IRAQI Academic Scientific Journals

SAMARRA JOURNAL OF ENGINEERING SCIENCE AND RESEARCH

S.J.E.S.R

Multi-class classification framework for ear imaging using attention-based hybrid dense CNN and non-linear manifold approximation

Mohammed Mahmood Ali Alezzi

School of Engineering and Natural Sciences, Electrical and Computer Engineering, Altınbaş University, Istanbul, Turkey

Article Information

Received: 10/11/2024
Accepted: 30/4/2025

Keywords

Ear Identification, Convolutional Neural Network, Attention, Deep Feature Fusion, Instance Learning.

Corresponding Author

Email: mohammed.mahmood1988ali@gmail.com

Abstract

The ear identification problem is the task of recognizing or authenticating an individual's identity based on the unique characteristics of their ear. Like fingerprints or face recognition, ear identification tasks mainly depend on the unique shape, structure, and geometry of the ear for accurate classification. Moreover, realizing high classification accuracy is challenging for both security systems and forensic investigations since it boosts the robustness of security systems and favors law enforcement efforts. In this work, we develop a novel deep learning-based multi-class classification approach for ear images that uses a hybrid-dense CNN architecture combined with a self-attention mechanism and uniform manifold approximation and projection (UMAP) for dimensionality reduction. We conducted our experiments with ear images labeled for multi-class classification, using a Dense CNN network built from Xception-Net and Dense-Net architectures to capture intricate patterns. The self-attention mechanism enhances feature representation, while UMAP optimizes feature space by simplifying high-dimensional data. The model processes low-dimensional features through fully connected layers and softmax for classification.

The experimental findings of the proposed methodology showed classification accuracies of 98.24% on the first dataset and 94.66% on the second. When compared to cutting-edge models like VGG-Net, Inception Networks, and ResNet-50, our technique outperformed them in terms of classification accuracy, precision, recall, and F1 score. This framework has great potential for use in a variety of applications, including biometric identification, medical diagnostics, forensic science, security systems, and assistive technology.

1. Introduction

In recent years, substantial developments in computer vision, and artificial intelligence (ai) have enabled automatic image interpretation and analysis for medical studies. Among these, realistic recreation of ear pictures shows considerable promise for biometrics and medical diagnosis. The internal architecture, shape, and structure of ear pictures include rich

information that modern imaging techniques can use. However, existing ear classification models depend upon handcrafted feature extraction approaches and conventional classification techniques. Most often, these models suffer from poor feature representation power and lower generalization ability. As a result, they struggle to maintain high classification accuracy, especially in real-world, large-scale datasets with significant variability and minor variances. This emphasizes the need for more robust deep learning algorithms that can improve feature extraction and discriminative capability and give more consistent results under a variety of situations.

Recent advances in the field of convolutional neural networks (CNNs) highlight their powerful feature representation capabilities, which aid in the development of deeper models for ear identification [1]. Common classification methods of handling ear recognition problems usually deal with it as a binary classification. These deep CNNs can automatically learn and extract hierarchical features from raw ear pictures, detecting intricate patterns that older approaches frequently ignore. As a result, deep CNNs can detect small differences in ear forms, structures, and textures, considerably enhancing classification accuracy. Furthermore, including techniques such as self-attention and hybrid architectures improves the model's ability to focus on the most important features, hence increasing performance. These advancements pave the path for more accurate and dependable ear recognition systems, making them more relevant to real-world biometric and medical applications."

Several approaches have been developed to explore ear classification problems using various deep learning (DL) methods [2-4]. For example, Korichi et al. [5] developed a fast CNN network for recognizing ear identifiers and used Principal Component Analysis (PCA) for geometrical normalization of scale and pose. In this study, Tied-Rank (TR) normalization [5] was employed to maximize the variance among the output patterns. Later, a soft-max average fusion scheme was implemented to incorporate deep networks to enhance recognition outcomes. Sharkas [6] implemented a two-step model for ear image classification. In the first step, discrete curvelet transforms were employed for feature extraction and fed into various classifiers. In the second step, various end-to-end DL networks were used for ear image classification. Features were collected from each network and fed to a shallow classifier for ear pattern recognition. Here, PCA was used to reduce irrelevant visual patterns from the extracted feature set. Overall, the research concluded the supremacy of ensemble methods in the identification of true visual patterns in ear classification.

Alshazly et al. [7] developed an ear recognition framework based on ensembles of deep CNN models similar to Visual Geometry Group (VGG) architectures for extracting discriminative deep ear imaging features. The outcomes showed that ensembles of models realized the best performance with significant enhancement over the state-of-the-art results. Tian et al. [8] implemented a CNN with three convolutional layers, a fully connected layer, and a soft-max classifier and computed competitive outcomes on the USTB ear database. Emerson et al. [9] designed a novel CNN for limited ear samples using ALEX-net-like architectures [10], VGG-net 16 [11], squeeze-net [12], and aggressive data augmentation techniques [13]. This framework enhanced the recognition rate by 25% compared to state-of-the-art models. Priyadharshini et al. [14] framed a six-layer deep CNN architecture for ear recognition. The proposed model was evaluated using the IITD-II [15] and AMI [16] ear datasets, achieving recognition rates of 97.36% and 96.99%, respectively. Ganapathy et al. [17] proposed an ear

recognition model using an ensemble of CNNs. The initial part of the model trains three models of CNN on a given dataset, whereas in the later part, the weighted average of the outputs of trained models is utilized to detect the ear regions.

In a different work [18], an ear classification system for real-time scenarios was presented using the Haar feature and cascaded AdaBoost classifier. Cascade AdaBoost classifier organizes classifiers such that a segment is passed to the next classifier; once a strong classifier accepts it, it is passed to the next classifier. Their research noticed that the cascaded AdaBoost classifiers take less time than the simple AdaBoost version because most unimportant segments are discarded at an early stage. On the other hand, they have a unique way of helping them through noisy environments, different-sized crowds, and those that are obscured. In a different study [19], authors developed a technique for detecting ears using edge recognition and template matching. Initially, the focus was on segmenting the skin and identifying the tip of the nose. Once both steps are completed, the face region is isolated as it is likely to contain an ear. In the edge-based approach, the extracted region was run through a connected component labeling process, and a rectangle was drawn around the area with the highest number of connected edges. On the other hand, the template-based method involved creating an ear image template by averaging intensity levels. The normalized correlation coefficient (NCC) [20] was then calculated for each pixel. The effectiveness of this method was tested on the CVL database, revealing that the edge-based approach yielded more precise results compared to the template-based one. A major strength of their approach was its simplicity, making it user-friendly and easy to adapt.

The authors proposed a promising alternative in [21] with a connected components graph-based approach for ear detection. This technique has been successfully tested on Indian Institute of Technology Kanpur Ear Dataset (IIT-K), University of Notre Dame Ear Dataset (UND-E), and University of Notre Dame Ear Dataset UND-J2. One notable advantage of this method is its ability to maximize the variations in the pose, scale, and shape of the ear. However, it may struggle to detect ears in images that are occluded by hair, noise, or other visual obstructions. In [22], ear detection was implemented through geometry, specifically using three parameters: elongation, compactness, and rounded boundary. The technique was tested on the UND-J2 database and proved to be efficient with promising results. However, it is important to mention that the UND-J2 database is a small dataset with a limited number of images. In an individual experiment [23], a novel entropy cum Hough transformation approach was employed. By utilizing both an ear localizer and an ellipsoid ear classifier, the presence of an ear in a facial image was successfully identified. The effectiveness of this technique was then tested on five prominent databases, namely FERET, Pointing Head Pose, UMIST, CMU-PIE, and FEI. To assess accurate ear region detection, the localization error rate was utilized and calculated by measuring the distance between the center of the detected ear region and the annotated ground truth.

The above-mentioned state-of-the-art studies reflect the potential of DL techniques in ear classification. The success of these models mainly depends on the discriminative nature of extracted features and applied fusion techniques following the minimum redundancy and maximum relevance paradigm [24]. In practice, these considerations are difficult to retain throughout the classification because of the black-box nature of DL approaches and limited technical expertise.

