International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

# Unmasking Reality: Advanced Deepfake Detection for Images and Videos

**Dr. S. Brindha[1], Ms. I. N. Sountharia[2], Mr. V. S. Thaneshvar[3], Mr. J. S. Jayanishanth[4], Mr. R. Rathivarman[5], Mr. R. Udaya Ganesh[6]**

[1]HoD, [2]Lecturer, [3, 4, 5, 6]Students

[1, 2, 3, 4, 5, 6]Department of Computer Networking, PSG Polytechnic College

**Abstract**

**Techniques for creating and manipulating multimedia information have progressed to the point where they can now ensure a high degree of realism. DeepFake is a generative deep learning algorithm that creates or modifies face features in a superrealistic form, making it difficult to distinguish between real and fake features. This technology has greatly advanced, promoting a wide range of applications in cinema, such as improving visual effects in movies, as well as various criminal activities, such as misinformation generation by mimicking famous people. To identify and classify DeepFakes, research in DeepFake detection using Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) has attracted increased interest. Essentially, DeepFake is regenerated media obtained by injecting or replacing some information within CNN and RNN models. This paper summarizes the DeepFake detection methods for face images and videos based on their results, performance, methodology used, and detection type. The challenges in generating a generalized DeepFake detection model are also analyzed.**

**Keywords: Deepfake, Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Resnet50, Long Short Term Memory (LSTM).**

## 1. INTRODUCTION:

Deepfakes are artificially generated media, such as images, videos, or audio recordings, created using deep learning algorithms. They combine existing content with new audio or video recordings, resulting in a fake but realistic-looking and sounding outcome. Deepfakes can be used for entertainment, education, or malicious purposes, such as spreading misinformation, identity theft, or reputation damage. deepfake technology advances, detecting and preventing their misuse becomes increasingly important. Deepfake detection for images and videos has become a crucial task in maintaining trust in digital media. To combat the spread of deepfakes, this paper employed deep learning-based approaches using Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). CNNs are effective in detecting manipulated images, while RNNs are suitable for analyzing temporal features in videos.

Methodologies such as two-stream networks, attention mechanisms, and multi-task learning have been utilized to improve detection accuracy. Despite promising results, challenges such as evasion attacks, generalizability, and explainability remain, highlighting the need for continued research in this field.

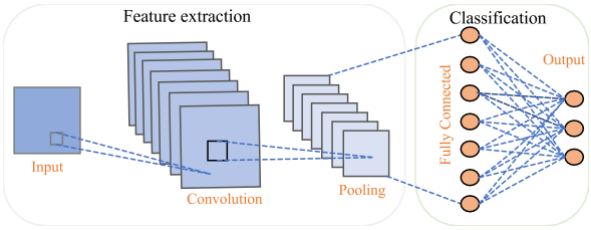
DeepFake images and videos increasingly pose risks to personal privacy and social security. Various methods have been proposed to detect manipulated images and videos. These methods can be categorized based on their approach:

1. General neural networks for classification tasks.
2. Temporal consistency features for detecting discontinuities between adjacent frames.
3. Visual artifacts generated during blending processes.
4. Fundamental features, such as camera fingerprinting and biological signals.


## 2. BACKGROUND:

Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are two fundamental architectures in deep learning. CNNs are designed to process spatial data, such as images, and are particularly effective in image recognition tasks, like facial recognition. RNNs, on the other hand, model temporal relationships between frames in videos, making them useful for video classification, activity recognition, and prediction tasks. While CNNs focus on spatial relationships within images and videos, RNNs focus on temporal relationships, highlighting their complementary strengths in image and video analysis.

### 2.1 CNN (Convolutional Neural Network)




The CNN or ConvNet is a special kind of deep-learning architecture that has gained much attention in computer vision and robotics. The initial idea of CNN, called *neocognitron*, was presented in 1979 by Kunihiko Fukushima, which later became known as the predecessor of CNN. Furthermore, the CNN architecture has been explained by Le-Cun *et al.* later, an improved version was explained in . A developed CNN network called LeNet-5 was found to be able to classify handwritten digits. Popular architectures from 2012 to 2015 are examined in, along with their basic components. The basic structure of the CNN model comprises three types of layers: convolutional, pooling, and fully connected. The

purpose of the convolution layer is to perform feature extraction. In the convolutional operation, an array of numbers (kernel) is applied across inputs (tensor) to construct the feature map. The procedure of constructing a feature map is an element wise product between each element of the kernel and the input tensor, and the outputs are summed to obtain the element of the kernel.

