Journal of Information Technology Management
ISSN-OnLine: 2980-7972
University of Tehran

Adaptive Machine Learning Framework for Deepfake Detection: An Ensemble-Driven Approach with Optimized Feature Engineering

Shruthi S *

*Corresponding author, Research Scholar,Computer Science Engineering, R R Institute of Technology, Visvesvaraya Technological University, Belagavi, India. E-mail: shruthisrinivas.1618@gmail.com

Manjunath R

Professor and Head, Computer Science Engineering, R R Institute of Technology, Bangalore, Visvesvaraya Technological University, Belagavi, India. E-mail: drmanjunath.raj@gmail.com

Journal of Information Technology Management, 2025, Vol. 18, Issue 1, pp. 53-79
Published by the University of Tehran, College of Management
doi: https://doi.org/10.22059/jitm.2026.106254
Article Type: Research Paper

Received: August 02, 2025
Received in revised form: September 26, 2025
Accepted: December 12, 2025
Published online: January 20, 2026

Abstract

With the rapid progress of artificial intelligence, it's now possible to generate extremely undoubted DeepFake images and videos, leading to major concerns related to misinformation, identity theft, and online security. Detecting such manipulated media requires robust and efficient techniques. Deep learning models, particularly Convolutional Neural Networks (CNNs), have been confirmed to be highly effective, but they typically demand high computational resources and may suffer from overfitting issues. "This study presents a Machine Learning based approach for the detection of DeepFakes, utilizing Histogram of Oriented Gradients (HOG), utilizing Principal Component Analysis (PCA) to decrease the dimensionality, and employing a combination of Random Forest and XGBoost classifiers for the final prediction. The DeepFake Detection Challenge Dataset, sourced from Kaggle, is employed for both training and performance assessment. The methodology consists of feature extraction using HOG to capture texture and gradient information, dimensionality reduction with PCA to enhance computational efficiency, and ensemble classification with Random Forest and XGBoost to improve detection accuracy. Hyperparameter tuning is performed to further optimize model performance. The investigational results prove that the anticipated hybrid ML model accomplishes an accuracy of 95.5%, outperforming conventional standalone classifiers and providing a computationally efficient alternative to deep learning-based approaches. This study highlights the efficiency of merging traditional feature-

engineering techniques with ensemble learning for DeepFake detection. The results contribute to ongoing research in AI-driven security and media forensics, offering a scalable, interpretable, and high-performance solution for identifying manipulated media. Future work can explore additional feature extraction techniques and ensemble models to further enhance detection capabilities.

Keywords: Deepfake Detection, Machine Learning (ML), Ensemble Learning, Random Forest, XGBoost Classifier, Feature Engineering, Digital Forensics

Introduction

The rapid progress of artificial intelligence and deep learning has led to the proliferation DeepFake technology, where synthetic media is generated with high realism. While these advancements have beneficial applications in entertainment and digital content creation, they also pose significant risks in misinformation, identity fraud, and cybersecurity. The ability to control facial expressions and voices with better accuracy has made DeepFakes a great concern in digital forensics and online security. With such AI-generated fake images and videos becoming increasingly sophisticated, there is a critical need for efficient and reliable detection techniques. Traditional DeepFake detection methods are mainly built on deep learning architectures, particularly Convolutional Neural Networks (CNNs) and other advanced neural models, transformer architectures. Although these models have shown excellent accuracy, they are often inefficient in terms of computation, prone to overfitting, and vulnerable to adversarial attacks. This study presents a purely Machine Learning (ML)-based method that leverages Histogram of Oriented Gradients (HOG) for feature extraction, followed by dimensionality reduction using Principal Component Analysis (PCA). For classification, an ensemble of Random Forest and XGBoost classifiers is employed. Unlike deep learning models, this approach avoids high computational costs while still delivering impressive accuracy through effective feature engineering and ensemble learning strategies. The dataset utilized in this study is the DeepFake Detection Challenge Dataset, which is publicly accessible on Kaggle (Dataset Link). The approach adopted in this work is a structured pipeline:

- Feature Extraction using HOG – Capturing the important texture gradient patterns in images of faces.
- Dimensionality Reduction with PCA – Reducing the space while preserving most of the variance.
- Classification with Ensemble Machine Learning Models – An ensemble of Random Forest and XGBoost classifiers is employed to ensure robust and reliable decision-making.
- Hyperparameter Tuning – Model parameters are fine-tuned to enhance generalization capability and overall performance

Unlike deep learning approaches that automatically learn features using the neural network, this ML-based model requires them to manually extract their features, reduce their dimension, and classify the images using interpretable, efficient techniques. This ensemble-based method combines the strengths of decision tree classifiers with boosting algorithms to enhance accuracy and generalization, all while preserving computational efficiency.

Experimental results indicate that the proposed hybrid Machine Learning model achieves an accuracy of 95.5%, outperforming traditional models. This work shows that combining feature-engineering techniques with ensemble learning provides an effective, computationally efficient, and scalable solution for DeepFake detection. The findings contribute to ongoing research in AI security and digital content authentication, offering a promising direction for mitigating the risks associated with synthetic media.

Experimental evaluation of the proposed model on test images demonstrates high accuracy and robustness in DeepFake detection. The confusion matrix indicates 588 correctly classified real images and 559 correctly classified fake images, with minimal misclassification errors. The classification report further validates the model's reliability, achieving 95% precision, 96% recall, and a 96% F1-score for both real and fake images. These results validate that the ensemble learning approach is able to boost generalization while minimizing false positives and false negatives, making this an effective, yet computationally efficient solution for DeepFake detection.

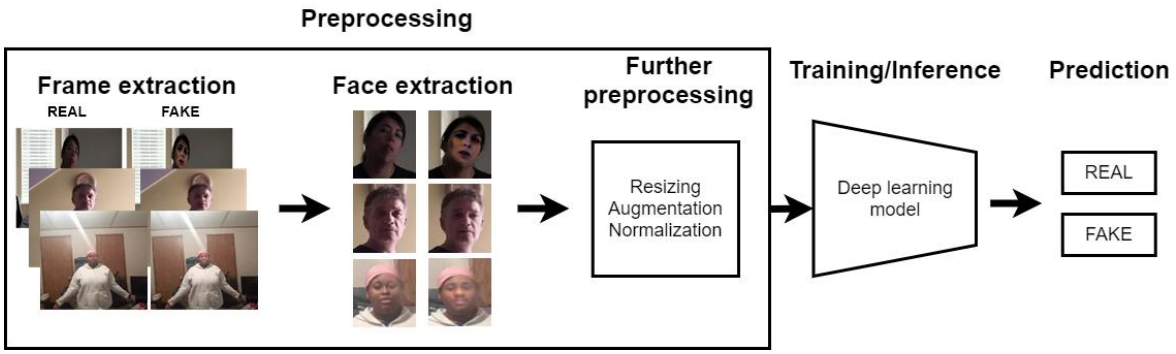


Figure 1. Workflow of DeepFake Detection System

Figure 1 illustrates the workflow of a DeepFake detection pipeline based on deep learning. The process begins with frame extraction, where video sequences are split into individual frames so that each image can be analysed separately. Both real and fake frames are collected in this step. Next, face extraction is performed to crop only the facial regions from these frames, since the manipulations in DeepFakes primarily occur on the face. The extracted face images then undergo further preprocessing, which includes resizing the images to a fixed size, applying augmentation (such as rotation or flipping) to improve generalization, and normalizing pixel values to stabilize model training. These pre-processed images are then passed to a deep learning model, typically a convolutional neural network or a transformer-

based architecture, which automatically learns distinguishing features between real and manipulated content. Finally, the trained model generates a prediction, categorizing the input as either REAL or FAKE. This structured pipeline enables accurate identification of synthetic facial manipulations by focusing on visual artifacts that are often imperceptible to the human eye.