In this study, we worked on two key steps: (1) In the first step, we extracted normal and scaled deep features by designing Cross-Axial attention [25] enabled two individual CNNs and merged them using Feature Pyramid Network (FPN) [26], and (2) A UMAP approach [27] is used for reducing high-dimensionality of the fused features with three fully-connected and one SoftMax layer for recognizing true class labels. Our choice of XceptionNet [28], DenseNet [29], and UMAP is driven by the requirement for a fast, high-performance feature extraction and dimensionality reduction pipeline for the classification of ear images. XceptionNet, using its depthwise separable convolution, effectively separates spatial and channel-wise feature learning, significantly alleviating computational overhead at the cost of retaining high-fidelity spatial representations essential for the analysis of ear structure. DenseNet, by contrast, exploits dense connectivity to promote feature reuse, reduce vanishing gradients, and improve the learning of fine-grained discriminative patterns. In order to further refine and combine the obtained deep and scaled features, we use Cross-Axial Attention and Feature Pyramid Network (FPN) in a way that provides strong multi-scale representation. Lastly, UMAP is selected for its local and global structure preservation in high-dimensional data and superior class separation compared to other conventional methods such as PCA and t-SNE. This fusion not only results in improved classification accuracy but also guarantees a computationally efficient and scalable solution that makes it perfectly applicable to real-world ear imaging problems.

The reduced features were further discriminated using three fully connected and one softmax layer for the detection of accurate ear labels. The proposed classification framework is visualized in Fig. 1. We validated the performance of our model using a five-fold cross-validation technique over two publically available datasets: (1) EarVN1.0 and (2) IIT DELHI Dataset and compared our various state-of-the-art CNN models.

The overall contribution of the proposed research work can be summarized into the following points:

1. Deep features extraction from two Cross-Axial Attention enabled individual CNNs: (1) Xception-net and (2) DENSE-net and merged using a customized Feature Pyramid Network.
2. The UMAP technique is used for dimensionality reduction of the fused features and reducing redundancies.
3. The classification pipeline constitutes three fully connected layers and one softmax layer and applies feature unification, resulting in a comprehensive representation of multi-scale information using CAFPN.
4. Performance validation on two popular open-source datasets and comparing the global performance with baseline CNN architectures (VGG Net [11], Inception-net [30], and Resnet-50 [31]).

The remaining structure of the paper can be summarized as: Section 2 recapitulates the dataset details and state-of-the-art CNN models used for comparing the global results. A brief discussion of the proposed methodology is covered in Section 3. The results comparison and relevance of the study are given in Section 4. Finally, the conclusion and future scope of the proposed study are summarized in Section 5.

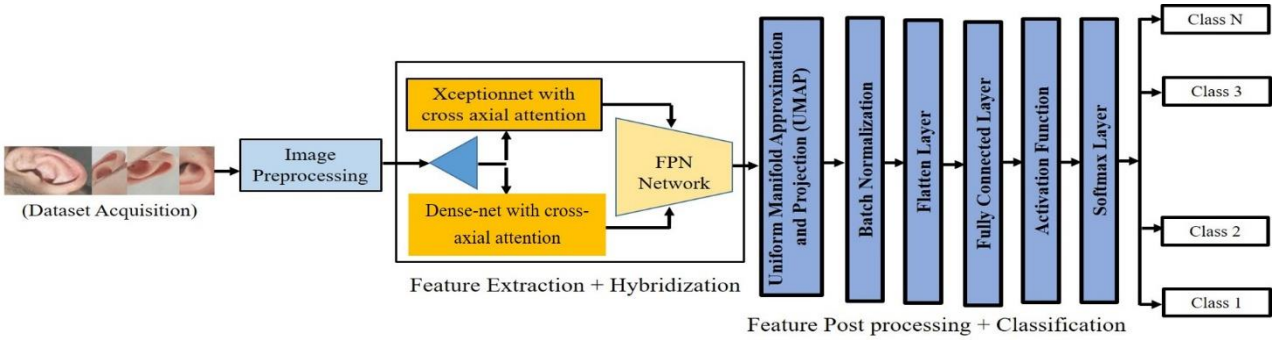

Fig. 1. Proposed hybrid classification framework for Ear Recognition.

2. Dataset Details

2.1 Dataset Description

This study evaluates the proposed classification framework using two publicly available benchmark datasets: EarVN1.0 and the IIT Delhi Ear Dataset. These datasets were selected for their diversity in ear image characteristics, scale, and acquisition conditions, ensuring a robust assessment of model generalizability.

2.1.1 Dataset 1 (DS1)

In our experiment, EarVN1.0 [16] was used as the first demonstration dataset. The EarVN1.0 dataset holds ear photographs of one hundred and sixty-four (164) Asian individuals who were sampled in the year 2018, making it one of the largest publicly available datasets. This dataset comprises 28,412 color ear images captured from 98 males and 66 females. This is unique from previous studies as it shows both ears of people under normal conditions. For example, uncontrolled environmental conditions, such as camera movements, lighting. These were captured in native facial photos. As a result, ear pictures are cropped out of face images to handle numerous kinds of changes like pose change, illumination, and scale variations that may be possible. Some machine learning tasks include ear recognition, among others, which are image classification or clustering, gender recognition right/left detection, and improved super-resolution. Sample images from EarVN1.0 are shown in Fig. 2(A).

2.1.2 Dataset 2 (DS2)

The IIT Delhi ear dataset is another popular repository, consisting of the ear image database gathered from the students and staff members at IIT Delhi, New Delhi, India [15]. It is a simple imaging setup that has allowed this to take place within the IIT Delhi campus during the period between October 2006 and June 2007 (still in progress). These images have been taken at a distance using a touchless method available through a simple imaging setup in an indoor environment. The currently available database is acquired from the 121 different subjects and each subject has at least three ear images. All the subjects in the database are aged between 14-58 years. The images consisting of these are numbered sequentially for every user with an integer identification/number with the total number of images being 471 on a resolution which is given as 272 × 204 pixels and all these images are available in jpeg format. In addition to original pictures, there are also automatically normalized and cropped

ear images, which measure about 50x180 pixels in size. Sample images from IIT Delhi are shown in Fig. 2(B).

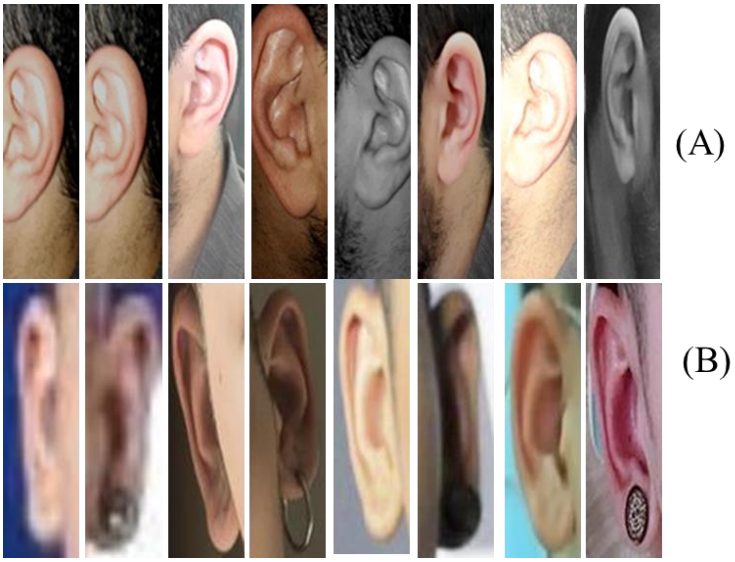

Fig. 2. Sample ear images from EarVN1.0 (A) and IIT Delhi dataset (B).

3. Proposed Methodology

The proposed deep learning-based ear classification procedure is implemented in two phases. In the initial phase, a training process is employed to detect spatial and cross-channel relationships and understand hierarchical feature representations of input images. The second step is the evaluation stage, where the trained ear classification framework is used to detect true class labels of unknown instances that were not used earlier during the training process. In this step, several performance measures, such as classification accuracy, precision, recall, and Receiver Operating Curve (ROC), were used to evaluate the performance of the proposed model and compare it with baseline studies. In both steps, the proposed approach involves data pre-processing, feature extraction, and classification. The primary goal of data pre-processing is to enhance the input data quality by minimizing noise and outliers from original ear samples. In the feature extraction phase, two well-known pre-trained DL classification frameworks, (1) XceptionNet and (2) DenseNet, were used to extract deep features from the input images. Here, we added a novel ear scan-specific cross-channel attention mechanism with both DL models to focus on relevant parts of the input information while ignoring irrelevant information. This cross-channel attention helped to enhance the model's ability to extract meaningful patterns from the input, particularly in scenarios where the input data is complex. The extracted features from both DL architectures were merged using a customized FPN, and the size of the hybridized feature set was further reduced by the UMAP technique. Finally, a set of pre-processing methods such as batch normalization, flattening, and softmax layer were applied to discriminate subject-specific ear imaging patterns. The flowchart of the proposed methodology is visualized in the Fig. 3. A detailed work of the proposed architecture and its performance are discussed in subsequent sections.

