




Hybrid Convolutional-Recurrent Neural Networks (CNN-RNN) Model with Temporal Attention and Particle Swarm Optimization for Deepfake Video Detection

Jeremias C. Esperanza¹, Jean Fidelio E. Marquez², Ron Anthony A. Sy³

College of Informatics and Computing Studies, New Era University, Quezon City, 1107, Philippines^{1,2,3}

✉ jcesperanza@neu.edu.ph; jeanfidelio.marquez@neu.edu.ph
ronanthony.sy@neu.edu.ph

RESEARCH ARTICLE INFORMATION	ABSTRACT
<p>Received: April 10, 2025 Reviewed: May 4, 2025 Accepted: June 19, 2025 Published: June 30, 2025</p> <p> Copyright © 2025 by the Author(s). This open-access article is distributed under the Creative Commons Attribution 4.0 International License.</p>	<p>The rapid advancement of deepfake technology presents a growing threat to information integrity and online security. To address this, this research proposed an efficient deepfake video detection framework that integrates Convolutional Neural Networks (CNNs) for spatial feature extraction, Recurrent Neural Networks (RNNs) with a temporal attention mechanism for modeling sequential dependencies, and Particle Swarm Optimization (PSO) for hyperparameter tuning. The pipeline included frame extraction, face alignment, and feature processing using a pre-trained CNN, followed by an RNN that emphasizes critical temporal artifacts through attention. PSO further enhanced model performance by optimizing key hyperparameters such as learning rate and hidden dimensions. To evaluate the effectiveness of the proposed model, a comparative analysis against existing deepfake detection methods, including XceptionNet, LSTM with frame-level features, and CNN-GRU without attention, was conducted. The proposed CNN-RNN model with Temporal Attention and PSO outperformed the baselines, demonstrating the model's improved generalization and reliability, particularly in reducing false negatives, making it a robust solution for real-world media forensics and platform integrity.</p>

Keywords: *Deepfake video detection, hybrid machine learning, Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), temporal attention, Particle Swarm Optimization (PSO)*

Introduction

The rapid advancement of machine learning has made it increasingly easy to generate convincing fake videos, commonly known as deepfakes. These manipulated videos, which often involve replacing a person's face with someone else's seamlessly, pose a serious threat to privacy, security, and public trust. Rahman et al. (2022) emphasized that deepfakes can appear highly realistic, deceiving even trained observers. While some applications of this technology are intended for entertainment or educational purposes, it has also been exploited for malicious activities such as identity theft, cyberbullying, misinformation campaigns, and non-consensual content creation.

Deepfake detection has become a critical area of research in response to these threats. Early works, such as MesoNet by Afchar et al. (2018), demonstrated the feasibility of using compact Convolutional Neural Networks (CNNs) to detect manipulated facial features. However, as Deepfake algorithms continue to evolve, more sophisticated detection approaches are required.

A distinguishing characteristic of videos, as highlighted by Yu et al. (2021), is their temporal continuity. Unlike images, videos consist of sequences of frames with strong temporal correlation. Deepfakes often disrupt this consistency, creating artifacts such as flickering or misaligned facial features. Güera and Delp (2018) leveraged this by employing Recurrent Neural Networks (RNNs) to model temporal patterns and detect such anomalies. Building on this, researchers have explored combining CNNs with RNNs, integrating attention mechanisms and optimization techniques to enhance performance.

Hybrid models that integrate CNNs for spatial feature extraction and RNNs for temporal modeling have shown promising results (Al-Adwan et al., 2023; Chen et al., 2020). The inclusion of a temporal attention mechanism allows the model to focus on discriminative frames, improving its ability to detect subtle manipulations (Gao et al., 2021; Yan et al., 2020). Furthermore, Particle Swarm Optimization (PSO)—a population-based optimization algorithm inspired by the social behavior of birds (Kennedy & Eberhart, 1995)—has been successfully used to fine-tune hyperparameters in deep learning models, leading to improved generalization and performance (Cunha et al., 2024; Shami et al., 2022).

In this study, the researchers presented a unified deepfake detection framework that integrates CNNs, RNNs with a temporal attention mechanism, and PSO. These components were designed to work in tandem: CNNs extract spatial features from each frame, RNNs capture temporal dependencies across frames, the attention mechanism identifies key temporal cues indicative of manipulation, and PSO optimizes model parameters to boost detection accuracy. The model was trained and evaluated using the Celeb-DF dataset, with performance assessed through standard metrics such as accuracy, precision, recall, and F1-score.

To ensure real-world applicability, the final model was deployed in a web-based application that enabled users to identify potentially manipulated videos. This research aimed to contribute to a more secure digital environment by advancing the capabilities of deepfake detection through the synergy of deep learning and optimization techniques.

Methods

Project Design

Figure 1 shows that the project adopted an experimental design implemented in three sequential phases: (1) Data Preprocessing, (2) Model Development, and (3) System Implementation and Deployment. This phased design allowed for modular testing and optimization of each component before integration. The goal was to create a reliable deepfake video detection model using a hybrid architecture enhanced by temporal attention and optimized using Particle Swarm Optimization (PSO).