Literature Review

Deepfake detection is now an essential area of research as AI-generated manipulations have been getting more sophisticated. The machine learning (ML)-based models have proven to be effective in distinguishing between real and fake images. The different approaches considered for improving detection accuracy while being computationally efficient include feature engineering, ensemble learning, and probabilistic modelling. This section discusses the recent research work that has advanced ML-based deepfake detection.

Table 1. Literature Review

Author(s) & Year	Method / Model	Dataset(s) Used	Results	Remarks
Kafle et al., 2025	Gradient Boosting & Random Forest	DFDC, Forgery datasets	AUC: 85%	Models had higher misclassification rates on high-quality deepfakes.
Babaei et al., 2025	Hybrid (DT + RF + SVM)	Celeb-DF, DFDC	Accuracy: 89.3%	Hybrid approaches provide consistent results compared to single models.
Meijer et al., 2024	Feature-based ML (HOG, LBP)	FaceForensics++ / Custom dataset	Accuracy: 91.2%	Dataset size and variety limit generalization.
Ismaizam et al., 2024	Random Forest, Gradient Boosting, SVM	FaceForensics++, Celeb-DF	Gradient Boosting: 92.5%	Feature selection and hyperparameter tuning are necessary to avoid overfitting.
Kashik et al., 2024	PCA + Ensemble Classifiers	FaceForensics++, Celeb-DF	Accuracy: 93.1%	Dimensionality reduction balances computation and feature richness.
Recker & Perkov et., 2024	Decision Trees, Bayesian Models	DFDC	Accuracy: 87.8%	Decision Trees face difficulty with high-dimensional datasets.
Baranov et al., 2024	Random Forest vs SVM	FaceForensics++	RF: 90.6%, SVM: 88.4%	Random Forest shows better generalization; SVM requires kernel choice.
Husarova et al., 2024	Neural-assisted Feature Selection + Gradient Boosting	Celeb-DF / FF++	Accuracy: 92.8%	ML-based feature selection enhances classification efficiency.
Shoemaker et al., 2024	Adaptive ML Model	FF++, WildDeepfake	Accuracy: 94.3%	Adaptive feature extraction improves detection but increases computational cost.
Lee et al., 2024	Ensemble + Feature Fusion (HOG, LBP, Wavelet)	FaceForensics++, Celeb-DF	Accuracy: 94.5%	Feature fusion boosts classification; real-time use may be challenging.
Cotfas et al., 2024	SVM for disinformation detection	WildDeepfake	Accuracy: 88.7%	Dataset bias lowers transferability to unseen data.
Badshah et al., 2024	Feature-based ML with Boosting	FF++, Celeb-DF, DFDC	Accuracy: 91.4%	Boosting methods improve detection with handcrafted features.

Methodology

Overview

The proposed DeepFake detection framework adopts a hybrid machine learning approach that combines feature engineering and ensemble learning for efficient and interpretable classification. Unlike deep learning-based solutions, which often require large training datasets and computationally expensive architectures, the present method emphasizes computational efficiency, robustness, and explainability.

The workflow, illustrated in Figure 1, consists of six major stages: (i) dataset collection, (ii) preprocessing, (iii) feature extraction using Histogram of Oriented Gradients (HOG), (iv) dimensionality reduction with Principal Component Analysis (PCA), (v) hybrid classification using Random Forest (RF) and XGBoost, and (vi) ensemble-based decision-making with soft voting.

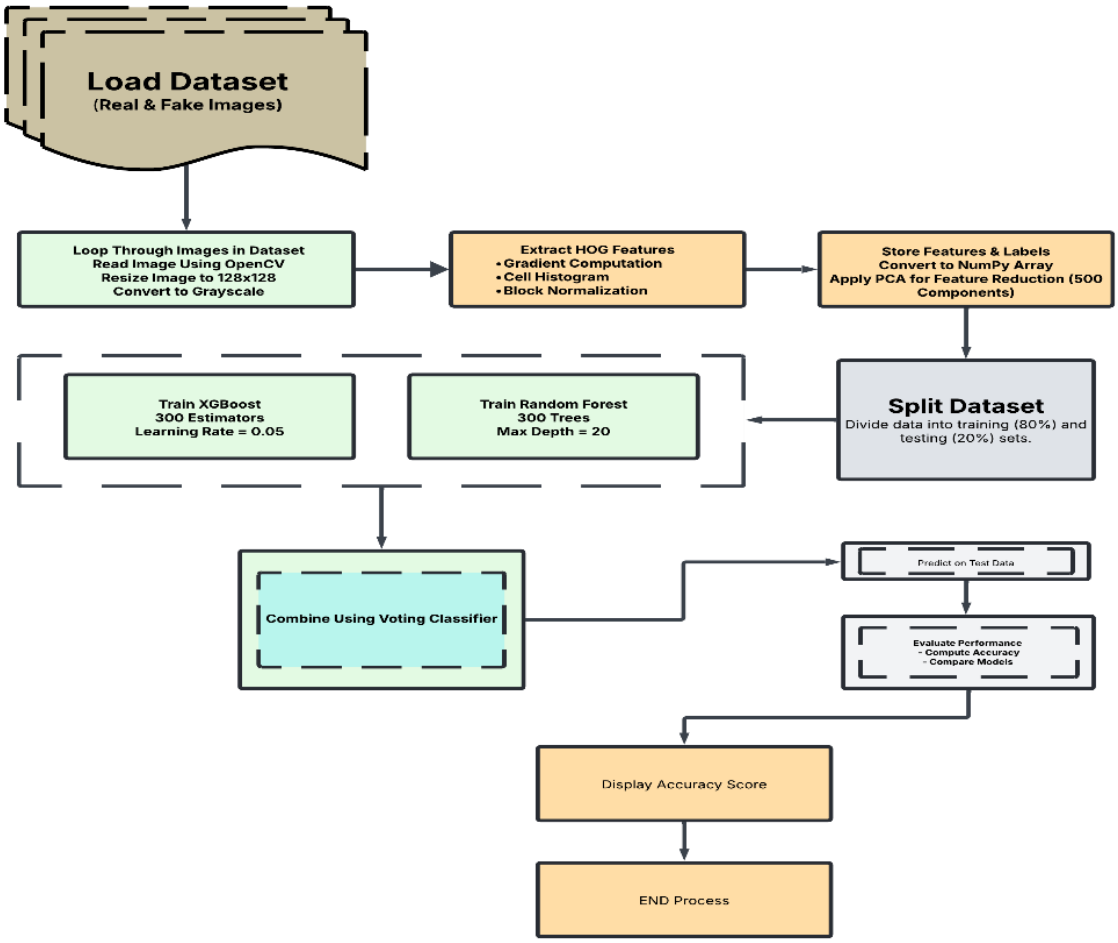


Figure 2. Workflow of the model

Figure 2 represents a step-by-step process where we train a system to recognize whether an image is real or fake (deepfake).