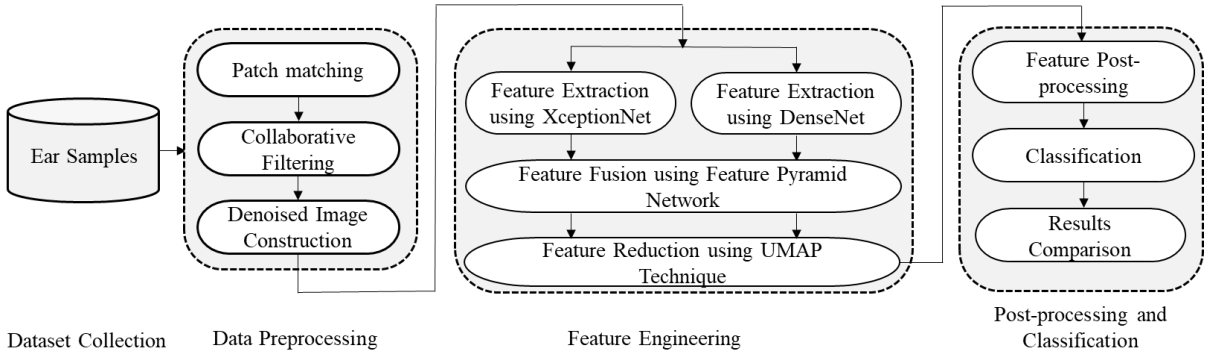

Fig. 3. Flowchart of the proposed methodology for ear classification.

3.1 Image pre-processing

Image pre-processing is an important step of the proposed classification framework. BM3D (Block-Matching 3D) filtering is an image-denoising technique proposed by Dabov et al. in 2007 and used to enhance the quality of input scans [32] by identifying and collaboratively filtering pixel blocks within an active window to yield a denoised image version. BM3D is essentially a coherent noise elimination device operating in the transform domain, for example, the discrete cosine transform (DCT), which utilizes the sparsity of natural images, leading to higher visual denoising quality. BM3D outperforms in preserving vital image structures and details and in noise canceling at the same time, so it is best used in situations where image quality and detail preservation are necessary. The working of the applied BM3D algorithm is given in Algorithm 1.

Algorithm 1: Pseudocode of customized BM3D algorithm

Input: Patches of size 224×224 pixels of original ear images	
Output:	$Output\ (i,j) = \frac{1}{N_{overlap}(i,j)} \sum_{Overlapping\ \ Blocks} D(i,j)$ where $N_{overlap}(i,j)$ is the number of overlapping blocks at pixel (i,j)
Step 1	The input patch I is partitioned into overlapping blocks B_i of size N×N where i represent the *ith* block.

Step 2	*# Patch matching using Eq. 1*
	for each noisy patch:
	denoised_patch = noisy_patch
	for iteration in range(num_iterations):
	# Search for similar patches
	similar_patches=find_similar_patches(noisy_patch)
	# Collaborative Filtering
	denoised_patch = collaborative_filtering(similar_patches)

Step 3	***Function*** *for Collaborative Filtering using Eq. 2 and 3* for each selected similar patch: clean_patch=estimate_clean_patch(similar_patch) clean_patch=wiener_filter(clean_patch)
Step 4	***Function*** *for Aggregation (Iterative) using Eq. 4* for each denoised patch: aggregated_denoised_image+=denoised_patch
Step 5	Display denoised image
	final_denoised_image = aggregate_patches(aggregated_denoised_image) display(final_denoised_image)

Step 1: ***Patch matching***

$$D(P_i, P_j) = \sqrt{\sum_{K=1}^{N}(p_i(K) - p_j(K))^2} \quad (1)$$

where P_i *and* P_j are two different patches and $D(P_i, P_j)$ refers Euceadian distance between patches.

Step 2: ***Collaborative Filtering***

- Clean patch $\hat{P}$ estimation using collaborative filtering

$$\hat{P} = \sum_{J=1}^{K} w_j \times P_j \quad (2)$$

- Estimation of w_j using Wiener filtering

$$w_j = \frac{1}{{\sigma_n}^2 + \sigma_d^2} \times \frac{|H|^2}{|H|^2 + \left|\frac{N}{P}\right| \times {\sigma_n}^2} \quad (3)$$

Step 3: ***Denoised Image Construction***

$$I_{denoised} = \frac{\sum_{i=1}^{K} w_i \times I_i(x,y)}{\sum_{i=1}^{K} w_i} \quad (4)$$

Here, $p_i(K)$ *and* $p_j(K)$ represents the Kth pixel intensity of patches *pi* and *pj* respectively, N is the patch size, K is the number of similar patches, ${\sigma_n}^2$ is the noise variance, σ_d^2 is the variance of the clean patch, H is the Fourier transform of the Wiener filter, N is the size of the search window, and P is the size of the patch. The parameters of the BM3D allow the possibility of tuning the model to achieve the best performance. Among the main critical factors, block size matters as it makes the resolution of block-matching and also has a great influence on denoising. In this case, the larger block sizes are favorable for image regions with relatively smooth planes, whereas the smaller ones are better for the textured regions. The size of the search window is another factor that impacts BM3D's capacity to detect like blocks, large windows capturing variations but potentially leading to high computational cost. The thresholding parameters should be realized, as they provide the anchor for blocking artifacts

between the noise and signal. The critical point in this is a balancing of these parameters, which is the key to achieving optimal performance of BM3D in image denoising tasks.

In the proposed denoising step, the best results are computed in terms of the Peak signal-to-noise (PSNR) ratio using the following tuned parameters.

Block_size = 32*32

Sigma (Standard deviation of noise) = 3.74

Transform_domain = Discrete Cosine Transform

Search_window = 16 pixels

SNR = 1

The samples of five noised and denoised images of class 1 are shown in Fig. 4.

Fig. 4. Samples of original and denoised images from class 1.

3.2 Feature Extraction using XceptionNet and DenseNet

Feature extraction is a process of identifying and retrieving relevant features from raw ear images for further input to neural networks or other machine learning model components. In the ear classification problem, two popular DL architectures, (1) Extreme Inception Net (XceptionNet) and (2) DenseNet with a cross-channel attention mechanism, were used to collect deep features from input scans. XceptionNet, the abbreviated form of "Extreme Inception," is a CNN architecture conceived by François Chollet in 2017 [28]. It is the transformation of the Inception architecture and presents a new idea, namely depth-wise separable convolutions. Depthwise separable convolutions replace the standard convolutional layers in traditional CNN architectures with two separate operations: depthwise convolution and pointwise convolution. This division decreases computational complexity, although it still maintains representational capacity. Therefore, such feature extraction is more efficient and effective. DenseNet is another popular neural network architecture that boosts performance and efficiency by using dense connections between layers [29]. In DenseNet, each layer receives inputs from all preceding layers and contributes its features to all subsequent layers, facilitating feature reuse and reducing the number of parameters needed. This dense connectivity improves the flow of gradients during training, making deep networks more stable and easier to optimize. The hybrid architecture of CNN based on XceptionNet and

DenseNet comprehensively inherits the feature extraction performance of XceptionNet and DenseNet, inserting attention mechanism and FPN to further higher image classification

accuracy. Fundamentally, architecture is based on block XceptionNet, each block has a convolution depth-wise separable multiplication by 128 filters with size 3 by 3, and they are followed by a point convolution convolution multiplication by 256 filters of size 1 by 1. These blocks emphasize dimensionality reduction and efficient capture of local characteristics across all the space levels. DenseNet-like layers, which consist of convolutional layers densely connected with a growth rate of 32 correspondingly, also play a significant role in information flow by transferring information from all preceding layers. Further, the Attention mechanism is introduced to the convolutional filter feature maps in both blocks to emphasize only relevant features. The dedicated attention layers interleaved with the convolution layers in each block generate attention weights that boost importance features by selective amplifying. Specifically, the attention layers have 64 filters of size 3×3 that give the model greater flexibility because they can be used to target discriminative regions of the inputs.

