Array 27 (2025) 100428

Contents lists available at ScienceDirect

Array

journal homepage: www.sciencedirect.com/journal/array

Revival of Muslin by Phuti Karpas plant identification with convolution neural network

Redwan Ahmed Rizvee [a,b], Omar Farrok [c], Mahamudul Hasan [b,*], Faisal Farhan [c], Md Hafanul Islam [b], Md Khalid Hasan [b], Abidur Rahman [c], Maheen Islam [b], Md Sawkat Ali [b], Taskeed Jabid [b], Mohammad Rifat Ahmmad Rashid [b], Mohammad Manzurul Islam [b]

[a] Department of Computer Science and Engineering, University of Dhaka, Dhaka, 1000, Bangladesh
[b] Department of Computer Science and Engineering, East West University, Dhaka, 1212, Bangladesh
[c] Department of Electrical and Electronic Engineering, Ahsanullah University of Science and Technology, Tejgaon, Dhaka, 1208, Bangladesh

ARTICLE INFO

Keywords:
AlexNet
CNN
Machine learning
Plant recognition
Tree identification
VGGNet

ABSTRACT

Phuti Karpas, historically central to cotton production and thought extinct, has re-emerged in botanical research, prompting a need for reliable identification methods. This study develops a systematic approach for classifying Phuti Karpas leaves using various convolutional neural networks (CNNs), including AlexNet, Inception, VGG16, MobileNetV2, and a custom-designed baseline model. A unique dataset of 2354 leaf images was curated, with two main classes: Phuti Karpas and Non Phuti Karpas, the latter including 14 other plant types to enhance model robustness. Each model was evaluated on metrics like accuracy, precision, recall, computational time, and memory efficiency. AlexNet yielded the highest average accuracy, while the custom baseline model, optimized for mobile deployment, provided comparable accuracy with faster inference. To demonstrate real-world usability, an Android app was created for real-time Phuti Karpas identification, offering an accessible tool for field researchers and conservationists. This work not only advances deep learning applications in plant taxonomy but also aids in the cultural and scientific revival of Phuti Karpas.

1. Introduction

In recent times, studies on the automatic plant identification and classification have drawn increased attention. Professionals such as agronomists, biologists, foresters, etc. are interested in classifying plants into relevant taxonomies [1]. Usually, plant identification and classification can be accomplished by visually inspecting the plant features, such as floral parts, leaves, plant fruits, etc. Modern computer vision techniques and several mobile applications are being developed to make the task of plant classification and identification easier [2]. Plant categorization in the modern era has become an essential task because of the applications of different plant species in the field of medicine and agriculture. The application also includes weed detection, growth estimation, and disease diagnosis [2]. Also in the field of medicine, different plant species are used as medicine to eradicate diabetes and cardiovascular diseases [3–6]. Plant taxonomy is used to categorize plants, where plants are divided into hierarchical groups, that may be searched endlessly based on biased plant features, according to recent studies[7]. This work also explores computer vision-based deep learning techniques in classifying a specific type of rare plant entitled, Phuti Karpas. It also evaluates the feasibility of designing a custom architecture that provides a competitive performance in accuracy with a quality computational efficiency by making the solution suitable for real-life mobile devices (see Figs. 22 and 23).

A good number of research works have concluded that the task of plant identification and classification based on plant leaves are more convenient and reliable [1,5]. However, due to the huge number of plant species and innumerable labeled or unlabelled data, visual identification of the plant species through manual inspection has become quite challenging [1]. Also, the variations in light, posture, and orientation impact

* Corresponding author.
E-mail addresses: rizvee@cse.du.ac.bd, redwan.rizvee@ewubd.edu (R.A. Rizvee), omarruet@gmail.com, omar.eee@aust.edu (O. Farrok), munna09bd@gmail.com (M. Hasan), faisalfarhan.eee@aust.edu (F. Farhan), mdhafanul@gmail.com (M.H. Islam), khalidrion@gmail.com (M.K. Hasan), abidr92@gmail.com (A. Rahman), maheen@ewubd.edu (M. Islam), alim@ewubd.edu (M.S. Ali), taskeed@ewubd.edu (T. Jabid), rifat.rashid@ewubd.edu (M.R. Ahmmad Rashid), mohammad.islam@ewubd.edu (M.M. Islam).

https://doi.org/10.1016/j.array.2025.100428
Received 1 September 2024; Received in revised form 19 December 2024; Accepted 11 June 2025
Available online 16 June 2025

the recognition task greatly. Over the time, the alteration in leaf and its colour change under diverse climate conditions entails many difficulties. Thus, due to plant diversity, variations in clutter, orientation, background, viewing perspective, etc, recognizing plant species from videos or pictures is challenging [2]. Due to the underlying challenges and versatile variations in features embedded in different plant species a significant number of studies have been addressing the plant recognition and classification problem with the goal of improving current performance.