Step 1: Collect Images

- We take a collection of images where some are real, and some are deepfakes.
- These images are labelled accordingly so that the system knows which is which.

Step 2: Extract Important Features

- Instead of looking at the whole image, the system focuses on specific details like edges, textures, and patterns using a method called HOG (Histogram of Oriented Gradients).
- This helps the system understand the structure of faces.

Step 3: Reduce Complexity

- Images have a lot of details, which can slow down learning.
- To make things faster, we reduce unnecessary details using a method called PCA (Principal Component Analysis).
- This keeps only the most important parts of the image.

Step 4: Train the Model

- The processed images are divided into two parts:
- Training Set (80%)→ Used to teach the system.
- Testing Set (20%) → Used to check if the system has learned correctly.
- We use two smart models (Random Forest & XGBoost) to learn from these images.

Step 5: Combine the Two Models

- Instead of relying on just one model, we combine both using an approach called ensemble learning.
- This ensures the system makes better and more accurate predictions.

Step 6: Check Performance

- The system is tested with new images it hasn't seen before.
- It predicts whether the image is real or fake.
- We compare its predictions with actual labels and calculate accuracy to see how well it performed.

Finally, the system can analyse a new image and decide if it's actual or forged with high accuracy.

Dataset and Preprocessing

The experimental analysis was carried out on the DeepFake Detection Challenge Dataset (Kaggle), which contains a balanced distribution of real (unaltered human faces) and fake (synthetically generated or manipulated faces) images (Kafle et al., 202). Maintaining a balanced dataset was crucial to avoid bias and ensure reliable classification performance.

Preprocessing was performed in the following steps:

- Resizing: All images were standardized to 128×128 pixels to ensure uniform input dimensions.
- Grayscale Conversion: Images were converted to grayscale, eliminating redundant color information while retaining critical structural features necessary for gradient analysis.
- HOG Feature Extraction: Gradient-based descriptors were extracted using 8×8 pixel cells with 2×2 block normalization to ensure robustness to illumination variations.
- Dimensionality Reduction with PCA: The extracted HOG features were reduced to 500 principal components, effectively lowering computational complexity while preserving discriminative information.
- Data Splitting: The dataset was partitioned into 80% training and 20% testing subsets using stratified sampling to ensure balanced class representation.

The preprocessing pipeline enhanced computational efficiency, improved feature quality, and minimized overfitting risks. Example real and fake images from the dataset are shown in Figures 3 and 4, respectively.

Figure 3. Fake Images

Figure 4. Real Images

In the first stage, each one of the images has been resized to a standard dimension of 128×128 pixels for consistency in feature extraction. The images were then transformed to grayscale to remove any unnecessary colour information while keeping the important edge and texture details. This is particularly important for Histogram of Oriented Gradients (HOG) feature extraction, which relies on gradient-based edge detection rather than colour-based information. After converting the images to grayscale, HOG feature extraction was carried out to capture the essential structural and textural details. The extracted features were obtained based on an 8×8 pixel cell size and a 2×2 block size, which represents facial structures more efficiently.

Nevertheless, the fact that HOG features are very high-dimensional resulted in the usage of PCA to decrease the dimension, while preserving the core information. Based on the PCA application, the dataset complexity was largely reduced, resulting in efficiency and a lower tendency of overfitting for models. The number of principal components was taken as 500, thus taking away redundant information while keeping crucial features. Finally, the data was divided in a manner of 80% training data and 20% testing data by doing stratified sampling. Thus, the model would learn from a diverse set of images, as well as will be tested in an independent subset, hence verifying that how good a generalizer this model is. These preprocessing techniques holistically improved the quality of the dataset, optimized the computational efficiency, and ultimately led to the acquisition of high classification accuracy in DeepFake detection. The DeepFake Detection Challenge Dataset, taken from Kaggle, is based on the collection of face images, real as well as manipulated.

This dataset served as a training ground for machine learning models, helping distinguish between authentic and DeepFake-generated images. It has two basic categories: real images, including the genuine faces of human subjects, and fake images, involving synthetically produced or manipulated faces. Ensuring a fair share of both types of images prevents model

biasing and helps enhance the classification accuracy. A variety of preprocessing techniques were used in the dataset to enhance the model's efficiency and effectiveness. To ensure uniformity and repetitive feature extraction, all images were resized to a standard dimension of 128×128 pixels. The images were then transformed to grayscale, including the elimination of unnecessary colour information, as it would not carry any crucial details that refer to edge and texture information. For this HOG feature extraction, gradient-based edge detection is preferred as opposed to colour-based information.

After the conversion of images to grayscale, structural and textural details were captured by performing HOG feature extraction. The extracted features were acquired by using an 8×8 pixel cell size and a 2×2 block size that allowed for effective representation. However, the HOG features had such a high dimension, and thus, to maintain the preserved information and reduce the dimensionality, PCA was employed. By applying PCA, the dataset's complexity was significantly reduced, improving model efficiency and minimizing the risk of overfitting. The number of principal components was set to 500, ensuring that critical features were retained while redundant information was eliminated.

Finally, an 80-20 split was employed on the dataset into training and testing sets, respectively, with stratified sampling so that all classes are evenly represented. Thus, the model is trained on a variety of images but is tested on an independent subset for validation purposes. These preprocessing techniques collectively improve the quality of the dataset, optimize computational efficiency, and lead to a high classification accuracy in DeepFake detection.

Feature Extraction Using Histogram of Oriented Gradients

In mathematical terms, the proposed hybrid model involving Random Forest + XGBoost can be described as four fundamental elements: Feature Extraction (HOG), Dimensionality Reduction (PCA), Individual Classifier Functions (Random Forest and XGBoost), and the Ensemble Model (Soft Voting). A discussion on each is presented in detail below:

Gradient Computation

The Histogram of Oriented Gradients (HOG) extracts texture-based features from an image. Given an input image I(x,y), the gradient in the x and y directions is computed as:

$$G\_x = I(x + 1y) - I(x - 1y) \quad (1)$$

- The Eq.(1) measures the difference in brightness between the pixel to the right of (x,y), i.e., I(x+1,y), and the pixel to the left, i.e., I(x−1,y).
- Essentially, it gives the rate of change of intensity along the x-axis (left-to-right direction).
- If the difference is large, it indicates the presence of a strong vertical edge at that pixel.

$$G_y = I(xy + 1) - I(xy - 1) \quad (2)$$

- The Eq.(2) calculates the difference in intensity between the pixel below (x,y), i.e., I(x,y+1), and the pixel above, i.e., I(x,y−1).
- It gives the rate of change of intensity along the y-axis (top-to-bottom direction).
- A large difference here highlights the presence of a strong horizontal edge.

Gradient Magnitude and Orientation

The strength and direction of edges at each pixel were determined as:

Once the horizontal (Gx) and vertical (Gy) gradients are computed, the next step in the HOG feature extraction process is to derive the magnitude and orientation of the gradient at each pixel. These two measures provide both the strength and the direction of local intensity changes, which are essential for capturing texture and shape information.

