Received: 28 March 2024 | Revised: 10 May 2024 | Accepted: 28 May 2024 | Published online: 7 June 2024

RESEARCH ARTICLE

Journal of Computational and Cognitive Engineering
2025, Vol. 4(4) 561–569
DOI: 10.47852/bonviewJCCE42022955

BON VIEW PUBLISHING

Coot Bird Optimization-Based ESkip-ResNet Classification for Deepfake Detection

V. Gokula Krishnan[1,*], R. Vadivel[2], K. Sankar[3], K. Sathyamoorthy[4] and B. Prathusha Laxmi[5]

[1]*Department of Computer Science and Engineering, Saveetha Institute of Medical and Technical Sciences, India*

[2]*Department of Computer Science and Engineering, B.N.M. Institute of Technology, India*

[3]*Department of Computer Science and Engineering, CVR College of Engineering, India*

[4]*Department of Computer Science and Engineering, Panimalar Engineering College, India*

[5]*Department of Artificial Intelligence and Data Science, R.M.K. College of Engineering and Technology, India*

Abstract: With increased digitization comes an increase in the speed at which threats to the data are emerging. Although it can be challenging to identify, fake image creation doesn't require any particular memory, computational equipment, or hardware. Consequently, this study uses deep learning to achieve accurate detection. In order to improve detection performance, the study strengthened the line separating the background from the object. It also used the adaptive 2D Wiener filter for preprocessing in order to attenuate noise that was unintentionally reinforced throughout the process of improving the image. This essay suggests an Efficient Skip Connections-based Residual Network (ESkip-ResNet) by utilizing skip connections with the Residual Network (ResNet). The ESkip-ResNet architecture also has a number of stages and progressively more leftover blocks to enhance the classification process. ESkip-ResNet uses the remaining blocks of identity mapping through skip connections in the ResNet architecture. Additionally, ESkip-ResNet has effective techniques for downsampling and stable batch normalization layers, which both improve its stable and dependable performance. The Coot Bird Optimization (CBO) method is used to fine-tune the hyper-parameters of the proposed classifier. The suggested model, ESkip-ResNet, was proposed to be more sensible and to offer better performance. The ESkip-ResNet architecture also has a number of stages and progressively more leftover blocks to enhance the classification process. The proposed model achieved 98.9% and 98.8% accuracy and precision, respectively. Comprehensive test results demonstrate that CBO-based ESkip-ResNet outperforms other approaches in fake detection. The proposed research also took into account every kind of facial alteration, improving the model's robustness, lightweight nature, and generalizability. It was able to identify every type of facial alteration found in images taken from the Deepfake Detection Challenge dataset.

Keywords: Deepfake Detection Challenge, deep learning, image enhancement, Residual Network, Coot Bird Optimization

1. Introduction

Deepfake detection has become an important area of research and technological advancement [1]. The objective is to create and put into use reliable algorithms and systems that can tell the difference between real faces and ones that were created artificially [2]. Combating various deceptive practices, like deepfake videos and manipulated images, is crucial because they can be used for malicious purposes like identity theft, disseminating false information, and eroding confidence in visual media [3].

In the fields of artificial intelligence (AI) and computer vision, scientists and researchers are actively investigating novel methods for identifying phony faces [4]. These approaches frequently make use of deep neural networks, facial recognition software, and sophisticated image analysis techniques. The creation of trustworthy fake face detection tools is essential for preserving the integrity of visual content as well as protecting people and organizations from the possible risks associated with manipulating and misleading images [5, 6].

By allowing machines to learn and make decisions in ways that are inspired by the human brain, one powerful branch of AI has revolutionized a number of industries [7]. These deep neural networks are capable of autonomous data classification, pattern recognition, and prediction. Their numerous hidden layers set them apart [8]. Numerous industries, including health care, finance, autonomous vehicles, and recommendation systems, are using deep learning (DL). The capabilities of machines to understand and process complex data are continuously being pushed by these applications [9].

***Corresponding author:** V. Gokula Krishnan, Department of Computer Science and Engineering, Saveetha Institute of Medical and Technical Sciences, India. Email: gokulakrishnan.sse@saveetha.com

DL has revolutionized the fake face detection field by providing an accurate and useful way to distinguish between real and manipulated facial imagery [10]. DL models have become indispensable tools in the ongoing battle against altered visual content, particularly in the area of facial recognition [11]. These models employ complex neural networks and sophisticated algorithms. The ability of these models to distinguish between facial representations created or modified through various manipulation techniques and those derived from real-world data stems from their exceptional skill at recognizing intricate patterns, subtleties, and characteristics in face data [12]. Through the analysis of large datasets and a multitude of facial variations, DL algorithms have refined their capacity to identify minute cues, disparities, and anomalies suggestive of artificial or manipulated facial images [13].

1.1. Motivation

In the current digital environment, data is becoming more and more vulnerable, which emphasizes the urgent need for robust defenses against emerging threats. Data protection is becoming increasingly challenging due to the concerning increase in the production of false images. This study, however, does more than just list the issues; it provides an innovative solution based on cutting-edge technology. This work presents a novel DL approach called Coot Bird Optimization (CBO)-based ESkip-ResNet. This architectural wonder combines state-of-the-art techniques like CBO for hyper-parameter tuning to improve its accuracy (ACC) and dependability while fortifying the lines between reality and manipulation. This model stands out due to its all-encompassing approach; not only does it perform remarkably well in performance metrics such as ACC and sensitivity, but it also takes into account the variety of facial modifications, ensuring adaptability and low-weight functionality. The remarkable outcomes of 99.6%, 99.2%, and 99.4% confirm its unparalleled efficacy in identifying fraudulent content. It paves the way for a safer and more secure digital future by handling face modification comprehensively and enhancing resilience. Its success is a triumph for both detection technology and data integrity preservation.