The discriminative plant features, such as unique colour, leaf texture, ventilation, eccentricity etc. are considered as key features in this regard [1]. Traditional machine learning approaches have been designed focusing on the color and shape of the plant [8,9], texture [10,11], and venation features [12,13] for plant recognition [17,20]. Moreover, with the advancement of modern deep-learning methods a pool of new solutions has emerged due to their efficiency in identifying features compared to the traditional machine-learning-based methods [14]. Modern deep Convolutional Neural Network (CNN) uses manually annotated data to extract features automatically in an end-to-end manner rather than depending on handcrafted features of plants such as color, shape, and texture [1,7]. Though CNN requires a huge amount of data for training and faces problems such as overfitting issues, this method is quite efficient in learning features and giving accurate results [15].

This work focuses on identifying the Phuti Karpas tree plant using CNN and image processing techniques by studying its different features. It was widely used in the earlier centuries for producing fine cotton. The cotton produced from these trees was used for making cotton cloth. They were widely found in Mesopotamia or present Iraq along with the Tigris River from 900 BCE – 270 CE. Their similar variety was also prevalent in Bengal and the cotton cloth produced from these trees was named “Muslin”. However, over the time being, this tree was disidentified and as a result Muslin in Bengal was no longer available since the last 200 years. Recently in Bangladesh, it was proposed to get the lost historical Phuti Karpas tree back, and hence samples of Muslin cloth were collected from the British Museum for adequate information. The DNA of Phuti Karpas was identified from dried preserved leaves that were available in the Royal Botanical Garden. The birthplace of Phuti Karpas was discovered on the riverbank of the Meghna River with the help of modern satellites measuring the diversion of the river. After a long investigation, a tree was found that matched 70 % with the Phuti Karpas plant in terms of DNA. A modern technology called DNA sequencing was used to match the DNA of the tree with the Phuti Karpas plant. Then with the aid of modern agro-technology, necessary steps were taken for their rebirth in Bengal. Therefore, Phuti karpas has become again a part of concern and this study focuses on developing an image analysis-based solution to properly distinguish Phuti Karpas leaves from the others. We believe that this effort would be supportive for various use cases related to its leading and widespread adoption of this long-lost precious agricultural entity. Main contribution of this paper is as follows.

- Collection and pre-processing of a custom-collected dataset containing Phuti Karpas plants and other relevant non-Phuti Karpas instances.
- Evaluating the effectiveness of the existing well known and widely used CNN-based architectures in recognizing Phuti Karpas plants along with testing a custom baseline architecture that provides the fastest speed, moderate accuracy with low resource consumption.
- Comparing the performance of all the architectures in terms of accuracy, precision, recall, learning stability, weight storage memory, and computational time.
- Deployment of the architecture in mobile as an Android application for various real-life use cases.

In a brief, our work is a pioneering direction in classifying Phuti Karpas plants. For the training and evaluation, we have custom collected and prepared a novel Phuti Karpas dataset. Furthermore, we also explore the quality of well-known deep learning image classification architectures in detecting such plants and exhibit the potential to design a custom architecture with a quality trade-off between speed and performance.

The rest of this paper is structured as follows. Section 2 comprises related works. Section 3 discusses existing and the proposed architectures. Next, Section 4 includes a discussion of the custom-prepared dataset along with an extensive performance analysis of the addressed architectures. Section 5 contains a discussion and summary of the whole work along and Section 6 provides a discussion on the current limitation and possible directions for further extensions.

2. Related works

To classify leaves including Phuti Karpas using convolutional neural networks (CNNs), several systematic approaches can be employed. Each approach involves capturing and analyzing unique features of the leaves, such as their shape, pattern, and texture. A short description of various methods that can be used for this purpose has been provided in the following.

Baseline CNN Approach starts with a basic CNN model that fits to the dataset [16]. It includes a few convolutional layers for feature extraction. Pooling layers support to reduce dimensionality of the data. Then the connected layers relate the extracted features to specific categories. This is a basic method which is suitable for small datasets.

