images.

A. Model 1: Spatial-Temporal Video Detection

To detect deepfake videos, we analyze both spatial and temporal data. We hypothesize that while individual frames (spatial) may be convincing, the temporal relationship between frames contains detectable artifacts (e.g., unnatural blinking, inconsistent lighting, or unnatural facial movements). Our method uses a hybrid CNN-LSTM architecture. A Convolutional Neural Network (CNN) acts as a feature extractor for each frame, identifying complex spatial features. The resulting sequence of feature vectors is then passed to a Long Short-Term Memory (LSTM) network. The LSTM is specifically designed to model temporal dependencies and identify anomalies across the sequence, allowing the model to "watch" the video and spot inconsistencies over time.

B. Model 2: Static Image Detection

For static image detection, the model focuses solely on spatial artifacts. These include inconsistencies in texture, lighting, logical errors (e.g., mismatched earrings), or digital "fingerprints" left by the generative (GAN) process. We use a transfer learning approach with a pre-trained, high-performance CNN, specifically the Xception architecture. The model leverages the powerful, generalized feature representations learned by Xception on the large-scale ImageNet dataset. We replace the original classification head with a new, custom classifier trained to distinguish between 'real' and 'fake' images based on these extracted features. Heavy data augmentation is used during training to prevent overfitting and improve the model's ability to generalize to new, unseen fakes.

V. System Implementation

This section details the technical implementation, data handling, and training procedures for the two models described in the methodology.

A. Video Detection Model (CNN-LSTM)

1) Data Pre-processing and Sequencing: Dataset: The model is trained on the Celeb-DF (v2) dataset, which contains real videos (Celeb-real) and corresponding deepfakes (Celeb-synthesis). Video-to-Sequence Generation: A custom `VideoSequenceGenerator` class is implemented to load and process video files in batches, as they cannot be loaded into memory at once. Frame Sampling: For each video, a sequence of $T = 10$ frames is sampled using uniform temporal sampling (np.linspace). Normalization: Each 224×224 frame's pixel values P are normalized from $[0, 255]$ to $[0, 1]$.

$$P_{\text{normalized}} = \frac{P_{\text{original}}}{255.0} \tag{1}$$

Final Tensor: The output for one video is a tensor of shape $(10, 224, 224, 3)$. Data Splitting: The dataset is split into an 80% training set and a 20% validation set, stratified to maintain the class balance.

2) CNN-LSTM Model Architecture: A Sequential Keras model is built to process the input sequences. Spatial Feature Extraction (CNN): A pre-trained Xception model with 'imagenet' weights is used as the convolutional base. Global Average Pooling (GAP) (`pooling='avg'`) is applied to the Xception output, converting each frame's feature map into a single feature vector f_k.

$$f_k = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} a_{ijk} \tag{2}$$

The Xception base weights are frozen (`trainable = False`). A TimeDistributed layer applies this frozen CNN-GAP model to each of the 10 frames in the sequence:

$$F_t = \text{CNN-GAP}(I_t) \quad \text{for } t = 1, \ldots, 10 \tag{3}$$

Temporal Processing (LSTM): The sequence of 10 feature vectors $(F_1, \ldots, F_{10})$ is fed into an LSTM layer with 128 units. The LSTM cell updates its cell state C_t and hidden state h_t at each time step t using input F_t and previous state h_{t-1}. The core update equation is:

$$C_t = (f_t \odot C_{t-1}) + (i_t \odot \tilde{C}_t) \tag{4}$$

Where f_t is the forget gate, i_t is the input gate, and $\tilde{C}_t$ is the candidate cell state. With `return_sequences = False`, the model outputs only the final hidden state h_{10}, summarizing the entire sequence.

Classification Head: The summary vector h_{10} is passed through a classifier. Dense (64, `activation='relu'`): A fully connected layer with ReLU activation, $\text{ReLU}(z) = \max(0, z)$. Dropout (0.5): Two dropout layers are used for regularization. Dense (1, `activation='sigmoid'`): The final output neuron uses a sigmoid function to produce a probability $\hat{y}$ between 0 (Fake) and 1 (Real).

3) Training and Optimization: Compiler: The model is compiled with the Adam optimizer (learning rate 1×10^{-4}). Loss Function: Binary Cross-Entropy is used to measure the loss $\mathcal{L}$ between the true label y and the prediction $\hat{y}$.

$$\mathcal{L}(y, \hat{y}) = -\left[y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})\right] \tag{5}$$

Early Stopping: An EarlyStopping callback monitors the validation loss (`val_loss`) and stops training if no improvement occurs for 5 epochs, restoring the best weights.