1.2. Main contributions

This work's primary contributions are:

1) Introduction of DL for fake image detection: DL techniques are used to correctly detect bogus photos.
2) Improvement in image preprocessing: Utilizing the adaptive 2D Wiener (A2D-Wiener) to minimize noise while improving images.
3) Architecture enhancement – ESkip-ResNet: Utilizing skip connections and multiple stages while utilizing ResNet, ESkip-ResNet is being developed.
4) Identity mapping integration: Improved performance of the model and identity mapping are achieved through the addition of skip connections.
5) Improvements in robustness: Include effective downsampling techniques and stabilizing batch normalization (BN) layers for increased model robustness.
6) CBO hyper-parameter tuning: The model's performance will be enhanced by modifying the hyper-parameters of the suggested classifier using CBO.
7) Performance results: Excellent performance in fake detection is indicated by ACC (99.6%), sensitivity (99.2%), and specificity (99.4%), together with 98.6% *F*1-score values.
8) Entire facial modification coverage: Addressing various face alterations and enhancing the model's robustness, lightweight, and generalizability.
9) Effective use of the DFDC dataset: Pictures from the DFDC dataset are used to show how well the model recognizes different types of facial alterations.

The remaining sections of this essay are organized as follows: Related works are presented in Section 2. Section 3 contains the experimental data as well as the methodological details. Section 4 presents a discussion and analysis of the experimental results. Section 5 finally brings the paper to a close.

2. Related Works

Using 190,335 RGB-resolution deepfake or real photos as a dataset, Iqbal et al. [14] classified deepfake images using data augmentation and transfer learning techniques. Convolutional neural networks (CNNs), Inception V3, VGG16, VGG19, and a transfer learning approach were used in the experiments. Important evaluation metrics were used to determine the effectiveness of the strategy, including the area under the curve-receiver operating characteristics (AUC-ROC) curve score, *F*1-score, recall (RC), ACC, precision (PR), and confusion matrix. The optimized VGG16 algorithm achieved 92.9% ACC, 90.6% PR, 90.2% *F*1-score, and 89.9% AUC-ROC score, outperforming other DL techniques in the differentiation of real and deepfake images.

GLFNet is a global–local facial fusion network that uses both global and local physiological receptive features. It was first used by Xue et al. [15]. GLFNet was divided into two branches: the local region detection branch, which used a residual connection to distinguish between real and fake images, and the global detection branch, which focused on features like the iris and pupils to detect facial fraud. GLFNet enhanced reliability over single-class detection techniques by effectively identifying forged traces using stable DL features with physiological properties by combining physiological features with DL.

A computer vision model based on CNNs was presented by Hamid et al. [16] for the identification of fake images. Six common machine learning (ML) models and six different CNN architectures were compared in order to determine which model was best for additional testing. With an overall ACC of 0.99, their suggested model, which was based on ResNet50 and efficient preprocessing techniques, performed as the best fake image detector.

DL and ML approaches should be used for the automated classification of deepfake images, according to Rafique et al. [17]. They brought attention to the shortcomings of traditional ML systems, which limit their applicability in changing real-world scenarios by making it difficult to extract complex patterns from poorly understood or complex data. Their proposed framework first applied error-level analysis to images and then used CNNs to extract deep features. They used K-Nearest Neighbors and Residual Network to attain the best ACC of 89.5% through hyper-parameter optimization and support vector machines.

Several DL architectures were examined by Coccomini et al. [18] in order to determine how well-suited they were for a broad interpretation of deepfakes. According to their research, CNNs outperformed other models when given datasets with fewer elements and manipulation techniques because of their capacity to identify particular anomalies. On the other hand, when trained on a variety of datasets, the Vision Transformer outperformed other methods in generalization. In cross-dataset scenarios, the Swin Transformer performed well, potentially replacing attention-based

techniques when data was limited. Experiments suggested that attention-based architectures performed better because of their important ability to generalize in real-world scenarios.

A multi-feature channel domain-weighted framework (MCW) for deepfake detection across databases was developed by Guan et al. [19] using meta-learning. By combining RGB and frequency domain data, this framework enhanced the model's feature extraction capabilities and improved its generalization by allocating meta weights to feature map channels. Insufficient data compression resistance and poor detection performance for samples generated by unknown algorithms were addressed by the MCW framework. The MCW framework performed better in cross-algorithm and cross-dataset detection than alternative algorithms in both zero-shot and few-shot scenarios. It showed resilience to compression, robustness in generalization, and improved fine-tuning potential in scenarios with subpar training images and different generation algorithms.

2.1. Research gap

Data threats are emerging at a faster rate due to the current state of digital advancements. The generation of counterfeit images, which lack any particular memory, computing apparatus, or hardware, is very difficult to detect. This study uses DL to achieve accurate detection in order to overcome this difficulty. The research enhances detection performance while reducing accidentally reinforced noise through preprocessing with the A2D-Wiener filter (A2WF). By combining skip connections with the Residual Network (ResNet) architecture, it presents the ESkip-ResNet. The consistent and dependable performance of ESkip-ResNet is enhanced by the integration of stable BN layers and effective downsampling techniques. The sensitivity and effectiveness of the suggested classifier are improved by tuning the hyper-parameters with the usage of the CBO method. In particular, the ESkip-ResNet model outperforms other methods in counterfeit detection with astounding ACC (98.9%) and PR (98.8%). This study supports the model's robustness, adaptability, and lightweight design while also demonstrating its ability to recognize different types of facial alterations found in the DFDC dataset. This offers a promising path toward improving the ACC and legitimacy of counterfeit image identification in dynamic digital environments.

3. Proposed Methodology

The diagram in Figure 1 illustrates the stages involved in putting the suggested technique into practice. This section covers image preprocessing, the classification procedure using ESkip-ResNet, and hyper-parameter tuning with CBO.

