VIDVERIFY: AN AI POWERED SYSTEM FOR DETECTING DEEPFAKE AND MANIPULATED VIDEO CONTENT USING FRAME-BASED ANALYSIS

Reddyvari Venkateswara Reddy1, Adire RamaKrishna2, Anant Goel3, Basupally Akshaya Reddy4

1Associate Professor, Department of CSE (Cybersecurity), CMR College of Engineering & Technology, Hyderabad, Telangana,India

2,3,4 B.Tech Student, Department of CSE (Cybersecurity), CMR College of Engineering & Technology, Hyderabad, Telangana, India

Corresponding author mail: venkatreddyvari@cmrcet.ac.in

Abstract - Deep learning has made manipulated videos more realistic, making it harder to tell the difference between authentic and fraudulent content. These deepfakes raise serious ethical and security issues because they carry a high risk of misinformation, identity theft, and media manipulation. The problem statement centers on creating an AI-based system that uses deep learning techniques to analyze extracted frames and classify them as real or fake in order to detect manipulated videos. While pre-trained deepfake detection models like XceptionNet, EfficientNet, and DeepFakeDetector frequently require significant computational power and large datasets, existing solutions like manual verification are laborious and subject to human error. In order to train a Convolutional Neural Network (CNN) model that can effectively identify manipulated content, the suggested system takes frames from both authentic and fraudulent videos. Unlike existing systems, this approach ensures automation, higher accuracy, and adaptability for future enhancements. Frame-based analysis makes detection faster and more efficient, and the system can be further improved using advanced deep learning techniques.

Keywords- Deepfake Detection, Machine Learning, CNN, Face Detection, TensorFlow, Manipulated.

INTRODUCTION

Deepfakes are an emerging trend that has emerged in recent years due to developments in computer vision and artificial intelligence. Digitally altering a person's likeness to create incredibly lifelike photos, videos, or audio recordings that are frequently indistinguishable from real content is known as deepfaking. This technology presents serious risks, such as disinformation, political manipulation, identity theft, and a decline in public trust, even though it also has promising uses in accessibility, education, and entertainment.

Deepfakes are AI-generated videos that replace one person’s likeness with another in a highly realistic manner. With tools becoming publicly accessible, even individuals without technical expertise can generate deepfakes. These can be used maliciously—for identity theft, political misinformation, or online scams. The need for a robust detection system is more crucial than ever. Our solution, VidVerify, is an AI-based system that focuses on identifying video manipulations at the frame level, thus improving speed and accuracy.



# LITERATURE REVIEW

A. Tolosana, R., Vera-Rodriguez, R., Fierrez, J., Morales, A., & Ortega-Garcia, J. (2020) – DeepFakes and beyond: *A survey of face manipulation and fake detection*

A comprehensive review of deepfake detection and generation methods was presented by Tolosana et al. The emergence of digital media and machine learning techniques for facial manipulation detection are covered in their work. The social and technical issues that our deepfake detection model tackles are explained by this crucial study. The kinds of deepfake techniques we emphasized were also influenced by their classification of multiple face manipulation techniques, which included, identity swap, attribute manipulation and expression. We chose accuracy, precision, and recall as the primary metrics based on their assessment of performance metrics and difficulties in the existing detection models. [1]

B. Afchar, D., Nozick, V., Yamagishi, J., & Echizen, I. (2018) – MesoNet*: A Compact Facial Video Forgery Detection Network*

MesoNet, a lightweight CNN designed for deepfake video analysis, was proposed by Afchar et al. The architecture's accuracy and speed served as inspiration for a number of detection frameworks. Our work, which uses a custom CNN for static image classification, is similarly organized and is inspired by MesoNet's small size. Specifically, the design philosophy of MesoNet—balancing detection performance with computational efficiency—influenced our decision to employ shallow but efficient layers that extract important facial features without consuming a lot of GPU power and to restrict the number of convolutional blocks. MesoNet's emphasis on mesoscopic features, which fall between low-level textures and high-level semantics, also matched our goal of spotting minute facial manipulations in deepfake photos. [2]

C. Zhang, Z., Yan, Y., & Wu, H. (2020) – *Face Anti-Spoofing: A Survey of Advances and Challenges*

This paper provides a comprehensive overview of face anti-spoofing methods, covering both software-based and hardware-based detection. The difficulties mentioned, such as identifying high-quality artificial faces, highlight the real-world importance of automated solutions like ours. The study's explanation of movement indicators and texture analysis, in particular, influenced the case for the use of CNN architectures, which are skilled at extracting fine-grained spatial features from facial textures. Our focus on frame-level CNN detection as a more scalable software-based alternative was influenced by their classification of presentation attacks (such as print, replay, and 3D mask attacks), which provided insights into the range of threats that deepfake detection models must handle. [3]

D. Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016) – *Joint Face Detection and Alignment Using Multi-task Cascaded Convolutional Networks*

The MTCNN framework that we used in facial recognition and alignment was first described in this paper. By separating face regions from images before supplying them to our CNN classifier, MTCNN plays a critical part in preprocessing. This reference was crucial in directing our multi-task approach, which included both detection and facial landmark localization to guarantee accurate cropping, going beyond simple face detection. We were able to locate faces with high accuracy in a variety of lighting conditions, poses, and occlusions thanks to MTCNN's cascaded structure, which consists of P-Net, R-Net, and O-Net. [4]

E. Chollet, F. (2017) – *Deep Learning with Python*

Our CNN architecture was implemented with information from Chollet's work on real-world deep learning using TensorFlow and Keras. Also, our strategy for reducing overfitting was directly influenced by Chollet's recommendations on model regularization techniques, such as dropout layers. Our layer configuration and output logic for binary classification were influenced by his focus on activation functions, specifically the use of ReLU and Sigmoid. Furthermore, our model training process was guided by the book's thorough explanation of the Adam optimizer, batch normalization, and the training-validation-testing split. Chollet's real-world Keras examples also made it easier to integrate crucial elements like Dense layers, MaxPooling2D, and Conv2D, which made up the framework of our system. [5]

F. Dang, H., Liu, F., Stehouwer, J., Liu, X., & Jain, A. K. (2020) On the Detection of Digital Face Manipulation”,

Our knowledge of common artifacts introduced during face manipulation, such as irregularities in texture and blending, was guided by the work of Dang et al. Their research highlighted the significance of identifying subtle facial defects which informed our CNN's design of emphasizing fine-grained spatial patterns. This source also made it easier to compare our method to the most advanced deepfake detection methods.

G. A. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, “FaceForensics++: Learning to detect manipulated facial images,”

The FaceForensics++ dataset was chosen and worked with thanks in large part to the insights provided by Dolhansky et al. We were able to better understand dataset quality and variance thanks to the useful information their large-scale benchmark provided about data collection, reducing levels, and manipulation techniques. This had a direct impact on how we set up our validation and training procedures to ensure adaptability to various kinds of deepfake videos.

H. Y. Li, M.-C. Chang, and S. Lyu, “In Ictu Oculi: Exposing AI created fake videos by detecting eye blinking,”

Critical benchmarking information for assessing deepfake detection models was supplied by Rossler et al.'s work. Our selection of a CNN-based lightweight model for frame-level classification was validated by their performance comparison across different architectures. We were also able to refine our preprocessing pipeline while focusing on subtle manipulations by following their emphasis on visual effects and spatial inconsistencies.

# PROBLEM STATEMENT

We aim to solve the problem of video content manipulation by creating a custom model that doesn't rely on pre-trained APIs or cloud-based solutions. Our focus is on building an independent, scalable solution that can detect forged content using only custom CNN architecture, making it suitable for both academic research and practical applications in low-resource environments.

The core objective is:

“To develop a lightweight, frame-level video classification system using CNNs, capable of identifying whether a video is real or manipulated.”

One of the primary challenges lies in detecting high-quality manipulations that are nearly indistinguishable to the human eye. These include subtle artifacts in facial expressions, lighting inconsistencies, and unnatural temporal transitions. Many deepfake models are trained to minimize such clues, making detection harder. Our approach overcomes these limitations by:

* Extracting frames from videos to isolate spatial data.
* Preprocessing faces using accurate face detection and alignment (MTCNN), ensuring consistency in input data.
* Training a custom CNN to detect manipulation patterns in facial regions, focusing on texture, edge inconsistencies, and feature distortion.

The challenge is detecting manipulations that are often subtle and visually convincing. We overcome this by analysing patterns across frames and training the model with real-world deepfake datasets.