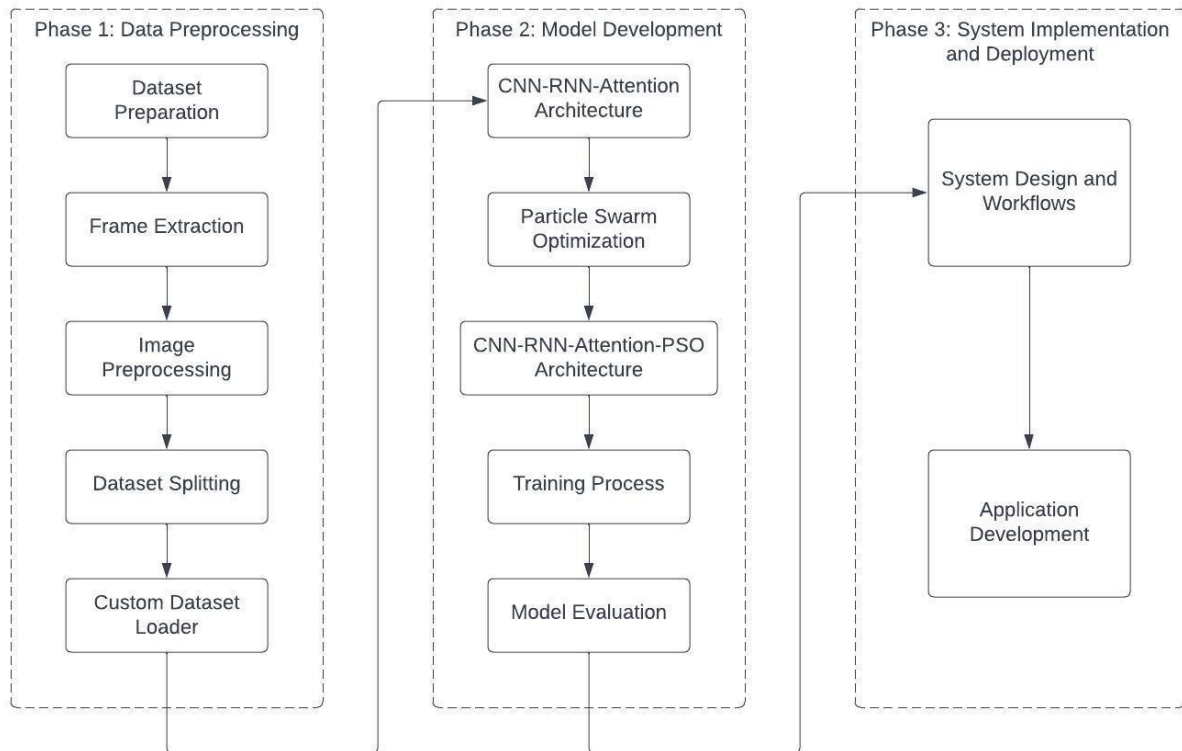


Figure 1. Project Design

Phase 1: Data Preprocessing

Dataset Preparation. The Celeb-DF v2 dataset was selected as the benchmark dataset for this study due to its diverse and high-quality content. It consists of 5,639 real videos collected from 59 YouTube celebrities (Celeb-real and YouTube-real) and 5,639 deepfake videos (Celeb-synthesis) generated using various synthesis models. The dataset is publicly available and widely accepted for research purposes, making it suitable for evaluating deepfake detection systems. Each video in the dataset was labeled as either real (1) or fake (0).

To ensure balanced class distribution across training, validation, and testing phases, real video samples were duplicated. The final dataset was split into 7,265 training, 1,557 validation, and 1,557 testing videos, with equal numbers of real and deepfake videos in each subset.

Frame Extraction. To extract frames from each video, the researchers used the OpenCV library (cv2) in Python. Specifically, five uniformly spaced frames were extracted per video to balance between computational efficiency and temporal information retention. The total frame count of each video was first determined, and the frame indices were calculated using the formula:

$$interval = \max(\lfloor \frac{total\ frames}{5} \rfloor, 1)$$

Equation 1. Frame Extraction Formula

This ensures the selected frames were evenly distributed across the video's timeline. If a video contained fewer than five frames, zero tensors (black images) were padded to maintain consistent input dimensions. Below is the summary of the extraction procedure:

1. Load the video using cv2.VideoCapture.
2. Compute the total number of frames.
3. Determine five equally spaced frame indices using the formula above.
4. Extract frames at these indices using cap.set(cv2.CAP_PROP_POS_FRAMES, index) and cap.read().
5. If fewer than five frames are available, pad with zero tensors.

Image Preprocessing. Each extracted frame underwent a series of image processing steps implemented using the torchvision.transforms library. The complete transformation pipeline was as follows:

1. *Convert to PIL Image:* Raw frames extracted as NumPy arrays via OpenCV were converted to PIL image format to enable compatibility with PyTorch transforms.
2. *Resize:* Each frame was resized to 224×224 pixels to ensure uniformity across all inputs and to match the input dimension required by the pre-trained CNN backbone.
3. *ToTensor:* Resized images were converted into PyTorch tensors, scaling pixel values to the range $[0,1][0,1]$.
4. *Normalize:* Tensors were normalized using the ImageNet mean $[0.485, 0.456, 0.406]$ and standard deviation $[0.229, 0.224, 0.225]$, aligning the input distribution with pre-trained model expectations and improving training stability.

This transformation pipeline was applied consistently across all training, validation, and testing data, and is reproducible using the following transform definition:

```
transform = transforms.Compose([transforms.ToPILImage(),
                               transforms.Resize((224, 224)), transforms.ToTensor(),
                               transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229,
0.224, 0.225])])
```

Dataset Splitting. The processed video dataset was divided into 70% for training, 15% for validation, and 15% for testing using the train_test_split function from scikit-learn, ensuring that class distributions remained balanced across the splits.