3.1. Dataset description

When creating a DL model, the dataset is essential. Results are frequently predicated on the chosen dataset. The DFDC (Deepfake Detection Challenge) dataset was the choice [20]. It is roughly 470 GB in size. This dataset was obtained from the Google-provided Kaggle Competition.

Although the dataset is generalized, some events such as eye blinking, face swapping, lip movement, etc., do not have images available. Consequently, the model is unable to forecast the intended outcome. To enhance the trained model's functionality, we must develop new techniques with more sophisticated parameters to improve the results before running any simulation with this dataset. This dataset was split up into testing, validation, and training sets. Real and fake videos can both be found in this dataset. Videos are transformed into frames, and faces in those frames were chosen to feed data into the model depicted in Figure 2. 68,258 fictitious images and 65,234 real images were used to train the model after breaking down all of the videos into frames and identifying faces in each frame. 5698 phony images and 5876 real images were used for validation. 9542 bogus images and 9785 real images were used for testing.

Table 1 displays the DFDC dataset's statistics, particularly the quantity of faces-containing frames, which were used for the model's training, validation, and testing. The study did not train or test the model using other faceless frames that are extracted from the videos. After eliminating the faceless frames, for model validation, the study found 65,234 real and 68,258 fictitious images; for model training, this work found 5876 real and 5698 fictitious images; and for model testing, the study found 9785 real as well as 9542

Figure 1
Workflow of the proposed model

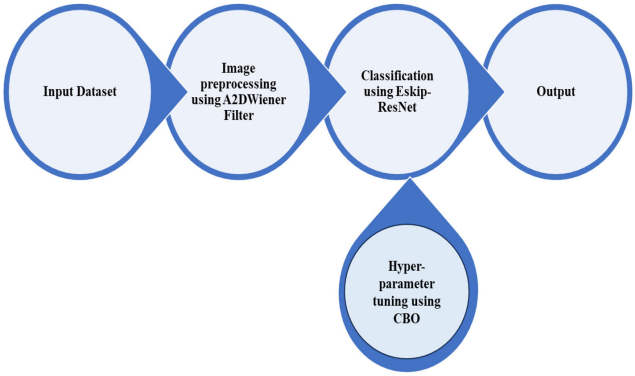

Figure 2
Sample images from the DFDC dataset

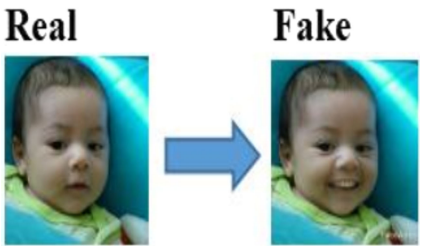

Table 1
Dataset statistics

	Dataset size	DFDC 470 GB
Training	Frames having faces (real)	65,234
	Training Frames having faces (fake)	68,258
Validation	Frames having faces (real)	5876
	Validation Frames having faces (fake)	5698
Testing	Frames having faces (real)	9785
	Testing Frames having faces (fake)	9542

fictitious images. For the model to function to learn as many features as possible, a lot of data are retained for training, and the proposed study also kept some unseen data for evaluating the model's effectiveness.

3.2. Preprocessing

Noise in images taken outside can come from a variety of sources. In particular, photos captured in dimly lit areas are less sharp and more prone to noise. When image enhancement techniques are applied to noisy images, they may inadvertently amplify the noise, which could hinder further data processing, like object detection. An A2WF [21] was used to lessen these noise effects. A2WF adapts to each pixel of the image and is especially good at eliminating additive white Gaussian noise.

$$I_D(x,y) = \mu + \frac{\sigma^2 - v^2}{\sigma^2}(I_E(x,y) - \mu) \tag{1}$$

$$\mu = \frac{1}{PQ}\sum_{x,y\in M} I_E(x,y) \tag{2}$$

$$\sigma^2 = \frac{1}{PQ}\sum_{x,y\in M} I_E(x,y)^2 - \mu^2 \tag{3}$$

Here, I_D is the picture shown in Equation (1) which is got by reducing the picture enhancement's noise; μ and σ^2 are the local variance and mean in relation to the size P; Q in Equation (2) is the pixel in the input picture (x, y) as defined in Equation (3); and v^2 is the variation in the image's noise level. If the noise variance is unknown, then one must use the mean of all local variances. Put differently, there is a significant smoothing of the pixels in the local region if the local variance of the A2WF is large; otherwise, there is very little smoothing. Its selectivity is therefore higher than that of a linear filter, and it operates adaptively to local variance. Consequently, the boundaries and additional high-frequency areas exhibiting rapid alterations are adequately maintained. After preprocessing, the classification should be carried out for model enhancement and ACC.

3.3. Classification using ESkip-ResNet

This paper, which takes inspiration from Kaur et al. [22], suggests a complete ESkip-ResNet. By addressing the problem of vanishing gradients, the ESkip-ResNet (see Algorithm 1) makes training deep neural networks easier by using residual blocks with skip connections. To improve feature learning and capture intricate patterns, five stages make up the structure of ESkip-ResNet, and each one gradually adds more residual blocks. Moreover, BN layers and effective downsampling strategies are incorporated into the ESkip-ResNet architecture to maximize computational efficiency and stabilize and accelerate the process of training.

