International Journal of Computing and Digital Systems
2024, VOL. 17, NO. 1, 1–10

http://dx.doi.org/10.12785/ijcds/XXXXXX

Enhancing Fake Face Detection with Meta-Learning and Ensemble Methods

Suman Saha*, Md. Zunead Abedin Eidmum and Bakhtiar Muiz

Department of IoT and Robotics Engineering, Bangabandhu Sheikh Mujibur Rahman Digital University, Bangladesh, Gazipur, Bangladesh

Abstract: Nowadays, facial recognition technology has improved significantly, greatly enhancing security, personalized customer experiences, and reliable biometric verification. But the emergence of deepfake technology, where AI can generate images that look very real, is a concerning issue. Deepfakes have been detected primarily through slow and unreliable visual clues. Currently, image processing and deep learning techniques are being used to identify fake faces quickly and accurately. Identifying deepfakes early is imperative to prevent their abuse and enhance security. IoT devices can rely on deep learning algorithms, which offer real-time data analysis and precise fake face detection. These algorithms use large amounts of data and sophisticated neural networks, becoming extraordinarily accurate at distinguishing real from fake images. This paper proposes a model that uses a meta-learning and ensemble approach for deepfake face detection. Atfirst, we have implemented five different deep learning models: EfficientNetV4, MobileNetV2, ResNet50, NASNetMobile, and DenseNet. These models provide detection accuracies of 89%, 76%, 79%, 83%, and 84%, respectively. Subsequently, we applied stacking and blending with meta-learning, using a random forest classifier as the meta-learner. Stacking achieved 87.40% accuracy, while blending reached 92.46%, demonstrating the effectiveness of our approach for fake face detection. The high accuracy of the proposed system enhances security and ensures applicability on various devices, thereby enabling large-scale security deployment.

Keywords: Fake face detection, Image processing, Deep-learning, Ensemble Approach, Meta learning.

1. INTRODUCTION

Deepfake technology helps to identify fake faces that may indicate potential abuse or security threats. Analyzing facial features, expressions, and movements is crucial, as these elements can provide significant insights in detecting deepfakes. But by knowing what to look for, we can identify fake content as early as possible, take measures counteracting it, and create more robust security frameworks. The detection of deepfakes has come a long way from the days when it was entirely up for human inspection to now very sophisticated automated systems. Early on, experts would scrutinize images for signs they are fakes. But with the advancement of deepfake technology, we require advanced detection methodologies [1]. Currently, we employ different sophisticated techniques ranging from deep learning algorithms to state-of-the-art image processing that process massive amounts of data and employ complex neural networks in order to more accurately differentiate between real and fake images. The integration of machine learning, image processing, sensor networks, and remote sensing has significantly advanced deepfake detection [2] and greatly improved its effectiveness. These technologies can automatically detect deep fakes in real-time by evaluating images, data, and patterns. This dramatically heightens safety, and we can find deepfake contents more efficiently and safely.

Due to the advances in deepfake technologies, developing advanced detective technologies is essential. Researchers and engineers are continually developing detection algorithms to remain ahead of potential adversaries. This improves the capability to detect tiny changes in facial features, facial expressions, and movements that can reveal the existence of a deepfake. Sustainable innovation in this area is critical to guaranteeing the integrity and safeguarding of digital content. Effective deepfake detection also impacts societal trust in digital media[3]. Given the surge of social media and online content, believing what we see is more critical than ever. These deepfake detection tools help to keep the trust intact as they prove whether the contents are genuine from a real person or fake images and voices have been misused to deceive you. Journalism also benefits from this because having an eyewitness account can significantly impact the direction of

*Corresponding Author
E-mail address: suman0001@bdu.ac.bd, 1801031@iot.bdu.ac.bd, 1801045@iot.bdu.ac.bd

public opinion and policy decisions. Legal safeguards also increasingly focus on deepfake detection. In cases where deepfakes can be used to generate visual fake evidence, the importance of auto-verification in a courtroom or during an investigation is apparent. In a country based on law, the legal community can use these tools to detect visual materials to ensure justice combined with facts. Also, the ethical issues of deepfakes with more and more great work requires reliable detection. Deepfakes can be used maliciously for creating non-consensual explicit material or spreading misinformation. Better research and development on the detection of these darker sides makes us capable of wiping out these unethical clandestine uses, which not only fake the data insights but equally harm disguisedly to an individual. Developing those tracking mechanisms is a considerable digital advancement in image security. It has come a long way, from expert manual inspections to the most recent deep learning algorithms. Embedding remote sensing, image processing, sensor networks, and machine learning provides robust systems that can detect in real-time and in an automated way. As deepfake technology continues to mature, we will need more effective detection means to preserve an authentic representation of digital content and protect against threats. Such abuse affects us all, and readers will benefit from seeing exactly how much there is at stake in keeping trust in digital media, firming up legal validity, and facing deepfake abuse - morally with the installation.

In this study, we developed a model combining deep learning and ensemble techniques to enhance deepfake detection. After training various deep learning models on a large image dataset, including EfficientNetv4, MobileNetV2, ResNet50, NasNetMobile, and DenseNet. The use of stacking and blending with a random forest meta-learner boost up the accuracy in fake face detection for enhanced security deployment. The significant contributions of this paper are summarized below:

The significant contributions of this paper are summarized below:

- We studied existing research on deepfake technology and identified limitations, including accuracy issues and a lack of transparency.
- We have analyzed different deep learning model as base model performance for detecting real and fake faces.
- We proposed a hybrid model using meta-learning and an ensemble approach to enhance the performance of fake face detection.
- Finally, we evaluated the performance of the proposed model using various metrics, including accuracy, precision, recall, and F1 score, and compared it with other existing models.

The remaining portions of this paper are outlined as follows. Section 2 provides literature review on the use of deep learning models for fake face detection. Section 3 describes the proposed procedures for the identification of fake face. Section 4 highlights the experimental results and finally section 5 concludes the paper.

2. Related Works

Advancements in machine learning-based deep learning (DL) offer promising solutions for identifying deep fake images, thereby reducing the spread of false information and improving the reliability of online content. In this section, we discuss some of the existing studies related to identifying fake images.

