Performance Analysis of NASNet on Unconstrained Ear Recognition

K. Radhika, K. Devika, T. Aswathi, P. Sreevidya, V. Sowmya and K. P. Soman

Abstract *Recent times are witnessing greater influence of Artificial Intelligence (AI) on identification of subjects based on biometrics. Traditional biometric recognition algorithms, which were constrained by their data acquisition methods, are now giving way to data collected in the unconstrained manner. Practically, the data can be exposed to factors like varying environmental conditions, image quality, pose, image clutter and background changes. Our research is focused on the biometric recognition, through identification of the subject from the ear. The images for the same are collected in an unconstrained manner. The advancements in deep neural network can be sighted as the main reason for such a quantum leap. The primary challenge of the present work is the selection of appropriate deep learning architecture for unconstrained ear recognition. Therefore the performance analysis of various pretrained networks such as VGGNet, Inception Net, ResNet, Mobile Net and NASNet is attempted here. The third challenge we addressed is to optimize the computational resources by reducing the number of learnable parameters while reducing the number of operations*. Optimization of selected cells as in NASNet architecture is a paradigm shift in this regard.

Keywords Deep learning · Convolutional neural network · Unconstrained ear recognition · NASNet

1 Introduction

Artificial Intelligence (AI) wraps the cycle, which contains Machine Learning (ML) and Deep Learning (DL) algorithms. The trend towards DL has increased day by day due to the large amount of data and high computational speed. The main advantage of DL over ML is elimination of the hard core feature extraction techniques. DL,

K. Radhika (✉) · K. Devika · T. Aswathi · P. Sreevidya · V. Sowmya · K. P. Soman
Centre for Computational Engineering and Networking Amrita School of Engineering, Coimbatore Amrita Vishwa Vidyapeetham, Coimbatore, India
e-mail: k_radhika@cb.students.amrita.edu

M. Rout et al. (eds.), *Nature Inspired Computing for Data Science*,
Studies in Computational Intelligence SCI 871,
https://doi.org/10.1007/978-3-030-33820-6_3

the subset of ML can learn from both unstructured and unlabeled data. According to Bengio [1], DL algorithms seek to exploit the unknown structure in the input distribution. This is to identify good representation often at multiple levels with higher-level learned features defined in terms of lower-level features. DL has brought about a explosion of data in all forms and from every region of the world by redefining the technological environments [2]. It has miscellaneous applications in the fields of signal processing, speech processing and synthesis, image processing, communication, optimization etc., as it discovers intricate structure in large data sets. For the past few decades, DL has shown a prominent role in computer vision problems such as image classification, object detection, object segmentation, image reconstruction etc. This chapter mainly focuses on image classification problems specific to biometric applications. These biometric applications are widely used for biomedical areas and in security purposes. One of the major issues faced in the modern world is the insecurity of personal information or any other types of data. As the usage of passwords and tokens becomes highly insecure or forgotten, biometric recognition became popular [3]. Each person has their own physical and behavioural characteristics therefore, biometric applications make use of these characteristics for person identification. Some of the biometrics such as ear, face, speech and signature can be collected unknowingly while others like iris, fingerprint, palm-print are collected with person's knowledge. Among all, finger print authentication is one of the famous and oldest biometric technique but, new studies claims that ear authentication is able to give comparatively good results to the situations in which, other authentication fails [4]. The factors like low cost, complexity, long distance recognition, static size etc., gives additional advantage to ear biometrics. Moreover, in distinguishing identical twins the difficulties with face recognition biometrics can be solved using ear biometrics [5]. However, recognition of biometrics in unconstrained environment is more challenging when compared to constrained environment.

Inorder to recognize ears in unconstrained environment, a challenge was conducted for the first time in 2017 by IEEE/IAPR, International Joint Conference on Biometrics (IJCB) [6]. Second series of group benchmarking effect (UERC-2019) was done for ear recognition by IAPR, International Conference on Biometrics 2019. There were two dataset for UERC 2019 challenge: one publically available dataset and one sequestered dataset (available only for the participants in the challenge) in which, the public dataset contains 11,000 images of 3,690 classes. Major contribution of the public dataset was carried out by Extended Annotated Web Ears (AWEx) and the remaining data were taken from UERC-2017 challenge. Sequestered dataset contains 500 images, belongs to 50 subjects which was used only for testing purpose in the challenge [7]. The trend towards DL can also be seen in biometric applications. It is observed that various Convolutional Neural Network (CNN) architectures could perform better as compared to other hand crafted feature models in UERC challenges [8].

1.1 Study on Foregoing CNN Architectures

From the year 1998, neural networks has shown an exponential growth due to the impact of two major developments in advancement of technology. During these days, data generated from various resources like social networks, cameras, mobile phones etc., has been escalated drastically. This growth demanded immense computational power inorder to process the data within a short span of time. This motivated trend towards data analysis task in various applications. DL algorithms necessitate massive amount of data to work efficiently irrespective of domain. Since DL algorithms can learn by itself from the data provided, the feature extraction task became effortless. DL has it's own supervised learning approaches such as Artificial Neural Network (ANN), CNN, Recurrent Neural Network (RNN) etc. For pattern and image recognition problems CNN has mould a prominent role [9]. Inorder to extract features automatically, CNN filters are convolved over original data. Convolutional layers, pooling layers and fully connected layers are the basic building blocks of CNN architecture. With the increase or decrease of these hidden layers, different CNN models were evolved. Evolution of most commonly used CNN architectures are shown in Table 1 [8].

In 1990s LeNet, a seven layer CNN architecture was introduced, but due to the lack of computation and memory facilities, the architecture was away from practical use till 2010. The concept of back propagation was primarily put forth by LeNet [10]. AlexNet, a much wider and deeper network than LeNet was proposed for visual object recognition tasks. Upon all classical machine learning algorithms, AlexNet attained better accuracy for recognition and is also capable of solving challenging ImageNet problems [11]. An elongation of AlexNet–ZFNet, with kernel size of 7×7, aids

Table 1 Evolution of CNN architectures

Sl.no	CNN architecture	Year	Developed by
1	LeNet	1998	Yann LeCun
2	AlexNet	2012	Alex Krizhevsky
3	ZFNet	2013	Matthew Zeiler and Rob Fergus
4	Network in network	2013	MinLin
5	VGGNet	2014	Simonyan and Zisserman
6	GoogLeNet or InceptionNet	2014	Google
7	ResNet	2015	Kaiming He
8	FractalNet	2016	G Larsson
9	DenseNet	2017	Gao Huang
10	MobileNet	2017	Google
11	XceptioNet	2017	Francois Chollet
12	NASNet	2017	Google
13	CapsuleNet	2017	Hinton

to break the raised count of weights, there upon reduced number of parameters and improved accuracies [12]. Network in Network came up with a new concept of multi layer perception convolution technique with 1×1 filters for the first time. Along with this, the network offers a technique called Global Average Pooling (GAP), which conveniently reduces the number of learnable parameters [13].

As researcher's passion towards visual recognition tasks using CNN has emerged, a new challenge called ImageNet Large Scale Visual Recognition Challenge (ILSVRC) was launched. There after, the outperformers of this challenge give rise to evolution of new models. A filter of size 3×3 was used in all the versions of VGGNet. Different versions of VGG (VGG11, VGG16 and VGG19) was developed with layers 11, 16 and 19 respectively. Each version of VGG has 8, 13 and 16 number of convolutional layers [14]. The salient feature (Inception layers) of InceptionNet or GoogLeNet made it winner of ILSVRC 2014. The advantage of GoogLeNet over other networks was the reduced number of parameters (7 M) with minor computations (1.53 G) [15]. All the evolved models concentrated on increasing the depth of the network inorder to improve accuracy. This came up with a problem of vanishing gradient. While back propagating, the value of gradient becomes too small, which does not make any sense and results in poor learning. ResNet came up with a solution for the above mentioned problem, which could increase the performance of the model. There was an exponential growth in the number of layers in various versions of ResNet [16]. An advanced version of ResNet named FractalNet arised with a new concept of drop path, which is capable of developing large models [17]. By connecting the predecessor layer output to the successive layer, a dense network (DenseNet) has formed for feature reuse [18]. For reducing the number of parameters, a new approach called separable-convolution, which includes depth-wise and point-wise separable-convolution was proposed in MobileNet. Thus, DL conquered the world of mobile and embedded applications too [19]. Standard inception modules were replaced with depth-wise separable-convolutions, which bring out another model called XceptionNet with minimum weight serialization [20]. Two promising concepts named AutoML and Neural Architectural Search (NAS) induce a new optimized architecture called NASNet. The concept of Reinforcement Learning (RL) and Evolutionary Algorithms (EAs) facilitate for the optimization task [21]. A recent model- CapsuleNet [22] on face detection has evolved for face recognition by relatively considering all the distinct facial features [23].