Custom Dataset Loader. A PyTorch Dataset class (VideoDataset) was created to load the video frames and labels. DataLoaders with custom `collate_fn` functions were used for batching sequences of frames during training and evaluation.

Phase 2: Model Development

During this stage, a tailored neural network was designed and trained specifically for deepfake detection. The model integrated both spatial and temporal analysis through an attention-based architecture and was further enhanced using Particle Swarm Optimization (PSO).

CNN-RNN-Attention Architecture. The proposed model was built on a hybrid architecture of Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM) units, and a Temporal Attention Mechanism. With ReLU activation and max pooling, the CNN module's two convolutional layers are intended to extract pertinent spatial features from every frame. These features were subsequently flattened and input into a bidirectional LSTM, allowing the model to learn temporal dependencies across frames. Following this, a temporal attention mechanism was introduced to assign dynamic weights to each LSTM output, permitting the model to focus on significant frames. In the final stage, a fully connected layer interprets the attention-weighted context vector—derived from the LSTM outputs—to perform binary classification, determining whether the video content is real or artificially generated.

Particle Swarm Optimization. In order to enhance the performance of the CNN-RNN-Attention model, Particle Swarm Optimization (PSO) was applied in order to automatically discover the optimal values of two very important hyperparameters: the learning rate and the number of units in the GRU layer. PSO is a population-based metaheuristic search inspired by the behavior of nature-inspired swarms. Every "particle" in PSO explores the solution space by changing its position based on its own memory and the overall knowledge of the swarm.

In this research, PSO aimed to reduce training loss, which was defined as the objective (fitness) function. Each particle, representing a unique pairing of learning rate and GRU dimension, initiated a new model configuration trained over 10 epochs using the Adam optimizer and a cross-entropy loss function. The resulting loss value for each configuration served as the evaluation metric for that particle.

The hyperparameter search was conducted within predefined bounds: the learning rate ranged from 0.0001 to 0.01, and the GRU hidden layer size varied between 64 and 256 units. The PSO process consisted of 10 particles over 5 iterations, with behavioral parameters set as follows: cognitive component ($c1$) = 0.5, social component ($c2$) = 0.3, and inertia weight (w) = 0.9. Leveraging the `pyswarms` Python library, the algorithm iteratively adjusted particle positions to identify the most effective configuration. The final set of optimal hyperparameters was then adopted for the model's full training process.

Overall Architecture of the CNN-RNN-Attention-PSO Framework. Figure 2 shows how the proposed framework integrates a hybrid deep learning model with a metaheuristic optimization technique to effectively detect deepfake videos. The architecture consists of two main components: (1) the CNN-RNN-Attention classification model, and (2) the Particle Swarm Optimization (PSO) module used for hyperparameter tuning.

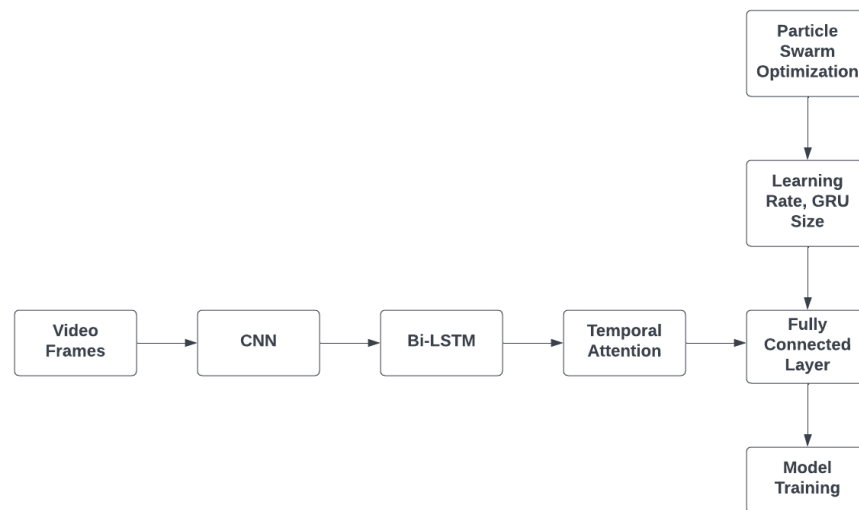


Figure 2. Overall Architecture of the CNN-RNN-Attention-PSO Framework

In the core model, CNNs extracted spatial features from individual video frames using two convolutional layers with ReLU activation and max pooling. These spatial features were flattened and passed into a Bidirectional Long Short-Term Memory (LSTM) network to capture temporal dependencies. A Temporal Attention Mechanism was applied to the LSTM outputs, dynamically weighting each time step to emphasize the most informative frames. Finally, the attention-weighted output was fed into a fully connected layer to perform binary classification, distinguishing between real and fake video content.

To optimize model performance, Particle Swarm Optimization (PSO) was employed to search for the best values for two critical hyperparameters: the learning rate and the number of GRU units in the recurrent layer. PSO operates by simulating a swarm of particles, each representing a unique hyperparameter configuration. The particles explored the search space iteratively, guided by individual and collective experience, to minimize the model's training loss. The best-performing configuration was then used for full model training.

This integrated approach ensured that both model architecture and hyperparameter selection contribute synergistically to maximizing classification accuracy and generalization performance.