Neves et al. [4] addresses the challenge of detecting GAN-generated synthetic facial images, which are increasingly realistic and pose significant risks for misuse. Here, author presented a new way to clean GAN ”fingerprints” from generated images with an autoencoder which detect as human made images while retaining the quality of the image. The paper introduced FSRemovalDB, a new dataset of GAN-fractured images without fingerprints to highlight the importance of building protections into spoof detection systems with respect to state-of-the-art face spoofing techniques. Finally, author shows a detailed comparative analysis of the existing spoofing detection algorithms, based on their accuracy to detect whole face synthesis manipulations. Suganthi et al.[5] introduced a deep learning model for deep by utilizing the FF-LBPH-DBN method, which is used for the precision of deepfake image detection. Here, author evaluates the performance of the proposed method using various performance evaluation metrics i.e. quality, accuracy, error detection rate, sensitivity and specificity. The authors also summarized the effectivity of FF-LBPH-DB technique to detect deepfake images and enabled a base for further works that can include extending it to different classifiers.

In [6], Wang et al. proposed local binary pattern (LBP) and ensemble modeling on fake face detection. The proposed architecture is build on a texture feature learning LPB-Net and an ensemble of some single models like ResNet and Gram-Net. The method aims at combatting the increased photorealism of GAN-generated fake faces that represent an imminent social peril. This study showed that LBP-Net is much better and robust in terms of detection accuracy with different blurring, brightness change and image-augmentations. The authors also demonstrated that the proposed approach outperformed other traditional methods, making significant advances in the classification of fake faces with a solution that is both accurate and robust against image manipulations. Baek et al. [7] introduced an innovative deepfake detection method leveraging generative adversarial ensemble learning. They proposed to train multiple discriminators and generators giving more discrimination power for improving the detection model. The approach used multiple discriminators to not only collaborate on the recognition of synthetically

generated face images but also provide feedback between generators and discriminators and across multiple models as well. Author demonstrated the efficiency of their method on FaceForensics++ dataset and compare it to existing detection approaches and find that our approach gives better results. This new arrangement is intended to promote the creation and refinement of robust deepfake detection models. Gbemisola et al in [8] proposed a an ensemble model by merging different schemes including convolutional neural networks (CNNs) and support vector machines (SVMs). The research underscores that ensemble learning techniques, in particular random forests and neural networks, can dramatically improve precision in spoof detection and achieved significant performance improvements compared to existing methods. Sharma et al. [9] proposed an GAN-CNN ensemble model in deepfake scenarios. This study overcomes catastrophic forgetting, by integrating architectural features of Generative Adversarial Networks (GANs) with Convolutional Neural Networks (CNNs), along with generative replay methods and continuous lifelong learning strategies to improve detection. At fine grain level, the model has not only achieved high accuracy rates, but also high performance of precision, recall and F1 scores overall, which makes it an efficient solution for real world application of detecting deepfake images on social media platforms and contributes as a powerful addition to existing research literature.

Rana et. al [10] introduced an effective mechanism for limiting a strong offense area of multimedia i.e., deep illustrated learning. In this paper, the authors presents a novel combination of various deep learning models including XceptionNet, ResNet101, InceptionResNetV2, MobileNet, InceptionV3, DenseNet121 and DenseNet169 to work as one composite classifier to boost detection performance. These base models are trained which in turn are used to train a meta-learner Deepfake Classifier (DFC) with the methodology proposed. It achieves a higher detection performance with this architecture, obtaining 99.65% accuracy, and an AUROC score of 1.0. This implies that the detection of deep fakes is paramount to ensure digital security and privacy as underlying tools become easier to access. Compared to existing methods, this work achieved superior performance which enables the development of fast-reacting, robust deepfake detection systems. Mansourifar et al. [11] presented a method of detecting synthesised fake faces using GANs and one-shot learning techniques along with scene understanding models. The proposed method successfully differentiate genuine faces from fake faces by extracting objects from face images and generating a sparse vector according to detected parts. Experiments testifying to the effectiveness of this technique in comparison to the traditional methods such as color texture descriptors (CMD) and DeepFD especially on higher-grade fake faces where objects out of context have been incorporated. Few-Shot Learning (based on meta-learning) is performed to counterbalance the low data issue in real vs. synthesized face recognition and able to cope with emerging generative models as well. Lin et al. [12], utilizing the meta-learning paradigm, aim to augment existing methods that are confined by specific conditions. The proposed method obtained up to 35.4-127.2% of IoU improvement and 2.0-48.9% AUC advantage in detecting DeepFakes, Face2Face, FaceSwap and NeuralTextures with few samples by weight refining. By comparing with previous studies, the proposed method provides more robust results, especially when works with small fine-tuned datasets. In general, this research work adds to the number of studies in the field of forgery detection by using meta-learning for discriminating unseen forgery methods.

3. MATERIALS AND METHODS

In this section, we discussed the proposed methodology of the system. The system includes data collection, data prepossessing, model building, performance analysis, ensembles models and meta learning. The architecture of the system is shown in figure 1.

A. Data Collection

The deepfake faces dataset is obtained from kaggle [13]. The deepfake faces dataset images depict real and fake images for human face. Approximately 95,634 RGB images comprise the dataset, which is classified into 2 distinct classifications representing real or fake class. Offline augmentation was used to recreate the original dataset to generate the dataset.The number of images and their trining, testing distribution is shown in the table I.

TABLE I. Training and Testing Data Distribution

	Training	Testing
Real	63,472 images	15,868 images
Fake	13,035 images	3,259 images

B. Data Preprocessing

1) Augmentation

We apply augmentation to remove class imbalance. There is a class imbalance between this classes. As a result, the classification result will be more biased towards the majority classes. By using image augmentation, we increase the number of samples in the real faces class.To augment image data, techniques such as 30 percent outward scaling, 45-degree clockwise rotation, shifting with width shift = 0.1 and height shift = 0.1, and horizontal flipping have been employed. When there is outward scaling, the new image is cropped to the same size as the original. Before augmentation, the real faces have only 16293 images. We increases the number of images in real class and make it 75210 images that is shown in table II.

