Multimedia Tools and Applications (2024) 83:49013–49037
https://doi.org/10.1007/s11042-023-17586-x

Fake-checker: A fusion of texture features and deep learning for deepfakes detection

Noor ul Huda[1] · Ali Javed[2] · Kholoud Maswadi[3] · Ali Alhazmi[4] · Rehan Ashraf[5]

Received: 17 February 2023 / Revised: 20 September 2023 / Accepted: 18 October 2023 /
Published online: 3 November 2023
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract
The evolution of sophisticated deep learning algorithms such as Generative Adversarial Networks has made it possible to create deepfakes videos with convincing reality. Deepfake identification is important to address internet disinformation campaigns and lessen negative social media effects. Existing studies either use handcrafted features or deep learning-based models for deepfake detection. To effectively combine the attributes of both approaches, this paper presents a fusion of deep features with handcrafted texture features to create a powerful fused feature vector for accurate deepfakes detection. We propose a Directional Magnitude Local Hexadecimal Pattern (DMLHP) to extract the 320-D texture features and extract the deep feature vector of 2048-D using inception V3. Next, we employ the Principal Component Analysis to reduce the feature dimensions to 320 for a balanced representation of features after fusion. The deep and handcrafted features are combined to form a fused feature vector of 640-D. Further, we employ the proposed features to train the XGBoost model for the classification of frames as genuine or forged. We evaluated our proposed model on Faceforensic++and Deepfake Detection Challenge Preview (DFDC-P) datasets. Our method achieved the accuracy and area under the curve of 97.7% and 99.3% on Faceforensic++, whereas 90.8% and 93.1% on the DFDC-P dataset, respectively. Moreover, we performed a cross-set and cross-dataset evaluation to show the generalization capability of our model.

Keywords Deepfakes · Deep Convolutional Neural Networks · Generative Adversarial Networks

✉ Rehan Ashraf
rehan@ntu.edu.pk

[1] Department of Computer Science, University of Engineering and Technology, Taxila 47050, Pakistan

[2] Department of Software Engineering, University of Engineering and Technology, Taxila 47050, Pakistan

[3] Department of Management Information Systems, Jazan University, 45142 Jazan, Saudi Arabia

[4] College of Computer Science and Information Technology, Jazan University, 45142 Jazan, Saudi Arabia

[5] Department of Computer Science, National Textile University, Faisalabad, Pakistan

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

1 Introduction

Due to the rise of various Online Social Platforms, thousands of videos have been shared daily for reporting and entertainment. Nowadays, digital video content can easily be altered by anyone with the help of many powerful tools like Adobe Premiere. Also, various state-of-the-art Generative Adversarial Networks (GANs) have been designed to manipulate video content with convincing reality. The term 'Deepfake media' refers to fake audio, videos, and images that are generated using deep learning techniques like GAN, and currently, synthetic videos are the most popular type of deepfake media. Presently, a video can be shared with many people using social media platforms within no time and that makes a huge impact on our lives [1].

Human facial deepfake comprises four different categories: (1) Face-swapping, (2) Face Synthesis, (3) Face Re-enactment, and (4) Face Editing [2]. Face-swapping is the most widely used deepfake technique in which the target person's face is swapped with the source person's face to create a fake video of the target person. Face-swap-oriented deepfakes are typically produced to defame individuals by showing them in circumstances they have never experienced. Face synthesis and face editing are other types of forgery that are used to create photorealistic face images and modification of facial attributes. This manipulation aims to create fake profiles on social media and propagate disinformation there. The last type of deepfake is face re-enactment, which involves mapping the expression of the source video onto the target video while keeping the target person's identity. In this paper, we mainly focus on detecting face re-enactment and face-swapping forgery as the majority of deepfakes videos contain these forgeries.

Most of the researchers have experimented with different manually engineered features to detect the forgeries [3, 4] which yields up to 96% classification accuracy. Texture features in the image can play a significant role in locating the real and fake frames because some deepfake video generation models cannot produce all the face's textures which can cause roughness in the image. The texture features provide discriminative information that can describe the image's regularity and roughness [5]. Local handcrafted descriptors have proved to be very effective in capturing textural information from images. Later, various deep learning approaches were also investigated for deepfake forgery detection and achieved better results in this area. Therefore, the combination of deep features with manual texture features is required to further enhance the detection accuracy. Furthermore, many deepfake detection models have been developed in the last few years to locate the forged frames in a video but these research works are not subjected to cross-dataset evaluation to build a generalized model. Later, some works [6] [7] tried to build a generalized model but failed to generalize better for other datasets. Therefore, there is a need to develop a better-generalized model for deepfake detection.

To address the requirements, we present a deepfake detection model that can effectively detect the face swap and face re-enactment facial forgeries using the fusion of deep and handcrafted texture features. We propose the Directional Magnitude Local Hexadecimal Pattern (DMLHP) descriptor to encode more discriminative texture features in the 16 directions of the neighborhood region. These texture features can reliably capture the roughness and regularity information of the bonafide and fake frames. Moreover, we employ the inception V3 for deep feature extraction. This model used the multi-scale feature fusion approach and, these multi-scale features can adequately analyze the given frame at various degrees of depth, which captures both global and local information. This approach can pick even minute details unique to real or fake frames that might be missed at a single scale by combining data from

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

different scales. Furthermore, various modifications or artifacts added to fake frames could show up differently at various scales. Because multi-scale characteristics capture a large range of visual cues and attributes, thus, offer more detailed information and improve the capacity to distinguish between legitimate and fake frames. Next, we fused the deep features with handcrafted features to improve the overall feature representation and to leverage the complementing benefits of both methods. Deep features extract complex patterns and hierarchical representations from raw data to ensure that high-level features can better represent the data semantics. The handcrafted features, on the other hand, are designed to counter the issues of low illumination conditions and variations in scale. A fused feature vector, which combines deep and handcrafted features, can capture a wider range of data, including high-level abstract representations from deep features and precise details from handcrafted features. The purpose of this fusion is to enhance the feature representation's performance and discriminative capacity for the subsequent classification of fake and real frames. At last, the proposed features are used to train an XGBoost model to distinguish between authentic and forged video frames. The following are the major contributions of this paper:

- We propose a novel DMLHP descriptor and concatenate them with the inceptionv3-based deep features to develop a powerful feature vector capable of capturing the distinctive traits of real and fake video frames.
- We propose an effective deepfake detection model robust to variations in gender, race, age group, facial angles, appearance and skin tone, camera positioning, lighting conditions, and background settings that can reliably detect face re-enactment and face swap forgeries from the videos.
- We have performed rigorous experimentation on Faceforensic + + and Deepfake Detection Challenge datasets to show the effectiveness of our model. Also, we conducted cross-set and cross-dataset evaluations to quantify the generalizability of our method for deepfakes detection.

The remaining article is organized as follows. We describe the earlier research in this field in Sect. 2. Section 3 describes the handcrafted texture feature extraction, deep feature extraction, feature transformation, feature fusion, and classification processes in detail. Section 4 focuses on the experimental findings. Section 5 provides the conclusion and recommendations for further research.

2 Related work

Over the past few years, various techniques based on deep features and handcrafted features have been employed for deepfake detection. Unlike previous research work, we propose a novel method for deepfake detection based on the fusion of deep features and handcrafted features, which is extracted using Inception V3 and DMLHP descriptor, respectively. This section presents a brief review of previous studies in the domain of deepfake detection.

2.1 Handcrafted features-based techniques

In recent years, many handcrafted feature-based techniques have been developed for forgery detection. Zhang et al. [3] presented a model for face swap forgery detection by using the Speeded Up Robust Features (SURF) descriptor for feature extraction. The extracted

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

features were employed to train the Support Vector Machine (SVM) for classification. This model is applicable for deepfake image detection only and is unable to detect video tampering. Ciftci et al. [8] discussed a video forgery detection method by estimating the heart rate from the face region in a video. In this study, the authors computed spatiotemporal characteristics of the facial feature and then used these features to train a convolutional neural network (CNN) and SVM model for the classification of manipulated videos. This model has a large feature vector space and requires a significant amount of time for training. Matern et al. [9] developed a deepfake detection method by using visual features like color and reflection detail in the eye and teeth region. These visual features were then used to train Logistic Regression (LogReg) and Multilayer Perceptron (MLP) neural network classifiers to identify the manipulated and non-manipulated content. The presented model is not appropriate if the subject has closed eyes or has not-so-clear teeth. Güera et al. [10] presented a facial deepfake detection technique by using the Multimedia stream descriptors (MSD) [11] for feature extraction that was then used to train the Random Forest and SVM to detect manipulated and non-manipulated faces. The authors provided an effective method for deepfake detection however this model performs poorly against video re-encoding attacks. Yang et al. [12] developed a method for deepfake detection based on the inconsistency between head movement and facial expression. The authors extracted facial landmarks and used these features to train the SVM model for classification. Jung et al. [13] introduced an approach to detect deepfakes by tracking the eye-blinking pattern within a video. In this study, the authors combined the Eye Aspect Ratio (EAR) method and the Fast-HyperFace method to detect eye-blinking. This approach has improved deepfake detection accuracy; however, this model is not suitable for people who are suffering from mental illness as in most cases they often experience irregular blinking patterns.