On the basis of the knowledge obtained from the above mentioned architectures, an elaborated study was performed for five major architectures for unconstrained ear recognition. The performance was compared by giving attention to the special features highlighted for each architecture with a special focus on the NASNEt architecture. NASNet is an optimized selection of layers which give us most satisfying results on unconstrained ear recognition. The chapter is organized as follows: Sect. 2 discusses about five DL architectures that we are discussing here. It includes VGG-16, Inception Net, ResNet, Mobile Net and NASNet. The architectural specialities are highlighted here. A detailed explanation about UERC-2019 dataset is presented in Sect. 3. Section 4 presents the design of various experiments carried

out and Sect. 5 presents analysis of the results with necessary image and plots. The chapter concludes with the inference from the comparative results and its future directions.

2 CNN Architectures

Architectural details of VGG-16, MobileNet, InceptionNet, ResNet and NASNet are discussed in this section. The deep neural architectures considered here except the NASNet are developed by human experts. NASNet on the other hand is formed automatically by optimizing the individual cells in it.

2.1 Parameter Calculation for CNN Models

Convolution, max-pooling and separable-convolution are the common operations found in CNN architectures. There is a unique way of calculating number of parameters and output size for each operation. Input size, filter size, zero padding and stride are the parameters used for calculating output size of the next block. General equation for the calculation of output size is given in Eq. (1).

$$Outputsize = \frac{Inputsize - Filtersize + 2 \times zeropadding}{Stride} + 1 \quad (1)$$

As shown in Fig. 1, an RGB (M = 3) image has taken of size 55 × 55. It is convolved with 5 filters of size 5 × 5 by stride of 2. So, using equation [1], the output size of next block become 26 × 26. Max-pooling reduces the input block size by 2, thus we get 13 × 13 image output. While moving to separable-convolution with zero padding and stride 2 the output size become 7 × 7.

The general formula for calculating number of learnable parameters by convolution and separable-convolution is given in Eqs. (2) and (3) respectively.

$$No.of\,Learnableparameters(convolution) = N \times M \times D_k^2 \quad (2)$$

$$No.of\,Learnableparameters(separable - convolution) = M(D_k^2 + N) \quad (3)$$

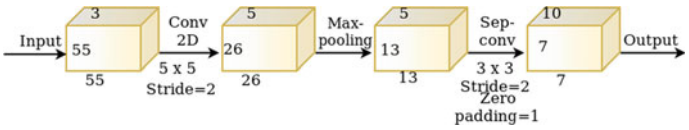


Fig. 1 Sample representation of CNN architecture

Table 2 Sample calculation of learnable parametres and output size

Operations	No. of learnable parameters	Output size
Input	0	55 × 55
Conv2D	3 × 5 × 5 × 5 = 375	26 × 26
Max-pooling	0	13 × 13
Sep-conv	(5 × 3 × 3) + (5 × 10) = 95	7 × 7

For normal convolution method, parameters are calculated by multiplying number of input channels (M) with number of filters (N) and filter size ($D_k{}^2$). From Table 2, we could observe separable-convolution reduces the number of learnable parameters. Thus we could see the phenomenon of using separable-convolution in the evolution of CNN models.

2.2 VGG16

Operations in VGG are much similar to that of AlexNet. Since VGGNet has a publically available weight configuration, it is used as baseline for image feature extraction problems. The VGGNet shown in Fig. 2, consists of stacked layers of convolution and pooling layers [14]. It can be observed that the depth of the network is 16 layers without the problem of vanishing gradients or exploding gradients. The number of learnable parameters in VGG16 is found to be 40,96,000 of 1000 class problems on ImageNet applications with 224 × 224 sized RGB images. There are 13 convolutional layers with 5 max-pooling layers and final 3 dense layers with two layers of size 4096. The size of the convolution output matrix reduces to 7 × 7 just before connecting as a dense layer. There is a fixed stride of 1, used throughout the network in convolution layers and a stride of 2, used in max-pooling layers. All the hidden layers of VGG16 uses non-linear ReLU activation functions and final layer is a softmax layer. The problem with this simple stacked layer architecture is that, it was extremely slow to train and consumes huge memory space. Varying the depth from 16–19 layers, were able to come up with better accuracy.

2.3 InceptionV3

Inception architecture of GoogLeNet [15] introduced to perform well with limited memory and computational budget. InceptionV3 is a CNN, trained on more than a million images from the ImageNet database with 1000 class [24]. The network is 48 layers deep and it has 23,885,392 learnable parameters on 1000 class problem. In order to maximize the information flow into the network, Inception module has

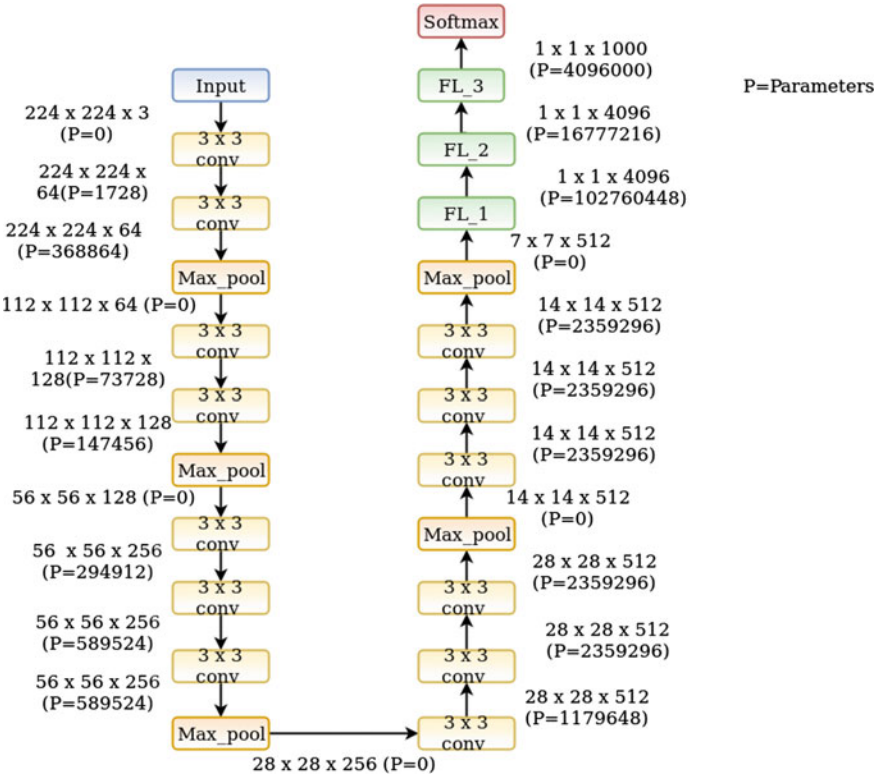


Fig. 2 VGG16 architecture with parameters

made layers not only in depth wise, but also in width wise, by carefully constructing networks that balance depth and width. The inception module shown in Fig. 3, where the module integrates a series of filters, includes a 3×3 filter, 5×5 filter and followed by 1×1 filter with a final concatenation operation. The bottleneck of 1×1 operation reduces computational overhead considerably. This is because, before 3×3 and 5×5 convolution, 1×1 convolution was calculated. The InceptionV3 module has adapted a asymmetric factorization for convolution. It decomposes a 3×3 filter into two, matrix size 1×3 and 3×1 filters respectively, thus reducing the number of learnable parameters from 9 to 6. The module in Fig. 3, takes up an input of $28 \times 28 \times 256$ and undergoes filtering operations with kernels of size 3×3, 5×5 and 1×1 along with max-pooling widthwise and finally a concatenation operation to get an output feature set of $28 \times 28 \times 672$. The computational load is reduced by 28% by factorizing the convolutional kernel of size 5×5 into two 3×3 consecutive kernels.