2) Segmentation

The segmentation of the image is performed to extract that area where all the keypoints are present. Unsegmented data - If data is not segmented and still contains too many

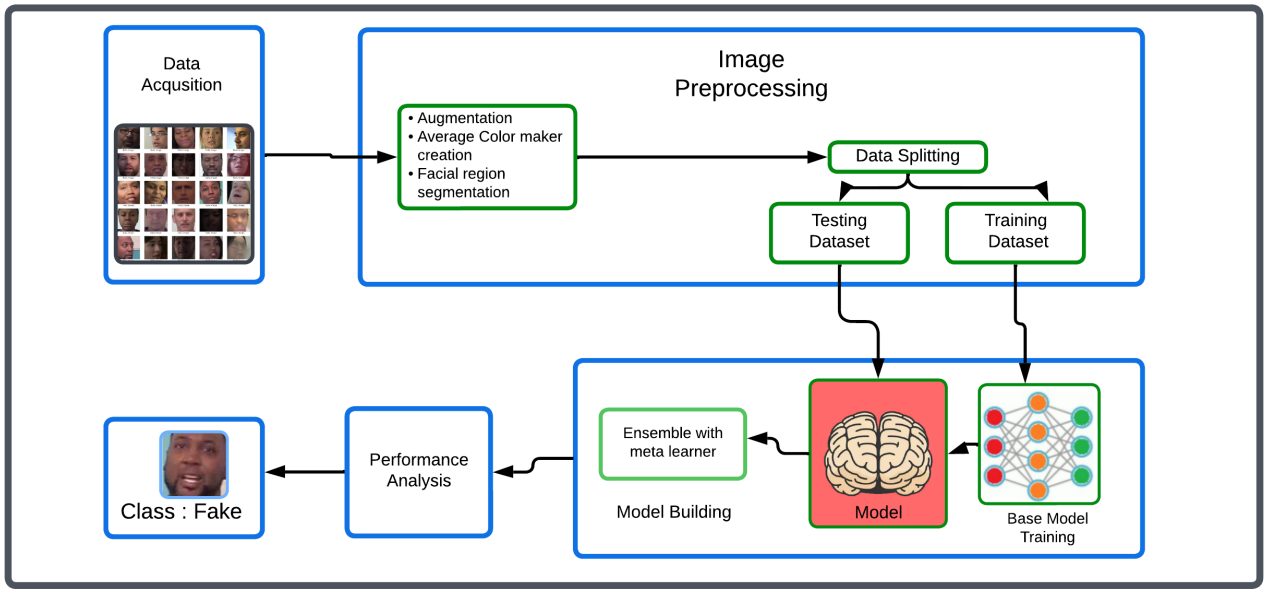


Figure 1. Workflow of the proposed model

TABLE II. Comparison of Before and After Augmentation

Classes	Before Augmentation	After Augmentation
Real	16293	75210
Fake	79341	79341

background objects contamination would occur during feature extraction that could reduce accuracy. The segmentation process of the system includes three steps; background removal, average color markers creation and key region segmentation.

3) Background removal

In this step, we remove any background rather than the key part so that it doesn't create any unwanted results in the training phases. Before performing detection the frame was cleaned using color-based segmentation to remove background or any non-key area. The proposed method helps to separate the key point related regions from the background for further analysis and measurements. A common technique to achieve this is with thresholding in a relevant color space (e.g. the Hue-Saturation-Value HSV color space). Within this space, the hue component effectively captures color information, rendering it especially adept at discerning the key regions.

Let I represent the input image, converted to HSV format. The segmentation process entails defining a range of values for the hue channel, H_{min} to H_{max}, that corresponds to the key color spectrum. Mathematically, this can be expressed as:

$$mask = \begin{cases} 1 & \text{if } H_{\text{min}} \leq H(I(x,y)) \leq H_{\text{max}} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $H(I(x,y))$ represents the hue value at pixel (x,y) in the image I. The resulting binary mask, *mask*, identifies pixels within the specified hue range, effectively segmenting the key regions. Subsequently, the inverse of the mask is applied to obtain the background. This process is mathematically represented as:

$$\text{background} = I \odot (\neg mask) \quad (2)$$

where ¬mask denotes the bitwise negation of the mask and $\odot$ represents the bitwise AND operation. Finally, the resulting image with the key background removed, denoted as result, is obtained by applying the mask to the original image:

$$\text{result} = I \odot \text{mask} \quad (3)$$

This forms the basis of the color-based segmentation technique, aiding in extracting features from the images more accurately. Equations (1), (2) , and (3) describe the key steps in the color-based segmentation process. Figure 2 shows the process of background removal: (a) shows the image before background removal, and (b) displays the result after the background has been removed.

4) Average color marker creation

The average color marker, derived from combining I_a and I_b channels, simplifies color representation and aids in efficient segmentation and feature extraction tasks in image analysis. Here, the `roipoly` function in MATLAB

Figure 2. (a) Before background removal (b) After background removal

is used to delineate distinct regions within each image in our dataset: a real region (R1), a fake region (R2), and a background region (R3). For each region, a binary mask is generated, denoted as M_i, where i represents the specific region (1 for R1, 2 for R2, and 3 for R3). This binary mask visually indicates pixel membership within the chosen region. The binary mask $M_i(x, y)$ is defined as in the following equation 4.

$$M_i(x,y) = \begin{cases} 1 & \text{if } I_{\text{RGB}}(x,y) \in R_i \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The binary mask M_i comprises pixels with values of either 1 or 0, indicating membership or non-membership to the respective region. Pixels within the mask are allocated a value of 1 (white) if they belong to the selected region and 0 (black) if they do not. Consequently, three binary masks are generated from a single image I_{RGB} for its three sample regions.