2.2 Deep features-based techniques

Deep learning (DL) techniques have also been employed for deepfake detection apart from the manually crafted feature methods. Afchar et al. [14] developed two distinct CNN models, each with a few layers that concentrated on mesoscopic image characteristics. The presented model was tested on a custom database and achieved 98.4% accuracy. Güera and Delp [15] presented a temporal aware pipeline to automatically recognize the deepfake media. The Inception V3 model is combined with the Long Short-term Memory (LSTM) model to identify whether the frame sequence has been tampered with or not. The accuracy of this model was tested using a confidential database. Nguyen et al. [6] presented a Multitask learning approach for segmenting and categorizing deepfake media simultaneously. This study used a novel method based on a Y-shaped encoder-decoder. The authors evaluated the presented approach on the Faceforensic++dataset and obtained promising results even with a small number of images. In this study, the author intended to make a generalised model, but this model seems not to generalize very well for other datasets. The generalization capability is an important property that describes a machine learning model's capacity to perform effectively on unseen datasets. It is an important part of model evaluation since it assesses how well the model can generalize patterns and relationships learned from the training data to make accurate predictions on the new and unseen instances. Rossler et al. [17] examined six deepfake detection techniques based on handcrafted and deep features. In this study, the authors used a face-tracking method for face detection from the video frames in the Faceforensic++dataset and fed these features to six classifiers separately to identify the deepfake videos. The XceptionNet system reports

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

the best performance among all classifiers. Moreover, this model only obtained outstanding results with uncompressed videos. Li and Lyu [30] proposed a method to identify the forensic manipulation made within a video. The authors extracted the facial landmarks and used these features to train VGG16, ResNet101, ResNet152, and ResNet50 to detect manipulated and non-manipulated videos. The presented model is not suitable for compressed videos. Sabir et al. [18] designed a model to identify manipulated videos by using temporal artefacts. In this study, CNN features were integrated with the gated recurrent unit cells and Dense Net to compute the temporal dependency across frames. The presented model achieves better detection accuracy; however, this model is suitable for static frames. Cozzolino et al. [16] presented a CNN-based deepfake detection model with better accuracy, but the presented work is computationally expensive. The summary of the literature is presented in Table 1.

3 Methodology

This section provides an overview of the proposed method. Our fake examiner model consists of two feature extractors: (1) Inception V3 and (2) DMLHP, and an XGBoost classifier. Our model is trained on cropped face images. These face images are fed to the CNN feature extractor and DMLHP descriptor for deep feature extraction and texture feature extraction, respectively. The deep features are fed to Principal Component Analysis (PCA) for feature transformation from 2048-D to 320-D. These reduced features and handcrafted texture features are concatenated to form a strong feature vector of 640-D. The proposed features are then fed to the XGBoost classifier for deepfake detection. Figure 1 depicts the overall pipeline of our model.

3.1 Data Preprocessing

In this step, the input videos are pre-processed by extracting the frames F_{fi} from the videos in each dataset, $f = 1, 2, 3, \ldots, N$, where f represents the video number in the dataset, N represents total videos and i represents the frame number in each video. In the deepfake detection method, faces are mostly forged, therefore we used a Multitask Cascaded CNN (MTCNN) [31] to extract faces from each frame. After face extraction, we resize all faces into 224×224 image size for further processing.

3.2 Handcrafted feature engineering

Handcrafted feature engineering approaches have been proposed in the past for various image classification tasks. Over the past few years, many texture feature descriptors have been designed such as Local Directional Pattern (LDP), Local Binary Pattern (LBP), and others, but these descriptors can compute features in limited directions and also disregard the magnitude information. While the additional directional information and higher-magnitude information can improve the performance of these texture descriptors. To better capture the useful texture patterns and cope with existing challenges, we employ the DMLHP texture descriptor that can encode more discriminative texture features in sixteen directions of the neighborhood region along with providing the texture magnitude information from the neighborhood region. In the DMLHP descriptor, the edge information is captured by using a texture magnitude-based pattern (TMBP), and the texture orientation-based pattern

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Table 1 Summary of the literature review

Ref	Methods	Features	Dataset	Results	Limitation
Handcrafted Features					
Zhang et al. [3]	SVM + SURF	Features extracted using SURF	Custom dataset	92% Acc	This model cannot deal with facial expression
Agarwal et al.[4]	SVM	Landmark features	Custom dataset	93% AUC	Degraded performance when a person not facing the camera
Matern et al.[9]	Logreg, MLP	Texture features	FF + +	78.4% AUC (Logreg) 85.1% AUC (MLP)	Degraded performance when the subject eyes are closed, and the subject have dirty teeth
Güera et al.[10]	RF, SVM	Facial features using MSD	Custom dataset	93% AUC (SVM) 96% AUC (RF)	This model performs poorly against video re-encoding attacks
Yang et al.[12]	SVM	Facial landmarks using DLib	UADFV	89% ROC	Not suitable for blurry images
Jung et al.[13]	EAR, Fast-HyperFace	Facial landmarks	Eye Blinking dataset	87.5% Acc	Not appropriate for people with a mental disorder
Ciftci et al.[8]	CNN	Heart rate features	Faceforensic	96% Acc	Large feature vector space
Deep Features					
Afchar et al.[14]	MesoInception- 4	Deep features	FF + +	83.71% Acc	Not suitable for highly compressed data
Güera and Delp [15]	RNN, CNN	Deep features	Custom dataset	97.1% Acc	Not suitable for long videos
Cozzolino et al.[16]	CNN	Deep features	Diverse Fake Face Dataset	99.43% Acc	Computationally expensive
Nguyen et al.[6]	CNN	Deep features	FF + +	83.71% Acc	Poor generalization capability
Rossler et al.[17]	CNN, SVM	Deep features	FF + +	90.29% Acc	Not suitable for compressed videos
Sabir et al.[18]	RNN, CNN	CNN features	FF + +	96.3% Acc	The model is not suitable for long video clips

Springer
Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Table 1 (continued)

Ref	Methods	Features	Dataset	Results	Limitation
Agarwal et al.[7]	VGG16	Deep features	Custom	0.0215 loss	Need to improve generalization capability
			DeepfakeTIMIT	0.0327 loss	
			Celeb-DF	99% AUC	
			FF	99% AUC	
			WLDR	99% AUC	
			DFDEx	93% AUC	
Montserrat et al.[19]	RNN, CNN	Deep features	DFDC	92.6% Acc	Need to improve performance score
Hu et al.[20]	CNN	Deep features	CelebDF	80.7% Acc 86.6% AUC	Poor generalization capability
			DFDC(Preview)	63.8% Acc 64.0% AUC	
Heo et al.[21]	CNN	Deep features	DFDC	97% AUC	Need to check the generalization capability of the model
Hu et al.[22]	CNN	Deep features	FF + +(DF)	94.6% Acc	Not effective for long videos
			CelebDF	80.7% Acc	
Hu et al.[23]	CNN	CNN features	DFDC(Preview)	80.3% Acc 82.8% AUC	Need to rise the performance accuracy
Aslam et al.[24]	CNN	CNN features	UCSD Ped1	96.0% AUC 11.8% EER	Need to check the generalization capability of the model
			UCSD Ped2	93.0% AUC 10.2% EER	
			CUHK Avenue	91.4% AUC 13.4% EER	

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Table 1 (continued)

Ref	Methods	Features	Dataset	Results	Limitation
Aslam et al.[25]	Attention-based adversarial autoencoder network (A3N)	CNN features	UCSD Ped1	90.7% AUC 13.2% EER	Need to evaluate the generalizability of the model
			UCSD Ped2	97.7% AUC 11.2% EER	
			CUHK Avenue	89.4% AUC 18.2% EER	
			ShanghaiTech	86.9% AUC 24.7% EER	
Aslam et al.[26]	CNN	CNN features	UCSD Ped1	90.7% AUC 13.2% EER	Need to improve performance score
			UCSD Ped2	96.2% AUC 11.2% EER	
			CUHK Avenue	83.4% AUC 18.2% EER	
Khalid et al.[27]	CNN	Deep features	FF++	87.76% Acc	Need to improve the classification performance
			WL dataset	99.03% Acc	
Khalid et al.[28]	CNN	Deep features	FF++	98% AUC	Need to improve the classification performance
			Celeb-DF	95% AUC	
			DFDC	92% AUC	
Ilyas et al.[29]	EfficienNet	Deep features	FF++	96.5% Acc	Need to evaluate the generalizability of the model
			DFDC (Preview)	88.41% Acc	

*Acc refers to accuracy, EER refers to equal error rate, and AUC refers to the area under the curve

Springer
Content courtesy of Springer Nature, terms of use apply. Rights reserved.