Training Process. The model was developed using a supervised learning strategy, employing cross-entropy loss as the objective function—ideal for binary classification problems like deepfake detection. Training was carried out across several epochs, where each batch consisted of input frames paired with their corresponding labels. These inputs were initially moved to the GPU, and the forward pass was executed using mixed precision via PyTorch's `torch.amp.autocast`, enabling operations in float16 format to improve efficiency and reduce memory consumption. To maintain numerical stability during backpropagation, the GradScaler tool was used to scale the loss before applying gradient descent. An externally defined optimizer handled the parameter updates. During training, the model's accuracy was monitored by tracking the number of correct predictions.

The model was evaluated on the validation dataset after each training epoch, with gradient updates turned off to minimize computational overhead. During this phase, predictions were generated for each batch, and both loss and accuracy were calculated. Key metrics, including learning rate, validation accuracy, validation loss, training loss, and training accuracy, were measured at the conclusion of each epoch. These ongoing assessments provided vital information about the model's convergence and performance, allowing for better training management and more informed modifications.

Model Evaluation. Once the optimal hyperparameters were determined through PSO, the finalized CNN-RNN-Attention model was tested on an independent dataset to evaluate its classification accuracy. This evaluation was conducted using a specific function that ran the model in inference mode, with gradient calculations turned off to improve computational efficiency.

During evaluation, batches of unseen video data were passed through the model to generate output logits. These outputs were then converted into class probabilities using a softmax activation function. Predictions were determined by selecting the highest probability class, and all predicted labels, true labels, and probability scores were stored for further metric computation. Several performance metrics were used to ensure a comprehensive assessment. The overall correctness of predictions was measured by accuracy. The formula is shown below:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Equation 2. Accuracy Formula

Although it provides a general overview of model performance, it may present misleading results, specifically in cases of class imbalance. Therefore, additional evaluation metrics were employed to deliver a more comprehensive analysis. To obtain a better understanding, metrics like F1-score, precision, and recall were computed. Precision refers to the ratio of correctly predicted positive instances to all predicted positives. This is especially critical in reducing false positives since wrongly tagging genuine videos as deepfakes can cause negative impacts. The accuracy formula is given as:

$$Precision = \frac{TP}{TP + FP}$$

Equation 3. Precision Formula

Recall evaluates how effectively the model can detect all true positive cases. In deepfake detection, it plays a vital role in ensuring that manipulated videos are accurately flagged and not incorrectly labeled as authentic, which could undermine the reliability of the detection system. The formula used to compute recall is:

$$Recall = \frac{TP}{TP + FN}$$

Equation 4. Recall Formula

The F1-score shows the average of recall and precision, combining both into a unified measure. It is especially beneficial in situations where reducing both false negatives and false positives is equally critical. Equation 4 shows the score formula:

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Equation 5. *F1-Score Formula*

These metrics were calculated using a weighted average to account for any class imbalance between real and deepfake videos. A detailed classification report further analyzed the performance of the model across each class.

All evaluation outputs were saved in a designated directory, ensuring that the results could be reviewed, validated, and included in the final project documentation. This systematic evaluation process helped confirm the model's strengths, revealed areas for further enhancement, and emphasized the importance of using multiple performance metrics in deepfake detection.

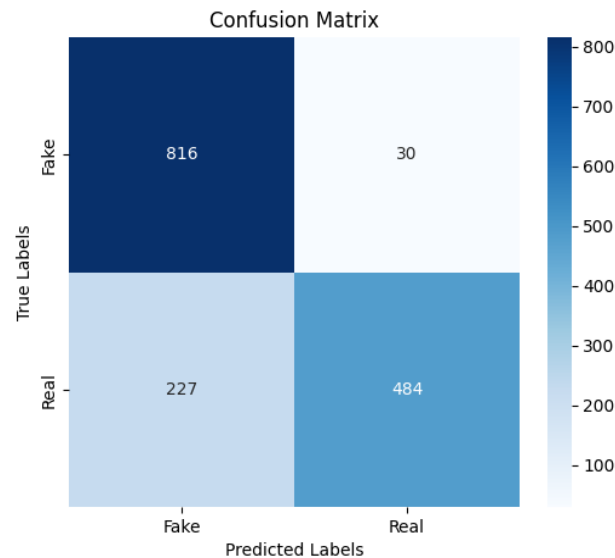


Figure 3. *Confusion Matrix*

Figure 3 shows the confusion matrix that displays the distribution of true versus predicted labels to support interpretability and transparency; various visualizations were generated and saved. A confusion matrix shows a clear overview of the performance of the model in classifying real and fake videos. Of all the fake videos, 30 were mistakenly identified as real (false negatives) and 816 were correctly identified as fake (true negatives). On the other hand, out of the real videos, 484 were correctly identified as real (true positives), whereas 227 were incorrectly classified as fake (false positives).

This indicates that the model demonstrates strong performance in detecting fake content, with very few false negatives. However, there is a notable number of false positives, meaning some real videos are being mistakenly flagged as fake. This may

suggest a slightly conservative bias toward classifying videos as fake, possibly as a precautionary response to potential deepfake threats. Overall, the confusion matrix reflects a well-performing model with high true classification rates for both classes. The relatively higher number of false positives compared to false negatives could be addressed with further tuning or through post-processing to improve real-video recognition without compromising the detection of deepfakes.