Algorithm 1: Efficient Skip Connections-based Residual Network (ESkip-ResNet)

Data: Input Image Tensor x
Result: Classification Result
Initialization: Initialize parameters and hyper-parameters;
inputLayer $\leftarrow$ Image Input Layer ($[224 \times 224 \times 3]$);
$F \leftarrow 64$;
layers $\leftarrow$ [inputLayer];
s for $i = 1$ to 4 do
layers $\leftarrow$ StackResidualBlock(layers, F, S);
$F \leftarrow F \times 2$;
$S \leftarrow 1$;
layers $\leftarrow$ [layers, $2D - GaP, F_C(128, d_1), F_C(64, d_2), F_C(3), S_M, C_L$];
return layers;

Function StackResidualBlock (layers, F, S):
conv1 $\leftarrow$ Conv $(3, F,$ Padding, $S)$;
bn 1 $\leftarrow$ BN(conv1);
relu1 $\leftarrow$ ReLU(bn 1);
conv 2 $\leftarrow$ Conv $(3, F,$ Padding)(relu1);
bn 2 $\leftarrow$ BN(conv 2);
skip $\leftarrow x +$ Conv$(3, F,$ Padding $)(BN)$;
relu2 $\leftarrow$ ReLU(skip);
conv 3 $\leftarrow$ Conv $(3, F,$ Padding)(relu2);
bn 3 $\leftarrow$ BN(conv3);
output $\leftarrow x +$ BN;

3.3.1. Convolution layers in a residual block

One of the essential components of the ResNet architecture is the residual block. It is composed of an addition operation (Add), multiple convolutional layers (Conv), BN, and skip connections (Skip). The residual block formula is stated in Equation (4) as follows:

$$\begin{aligned} &\text{function layers} = \text{residualBlock}\ (x, F, S) \\ &\left[\begin{array}{c} \text{Conv}(3, F,\ \text{Padding}, S)(x) \\ \text{BN(Conv)}(x) \\ \text{ReLU}(BN) \\ \text{Conv}(3, F,\ \text{Padding})(\text{ReLU}) \\ \text{Skip} : x + \text{Conv}(3, F,\ \text{Padding})(BN) \\ \text{ReLU(Skip)} \\ \text{Conv}(3, F,\ \text{Padding})(\text{ReLU}) \\ \text{BN(Conv)} \\ \text{Final Output} : x + \text{BN} \end{array}\right] \end{aligned} \tag{4}$$

where x stands for the tensor of input, F indicates how many filters are used in the layers of convolutions, and S symbolizes the process by which these convolutional layers are applied to the input tensor to extract features at various levels of abstraction. Additionally, "Conv" refers to the convolutional layer, "BN" to the batch normalization layer, "ReLU" to the rectified linear unit activation mechanism, "Skip" to the identity mapping that performs the skip connection, and "Add" to the element-wise addition operation that performs the skip connection. But rather than using the "Add" operation directly, the "Skip" connection is used, which essentially represents an addition process (element-by-element addition) between x and the "Conv" and "BN" results.

1) Stage 1: building depth and feature learning

In order to improve the feature learning and depth capabilities of the network, stacking multiple residual blocks is a crucial step in the construction of ESRNet. There are F distinct stages to this architectural design, as well as the quantity of leftover blocks in each step increases gradually. The maintenance of skip connections is crucial in order to guarantee a gradient flow that is smooth during training. The underlying structure of ESRNet can be better understood by examining Equation (5) that follows:

$$\begin{aligned} \text{input Layer} &= \text{Image Input Layer}([224\ 224\ 3]) \\ \text{layers} &= \left[\begin{array}{c} \text{input Layer} \\ \text{Conv}(7, \text{F}, \text{Padding},\ \text{S}) \\ \text{BN(Conv)} \\ \text{ReLU}(BN) \\ \text{MaxPooling}(3, S) \end{array}\right] \end{aligned} \tag{5}$$

Starting at the beginning, an input layer is made to hold image data that has $224 \times 224 \times 3$ dimensions. The filter count, represented by the letter F, is 64. Several key layers are stacked in this step in order: rectified linear unit activation, BN after the convolution, a layer of

convolution with a kernel size of 7, and three kernel sizes in the max-pooling layer and suitable stride (S). These procedures work to gradually extract and handle the characteristics of the input data. The solid architecture of the ResNet model is formed by the successive stages building upon this first one.

2) Stage 2: three residual blocks stack with skip connections

In the ESkip-ResNet second stage, skip connections are stacked on top of three residual blocks. In order to decrease the feature map's sample size, the first block's stride is set to 2. It has the following definition in Equation (6) as follows:

$$S = \begin{cases} 1 & \text{for } i = 1 \\ 2 & \text{for } i > 1 \end{cases} \tag{6}$$

In this case, in order to account for the downsampling in the first block and keep the stride at 1 for the blocks that follow, the stride value S fluctuates between 1 and 2 depending on the iteration index i. Utilizing these incremental changes while stacking the three residual blocks allows for effective control of the second stage map of feature size.

3) Stage 3: capture complex elements

In Stage 3, the study improves ESkip-ResNet's ability to extract complex features even more. With a few significant exceptions, this stage expands on the framework established in Stage 2. First, in comparison to Stage 2, the quantity of filters (F) increases by two, allowing ESkip-ResNet to explore more complex representations and patterns. Second, to guarantee spatial reduction, the stride of 2 marks the start of the first residual block, just like in Stage 2. That being said, unlike in Stage 2, where the study keeps the stride of 1 for every residual block that follows, in Stage 3, this work doesn't stop at 1. This tactical decision maintains the feature maps' spatial dimensions for the duration of this phase. The changes made between Stages 2 and 3 support ESkip-ResNet's progressive feature learning, which improves the network's ability to accurately classify brain tumors.

4) Stage 4: elevated-level abstractions

In Stage 4, specific modifications were made in comparison to Stage 3 while still deepening ESkip-ResNet. The proposed study uses twice as many filters (F), just like in the previous stage, so that ESkip-ResNet can obtain even more intricate representations and features. The way the study downsamples the feature maps is where the main distinction is found, though. Although the first block in Stage 3 had a downsampling stride of 2, this stride of 2 was kept for the first block in Stage 4 in order to successfully reduce the spatial dimensions. By lowering the spatial resolution, this decision enables ESRNet to focus on abstractions at the highest level. Rather than stacking f'th residual blocks in Stage 3, it stacks six in Stage 4, which improves ESkip-ResNet's ability to acquire intricate features even more. These changes between Stages 3 and 4 add to ESkip-ResNet's growing representational power and depth, which improves its ACC in fake detection.