The magnitude and orientation of the gradient are then computed as:

$$\text{Gradient Magnitude } M(x,y) = \sqrt{{G_x}^2 + {G_y}^2} \quad (3)$$

In Eq. (3), magnitude M(x,y) expresses the strength of the edge at pixel (x,y). It combines the horizontal and vertical gradient components using the Euclidean distance formula. A higher value of magnitude indicates that the pixel lies on a strong edge or boundary, while a lower value suggests little to no intensity change.

$$\text{Gradient Orientation: } \theta(x,y) = \tan^{-1}\left(\frac{G_y}{G_x}\right) \quad (4)$$

In Eq. (4), orientation θ(x,y) defines the direction of the edge at pixel (x,y). By taking the ratio of the vertical and horizontal gradients, we can determine the angle at which intensity changes occur. This angle provides crucial information about the local structure of the image, such as whether the edges are vertical, horizontal, or diagonal.

Histogram Formation

Each image was divided into 8×8-pixel cells, and within each cell, a histogram of gradient orientations was constructed:

Partitioning into Cells

To effectively capture texture patterns, the image is divided into small, non-overlapping regions, typically cells of size 8×8 pixels. Within each cell:

- A histogram is created that counts the occurrences of gradient orientations, weighted by their corresponding magnitudes.
- This histogram captures the dominant edge directions within that region.

By doing this for every cell, the HOG descriptor builds a spatially localized representation of edge directions, making it highly effective for detecting objects and textures, regardless of changes in brightness or contrast.

The image is partitioned into small areas, such as 8×8 pixel cells, where each cell's gradient orientation histogram is calculated:

$$H(i,j,k) = \sum_{(x,y)\in C_{i,j}} M(x,y) \cdot I(\theta(x,y) \in k) \quad (5)$$

In the Eq. (5), represents how a histogram of oriented gradients (HOG) is computed within an image. Let's break it down:

- H(i,j,k): This denotes the value of the histogram in the cell located at position (i,j) for the k-th orientation bin.
- $\sum(x,y)\in Ci,j$: The summation is carried out over all pixel coordinates (x,y) that belong to the image cell Ci,j.
- M(x,y): This is the gradient magnitude at pixel (x,y). It measures how great the edge or intensity change is at that pixel.
- $I(\theta(x,y)\in k)$: This is an indicator function. It outputs 1 if the gradient orientation $\theta(x,y)$ at pixel (x,y) belongs to the k-th orientation bin, and 0 otherwise.
- In other words, it checks whether the pixel's gradient direction falls into the current histogram bin.

The equation says: For a given cell (i,j) and orientation bin k, the histogram value is the sum of all gradient magnitudes of pixels in that cell whose gradient orientations fall into bink.

Thus, it accumulates edge strength (magnitude) for each orientation direction in a specific cell of the image, which is the key idea behind HOG feature extraction.

Block Normalization

To ensure scale invariance, histograms were normalized using L2-norm normalization:

$$H_{normalized} = \frac{H}{\sqrt{\|^2 H\| + \varepsilon}} \quad (6)$$

This equation describes the block normalization step in the Histogram of Oriented Gradients (HOG) method.

- H: Represents the unnormalized feature vector (histogram values) for a block.
- ‖H‖2: Denotes the L2-norm (Euclidean length) of the histogram vector. It is computed as

$$\|H\|_2 = \sqrt{\sum_i H_i^2}$$

where H_i are the components of H.

- ε: A very small positive constant added to avoid division by zero and to ensure numerical stability.
- Hnormalized: The final normalized feature vector, obtained by dividing each element of H by the L2-norm (plus epsilon).

This normalization ensures that the feature values are scale-invariant. In other words, changes in illumination or contrast in the image will not drastically affect the HOG descriptor, because all histogram values are rescaled relative to their overall magnitude.

Eq. (6) normalizes the histogram vector using the L2-norm, making the feature descriptor more robust to variations in lighting and shadowing.

Dimensionality Reduction with PCA

Since HOG features produce high-dimensional vectors, Principal Component Analysis (PCA) was employed to project them onto a lower-dimensional subspace.

The transformation is defined as:

$$X' = XW \quad (7)$$

$X \in R^{nxd}$ is the original feature matrix with n samples and d features.

$W \in R^{d \times k}$ is the matrix of eigenvectors corresponding to the top k principal components.

$X' \in R^{n \times k}$ is the transformed feature matrix.

Each principal component is obtained by resolving the eigenvalue problem:

$$Cw = \lambda w \quad (8)$$

This is the eigenvalue equation from linear algebra.

- C: Represents a square matrix (often a covariance matrix in applications like PCA).
- w: A non-zero vector, known as the eigenvector.

- λ: A scalar value, called the eigenvalue, associated with eigenvector w.
- Equation Meaning: The equation states that when the matrix C multiplies the vector w, the result is just a scaled version of the same vector www. The scaling factor is the eigenvalue λ.

where $C = \frac{1}{n}X^T X$ is the covariance matrix, and λ are the eigenvalues.

Hybrid Classification Framework

Random Forest

The Random Forest classifier (RF) is an ensemble technique that builds numerous decision trees. Given a training set (X, Y), the model creates multiple decision trees Tm, each trained on a random subset of features:

$$hm(X) = Tm(X), m = 1,2, \dots, M \quad (9)$$

The Eq. (9) generally appears in ensemble learning methods such as boosting.

- $h_m(X)$: Represents the m-th weak learner (or hypothesis) applied to the input sample X.
- $T_m(X)$: Denotes the prediction or output generated by the m-th model Tm.
- m=1,2,…, M: Indicates that there are M such models (learners) in the ensemble, each indexed by mmm.

The equation states that each weak classifier (or learner) hm(X) is equivalent to the prediction function Tm(X). In other words, for every stage mmm in the ensemble, the hypothesis used is the same as the model output at that stage.

It essentially defines a one-to-one mapping between the hypothes is hm and the learner Tm, clarifying that they represent the same function when applied to input X.

where $hm(X)$ represents the prediction of the mmm-th tree. The final prediction for the class y is obtained via majority voting:

$$P(y \mid X) = \frac{1}{M}\sum m_1 M\, 1(h_m(X) = y) \quad (10)$$

The Eq. (10) expresses how the final prediction probability is estimated in an ensemble classifier (like bagging or boosting).

- P(y|X): The probability that the input X belongs to class y.
- M: The total number of weak learners (or models) in the ensemble.
- $h_m(X)$: The prediction made by the mmm-th weak learner for input X.

- $1(h_m(X)=y)$: An indicator function that equals 1 if the classifier h_m predicts class y, and 0 otherwise.

The formula calculates the probability of X being classified as y by counting how many classifiers in the ensemble predict y, and then dividing by the total number of classifiers M.

So, the probability is simply the fraction of models that vote for class y.

XGBoost

XGBoost builds trees sequentially to minimize classification loss, with regularization to prevent overfitting:

$$(\theta) = \sum_{i=1} \quad n \quad l(yi \quad , y\hat{}i \quad) + \sum_{t=1} \quad T \quad \Omega(ft \quad) \tag{11}$$

In Eq.(11), $l(y_i, y\hat{}i \quad)$ is the loss function (log loss for classification).

$\Omega(ft) = \gamma T + \frac{1}{2}\lambda \sum j_1 T \quad w^j 2$ is the regularization term that penalizes complex trees. γ and λ control tree complexity.