The approach of Transfer Learning Using Pretrained Models influences existing CNN architecture such as VGG16, ResNet, MobileNet, which have been pretrained on large datasets [19,21,22]. It can extract meaningful features from the images. This model can be adopted to classify leaves effectively. It is useful for a limited number of labeled data because it minimizes the training time and improves accuracy. Ensemble learning with multiple CNN combines predictions of multiple CNNs to make an improved classification system [34–38]. In this method, each CNN is trained independently focusing on specific features or patterns of the leaves. By combining the outputs, the final prediction is obtained. Thus, the prediction becomes accurate. Although this method is effective, it is resource intensive. For this reason, it requires high computational resource.

Custom Architectures for Leaf-Specific Features includes CNN architecture aligned to specific features of Phuti Karpas leaves, such as their edge shapes, textures, vein structures. Edge detection can focus on the patterns, which can be processed by specific features within the CNN architecture [40–44]. This approach is highly customized and leads to better performance. Applying this approach requires in-depth understanding of the dataset’s properties.

Lightweight Models for Edge Deployment means application of lightweight CNN. MobileNet3 is an example of this kind of model. It requires low computational capacity. It results in high efficiency and accuracy. The training can further be enhanced with pruning or quantization to reduce model and improve inference speed. Thus, it is suitable for real time applications. Hierarchical classification involves multistage approaches, and the task is divided into a few levels. Firstly, CNN broadly categorizes leaves, then another CNN refines the classification. It identifies specific species. It is effective for datasets with complex hierarchical relationships.

In Attention Mechanisms for feature integrates attention mechanisms into CNNs. It focuses on the unique features of the leaves. This kind of approach supports the CNN to prioritize specific regions of the image. It increases the accuracy and interpretability of the model because of focusing on the most relevant features of the images/data. In Data Augmentation and Synthetic Data Generation expands the dataset using augmentation techniques such as rotating, flipping, or adjusting the color of leaf. Generative Adversarial Networks can create synthetic data to simulate diverse leaf appearance. They can address the challenges of small datasets. It can provide diverse examples for CNN to learn from. A precaution of this method is to be careful about inclusion

Table 1
Advantages, weaknesses, and challenges of applying different approaches.

Approach	Strength/Advantages	Weakness	Challenges
Baseline CNN	Simple, easy to implement	Limited scalability	Limited performance on complex data
Transfer Learning	High accuracy, quick/faster training	High computational requirements	May not capture unique features
Ensemble Learning	High accuracy, robust performance/predictions	Expensive in resources	Time-intensive, risk of overfitting
Custom Architectures	Full control, fit to problem	Requires domain expertise	Requires extensive experimentation
Lightweight Models	Can easily be applied	Slight trade-off in accuracy	May sacrifice accuracy
Hierarchical Classification	Captures class relationships, can handle complex datasets	More annotation and training needed	Complex to implement
Attention Mechanisms	Enhanced focus on features	Architectural complexity	Computationally intensive
Data Augmentation	Better generalization, reduces overfitting	Risk of overfitting on synthetic data	Can introduce noisy transformations

of unrealistic samples.

It is noteworthy that each of the approaches has its own advantages and disadvantages. Combined approaches can be applied to obtain better accuracy. For example, transfer learning can be paired with data augmentation to improve the utility of a small dataset. Similarly, lightweight models can incorporate attention mechanisms for proper classification. Before applying combined approach, dataset size, computational capacity, and intended application can be considered. By systematically exploring and implementing the strategies, effective systems can be built for classifying Phuti Karpas leaves, catering to diverse scientific and practical needs. Advantages, weaknesses, and challenges of applying different approaches for identifying Phuti Karpas are summarized in Table 1. A summary of various possible approaches for this purpose are illustrated in Fig. 1.

3. Methodology

This section incorporates the detailed discussion on the deployed architectures for plant leaf identification.