The kernel convolves across all the elements on the input tensor to construct the elements of the feature map for that kernel. An arbitrary number of feature maps can be obtained by implementing the convolution operation with different kernels. While training, the convolution operation is called forward propagation; during back propagation, the gradient descent optimization technique updates the learnable parameters (kernels and weights) according to the loss value.

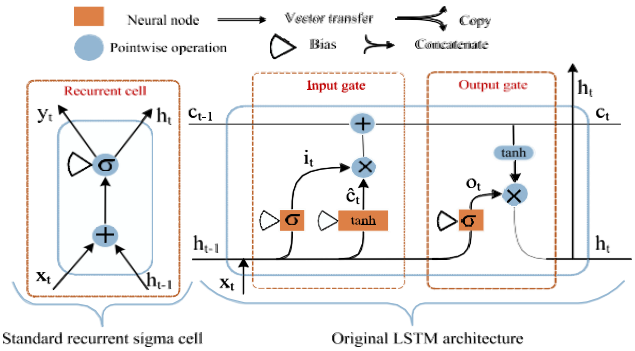
A pooling layer provides a typical down sampling operation to reduce the dimensionality of the feature maps to introduce translation invariance to small shifts and distortions and thereby decrease the number of subsequent learnable parameters. The pooling function is *pool*(·); for each feature *j* is a local neighborhood around location (*i, j*). The fully connected layers are the final outputs of the CNN, such as the probabilities for each class in classification tasks. The number of output nodes in the final fully connected layer is usually equal to the number of classes. A nonlinear function, such as ReLU, follows each fully connected layer. Finally, a loss function is calculated to assess the compatibility of the CNN's forward propagation output predictions with the provided ground truth labels.
The loss function for CNN optimization is given by**:**

$$L = \frac{1}{N}\sum_{n=1}^{N}(y_n \log(y_n) + (1-y_n)\log(1-y_n))$$

where N is the number of samples, $y_n$ is the actual label, and $\hat{y}_n$ is the predicted probability

Training a CNN determines the global minima, which identify the best-fitting set of parameters by minimizing the loss function. Currently, many CNN models exist, such as AlexNet , ZFNet, VGGNet , GoogLeNet/Inception and ResNet.

### 2.2 RNN ( Recurrent Neural Network )




An RNN is a neural network in which the output from the previous step is used as input in the next phase. All inputs and outputs in typical neural networks are independent of one another; however, in some situations, such as when predicting the next word of a phrase, the prior words are necessary, and therefore, the previous words must be remembered. Consequently, RNNs were created, which use a hidden layer to overcome the problem. The hidden state, which remembers certain information about a

sequence, is the most significant aspect of RNNs.

RNNs have a ‘‘memory’’ that stores all information about the calculations. This memory utilizes the same settings for each input since it produces the same outcome by performing the same job on all inputs or hidden layers. Unlike in other neural networks, this method minimizes the complexity of the parameters. When the gap between the relevant input data is large, Hochreiter and Schmidhuber proposed long short-term memory (LSTM) in 1997, which handles long-term dependencies. LSTM has been the focus of deep learning since it accomplishes nearly all the exciting outcomes based on RNNs. The recurrent layers, also known as hidden layers in RNNs, are made up of recurrent cells whose states are influenced by both previous states and current input via feedback connections.

LSTMs, introduced by Hochreiter and Schmidhuber in 1997, address long-term dependency issues in RNNs. The key LSTM operations are:

- **Forget Gate:** $f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$
- **Input Gate:** $i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$
- **Cell State Update:** $C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}$
- **Output Gate:** $o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$

## 3. DATASETS:

Forensics datasets can be classified into two broad types: traditional and DeepFake datasets. Traditional forensics datasets are created manually with extensive manual effort under carefully controlled conditions such as camera artifacts, splicing, inpainting, resampling and rotation detection. The Dresden Image Database (DID) is based on camera fingerprinting and consists of 14,000 images from 73 cameras. The 73 different cameras were of 25 different models and camera fingerprinting types (indoor and outdoor scenes). While most traditional datasets incorporate image alteration forensics, only some of them cover video-based manipulation forensics.For example, MICC-F220, MICC F2000, and MICC-F600 are image datasets used to detect copy-move modifications. MICC- F220 is composed of 110 tampered and 110 original images, MICC-F2000 is composed of 700 tampered and 1300 original images, and MICC-F600 is composed of 160 tampered and 440 original images. The IEEE Information Forensics and Security Technical Committee (IFS-TC) conducted the First Image Forensics Challenge (2013), which is an international competition that collected thousands of photographs of varied scenes, both indoors and outdoors, using 25 digital cameras. The Wild Web Dataset (WWD) contains 82 cases of 92 forgery variants and 101 unique mask splice detections. The WWD aims to address that gap in the evaluation of image tampering localization algorithms. The performance of [45] is evaluated in [60]. The CelebFaces Attributes Dataset (CelebA) is a large-scale face attribute dataset with more than 200K celebrity images, each with 40 attribute annotations. The images in this dataset cover large pose variations and background clutter. CelebA has large diversities, large quantities, and rich annotations, including 10,177 identities, 202,599 face images, 5 landmark locations, and 40 binary attribute annotations per image.