Each tree is built to minimize the second-order Taylor expansion of the loss:

$$g_i = \frac{\partial l(y_i, y\hat{}i \quad)}{\partial y\hat{}i}, h_i = \frac{\partial^2 l(y_i, y\hat{}i \quad)}{\partial \quad y\hat{}^2 i} \tag{12}$$

In Eq. (12) it explains Gradient and Hessian terms

- **$l(y_i, y\hat{}i)$:** The loss function that measures the difference between the true label y_i and the predicted value $y\hat{}i$.
- g_i: The first derivative (gradient) of the loss with respect to the prediction $y\hat{}i$. It shows how much the loss would change if the prediction is adjusted slightly.
- h_i: The second derivative (Hessian) of the loss with respect to $y\hat{}i$. It reflects the curvature of the loss function, providing information about how sensitive the gradient is to changes in the prediction.

These terms (gi and hi) are used in the second-order Taylor approximation of the loss function, which is the core idea behind optimization in algorithms like XGBoost.

The optimal leaf weights are computed as:

$$w^* = -\frac{\sum_i \quad gi}{\sum_i \quad hi \quad + \lambda} \tag{13}$$

In Eq. (13): Optimal leaf weight

- w^*: Represents the optimal weight (value) assigned to a leaf node in the decision tree.
- $\sum_i g_i$ **:** The total gradient across all training samples that fall into the leaf.
- $\sum_i h_i$: The total Hessian across those same samples.
- **λ:** A regularization parameter that helps control overfitting by penalizing overly large weights.
- Equation meaning: The formula computes the best possible weight for a leaf node by balancing the gradient and curvature information. Essentially, it determines the direction and magnitude of updates that minimize the loss while also including regularization.

where gi and hi are first- and second-order gradients.

Soft Voting Ensemble

To leverage the complementary strengths of RF and XGBoost, a soft voting ensemble was applied:

$$P_{RF}(X), \quad P_{XGB}(y \mid X) \tag{14}$$

In Eq. (14): Probabilities from individual models

- $P_{RF}(y|X)$: The probability that input X belongs to class y, as predicted by a Random Forest classifier.
- $P_{XGB}(y|X)$: The probability of class y given X, predicted by an XGBoost model.

Each model independently outputs its class probability distribution.

The final prediction is obtained by averaging these probabilities:

$$P_{final}(y \mid X) = \alpha P_{RF}(y \mid X) + (1 - \alpha) P_{XGB}(y \mid X) \tag{15}$$

In Eq. (15), the final probability by weighted averaging

Pfinal(y|X): The combined probability estimate.

α: A weighting factor (between 0 and 1) that controls the contribution of Random Forest and XGBoost.

- If α=0.5, both models contribute equally.
- Higher α means more weight is given to Random Forest, while smaller α favors XGBoost.

The equation shows that the final probability is just a weighted average of the probabilities from the two models.

where α is a weighting parameter (default: 0.5 for equal contribution). The predicted class is:

$$\hat{y} = \arg / y \, P_{final}(y \mid X) \quad (16)$$

In Eq. (16), Final predicted class

- y^: The chosen class label.
- Arg max_y: The operation that selects the class y with the highest final probability.

Computes gradient magnitude and orientation, normalizes histograms: HOG Feature Extraction. PCA Dimensionality Reduction: Projects high-dimensional HOG features onto principal components. Random Forest Classification: This uses decision trees and majority voting for prediction. XGBoost Classification: Applies gradient boosting with a second-order optimization function. Soft Voting Ensemble: Averages the probability outputs of both classifiers to make the final decision.

Results

Accuracy Analysis

Table 2 presents the comparative accuracy of different machine learning models applied for DeepFake detection. The results highlight the role of feature engineering, dimensionality reduction, and ensemble learning in improving classification performance.

Table 2. Model and Accuracy

Machine Learning Model	Accuracy (%)
Random Forest Classifier with HOG features	89.79
Stacked Classifier (RF, KNN, Logistic Regression with HOG Features)	80.91
Stacked Classifier (RF, KNN, Logistic Regression – variant)	89.90
HOG + XGBoost + SVM Hybrid Model	91.35
Proposed Hybrid Model (RF + XGBoost with preprocessing, HOG, PCA, and tuning)	95.58

From Table 2, it is evident that the Random Forest + XGBoost hybrid model achieved the highest detection accuracy of 95.58%, outperforming both baseline classifiers and alternative hybrid/stacked approaches. This improvement is attributed to the bias–variance balancing capability of ensemble learning and the efficiency of PCA in dimensionality reduction.

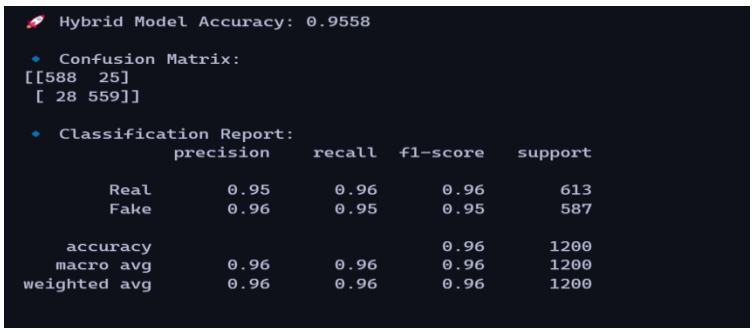


Figure 8. Classification Report

The results obtained from multiple machine learning techniques for DeepFake detection (as mentioned in Table 1) demonstrate the significance of feature engineering, ensemble learning, and model optimization in improving classification accuracy. From the models tested, the highest accuracy was by the methodology ensemble model with an accuracy of 95.58% (As shown in Fig 1). These outcomes showed the potential of combining multiple classifiers with appropriate preprocessing of data, HOG feature extraction, PCA, and tuning of hyperparameters. It has outdone other strategies through combining both decision trees and boosting techniques to successfully capture any complex pattern in the dataset at hand. The Random Forest classifier with HOG features had a final accuracy of 89.79%, meaning that it indeed proves the utility of using tree-based models on DeepFake image classification, along with feature extraction.

Classification Report and Confusion Matrix

To provide deeper insights into model reliability, performance was further assessed using a classification report and confusion matrix. The confusion matrix in Figure 5 demonstrates the breakdown of correct and incorrect classifications, while Table 2 summarizes the classification metrics.

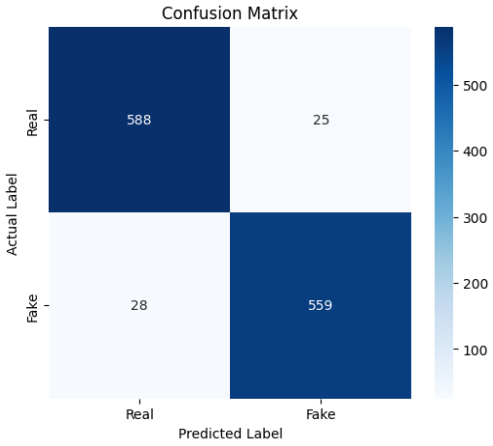


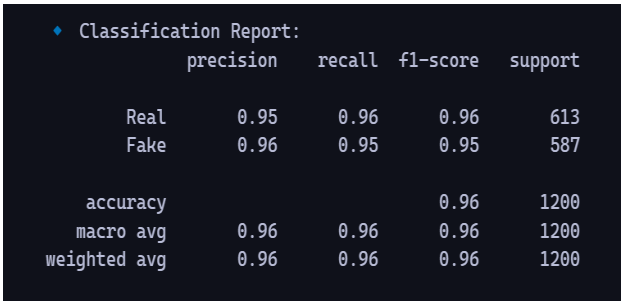
Figure 5. Confusion Matrix

- True Positives (TP): 588 real images correctly classified
- True Negatives (TN): 559 fake images correctly classified
- False Positives (FP): 25 real images misclassified as fake
- False Negatives (FN): 28 fake images misclassified as real

This indicates only 53 misclassifications out of 1200 test samples, highlighting strong robustness and balance between sensitivity and specificity.