3.1. Convolutional neural network

CNN is a special type of deep learning method, broadly implemented in image recognition, object classification, and other computer vision research [18]. A typical CNN mainly constitutes of Convolutional Layers (CONV), Pooling Layers (Pool), and Fully Connected Layers (FC) [24]. CONV and Pool layers contain different size of filters or kernels. There also exists various activation functions, e.g., Rectified Linear Unit (ReLU) function, tanh, etc. to control the firing number of different neurons (or layers). CNNs are made up of a series of layers that take an input image, pass through different layers and activation functions, and predict the output class or label probabilities. One obvious advantage of CNN is that it takes raw pixel intensity as a flattened input image vector rather than deploying handcrafted feature extraction methods [39]. The different layers in CNN consist of some learnable filters that can automatically identify the complex filters in an image and after combining the results of the filters, it can predict the label probabilities or class of input image [25]. In CNN layer, the neurons are connected to a small area of previous layer neurons. The first layer detects the minimum level features; subsequent layers detect mid-level and high-level features, respectively. CNNs have gained much popularity in image recognition due to this distinctive technique of building up successive layers that extract from lower to higher-level features [23].

CONV is an essential layer for CNN architecture as CNN learn features of an input image by applying filters. The CONV layer mainly consists of small sized filters usually in the dimension of [3 × 3] or [5× 5] and some feature maps. Instead of going through the full image, the input volume is convolved with the filters and the characteristics are learnt at a specific spatial point. After all the required filters have been implemented to the input volume, the final output volume is created by combining all to a matching 2-D feature map which was formed as a filter slide across the network's width and height. Let the input volume in CONV layer is represented as [W_{inconv} × H_{inconv} × D_{inconv}] that corresponds to the input image's spatial dimensions and the output volume in CONV layer is represented as [$W_{outconv}$ × $H_{outconv}$ × $D_{outconv}$]. Let the four hyper parameters that correspond to the filter numbers, filter size, stride, and zero padding amount are represented as [K_{conv}, F_{conv}, S_{conv}, P_{conv}]. The mathematical relationship among the parameters can be shown as follows.

$$W_{outconv} = \left(\frac{W_{inconv} - F_{conv} + 2P_{conv}}{S_{conv}} + 1\right) \tag{1}$$

$$H_{outconv} = \left(\frac{H_{inconv} - F_{conv} + 2P_{conv}}{S_{conv}} + 1\right) \tag{2}$$

$$D_{outconv} = K_{conv} \tag{3}$$

Pooling Layer is termed as the intermediate layer in Convolutional Neural Network. This layer's primary function is to compress the size of the incoming input along the spatial dimensions. A pooling layer can down sample or compress an incoming volume of [64 × 64 × 12] to a volume of [32 × 32 × 12]. Hence, it reduces over-fitting and network computations by down sampling the previous layer's feature maps derived from various filters. If, the input volume for a pooling layer is represented as [W_{inpool} × H_{inpool} × D_{inpool}], the output volume is represented as [$W_{outpool}$ × $H_{outpool}$ × $D_{outpool}$], and the two parameters that corresponds to the filter size and stride are represented as [F_{pool}, S_{pool}], then the mathematical relation that combines all of them can be written as follows.

$$W_{outpool} = \left(\frac{W_{inpool} - F_{pool}}{S_{pool}} + 1\right) \tag{4}$$

$$W_{outpool} = \left(\frac{H_{inpool} - F_{pool}}{S_{pool}} + 1\right) \tag{5}$$

$$D_{outpool} = D_{inpool} \tag{6}$$

FC is denoted as the final layer of CNN that predicts the final class or label based on the given input. The neurons in this layer are completely linked to the neurons in the previous layer. The output dimension of a Fully Connected Layer is [W_{FC} × H_{FC} × N_{FC}] where N_{FC} refers to the number of classes, H_{FC} represent height and W_{FC} represent width that are considered for classification.

In this research, CNN along with some of its popular architectures such as AlexNet, VGGNet and a newly proposed network is used for detecting Phuti Karpas plant. The research involves object recognition that detects plant in an image and image classification to recognize which species of plant it belongs to. For this approach the raw dataset of the plant images is trained and later tested with an unseen image to predict the labels correctly. Furthermore, the research aims to find out which of the tested CNN architectures can perform effectively with higher accuracy for the Phuti Karpas plant recognition.