Next, the RGB image is converted into 3 color channels: I_L, I_a, and I_b, each with dimensions 256×256. Here, I_L represents the grayscale or intensity component, I_a represents the intensity of the red-green axis, and I_b represents the intensity of the blue-yellow axis. The I_a and I_b together make the average color marker. After that, bitwise logical AND operations are performed on I_a and I_b after applying the mask M_i. Two column vectors, $V_{ai}(x, y)$ and $V_{bi}(x, y)$, are generated as follows:

$$V_{ai}(x,y) = I_a(x,y) \quad \text{if } M_i(x,y) \neq 0 \tag{5}$$

$$V_{bi}(x,y) = I_b(x,y) \quad \text{if } M_i(x,y) \neq 0 \tag{6}$$

If an element (x, y) has a non-zero binary mask, then the pixel value of this element of I_a will be inserted into V_{ai}. V_{bi} will be created in the same way. The number of non-zero elements in M_i equals the length of V_{ai} and V_{bi}. If V_{ai} and V_{bi} have a size of P, their average value is:

$$f(ai) = \sum_{j=1}^{P} V_{ai}(X_j, 1) \tag{7}$$

$$f(bi) = \sum_{j=1}^{P} V_{bi}(X_j, 1) \tag{8}$$

In $a * b$ color space, we thus obtain the color marker of $[f(ai), f(bi)]$ for region R_i. Three color markers can be obtained from a single image since we have three sample regions, R1, R2, and R3. These color markers are real, fake, and background color markers. We repeat above steps for all images in our sub-dataset. The following formulas are used to determine the average color marker:

Average real color marker; $[f(a1), f(b1)]$:

$$= \frac{1}{600} \sum_{j=1}^{600} [\varepsilon a1, \varepsilon b1]_j \tag{9}$$

Average fake color marker $[f(a2), f(b2)]$:

$$= \frac{1}{600} \sum_{j=1}^{600} [\varepsilon a2, \varepsilon b2]_j \tag{10}$$

Average background color marker; $[f(a3), f(b3)]$:

$$= \frac{1}{600} \sum_{j=1}^{600} [\varepsilon a3, \varepsilon b3]_j \tag{11}$$

After combining all of the color markers into a matrix p we get:

$$p = \begin{bmatrix} f(a1) & f(b1) \\ f(a2) & f(b2) \\ f(a3) & f(b3) \end{bmatrix} \tag{12}$$

C. Face Region Segmentation

Face region segmentation is used to precisely identify and outline the facial areas within images. This process consists of three steps: Color Space Conversion, Distance Calculation, and Pixel Classification.

In color space conversion, we convert the RGB image I_{RGB} to the Lab color space I_{Lab}. After that, we split I_{Lab} into its three components: $I(L)$, $I(a)$, and $I(b)$, where L represents luminance, a represents the green-red axis, and b represents the blue-yellow axis.

Next, we calculate three distances D_{facial}, $D_{\text{non-facial}}$, and $D_{\text{background}}$ based on $I(a)$ and $I(b)$ compared to the facial color marker p:

$$D_{\text{facial}} = \sqrt{\{I_a(x,y) - p(1,1)\}^2 + \{I_b(x,y) - p(1,2)\}^2} \tag{13}$$

$$D_{\text{non-facial}} = \sqrt{\{I_a(x,y) - p(2,1)\}^2 + \{I_b(x,y) - p(2,2)\}^2} \tag{14}$$

$$D_{\text{background}} = \sqrt{\{I_a(x,y) - p(3,1)\}^2 + \{I_b(x,y) - p(3,2)\}^2} \tag{15}$$

After calculating these distances, each pixel (x, y) is classified based on the minimum distance:

$$\text{Region}(x,y) = \begin{cases} \text{Facial}, & \text{if } \min(D_{\text{facial}}, D_{\text{non-facial}}, D_{\text{background}}) \\ & \quad = D_{\text{facial}}, \\ \text{Non-Facial}, & \text{if } \min(D_{\text{facial}}, D_{\text{non-facial}}, D_{\text{background}}) \\ & \quad = D_{\text{non-facial}}, \\ \text{Background}, & \text{if } \min(D_{\text{facial}}, D_{\text{non-facial}}, D_{\text{background}}) \\ & \quad = D_{\text{background}}. \end{cases} \tag{16}$$

D. Applied Deep learning Models

We have implemented various deep learning models such as EfficientNetB4, MobileNetV2, ResNet50, NASNetMobile and DenseNet121. Each of these models are separately trained on a training dataset with SGD optimizer. The method not only employs learning rate scheduling and early stopping techniques to mitigate the risk of overfitting and achieve a global minimum, but also uses the validation set independently to assess the performance of individual models and understand their contribution to the ensemble. The following subsections give a brief overview of the implemented deep learning models. Also, table III provides the comparative analysis of different deep leanring models interms of architecture, layers, trainable parameters, input size, etc.

1) DenseNet121

In 2017, Huang et al. introduced DenseNet121, a deep convolutional neural network (CNN) with 121 layers. It contains a fully connected layer where every layer is connected with one another in a feed-forward fashion. By building this way, we encourage features to be reused and the gradients experience less vanishing gradient. With the leading results of top 1 accuracy 74.85% and top 5 accuracy 92.86% on ImageNet dataset, DenseNet121 model has presented its excellent ability of solving image classification problems. Due to how fast and effective it was, people have been using this for many different types of transfer learning[14].

2) MobileNetV2

In 2018 Sandler et al. introduced MobileNetV2, a compact CNN architecture to be run on mobile and embedded devices. It uses lightweight modules to balance those factors, which stand for both accuracy and efficiency, i.e. inverted residual block and depthwise separable convolutions. MobileNetV2 is efficient at around 71.8% top-1 and 90.2% top-5 accuracy. Standard usage is rooted in providing considerable efficiency with low computation costs[15].

3) ResNet50

ResNet50, a 50-layer variant of the ResNet (Residual Networks) architecture family presented by Microsoft in 2015, adopts shortcuts to get tremendous depth in the network. ResNet50 improves the gradient flow by letting information to flow directly, i.e learn an identity function through skip connections which paves way to training very deep models. The model is showing to give one of the best

results in many different kinds of computer vision such as image classificaiton, feature Extraction, etc[16].

4) NASNetMobile

Nasnetmobile is developed by Google in 2018 with neural architecture search (NAS) techniques, and is designed to work well for mobile devices with no/lower compute. It uses a Stacked Normal and Reduction Cells for high image classification accuracy while keeping computationally-efficient. NASNetMobile is a pioneering step in automating the discovery of directly-usable, specialized computational structures for targeted constraints and tasks in neural network architecture as used by mobile high performance applications[17].