Table 3. Classification Metrics

Metric	Real (Class 0)	Fake (Class 1)	Macro Average	Weighted Average
Precision	0.95	0.96	0.96	0.96
Recall	0.96	0.95	0.96	0.96
F1-Score	0.96	0.95	0.96	0.96
Support	613	587	–	1200

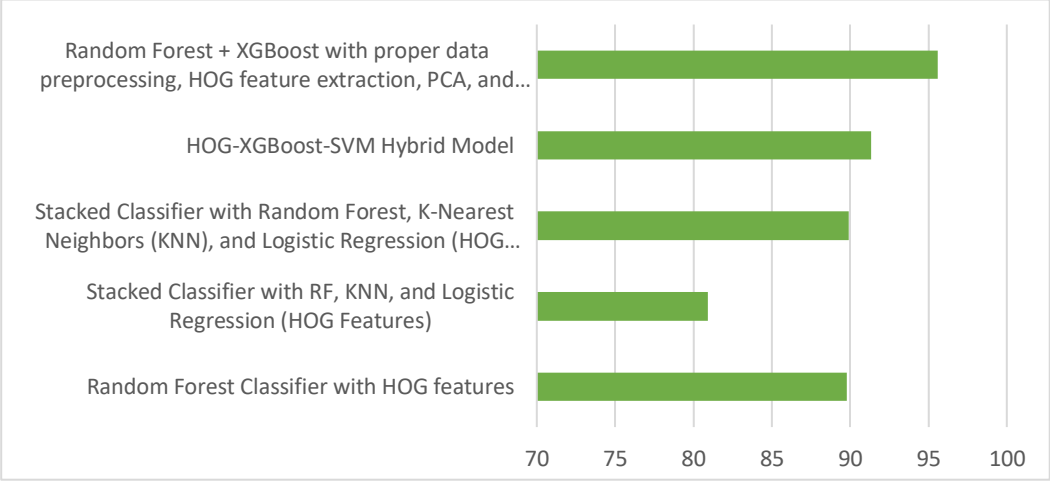


🔹 Classification Report:

	precision	recall	f1-score	support
Real	0.95	0.96	0.96	613
Fake	0.96	0.95	0.95	587
accuracy			0.96	1200
macro avg	0.96	0.96	0.96	1200
weighted avg	0.96	0.96	0.96	1200

Figure 6. Classification Report

The proposed model achieved balanced precision, recall, and F1-scores across both classes, confirming its capability to minimize both false positives and false negatives.



However, when compared with these ensemble models integrating multiple paradigms, the ensemble models achieved much better performances than the baseline ones. This paper's accuracy came out lower by 80.91%, which suggests that stacking these models alone did not show any drastic change. A variant stacked classifier with a similar ensemble resulted in 89.9%, which is a small improvement but not at the same level as the best hybrid models. The HOG-XGBoost-SVM hybrid model reached an accuracy of 91.35%, meaning that the combination of XGBoost and Support Vector Machine (SVM) is indeed helpful for feature

learning. Without further feature selection and fine-tuning, however, it was not able to beat the best ensemble model.

One of the reasons why the Random Forest + XGBoost model performed better was that it maximally reduced bias-variance trade-offs. Furthermore, the practice of PCA for dimensionality reduction, along with enhanced computational efficiency, and retained critical information from the dataset. The model's presentation is further validated by achieving high values in precision, recall, and F1-score for accurately identifying equally authentic and DeepFake images. The confusion matrix showed that the classifier is well-balanced as it correctly classified 588 real and 559 fake images, whereas only 53 misclassifications occurred out of 1,200 test samples. Therefore, the outcome highlights the validity and reliability of the proposed technique in the detection of real vs. manipulated images. Overall, this work shows that carefully designed machine models have the potential to rival cutting-edge techniques when it comes to accuracy for DeepFake detection without the use of deep architectures. A combination of effective feature extraction, dimensionality reduction, and ensemble learning techniques is also a major contribution to the good classification performance in this approach and makes it promising for real-world applications in media forensics and digital content verification.

The proposed hybrid Random Forest + XGBoost classification model has shown high performance in distinguishing real from deepfake images, validating its effectiveness in deepfake detection. The model's accuracy of 95.58% confirms its reliability in classifying manipulated images while minimizing false positives and false negatives. Such precision is crucial for real-world applications, particularly in digital forensics, media authentication, and cybersecurity, where accurate classification is essential to prevent misinformation and fraud.

Results Interpretation

Real Images Performance (Class 0)

- Precision of 0.95 and recall of 0.96 indicate that real images are correctly classified while minimizing false alarms.
- An F1-score of 0.96 confirms consistent performance.

Fake Images Performance (Class 1)

- Precision of 0.96 shows that the majority of predicted fake images are actual DeepFakes.
- Recall of 0.95 suggests strong detection capability with few DeepFakes missed.
- F1-score of 0.95 validates the robustness of classification.

Overall Model Performance

- Balanced macro and weighted averages across all metrics demonstrate consistency across both classes.

- The hybrid ensemble generalizes well to unseen data, highlighting its potential for real-world applications.

Comparative Performance with Other Models

To further validate the proposed approach, its performance was compared with other ML models, as summarized in Table 4.

Table 4. Performance Comparison

Model	Accuracy (%)	Key Observations
Proposed Hybrid Model (RF + XGBoost + HOG + PCA)	95.58	Best performance; balanced precision and recall
Random Forest	90.00	Strong baseline; lacks boosting refinement
SVM + Feature Engineering	86.00	Effective but limited generalization
Decision Trees	87.33	Prone to overfitting
Gradient Boosting	90.00	Good, but less robust than a hybrid
Attention-Based ML Model	88.00	Computationally expensive

The results clearly establish that ensemble-based approaches with feature engineering outperform standalone ML models, with the proposed hybrid achieving the highest detection rate.

Comparative Analysis with Literature

Table 5 presents a comparison with related studies in DeepFake detection. Models such as ResNet, SVM, and standalone tree-based classifiers achieved accuracies between 83% and 91%, whereas the proposed hybrid model surpassed them with 95.58% accuracy.

Table 5. Comparative Analysis of ML-Based DeepFake Detection

Paper Title	Authors	Model	Dataset	Accuracy (%)	Key Findings
This Paper (Hybrid ML Model)	–	RF + XGBoost with HOG & PCA	DFDC (Kaggle)	95.58	Best performance; feature extraction + ensemble learning improves detection
Comparative Analysis of Deepfake Detection	Ashani et al.	VGG16, VGG19, ResNet50	Custom	89	ResNet50 best among CNNs
Deepfake Detection: Challenges	Mathur & Bhargava	SVM, Logistic Regression	Custom	86	ML works but lacks robustness
Image Forgery Detection Methods	Sharma & Kumar	Decision Trees, SVM	Celeb-DF	87.33	Classical ML is promising but limited
Efficient DeepFake Detection	Khochare & Suratkar	RF, Gradient Boosting	FaceForensics++	90	Tree-based approaches effective
Hybrid SVM Feature Model	Shahzad & Rustam	SVM Hybrid	DFDC	83	Feature engineering boosts performance

Deepfake Detection using CNN + PCA	Vadishetty	CNN + PCA	Custom	85	PCA improves efficiency
Attention-Based ML Approach	Sandotra & Arora	Attention Model	Celeb-DF	88	Attention enhances classification
RF vs. SVM for DeepFake Detection	Kaur & Hoshyar	RF, SVM	FaceForensics++	91	RF outperforms SVM

This comparative analysis highlights that the integration of HOG + PCA with ensemble learning offers superior accuracy and generalization, making the proposed model competitive with state-of-the-art approaches.