Algorithm: Video Detection Training:

1) Load video paths and labels from Celeb-DF dataset.
2) Split data into 80% training, 20% validation (stratified).

3) Initialize `VideoSequenceGenerator` for train and val sets.
4) Define CNN-LSTM model: Input layer: $(10, 224, 224, 3)$ → TimeDistributed (Xception (frozen) + GAP) → LSTM (128, `return_sequences=False`) → Dense (64, `'relu'`) → Dropout (0.5) → Dense (1, `'sigmoid'`).
5) Compile model: Adam ($lr = 1 \times 10^{-4}$), `binary_crossentropy`.
6) Initialize EarlyStopping (`monitor='val_loss'`, `patience=5`).
7) Train model using generators and callback.
8) Save the best performing model weights.

B. Static Image Detection Model (Xception CNN)

1) Data Pre-processing and Augmentation: Dataset: The model uses a directory of `processed_frames`, sorted into fake and real sub-folders. Data Augmentation: The `ImageDataGenerator` applies random transformations to the training set to improve robustness and prevent overfitting: Rotation, width/height shift, shear, zoom, and horizontal flip. Normalization and Loading: All images are rescaled (pixels divided by 255.0) and loaded from the directory using `flow_from_directory`. The dataset is split 80% for training and 20% for validation, resized to 224×224 and processed in batches of 32.

2) Model Architecture (Transfer Learning): A Sequential Keras model is constructed using the Xception architecture. Base Model (Feature Extractor): The Xception model, pre-trained on ImageNet, is loaded with `include_top=False`. The entire base model is frozen (`base_model.trainable = False`) to act as a fixed feature extractor. New Classification Head: `GlobalAveragePooling2D()`: This layer is applied to the output of the Xception base, converting the spatial feature map into a single feature vector. Dense (256, `activation='relu'`): A fully connected layer for high-level feature interpretation. Dropout (0.5): A dropout layer for regularization. Dense (1, `activation='sigmoid'`): The final output neuron for binary classification, producing a probability $\hat{y}$.

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{6}$$

3) Training and Optimization: Compiler: The model uses the Adam optimizer with a low learning rate of 1×10^{-4}. Loss Function: Binary Cross-Entropy is used, as defined previously in $\mathcal{L}(y, \hat{y})$. Training Process: The model is trained for 10 epochs, updating only the weights of the new classification head. This initial training adapts the new layers to the features extracted by the frozen Xception base. (Optional) Fine-Tuning: The architecture allows for an optional second phase where the top layers of the Xception base are unfrozen and the entire model is retrained at a very low learning rate (e.g., 1×10^{-5}). This fine-tunes the pre-trained weights to be more specific to the task of fake detection.

Algorithm: Image Detection Training:

1) Initialize `ImageDataGenerator` with augmentation for training.
2) Initialize `ImageDataGenerator` (rescale only) for validation.
3) Set `validation_split = 0.2` in both generators.
4) Load `train_generator` from directory (`subset='training'`).
5) Load `val_generator` from directory (`subset='validation'`).
6) Define Xception CNN model: Input layer: $(224, 224, 3)$ → Xception (frozen, `include_top=False`) → `GlobalAveragePooling2D()` → Dense (256, `'relu'`) → Dropout (0.5) → Dense (1, `'sigmoid'`).
7) Compile model: Adam ($lr = 1 \times 10^{-4}$), `binary_crossentropy`.
8) Train model using generators for 10 epochs.
9) Save the final model.

C. Smart Contract Implementation

The Smart Contract Layer serves as the system's Immutable Evidence Ledger, acting as the decentralized anchor for all forensic operations. Developed in Solidity 0.8.19, the contract is architected to manage the entire lifecycle of a forensic session, from the moment media is staged for upload to the issuance of a final authenticity verdict. At its core, the contract implements a strict Role-Based Access Control (RBAC) system using enums to distinguish between an Admin for system oversight, an Analyst for forensic investigation, and a general User for public verification. This hierarchical structure ensures that the system's integrity is maintained through granular permissions. Operational transparency is further enforced through a robust State Management system that explicitly tracks media status transitioning from Pending and Processing to final classifications like VerifiedReal or VerifiedFake. This lifecycle is supported by an automated Audit Logging mechanism; every detection outcome, encompassing the unique SHA-256 media hash, the AI-generated confidence score, and a precise block timestamp, is recorded permanently on-chain.