5) Stage 5: improved feature and depth learning

In the last and fifth stages, particular modifications are introduced to adjust to the increasing depth while keeping the pattern of architecture established in the previous phases. Similar to Stage 4, in order to enable ESkip-ResNet to efficiently capture high-level features, the proposed work adds two more filters (F). But when it comes to downsampling, the stride value (S) is where the crucial change is found. Similar to Stage 4, the first block in this stage reduces the feature maps' spatial dimensions by using a stride of 2, sharpening the network's ability to prioritize more ethereal depictions. Following the same arrangement as in the earlier phases, three residual blocks are then stacked. The changes made at this stage, particularly the addition of more filters and the purposeful use of stride to downsample, enhance the feature and depth learning of ESkip-ResNet and qualify it for precise deepfake detection.

6) Stage 6: final layers

Last but not least, completely integrated layers with dropout support for regularization come after a global average pooling layer in ESkip-ResNet. A classification layer (CL) and a softmax activation layer complete the architecture.

$$layers = \begin{bmatrix} \text{layers} \\ 2\text{D} - \text{GaP} \\ F_C(128, d_1) \\ F_C(64, d_2) \\ F_C(3) \\ S_M \\ C_L \end{bmatrix} \tag{7}$$

By utilizing the feature maps' global average pooling, the 2D-GaP layer in Equation (7) is essential to global feature extraction. Two crucial fully connected (FC) layers, namely, FC (512, $d1$) and FC (256, $d2$), are positioned within the purpose-driven network. The former has a dropout mechanism and 128 units that drop out at a rate of d_1 for regularization, whereas the latter uses dropout at a rate of 64 units to improve the generalization of the model. The architectural design concludes with an $FC(3)$, d_2 containing three units of output, each of which represents one of the three different classes. After calculating probability distributions using softmax activation, the SoftMax (SM) layer assigns the input data to a class based on the softmax probabilities, and the Classification Layer (CL) performs the final classification. This intricate arrangement of components and layers creates a solid and efficient foundation for precise Bitcoin classification (BTC).

a. Categorical sparse cross-entropy loss

During the training process of ESkip-ResNet for BTC, the sparse categorical cross-entropy (SCCE) loss was selected as the loss function. When representing class labels as integers rather than as one-hot encoded vectors, for tasks involving multi-class classification, this loss function performs well.

The SCCE loss quantifies the difference between the input data samples' actual integer class labels and the class probabilities that the model predicted. By measuring the discrepancy between the predictions, it efficiently directs the training procedure ($y\hat{}$) as well as the labels for ground truth (y), assisting in the neural network's parameter optimization for precise classification outcomes. SCCE has the following mathematical definitions:

$$L(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C}\left(1\{y_i = j\} \cdot \log\left(\hat{y}_{ij}\right)\right) \tag{8}$$

where $L(y, \hat{y})$ symbolizes the loss function in Equation (8), N is the quantity of training examples, C is the quantity of courses (in this case, three for a pituitary tumor, glioma, and meningioma), y_i indicates the ith sample's actual class label, $\hat{y}_{ij}$ shows the expected likelihood that the ith sample will belong to the class j, and

$1\{y_i = j\}$ is a function that indicates and equals 1 when y_i is equal to j and 0 otherwise.

A key step in the training process of ESkip-ResNet is the application of this loss function, which guarantees that the model learns to accurately and intelligently classify fake images into different categories.

3.4. Hyper-parameter tuning using CBO

An algorithm for meta-heuristic optimization is the recently proposed CBO for the hyper-parameter tuning process. It takes its cues from the swarming behavior of the coot, an incredible species of waterbird [23, 24]. In an amazing formation, the coot birds repeatedly rearrange themselves in order to break through large waves by sending their strongest leaders toward the front of the flock. Weak leaders are swapped out for stronger ones in this process, which is repeated. The flock members' energy is conserved, and the flock moves faster toward its objective thanks to this swarming behavior. Below is a brief explanation of the algorithm used by CBO, which draws inspiration from nature.

3.4.1. CBO inspiration

The configuration of the swarming flock of coots alternating between two foraging strategies is incredibly inspirational. The first movement strategy uses a low-density flock of coots and a disjointed allocation. The second flock of coots is more uniformly distributed and regularized in a high-density flock. To speed up their surfing in search of food, they can fly in a third direction in addition to moving on the surface of the water, in two directions [25]. These tactics have been converted into mechanisms for exploration and exploitation, which serve as the foundation concerning the CBO algorithm [26].

3.4.2. CBO code

Each member of the coot flock, including the leader and subordinates, makes up a fraction of the total number of coots in the flock, according to the original algorithm in (N_{Pop}), and there is a mathematical way to express this by $N_{\text{Pop}} = N_{\text{leader}} + N_{\text{coot}}$. The symbols represent the progression of competent subordinate coots and the movement of coots to take the place of less effective leaders Pos_{coot} and $\text{Pos}_{\text{leader}}$, respectively. The following equations are used to initially randomize the coots' positions to begin the algorithm with Equations (9) and (10):

$$\text{Pos}_{\text{coot}} = \text{rand}_{\text{coot}}\text{-}(U_b - L_b) + L_b \tag{9}$$

$$\text{Pos leader}_{\text{le}} = \text{rand}_{\text{leader}} \cdot (U_b - L_b) + L_b \tag{10}$$

wherein the symbols U_b and L_b, respectively, refer to the problem's upper and lower boundaries. As a result, each subordinate coot's fitness is extracted, and the ideal location and score can then be calculated in Equation (11) as follows:

$$\text{Fit}_{\text{coot}}(1 \cdot i) = F_{\text{obj}}(\text{Pos}_{\text{coot}}(i)) \tag{11}$$