| Dataset | Type | Description | Key Features |
|---|---|---|---|

| | | | |
|---|---|---|---|
| **Dresden Image Database (DID)** | Traditional | Camera fingerprinting dataset with 14,000 images from 73 cameras, covering 25 models and both indoor and outdoor scenes. | 14,000 images, 73 cameras, 25 models |
| **MICC-F220** | Traditional | Dataset used for detecting copy-move modifications, composed of 110 tampered and 110 original images. | 220 images, tampered and original |
| **MICC-F2000** | Traditional | Dataset for copy-move modification detection, with 700 tampered and 1300 original images. | 2000 images, tampered and original |
| **MICC-F600** | Traditional | Used for detecting copy-move modifications, consisting of 160 tampered and 440 original images. | 600 images, tampered and original |
| **IEEE IFS-TC Challenge (2013)** | Traditional | International competition dataset with thousands of photographs collected using 25 digital cameras in varied indoor and outdoor scenes. | Thousands of photos, 25 cameras |
| **Wild Web Dataset (WWD)** | Traditional | Contains 82 cases of 92 forgery variants and 101 unique mask splice detections, aimed at evaluating image tampering localization algorithms. | 82 cases, 92 forgery variants, 101 splice detections |
| **CelebA** | Traditional | Large-scale face attribute dataset with 200K+ celebrity images, 40 binary attribute annotations, and pose variations and background clutter. | 202,599 images, 10,177 identities, 40 attributes |
| **WildDeepfake (WDF)** | DeepFake | Consists of 7,314 face sequences extracted from 707 DeepFake videos collected from the internet, aimed at testing DeepFake detectors. | 7,314 sequences, 707 videos |
| **OpenForensics (OF)** | DeepFake | Contains 115K unrestricted images with 334K human faces, used to detect DeepFake forgeries involving multiple faces in a single image. | 115K images, 334K faces |

**Table: 1 Comparison of Forensics Datasets for Image and Video Manipulation Detection**

## 4. PROPOSED SYSTEM

### 4.1 Proposed System Overview

The proposed system is designed to detect deepfake content by combining Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). For image-based detection, CNNs analyze

spatial features within individual frames to identify anomalies such as inconsistent textures or unnatural facial features. In video-based detection, the system integrates CNNs with RNNs to capture temporal dependencies across frames. The CNN component processes each frame to extract spatial features, while the RNN component, specifically utilizing Long Short-Term Memory (LSTM) units, analyzes the sequence of frames to detect temporal inconsistencies like unnatural facial movements or irregular blinking patterns. This hybrid approach ensures a comprehensive analysis by combining spatial and temporal features, thereby enhancing the accuracy of deepfake detection. The system undergoes a training phase where it learns to distinguish between authentic and manipulated content, followed by a testing phase to evaluate its performance. By integrating these methodologies, the proposed system provides a robust solution for identifying deepfake content in both images and videos.
Equations in the Proposed System

1. Convolution Operation in CNNs:

The convolution operation applies a filter (kernel) to the input image to extract features. Mathematically, this is represented as:

$$(\boldsymbol{I} * \boldsymbol{K})(\boldsymbol{i}, \boldsymbol{j}) = \sum \sum \boldsymbol{I}(\boldsymbol{m}, \boldsymbol{n}). \boldsymbol{K}(\boldsymbol{i} - \boldsymbol{m}, \boldsymbol{j} - \boldsymbol{n})$$

$$\boldsymbol{m} \quad \boldsymbol{n}$$

where I is the input image, Kis the kernel, and (i,j) denotes the position in the output feature map.