The hybrid CNN architecture features various blocks of various sizes, and it further combines these blocks with different configurations while extracting various spatial scales of the feature. The XceptionNet-type blocks compose two depth-wise separable convolutions with output from 128 filters of 3x3 followed by 1x1 point-wise convolutions with 256 filters. However, the DenseNet-like layers that have high connectivity are comprised of several layers, with a growth rate of 32; it's what is needed to ensure feature reuse and propagation. As a result, instead of 64 filters in the 3x3 of attention layers within each block, the attention uses feature map modulation based on learned attention weights. The detailed working of the feature hybridization process is shown in Fig. 5.

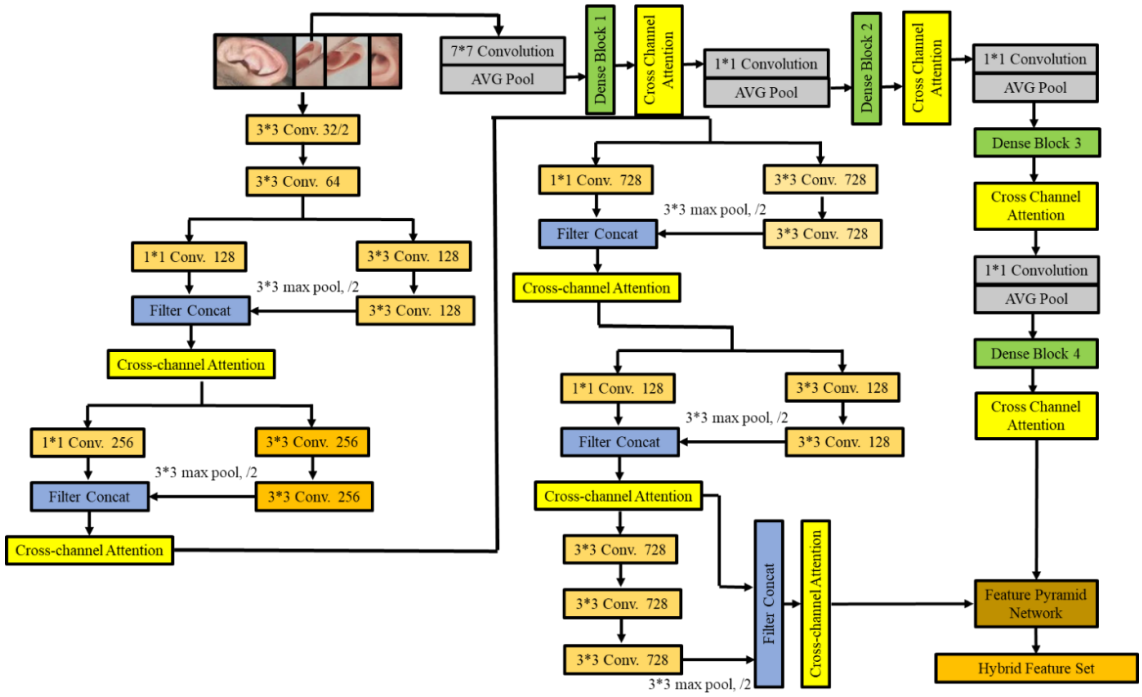

Fig. 5. Proposed hybridization procedure using cross-channel attention enabled XceptionNet and DenseNet architectures.

3.2.1 Integration using Feature Pyramid Network (FPN)

Besides feature extraction from both XceptionNet and DenseNet blocks, the Feature Pyramid Network (FPN) scheme is implemented to maintain the multi-scale features fusion respectively. FPN is based on lateral connections and top-down directed pathways that sum up the input of different resolutions into a single representation. First, a feature pyramid is created from the outputs of both XceptionNet and DenseNet blocks. Feature maps of the highest resolution layer from each block are utilized as the starting point. Therefore, these feature maps are further stacked with layers to get a pyramid of features with additional resolutions of space.

The combination of feature mapping of different resolutions is implemented via lateral connections and top-down pathways. In lateral connections, some convolutional layers with one-by-one filters with 64 filters are added to lower-resolution feature maps to match the channel dimensions of the higher-resolution feature maps. The lateral connections in this architecture affect the smooth blending of features across resolutions. Finally, high-resolution feature maps utilize top-down pathways to transfer information to lower-resolution ones. This operation is done by interpolation of the feature maps from the higher resolution and adding them to the corresponding feature maps from the lower resolution. This procedure makes sure that all the details from every resolution are added to the final output. Moreover, attention mechanisms are incorporated into the fusion process to selectively upscale functionality. Attention layers are used to implement 32 filters of the size 3x3 on fused feature maps to obtain modulated feature responses depending on the learned attention weights.

In our case, DenseNet yields feature maps at 64x64, 32x32, and 16x16 quadratures. These feature maps correspond to different levels of abstraction from the higher resolutions with the finer details captured in the (64x64) maps and more abstract representations found in the (16x16) maps. In addition, XceptionNet, an evolution of InceptionNet with separable depthwise convolutions, replicates different degrees of complexity quite flawlessly. The XceptionNet produces the feature maps at the resolution levels of 128x128, 64x64, and 32x32 pixels. Like DenseNet, these feature maps contain information at various scales, and the highest resolution maps (128x128) capture fine details while the lowest resolution maps (32x32) retain more abstract features.

After feature extraction, FPN creates a feature pyramid which is composed of levels taking different scales. The pyramidal structure of the network is represented by each level which refers to a specific spatial resolution that helps in making the network capable of multi-scale information capture. The feature pyramid is organized hierarchically, starting with the highest spatial resolution in Level 1 (128x128 pixels), Level 2 (64x64 pixels) with a slightly lower resolution, and Level 3 (32x32 pixels) with an even lower resolution, and so on. At each level of the pyramid, where the features of DenseNet and XceptionNet do not have the same resolution, we employed bi-linear interpolation to alienate those features. This process is performed on the DenseNet features to match them with the resolution of the XceptionNet features. In Equation (5) and (6), level-wise, two upsampled procedures are shown.

At Level 2:

$$F_{upsampled}^{64} = Upsample\_Bilinear\ F_{DenseNet}^{64} \quad (5)$$

At Level 3:

$$F_{upsampled}^{32} = Upsample\_Bilinear\ F_{DenseNet}^{16} \quad (6)$$

The bilinear upsampling operation is mathematically represented in represented in Eq. (7).

$$F'(x', y') = \sum_{i=1}^{1} \sum_{j=0}^{1} F\left(x^1 + i, y^i + j\right).w\left(i, x^i\right).w(j, y^i) \quad (7)$$

where F' is the upsampled feature map, F is the original feature map, and w is the bilinear interpolation kernel weight function. For computing weight functions, we employed Rule 1 and Rule 2 mentioned below:

Rule 1:

For $w(i,x')$:

- When i=0, $w(0,x')$ represents the weight assigned to the pixel to the left of the new pixel.
- When i=1, $w(1,x')$ represents the weight assigned to the pixel to the right of the new pixel.
- The weights are determined based on the distance between the new pixel's x-coordinate x' and the x-coordinates of the surrounding pixels.

Rule 2:

For $w(j,y')$:

- Similarly, $w(j,y')$ represents the weight assigned to the pixels above and below the new pixel, with j=0 for the pixel above and j=1 for the pixel below.
- The weights are determined based on the distance between the new pixel's y-coordinate y' and the y-coordinates of the surrounding pixels.

The weights are determined by using bi-linear interpolation where weights are assigned to the adjoining pixels based on the distances between the new pixel and the surrounding pixels. These weights therefore make sure that pixels nearby possess a dominant impact on the newly formed pixel, as opposed to pixels of lower influence that are more distanced from the new pixel. The final normalized weight matrix for level 2 and level 3 is given below. The detailed working of FPN is shown in Fig. 6.