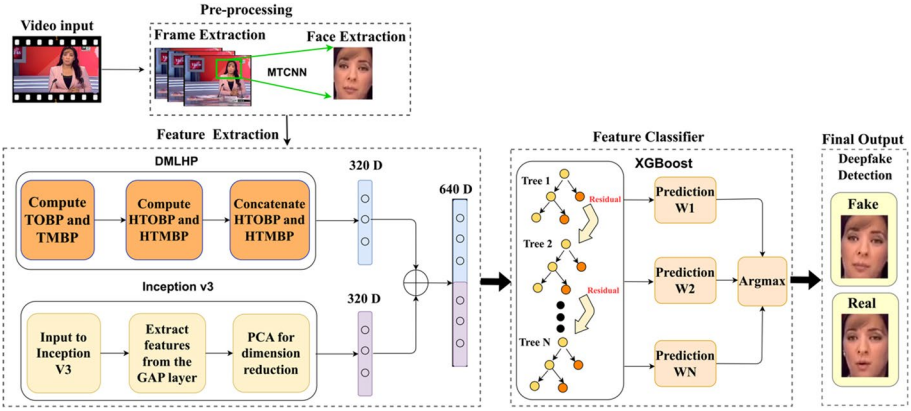

Fig. 1 Process flow of the proposed method

(TOMP), which further extracts discriminative information in the sixteen directions from the neighboring pixels. We gave the cropped faces to the DMLHP descriptor as input and extracted the texture patterns from the faces. After extracting these patterns, we compute the histograms (HTMBP) to compute a final texture feature vector.

3.2.1 Texture orientation-based pattern

To compute the TOBP for a given frame $F(x, y)$, firstly we convert the given frames into grayscale images and calculate the first order derivation of the central pixel along 135°, 90° ,45°, and 0° directions as:

$$f^1_{135^o}(n_c) = I(n_{db}) - I(n_c) \tag{1}$$

$$f^1_{90^o}(n_c) = I(n_v) - I(n_c) \tag{2}$$

$$f^1_{45^o}(n_c) = I(n_d) - I(n_c) \tag{3}$$

$$f^1_{0^o}(n_c) = I(n_h) - I(n_c) \tag{4}$$

Where n_{db}, n_v, n_h, and n_d denote the diagonal-back, vertical, horizontal, and diagonal neighborhoods direction of the central pixel, respectively. Using Eq. 5, we compute the direction of the center pixel based on the 1st-order derivation values of the center pixel.

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

$$f_{dir}^{1}\left(n_c\right)=\begin{cases}1,f_{0^\circ}^{1}\left(n_c\right)\geq 0 and f_{45^\circ}^{1}\left(n_c\right)\geq 0 and f_{90^\circ}^{1}\left(n_c\right)\geq 0 and f_{135^\circ}^{1}\left(n_c\right)\geq 0\\ 2,f_{0^\circ}^{1}\left(n_c\right)\geq 0 and f_{45^\circ}^{1}\left(n_c\right)\geq 0 and f_{90^\circ}^{1}\left(n_c\right)\geq 0 and f_{135^\circ}^{1}\left(n_c\right)<0\\ 3,f_{0^\circ}^{1}\left(n_c\right)\geq 0 and f_{45^\circ}^{1}\left(n_c\right)\geq 0 and f_{90^\circ}^{1}\left(n_c\right)<0 and f_{135^\circ}^{1}\left(n_c\right)\geq 0\\ 4,f_{0^\circ}^{1}\left(n_c\right)\geq 0 and f_{45^\circ}^{1}\left(n_c\right)\geq 0 and f_{90^\circ}^{1}\left(n_c\right)<0 and f_{135^\circ}^{1}\left(n_c\right)<0\\ 5,f_{0^\circ}^{1}\left(n_c\right)\geq 0 and f_{45^\circ}^{1}\left(n_c\right)<0 and f_{90^\circ}^{1}\left(n_c\right)\geq 0 and f_{135^\circ}^{1}\left(n_c\right)\geq 0\\ 6,f_{0^\circ}^{1}\left(n_c\right)\geq 0 and f_{45^\circ}^{1}\left(n_c\right)<0 and f_{90^\circ}^{1}\left(n_c\right)\geq 0 and f_{135^\circ}^{1}\left(n_c\right)<0\\ 7,f_{0^\circ}^{1}\left(n_c\right)\geq 0 and f_{45^\circ}^{1}\left(n_c\right)<0 and f_{90^\circ}^{1}\left(n_c\right)<0 and f_{135^\circ}^{1}\left(n_c\right)\geq 0\\ 8,f_{0^\circ}^{1}\left(n_c\right)\geq 0 and f_{45^\circ}^{1}\left(n_c\right)<0 and f_{90^\circ}^{1}\left(n_c\right)<0 and f_{135^\circ}^{1}\left(n_c\right)<0\\ 9,f_{0^\circ}^{1}\left(n_c\right)<0 and f_{45^\circ}^{1}\left(n_c\right)\geq 0 and f_{90^\circ}^{1}\left(n_c\right)\geq 0 and f_{135^\circ}^{1}\left(n_c\right)\geq 0\\ 10,f_{0^\circ}^{1}\left(n_c\right)<0 and f_{45^\circ}^{1}\left(n_c\right)\geq 0 and f_{90^\circ}^{1}\left(n_c\right)\geq 0 and f_{135^\circ}^{1}\left(n_c\right)<0\\ 11,f_{0^\circ}^{1}\left(n_c\right)<0 and f_{45^\circ}^{1}\left(n_c\right)\geq 0 and f_{90^\circ}^{1}\left(n_c\right)<0 and f_{135^\circ}^{1}\left(n_c\right)\geq 0\\ 12,f_{0^\circ}^{1}\left(n_c\right)<0 and f_{45^\circ}^{1}\left(n_c\right)\geq 0 and f_{90^\circ}^{1}\left(n_c\right)<0 and f_{135^\circ}^{1}\left(n_c\right)<0\\ 13,f_{0^\circ}^{1}\left(n_c\right)<0 and f_{45^\circ}^{1}\left(n_c\right)<0 and f_{90^\circ}^{1}\left(n_c\right)\geq 0 and f_{135^\circ}^{1}\left(n_c\right)\geq 0\\ 14,f_{0^\circ}^{1}\left(n_c\right)<0 and f_{45^\circ}^{1}\left(n_c\right)<0 and f_{90^\circ}^{1}\left(n_c\right)\geq 0 and f_{135^\circ}^{1}\left(n_c\right)<0\\ 15,f_{0^\circ}^{1}\left(n_c\right)<0 and f_{45^\circ}^{1}\left(n_c\right)<0 and f_{90^\circ}^{1}\left(n_c\right)<0 and f_{135^\circ}^{1}\left(n_c\right)\geq 0\\ 16,f_{0^\circ}^{1}\left(n_c\right)<0 and f_{45^\circ}^{1}\left(n_c\right)<0 and f_{90^\circ}^{1}\left(n_c\right)<0 and f_{135^\circ}^{1}\left(n_c\right)<0\end{cases}\tag{5}$$

Let $f_{dir}^{1}(n_c)|_{h=1,2,3\ldots 8}$ denote the direction of eight neighboring pixels and using the above equation we can calculate the direction of the 3×3 neighborhood in the same manner as the central pixel. After the first derivation, the second derivation is computed as:

$$TOBP^{2}=\{D_1\left(f_{dir}^{1}\left(n_c\right),f_{dir}^{1}\left(n_1\right)\right),D_1\left(f_{dir}^{1}\left(n_c\right),f_{dir}^{1}\left(n_2\right)\right),\ldots D_1\left(f_{dir}^{1}\left(n_c\right),f_{dir}^{1}\left(n_h\right)\right)\}|_{h=8}\tag{6}$$

Similarly, the n^{th} order derivation is computed as:

$$TOBP^{n}=\{D_1\left(f_{dir}^{n-1}\left(n_c\right),f_{dir}^{n-1}\left(n_1\right)\right),D_1\left(f_{dir}^{n-1}\left(n_c\right),f_{dir}^{n-1}\left(n_2\right)\right),\ldots D_1\left(f_{dir}^{n-1}\left(n_c\right),f_{dir}^{n-1}\left(n_h\right)\right)\}|_{h=8}\tag{7}$$

$$f_{dir}^{1}\left(n_c\right),f_{dir}^{1}\left(n_h\right)=\begin{cases}0, & \left(f_{dir}^{1}\left(n_c\right)=f_{dir}^{1}\left(n_h\right)\right)\\ f_{dir}^{1}\left(n_h\right), & otherwise\end{cases}\tag{8}$$

According to Eq. (8), if the direction of the central pixel is the same as the direction of its neighboring pixel, the adjacent pixel's direction will shift to 0 direction. Otherwise, nothing will alter the direction. By using Eq. (5) and Eq. (8), we computed the 8-bit texture orientation-based pattern and separate it into 15 binary patterns. An example of the above calculation is explained in Fig. 2.