Since F_{obj} in Equation (12) shows the function for the fitness goal and i receive numbers between 1 and N_{coot}:

$$\begin{aligned} \text{If Optim}_{\text{score}} &> \text{Fit}_{\text{coot}}(1, i) \\ \text{Optim}_{\text{score}} &= \text{Fit}_{\text{coot}}(1, i) \\ \text{Optim}_{\text{pos}} &= \text{Pos}_{\text{coot}}(i) \end{aligned} \tag{12}$$

Similarly, Equation (13) can be used to extract each leader's fitness, and the following formula can be used to determine the ideal score and position:

$$\text{Fit}_{\text{leader}}(1 \cdot i) = F_{\text{obj}}(\text{Pos}_{\text{leader}}(i)), i \text{ belongs to } N_{\text{leader}} \tag{13}$$

$$\begin{aligned} \text{If Optim}_{\text{score}} &> \text{Fit}_{\text{leader}}(1, i) \\ \text{Optim}_{\text{score}} &= \text{Fit}_{\text{leader}}(1, i) \\ \text{Optim}_{\text{pos}} &= \text{Pos}_{\text{leader}}(i) \end{aligned} \tag{14}$$

Since N_{leader} is the number of leaders in the flock of coots, which makes up a portion of the entire flock N_{Pop}, the rest of N_{Pop} is determined by how many coots are beneath N_{Coot}.

Every leader coot has a subordinate coot that the algorithm randomly selects, and each time around, their positions are improved until the maximum number of iterations is reached $(It_{\max})$ using Equations (15) and (16). From Equation (16), the boundaries are guaranteed to be respected by the new subordinate coot position.

$$\text{Pos}_{\text{coot}}(i) = 2 \cdot \text{rand}_{\text{coot}} \cdot \cos(2 \cdot \pi \cdot r) \text{since, } r = 1 + 2 \cdot \text{rand}_{\text{coot}} \tag{15}$$

$$[\text{Pos}_{\text{leader}}(k) - \text{Pos}_{\text{coot}}(i)] + \text{Pos}_{\text{leader}}(k) \cdot i \in N_{\text{Coot}} \text{ and } k \in N_{\text{leader}} \tag{16}$$

$$\begin{aligned} \text{If Pos}_{\text{coot}}(i) > U_b, \text{ make Pos}_{\text{coot}}(i) = U_b \text{ . If Pos}_{\text{coot}}(i) \\ < L_b, \text{ make Pos}_{\text{coot}}(i) = L_b \end{aligned} \tag{17}$$

where rand $_{\text{coot}}$ and rand $_{\text{leader}}$ are operators that are chosen at random to indicate the leader and subordinate coots' respective positions. Equation (17) is used to calculate each type's fitness in a way that makes it possible to compare and swap out a subordinate coot that performs worse for a stronger one, and vice versa.

$$\begin{aligned} \text{If Fit}_{\text{coot}}(1 \cdot i) < \text{Fit}_{\text{leader}}(1, k), \\ \text{make } \textit{Fit} \text{ leader }_{\text{let}}(1 \cdot k) = \text{Fit}_{\text{coot}}(1, i), \\ \text{and Pos}_{\text{leader}}(k) = \textit{Pos}_{\text{coot}}(i) \end{aligned} \tag{18}$$

The leader coot's positions are upgraded at random making use of Equations (18) and (19). Consequently, the best possible score $(\text{Optim}_{\text{score}})$ and the matching places $(\text{Optim pos}_{\text{pos}})$ are calculated utilizing Equation (19).

$$\begin{aligned} b &= 2 - (\text{It } (L)/It_{\max}) \\ r &= 1 + 2 \cdot \text{rand}_{\text{leader}} \end{aligned} \tag{19}$$

where $(It(L))$ connects to the index of the current iteration and $(\text{I } t_{\max})$. When the optimization process reaches its maximum iteration, it is referred to as in Equation (20).

$$\begin{aligned} \text{Pos}_{\text{leader}} = b \cdot \text{rand}_{\text{leader}} \cdot \cos(2 \cdot \pi \cdot r) \\ \left[\text{Optim}_{\text{pos}} - \text{Pos}_{\text{leader}}(k)\right] + \text{Optim}_{\text{pos}} \\ \text{If Optim} \\ \text{score} \end{aligned} \tag{20}$$

Notably, there are just two parameters in the CBO algorithm that need to be adjusted: the number of people, also referred to as the search agents (N_{Pop}), and the maximum quantity of repetitions

(It_{max}). This demonstrates the benefits of the CBO algorithm in tuning the hyper-parameters of the proposed classifier. Hence, CBO-based ESkip-ResNet classification produces the best results in fake detection.

4. Results and Discussion

4.1. Experimental setup

MATLAB 2023a was used to conduct the experiments. The computer platform had an 11th generation Intel® Core™ i9-11950H vPro® Processor, which had a maximum turbo frequency of 5.00 GHz and a base clock speed of 2.60 GHz. Processing was accelerated by using an NVIDIA® RTX™ A4000 Laptop GPU with 8 GB of GDDR6 graphics memory. 32 GB of DDR4-3200MHz SODIMM RAM, organized into two 16 GB modules, made up the memory capacity, which allowed for effective handling and processing of data throughout the experiments.