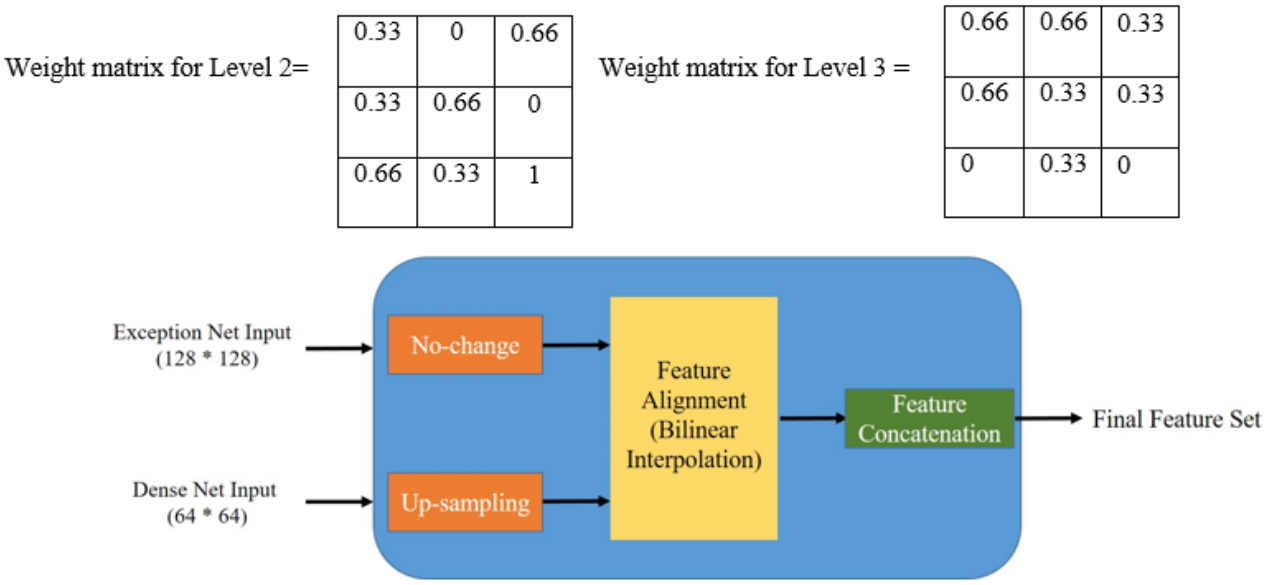

Fig. 6. Visualization of Internal Mechanism of Feature Concatenation within FPN.

3.3 Classification

The extracted features are further used in the dimensionality reduction process. The UMAP is a high dimensionality reduction algorithm that is executed in a nonlinear manner [26]. It uses various manifold design and projection techniques to get an in-depth understanding of the extracted features. The distances function gets the first step, where the distances among points in a high-dimensional space are calculated and then projected to a constricted space. Subsequently, the distance of the points in that vast, low-dimensional space is estimated. Afterward, we opted for stochastic gradient descent to make the said distances as close to each other as possible. One of the remarkable traits of UMAP output is the equidistance of local and global structures: UMAP preserves more of the global structure in the final projections.

The UMAP algorithm consists of two components—features that capture the inherent similarities of data and the dimensionality reduction step for the embedded manifold.

STEP 1: For a given feature set X= ($X1$, $X2$,$X3\cdots Xn$), dimension reduction is initialized to compute a low-dimensional dataset Y = ($Y1$,$Y2$,$Y3\cdots Yn$) $\sim$ N (0,10−4× I_n). Then, p_{ij} and initial q_{ij} are calculated.

The conditional probability of *i* for *j* is computed by

$$P_{i|j} = e^{\frac{d(x_i,x_j)}{\sigma_i}} \tag{8}$$

The symmetric formula of the similarity matrix P for X is defined as:

$$P_{ij} = P_{i|j} + P_{j|i} - P_{i|j} * P_{j|i} \tag{9}$$

The similarity matrix Q of Y is given by

$$Q_i = \left[1 + a(y_i - y_j)^{2b}\right]^{-1} \quad (10)$$

where $a \approx 1.73$ and $b \approx 0.87$ for default UMAP hyperparameters.

P_{ij} computes the similarity X_i and $X_{j,}$ and Q_{ij} measures the similarity Y_i and Y_j.

STEP 2: The cost loss is calculated using binary cross entropy (CE):

$$\text{CE}\ (p,q) = \sum_i \sum_j [P_{ij}.\, log\, \frac{P_{ij}}{Q\text{ij}} + \left(1 - P_{ij}\right).\log(\frac{(1-P_{ij})}{1-\, Qij}) \quad (11)$$

STEP 3: The parameters are optimized, and the number of iterations t, learning speed v, and momentum a are set. The

target result is a low-dimensional data representation Y = Y_1, Y_2,$Y_3 \cdots Y_n$.

STEP 4: The optimization phase is executed using the following equations:

$Y^t = Y^{t-1} + v.\frac{\delta C}{\delta Y} + a.(Y^{t-1} + Y^{t-2})$, where $\frac{\delta C}{\delta Y}$ is the gradient vector of the loss function for Y; $\frac{\delta C}{\delta Y} = (\frac{\delta C}{\delta Y_i})_{1*n}$

Finally, reduced features seta is normalized with the help of batch normalization, and this layer normalizes every layer's input, making training stable. It involves learning two parameters per feature per layer: the mean and the standard deviation. Following that, the data are stabilized into the form of fully connected layers. Statistical settings comprise the number of neurons in each layer('units') and the kind of activation function to use. Conversely, the Rectified Linear Unit activation function is defined as ReLU (x) = max (0, x), and facilitates nonlinearity. Additionally, the softmax layer uses the softmax function for class identification, and results are reported.

4. Results & Discussion

The proposed model was trained using a PC equipped with Intel Core I5 Processor and 16 GB RAM. It was implemented on Python 2.9, using Keras library and TensorFlow. In this section, the performance of the proposed model is validated on both datasets mentioned in section 2, and performance was compared with three benchmark deep neural network architectures: (1) VGG Net, (2) Inception Net, and (3) ResNet. Initially, a five-fold cross-validation technique was employed to split the original dataset into five groups or folds and then train the model iteratively using four folds to feed the training data, with the remaining fold being used for testing. This procedure is carried out five times, after which each one is used as the test set exactly once during this procedure. During the training process, both models (DenseNet, ExceptionNet) summary for dataset 1 is provided in Tables 1 and 2. Similarly, the details of both models for dataset 2 are given in Tables 3 and 4. In both cases, a

grid search approach was applied for optimal parameter tuning and to minimize the number of zeros below a certain threshold (<= 20%). In addition, an early-stopping strategy was applied to avoid overfitting issues during model training. The experiment was repeated ten times, and the mean arithmetic value was calculated to present the correct result. Different tests are to be performed before the final model evaluation. This is done to ensure that the choice of hyper-parameter values is proper. According to Tables 5 and 6, the best optimizer is Adagrad, which sets the learning rate to 0.003 (Brown Column-Table 5). The number of epochs used in the experiment was 150, and the batch size was 16.

Table 1: Layer-Wise Details of DenseNet Architecture for Dataset 1.

Layer (type)	Output Shape	Param #
Input_Layer (InputLayer)	[(None, 224, 224, 3)]	0
conv2d_1 (Conv2D)	(None, 112, 112, 64)	9472
batch_normalization_1	Batch (None, 112, 112, 64)	256
activation_1 (Activation)	(None, 112, 112, 64)	0
Dense_block_1 (DenseBlock)	(None, 28, 28, 256)	526336
transition_layer_1	Transition (None, 14, 14, 128)	33280
Dense_block_2 (DenseBlock)	(None, 7, 7, 512)	1078784
transition_layer_2	Transition (None, 3, 3, 256)	131328
global_average_pooling2d	(Gl (None, 256)	0
Dense_1 (Dense)	(None, 10)	2570
Total params: 7,037,578 Trainable params: 6,963,978 Non-trainable params: 73,600		

Table 2: Layer-Wise Details of ExceptionNet Architecture for Dataset 1.