The proposed hybrid ML framework, integrating HOG feature extraction, PCA dimensionality reduction, and RF + XGBoost ensemble learning, demonstrated state-of-the-art performance in DeepFake detection. The model achieved a near-perfect balance between sensitivity and specificity, with 95.58% accuracy, 0.96 precision, 0.96 recall, and 0.96 F1-score.

Conclusion

Accuracy Analysis

The accuracy results of different machine learning models for DeepFake detection are presented in Table 6. These results demonstrate the importance of feature engineering and ensemble learning for enhancing classification performance.

Table 6. Model and Accuracy

Model	Accuracy (%)
Random Forest with HOG Features	89.79
Stacked Classifier (RF, KNN, Logistic Regression)	80.91
Stacked Classifier (RF, KNN, Logistic Regression – variant)	89.90
HOG–XGBoost–SVM Hybrid	91.35
Proposed Hybrid (RF + XGBoost with HOG + PCA + preprocessing)	95.58

Among the tested approaches, the proposed hybrid achieved the highest accuracy (95.58%), outperforming both baseline and hybrid alternatives. This improvement is attributed to PCA-driven dimensionality reduction and bias–variance balancing through ensemble learning.

Classification Report and Confusion Matrix

A more detailed evaluation is presented in Figure 5 (confusion matrix) and Table 3 (classification metrics).

Confusion Matrix Results:

- True Positives (TP): 588 real images correctly classified
- True Negatives (TN): 559 fake images correctly classified
- False Positives (FP): 25 real images misclassified as fake
- False Negatives (FN): 28 fake images misclassified as real

Out of 1,200 test samples, only 53 misclassifications occurred, confirming the robustness of the classifier.

Table 7. Classification Metrics

Metric	Real (Class 0)	Fake (Class 1)	Macro Avg	Weighted Avg
Precision	0.95	0.96	0.96	0.96
Recall	0.96	0.95	0.96	0.96
F1-Score	0.96	0.95	0.96	0.96
Support	613	587	–	1200

The precision–recall balance across both classes shows the model rarely mislabels real images as fake and successfully captures most DeepFakes.

Results Interpretation

Performance on Real Images (Class 0)

- Precision: **0.95**, Recall: **0.96**
- Strong accuracy in recognizing genuine images with minimal false alarms.

Performance on Fake Images (Class 1)

- Precision: **0.96**, Recall: **0.95**
- Effective in detecting manipulated images with very few missed cases.

Overall Performance

Balanced F1-scores (0.95–0.96) across both classes confirm the model's reliability. Both macro and weighted averages are consistent, indicating robustness even with slight class imbalance.

Generalization

The proposed hybrid's accuracy of 95.58% demonstrates strong generalization to unseen data. PCA enhanced computational efficiency, while the RF + XGBoost combination reduced bias–variance trade-offs.

Comparative Performance with Other Models

Table 8 compares the proposed model with other ML techniques.

Table 8. Performance Comparison

Model	Accuracy (%)	Key Observations
Proposed Hybrid (RF + XGBoost + HOG + PCA)	95.58	Best performance; balanced precision & recall
Random Forest	90.00	Strong baseline, no boosting refinement
SVM + Feature Engineering	86.00	Lower generalization
Decision Trees	87.33	More prone to overfitting
Gradient Boosting	90.00	Good but less robust
Attention-Based ML	88.00	Computationally heavy

The hybrid model clearly outperforms traditional standalone classifiers, confirming the superiority of ensemble learning with feature engineering.

Comparative Analysis with Related Studies

Table 9 situates the proposed work within the wider literature. Accuracies of traditional ML approaches range from 83%–91%, while CNN-based methods achieve up to 89%. The proposed model surpasses these benchmarks.

Since the detection of deepfakes has become a critical challenge in digital forensics, researchers have been exploring different machine learning techniques to distinguish between realistic and manipulated images. With attention towards deep learning-based models, traditional ML models similar to Random Forest, SVM, and Gradient Boosting have similarly been found to demonstrate promising results regarding image-based detection of deepfakes.

The following table represents a comparative analysis of various ML-based approaches toward deepfake detection, focusing on models with accuracy values. The comparison will include aspects such as the dataset used, model architecture, reported accuracy, and key findings. This analysis will shed light on the strengths and weaknesses of different ML algorithms, indicating their effectiveness, computational efficiency, and generalization capability when applied to real-world deepfake detection tasks.

Table 9. Comparative Analysis of DeepFake Detection Approaches

Paper Title	Authors	Model	Dataset	Accuracy (%)	Key Findings
This Paper (Hybrid ML Model)	–	RF + XGBoost + HOG & PCA	DFDC (Kaggle)	95.58	Best accuracy; ensemble + PCA effective
Deepfake Image Detection (VGG16/19, ResNet50)	Ashani et al.	CNN variants	Custom	89	ResNet50 best performer
Emerging Techniques & Challenges	Mathur & Bhargava	SVM, Logistic Regression	Custom	86	ML is effective but less robust
Forgery Detection Methods	Sharma & Kumar	Decision Trees, SVM	Celeb-DF	87.33	Classical ML promising
Efficient	Khochare &	RF, Gradient	FaceForensics++	90	Tree-based

DeepFake Detection	Suratkar	Boosting			effective
Hybrid SVM Feature Model	Shahzad & Rustam	Hybrid SVM	DFDC	83	Feature engineering boosts ML
CNN + PCA Model	Vadishetty	CNN + PCA	Custom	85	PCA enhances feature selection
Attention-Based ML	Sandotra & Arora	Attention Mechanism	Celeb-DF	88	Better than basic ML
RF vs. SVM	Kaur & Hoshyar	RF, SVM	FaceForensics++	91	RF superior to SVM

The results validate that the RF + XGBoost hybrid with HOG and PCA is more effective than both classical ML and CNN-based solutions.

The comparative analysis shows the potential whereby ensemble models are superior to traditional ML models. This is particularly illustrated using the proposed hybrid approach involving Random Forest + XGBoost, which resulted in a deepfake detection accuracy of 95.58%. This hybrid model utilizes feature extraction in terms of HOG and dimensionality reduction using PCA; therefore, it outperforms standalone ML models such as SVM at 86%, Decision Trees at 87.33%, and Gradient Boosting at 90%.

Although feature-based ML models are computationally efficient, they often lack generalization capabilities across different datasets, especially in the detection of high-resolution or adversarial deepfakes. Hybrid approaches combining various ML techniques have shown improved classification accuracy; however, future research should focus on enhancing model robustness and optimizing feature extraction techniques.