The final 8-bit TOBP is 1, 1, 12, 1, 4, 1, 15, and 11. In our case, the direction of the center pixel is direction 14, and Fig. 3 illustrates the construction of a binary pattern for the remaining 15 directions, 1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, and 16. The Binary patterns are created by inserting a 1 where the TOBP value occurs and encoding the other bits with 0. The 15-bit pattern for the computed 8-bit TOBP value is 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0. We computed a total of 240 binary patterns in this manner.

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

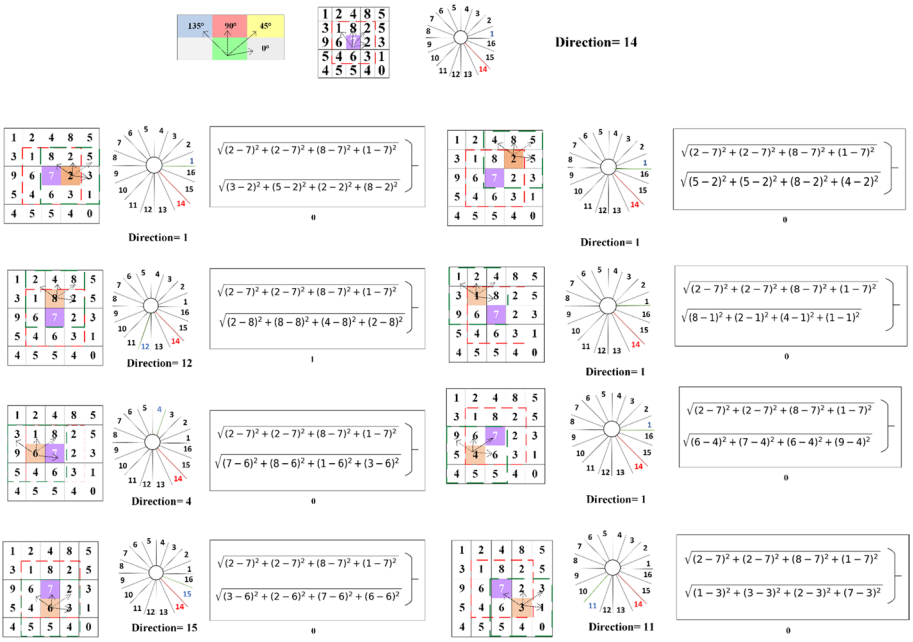

Fig. 2 An illustration of how TOBP and TMBP are computed. The TOBP calculation step for the central pixel is displayed in the orange box, while the 8 neighborhood regions are represented in the green box. In the given example orange box represents the center pixel 7 and its adjacent neighbors 1,8,2,2 in 135,90, 45, and 0 directions, whereas the green box represents the center pixel's eight surrounding neighbors with their adjacent neighbors in the four directions. The central pixel, which is "7," is highlighted by the purple box, while the yellow box indicates the neighboring pixel

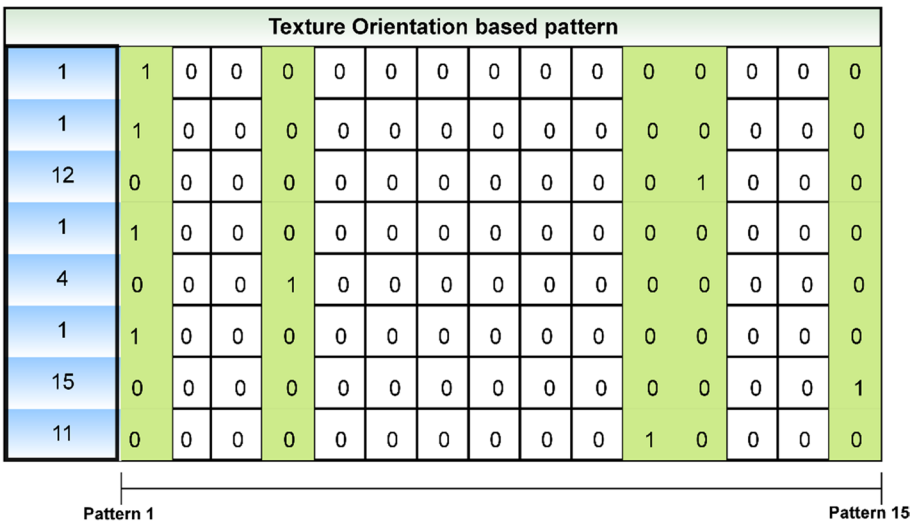

Fig. 3 Computation of binary pattern, while the blue column represents the TOBP 8-bit and the green column represents the occurrence of 8-bit pattern values (1, 1, 12, 1, 4, 1, 15, 11)

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

3.2.2 Texture magnitude-based pattern

Texture Magnitude-Based Pattern (TMBP) computes the magnitude information that is omitted in binary patterns. A better representation of the edge and gradient structure is provided by the magnitude information. We designed this pattern to investigate the relationship between neighboring pixels by using magnitude information along the horizontal, diagonal, vertical, and diagonal-back dimensions since compact texture information is in these four dimensions.

TMBP is computed as:

$$Z_I^1(n_c) = \sqrt{(f_{0^\circ}^1(n_h))^2 + (f_{45^\circ}^1(n_h))^2 (f_{90^\circ}^1(n_h))^2 + (f_{135^\circ}^1(n_h))^2} \tag{9}$$

$$\text{TMBP} = \sum_{h=1}^{h} 2^{(h-1)} \times D_3(Z_I^1(n_c) - Z_I^1(n_h))|_{h=8} \tag{10}$$

$$D_3(\text{x}) = \begin{cases} 1, & x \geq 0 \\ 0, & otherwise \end{cases} \tag{11}$$

Here D_3 is a function where x is derived by comparing the magnitudes of the central pixel and adjacent pixels; if the magnitude of the neighboring pixel is higher than the magnitude of the central pixel, the TMBP is encoded with the value 1, otherwise, 0 is assigned. For the given frame $F(x, y)$, we calculated the TMBP using Eq. (9) and Eq. (10). Figure 2. illustrates the calculation of the TMBP where the resultant TMBP is 0 0 0 1 0 0 0 0.

3.2.3 Final feature descriptor

After extracting the TOBP and TMBP patterns, the histograms of TOBP (HTOBP) and TMBP are obtained (HTMBP) using Eq. 12 and Eq. 13.

$$H_h(L) = \frac{1}{m \times n} \sum_p^m \sum_q^n D_4(LP(p, q), L) \tag{12}$$

LP represents the local pattern of each pixel (p,q), L represents the maximum LP, $m \times n$ refers to input size, and D_4 is expressed as:

$$D_4(\text{x}) = \begin{cases} 1, & x = y \\ 0, & otherwise \end{cases} \tag{13}$$

whereas $y{=}L$ and $x{=}LP$ in the Eq. 12. Further we concatenate these two histograms to form the final feature vector. The procedure of handcrafted feature engineering is discussed in Algorithm 1.