5) EfficientNetB4

The EfficientNetB4 is presented by Tan and Le in 2019, scales up the network dimensions (depth, width, resolution) asymmetrically to provide more performance gain with fewer number of parameters. It uses compound scaling techniques to provide the best trade-off between model performance and computational resources. EfficientNetB4 has shown excellent performance on many image classification benchmarks, This makes it a strong candidate for transfer learning and general purpose image recognition tasks[18].

Table III shows the comparison of different CNN architectures in terms of layers, Trainable parameters, input size, and output size.

E. Proposed ensemble training approach

Figure 3 shows the workflow of the proposed ensemble architecture. The approach begins with training EfficientNetB4, MobileNetV2, ResNet50, NASNetMobile, and DenseNet121 separately on the training dataset. Each model is optimized using SGD with specific hyperparameters tailored to maximize performance. We then compared the performance of feature extraction by each model. During this evaluation, the ensemble learning weight of each model was calculated, giving higher weights to models that demonstrated superior feature extraction capabilities. We calculated these weights by evaluating each model's ability to extract proper features, using the vector distribution of the training set. Additionally, we calculated feature vectors for each model and assessed how well they clustered around category centroids, as this is indicative of the effectiveness of the features in categorizing images. Given the softmax outputs of each model, for a new image, we use both stacking and blending to improve prediction.

In Stacking, we trained meta-classifiers, such as RandomForest, to learn how to combine the predictions of the individual models. The concatenation of the predictions (meta-features) from EfficientNetB4, MobileNetV2, ResNet50, DenseNet121, and NASNetMobile models were used as input for this meta-classifier. On the other hand, in blending, we combined the softmax outputs from all models by taking their direct average to make a final prediction. During the ensemble learning phase, blending was

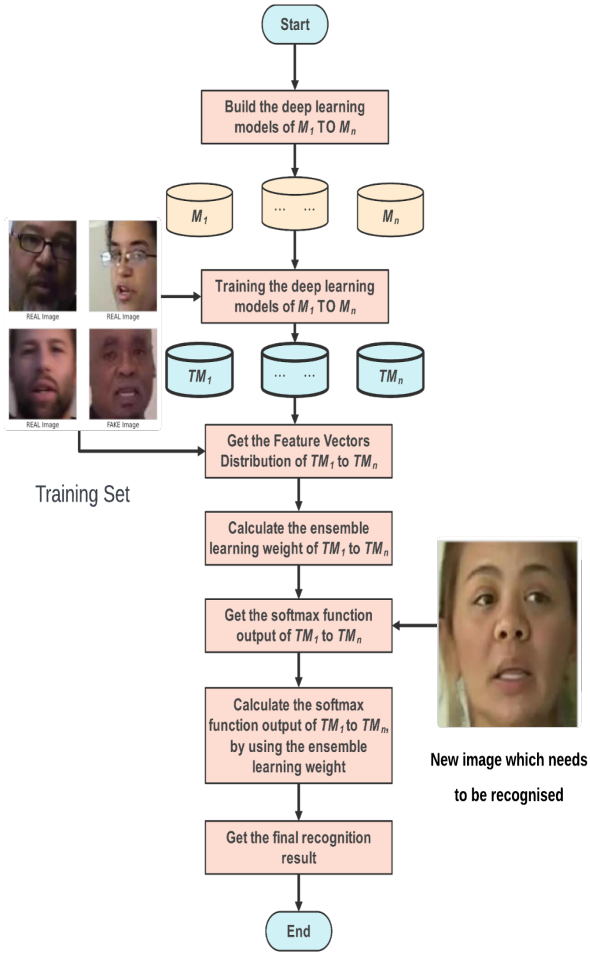


Figure 3. Flowchart of Proposed Ensemble architecture

evaluated using performance metrics such as precision and recall. Weights of zero were assigned to lower-performing models in the blending process, allowing high-performing sub-models to contribute more significantly to the final prediction. Figure 4 and 5 illustrate the proposed ensemble architecture using stacking meta classifier and proposed ensemble architecture using blending meta classifier to enhance overall model performance respectively. Fine-tuning of the ensemble model may be performed based on its performance on the validation set to optimize hyperparameters or adjust the ensemble strategy. Finally, the performance of the ultimate ensemble model is evaluated using the independent testing dataset, assessing critical metrics such as accuracy, precision, and recall. Ensembling multiple deep learning models can often lead to improved performance compared to using individual models alone.

TABLE III. Comparison of CNN Architectures

Model	Year	Architecture	Layers	Trainable Params	Input Size	Output Size
EfficientNetB4	2019	Compound Scaling, Efficient	389	19.0M	380x380x3	1000
MobileNetV2	2018	Depthwise Separable Conv	53	3.5M	224x224x3	1000
ResNet50	2015	Residual Connections	50	25.6M	224x224x3	1000
NASNetMobile	2017	Neural Architecture Search	53	5.3M	224x224x3	1000
DenseNet121	2017	Dense Connectivity Pattern	121	6.9M	224x224x3	1000