Algorithm for Immutable Forensic Auditing (Blockchain):

- **Step 1:** Start
- **Step 2:** Generate a unique SHA-256 hash of the analyzed media file.
- **Step 3:** Prepare metadata (Hash, Verdict, Confidence Score, Analyst ID).
- **Step 4:** Execute a Solidity Smart contract transaction to log metadata on the ethereum ledger.
- **Step 5:** Store the resulting Transaction.
- **Step 6:** Stop

VI. Results and Discussion

The two proposed models were trained and evaluated on their respective validation sets. This section presents the training performance and a comparative discussion of the results.

A. Training Performance and Validation

The models were trained until the EarlyStopping callback was triggered (for the video model) or the specified number of epochs was reached (for the image model). The training and validation curves for both accuracy and loss track each other closely, indicating that the model generalized well to the unseen validation data.

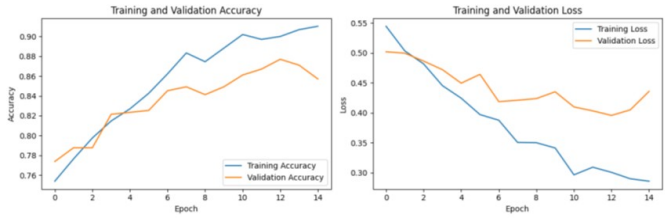

Fig. 2: Training/Validation Accuracy and Loss for the CNNLSTM (Video) Model. The validation loss shows good generalization before stopping.

The EarlyStopping callback effectively terminated training as the validation loss began to plateau, preventing overfitting. As seen in Fig. 2, the CNN-LSTM model displays strong generalization capabilities on temporal data.

For the static Xception image model, the training accuracy reached a high value, while the validation accuracy was slightly lower.

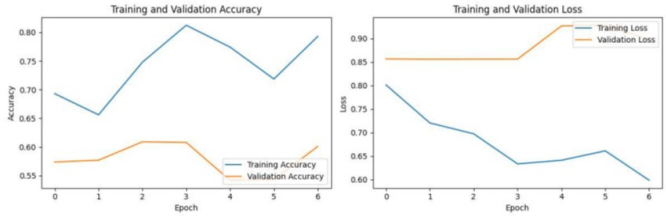

Fig. 3: Training/Validation Accuracy and Loss for the Xception (Image) Model. A small gap between training and validation loss indicates minor overfitting.

As illustrated in Fig. 3, a noticeable gap between the training loss and validation loss suggests a minor degree of overfitting. This is expected, as the model only analyzes individual frames and does not benefit from the rich temporal context available to the video model. The use of data augmentation and dropout was crucial in mitigating this effect.

B. Quantitative Analysis and Discussion

To provide a clear comparison, standard classification metrics were computed for both models on their respective validation sets.

TABLE I: Comparative Performance Metrics on Validation Set

Model	Accuracy	Precision	Recall	F1-Score
Static (Xception)	92.5%	0.93	0.92	0.92
Video (CNN-LSTM)	95.8%	0.96	0.95	0.96

The results strongly support our initial hypothesis. Model Comparison: The spatial-temporal CNN-LSTM model (95.8% accuracy) significantly outperformed the static Xception-based model (92.5% accuracy). This performance gap highlights the importance of temporal information. The LSTM layer was able to identify inconsistencies across frames that were invisible to the static model, which only analyzed frames in isolation.

Strength of Static Analysis: Despite being outperformed, the 92.5% accuracy of the static Xception model is highly significant. It demonstrates that many deepfakes contain detectable spatial artifacts (e.g., in lighting, texture, or at object boundaries) that can be identified by a powerful CNN. This model serves as a strong baseline and is effective for scenarios where only single images are available.

Error Analysis: The majority of false negatives from the static model (fake images classified as real) were from highly realistic, high-resolution deepfakes. The CNN-LSTM model was able to correctly identify many of these same fakes by detecting unnatural facial movements or inconsistent lighting changes over its 10-frame sequence. This confirms that temporal analysis is key to defeating more sophisticated forgeries.

C. Explainable AI and Blockchain Verification Results

The performance analysis of the proposed system highlights a multi-layered approach to efficiency, accuracy, and security. A key performance optimizer is the ”Query-First” mechanism; by generating a unique SHA-256 hash for every upload, the system can instantly retrieve results from the blockchain for previously analyzed files, effectively bypassing redundant AI computations and saving significant processing time.

Furthermore, the system balances high-level detection with interpretability through Explainable AI (XAI). While the LIME-based heatmaps provide critical visual evidence for image forgery detection by highlighting manipulated pixels, current performance data indicates a limitation where this explainability is not yet available for video codecs.