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Input: Frame repository F = $F_1, F_2, \ldots . F_f$, Query image X
Output: Feature vector of 320-D handcrafted features H_i
Directional magnitude local hexadecimal pattern (DMLHP) descriptor;
for TOBP do
  $f^1_{0^o}(n_h)|_{\alpha=0^\circ,45^\circ,90^\circ,135^\circ}$; // 1st derivation
  $f^1_{dir}(n_h)|_{h=1,2,3\ldots8}$ // direction of 8 neighboring pixels
  If D1:then //D = direction
    [TOBP-1]; // eight bit TOBP
    [TOBP-1_T],T= (2,3,…,16); // convert into 15 binary pattern
  end
  else If D16:then //D = direction
    [TOBP-16]; // eight bit TOBP
    [TOBP-16_T],T= (2,3,…,16); // convert into 15 binary pattern
  end
 [HTOBP]; // Histograms of orientation patterns
end
for TMBP do
  $Z^1_I(n_h))|_{h=1,2,\ldots\ldots8}$ // magnitude of pixels
  [TMBP]; // eight bit TOBP
  [HTMBP]; // Histograms of magnitude patterns
end
$HDMLHP$= [H$TOBP, HTMBP$]. // histogram concatenation
$H_i = X_i \in R^n$ // construct feature space
Return H_i

Algorithm 1 Handcrafted feature engineering

3.3 Deep feature engineering

Deep feature engineering is the process of transforming raw data into features that can be used to design a predictive model using deep learning. The proposed method uses a deep learning model 'Inception V3' based on CNN as a deep feature extractor. This model employs different convolution kernel sizes for multi-scale feature fusion; different sizes of convolution kernels are helpful for feature extraction as they have different sizes of receptive fields. Also, these multi-scale features can adequately express the information from the given frames in the video which can assist the classifier in locating forged frames in the video.

Table 2 displays the sequential information of the inception V3 model used in this study, including the layer name, output shape, and total learnable parameters of each layer. We discussed the details of the first and final 10 layers that contribute to the feature extraction process in Table 2. The default input size of inception V3 is 229×299 but we used 224×224 input size during the training and testing procedure. Because the 224×224 image size required less memory and network parameters as compared to the 229×299 input size, and it did not change the number of channels (2048 channels). The inception V3 is based on inception modules, these modules are used to reduce the parameter's value and generate the discriminative features. Several convolutional and pooling layers are used in parallel in each inception module. Convolution layers with convolutional kernel windows of size 3×3, and 1×1 is used to reduce the parameters that are linked with each other. The pooling layers are added behind the convolutional layers to reduce the dimension of the output of the previous convolutional layers. After the inception modules, we added a global

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Table 2 Details of inception V3 network

Layer (type)	Output Shape	Param #
Input Layer	[(None, 224, 224, 3)]	0
Conv2D	(None, 111, 111, 32)	864
Batch Normalization	(None, 111, 111, 32)	96
Activation	(None, 111, 111, 32)	0
Conv2D	(None, 109, 109, 32)	9216
Batch Normalization	(None, 109, 109, 32)	96
Activation	(None, 109, 109, 32)	0
Conv2D	(None, 109, 109, 64)	18,432
Batch Normalization	(None, 109, 109, 64)	192
Activation	(None, 109, 109, 64)	0
MaxPooling2D	(None, 54, 54, 64)	0
Activation	(None, 5, 5, 384)	0
Activation	(None, 5, 5, 384)	0
Activation	(None, 5, 5, 384)	0
Activation	(None, 5, 5, 192)	576
Batch Normalization	(None, 5, 5, 320)	0
Activation	(None, 5, 5, 768)	0
Concatenate	(None, 5, 5, 768)	0
Activation	(None, 5, 5, 192)	0
Concatenate	(None, 5, 5, 2048)	0
GlobalAveragePooling2D	(None, 2048)	0

average pooling (GAP) layer that is used to reduce the feature map's spatial dimension by calculating the average of all values in height and width. We extract the deep features from the GAP layer and obtain a 2048-D feature vector denoted as *DP*.

3.4 Feature transform

Before concatenation of these two features, we apply a PCA only to the deep feature, because the feature extracted from deep learning always has a high dimension, and usually, the increase in dimension can decrease the model's accuracy since the model needs to generalize more data [32]. Also, the distribution of these two features is not on the same scale, therefore, the direct fusion of deep features and handcrafted features is unreasonable and may cause an unbalanced representation of features during classification. Therefore, we apply PCA [33] for the dimensionality reduction of deep features from 2048-D to 320-D. We used a Python built-in PCA class and set the number of principal components to 320 to get the reduced feature dimension of 320. Next, we input our deep features into the PCA model. The model checks the variance on the features and preserves high-variance features. The high variance features are the rich features that provide more information from the given input. PCA removes the low variance features to reduce the deep feature vector's dimensionality.

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

3.5 Fusion of features

In this paper, we used the feature fusion approach to better represent the frames for deepfake detection. After computing the handcrafted features of 320-D and deep features of 320-D (after applying PCA), both features are concatenated to form a strong feature vector of 640-D. The proposed feature vector combines the advantages of both manually created texture features and deep learning features, and it offers supplemental information for each frame that can raise the model detection accuracy.

3.6 Classification using XGboost

After feature extraction and fusion of features, we fed these features to train the XGBoost model for classification. XGBoost model is an implementation of gradient-boosting decision trees that can accurately identify complex relationships and patterns within the feature space, producing extremely accurate classification outcomes. Furthermore, XGBoost uses regularization approaches to reduce overfitting and provide a strong generalization to new data. Additionally, XGBoost is effective for large-scale frame classification jobs due to its parallel processing capabilities, which allow for faster model training and prediction. XGBoost makes use of a combination of regression and classification trees (CARTs). The final prediction result is the sum of all of these T trees' prediction scores which is expressed as:

$$\widehat{y_a} = \sum_{t=1}^{T} d_t(x_a), d_t \in D \tag{14}$$

where y is the true value and x is the fused features, $a = 1, \ldots N$, here N represents the total samples in the training set. D represents the decision trees set, and expressed as:

$$D = (d(x) = wt_{s(x)}) \tag{15}$$

Here $d(x)$ refers to one of the trees, $wt_{s(x)}$ represents the leaf node weight, and s represents the structure of the tree. Therefore, the final predicted value of the XGBoost model is the sum of the leaf nodes of each tree. Overall, XGBoost decreases model complexity and adds regularization to the standard function. Additionally, to lessen overfitting and computation, this approach enables a column sampling approach.

For the experiments, we trained our classifier on a variety of hyper-parameters and selected those parameter values where we got the best results. We fine-tuned the XGBoost model and set the parameters Maximum depth = 30, Learning rate = 0.1, Maximum round = 500, and Objective = Softmax, because using these settings we obtained the best results. For genuine and forged frame classification, we used a logistic loss function as an evaluation metric. The logistic loss is expressed as:

$$Loss_{Logistic} = -\frac{1}{N} \sum_{a=1}^{N} \left(y_a \log(p_a) + (1 - y_a) \log(1 - p_a)\right) \tag{16}$$

If the $Loss_{Logistic} = 0$ in Eq. 16, then the given input belongs to the real class otherwise fake class. Whereas p and y represent the predictive and actual values, respectively.

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

4 Experiment and results

This section provides the details of different experiments conducted to assess our method. The experiments are performed on an Intel Core i5 dual-core processor running at 3.1 GHz and 8 GB of RAM. The handcrafted texture features are extracted using MATLAB R2022a, whereas the deep features are extracted using Google Colab. This platform allows us to integrate various open-source libraries like Keras and TensorFlow. We have used the Keras library to implement our DL model for feature extraction.

4.1 Dataset

We used the Faceforensic++ (FF++) [17] and Deepfake Detection Challenge Preview (DFDC-P) [34] datasets to assess the performance of our study. Faceforensic++ is the most used benchmark dataset in the forgery detection domain. Faceforensic++ consists of 4,000 fake videos and 1,000 real videos. Four different manipulation techniques are applied to each real video to create the deepfakes namely Neural Texture (NT), Deep Fake (DF), FaceSwap (FS), and Face2Face (F2F). DF and FS videos are generated using face-swapping techniques whereas NT and F2F are generated using face re-enactment techniques. This dataset provides three quality levels (high, low, and raw quality) of forged videos, but we adopted high-quality and light-compressed (HQ/c23) videos for our experiments.

The DFDC-P is the preliminary version of the Deepfake Detection Challenge (DFDC) dataset, which contains 1131 real videos and 4119 face videos. Two unknown deepfake creation approaches are applied to real videos for generating fake videos. We used 80% of the dataset for training and the remaining 20% for testing to evaluate the performance of our model.

4.2 Evaluation metrics

We assessed the performance of the proposed model using the area under the curve (AUC) and accuracy evaluation metrics as also adopted by contemporary methods. The AUC captures the area under the receiver operating characteristic curve and compares the relationship between the false positive rate and the true positive rate. The percentage of mistakenly predicted negative occurrences is known as the false positive rate, whereas the percentage of correctly predicted positive instances is known as the true positive ratio.

Accuracy is another important evaluation metric that is computed as:

$$A_C = \frac{F_{TP} + F_{TN}}{F_{TP} + F_{TN} + F_{FP} + F_{FN}} \tag{17}$$

F_{TP}refersto the total number of genuine video frames that the classifier has successfully identified as real, F_{TN}refersto the total number of fake video frames that the classifier has successfully identified as fake, F_{Fn} refers to the total number of genuine video frames that the classifier misclassifies as fake, and F_{FP} shows the total number of fake video frames that the classifier misclassifies as real.