# METHODOLOGY

Data Collection and Preprocessing:

The dataset for this project was sourced from publicly available Kaggle repositories focused on deepfake detection.  
The dataset includes a balanced set of real and manipulated (deepfake) videos and images. Each video was processed as follows:

* **Frame Extraction**: We extracted individual frames from both authentic and manipulated (deepfake) videos in order to prepare the dataset for training the deepfake detection model. This step is essential because deep learning models can learn spatial features that differentiate real faces from fake faces more effectively when working with still images than with video streams. We automated the frame extraction procedure using Python's OpenCV library. Using OpenCV, frames were extracted from each video at a fixed interval to ensure temporal diversity without redundancy.
* **Face Detection and Cropping**: Since facial regions are the most important regions for detecting deepfake artifacts, face detection was done after frame extraction to separate them from each frame. Because the Multi-task Cascaded Convolutional Neural Network (MTCNN) has a high accuracy and resilience in identifying facial landmarks even in the face of difficult circumstances like changing lighting and angles, we chose it for this task. After a face was identified, it was automatically cropped out of the frame to eliminate extraneous background and concentrate the model's learning on the features of the face. The MTCNN detector was used to identify facial regions in each extracted frame. Irrelevant background areas were removed, allowing only the identified face regions cropped and kept for classification. This standardization ensured uniform input dimensions across the dataset, reduced computational load, and improved model convergence during training.
* **Image Resizing**: Every cropped face image was resized to 224 x 224 pixels to ensure consistency across all input data. Convolutional Neural Networks (CNNs) require fixed input dimensions, so this resizing was necessary. The size selection strikes a compromise between computational effectiveness and the preservation of crucial facial features required for precise deepfake identification. OpenCV was used for resizing, guaranteeing that every image fed into the model had the same shape. This simplified the training process and reduced the likelihood of errors resulting from mismatched input dimensions.
* **Normalization**: Pixel values were normalized to fall between 0 and 1 by dividing each pixel value by 255 prior to being fed into the Convolutional Neural Network (CNN). By decreasing the internal covariate shift and speeding up convergence, this step guarantees that the model trains more effectively. Additionally, normalization makes the learning process more stable and consistent by preventing problems like gradients that vanish or explode.
* **Data Augmentation**: Augmentation techniques such as random horizontal flipping, small rotations, brightness adjustments, and zooming were applied to enhance model robustness and prevent overfitting. To improve model generalization and reduce overfitting, data augmentation techniques were applied to the training set. These included operations such as horizontal flipping, slight rotation, zooming, and brightness adjustment. Data augmentation artificially increases the size of the training dataset and simulates real-world variations, enabling the model to become more robust to different facial orientations, lighting conditions, and minor occlusions. This step was implemented using Keras’ ImageDataGenerator.

The final pre-processed dataset was split into training (70%), validation (15%), and testing (15%) sets to evaluate the model effectively.

Classification Using CNN:

The classification task was approached using a custom-designed Convolutional Neural Network (CNN).  
The model takes the pre-processed facial images as input and outputs a binary label: Real (0) or Fake (1).

The CNN architecture consists of:

* **Convolutional Layers**: To extract both low-level features (like edges and textures) and high-level features (like facial structures and patterns), we used multiple Conv2D layers with ReLU (Rectified Linear Unit) activation functions. In order to preserve the spatial hierarchy and enable automatic feature learning tailored to facial forgery detection, each convolutional execution applies a set of learnable filters (kernels) to the input image or feature maps.
* **Pooling Layers**: MaxPooling2D layers follow convolutional blocks to reduce the spatial resolution of feature maps, thereby decreasing computational complexity and preventing overfitting. These layers help the model focus on the most prominent features by selecting the maximum value from small regions (typically 2×2 windows), retaining the most significant information while discarding minor variations that are unlikely to impact classification.
* **Dropout Layers**: To avoid overfitting during training, dropout layers are put in after convolution and pooling blocks. Every iteration, a portion of the layer's neurons are randomly disabled by applying a dropout rate (e.g., 0.5). The model performs better on unseen data thanks to this regularization technique, which forces the network to become more secure and general because it cannot rely too heavily on any one neuron.
* **Fully Connected Dense Layers**:  
  Following feature extraction, dense (fully connected) layers are applied to the flattened output, which ends with a Sigmoid activation function in the last layer. With this configuration, the network can combine all of the features that were extracted to produce a binary prediction: 0 for real and 1 for fake. In order to improve classification accuracy for subtle manipulations in deepfake content, the dense layers make sure that non-linear combinations of high-level features contribute to decision-making.The model was compiled using:
* **Optimizer**: We used the Adam (Adaptive Moment Estimation) optimizer, which combines the benefits of RMSProp and AdaGrad, two additional stochastic gradient descent extensions. Adam uses estimates of the gradients' first-order moments (mean) and second-order moments (uncentered variance) to adaptively update each parameter while maintaining individual learning rates. This makes the optimization process effective, particularly when dealing with noisy data and sparse gradients, which are frequent issues in image classification tasks like deepfake detection.
* **Evaluation Metrics**: We used a number of evaluation metrics for classification model's performance:
* Accuracy: The ratio of correctly predicted samples to total samples is used to calculate accuracy, which gauges how accurate predictions are overall.
* Precision: Reduces false positives by showing the proportion of positively predicted instances (Fake) that are true.
* Recall: Indicates the proportion of true positive cases (Fake) that the model accurately predicted, reducing false negatives.