Phase 3: System Implementation and Deployment

After model development and evaluation, the trained model was prepared for deployment to serve as a backend system for real-time deepfake detection.

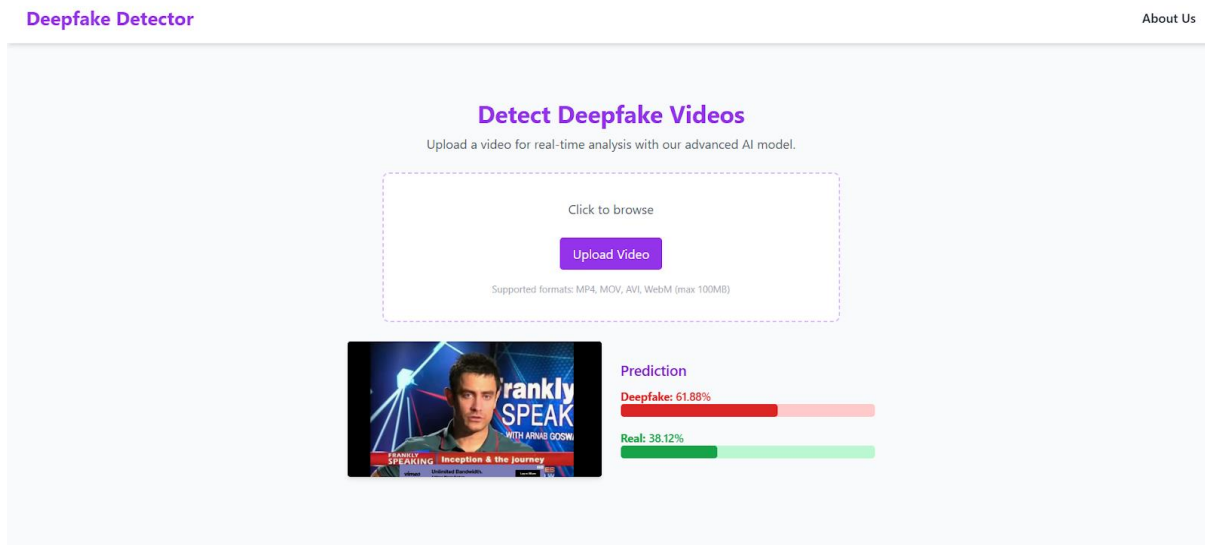


Figure 4. Web-Based User Interface

Figure 4 shows that the web-based interface of the system serves as the main access point for users to interact with the deepfake detection model. It was designed to allow a user to upload a video file directly from their local device. Upon selecting a video, the system immediately begins processing without requiring any further input, ensuring a straightforward and uninterrupted user experience.

Once a video is uploaded, the system performs preprocessing in the background. This includes extracting individual frames from the video, which are essential for frame-wise analysis by the deep learning model. The preprocessing step is automated and hidden from the user to maintain the simplicity of the interface while ensuring the input is correctly prepared for inference.

After preprocessing and model inference, the interface displays a video preview along with two prediction bars. These bars show the confidence scores for both the “Deepfake” and “Real” classes. Each bar is color-coded and labeled with the corresponding percentage, providing users with a quick visual understanding of the model’s assessment of the uploaded video.

System Design and Workflows. The system initiates a structured workflow to process and classify the input. First, the video is received by the backend server and

saved temporarily. The system then extracts individual frames from the video using OpenCV, and each frame undergoes preprocessing, including resizing and normalization, to prepare it for model inference. These frames are passed through the trained CNN-RNN-Attention model, which performs spatial and temporal analysis. After processing all frames, the model aggregates the results and returns a confidence score for both the "Deepfake" and "Real" classes. These results are then displayed on the user interface as visual prediction bars, along with a preview of the uploaded video.

Application Development. The web application was developed using a combination of frontend and backend technologies. The backend was built using Flask, which served as the core framework for handling file uploads, preprocessing, and model inference. The deep learning model was implemented in PyTorch, while OpenCV (cv2) was used for extracting frames from uploaded videos. On the frontend, the interface was styled using Tailwind CSS to ensure a responsive and clean layout. JavaScript was integrated for dynamic behavior, such as previewing the uploaded video and updating prediction results in real time. This combination of technologies allowed for a lightweight yet functional application capable of handling real-time video analysis.

Ethical Considerations

This study adhered to rigorous ethical standards to ensure the protection of data, integrity of the research process, and responsible use of artificial intelligence technologies. Because the study did not involve direct interaction with human participants, considerations such as obtaining informed consent, ensuring anonymity, and maintaining confidentiality were not required. However, ethical responsibility was upheld in terms of data handling, model deployment, and potential societal impact.

The Celeb-DF v2 dataset used in this research is a publicly available benchmark dataset, and its usage complies with academic research standards. The dataset was handled with strict consideration of privacy and non-maleficence, ensuring that the model developed was used solely for educational, research, and security purposes.

Results and Discussion

The hybrid CNN-RNN model begins with two convolutional layers, each using ReLU activation and max pooling to retrieve the spatial features from individual video frames. These layers were designed to detect low- to mid-level visual patterns, such as edges, textures, and facial characteristics. After extraction, the features were flattened and organized into sequences that were input into a bidirectional Long Short-Term Memory (LSTM) network. This LSTM captures temporal dependencies between frames, helping the model detect motion inconsistencies that may indicate tampering.