4.2. Performance metrics

A popular statistic for assessing the effectiveness of classification models is ACC. According to Equation (21), it calculates the percentage of properly recognized samples out of all samples.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{21}$$

The $F1$-score, a measurement of ACC obtained from a weighted average of PR and RC, is defined by Equation (22). It offers an assessment of the test's capacity to differentiate between favorable and unfavorable results.

$$F1 = \frac{2\times precision\times recall}{precision+recall} \tag{22}$$

Equation (23), which introduces the PR rate, assesses how well a model can predict positive samples among those it considers to be positive.

$$Precision = \frac{TP}{TP+FP} \tag{23}$$

Equation (24), sometimes referred to as the true positive rate or sensitivity, measures how well a prediction model picks up on true positive data. The true positive to total true positive and false negative ratio is used to calculate it.

$$Recall = \frac{TP}{TP+FN} \tag{24}$$

4.3. Analysis of classification process

In Table 2, the models' performance metrics including *ACC*, *PR*, *RC*, and $F1$-score are presented. VGGNet, ResNet, PolyNet, and the proposed model are among the models. VGGNet shows 92.9% ACC with PR, RC, and $F1$-score values of 90.6%, 89.9%, and 90.2%, respectively. With a higher RC (94.3%) and PR (91.5%), ResNet's ACC of 93.5% is marginally higher than other networks, resulting in an $F1$-score of 92.5%. Although PolyNet has a 95.6% ACC rate, it has a slightly lower PR, RC, and $F1$-score (93.9%, 90.2%, and 89.5%, respectively). But overall, PolyNet performs better. In comparison with the other models in the table, the proposed model performed better in fake detection, with an ACC of 96.2% and better PR (94.1%), RC (95.2%), and $F1$-score (95.4%).

Table 2
ESkip-ResNet classification without CBO

Models	*ACC* (%)	*PR* (%)	*RC* (%)	*F*1 (%)
VGGNet	92.9	90.6	89.9	90.2
ResNet	93.5	91.5	94.3	92.5
PolyNet	95.6	93.9	90.2	89.5
Proposed model	**96.2**	**94.1**	**95.2**	**95.4**

Table 3
CBO-based ESkip-ResNet classification

Models	*ACC* (%)	*PR* (%)	*RC* (%)	*F*1 (%)
AlexNet	94.3	93.9	94.5	94.8
ResNet	95.7	95.6	96.7	95.8
PolyNet	96.5	96.4	96.3	95.9
Proposed model	**98.9**	**98.8**	**98.7**	**98.6**

Table 3 shows that AlexNet achieves an ACC of 94.3% with high RC (94.5%), PR (93.9%), and $F1$-score (94.8%). With a 95.7% ACC rate, superior PR (95.6%), RC (96.7%), and an $F1$-score of 95.8%, ResNet is still getting better. PolyNet keeps performing well, producing an $F1$-score of 95.9% with an ACC of 96.5%, well-balanced RC (96.3%), and PR (96.4%). However, the suggested model outperforms the other models in the table, achieving an astounding 98.9% ACC as well as remarkable RC (98.7%), PR (98.8%), and $F1$-score (98.6%). Its outstanding performance is apparent in every metric that has been assessed. VGGNet is applicable without the CBO technique, and AlexNet works with the CBO technique only, and that's why each table contains different methods during the comparison.

5. Conclusion

Proposed detection methods need to advance more quickly to keep up with the rapidly rising quality of deepfakes. Deepfake producers and catchers are facing more and more competition every day. The idea behind the deepfake detection model that has been suggested is that DL methods can address issues that arise from other DL methods. The DFDC dataset was used in this work for additional processing. Subsequently, the A2WF was employed in the proposed study's preprocessing phase to minimize undesired noise and improve contrast while enhancing the images. Accurate detection is achieved through the classification process that follows the preprocessing phase. Therefore, ESkip-ResNet is suggested as a method for accurately detecting and classifying images. A hyper-parameter tuning procedure was suggested using this work, which is based on the CBO algorithm with inspiration from nature. The suggested ESkip-ResNet classifier's hyper-parameters are adjusted using this CBO algorithm. Overall, with 98.9% ACC for fake detection, ESkip-ResNet proved to be a strong and effective framework, offering better performance and efficiency in addressing crucial issues in the fake network domain.

Acknowledgment

The authors are grateful to Saveetha School of Engineering, SIMATS University, Chennai, for providing the support in producing this research paper.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

The DFDC data that support the findings of this study are openly available at https://doi.org/10.48550/arXiv.2102.11126, reference number [20]. The Deepfake Detection Challenge datasets that support the findings of this study are openly available at https://www.kaggle.com/c/deepfake-detection-challenge.

Author Contribution Statement

V. Gokula Krishnan: Conceptualization, Methodology, Writing – original draft, Writing – review & editing, Project administration. **R. Vadivel:** Software, Visualization. **K. Sankar:** Validation, Resources. **K. Sathyamoorthy:** Formal analysis, Data curation. **B. Prathusha Laxmi:** Investigation, Supervision.

References

[1] GitHub. (2024). *Deepfakes software for all.* https://github.com/deepfakes/faceswap

[2] Kietzmann, J., Lee, L. W., McCarthy, I. P., & Kietzmann, T. C. (2020). Deepfakes: Trick or treat? *Business Horizons*, *63*(2), 135–146. https://doi.org/10.1016/j.bushor.2019.11.006

[3] Tucker, P. (2019). *The newest AI-enabled weapon: 'Deep-faking' photos of the earth.* DefenseOne. https://www.defenseone.com/technology/2019/03/next-phase-ai-deep-faking-whole-world-and-china-ahead/155944/

[4] Wu, Z., Shen, C., & van den Hengel, A. (2019). Wider or deeper: Revisiting the ResNet model for visual recognition. *Pattern Recognition*, *90*, 119–133. https://doi.org/10.1016/j.patcog.2019.01.006

[5] Tolosana, R., Vera-Rodriguez, R., Fierrez, J., Morales, A., & Ortega-Garcia, J. (2020). Deepfakes and beyond: A survey of face manipulation and fake detection. *Information Fusion*, *64*, 131–148. https://doi.org/10.1016/j.inffus.2020.06.014

[6] Ismail, A., Elpeltagy, M., Zaki, M. S., & Eldahshan, K. (2021). A new deep learning-based methodology for video deepfake detection using XGBoost. *Sensors*, *21*(16), 5413. https://doi.org/10.3390/s21165413