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

4.3 Performance analysis of the proposed model

The objective of this evaluation is to show the effectiveness of the proposed framework for deepfake detection. The performance of our deepfake finder framework was evaluated using AUC and accuracy evaluation metrics on the Faceforensic + + dataset and DFDC-P dataset individually. In the first experiment, we evaluated the classification performance of our model on the Faceforensic + + dataset. For this, we used 80% videos of the dataset for training and the rest 20% of the videos for testing. For classification, we used the XGBoost classifier and fine-tuned it using the parameter values that are mentioned in Sect. 3.6. In this experiment, we used our proposed features to train XGBoost for the classification of frames as forged or bonafide and obtained 97.7% accuracy and 99.3% AUC on the Faceforensic + + dataset. Table 3 shows the summarized performance of our proposed model in terms of accuracy and AUC on Faceforensic + + datasets. This dataset contains videos generated by 4 different manipulation methods (NT, FS, DF, and F2F). The noteworthy outcomes on this dataset demonstrated that our algorithm can properly identify various types of manipulation techniques. We repeated the same experiment on the DFDC-P dataset for deepfakes detection. The proposed fusion-based model gives a significant performance of 90.8% accuracy and 93.1% AUC on the DFDC-P dataset. Table 3 demonstrates the results of our model on DFDC-P datasets in terms of AUC and accuracy. The DFDC-P is a diverse dataset and contains videos of people belonging to different geographical regions. This dataset has varied facial angles, gender, race, illumination changes, and background settings. The remarkable results on this dataset show that our model is robust to different background settings, variations in gender, race, age groups, facial angles, facial appearance, and illumination changes. From Table 3, we can observe that our model performed better on the Faceforensic + + dataset as compared to the DFDC-P dataset to detect the forged and genuine frames. Because the quality of the majority of videos in the DFDC-P dataset is poor and some of the faces are barely recognizable, therefore the performance of our model decreases to some extent when we evaluate it on DFDC-P.

4.4 Detection performance of deep feature, texture feature, and fusion

To quantify the effectiveness of our proposed fused feature, we designed a multi-stage experiment to assess the performance of texture and deep features alone and their fusion on both the Faceforensic + + dataset and the DFDC-P dataset individually.

In the first stage of this experiment, we evaluated the performance of our model using only the deep features. We used Inceptionv3 for deep feature extraction and employed these features to train the XGBoost for deepfake detection. Our proposed model obtained the accuracy and AUC of 90.4% and 93.3% on the Faceforensic + + dataset, and 88.6% and 90.0% on the DFDC-P dataset. In the second stage of this experiment, we employed our DMLHP texture features with Xgboost to detect the manipulated content and the results

Table 3 Performance evaluation of the proposed model

Method	Features	Dataset	AUC (%)	Accuracy (%)
Proposed Method (Inception V3 + DMLHP—XGBoost)	Deep Features + Texture Feature	Faceforensic + +	99.3	97.7
		DFDC-P	93.1	90.8

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Table 4 Comparative analysis of deep features, DMLHP texture features, and fused features

Method	Features	Faceforensic++		DFDC-P	
		Accuracy (%)	AUC (%)	Accuracy (%)	AUC (%)
Inception V3 + XGBoost	Deep features	90.4	93.3	88.6	90.0
DMLHP descriptor + XGBoost	Texture features	86.6	88.2	79.6	82.9
Proposed method	Deep features + Texture features	97.7	99.3	90.8	93.1

are mentioned in Table 4. In the last stage of this experiment, we employed the proposed fused features with XGBoost for deepfakes detection. From the results in Table 4, we can see that the deep features provide better performance as compared to texture features when used alone, but the fusion of deep and texture features offers the best performance. The fused feature vector holds the advantages of both feature extractors and creates a more reliable descriptor that can capture the most unique traits of the bonafide and forged frames.

4.5 Performance comparison with existing models

The objective of this evaluation is to show the effectiveness of our model for deepfake detection over contemporary techniques on two benchmark datasets. We conducted some experiments to compare the performance of our model to the existing deepfake detection methods.

In the first experiment, we compared our system performance to the existing deepfake detection methods on the Faceforensic++ dataset, and the results in terms of AUC and accuracy are provided in Table 5. From Table 5, we can infer that the proposed method outperforms the contemporary deepfake detection methods by achieving an accuracy of 97.7% and an AUC of 99.3%. Sabir et al. [18] was the second-best method in terms of accuracy and obtained an accuracy of 96.3%. Whereas, Qian et al. [35] was the second best method in terms of AUC and obtained an AUC of 97.9%. In [18, 35], the authors used only deep features for the deepfake detection whereas our model used the fusion of deep features and handcrafted features, and these fused features can better capture the unique traits of the bonafide and forged frames as compared to deep features only. Table 5 demonstrates that our model can detect different manipulated methods with higher detection accuracy than the comparative studies.

In the second experiment, we compared our method to the state-of-the-art deepfakes detection methods on the DFDC-P dataset, and the results are reported in terms of AUC and accuracy in Table 6. Our proposed model provides the best detection accuracy and achieved 93.1% of AUC and 90.8% of accuracy for deepfake detection on the DFDC-P dataset. Tolosana et al. [36] was the second-best system for deepfakes detection with an AUC of 91.1%, whereas Ilyas et al. [29] was the second-best model in terms of accuracy of 88.4%. Both second-best methods are based on deep features, in contrast, our method is based on the fusion of handcrafted and deep features, which ensures to capture both the low- and high-level significant traits from data. The accuracy of deepfake detection is improved when handcrafted features and deep features are combined because it takes advantage of both methods. To detect abnormalities or manipulations in photos or videos, such as subtle face inconsistencies or artefacts, handcrafted features are created to capture domain-specific information and traits. On the other hand, deep features that are derived from deep learning models can recognise and learn complicated, hidden patterns in the

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Table 5 Comparative study on Faceforensic++dataset in terms of accuracy score (Acc) and AUC

Study	Deep learning-based Models	Handcrafted feature-based Models	AUC (%)	Acc (%)
Afchar et al.[14]		✓	84.7	83.1
Matern et al.[9]		✓	66.4	-
Yang et al.[12]		✓	47.3	-
Nguyen et al.[6]	✓		76.3	92.8
Sabir et al.[18]	✓		-	96.9
Nguyen et al.[6]	✓		-	92.8
Wang et al.[37]	✓		-	85
Amerini and Caldelli [38]	✓		-	94.2
Qian et al.[35]	✓		97.9	-
Liu et al.[39]	✓		96.9	-
Khormali and Yuan [40]	✓		94.4	93.6
Khalid et al.[27]	✓		87.7	
Khalid et al.[28]	✓		98	
Ilyas et al.[29]	✓			96.5
Proposed	✓	✓	99.3	97.7

data. Combining these two different types of features allows the deepfake detection system to benefit from both precise, human-designed feature engineering and the capacity to automatically learn and adapt to developing deepfake approaches, producing more reliable and accurate detection results. This combination improves overall detection performance by adequately addressing the dynamic nature of deepfake production. From Table 6, the reported results confirm that our model can better detect the deepfake content in case of variation in gender, age groups, race, facial angles, lighting condition, and facial appearance than the prior deepfake detection methods.

4.6 Performance evaluation with PCA and without PCA

The objective of the experiment is to assess the impact of Principal Component Analysis (PCA) on the performance and efficiency of our deepfake detection algorithm. In the first scenario, we applied the PCA-based feature selection on the deep 2048-D feature vector and reduced its dimensionality to 320 principal components. After that, we fused these

Table 6 Performance comparison of our model with existing models on the DFDC-P dataset

Study	Deep learning-based Models	Handcrafted feature-based Models	AUC (%)	Acc (%)
Lang et al.[41]	✓		85.1	-
Hu et al. [20]	✓		64.0	63.8
Tolosana et al. [36]	✓		91.1	-
Hu et al.[23]	✓		82.8	80.3
Lee et al. [42]		✓	-	61.1
Ilyas et al. [29]	✓		-	88.41
Proposed Model	✓	✓	**93.1**	**90.8**

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

components with 320-D handcrafted features and used these fused features as input for our XGBoost classification model. Our proposed model obtained the accuracy and AUC of 97.7% and 99.3% on the Faceforensic + + dataset, and 90.8% and 93.1% on the DFDC-P dataset. In the second stage of this experiment, we evaluated our performance without using PCA. For this, we directly used the 2048 high-dimension deep feature vector as input to the algorithm and followed the same procedure mentioned in the first experiment. The proposed model obtained the classification accuracy and AUC of 94.2% and 96.1% on the Faceforensic + + dataset, and 89.9% and 87.8% on the DFDC-P dataset. The experiment results are mentioned in Table 7. From the results, we can demonstrate that our model provides better results with the inclusion of PCA, as it simplifies the feature vector by reducing the redundancy by minimizing the impact of irrelevant features. Thus, enhances the effectiveness of proposed features for better representation of genuine and forged frames and ultimately improve model generalization and accuracy for deepfake detection.