2. Activation Function (ReLU):

After convolution, the Rectified Linear Unit (ReLU) activation function is applied to introduce non-linearity:

$$\boldsymbol{ReLU}(\boldsymbol{x}) = \boldsymbol{max}(\boldsymbol{0}, \boldsymbol{x})$$

3. Pooling Operation:

Pooling layers reduce the spatial dimensions of the feature maps. For max pooling:
Where F is the input feature map, and P is the pooled feature map.

$$\boldsymbol{P}(\boldsymbol{i}, \boldsymbol{j}) = \max_{(m,n) \in pool\_region} \boldsymbol{F}(\boldsymbol{m}, \boldsymbol{n})$$

Where F is the input feature map, and P is the pooled feature map.

4. Flattening and Fully Connected Layers:

The pooled feature maps are flattened into a vector and passed through fully connected layers:

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{W}.\boldsymbol{x} + \boldsymbol{b})$$

Where $x$ is the input vector, W is the weight matrix, b is the bias vector, and $f$ is the activation function (e.g., softmax or sigmoid).

5. Sequence Modeling with LSTM in RNNs:

For video analysis, the system uses LSTM units to capture temporal dependencies. The LSTM updates are defined by:

$$f_t = \sigma(W_f . [h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i . [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = tanh(W_c . [h_{t-1}, x_t] + b_C$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \bar{C}_t$$

$$o_t = \sigma(W_o . [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \odot tanh(C_t)$$

where $f_t$ is the forget gate, $i_t$ is the input gate, $\tilde{C}_t$ is the candidate cell state, $C_t$ is the cell state, $o_t$ is the output gate, $h_t$ is the hidden state, $x_t$ is the input at time step t, W denotes weight matrices, b denotes biases, σ is the sigmoid function, and ⊙ represents element-wise multiplication.

6. Loss Function:

The system uses a loss function to measure the difference between the predicted output and the actual label. For binary classification, the Binary Cross-Entropy Loss is used:

$$L = -\frac{1}{N}\sum_{n=1}^{N} [y_n \log(y_n) + (1 - y_n) \log(1 - y_n)]$$

Where:

- L is the overall loss value.
- N is the number of samples in the dataset.
- $y_n$is the true label for the n-th sample (either 0 or 1).
- $\widehat{y_n}$ is the predicted probability of the positive class for the n-th sample.
- log $\widehat{y_n}$ calculates the logarithm of the predicted probability.
- $1 - \widehat{y_n}$ represents the negative class (if the true label is 0).
- log( $1 - \widehat{y_n}$ ) calculates the logarithm of the probability of the negative class.
-

**4.2 WORKING of Proposed System:**

**4.2.1 Deepfake Detection Using CNNs (Image Analysis):**

CNNs are highly effective for detecting fake images because they are designed to automatically learn spatial hierarchies of features. Here's a detailed process for using CNNs in image-based deepfake detection:

**4.2.2 Input Image Preprocessing:**

The first step is to preprocess the input image (deepfake or real). This typically involves resizing the image to a fixed size, normalizing the pixel values, and possibly augmenting the data (rotating, flipping) to prevent overfitting. Example: Resize the image to 224x224 pixels, which is a common input size for models like ResNet, Inception, or Xception.

**4.2.3 Feature Extraction via Convolutional Layers:**

The CNN applies convolutional layers to the image. Each convolution operation uses a filter or kernel to extract low-level features (like edges, textures, and colors). Multiple filters are applied at different layers, gradually extracting more complex features (like shapes, eyes, or mouth) as the network goes deeper.
Example of filters in CNN layers:

- Early layers: detect simple features like edges.
- Middle layers: may detect more complex features like eyes, noses, and facial contours.
- Deeper layers: focus on the overall structure or patterns like the texture of a face and potential artifacts introduced by the manipulation.

**4.2.4 Pooling Layers:**

Max pooling or average pooling layers are used after convolution layers to down- sample the image while preserving important features. This helps reduce computational complexity.
- Pooling also makes the network invariant to small translations and distortions, which is crucial for detecting deepfake artifacts.

**4.2.5 Fully Connected Layers:**

After several convolution and pooling layers, the network has a set of feature maps. These are flattened into a vector and passed through one or more fully connected layers to perform the classification.
These layers act as classifiers, determining whether the image is real or fake based on the features extracted by the convolutional layers.

**4.2.6 Output Layer:**

The final output layer is typically a softmax layer (for multi-class classification) or a sigmoid layer (for binary classification, i.e., real vs fake). If the model detects visual artifacts, inconsistent lighting, or unnatural face structure, it classifies the image as a deepfake.
**Example of CNN Architectures for Deepfake Detection are** XceptionNet: Known for its depth and ability to extract fine-grained features, ResNet50: Uses residual connections to avoid vanishing gradients, effective in deep networks, VGG16/VGG19: Known for simplicity and effectiveness in image classification tasks.