When combined, these metrics offer a thorough assessment of the model's ability to differentiate between authentic and fraudulent facial images.

Training was performed over multiple epochs with a suitable batch size, and early stopping was optionally used to prevent overfitting.

A diagram of a process

AI-generated content may be incorrect.

Model Training:

Following preprocessing, the CNN architecture was trained using the cropped and normalized facial images. Three subsets of the dataset were created: 15% for testing, 15% for validation, and 70% for training. A balanced representation of both real and fake classes is ensured all over all phases thanks to this stratified division.

The Adam optimizer with binary cross-entropy loss, appropriate for binary classification tasks, was used to optimize the CNN model during training. To keep an eye on validation performance and avoid overfitting, early halting and model checkpointing strategies were used. To increase expansion and adaptability to invisible manipulations, randomly augmented images made up each training batch.

The model's performance was assessed using accuracy, precision, and recall on the validation after it was trained for a predetermined number of epochs or until integration. The model was compiled with the following parameters:

* **Loss function**: Binary Crossentropy (since the task is binary classification — real vs. fake)
* **Optimizer**: Adam optimizer with an initial learning rate of 0.001
* **Evaluation Metric**: Accuracy

Training was conducted over 5 epochs with a batch size of 32.  
The input images were resized to a target size of (224, 224) pixels.  
Since there are only two classes (real and fake), the class mode was set to binary.

To increase the robustness of the model, data augmentation techniques like random horizontal flips, small phases, and clarity transformations were used during training.

The training process was carried out locally on a Visual Studio Code (VS Code) environment using TensorFlow and Keras libraries.  
To avoid overfitting and preserve the top-performing model, early stopping and model checkpointing were used.

After training, the model achieved 72% accuracy on the unseen test dataset, demonstrating effective classification performance.

System Requirements:

Programming Language: Python

ML Libraries: Torch, DataLoader, sklearn

Data Processing: Numpy, Pandas, os

Image Handling: openCV

Visualisation: Seaborn, Matplotlib

Deployment:

The Visual Studio Code (VS Code) environment was used to assess and test the trained Convolutional Neural Network (CNN) mode..locally.   
By running test images through the model, predictions were generated directly within the Python scripts, and the console showed the results (real or fake classification).  
Model training and performance assessment were the main topics. The trained Convolutional Neural Network (CNN) model was deployed locally, tested, and evaluated using the Visual Studio Code (VS Code) environment. Writing Python scripts to load the trained model and process test inputs, specifically cropped and pre-processed face images taken from video frames, was part of the deployment process.

Real-time predictions were produced by feeding these test images into the model. The output, which was categorized as either Fake (1) or Real (0), was either displayed using helper libraries like matplotlib and OpenCV or printed straight to the terminal.

To improve the testing environment even more:

Before feeding the images into the CNN, an optimized inference pipeline was developed for efficient preprocessing (resizing, normalization).In order to simulate real-world usage, the deployment script featured batch testing options that allowed multiple frames from a single video to be carefully classified.

To monitor prediction confidence and classification time per frame, more logging was added.

With this configuration, the model's performance could be quickly and iteratively assessed, and thresholds and parameters could be adjusted without requiring a cloud-based deployment platform. For future development, the model can be readily transferred to other environments, such as Google Colab, Jupyter Notebook, or lightweight embedded systems with TensorFlow Lite support. It is still platform-independent.

## OUTPUT COMPARISION

To evaluate the performance of our CNN-based deepfake detection model, we plotted the training and validation accuracy and loss over 7 epochs. The graphs offer insights into how well the model learned from the data and how effectively it generalized.

A graph with lines and numbers

AI-generated content may be incorrect.

Model Accuracy:

The training accuracy shows a consistent upward trend, gradually improving with each epoch. Additionally, the validation accuracy increases quickly in the early epochs and varies a little in the later phases, but it stays somewhat comparable to the training accuracy, ranging between 71 and 73%.

Given that the validation accuracy does not substantially decline in comparison to the training accuracy, this suggests that the model can expand well without overfitting.