[7] Hsu, C.-C., Zhuang, Y.-X., & Lee, C.-Y. (2020). Deep fake image detection based on pairwise learning. *Applied Sciences*, *10*(1), 370. https://doi.org/10.3390/app10010370

[8] Mirsky, Y., & Lee, W. (2022). The creation and detection of deepfakes: A survey. *ACM Computing Surveys*, *54*(1), 7. https://doi.org/10.1145/3425780

[9] Jung, T., Kim, S., & Kim, K. (2020). DeepVision: Deepfakes detection using human eye blinking pattern. *IEEE Access*, *8*, 83144–83154. https://doi.org/10.1109/ACCESS.2020.2988660

[10] Nguyen, T. T., Nguyen, Q. V. H., Nguyen, D. T., Nguyen, D. T., Huynh-The, T., Nahavandi, S., . . . , & Nguyen, C. M. (2022). Deep learning for deep fakes creation and detection: A survey. *Computer Vision and Image Understanding, 223*, 103525. https://doi.org/10.1016/j.cviu.2022.103525

[11] Khormali, A., & Yuan, J.-S. (2021). ADD: Attention-based DeepFake detection approach. *Big Data and Cognitive Computing*, *5*(4), 49. https://doi.org/10.3390/bdcc5040049

[12] Montserrat, D. M., Hao, H., Yarlagadda, S. K., Baireddy, S., Shao, R., Horváth, J., . . . , & Delp, E. J. (2020). Deepfakes detection with automatic face weighting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2851–2859. https://doi.org/10.1109/CVPRW50498.2020.00342

[13] Su, Y., Xia, H., Liang, Q., & Nie, W. (2021). Exposing deepfake videos using attention based convolutional LSTM network. *Neural Processing Letters*, *53*(6), 4159–4175. https://doi.org/10.1007/s11063-021-10588-6

[14] Iqbal, F., Abbasi, A., Javed, A. R., Almadhor, A., Jalil, Z., Anwar, S., & Rida, I. (2024). Data augmentation-based novel deep learning method for deepfaked images detection. *ACM Transactions on Multimedia Computing, Communications and Applications, 20*(11), 339. https://dl.acm.org/doi/pdf/10.1145/3592615

[15] Xue, Z., Jiang, X., Liu, Q., & Wei, Z. (2023). Global–local facial fusion based GAN generated fake face detection. *Sensors*, *23*(2), 616. https://doi.org/10.3390/s23020616

[16] Hamid, Y., Elyassami, S., Gulzar, Y., Balasaraswathi, V. R., Habuza, T., & Wani, S. (2023). An improvised CNN model for fake image detection. *International Journal of Information Technology*, *15*(1), 5–15. https://doi.org/10.1007/s41870-022-01130-5

[17] Rafique, R., Gantassi, R., Amin, R., Frnda, J., Mustapha, A., & Alshehri, A. H. (2023). Deep fake detection and classification using error-level analysis and deep learning. *Scientific Reports*, *13*(1), 7422. https://doi.org/10.1038/s41598-023-34629-3

[18] Coccomini, D. A., Caldelli, R., Falchi, F., & Gennaro, C. (2023). On the generalization of deep learning models in video deepfake detection. *Journal of Imaging*, *9*(5), 89. https://doi.org/10.3390/jimaging9050089

[19] Guan, L., Liu, F., Zhang, R., Liu, J., & Tang, Y. (2023). MCW: A generalizable deepfake detection method for few-shot learning. *Sensors*, *23*(21), 8763. https://doi.org/10.3390/s23218763

[20] Wodajo, D., & Atnafu, S. (2021). Deepfake video detection using convolutional vision transformer. *arXiv Preprint:2102.11126*. https://doi.org/10.48550/arXiv.2102.11126

[21] Yoon, S., & Cho, J. (2023). Low-light image contrast enhancement with adaptive noise attenuator for augmented vehicle detection. *Electronics*, *12*(16), 3517. https://doi.org/10.3390/electronics12163517

[22] Kaur, M., AlZubi, A. A., Jain, A., Singh, D., Yadav, V., & Alkhayyat, A. (2023). DSCNet: Deep skip connections-based dense network for ALL diagnosis using peripheral blood smear images. *Diagnostics*, *13*(17), 2752. https://doi.org/10.3390/diagnostics13172752

[23] Naruei, I., & Keynia, F. (2021). A new optimization method based on COOT bird natural life model. *Expert Systems with Applications*, *183*, 115352. https://doi.org/10.1016/j.eswa.2021.115352

[24] Lyon, B. E., & Shizuka, D. (2020). Extreme offspring ornamentation in American coots is favored by selection

within families, not benefits to conspecific brood parasites. *Proceedings of the National Academy of Sciences*, *117*(4), 2056–2064. https://doi.org/10.1073/pnas.1913615117

[25] Salai, K. E., Mansouri, I., Squalli, W., El Hassani, A., Dakki, M., & Zine, N. E. (2021). Nesting features and breeding chronology of the crested coot (*Fulica cristata*) in two North African high altitude wetlands. *Journal of Animal Behaviour and Biometeorology*, *9*(3), 2129. https://doi.org/10.31893/jabb.21029

[26] Gouda, E. A., Kotb, M. F., Ghoneim, S. S. M., Al-Harthi, M. M., & El-Fergany, A. A. (2021). Performance assessment of solar generating units based on coot bird metaheuristic optimizer. *IEEE Access*, *9*, 111616–111632. https://doi.org/10.1109/ACCESS.2021.3103146

How to Cite: Gokula Krishnan, V., Vadivel, R., Sankar, K., Sathyamoorthy, K., & Prathusha Laxmi, B. (2025). Coot Bird Optimization-Based ESkip-ResNet Classification for Deepfake Detection. *Journal of Computational and Cognitive Engineering*, *4*(4), 561–569. https://doi.org/10.47852/bonviewJCCE42022955