Apart from the convolutional layer, the network has max-pooling layer and global average pooling layer. Dropouts are successfully implemented for performance optimization. The introduction of dropouts effectively handled the over fitting problems

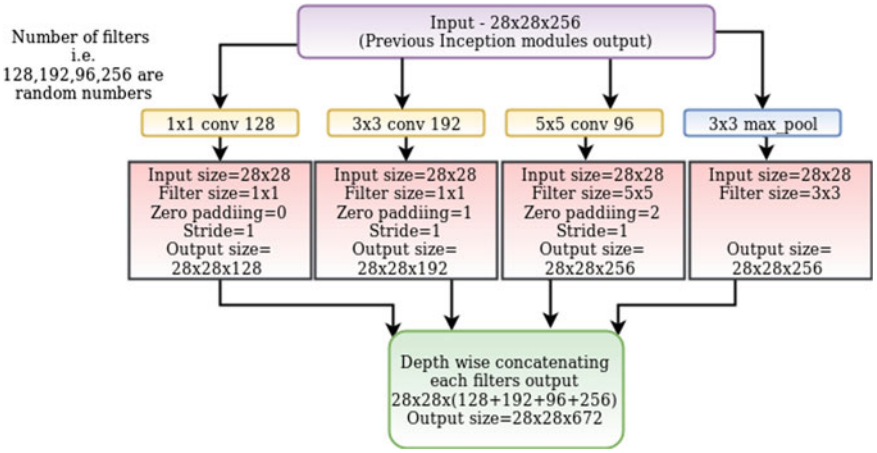


Fig. 3 Naive inception module

in the model, by actively switching connections with activation layer. There is also an auxiliary classifier for regularization. The whole Inception architecture has 8 inception modules and two auxiliary classification layer as shown in Fig. 4.

2.4 ResNet

The key point of ResNet is residual learning and identity mapping by shortcuts using a simple logic depicted in Fig. 5. The skip connection forwards the block inputs as an identity mapping. In normal convolutional networks, going deeper will affect the performance of the network considerably. These networks are prone to the problem of vanishing gradients where the value of gradient become too small during back propagation due to successive multiplication. The residual blocks ensure that back propagation can also be operated without interference [25].

As shown in Fig. 6, residual learning is carried out by feeding the input data directly to the output of single layered or multi-layered convolutional network, by adding them together and further applying the ReLU activation function. Necessary padding is applied on the output convolution network to make it same size as that of the input. Other than tackling vanishing gradients problem, the network encourages feature reuse, thus increasing the feature variance of the outputs. For the 1000 class problem of ImageNet, the network uses 0.85 million learnable parameters.

The ResNet takes up an input of standard size $224 \times 224 \times 3$. As shown in Fig. 6, the architecture has 64 filters of size 7×7 in the first layer. All other filters are of size 3×3 and the number of filers increases gradually up to 512 in the final convolution layer.

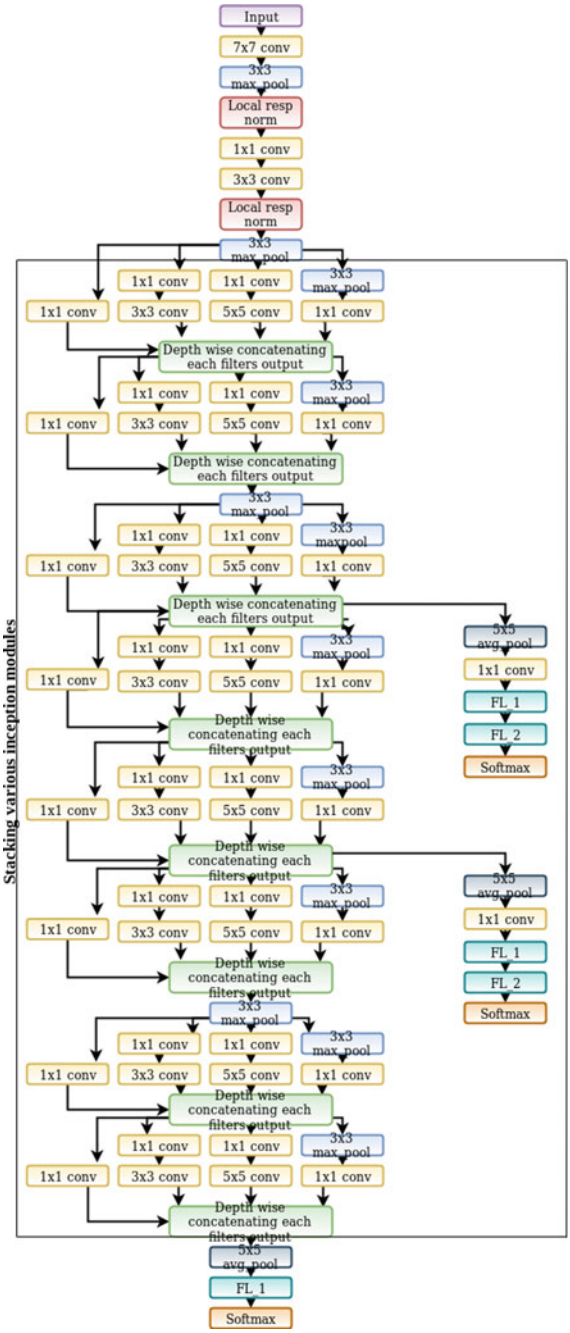


Fig. 4 Inception architecture

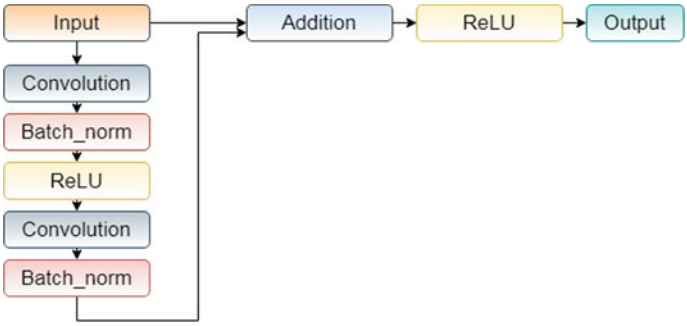


Fig. 5 Residual block of ResNet architecture

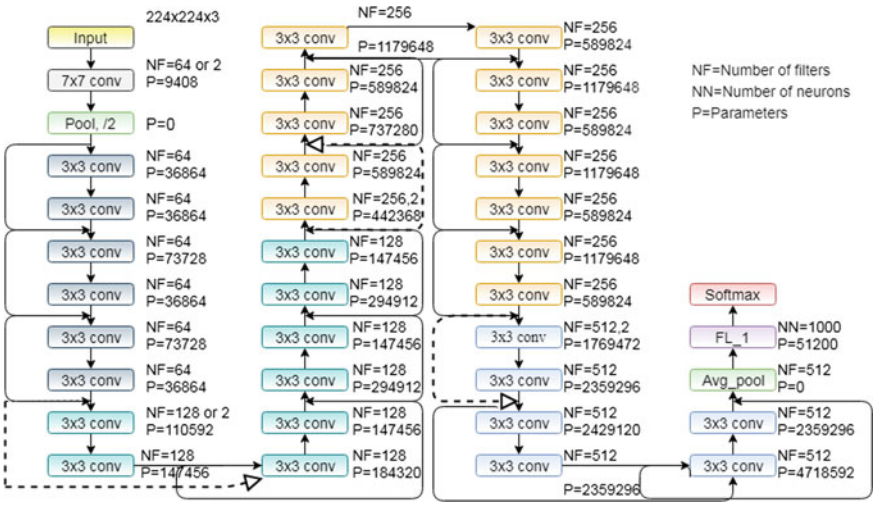


Fig. 6 ResNet architecture

2.5 MobileNet

MobileNet is designed to effectively maximize accuracy, while being careful of the restricted resources for an on-device or an exclusive embedded application. MobileNet has low-latency, low-power models, which is parameterized to meet the resource constraints of the computing devices [19]. Similar to the large scale models, MobileNet can be used for applications such as classification, detection, embeddings and segmentation. MobileNet exclusively uses separable filters, which is a combination of depth-wise convolution and point-wise convolution. It uses 1×1 filters for reducing the computational overheads of normal convolution operation. This makes the network lighter (as shown in Fig. 7), in terms of computational complexity as well as size. The MobileNet- 224 has around 5 million parameters on ImageNet classi-