Layer (type)	Output Shape	Param #
Input_Layer (InputLayer)	[(None, 299, 299, 3)]	0
conv2d_71 (Conv2D)	(None, 10, 10, 2048)	1050624
activation_33 (Activation)	(None, 10, 10, 2048)	0
global_average_pooling2d_1	(None, 2048)	0
Dense_2 (Dense)	(None, 10)	20490
Total params: 21,914,754 Trainable params: 21,859,210 Non-trainable params: 55,544		

Table 3: Layer-Wise Details of DenseNet Architecture for Dataset 2.

Layer (type)	Output Shape	Param #
Input_Layer (InputLayer)	[(None, 224, 224, 3)]	0
conv2d_1 (Conv2D)	(None, 112, 112, 64)	9472
batch_normalization_1	Batch (None, 112, 112, 64)	256
activation_1 (Activation)	(None, 112, 112, 64)	0
Dense_block_1 (DenseBlock)	(None, 28, 28, 256)	526336
transition_layer_1	Transition (None, 14, 14, 128)	33280
Dense_block_2 (DenseBlock)	(None, 7, 7, 512)	1078784
transition_layer_2	Transition (None, 3, 3, 256)	131328
global_average_pooling2d	(Gl (None, 256)	0
Dense_1 (Dense)	(None, 10)	2570
Total params: 7,037,578 Trainable params: 6,963,978 Non-trainable params: 73,600		

Table 4: Layer-Wise Details of ExceptionNet Architecture for Dataset 2.

Layer (type)	Output Shape	Param #
Input_Layer (InputLayer)	[(None, 299, 299, 3)]	0
conv2d_71 (Conv2D)	(None, 10, 10, 2048)	1050624
activation_33 (Activation)	(None, 10, 10, 2048)	0
global_average_pooling2d_1	(None, 2048)	0
Dense_2 (Dense)	(None, 10)	20490
Total params: 21,914,754 Trainable params: 21,859,210 Non-trainable params: 55,544		

Table 5. Performance comparison between different optimizers and learning rates with the proposed architecture.

Optimizer	Learning Rate					
	0.1	0.01	0.001	0.002	0.003	0.004
Adam	59.06%	70.69%	71.30%	66.30%	91.80%	76.30%
SGD	71.80%	91.09%	76.98%	71.52%	88.40%	82.34%
Adadelta	59.40%	83.35%	81.54%	76.24%	90.88%	86.19%
Adagrad	78.20%	88.83%	89.66%	84.33%	97.30%	90.33%

The proposed architecture was implemented to discriminate multiple true labels of both datasets. The testing accuracies after each 20 epochs are reported on both datasets and given in Table 6. During model testing, the maximum classification accuracy was reported 98.24% for Dataset 1, and 94.66% for Dataset 2. In Fig. 7. (A), the training accuracy curve (represented by a blue curve) throughout the epochs. It reached a maximum accuracy of 98.24% at epoch no. 113 and then it maintained this threshold. Similarly, the proposed approach realized a maximum of 94.66% classification at epoch no. 118 (Fig. 7. (B)). In the early phases, the learning ability of our model is very slow and consistent which shows the noisy behavior of both datasets. However, the model gains a significant improvement after epoch no. 80 and converged to the reported accuracy levels.

Table 6. Performance comparison between different optimizers and learning rates with the proposed architecture.

Dataset	Epochs				
	20	40	60	80	91-100
Dataset 1	21.04%	41.33%	69.84%	84.52%	98.24%
Dataset 2	18.54%	36.20%	61.53%	79.62%	94.66%

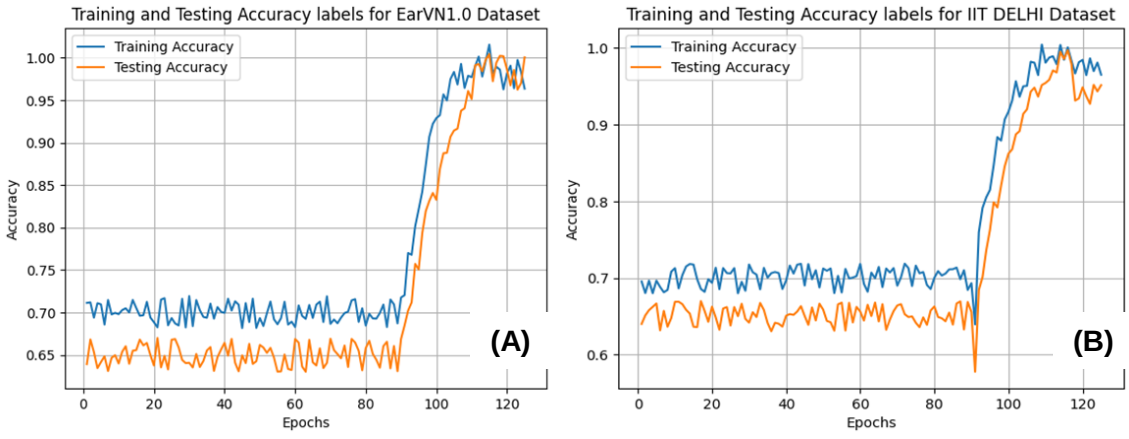

Fig.7. The classification accuracy vs epochs graph for both ENV1.0 dataset (A) and IIT DELHI dataset (B).

The ROC curve for both datasets is shown in Fig. 8, respectively. In the image, the Area Under the Curve (AUC) values for most classes range from 0.74 to 1.00, which indicates a good performance for those classes. However, there are very few classes having lower AUC, such as class 110 (AUC = 0.77) and class 130 (AUC = 0.74). This suggests that the model may have difficulty classifying instances of these particular classes. The experimental analysis concludes

that avg. AUC for 165 classes is 0.8391 which reflects a superior discriminative ability of the proposed architecture. Similarly, our approach realized a competitive AUC score for all 121 subjects. In this case, the avg. AUC score was 0.91 which is higher than state-of-the-art methods.

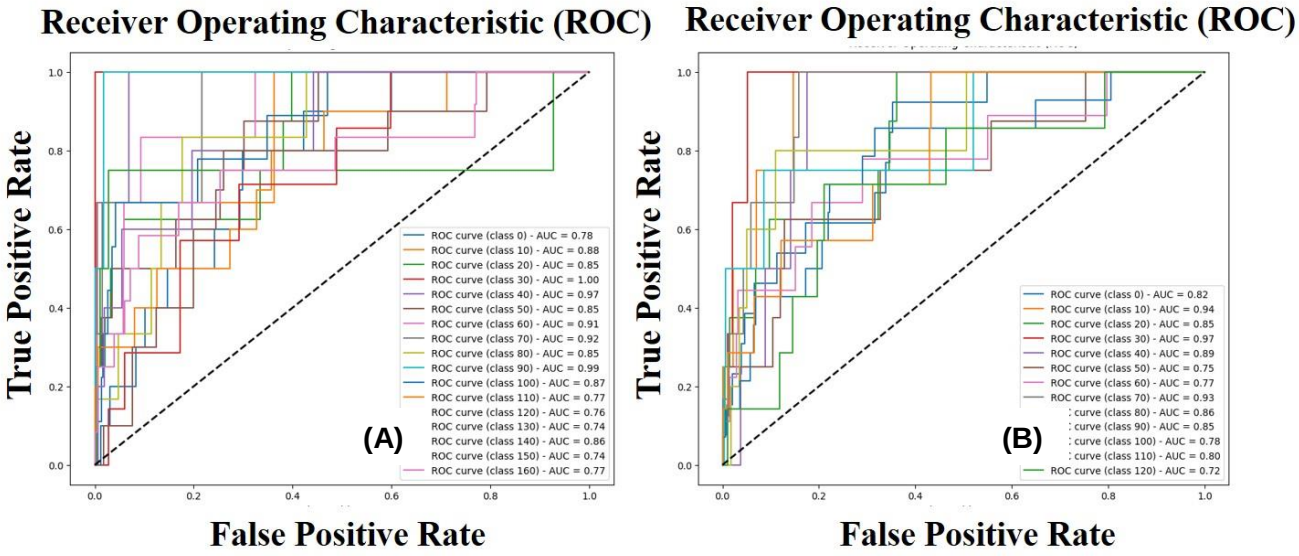

Fig. 8. Receiver operating characteristics for various classes of ENV1.0 dataset (A) and IIT DELHI dataset (B).