4.7 Performance comparison with other classifiers

To evaluate the effectiveness of the XGBoost classifier with the proposed fused feature vector, we compared the performance of XGBoost against conventional classifiers, including K-nearest neighbors (KNN) and SVM. Initially, we fine-tuned these three classifiers, for KNN, we set the number of neighbors parameter at 7 and the chosen distance metric is Euclidean. For SVM, the kernel scale parameter is configured at the value of 18, while the selected kernel is the radial basis kernel (RBF). For XGBoost, we used the same hyperparameters that are mentioned in Sect. 3.6. We selected these parameters as we achieved the best results on these settings.

In our first experiment, we extracted the fused feature vector using the Faceforensic + + and DFDC-P datasets individually and used these features one by one as input to the SVM model for classification. The classification results in terms of AUC and accuracy are mentioned in Table 8. In our second and third experiments, we repeat the same procedure for KNN and XGBoost and the performance outcomes are mentioned in Table 8. From the results listed in Table 8, we can demonstrate that the XGBoost classifier outperforms others and achieved the highest AUC and accuracy of 99.3% and 97.7% respectively using Faceforensic + + data, while we obtained an AUC of 93.1%, and accuracy of 90.8% on DFDC-P dataset. The SVM model performs second best by obtaining an AUC of 96.7%, and an accuracy of 94.7% on the Faceforensic + + dataset, whereas we achieved an AUC of 89.1% and an accuracy of 87.8% on the DFDC-P dataset. While KNN performed the worst and achieved 94.1% of AUC, and 92.2% of accuracy on the Faceforensic + + dataset, whereas we achieved an AUC of 86.9% and an accuracy of 84.8% on the DFDC-P dataset.

Table 7 Performance comparison of our model with PCA and without PCA

Method	FF + +		DFDC-P	
	AUC	Acc	AUC	Acc
With PCA	99.3	97.7	93.1	90.8
Without PCA	96.1	94.2	89.9	87.8

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Table 8 Comparative analysis of machine learning classifiers

Method	FF++		DFDC-P	
	AUC	Acc	AUC	Acc
SVM	96.7	94.7	89.1	87.8
KNN	94.1	92.2	86.9	84.9

Deepfake detection requires identifying subtle modifications and patterns in the video frames. These complex relationships are well-captured by the XGBoost ensemble learning approach, which integrates numerous decision trees. In contrast to SVM, which depends on creating rigid hyperplanes to divide data points and may have trouble with complex, non-linear decision boundaries. Also, XGBoost can adapt and generalize well to the different patterns encountered in forgery detection, contributing to its superior performance than other classifiers.

4.8 Cross-set evaluation on the faceforensic++subsets

The objective of this evaluation is to show the generalization capability of our model on unseen subsets of the Faceforensic++dataset. In the first experiment, we check the generalizability of our model on unseen neural texture videos. For this purpose, we trained our model using DF, FS, F2F, and real data and evaluated it on NT data. The proposed model achieved 73.8% accuracy for neural texture forgery detection. The outcomes are highlighted in Table 9. In the second experiment, we check the generalizability of our model on unseen DF subset videos. For the training process, we used FS, F2F, real, and NT videos whereas for testing we used the DF videos. The proposed model achieved 86.7% accuracy for deepfake forgery detection. In the third experiment, we check the classification performance of our model on unseen face swap videos. To train our proposed model, we used real videos, neural texture videos, deep fake videos, and face2face videos. To test our model, we used face swap videos and achieved 81.4% accuracy for face swap forgery detection. In the last experiment, we identify the generalizability of our model on unseen face2face videos. For this experiment, we used NT, DF, FS, and real videos for the training procedure, and face2face videos for the testing procedure. The proposed model obtained 74.3% accuracy for face2face forgery detection. The outcomes of the above experiments are reported in Table 9.

Table 9 shows that our model has the best classification performance for deepfake forgery detection while we obtained the second-best classification performance for face swap forgery detection. Our finding shows that we achieved poor results when we train our model on DF, FS, F2F, and Real videos and test our model on the NT videos, as the major portion of the training set contains videos generated by face-swapping techniques

Table 9 Cross-set evaluation on each subset of Faceforensic++dataset

Train set	Test set	ACC (%)
DF, FS, F2F, Real	NT	73.8
FS, F2F, Real, NT	DF	86.7
F2F, NT, DF, Real	FS	81.4
NT, DF, FS, Real	F2F	74.3

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

while the test set contains videos generated by facial re-enactment techniques. Additionally, it is quite challenging to detect NT videos because in these videos only the mouth-related facial expression is manipulated.

4.9 Cross-dataset evaluation

In this section, we reported the cross-dataset evaluation of our model on the unseen dataset to show the generalization capability of our model. Both datasets (Faceforensic++and DFDC-P) are first split into an 80:20 ratio (20% for the testing procedure and 80% for training). In the first experiment, we trained our model on the Faceforensic++dataset, tested it on the DFDC-P dataset, and achieved 54.3% accuracy. In the second experiment, the proposed model is trained on the DFDC-P dataset, tested on the Faceforensic++dataset, and obtained 69.4% accuracy. In the third experiment, we evaluate the cross-dataset performance of our model on an unseen manipulated method. For this purpose, we trained our model on the DFDC-P dataset and tested it on the face swap videos from the Faceforensic++dataset.

The proposed model obtained 67.3% accuracy on unseen face swap videos and the results are listed in Table 10. Using the same experimental protocols mentioned in the third experiment, we individually tested our model on the face2face videos, deepfake videos, and neural texture videos from the Faceforensic++dataset. The proposed model obtained 69.8% accuracy for face2face forgery detection, 81.8% for deepfake forgery detection, and 58.8% for neural texture forgery detection. The results of cross-dataset experiments are stated in Table 10. From Table 10, we can infer that our model provides remarkable performance on unseen deepfake videos from the Faceforensic++dataset with an accuracy of 81.8%. The proposed model provides the second-best performance on unseen face2face videos from the Faceforensic++dataset with 69.8% classification accuracy. It can be observed from the results that our proposed model achieved exceptional results when we trained our model on the DFDC-P dataset and tested it on the Faceforensic++dataset with 69.4% accuracy. In contrast, our model provides the worst performance when we trained it on the Faceforensic++dataset and tested it on the DFDC-P dataset. This is due to the reason that the DFDC-P dataset is a challenging dataset and is collected under different scenarios like outdoors and indoors with different light conditions (like day, night, midnight, etc.), pose variations, and camera positions that make it difficult to detect. Moreover, the subjects who participated in making this dataset are of different age groups and gender. Additionally, these individuals belong to different geographical regions (South Asia, Africa, America, East Asia, and Caucasia) and have dissimilarities in facial attributes, i.e., the skin tone of Americans is different from Africans. These variations in race, gender, skin tone, and age group make the cross-dataset evaluation on the DFDC-P dataset more

Table 10 Cross-dataset evaluation

Train Dataset	Test Dataset	Accuracy (%)
DFDC-P	DF subset	81.8
	F2F subset	69.8
	FS subset	67.3
	NT subset	58.8
	Entire FF+	69.4
FF++	DFDC-P	54.3

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

challenging. Despite these reasons, we can infer that our model trained on the Faceforensic++dataset and tested on the DFDC-P dataset obtained 54.3% accuracy. This outcome is encouraging after considering the diversity in the DFDC-P dataset.

5 Conclusion and future work

In this paper, we have proposed a novel fake-checker framework for the detection of forged frames in a video that is reliable as well as robust to the different deepfakes generating methods. More specifically, our method can distinguish between face-reenactment and face-swap deepfakes that are produced using various methods. Our proposed model comprises the fusion of deep features with handcrafted features that can capture the distinctive traits of real and fake frames. The proposed model is tested on the DFDC-preview and Faceforensic++datasets and exhibits exceptional identification performance in spite of challenging scenarios like deepfakes produced by several algorithms with different facial angles, skin tone, camera positioning, lighting conditions, and background settings. To show the efficacy of our model, we compared the performance of our method with contemporary techniques. The experimental outcomes show that our model outperformed the majority of deepfake detection models in terms of AUC and accuracy. We also performed the cross-set and cross-dataset evaluation of our model. Our method performed reasonably well on the cross-set scenario but showed lower results on cross-dataset evaluation. In the future, we intend to test our method on other datasets i.e., UADFV, Celeb-DF, etc., and improve the generalizability of our method in terms of cross-dataset evaluation.