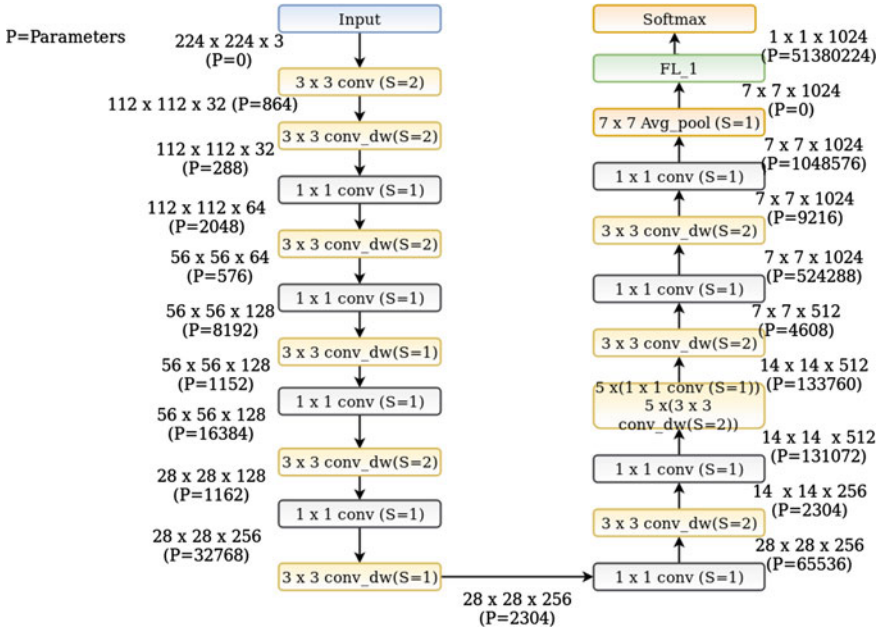


Fig. 7 MobileNet architecture with parameters

fication. MobileNet also takes up an input of size 224 × 224 × 3. The number of filters may vary from 32 to 1024 as in Fig. 7.

2.6 NASNet

The deep neural network has witnessed growth to the next generation by introducing the concept of optimized network. This idea was materialized through the concept of NAS by Google ML group. Their approach was based on reinforcement learning. The parent AI reviews the efficiency of the child AI and makes adjustments to the neural network architecture. Several modifications was done based on number of layers, weights, regularization methods, etc., to improve efficiency of the network. The architecture consists of Controller Recurrent Neural Network (CRNN) and CNN, which is to be trained as depicted in Fig. 8. The NASNet A, B, C version algorithms, choose the best cells by the reinforcement learning method as in [21]. According to Chen et al. [26], reinforced evolutionary algorithm can be implemented for selecting the best candidates. Tournament selection algorithm is implemented to eliminate the worst performing cell. The child fitness function is optimized and reinforcement mutations are performed. This further optimizes the performance of the cell structure.

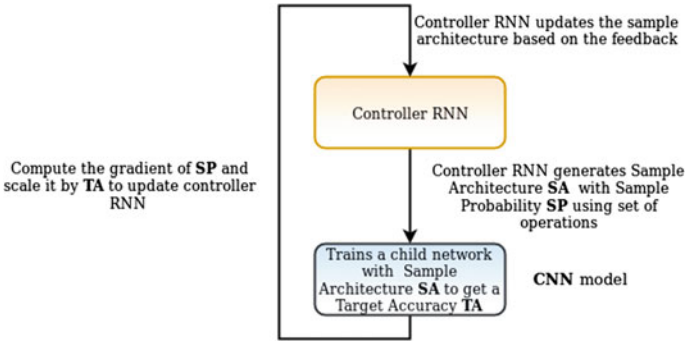


Fig. 8 Controller RNN in NASNet architecture

The set of operational blocks in the NASNet architecture is listed as below:

- Identity
- 1 × 3 then 3 × 1 convolution
- 1 × 7 then 7 × 1 convolution
- 3 × 3 dilated convolution
- 3 × 3 average pooling
- 3 × 3 max pooling
- 5 × 5 max pooling
- 7 × 7 max pooling
- 1 × 1 convolution
- 3 × 3 convolution
- 3 × 3 depthwise-separable—convolution
- 5 × 5 depthwise-separable—convolution
- 7 × 7 depthwise-separable—convolution

The NASNet architecture is trained with two types of input images of size 331 × 331 and 224 × 224, to get NASNetLarge and NASNet Mobile architectures respectively. There is an extensive growth in number of parameters while moving from NASNetMobile to NASNetLarge. NASNetMobile has 53,26,716 parameters and NASNetLarge has 8,89,49,818 parameters. This makes NASNetMobile more reliable.

Block is the smallest unit in NASNet architecture and cell is the combination of blocks, as shown in Fig. 9. A cell is formed by concatenating different blocks as in Fig. 10. The search space introduced in NASNet is by factorizing the network into cells and further subdividing it into blocks. These cells and blocks are not fixed in number or type, but are optimized for the selected dataset.

Block is an operational module as depicted in Fig. 11. The possible operations of a block include normal convolutions, separable-convolutions, max-pooling, average-pooling, identity mapping etc. The blocks maps two inputs (present and one previous, say H_0 and H_1) to a single output feature map. It takes element wise addition. If the

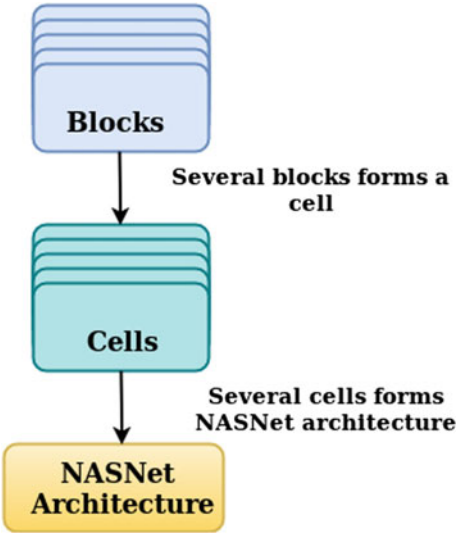


Fig. 9 Taxonomy of NASNet architecture

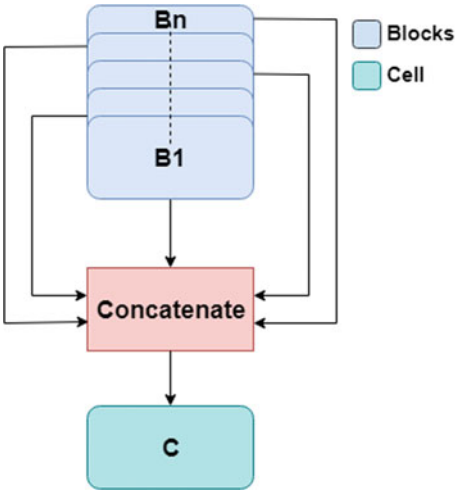


Fig. 10 Formation of a cell in NASNet architecture

cell takes a block having feature map of size H × W and stride of 1, output will be the same size as that of feature map. If the stride is 2, the size will be reduced by 2. The cells are combined in an optimized manner. The network development is focused on three factors: the cell structure, the number of cells to be stacked (N) and the number of filters in the first layer (F). Initially N and F are fixed during the search. Later N and F in the first layer are adjusted to control the depth and width of network. Once search is finished, models are constructed with different sizes to fit the datasets.

The cells are then connected in an optimized manner to develop into the NASNet architecture. Each cell is connected to two input states, termed as hidden state.