The weighted average classification accuracies for both datasets are given in Tables 7 and 8 and compared in terms of epochs (per 10 epochs) with three state-of-the-art architectures: (1) InceptionV3 Net, (2) ResNet-50, and (3) VGG-Net. It can be observed that the proposed classification model realized better accuracy scores when compared with the above-mentioned Deep Neural Networks (DNNS). Moreover, the key observation is the slower speed of the proposed classification model in the early epochs when compared with all baseline models. In the ENV1.0 dataset, our model achieved significant speedup after 60 epochs while it achieved maximum classification accuracy scores after 80 epochs when compared with all three neural networks. In this context, our model improved the accuracy levels by 11.21% (InceptionNet), 17.39% (ResNet-50), and 22.29% (VGG-Net) on the ENV1.0 Dataset. In the second case, our model enhanced the global weighted accuracy levels by 12% (InceptionNet), 2.73% (ResNet-50), and 29% (VGG-Net). Moreover, the VGG-Net performed worst among all the classification models on both datasets.

Table 7. Epoch-wise classification accuracies between the proposed model and baseline architecture models on ENV1.0 Dataset.

Epoch-count	Proposed Model	InceptionV3 Net	ResNet-50	VGG-Net
After 10 epoch	09.08%	11.05%	23.56%	09.11%
After 20 epoch	21.04%	29.39%	29.33%	19.33%
After 30 epoch	36.38%	41.33%	41.18%	33.80%
After 40 epoch	41.33%	53.48%	51.48%	39.56%
After 50 epoch	59.88%	61.33%	63.78%	51.39%
After 60 epoch	69.84%	67.80%	69.40%	59.20%

After 70 epoch	91.53%	74.40%	76.59%	71.12%
After 80 epoch	94.52%	81.87%	81.02%	77.92%
After 90 epoch	98.24%	88.33%	83.68%	80.33%

Table 8. Epoch-wise classification accuracies between the proposed model and baseline architecture models on the IIT DELHI Dataset.

Epoch-count	Proposed Model	InceptionV3 Net	ResNet-50	VGG-Net
After 10 epoch	11.08%	18.49%	21.06%	17.50%
After 20 epoch	18.54%	29.13%	27.33%	31.04%
After 30 epoch	33.38%	33.39%	47.39%	36.40%
After 40 epoch	36.20%	39.43%	61.28%	48.12%
After 50 epoch	51.88%	51.92%	68.40%	54.66%
After 60 epoch	61.53%	68.02%	72.56%	62.46%
After 70 epoch	73.53%	79.33%	84.66%	68.44%
After 80 epoch	79.62%	80.02%	88.44%	70.94%
After 90 epoch	94.66%	84.50%	92.14%	73.38%

Along with classification accuracy, we compared the performance of our proposed approach in terms of precision, recall, score, and Area Under the Curve (AUC) with all three conventional models. For both datasets, the detailed performance of all these models is represented in Tables 9 and 10, respectively. Regarding dataset 1, the findings in Table 9 show that the proposed model performs better than all the baseline models (InceptionV3 Net, ResNet-50, and VGG-Net) in all metrics, including precision, recall, F1 Score, and Area Under the Curve (AUC). With a precision score of 0.9831, the suggested model excels at properly recognizing positive instances, indicating a low false positive rate. In comparison to other models, this makes it extremely reliable in categorizing DDoS threats. In comparison, InceptionV3 (0.9222), ResNet-50 (0.9018), and VGG-Net (0.8826) have poorer precision, indicating that they are more likely to misclassify innocuous events as threats.

In terms of recall, the suggested model receives a high score of 0.9741, suggesting a great ability to recognize nearly all true positive situations, guaranteeing that very few genuine attacks are missed. The remaining models, InceptionV3 (0.9188), ResNet-50 (0.8912), and VGG-Net (0.8719), perform poorly in the recall, meaning that they miss more actual attacks or positive events. This lower recall indicates that these models are less effective at minimizing false negatives than the suggested approach. The F1 Score, which balances precision and recall, gives the proposed model a score of 0.97531, emphasizing its superiority. This high number demonstrates its strong effectiveness in reducing both false positives and false negatives.

The competing models, with F1 scores of 0.91552 (InceptionV3), 0.8953 (ResNet-50), and 0.8752 (VGG-Net), demonstrate a lower capacity to manage the trade-off between precision and recall, validating the suggested model's overall advantage in handling classification tasks well.

Finally, the suggested model's AUC (0.9966) is near-perfect, demonstrating its excellent ability to discern between positive and negative classes. This means that the model can reliably distinguish between DDoS attacks and normal network traffic. In comparison, the AUC ratings for InceptionV3 (0.9322), ResNet-50 (0.9272), and VGG-Net (0.9028) are significantly lower, indicating that these models are less good at distinguishing between attack and normal cases. Overall, the results demonstrate the suggested model's ability to handle the complexities of modern network security threats, particularly DDoS detection and mitigation.

Table 9. Comparative performance measures for all the baseline models and the proposed methodology on the ENV1.0 Dataset.

Model	Precision	Recall	F1 Score	Area under the Curve (AUC)
Proposed Model	0.9831	0.9741	0.97531	0.9966
InceptionV3 Net [33]	0.9222	0.9188	0.91552	0.9322
ResNet-50 [34]	0.9018	0.8912	0.8953	0.9272
VGG-Net [35]	0.8826	0.8719	0.8752	0.9028

Table 10 refers to the performance of all classification models on the dataset 2. The performance levels of the four models (Proposed Model, InceptionV3 Net, ResNet-50, and VGG-Net) provide a thorough grasp of their usefulness in identifying DDoS attacks. Starting with precision, the proposed model shines out with a score of 0.9761, showing that it is very good at reducing false positives. This means that it usually always forecasts an attack correctly. In comparison, InceptionV3 and VGG-Net have reasonable precision scores of 0.8834 and 0.8826, respectively, although they are less precise than our suggested model. ResNet-50 trails behind with a precision of 0.7941, indicating a higher potential for false positives than the other models. Recall conveys a different narrative. InceptionV3 has a high recall score of 0.9714, indicating that it detects virtually all true positives or actual assaults, demonstrating its ability to minimize false negatives. However, the proposed model's recall of 0.9251, while somewhat lower, shows that it can identify the majority of attacks. ResNet-50, with a recall of 0.9234, and VGG-Net, with 0.8722, perform reasonably well, however, they miss more attacks than InceptionV3, indicating a lower sensitivity.

The F1 Score, which balances precision and recall, gives the proposed model a score of 0.9529, indicating its strong performance. This score reflects a decent balance of effectively identifying assaults and minimizing false positives. InceptionV3 follows with an F1 score of 0.9254, indicating that it can find a decent balance between precision and recall, albeit not as well as the suggested model. ResNet-50, with a lower F1 score of 0.8053, appears to struggle more in maintaining this balance, whereas VGG-Net's F1 score of 0.8768 indicates reasonable performance when compared to the leading models.

Table 10. Comparative performance measures for all the baseline models and the proposed methodology on the IIT DELHI Dataset.

Model	Precision	Recall	F1 Score	Area under the Curve (AUC)
Proposed Model	0.9761	0.9251	0.9529	0.9141
InceptionV3 Net [33]	0.8834	0.9714	0.9254	0.8368
ResNet-50 [34]	0.7941	0.9234	0.8053	0.7248
VGG-Net [35]	0.8826	0.8722	0.8768	0.8020

5. Conclusions and Future Scope

In this work, we designed a novel vision-based hybrid CNN classification framework by merging attention-enabled XceptionNet and DenseNet architectures. The proposed architecture realized a maximum of 98.24% and 94.66% accuracy scores on ENVN1.0 and IIT DELHI datasets, respectively. Further, it improved the performance of other baseline models, such as VGGNet, InceptionNet, and ResNet50, with a significant margin of 22.29%, 11.21%, and 17.39%, respectively, on the ENV1.0 Dataset. In addition, our hybrid architecture enhanced the accuracy levels by 12% (InceptionNet), 2.73% (ResNet-50), and 29% (VGG-Net) on the IIT DELHI dataset. Moreover, we enhanced the performance of existing state-of-the-art classification models in terms of high AUC scores on both datasets. The primary limitation of our proposed model is its high computational complexity (slower speed) and resource dependency. The complex structure uses various kernel functions at different levels and multiple Dense blocks to obtain maximum classification accuracy levels. Researchers may explore several optimization techniques, such as lightweight models based on model compression and low-rank approximation methods [35], transfer learning [36], parametric reduction [37], and Gabor tensors [38] approaches to resolve this issue. In addition, various hardware acceleration modules such as Graphics Processing Units (GPUs) [39] or Tensor Processing Units (TPUs) [40] and distributed computing [41] can also be used to deal with this limitation.