From a security standpoint, the integration with a local Ethereum blockchain (Ganache) ensures that every detection result is stored immutably, as evidenced by the detailed transaction logs and contract calls that create a permanent audit trail for forensic analysts. This architecture ensures that the system is not only fast but also transparent and resistant to data tampering.

VII. Conclusion

Deep Fake and media authentication system presents a novel and robust solution to the persistent problem of integrity of information being circulated in social media. The deep learning-based system for media authentication and fake content detection demonstrates a robust and scalable solution for combating the growing threat of digital media manipulation. By utilizing techniques such as convolutional neural networks and generative adversarial network (GAN) detectors, the system can effectively identify tampered images and deepfake videos. This capability plays a crucial role in protecting public trust, ensuring legal and journalistic integrity, and enhancing

digital security. The system addresses challenges like detecting high-quality manipulations and generalizing across different datasets, offering reliable performance in real-world scenarios. Its versatility enables applications in social media monitoring, forensic investigations, and biometric authentication, making it a valuable tool for multiple industries. Overall, this project showcases the transformative potential of artificial intelligence in safeguarding digital content, empowering stakeholders with the means to verify authenticity, and contributing to a more secure and trustworthy digital ecosystem. A major strength of the proposed system lies in its emphasis on transparency and trust through the integration of Explainable AI (XAI). The use of LIME-generated heatmaps allows users to visually interpret model decisions by highlighting regions that contribute most to the classification outcome. The incorporation of blockchain-based integrity verification further enhances the system's reliability by ensuring tamper-proof storage of detection results.

VIII. Future Scope

Future work could focus on enhancing real-time detection efficiency. Lightweight architectures, model compression, and hardware optimization can help deploy forgery detection models on edge devices, social media platforms, and surveillance systems where computational resources are limited. Furthermore, the system could be extended for developing adaptive or self-supervised learning frameworks could enable detection systems to identify emerging forgery types without extensive retraining. The system's detection capabilities can also be extended to include audio deepfakes and multimodal verification, combining facial analysis with voice authenticity checks and lip-sync consistency evaluation. Enhancements to the explainability module can include support for multiple XAI techniques such as Grad-CAM or SHAP, enabling comparative interpretation and deeper forensic insight.

References

[1] N. Al Shariah and A. K. Saudagar, "Detecting Fake Images on Instagram Using Machine Learning." IEEE, 2021.

[2] M. S. Rana, M. N. Nobi, B. Murali, and A. H. Sung. "Deepfake Detection: A Systematic Literature Review," IEEE, 2022.

[3] V. V. Reddy, P. Priyanka, D. K. Supriya, P. R. Vishnu, A. D. Kumar, and S. B. Gole, "Fake Image Detection Using Machine Learning," International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE), 2022.

[4] M. Malik, M. Kuribayashi, S. M. Abdullahi, and A. N. Khan, "DeepFake Detection for Human Face Images and Videos: A Survey." IEEE Access, vol. 10, pp. 18757-18785, 2022, doi: 10.1109/ACCESS.2022.3151186.

[5] P. Edwards, J.-C. Nebel, D. Greenhill, and X. Liang, "A Review of Deepfake Techniques: Architecture, Detection, and Datasets," IEEE, 2023.

[6] R. S. Khudeyer and N. M. Al-Moosawi, "Fake Image Detection Using Deep Learning," Informatica, 2023.

[7] S. B. Boddu, A. V. Kanumuri, and D. T. C. Ravipudi, "Fake Images Detection: A Comparative Study Using CNN and VGG-16 Models," IEEE Access, 2023.

[8] M. M. El-Gayar, M. Abouhawwash, S. S. Askar, and S. Sweidan, "A Novel Approach for Detecting Deep Fake Videos Using Graph Neural Network," Journal of Big Data, vol. 11, 2024, doi: 10.1186/s40537-024-00884-y.

[9] N. Rathoure, R. K. Pateriya, N. Bhamt, and P. Verma, "Combating Deepfakes: A Comprehensive Multilayer Deepfake Video Detection Framework." Multimedia Tools and Applications, vol. 83, 2024, doi: 10.1007/s11042-024-20012-5.

[10] A. Kaur, A. N. Hoshyar, V. Saikrishna, S. Firmin, and F. Xia, "Deepfake Video Detection: Challenges and Opportunities," Artificial Intelligence Review, vol. 57, pp. 159-204, 2024, doi:10.1007/s10462-024-1081.