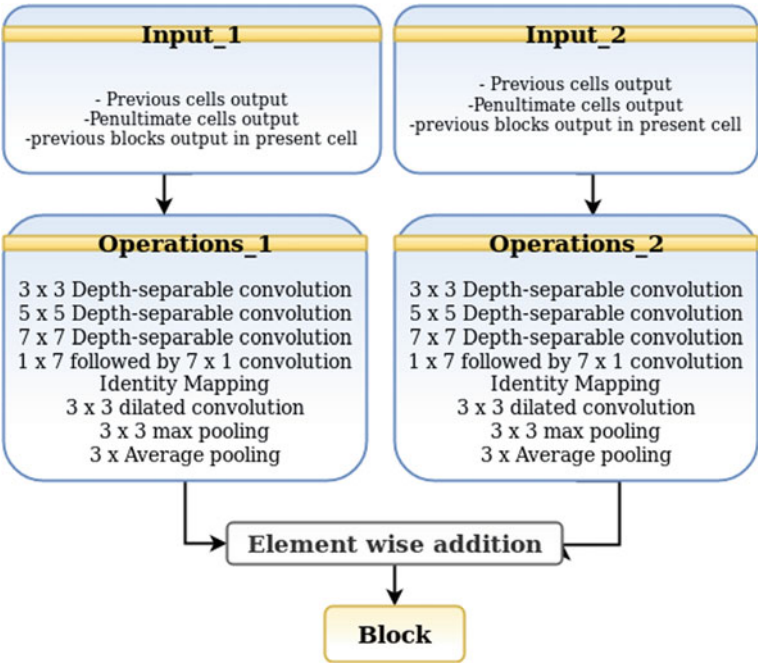


Fig. 11 Formation of a block in NASNet architecture

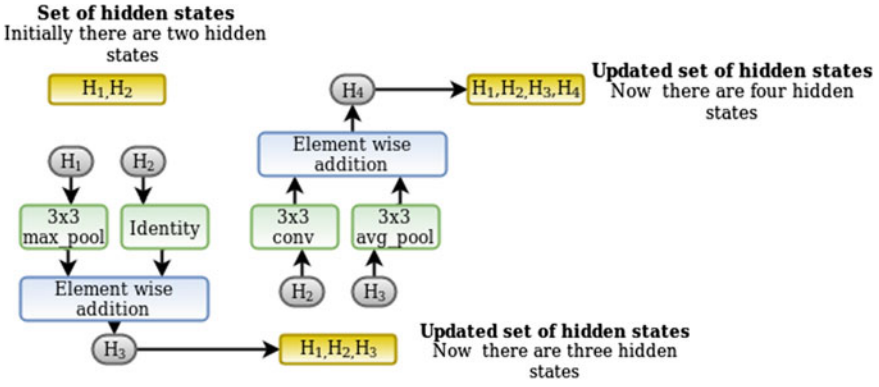


Fig. 12 Formation of hidden states inside a block

A sample of hidden state formation (4 hidden layers) is shown in Fig. 12. The hidden layers are formed through pairwise combinations and updated by concatenation operation.

Hidden layers can undergo a set of convolution and pooling operations. Instead of searching the entire cells, only the best cells are selected in NASNet architecture. This will make the search faster and thus more generalized features could be obtained. The NAS search space has a controller-child structure, where the controller is a normal cell and the child is a reduction cell, as shown in Figs. 13 and 14 respectively. Normal

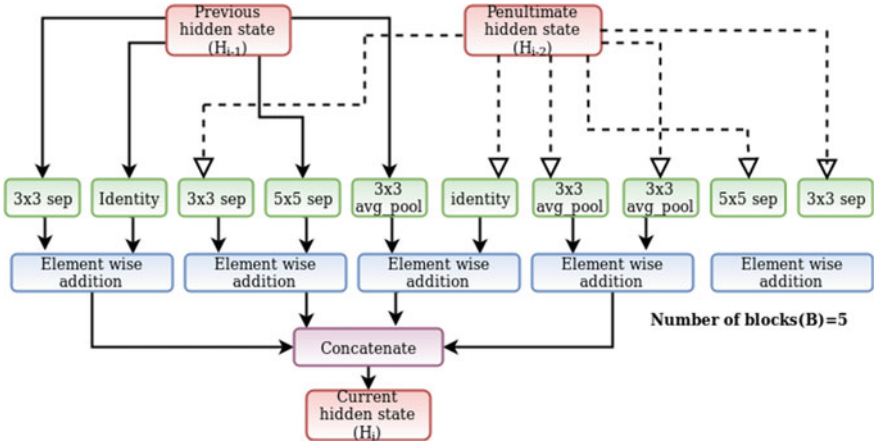


Fig. 13 Normal cell in NASNet architecture

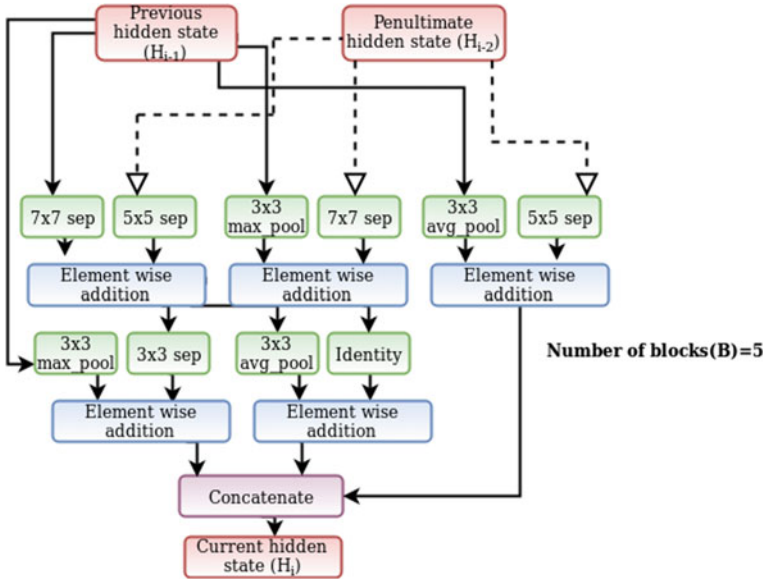


Fig. 14 Reduction cell in NASNet architecture

cell maintains the height and width dimension as same as that of the input feature map where as, the reduction cell reduces these dimensions by two. N number of normal cells are stacked in between the reduction cells to form NASNet architecture. To provide high accuracy NASNetLarge has taken N as 6 while the main concern for NASNetMobile was to run with limited resources.

For both normal cell and reduction cell, the input size of $224 \times 224 \times 3$ is reduced to a size of 7×7 at the output through the selected set of operations using

Table 3 Learnable parameters of standard deep learning architectures

Sl.no	Model	Total number of parameters
1	VGG19	14,36,67,240
2	InceptionV3	2,38,51,784
3	ResNet50	2,56,36,712
4	MobileNet	42,53,864
5	NASNet mobile	53,26,716

5B cells. A new concept named scheduled droppath is introduced in NASNet, where each path in the cell is dropped with a linearly increasing probability as training of the network progresses [21]. The reduction cell Fig. 14 implies that the H_i is formed by a series of operations on H_{i-1} and H_{i-2} cells and finally concatenating them together. The RNN repeats these prediction steps B times. This NASNet structure improved the speed of operation remarkably.

Total number of learnable parameters in each model is shown in Table 3. While coming to newer architectures a scenario on reducing the number of learnable parameters was observed.

Major operations in the above discussed architectures are consolidated in Table 4. It is observed that during the evolution, each architecture has contributed unique features to CNN models. In the early stage, combination of convolution and max-pooling were the only operations used. By giving more concern to spatial dimension, Inception module was introduced. During those times, vanishing gradient was the threat faced by CNN models. An admirable move was proposed by ResNet for resolving this problem using identity mapping. Inorder to reduce the number of learnable parameters, MobileNet came with a concept of separable-convolution. Without human involvement, NASNet has introduced a new optimized architecture using controller RNN module. How well these unique operations in each architecture were able to attain results on UERC-2019 dataset is discussed in Sect. 5.

Table 4 Salient operations of standard architectures

Operations	VGG	Inception	ResNet	MobileNet	NASNet
Normal convolution	Y	Y	Y	Y	Y
Identity			Y		Y
Average pooling			Y	Y	Y
Max pooling	Y	Y			Y
Separable—convolution				Y	Y
Batch normalization				Y	Y
Global average pooling		Y	Y	Y	Y
Factorization		Y			
Grid size reduction		Y			
Auxiliary classification		Y			
Scheduled Drop Path					Y

3 UERC-2019 Dataset Description

As ear recognition is becoming one of the most promising consideration for person identification, researchers has shown a large interest for resolving various problems equipped with ear biometric. In the beginning, researchers concentrated mainly on images of ears in constrained environment, but recent studies are progressing in unconstrained environment. Some of the available datasets for ear recognition are AWE [4], AWEx [27], WebEars [28] etc., Performance analysis of the architectures presented in Sect. 2 has evaluated using UERC-2019 dataset. Open problems emerged from UERC-2017 challenge has motivated to conduct the second series of the challenge (UERC-2019). Apart from 2017, UERC-2019 challenge focuses on the effectiveness of technology on various aspects such as gender and ethnicity [7]. Figure 15 depicts the UERC-2019 dataset, which comprises of 11,500 RGB images from 3,740 subjects. The dataset consists of varieties of realistic images obtained using web spiders, which makes automatic ear identification a difficult task. It includes images of size varying from 15 $\times$ 29 pixels to 473 $\times$ 1022 pixels, which shows the divergent nature of the input data in terms of image resolution as well as visual quality.