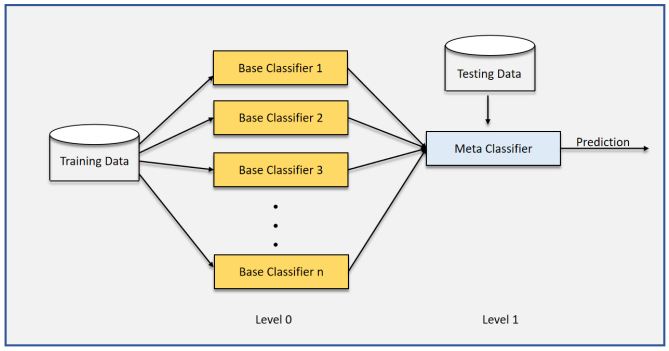


Figure 4. Stacking ensemble architecture

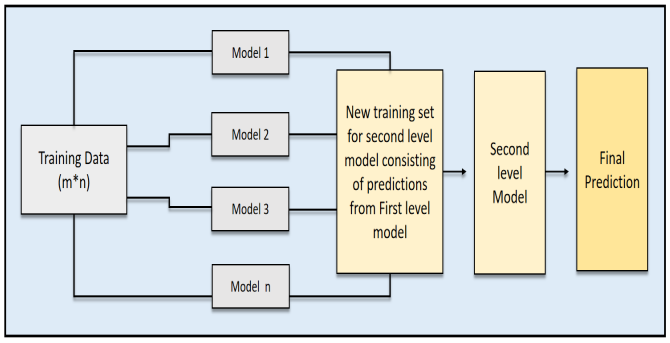


Figure 5. Blending ensemble architecture

4. Results and Discussion

A. Simulation Set and Environment

To conduct this research, we used Intel Core i7 processor and 8 GB of RAM. The codebase was run using Jupyter Notebook. We employed a Gigabyte GTX 1050 Ti OC 4GB DDR5 graphics card as the graphical unit. At the beginning, we downloaded the "deepfake faces" dataset from Kaggle [13]. The dataset consists of two classes. After performing the necessary preprocessing, the five base models—EfficientNetB4, MobileNetV2, ResNet50, NASNetMobile, and DenseNet121—were implemented. Next, stacking and blending were used to ensemble these models. Here, the mean was calculated based on the median predicted probabilities of these two models. The Adam optimizer was used to compile the model, adjusting the learning rate during training. Training was done for 10 epochs at a time in batches using the training data generator. The model iteratively adjusted the weights to minimize cross-entropy loss and increase accuracy over all batches in the training data.

B. Confusion Matrix of the proposed ensemble model

There are some specific performance metrics that are used to assess various deep learning based machine learning techniques. A confusion matrix evaluates the performance of a classification model by detailing the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). TP and TN represent correct predictions, while FP and FN show where the model has made errors. These metrics help in understanding the model's accuracy and the types of errors it tends to make, such as mistaking negatives for positives or vice versa. Figures 8(a) to 8(e) present the confusion matrices for EfficientNet, MobileNet, ResNet, NasNet, and DenseNet, providing a comparative view of their performance. This helps in identifying the strengths and weaknesses of each model, guiding improvements in classification accuracy and error reduction.

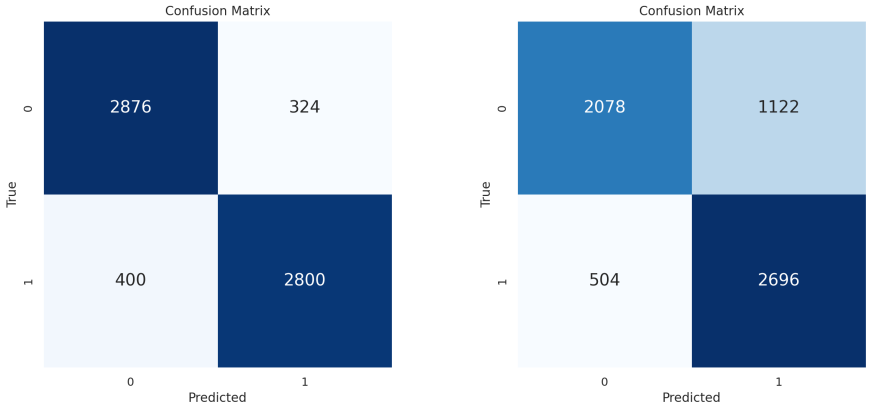
- **True Positive (TP):** Instances that were correctly predicted as positive. The model predicted a positive class, which was also positive.
- **True Negative (TN):** Instances that were correctly predicted as negative. The model predicted the negative class, and the actual class was also negative.
- **False Positive (FP):** Instances that were incorrectly predicted as positive. The model predicted the positive class, but the actual class was negative (a Type I error).
- **False Negative (FN):** Instances that were incorrectly predicted as negative. The model predicted the negative class, but the actual class was positive (a Type II error).

Figure 6(a), 6(a), 6(b), 6(c),6(d) and 6(e) display the confusion matrix of EfficientNet, MobileNet, ResNet, NasNet, and DenseNet serially.

C. Performance Analysis

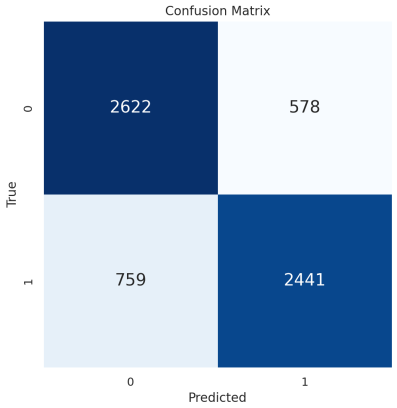
The following parameters are used to evaluate the performance of the base models and proposed model.

- **Accuracy:** Accuracy is the ratio of correct predictions across the whole dataset. It is calculated as the ratio

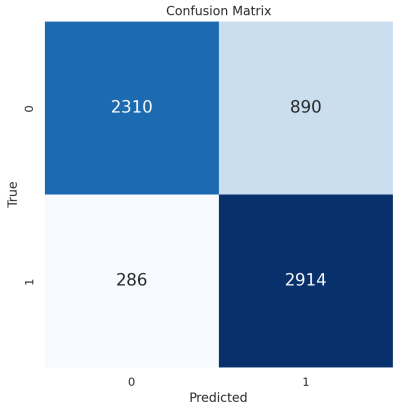


(a) EfficientNetV4

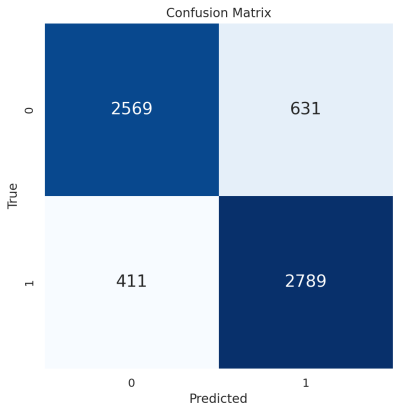
(b) MobileNetV2



(c) ResNet50



(d) NasNetMobile



(e) DenseNet121

Figure 6. Confusion matrices for various neural network models (a) EfficientNet (b) MobileNetV2 (c) ResNet (d) NasNet, and (e) DenseNet.