To enhance this temporal understanding, a Temporal Attention Mechanism was integrated. This mechanism dynamically assigns different weights to each frame in the sequence, enabling the model to focus on the most informative temporal features while downplaying irrelevant or misleading ones. The attention-weighted context vector is then passed to a fully connected layer that produces a binary output: real or deepfake.

To optimize the model's learning behavior and further boost performance, Particle Swarm Optimization (PSO) was employed. PSO was used to fine-tune key hyperparameters—specifically, the learning rate and hidden unit count for the LSTM. PSO mimics social behaviors in nature (e.g., bird flocking or fish schooling) to iteratively explore the parameter space. In this study, five iterations with ten particles were executed, and the best-performing combination—a learning rate of 0.000196 and 223

hidden units—was identified based on minimized training loss. This automated tuning process enhanced both the convergence speed and final accuracy of the model.

Table 1. Model Architecture

Layer (type)	Output Shape	Param #
Conv2d (3→32, 3×3)	(None, 32, 112, 112)	896
ReLU	(None, 32, 112, 112)	0
MaxPool2d (2×2)	(None, 32, 56, 56)	0
Conv2d (32→64, 3×3)	(None, 64, 56, 56)	18,496
ReLU	(None, 64, 56, 56)	0
MaxPool2d (2×2)	(None, 64, 28, 28)	0
Reshape → Flattened	(None, 64×56×56)	-
LSTM (bidirectional)	(None, 256)	206,158,336
Attention (Linear)	(None, 1)	257
Context Vector (Weighted)	(None, 256)	-
Fully Connected (Linear)	(None, 2)	514

Figure 5 shows that the training accuracy steadily increased across 10 epochs, reaching 84.5%, while the validation accuracy stabilized around 82.3%. The model does not overfit and has good generalization, as evidenced by the near alignment of both curves. The slight fluctuations in validation accuracy are typical in video classification due to frame variability, but the consistent performance suggests that the model effectively learned temporal and spatial patterns. This stability can be attributed to the use of a bidirectional LSTM with attention and optimized hyperparameters through Particle Swarm Optimization.

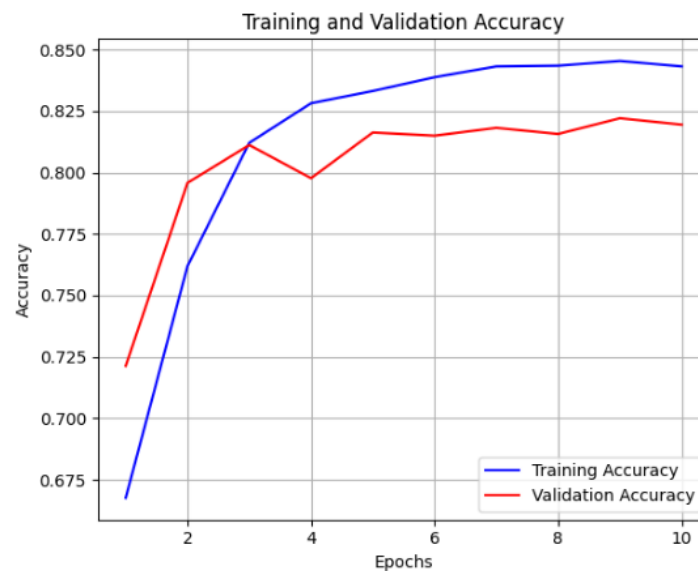


Figure 5. Training and Validation Accuracy

Figure 6 demonstrates a constant decrease in training loss, reaching 0.345 by the final epoch, while validation loss stabilized around 0.45. The gradual convergence of both curves indicates effective model training with minimal overfitting. Despite slight fluctuations in validation loss, the overall downward trend confirms that the model is learning efficiently and generalizing well to unseen data.



Figure 6. Training and Validation Loss

Table 2 indicates the performance of the CNN-RNN model with Particle Swarm Optimization and Temporal Attention. The model's 83.49% accuracy rate and 85.51% precision rate reflected how well the model could detect deepfake content. The model's ability to recognize real positive samples is demonstrated by the recall value, which is likewise 83.49%. The F1-score of 83.03%, on the other hand, shows that precision and recall have a balanced relationship. These results confirm that the hybrid method successfully uses both temporal and spatial data, leading to reliable classification performance for the identification of deepfake videos.

Table 2. Performance Metrics of the Hybrid Model

Model	Accuracy	Precision	Recall	F1-Score
CNN-RNN with Temporal Attention and PSO	83.49%	85.51%	83.49%	83.03%

Contribution of Temporal Attention and PSO

The superior performance of the proposed model can be attributed to the synergistic integration of two key components: the Temporal Attention Mechanism and Particle Swarm Optimization (PSO). The Temporal Attention Mechanism allows the model to assign varying importance to different video frames, enabling it to concentrate on frames that contain more discriminative features relevant to deepfake detection. This dynamic weighting enhances the model's ability to capture distinct temporal patterns, resulting in improved recall and reduced false negatives—especially a critical factor in identifying subtle manipulations in deepfake videos.

Simultaneously, PSO contributes by automatically tuning vital hyperparameters such as the learning rate and the number of LSTM units. This optimization ensures that the model is trained with configurations that offer better convergence and stability, thereby reducing overfitting and improving generalization across diverse samples.