Sample images from UERC-2019 dataset are shown in Fig. 16. The dataset is basically branched into two parts such as public dataset and sequestered dataset. The sequestered dataset was provided only to the participants in the challenge, which consist of 500 images from 50 subjects for crosschecking the algorithms developed by the participants. So, we have chosen publically available UERC-2019 dataset for our analysis. Public dataset was split into two divisions as training and testing datasets. The images for training dataset was entirely taken from AWEx [6, 29]. The test data from public dataset contains 9,500 images from 3,540 subjects, taken from AWEx and UERC-2017 dataset. Annotations were not provided for test data as well as some subjects in the train data. The train data in UERC-2019 has 166 subjects, in which annotations were provided till 150 subjects with equal distribution of images (10 images per subject), remaining subjects appeared without annotations as well as

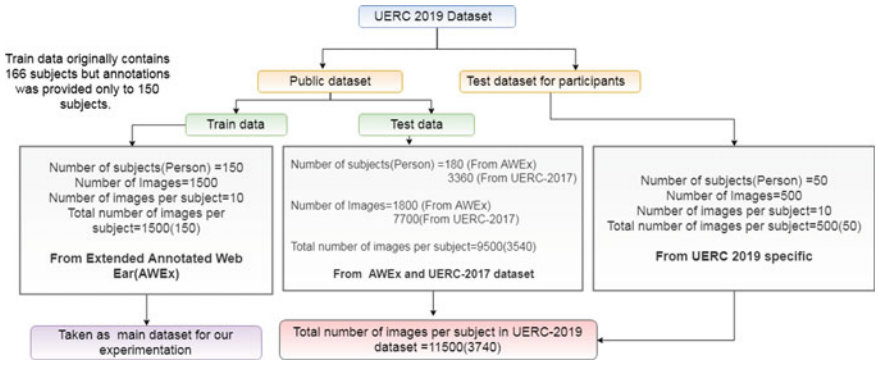


Fig. 15 UERC-2019 dataset

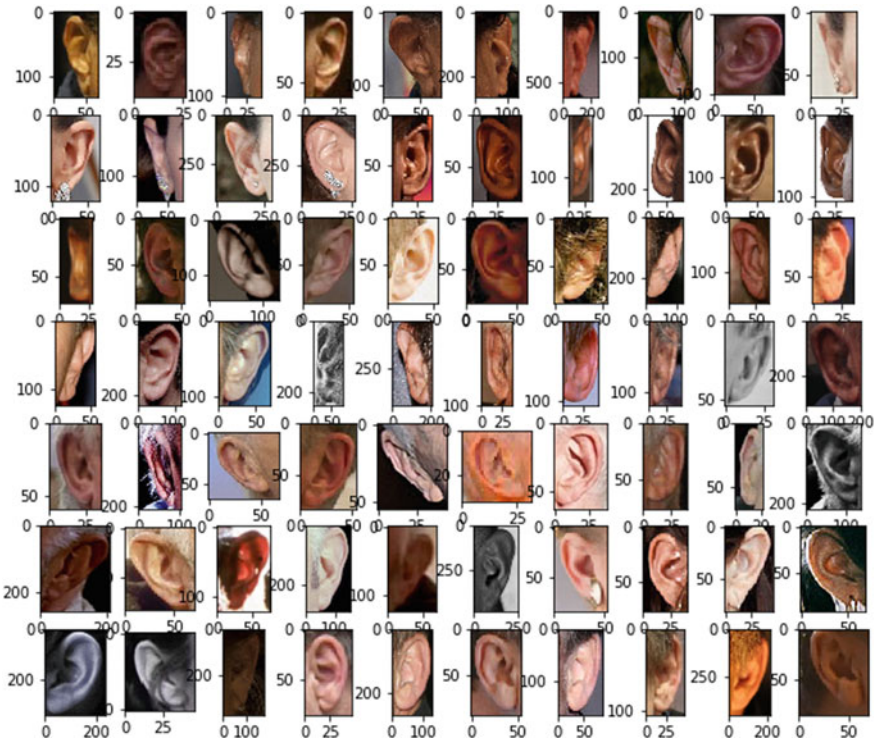
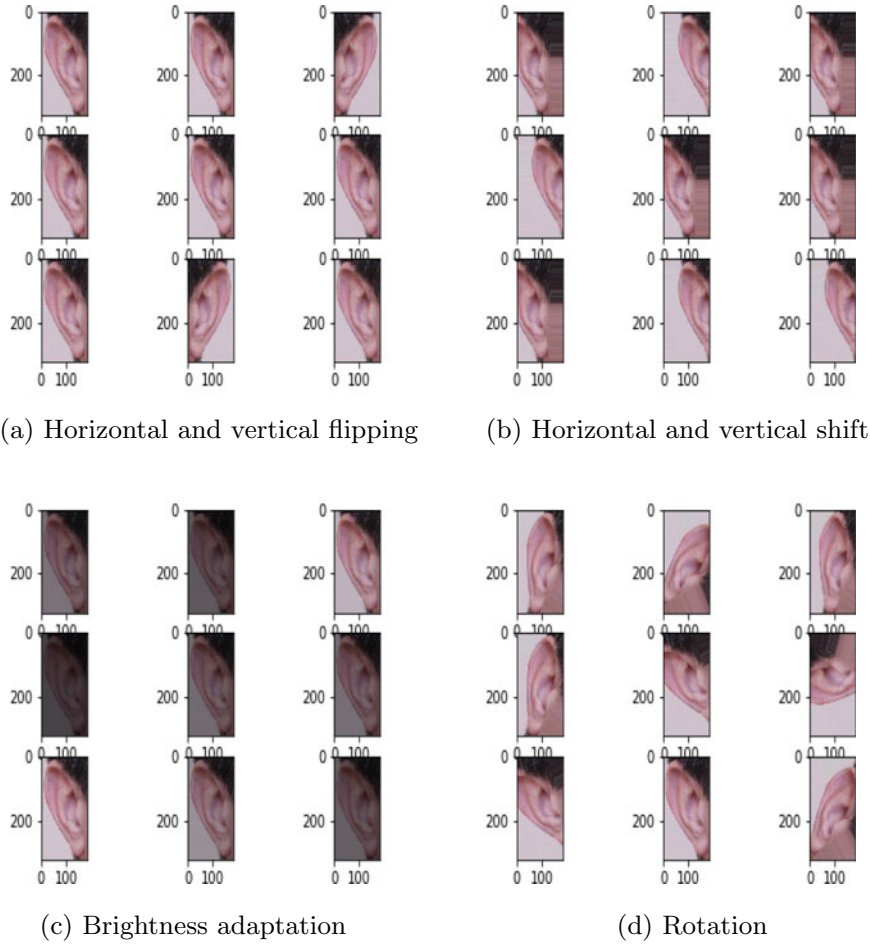


Fig. 16 Sample of images from UERC-2019 dataset

uneven distribution of images. Since the number of images in the training dataset is small, various augmentation techniques has been used inorder to increase the size of the data.

4 Design of Experiment

Data for all the experiments were taken from UERC-2019 dataset. Experiment was evaluated on 5 architectures which are having unique characteristics and are commonly used for image classification problems in Keras framework. Even though NASNet was released in 2017, we were unable to find much of its applications in various domains. This made us eager to learn more about NASNet architecture. From the study on Sect. 2, it is observed that NASNet could perform efficiently on small dataset (CIFAR-10) as well as large dataset (ImageNet). Since the dataset taken for our experiment is also small, the intention was to explore the NASNet architecture on UERC-2019 dataset. As NASNetMobile has lesser number of parameters, computations were possible on GPU as well as in CPU. Pre-processing techniques plays a vital role in image classification tasks such as upgrading features of images,



(a) Horizontal and vertical flipping (b) Horizontal and vertical shift

(c) Brightness adaptation (d) Rotation

Fig. 17 Sample of image augmentation

increasing the number of images etc. Initially, experiments were carried out on various pre-processing techniques such as data augmentation and data normalization. We have tried commonly used image augmentation techniques like shifting, flipping, brightness adaptation, rotation, zooming etc. Sample results obtained from each augmentation technique is shown in Fig. 17.

Environment Setup

- **Processor**: Intel@ Core^{TM} i5-7200U CPU
- **Platform**: Google Colaboratory (GPU support)
- **RAM**: 16 GB (CPU), 12.72(GPU)
- **Disk space**: 358.27GB (GPU)
- **Python version**: 2.7
- **Python framework**: Keras framework (2.2.4) with Tensorflow (1.13.1) backend.