References

[1] Ž. Emeršič, V. Štruc, and P. Peer, "Ear recognition: More than a survey," *Neurocomputing*, vol. 255, pp. 26–39, 2017.

[2] H. Chen and B. Bhanu, "Human ear recognition in 3D," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 4, pp. 718–737, 2007.

[3] I. I. Ganapathi, S. S. Ali, S. Prakash, N. S. Vu, and N. Werghi, "A survey of 3D ear recognition techniques," *ACM Comput. Surv.*, vol. 55, no. 10, pp. 1–36, 2023.

[4] A. S. Anwar, K. K. A. Ghany, and H. Elmahdy, "Human ear recognition using geometrical features extraction," *Procedia Comput. Sci.*, vol. 65, pp. 529–537, 2015.

[5] A. Korichi, S. Slatnia, and O. Aiadi, "TR-ICANet: A fast unsupervised deep-learning-based scheme for unconstrained ear recognition," *Arabian J. Sci. Eng.*, vol. 47, no. 8, pp. 9887–9898, 2022.

[6] M. Sharkas, "Ear recognition with ensemble classifiers: A deep learning approach," *Multimedia Tools Appl.*, vol. 81, no. 30, pp. 43919–43945, 2022.

[7] H. Alshazly, C. Linse, E. Barth, S. A. Idris, and T. Martinetz, "Towards explainable ear recognition systems using deep residual networks," *IEEE Access*, vol. 9, pp. 122254–122273, 2021.

[8] L. Tian and Z. Mu, "Ear recognition based on deep convolutional network," in *Proc. 9th Int. Congr. Image Signal Process., Biomed. Eng. Informatics (CISP-BMEI)*, Oct. 2016, pp. 437–441.

[9] D. P. Chowdhury, S. Bakshi, G. Guo, and P. K. Sa, "On applicability of tunable filter bank based feature for ear biometrics: A study from constrained to unconstrained," *J. Med. Syst.*, vol. 42, 2018, Art. no. 1.

[10] M. Z. Alom et al., "The history began from AlexNet: A comprehensive survey on deep learning approaches," *arXiv preprint arXiv:1803.01164*, 2018.

[11] A. Vedaldi and A. Zisserman, "VGG convolutional neural networks practical," Dept. Eng. Sci., Univ. Oxford, Tech. Rep., 2016.

[12] B. Koonce, *SqueezeNet*, in *Convolutional Neural Networks with Swift for TensorFlow: Image Recognition and Dataset Categorization*, 2021, pp. 73–85.

[13] K. Maharana, S. Mondal, and B. Nemade, "A review: Data pre-processing and data augmentation techniques," *Global Transitions Proc.*, vol. 3, no. 1, pp. 91–99, 2022.

[14] R. A. Priyadharshini, S. Arivazhagan, and M. Arun, "A deep learning approach for person identification using ear biometrics," *Appl. Intell.*, vol. 51, no. 4, pp. 2161–2172, 2021.

[15] A. Kumar and C. Wu, "Automated human identification using ear imaging," *Pattern Recognit.*, vol. 45, no. 3, pp. 956–968, 2012.

[16] E. Gonzalez, L. Alvarez, and L. Mazorra, "AMI ear database," [Online]. Available: https://webctim.ulpgc.es/research_works/ami_ear_database/, Accessed Apr. 14, 2024.

[17] I. I. Ganapathi, S. Prakash, I. R. Dave, and S. Bakshi, "Unconstrained ear detection using ensemble-based convolutional neural network model," *Concurrency Comput. Pract. Exp.*, vol. 32, no. 1, 2020, Art. no. e5197.

[18] L. Yuan and F. Zhang, "Ear detection based on improved AdaBoost algorithm," in *Proc. Int. Conf. Mach. Learn. Cybern.*, Jul. 2009, vol. 4, pp. 2414–2417.

[19] K. V. Joshi and N. C. Chauhan, "Edge detection and template matching approaches for human ear detection," in *Proc. Int. Conf. Intell. Syst. Data Process. (ICISD)*, 2011, pp. 50–55.

[20] J. C. Yoo and T. H. Han, "Fast normalized cross-correlation," *Circuits Syst. Signal Process.*, vol. 28, pp. 819–843, 2009.

[21] S. Prakash and P. Gupta, "An efficient ear localization technique," *Image Vis. Comput.*, vol. 30, no. 1, pp. 38–50, 2012.

[22] N. K. A. Wahab, E. E. Hemayed, and M. B. Fayek, "HEARD: An automatic human ear detection technique," in *Proc. Int. Conf. Eng. Technol. (ICET)*, Oct. 2012, pp. 1–7.

[23] P. Chidananda, P. Srinivas, K. Manikantan, and S. Ramachandran, "Entropy-cum-Hough-transform-based ear detection using ellipsoid particle swarm optimization," *Mach. Vis. Appl.*, vol. 26, pp. 185–203, 2015.

[24] H. Peng and C. Ding, "Minimum redundancy and maximum relevance feature selection and recent advances in cancer classification," in *Feature Selection for Data Mining*, 2005, p. 52.

[25] L. Wang and H. Li, "HMCNet: Hybrid efficient remote sensing images change detection network based on cross-axis attention MLP and CNN," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, pp. 1–14, 2022.

[26] T. Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2117–2125.

[27] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform manifold approximation and projection for dimension reduction," *arXiv preprint arXiv:1802.03426*, 2018.

[28] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1251–1258.

[29] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer, "DenseNet: Implementing efficient convnet descriptor pyramids," *arXiv preprint arXiv:1404.1869*, 2014.

[30] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2818–2826.

[31] B. Koonce, *ResNet 50*, in *Convolutional Neural Networks with Swift for TensorFlow: Image Recognition and Dataset Categorization*, 2021, pp. 63–72.

[32] M. Lebrun, "An analysis and implementation of the BM3D image denoising method," *Image Process. OnLine*, vol. 2, pp. 175–213, 2012.

[33] X. Xia, C. Xu, and B. Nan, "Inception-v3 for flower classification," in *Proc. 2nd Int. Conf. Image Vis. Comput. (ICIVC)*, Jun. 2017, pp. 783–787.

[34] B. Koonce, *ResNet 50*, in *Convolutional Neural Networks with Swift for TensorFlow: Image Recognition and Dataset Categorization*, 2021, pp. 63–72.

[35] N. K. Kumar and J. Schneider, "Literature survey on low rank approximation of matrices," *Linear Multilinear Algebra*, vol. 65, no. 11, [incomplete reference].

[36] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, IGI Global, 2010, pp. 242–264.

[37] A. Tiwari and A. Chaturvedi, "A hybrid feature selection approach based on information theory and dynamic butterfly optimization algorithm for data classification," *Expert Syst. Appl.*, vol. 196, p. 116621, 2022.

[38] A. Tiwari, "Wilson's disease classification using higher-order Gabor tensors and various classifiers on a small and imbalanced brain MRI dataset," *Multimedia Tools Appl.*, vol. 82, no. 23, pp. 35121–35147, 2023.

[39] A. U. Rahman, G. Tikhonov, J. Oksanen, T. Rossi, and O. Ovaskainen, "Accelerating joint species distribution modelling with Hmsc-HPC by GPU porting," *PLoS Comput. Biol.*, vol. 20, no. 9, p. e1011914, 2024.

[40] F. Belletti, D. King, K. Yang, R. Nelet, Y. Shafi, Y. F. Shen, and J. Anderson, "Tensor processing units for financial Monte Carlo," in *Proc. SIAM Conf. Parallel Process. Sci. Comput.*, 2020, pp. 12–23.

[41] J. Waldo, G. Wyant, A. Wollrath, and S. Kendall, "A note on distributed computing," in *Int. Workshop Mobile Object Syst.*, Jul. 1996, pp. 49–64.