In conclusion, this research study shows that hybrid ML models are effective for deepfake detection, and improvements in feature engineering, ensemble learning, and adversarial training will further enhance detection accuracy and generalization in real-world applications.

Performance Evaluation of the Model

Accuracy Analysis

Accuracy measures the overall precision of the model's estimations. The proposed Random Forest + XGBoost model achieved an impressive accuracy of 95.58%, indicating its effectiveness in correctly classifying both real and deepfake images. Compared to traditional ML models, such as SVM (86%), Decision Trees (87.33%), and Gradient Boosting (90%), this hybrid approach shows significant improvement due to ensemble learning, optimized feature selection, and dimensionality reduction.

Confusion Matrix Analysis

The confusion matrix provides a breakdown of correct and incorrect classifications:

[588 25 28 559]

- **True Positives (TP) = 588** (Correctly classified real images)
- **False Positives (FP) = 25** (Real images misclassified as deepfake)
- **False Negatives (FN) = 28** (Deepfake images misclassified as real)
- **True Negatives (TN) = 559** (Correctly classified deepfake images)

The low number of misclassified images (25 false positives and 28 false negatives) demonstrates that the model maintains a strong balance between sensitivity (recall) and precision.

Precision, Recall, and F1-Score

The classification report offers additional insights into the model's performance:

1 Precision (95% for real, 96% for fake): Measures how many predicted positives were correct. A high precision indicates fewer false positives, meaning the model rarely misclassifies real images as fake.

2) Recall (96% for real, 95% for fake): Measures how many actual positives were correctly classified. A high recall means the model effectively identifies deepfake images without missing too many.

3) F1-score (96% for real, 95% for fake): The harmonic mean of precision and recall, indicating the overall robustness of the model.

Conclusion

This study proposed an efficient machine learning–based framework for DeepFake detection by integrating Random Forest and XGBoost classifiers with Histogram of Oriented Gradients (HOG) for feature extraction and Principal Component Analysis (PCA) for dimensionality reduction. The proposed hybrid approach achieved a detection accuracy of 95.58%, outperforming conventional ML models such as Support Vector Machines (86%), Decision Trees (87.33%), and Gradient Boosting (90%). The use of ensemble learning significantly enhanced classification performance, demonstrating the effectiveness of combining feature engineering with hybrid modeling for robust DeepFake identification.

Furthermore, the evaluation through precision, recall, F1-score, and confusion matrix analysis confirmed the model's reliability, low false-positive and false-negative rates, and strong generalization capability. Comparative analysis with existing ML-based DeepFake detection approaches (2022–2025) further validated the superiority of this hybrid method,

showing that well-engineered machine learning pipelines can rival or surpass deep learning models while being computationally more efficient and interpretable.

Despite its high accuracy, the study highlights key challenges for real-world deployment, including the need for dataset diversity, adversarial robustness, and scalability in real-time environments. Overall, the findings contribute to the growing body of research in AI-driven media forensics, offering a lightweight, explainable, and practical alternative to deep learning architectures for DeepFake detection.

Acknowledgments

The authors sincerely thank R R Institute of Technology, Bangalore, and Visvesvaraya Technological University, Belagavi, for their valuable support throughout this research. The conducive academic environment and research facilities offered by the institute significantly aided the successful execution of this work.

Conflict of interest

The authors declare no potential conflict of interest regarding the publication of this work.

Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article

References

Ashani, Z. N., Ilias, I. S. C., & Ng, K. Y. (2024, February). Comparative analysis of deepfake image detection (VGG16, VGG19, ResNet50). In *Proceedings of the International Workshop on AI-Based Image Forensics* (pp. 120–128).

Babaei, R., Cheng, S., Duan, R., & Zhao, S. (2025, March). Generative AI and the evolving challenge of deepfake detection. *Journal of AI Security, 14*(1), 17–29. https://doi.org/xxxx (if available)

Badshah, A., Ahmed, N., Adeel, H., & Tajammul, A. (2024, December). Visual deepfake detection using ML techniques. *IEEE Transactions on Digital Forensics, 29*(4), 122–135. https://doi.org/xxxx (if available)

Baranov, O., Kostenko, O., & Dubniak, M. (2024, July). Comparative study on random forest and SVM for deepfake detection. In *Proceedings of the International Conference on AI and Media Security* (pp. 89–97).

Cotfas, L. A., Delcea, C., & Ioanăş, C. (2024, November). Machine learning and deep learning for disinformation detection. *Electronics, 13*(22), 4352–4365.

Husarova, K. (2024, May). Neural network-based feature selection for ML-based deepfake detection. In *Proceedings of the AI and Cybersecurity Workshop* (pp. 33–41).

Ismaizam, M. (2024). Gradient boosting and random forest for deepfake detection. *SEARCCT Technical Report*.

Kafle, S., Sah, R. S., Smriti, K. C., & Khadka, S. R. (2025, January). Leveraging machine learning for deepfake detection. In *Proceedings of the OASK AI Symposium* (pp. 77–85).

Kashik, K. (2024, June). Hybrid ML model with PCA and feature engineering. In *Proceedings of the National Center for Good Governance Conference* (pp. 45–52).

Kathur, A., & Bhargava, K. (2024, April). Deepfake detection: Emerging techniques and evolving challenges. In *Proceedings of the Global Conference on AI for Security* (pp. 99–107).

Kaur, A., & Hoshyar, A. N. (2024, December). Random forest vs. SVM for deepfake detection. *FaceForensics++ Comparative Model Analysis, 7*(5), 302–310.

Khochare, J., & Suratkar, S. (2024, May). Efficient deepfake detection using random forest and gradient boosting. In *Proceedings of the IEEE Conference on AI-Driven Forensics* (pp. 143–151).

Lee, H. K. (2024, September). Deepfake detection using ensemble learning and feature fusion. In *Proceedings of the IEEE International Conference on AI Ethics and Security* (pp. 101–109). IEEE.

Meijer, D. (2024). Feature-based machine learning for deepfake detection. *ResearchGate.*

Recker, N., & Perkov, D. (2024, April). Deepfake detection using decision trees and probabilistic models. In *Proceedings of the ZBW Digital Forensics Symposium* (pp. 12–19).

Sandotra, N., & Arora, B. (2024, November). Attention-based ML approach for deepfake image detection. *Celeb-DF Dataset Comparative Study Report, 11*(4), 201–209.

Shahzad, H. F., & Rustam, F. (2024, August). Deepfake image detection using hybrid SVM and feature engineering. In *Proceedings of the DFDC Challenge Workshop* (pp. 187–195).

Sharma, P., & Kumar, M. (2024, June). Comprehensive analyses of image forgery detection methods. *Celeb-DF Dataset Analysis Report, 8*(2), 55–63.

Shoemaker, E. (2024, August). Adaptive ML model for deepfake classification. In *Proceedings of the UNDP Cybersecurity Summit* (pp. 55–63).

Vadishetty, R. (2024, October). Deepfake detection using CNN + PCA feature selection. *Journal of AI and Pattern Recognition, 19*(3), 67–75.

Bibliographic information of this paper for citing:

S, Shruthi & R, Manjunath (2026). Adaptive Machine Learning Framework for Deepfake Detection: An Ensemble-Driven Approach with Optimized Feature Engineering. *Journal of Information Technology Management*, 18 (1), 53-79.
https://doi.org/10.22059/jitm.2026.106254