Acknowledgements This work was supported by the grant of the Punjab Higher Education Commission (PHEC) of Pakistan via Award No. (PHEC/ARA/PIRCA/20527/21).

Data availability The dataset used in the current study is available publicly. Data sharing is not applicable to this article as no datasets were generated during the current study.

Declarations

Conflict of interest There is no conflict of interest.

References

1. Khan SA (2022) Hybrid transformer network for deepfake detection. ArXiv. /abs/2208.05820
2. Mirsky Y, Lee W (2021) The creation and detection of deepfakes: A survey. ACM Comput Surv (CSUR) 54(1):1–41
3. Zhang Y, Zheng L, Thing VL (2017) Automated face swapping and its detection. In: 2017 IEEE 2nd international conference on signal and image processing (ICSIP). IEEE, pp 15–19
4. Agarwal S, Farid H, Gu Y, He M, Nagano K, Li H (2019) Protecting world leaders against deep fakes. In: CVPR workshops, vol 1, p 38
5. Xu B, Liu J, Liang J, Lu W, Zhang Y (2021) DeepFake videos detection based on texture features. CMC-Comput Mater Contin 68(1):1375–1388
6. Nguyen HH, Fang F, Yamagishi J, Echizen I (2019) Multi-task learning for detecting and segmenting manipulated facial images and videos. In: 2019 IEEE 10th international conference on biometrics theory, applications and systems (BTAS). IEEE, pp 1–8
7. Agarwal S, Farid H, El-Gaaly T, Lim S-N (2020) Detecting deep-fake videos from appearance and behavior. 2020 IEEE international workshop on information forensics and security (WIFS): IEEE. p 1–6

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

8. Ciftci UA, Demir I, Yin L (2020) Fakecatcher: Detection of synthetic portrait videos using biological signals. IEEE transactions on pattern analysis and machine intelligence
9. Matern F, Riess C, Stamminger M (2019) Exploiting visual artifacts to expose deepfakes and face manipulations. 2019 IEEE Winter Applications of Computer Vision Workshops (WACVW): IEEE. p 83–92
10. Güera D, Baireddy S, Bestagini P, Tubaro S, Delp EJ (2019) We need no pixels: Video manipulation detection using stream descriptors. arXiv preprint arXiv:190608743
11. Jack K (2011) Video demystified: a handbook for the digital engineer. Elsevier
12. Yang X, Li Y, Lyu S (2019) Exposing deep fakes using inconsistent head poses. In: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, pp 8261–8265
13. Jung T, Kim S, Kim K (2020) Deepvision: Deepfakes detection using human eye blinking pattern. IEEE Access 8:83144–83154
14. 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): IEEE, p 1–7
15. Güera D, Delp EJ (2018) Deepfake video detection using recurrent neural networks. 2018 15th IEEE international conference on advanced video and signal based surveillance (AVSS): IEEE. p 1–6
16. Cozzolino D, Thies J, Rössler A, Riess C, Nießner M, Verdoliva L (2018) Forensictransfer: Weakly-supervised domain adaptation for forgery detection. arXiv preprint arXiv:181202510
17. Rossler 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, pp 1–11
18. Sabir E, Cheng J, Jaiswal A, AbdAlmageed W, Masi I, Natarajan P (2019) Recurrent convolutional strategies for face manipulation detection in videos. Interfaces (GUI) 3(1):80–87
19. Montserrat DM, Hao H, Yarlagadda SK, Baireddy S, Shao R, Horváth J, ... Delp EJ (2020) Deepfakes detection with automatic face weighting. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, pp 668–669
20. Hu J, Liao X, Wang W, Qin Z (2021) Detecting compressed deepfake videos in social networks using frame-temporality two-stream convolutional network. IEEE Trans Circuits Syst Video Technol 32(3):1089–1102
21. Heo YJ, Choi YJ, Lee YW, Kim BG (2021) Deepfake detection scheme based on vision transformer and distillation. arXiv preprint arXiv:2104.01353
22. Hu J, Liao X, Wang W, Qin Z (2022) Detecting compressed deepfake videos in social networks using frame-temporality two-stream convolutional network. IEEE Trans Circuits Syst Video Technol 32(3):1089–1102
23. Hu J, Liao X, Liang J, Zhou W, Qin Z (2022) Finfer: frame inference-based deepfake detection for high-visual-quality videos. In: Proceedings of the AAAI conference on artificial intelligence, vol 36, no 1, pp 951–959
24. Aslam N, Kolekar MH (2022) Unsupervised anomalous event detection in videos using spatio-temporal inter-fused autoencoder. Multimed Tools Appl 81(29):42457–42482
25. Aslam N, Rai PK, Kolekar MH (2022) A3N: Attention-based adversarial autoencoder network for detecting anomalies in video sequence. J Vis Commun Image Represent 87:103598
26 Aslam N, Kolekar MH (2023) DeMAAE: deep multiplicative attention-based autoencoder for identification of peculiarities in video sequences. Visual Comput 2023:1–15
27. Khalid F, Javed A, Irtaza A, Malik KM (2023) Deepfakes catcher: a novel fused truncated densenet model for deepfakes detection. In: Proceedings of international conference on information technology and applications: ICITA 2022. Springer Nature Singapore, Singapore, pp 239–250
28. Khalid F, Javed A, Ilyas H, Irtaza A (2023) DFGNN: An interpretable and generalized graph neural network for deepfakes detection. Expert Syst Appl 222:119843
29. Ilyas H, Javed A, Aljasem MM, Alhababi M (2023) Fused swish-ReLU efficient-net model for deepfakes detection. In: 2023 9th International Conference on Automation, Robotics and Applications (ICARA). IEEE, pp 368–372
30. Li Y, Lyu S (2018) Exposing deepfake videos by detecting face warping artifacts. arXiv preprint arXiv:181100656
31. Zhang K, Zhang Z, Li Z, Qiao Y (2016) Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Signal Process Lett 23(10):1499–1503
32. Ma J, Yuan Y (2019) Dimension reduction of image deep feature using PCA. J Vis Commun Image Represent 63:102578

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

33 Yang J, Zhang D, Frangi AF, Yang J-Y (2004) Two-dimensional PCA: a new approach to appearance-based face representation and recognition. IEEE Trans Pattern Anal Mach Intell 26(1):131–7
34. Dolhansky B, Howes R, Pflaum B, Baram N, Ferrer CC (2019) The deepfake detection challenge (dfdc) preview dataset. arXiv preprint arXiv:1910.08854
35. Qian Y, Yin G, Sheng L, Chen Z, Shao J (2020) Thinking in frequency: Face forgery detection by mining frequency-aware clues. Springer, European conference on computer vision, pp 86–103
36. Tolosana R, Romero-Tapiador S, Fierrez J, Vera-Rodriguez R (2021, January) Deepfakes evolution: analysis of facial regions and fake detection performance. In: International conference on pattern recognition. Springer International Publishing, Cham, pp 442–456
37. Wang R, Juefei-Xu F, Ma L, Xie X, Huang Y, Wang J, Liu Y (2019) Fakespotter: a simple yet robust baseline for spotting ai-synthesized fake faces. arXiv preprint arXiv:1909.06122
38. Amerini I, Caldelli R (2020, June) Exploiting prediction error inconsistencies through LSTM-based classifiers to detect deepfake videos. In: Proceedings of the 2020 ACM workshop on information hiding and multimedia security, pp 97–102
39. Liu H, Li X, Zhou W, Chen Y, He Y, Xue H, ... Yu N (2021) Spatial-phase shallow learning: rethinking face forgery detection in frequency domain. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 772–781
40. Khormali A, Yuan J-S (2022) DFDT: An End-to-End DeepFake Detection Framework Using Vision Transformer. Appl Sci 12(6):2953
41. Li X, Lang Y, Chen Y, Mao X, He Y, Wang S, ... Lu Q (2020) Sharp multiple instance learning for deepfake video detection. In: Proceedings of the 28th ACM international conference on multimedia, pp 1864–1872
42. Lee S, An J, Woo SS (2022) BZNet: unsupervised multi-scale branch zooming network for detecting low-quality deepfake videos. In: Proceedings of the ACM Web Conference 2022, pp 3500–3510

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

Content courtesy of Springer Nature, terms of use apply. Rights reserved.

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH ("Springer Nature").

Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users ("Users"), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use ("Terms"). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.

These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.

We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.

While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.

These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.

To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.

Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.

If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com