TABLE IV. Classification performance for real and fake images.

Model	Accuracy		Precision		Recall		F1 Score	
	Real	Fake	Real	Fake	Real	Fake	Real	Fake
EfficientNet	0.89	0.89	0.88	0.88	0.90	0.90	0.89	0.89
MobileNetV2	0.75	0.75	0.71	0.80	0.84	0.65	0.77	0.72
ResNet50	0.79	0.79	0.81	0.78	0.76	0.82	0.79	0.80
NasNetMobile	0.82	0.82	0.77	0.89	0.91	0.72	0.83	0.80
DenseNet101	0.84	0.84	0.82	0.86	0.87	0.80	0.84	0.83
Staking (Proposed Method)	0.87	0.87	0.86	0.86	0.87	0.87	0.87	0.87
Blending (Proposed Method)	0.924	0.924	0.91	0.91	0.91	0.91	0.92	0.92

of true positive and false positive with respect to the whole dataset.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{17}$$

- **Precision:** Precision is calculated as the ratio of true positive predictions and the all postove predictions across the dataset.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{18}$$

- **Recall:** Recall is calculated as the proportions of true positive predictions among the whole actual positive instances.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{19}$$

- **F1 Score:** F1 balances the precision and recall. It is the harmonic average of precision and recall.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{20}$$

Table IV displays a comparative analysis of the performance of each individual model with that of the proposed ensemble model in both stacking and blending cases in terms of four evaluation metrics.

From table IV, the proposed blending model achieves the highest accuracy for both "Real" and "Fake" data (92.4% for both cases). EfficientNet also performs relatively well with an accuracy of 89% for both "Real" and "Fake" data. The accuracy of other models Stacking, DenseNet-101, NasNetMobile, ResNet-50 and MobileNetV2 are 87%, 84%, 82%, 79%, 75% respectively. On the other hand, precision measures the proportion of true positive predictions out of all positive predictions. High precision ensures that when the model predicts a class, it is highly likely to be correct, minimizing false positives. From the table, we can see that the Blending model has the highest precision (91%) for both cases. EfficientNet comes next with a precision of 88%, followed by Staking with 86%. MobileNetV2 has the lowest precision at 71%. The Blending model also leads in recall with a score of 91% for both real and fake images, closely followed by EfficientNet, DenseNet101, and stacking with a value of 90%, 87% and 87% respectively. In terms of F1 Score, the Blending model tops the list with 92%, indicating a balanced performance in precision and recall, while the EfficientNet and Staking model trail behind with scores of 89% and 87%, respectively. In summary, the proposed blending model outperforms the other models in all metrics, making it ideal for the classification task.

D. Specificity and Sensitivity Analysis

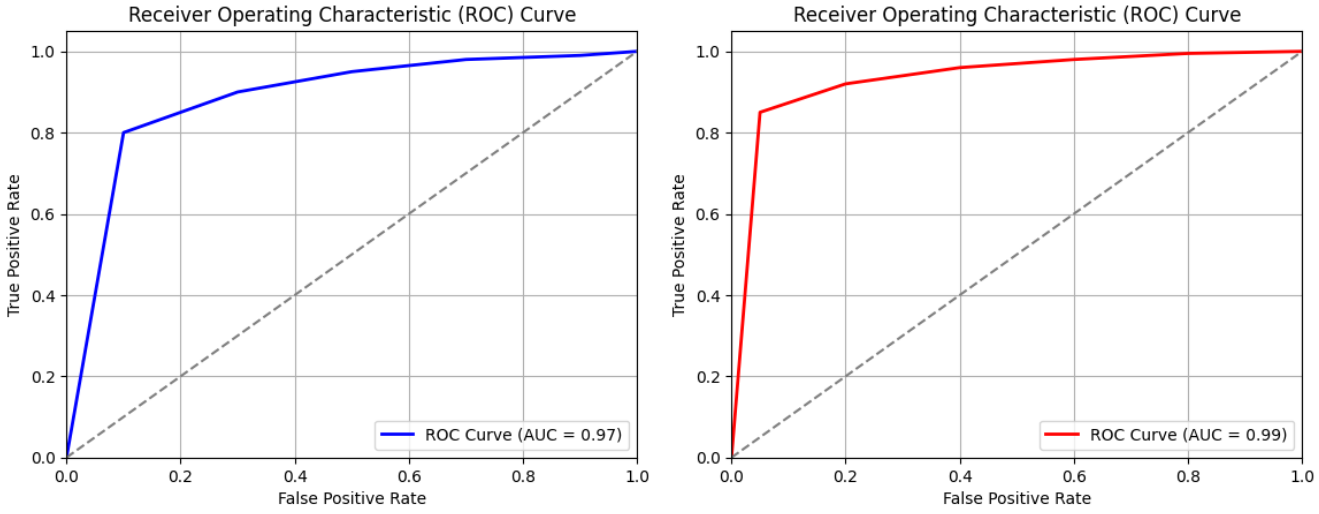
We have generated receiver operating characteristic (ROC) curves for each class using the one-vs-rest method to assess the trade-off between sensitivity and specificity of our proposed blending method. Figure 7 displays the ROC curves for real and fake images of the proposed blending method. As of figure 7, the ROC curves are extremely close to the upper left corner for each class, indicating that the model is performing exceptionally well. For real and fake images , the Area Under the Curve (AUC) values are 0.97, 0.990, and 0.99, respectively. These high AUC values demonstrate the suggested model's strong discriminating power. There is a greater than 99% chance that our model will select a random positive sample over a random negative sample.

E. Comparison with Existing Approaches

Table V compares the proposed model against other existing available state-of-the-art techniques for deepfake detection in order to assess its applicability. The metrics used for comparison are Accuracy, Precision and Recall. From the table, it is clear that the proposed system has an accuracy of 92.4%, which is better than others, including those by Shraddha Suratkar[19] and Anuj Badale[20].