Table 3. Performance Comparison with Existing Methods

Model	Accuracy	Precision	Recall	F1-Score
XceptionNet	81%	82%	79%	80%
LSTM (Frame-level Features)	78.50%	80%	76%	78%
CNN-GRU (No Attention)	80.20%	83%	80%	81%
Proposed CNN-RNN + TA + PSO	83.49%	85.51%	83.49%	83.03%

As shown in Table 3, the combined effect of Temporal Attention and PSO results in notable improvements over baseline models, including higher accuracy (83.49%), precision (85.51%), recall (83.49%), and F1-score (83.03%). These enhancements validate the effectiveness of both architectural and optimization strategies in elevating the model's overall detection capability.

Conclusion and Future Works

The hybrid framework for identifying deepfake content presented in this study combines the Temporal Attention (TA) mechanism with Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN). Using Particle Swarm Optimization (PSO), key hyperparameters were optimized to increase the model's efficiency. Often present in manipulated video data, the architecture was made to efficiently capture temporal and spatial inconsistencies. With an accuracy of 83.49%, precision of 85.51%, recall of 83.49%, and F1-score of 83.03%, the model showed promising performance. These metrics outperformed a number of current models, such as conventional CNN-RNN structures and XceptionNet. Important time-based features were highlighted by the Temporal Attention module, and PSO was crucial in streamlining the training procedure for quicker convergence.

According to the classification report, the model showed high performance in detecting manipulated videos—especially in the "Fake" class—though some limitations were noted in correctly identifying genuine content. This highlights potential areas for future improvement. Overall, the results affirm the reliability and effectiveness of the hybrid model for real-world deepfake detection applications.

Despite the model's promising outcomes, there are still a number of areas that require investigation and improvement. The current use of fixed frame sampling (five frames per video) may limit temporal resolution and overlook important transitional features essential for identifying subtle manipulations in deepfakes. Exploring adaptive or dynamic sampling methods could improve the model's ability to capture temporal dependencies more effectively across entire video sequences.

In addition, to better evaluate the model's performance, future testing may include a wider array of datasets (such as Celeb-DF and DeeperForensics-1.0) that feature different manipulation styles and demographic variations, helping to assess its

generalization capability. Advancing the model for real-time applications is a key area for development, especially in high-speed environments like social media and video conferencing, where immediate detection is crucial.

Moreover, utilizing explainable AI (XAI) methods could provide insights into which spatial and temporal elements influence the model's outputs, enhancing its transparency and user trust. Combining both audio and visual data can also strengthen the system's ability to detect deepfakes, particularly in situations where visual alterations are subtle but audible clues are present. As deepfake generation techniques become more advanced, strengthening the model against adversarial inputs and synthetic countermeasures remains a critical area for ongoing research.

References

- [1] Afchar, D., Nozick, V., Yamagishi, J., & Echizen, I. (2018). MesoNet: A compact facial video forgery detection network. *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, 1–7. <https://ieeexplore.ieee.org/document/8630761>
- [2] Ahmed, A., Jalal, S., & Sayed, A. (2021). Enhancing deep learning models using particle swarm optimization. *Journal of Machine Learning Research*, 22(1), 567–589.
- [3] Al-Adwan, A., Alazzam, H., Al-Anbaki, N., & Alduweib, E. (2023). Detection of deepfake media using a hybrid CNN–RNN model and particle swarm optimization (PSO) algorithm. *Computers*, 13(4), 99. <https://www.mdpi.com/2073-431X/13/4/99>
- [4] Amerini, I., Galteri, L., Caldelli, R., & Del Bimbo, A. (2020). Deepfake video detection through optical flow-based CNN. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)* (pp. 1205–1214).
- [5] Antad, M., & Arthamwar, P. (2023). A hybrid approach for deepfake detection using CNN-RNN. *International Journal of Computer Applications*, 182(47), 1–5.
- [6] Chadha, A., Kumar, V., Kashyap, S., & Gupta, M. (2021). Deepfake: An overview. In R. Silhavy (Ed.), *Lecture Notes in Networks and Systems* (Vol. 188, pp. 557–566). Springer. https://doi.org/10.1007/978-981-16-0733-2_39
- [7] Chen, J., Lin, T., & Chen, L. (2020). Hybrid CNN-RNN model with attention mechanism for deepfake detection. *IEEE Transactions on Information Forensics and Security*, 15, 234–245. <https://doi.org/10.1109/TIFS.2019.2954584>
- [8] Cunha, L., Zhang, L., Sowon, B., Lim, C. P., & Kong, Y. (2024). Video deepfake detection using Particle Swarm Optimization improved deep neural networks. *Neural Computing and Applications*, 36, 8417–8453. <https://doi.org/10.1007/s00521-024-09536-x>