* The training accuracy (blue line) increases gradually and steadily, beginning just under 70% and peaking at about 72.2% by the sixth epoch. This suggests that with every epoch, the model is learning and getting better.
* The validation accuracy (orange line) rises sharply in the first epoch, from about 59% to over 71%. It then varies slightly over the following epochs but remains relatively close to the training accuracy.
* The small space between the two lines indicates low overfitting; the model is learning general features that work well on unseen data in addition to memorizing training data.
* It's common for validation accuracy to vary slightly, particularly when working with smaller datasets or fewer epochs.
* **Conclusion**: The model is learning efficiently and is well-generalized for the deepfake classification task, according to the accuracy plot.

A graph with numbers and lines

AI-generated content may be incorrect.

Model Loss:

Training loss steadily decreases across epochs, suggesting the model is effectively minimizing error on the training set In the first epoch, the validation loss exhibits a steep decline and then stabilizes with only slight variations.

Good model stability and the absence of overfitting or underfitting are indicated by the validation loss closely following the training loss and not increasing significantly.

* Validation Loss (orange line) begins higher at about 0.67, drops sharply in the first epoch, and then shows minor fluctuations around 0.59–0.61, staying close to training loss.
* Importantly, validation loss does not increase significantly, meaning the model is not overfitting and is maintaining its performance on unseen data.
* The closeness of training and validation loss implies that both datasets are being learned at similar rates, which is ideal.

Conclusion: The loss graph confirms the model is stable, well-optimized, and avoids overfitting throughout training.

Overall Analysis:

A good sign of model performance is the small difference in accuracy/loss between training and validation.   
  
The model's robustness and suitability for frame-level deepfake detection tasks in the real world are supported by its consistent performance on unseen data.

## CONCLUSION

With the help of a custom-trained Convolutional Neural Network (CNN), our project successfully creates a strong deepfake detection model without the need for costly commercial tools or dependence on third-party APIs. The system shows promising results in identifying manipulated facial content, exhibiting good generalization across different samples. Additionally, the architecture of the model is scalable and modular, making it easier to integrate sophisticated techniques like transfer learning using pre-trained models like XceptionNet for better performance in future iterations.

‘It’s completely custom-trained, scalable, and can be enhanced later with pre-trained models like XceptionNet if needed.

VidVerify:

* Offers a practical and cost-effective deepfake detection method.
* Is suitable for use in social platforms, education, and media authenticity checking.
* Provides a strong foundation for upcoming research in video security and AI ethics.

## FUTURE SCOPE

Using CNN-based facial image classification, the suggested deepfake detection system shows positive findings. There is, nevertheless, a great deal of room for improvement. By examining numerous frames over time, the model can eventually be expanded to identify deepfakes in complete videos. By using lightweight web frameworks like Flask or Streamlit to deploy the model, real-time detection capabilities can also be developed. Furthermore, using transfer learning with sophisticated pre-trained models could shorten training times and increase accuracy. Combining audio and video cues, or multimodal detection, can increase the system's resilience. Application will be improved by enlarging the dataset to include a wider range of facial conditions and appearances. The model's decision-making process can be visually understood through the application of explainable AI techniques.

## REFERENCES

## [1] R. Tolosana, R. Vera-Rodriguez, J. Fierrez, A. Morales, and J. Ortega-Garcia, “DeepFakes and beyond: A survey of face manipulation and fake detection,” *Information Fusion*, vol. 64, pp. 131–148, 2020.

[2] D. Afchar, V. Nozick, J. Yamagishi, and I. Echizen, “MesoNet: a compact facial video forgery detection network,” in *Proc. IEEE International Workshop on Information Forensics and Security (WIFS)*, 2018, pp. 1–7.

[3] Z. Zhang, Y. Yan, and H. Wu, “Face Anti-Spoofing: A Survey of Advances and Challenges,” *Frontiers in Computer Science*, vol. 15, no. 3, pp. 1–17, 2020.

[4] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, “Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks,” *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016.

[5] F. Chollet, *Deep Learning with Python*, 1st ed., Manning Publications, 2017.

[6] H. Dang, F. Liu, J. Stehouwer, X. Liu, and A. K. Jain, “On the detection of digital face manipulation,” in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 5781–5790.

[7] A. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, “FaceForensics++: Learning to detect manipulated facial images,” in *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 1–11.

[8] Y. Li, M.-C. Chang, and S. Lyu, “In Ictu Oculi: Exposing AI created fake videos by detecting eye blinking,” in *Proc. IEEE International Workshop on Information Forensics and Security (WIFS)*, 2018, pp. 1–7.