5. Conclusions

This research demonstrates significant advancements in deepfake detection through the use of deep learning and ensemble learning techniques. By training and testing models such as EfficientNetv4, MobileNetV2, ResNet50, NasNetMobile, and DenseNet, we evaluated their performance and achieved acceptable detection accuracies. The implementation of meta-learning methods, particularly blending with a random forest classifier, resulted in superior detection performance. Our proposed hybrid model effectively identifies fake faces in real time, significantly enhancing security and reducing the need for time-consuming and error-prone manual analysis. This innovation boosts trust and security in critical sectors such as banking and finance, preventing fraud and maintaining privacy. Additionally, these advanced



(a) (b)

Figure 7. ROC curve for each class using the one-vs-rest method. (a) Real Image; (b) Fake Image

TABLE V. Summary of Deep Fake Detection Methods

Author	Year	Model/Method	Accuracy	Precision	Recall
Anuj Badale[20]	2021	Adam optimizer(NN)	91%	-	-
Rimsha Rafique[21]	2023	ResNet	89.5%	89.5%	89.5%
Shraddha Suratkar[19]	2022	RNN	92%	90%	
Xinyi Ding[22]	2020	Transfer learning	90%	-	-
Proposed	-	Meta learner	92.4%	91%	91%

detection systems alleviate the burden on human analysts and reduce the costs associated with security incidents. It enables new applications in biometrics and digital forensics, providing robust security solutions across various platforms.

References

[1] P. Yu, Z. Xia, J. Fei, and Y. Lu, "A survey on deepfake video detection," *Iet Biometrics*, vol. 10, no. 6, pp. 607–624, 2021.

[2] Y. Zhu, M. Wang, X. Yin, J. Zhang, E. Meijering, and J. Hu, "Deep learning in diverse intelligent sensor based systems," *Sensors*, vol. 23, no. 1, p. 62, 2022.

[3] J. T. Hancock and J. N. Bailenson, "The social impact of deepfakes," pp. 149–152, 2021.

[4] J. C. Neves, R. Tolosana, R. Vera-Rodriguez, V. Lopes, and H. Proença, "Real or fake? spoofing state-of-the-art face synthesis detection systems," *arXiv preprint arXiv:1911.05351*, vol. 2, p. 6, 2019.

[5] S. Suganthi, M. U. A. Ayoobkhan, N. Bacanin, K. Venkatachalam, H. Štěpán, T. Pavel *et al.*, "Deep learning model for deep fake face recognition and detection," *PeerJ Computer Science*, vol. 8, p. e881, 2022.

[6] Y. Wang, V. Zarghami, and S. Cui, "Fake face detection using local binary pattern and ensemble modeling," in *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2021, pp. 3917–3921.

[7] J.-Y. Baek, Y.-S. Yoo, and S.-H. Bae, "Generative adversarial ensemble learning for face forensics," *Ieee Access*, vol. 8, pp. 45 421–45 431, 2020.

[8] G. Ajomale, "Face spoof detection using ensemble classifier," Ph.D. dissertation, Dublin, National College of Ireland, 2021.

[9] P. Sharma, M. Kumar, and H. K. Sharma, "Gan-cnn ensemble: A robust deepfake detection model of social media images using minimized catastrophic forgetting and generative replay technique," *Procedia Computer Science*, vol. 235, pp. 948–960, 2024.

[10] M. S. Rana and A. H. Sung, "Deepfakestack: A deep ensemble-based learning technique for deepfake detection," in *2020 7th IEEE international conference on cyber security and cloud computing (CSCloud)/2020 6th IEEE international conference on edge computing and scalable cloud (EdgeCom)*. IEEE, 2020, pp. 70–75.

[11] H. Mansourifar and W. Shi, "One-shot gan generated fake face detection," *arXiv preprint arXiv:2003.12244*, 2020.

[12] Y.-K. Lin and T.-Y. Yen, "A meta-learning approach for few-shot

face forgery segmentation and classification," *Sensors*, vol. 23, no. 7, p. 3647, 2023.

[13] "deepfake_faces–kaggle.com," https://www.kaggle.com/datasets/dagnelies/deepfake-faces/data, [Accessed 26-06-2024].

[14] A. Alzahrani, "Digital image forensics: An improved densenet architecture for forged image detection," *Engineering, Technology & Applied Science Research*, vol. 14, no. 2, pp. 13 671–13 680, 2024.

[15] R. Chithra, A. Teijas, G. Thangavel, and R. Vasantheeswaran, "Deep learning-based facial deepfake detection using mobilenetv2 and vgg16," in *International Conference on Smart Data Intelligence*. Springer, 2024, pp. 155–167.

[16] Z. N. Ashani, I. S. C. Ilias, K. Y. Ng, M. R. K. Ariffin, A. D. Jarno, and N. Z. Zamri, "Comparative analysis of deepfake image detection method using vgg16, vgg19 and resnet50," *Journal of Advanced Research in Applied Sciences and Engineering Technology*, vol. 47, no. 1, pp. 16–28, 2024.

[17] D. R. Arulanandar, V. Padmanabhan, P. Nammalwar, and S. P. Govindan, "Static signature verification using nasnet deep learning architectures," in *AIP Conference Proceedings*, vol. 2829, no. 1. AIP Publishing, 2023.

[18] A. Zhalgasbayev, T. Aiteni, and N. Khaimuldin, "Using the cnn architecture based on the efficientnetb4 model to efficiently detect deepfake images," in *2024 IEEE AITU: Digital Generation*. IEEE, 2024, pp. 14–19.

[19] S. Suratkar and F. Kazi, "Deep fake video detection using transfer learning approach," *Arabian Journal for Science and Engineering*, vol. 48, no. 8, pp. 9727–9737, 2023.

[20] A. Badale, L. Castelino, C. Darekar, and J. Gomes, "Deep fake detection using neural networks," in *15th IEEE international conference on advanced video and signal based surveillance (AVSS)*, 2018.

[21] R. Rafique, R. Gantassi, R. Amin, J. Frnda, A. Mustapha, and A. H. Alshehri, "Deep fake detection and classification using error-level analysis and deep learning," *Scientific Reports*, vol. 13, no. 1, p. 7422, 2023.

[22] X. Ding, Z. Raziei, E. C. Larson, E. V. Olinick, P. Krueger, and M. Hahsler, "Swapped face detection using deep learning and subjective assessment," *EURASIP Journal on Information Security*, vol. 2020, pp. 1–12, 2020.