The memory utilization of traditional data augmentation techniques are found to be more. To overcome this problem, Keras has introduced a class named Image-DataGenerator, which is used all over the experiment. Flow_from_directory in ImageDataGenerator is used to make the image retrieval task much easier. Keras model provides pre-trained weights of ImageNet for image classification problems. Performance analysis of all the models without pre-trained network became a curious intention. Therefore, in the beginning all the model were developed from scratch, which result in low accuracy. Later pre-trained weights from ImageNet were taken for rest of the experimentation. Inorder to reduce the number of classes from 1000 (ImageNet) to 150 (UERC-2019), various fine-tuning techniques were adopted. Then, we started the fine-tuning by stacking convolution layers and max-pooling layers over pre-trained model which, could not produce better results. Then the experimentation was based on adding only dense layers. While proceeding, it is observed that adding more dense layers on a smaller dataset results in a common problem of overfitting, so we fixed 2 dense layers on top of pre-trained model. This could reduce the problem only up to a limit, but introduction of checkpoints could reduce overfitting to a greater extend.

On moving to hyper parameter tuning the main consideration was to find best optimizer and it's learning rate. Evaluation of the performance of commonly used optimizers (SGD, RMSprop, Adagrad, Adadelta and Adam) was carried out on NASNet model. At the initial stage itself RMSprop was unable to give good results thus the evaluation was carry forwarded with rest of the optimizers. From the Fig. 18 it is clear that Adam could achieve best result in training compared to other 3 optimizers.

Next challenge was to find out optimal value learning rate for Adam. A series of trial and error method came up with a conclusion that choosing 0.001 as learning rate yields the best result. Batch sizes 32, 64 and 128 were tried with each models and optimal batch sizes were adopted to each model for further experimentation. Since the analysis to be carried out on multi-class problem the loss function chosen was Categorical Cross-Entropy. In general, each model was build using a pre-trained

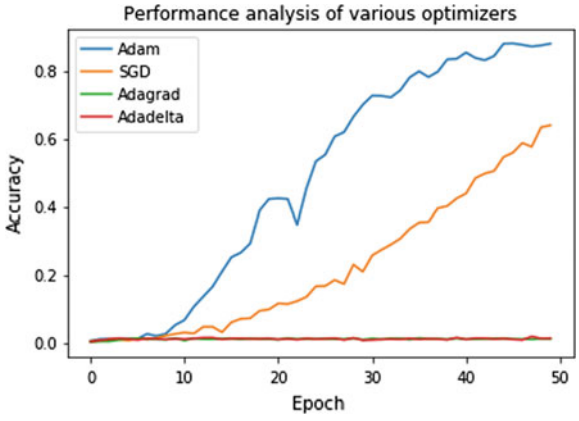


Fig. 18 Comparison of performance of optimizers on the dataset for NASNet model

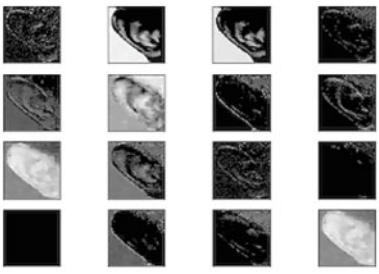
model stacked with 2 dense layers, Categorical Cross-Entropy as loss function and Adam as optimizer with a learning rate of 0.001. Finally each model were deployed for testing.

5 Results and Discussion

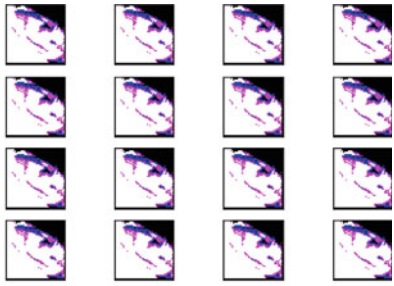
The constructed model was deployed for evaluation. Primary layer feature extraction on each model is shown in Fig. 19.

Figure 20 shows representation of feature extraction in intermediate blocks on different models. As we could see various models are learning features in a different manner. Like this, on each blocks the features learned are updated.

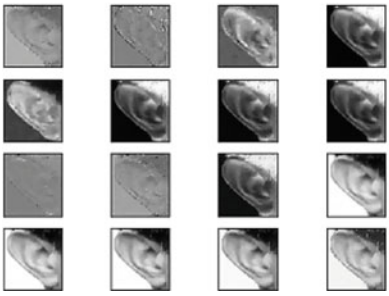
The results exhibited same behaviour even scaling to higher number of epochs. Therefore, we kept a baseline of 100 epochs which is reliable in CPU as well. Based on the fair experiments mentioned above, we observed that NASNet could perform slightly better while comparing to other models for this particular experimental setup. The epoch which gave highest testing accuracy throughout the experiment has given



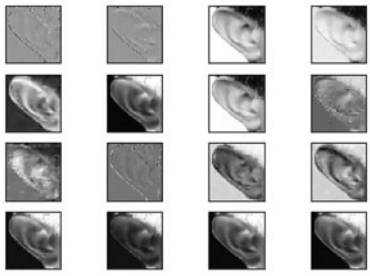
(a) Feature map from first CNN layer of VGG19



(b) Feature map from first CNN layer of ResNet

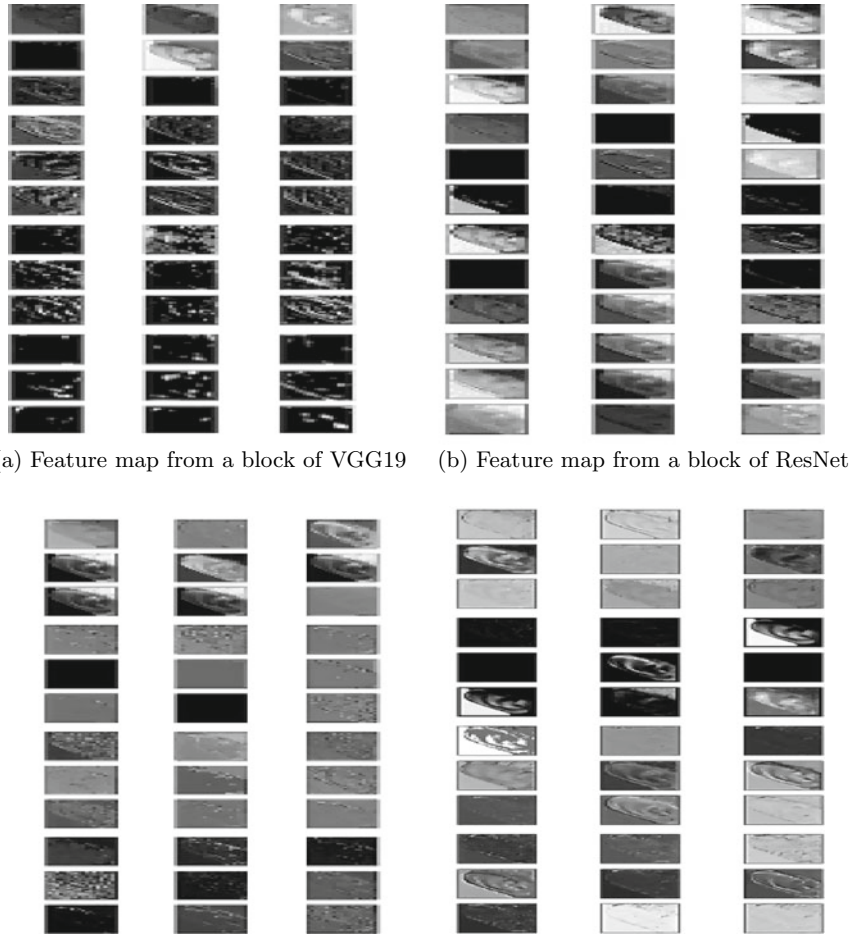


(c) Feature map from first CNN layer of MobileNet



(d) Feature map from first CNN layer of NASNet

Fig. 19 Feature map on each model



(a) Feature map from a block of VGG19 (b) Feature map from a block of ResNet

(c) Feature map from a block of MobileNet (d) Feature map from a block of NASNet

Fig. 20 Feature map from a block on each model

in the table. Other criterion considered for the performance analysis of developed model was computation time, batch size and total number of parameters depicted in Table 5.

InceptionV3 was showing poor accuracy (less than 5%) from training itself, therefore we were forced to stop further evaluation on this model. Since VGG16 also could not yield better result while training, we went for VGG19. Pictorial representation of final models are given in Fig. 21. As we could see all the models except VGG19 were performing well during training period, while there was a drastic drop in test accuracy.