3.2. CNN architectures

3.2.1. AlexNet

AlexNet is a popularly used network structure of CNNs consisting of numerous innovations and contributions. This special network structure

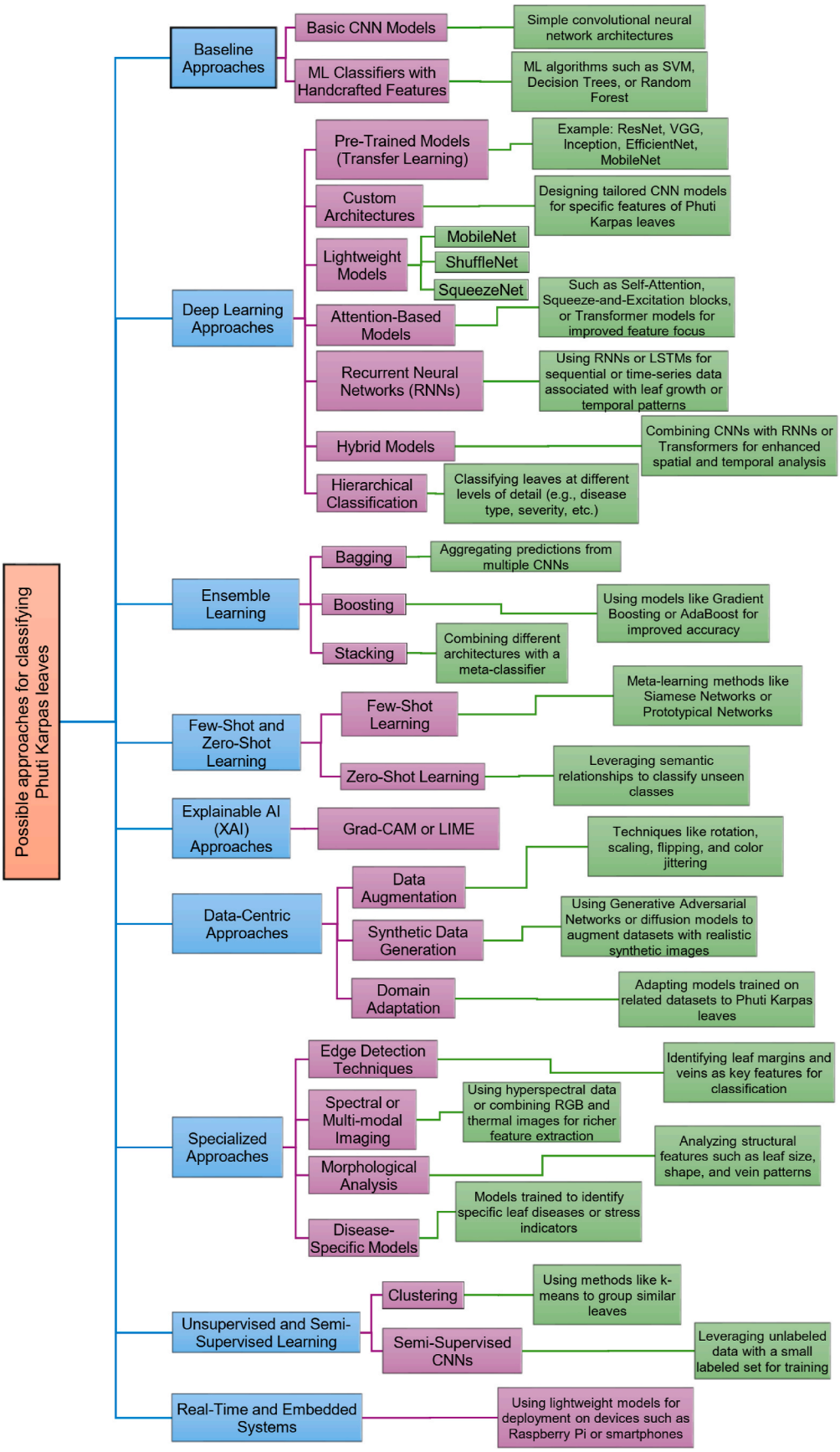

Fig. 1. Illustration of various approaches for classifying/identifying Phuti Karpas in a concise format.