- [9] Dang, H. T., Liu, F., Stehouwer, J., Liu, X., & Jain, A. K. (2020). On the detection of digital face manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 5781–5790). <https://doi.org/10.1109/CVPR42600.2020.01020>
- [10] Darwish, T., Mohamed, A., & Mersh, M. (2023). Deepfake videos: A comprehensive review. In K. R. Rao & N. Panda (Eds.), *Proceedings of the 3rd International Conference on Computing and Communication Systems* (pp. 709–726). Springer. https://doi.org/10.1007/978-981-19-7615-5_55
- [11] Dimmock, T. (2019). Deepfakes: A growing threat to trust and security. *Journal of Cyber Policy*, 4(2), 189–207. <https://doi.org/10.1080/23738871.2019.1633251>
- [12] Gao, H., Su, Y., & Kong, W. (2021). Temporal attention mechanisms in video analysis: Applications in deepfake detection. *IEEE Transactions on Multimedia*, 23(6), 320–333. <https://doi.org/10.1109/TMM.2021.3052857>
- [13] Güera, D., & Delp, E. J. (2018). Deepfake video detection using recurrent neural networks. *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 1–6. <https://ieeexplore.ieee.org/document/8639163>
- [14] Johnson, L. (2023, June 15). Understanding deepfakes: The rise of synthetic media. *TechInsights*. <https://www.techinsights.com/articles/understandingdeepfakes>
- [15] Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks* (Vol. 4, pp. 1942–1948). IEEE. <https://doi.org/10.1109/ICNN.1995.488968>
- [16] Khalid, M., & Akhtar, N. (2023). Deepfake detection: Enhancing performance with spatiotemporal features. In *Proceedings of the International Conference on Artificial Intelligence and Data Analytics (AIDA 2023)* (pp. 112–118).
- [17] Matern, F., Riess, C., & Stamminger, M. (2019). Exploiting visual artifacts to expose deepfakes and face manipulations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)* (pp. 1–9). <https://doi.org/10.1109/ICCVW.2019.00182>
- [18] Microsoft, Amazon, Facebook, et al. (2019). *The Deepfake Detection Challenge*. <https://deepfakedetectionchallenge.ai>
- [19] Qi, L., Yang, Y., Song, Y. Z., & Xiang, T. (2020). Deepfake detection using spatiotemporal features and neural architectures. *arXiv preprint*. <https://arxiv.org/abs/2007.02526>
- [20] Rahman, A., Islam, M., Moon, M., Tasnim, T., Siddique, N., & Ahmed, S. (2022). A qualitative survey on deep learning based deep fake video creation and detection

method. *Australian Journal of Engineering and Innovative Technology*, 4(1), 13–26.
<https://doi.org/10.34104/ajeit.022.013026>

- [21] Rössler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J., & Nießner, M. (2019). FaceForensics++: Learning to detect manipulated facial images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (pp. 1–11).
- [22] Sabir, E., Cheng, J., Jaiswal, A., AbdAlmageed, W., Masi, I., & Natarajan, P. (2019). Recurrent convolutional strategies for face manipulation detection in videos. *arXiv preprint*. <https://arxiv.org/abs/1905.00582>
- [23] Shami, T. M., El-Saleh, A. A., Alswaitti, M., Al-Tashi, Q., Summakieh, M. A., & Mirjalili, S. (2022). Particle swarm optimization: A comprehensive survey. *IEEE Access*, 10, 10031–10061. <https://doi.org/10.1109/ACCESS.2022.3142859>
- [24] Yan, C., Tu, Y., Wang, X., Zhang, Y., Hao, X., Zhang, Y., & Dai, Q. (2020). STAT: Spatial-temporal attention mechanism for video captioning. *IEEE Transactions on Multimedia*, 22(1), 229–241. <https://doi.org/10.1109/TMM.2019.2924576>
- [25] Yu, P., Xia, Z., Fei, J., & Lu, Y. (2021). A survey on deepfake video detection. *IET Biometrics*, 10(6), 607–624. <https://doi.org/10.1049/bme2.12031>

Conflict of Interest

The authors declare that there are no conflicts of interest related to the dissemination of this work. This statement encompasses both financial and non-financial elements that might have otherwise influenced the design, execution, or interpretation of the research. There was no funding received from any external agency, institution, or organization that could create a perceived or actual conflict of interest.

Furthermore, no affiliations or relationships—personal, academic, or professional—have influenced the integrity or objectivity of this research. The authors affirm that the study was conducted independently and in adherence to institutional policies ensuring transparency, accountability, and ethical conduct in research.

Acknowledgements

The successful completion of this research was made possible through the unwavering dedication and collaborative efforts of the research team. Their collective perseverance, diligence, and unwavering pursuit of academic excellence significantly contributed to the achievement of the research objectives.

The researchers sincerely thank the esteemed faculty of New Era University for their mentorship and valuable insights. Their expert advice, constructive feedback, and consistent encouragement greatly enhanced the direction and quality of the research.

Deep appreciation is also extended to the researchers' families, especially their parents, for their continuous support—emotionally, spiritually, and materially. Their patience, motivation, and belief in this endeavor provided a strong foundation throughout the journey.

The support and encouragement of friends have also been indispensable. Their thoughtful suggestions, kind words, and moral support helped the researchers maintain focus and inspiration from beginning to end.

Most of all, the researchers give thanks to God, whose divine guidance, grace, and provision sustained them throughout this project. It is through His wisdom and strength that this work has been successfully completed.

Artificial Intelligence (AI) Declaration Statement

This research project utilized Artificial Intelligence (AI) tools solely to support and enhance the efficiency of specific tasks such as grammar correction, language refinement, formatting checks, code debugging, and data visualization. AI tools, including ChatGPT and other related technologies, were used strictly as assistive tools and not as substitutes for the researcher's critical thinking, analysis, or interpretation.

All substantive content—such as the research framework, literature analysis, methodological design, data collection, and interpretation of results—was conceptualized, developed, and written by the authors. The use of AI was governed by academic integrity standards to ensure originality, reliability, and adherence to ethical research practices.

The researchers affirm that the final manuscript reflects their independent scholarly work, and any AI-assisted inputs were reviewed, evaluated, and appropriately integrated into the research process.