**4.3 Deepfake Detection Using CNNs and RNNs (Video Analysis):**

Deepfake detection becomes more complex because it has to consider temporal dependencies between frames. For this, CNNs are used to extract spatial features from each frame, while RNNs (specifically LSTMs or GRUs) analyze the sequence of frames over time to capture the motion and temporal patterns.

**4.3.1 Frame Extraction (CNN for Each Frame):**

A video is broken down into individual frames. For each frame, the CNN processes the image to extract spatial features (like we described in the image detection process).Each frame is passed through a CNN (such as ResNet or Xception) to extract its visual features.

**4.3.2 Sequence Modeling with RNN (LSTM or GRU):**

Once spatial features are extracted from all frames, these features are passed to an RNN, particularly an LSTM (Long Short-Term Memory) network or GRU (Gated Recurrent Unit). The RNN analyzes the temporal sequence of frames to detect inconsistencies in facial motion, blinking patterns, or lip sync that are indicative of a deepfake. The network can capture patterns such as unnatural transitions between frames, incorrect facial movements, or inconsistencies in background motion.

**4.4 Example RNN Use Cases:**

1.Facial Movements: The RNN can track facial expressions (such as blinking and mouth movement) over time, detecting any unnatural behaviors that might arise from deepfake creation. Audio-Visual Synchronization: If you have audio input, RNNs can check whether the lip movements in the video match the audio correctly, as deepfakes may have mismatched audio and video.

2. Integration of CNN and RNN: The output from the CNN is treated as a feature vector for each frame, and these vectors are passed into the RNN. The RNN captures the temporal aspect of the video sequence and can spot subtle inconsistencies that happen over time, like unnatural head movements or artifacts in the generated video.

3. Final Classification: After processing the frames through the RNN, the output can be fed into a final fully connected layer that performs classification. This classification layer predicts whether the video is real or fake based on both the spatial features (from CNN) and temporal patterns (from RNN).

**4.5 Example of CNN and RNN Model for Deepfake Video Detection:**

Input: A sequence of video frames (e.g., 16-32 frames).
CNN Stage: Each frame is processed by a CNN (e.g., Xception) to extract spatial features. RNN Stage: The extracted features are passed to an RNN (LSTM or GRU) that learns the temporal dependencies.
Output Layer: A final dense layer classifies the sequence as real or fake based on both spatial and temporal features

## 5. CHALLENGES FOR CREATION DEEPFAKE and DETECTION OF DEEPFAKE:

### 5.1 CHALLENERS FOR CREATION:

Despite the fact that significant efforts have been made to increase the visual quality of created DeepFakes, there are still a number of hurdles to overcome. Some challenges related to creating DeepFakes include generalization, temporal coherence, illumination stipulations, lack of realism in eyes and lips, hand movement behavior and identity leakage.

**Generalization:** The characteristics of generative models depend on the type of dataset provided during training. Therefore, after finishing training on a particular dataset, the output produced by the model reflects the learned characteristics (fingerprint). In addition, the output quality depends on the size of the dataset provided during training.

**Temporal coherence:** Other flaws include visible abnormalities such as flickering and jittering between frames. These flaws occur because the DeepFake generation frameworks work on each frame without considering temporal consistency. To overcome these flaws, some researchers offer this context to the generator or discriminator, consider temporal coherence losses, use RNNs, or use a combination of these approaches.

**Illumination stipulations:** Most available DeepFake datasets are produced in a controlled environment, such as using the same type of lighting and background. However, a sudden shift in lighting circumstances in indoor/outdoor scenarios causes color discrepancies and odd abnormalities in the resultant output.

**Lack of realism in eyes and lips:** The lack of natural emotions, interruptions, and the rate at which the target talks are the primary difficulties of eye and lip synchronization-based DeepFake creation

### 5.2 CHALLENGES FOR DETECTION:

Although significant progress has been achieved in the performance of DeepFake detectors, several issues related to the current detection algorithms need to be addressed. Some of the difficulties faced by DeepFake detection techniques include a lack of datasets, unknown types of attacks on media, temporal aggregation and unlabeled data.

**Artifacts in Deepfakes:** While CNNs are good at detecting artifacts in images, high- quality deepfakes can be difficult to distinguish from real images due to better generation techniques (e.g., GANs).