Table 5 Comparison of models in terms of computational time, batch size and total no. of parameters

Model	Total no. of epochs	Computation time (s)	Batch size	Total no. of parameters
VGG19	100	19	128	3,08,40,106
Resnet50	100	13	32	3,44,48,086
MobileNet	100	11	128	7,50,45,398
NASNet	100	16	128	2,37,87,734

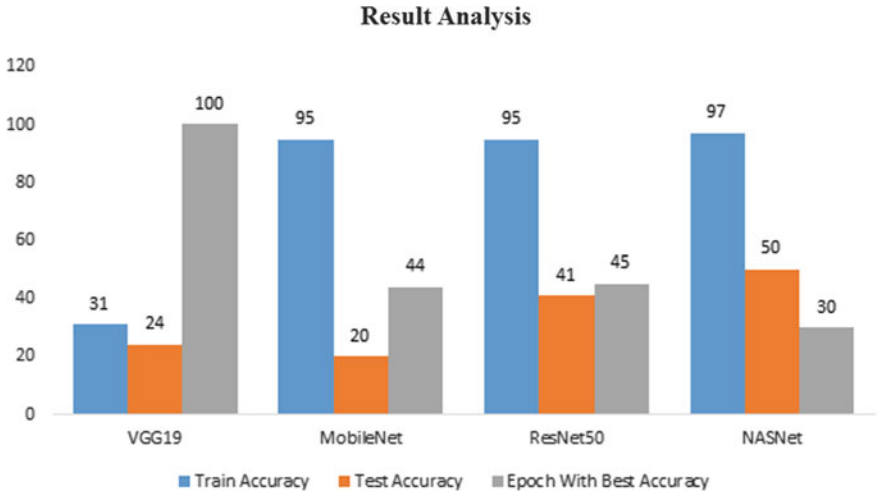


Fig. 21 Performance analysis of standard CNN architectures based on classification accuracy

Table 6 The performance metrics using NASNet model

Accuracy	Precision	Recall	F1-Score
50.4%	47.3%	47.1%	53.4%

Diagrammatic representation of epoch with high accuracy is depicted in Fig. 22. From the figure it is clear that NASNet could achieve it's best accuracy on an average of 30 epoch but the same could be achieve by ResNet, MobileNet and VGG19 on 45, 44 and 100 epochs respectively.

The optimized search architecture of the NASNet model has made it a suitable candidate for unconstrained ear recognition over the other architectures considered here. The consistency of the performance was further evaluated through the performance measures such as accuracy, precision, recall and f1-score as an average of 150 classes. The performance measures obtained for ear recognition using NASNet model is tabulated in Table 6.

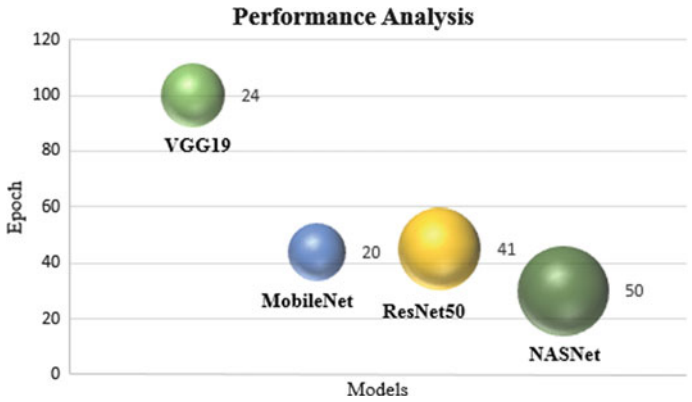


Fig. 22 Performance analysis of standard architectures used for ear recognition based on number of epochs

Table 7 GPU memory utilization of various models

Model	RAM size out of 12.72 GB	Disk space out of 358.27 GB
VGG19	2.51 GB	28.04 GB
ResNet	3.35 GB	31.16 GB
MobileNet	2.82 GB	40.10 GB
NASNet	5.22 GB	30.82 GB

The Table 7 shows memory utilization of each model on GPU in our constrained environmental setup. It is observed that all the models have used limited amount of memory as well as disk space.

6 Conclusions and Future Directions

Applications of deep neural networks are growing exponentially and continues to be one of the major verticals of AI. The prime objective of this chapter was to analyse the performance of unique DL architectures, especially on trivial problems of biometric classifications using unconstrained ear dataset. We explored the major features of the selected state of the art networks. Along with that, the possibilities of the new paradigm of optimized deep neural network was also analysed. From the study we could observe that many of the selected architectures could perform well in training but during evaluation with small test data provided, there was a drop in performance. In this constrained environmental setup we could observe that NASNet was giving moderately better results. This opens up a new door for optimized neural networks in image classification. By incorporating datasets of larger size, better models can be generated. In that case, the results could be further enhanced. Further

enhancements has to be done in order to improve the performance of the system using various environmental setup and also concentrating on each model separately. Thus, in future, ear as a bio metric identification method can be introduced by fine tuning the training models. A person can be identified by applying the test image from any environment.

References

1. LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. Nature 521.7553: 436.
2. Hatcher, William Grant, and Wei Yu. 2018. A survey of deep learning: platforms, applications and emerging research trends. *IEEE Access* 6: 24411–24432.
3. Sundararajan, Kalaivani, and Damon L. Woodard. (2018). Deep learning for biometrics: A survey. *ACM Computing Surveys (CSUR)* 51 (3): 65.
4. Kumar, Ajay, and Wu Chenye. 2012. Automated human identification using ear imaging. *Pattern Recognition* 45 (3): 956–968.
5. Nejati, Hossein, Li Zhang, Terence Sim, Elisa Martinez-Marroquin, and Guo Dong. (2012). Wonder ears: Identification of identical twins from ear images. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, 1201–1204. IEEE.
6. Emersic, Ziga, Dejan Stepec, Vitomir Struc, Peter Peer, Anjith George, Adii Ahmad, Elshibani Omar et al. 2017. The unconstrained ear recognition challenge. In *2017 IEEE International Joint Conference on Biometrics (IJCB)*, 715–724. IEEE.
7. Emersic, Ziga, B. S. Harish, Weronika Gutfeter, Jalil Nourmohammadi Khiarak, Andrzej Pacut, Earnest Hansley, Mauricio Pamplona Segundo et al. 2019. The unconstrained ear recognition challenge 2019-arxiv version with appendix. arXiv:1903.04143.
8. Alom, Md Zahangir, et al. 2018. The history began from AlexNet: A comprehensive survey on deep learning approaches. arXiv:1803.01164.
9. Unnikrishnan, Anju, V. Sowmya, and K. P. Soman. 2019. Deep learning architectures for land cover classification using red and near-infrared satellite images. *Multimedia Tools and Applications* 1–16.
10. LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86 (11): 2278–2324.
11. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 1097–1105.
12. Zeiler, Matthew D., and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, 818–833. Springer, Cham.
13. Lin, Min, Qiang Chen, and Shuicheng Yan. 2013. Network in network. arXiv:1312.4400.
14. Simonyan, Karen, and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556.
15. Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1–9.
16. He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.
17. Larsson, Gustav, Michael Maire, and Gregory Shakhnarovich. 2016. Fractalnet: Ultra-deep neural networks without residuals. arXiv:1605.07648.

18. Huang, Gao, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4700–4708.
19. Howard, Andrew G., Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861.
20. Chollet, François. 2017. Xception: Deep learning with depthwise separable-convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1251–1258.
21. Zoph, Barret, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* 8697–8710.
22. Kurup, R. Vimal, V. Sowmya, and K. P. Soman. 2019. Effect of data pre-processing on brain tumor classification using capsulenet. In *International Conference on Intelligent Computing and Communication Technologies* 110–119. Singapore:Springer.
23. Sabour, Sara, Nicholas Frosst, and Geoffrey E. Hinton. 2017. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems* 3856–3866.
24. Szegedy, Christian, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* 2818–2826.
25. K. He, X. Zhang, S. Ren, J. Sun. 2016. Deep residual learning for image recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 770–778.
26. Chen, Yukang, Qian Zhang, Chang Huang, Mu Lisen, Gaofeng Meng, and Xinggang Wang. 2018. *Reinforced evolutionary neural architecture search*. CoRR.
27. Emersic, Žiga, Blaž Meden, Peter Peer, and Vitomir Štruc. 2018. Evaluation and analysis of ear recognition models: Performance, complexity and resource requirements. *Neural Computing and Applications* 1–16.
28. Zhang, Yi, and Zhichun Mu. 2017. Ear detection under uncontrolled conditions with multiple scale faster region-based convolutional neural networks. *Symmetry* 9 (4): 53.
29. Emersic, Z., V. Struc, and P. Peer. 2017. Ear recognition: More than a survey. *Neurocomputing* 255: 26–39.