**Generalization Across Datasets:** A model trained on a specific dataset may not generalize well to another dataset, as the deepfake generation methods vary.

**Real-time Processing:** For video deepfake detection, real-time processing of large video files can be computationally expensive. Using advanced hardware (GPUs) and optimizations in the model architecture is crucial.

**Unknown type of attack:** Another challenging task is to design a robust DeepFake detection model against unknown types of attacks such as the fast gradient sign method (FGSM) [129] and the Carlini and Wagner L2 norm attack (CW-L2) [130]. These attacks are used to fool classifiers in their actual output. An example of a DeepFake creation using source and target faces, with adversarial perturbations. DeepFakes are accurately classified as fake by a DeepFake detector, but adversarially perturbed DeepFakes are classified as real.

**Temporal Aggregation:** Existing DeepFake detection algorithms use binary frame- level classification, which involves determining whether each video frame is real
or fake. However, as these methods do not take interframe temporal consistency into consideration, they may encounter issues, such as exhibiting temporal abnormalities and
real/artificial frames occurring in consecutive intervals. Furthermore, these methods necessitate an extra step to compute the video integrity score, which must be integrated for each frame to obtain the final result.

**Future work:**

- The proposed system is expected to achieve high detection accuracy for both images and videos.
- It achieves to allow the system or mobile to detect the real or fake image or video. If real it allows the image or video to use by the user. When the image or video is fake it automatically deletes it from the system or mobile.

**CONCLUSION:**

Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are two powerful deep learning architectures that have revolutionized the field of image and video analysis. CNNs, with their ability to excel in spatial analysis, have become the go-to architecture for tasks such as image classification, object detection, and image segmentation. On the other hand, RNNs, with their ability to model temporal relationships, have proven to be highly effective in tasks such as video classification, activity recognition, and language modeling. The complementary strengths of CNNs and RNNs have enabled the development of various applications, including self-driving cars, facial recognition systems, and video surveillance systems. Furthermore, the continued advancement of CNNs and RNNs is expected to lead to even more innovative applications in the future, making them a crucial part of the rapidly evolving field of artificial intelligence.CNNs are excellent for image-based deepfake detection, focusing on spatial features. RNN (especially LSTMs) combined with CNNs are powerful for video deepfake detection, as they handle temporal features (movement across frames) and spatial features (from individual frames).

Hybrid CNN + RNN models can provide high accuracy in detecting both subtle visual anomalies and temporal inconsistencies, which is crucial for identifying deepfake images and videos. By leveraging these deep learning models, we can effectively detect deepfakes, though the technology is continuously evolving, so ongoing advancements are necessary to keep up with new techniques.

**REFERENCE:**

[1] H. Farid, ‘‘Image forgery detection,’’ *IEEE Signal Process. Mag.*, vol. 26, no. 2, pp. 16– 25, Mar. 2009.

[2] L. Verdoliva, ‘‘Media forensics and DeepFakes: An overview,’’ *IEEE J. Sel. Topics Signal Process.*, vol. 14, no. 5, pp. 910–932, Aug. 2020.

[3] S. Agarwal, H. Farid, O. Fried, and M. Agrawala, ‘‘Detecting deep- fake videos from phoneme-viseme mismatches,’’ in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*

[4] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, ‘‘Star- GAN: Unified generative adversarial networks for multi-domain image- to-image translation,’’ in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*

[5] S. Dong, P. Wang, and K. Abbas, ‘‘A survey on deep learning and its applications,’’

[6] A. Creswell and A. A. Bharath, ‘‘Inverting the generator of a generative adversarial network,’’ *IEEE Trans. Neural Network.*

[7] D. Guera and E. J. Delp, ‘‘Deepfake video detection using recurrent neural networks,’’ in *Proc. 15th IEEE Int. Conf. Adv. Video Signal Based Surveill. (AVSS)*

[8] L. Guarnera, O. Giudice, and S. Battiato, ‘‘DeepFake detection by analyz- ing convolutional traces,’’ in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW),*

[9] Wang et al, “Deepfake Detection using Residual Networks”.aug 2020,

[10] Malik et al, “Evaluating face image and video deepfake techniques”. 2020,

[11] Dr. S. Brindha, Ms. I.N. Sountharia, Mr. S. Dinakar, Mr. K. Mohanaprasad and Mr. M. Manusanjay,“Steganographic synergy: AES scrambling, FHSS embedding, and VVC compression for video concealment” Volume:06/Issue:03/March-2024