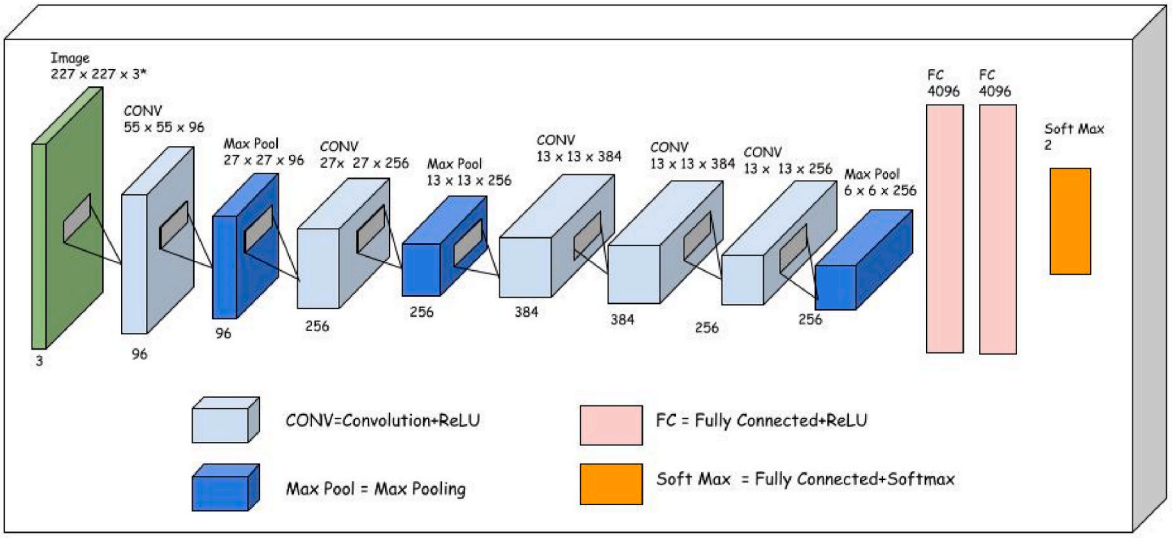

Fig. 2. Visualization of the layers of the AlexNet architecture.

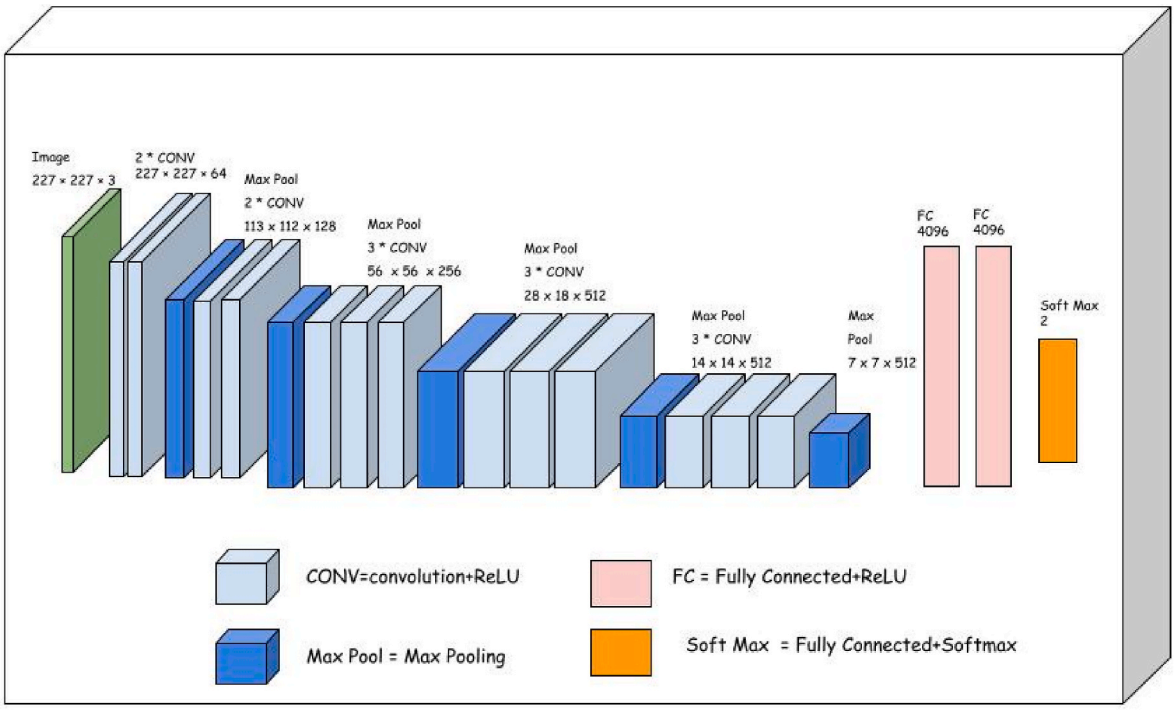

Fig. 3. Visualization of the layers of the VGG16 architecture.

was proposed in Ref. [25] that got much recognition due to its novel dropout techniques and development of the activation function-ReLU which resolves over-fitting problem. The function ReLU is preferred over traditional activation function as it can eradicate gradient vanishing problem [26]. It is defined in the following.

$$ReLU(x) = max(x, 0). \tag{7}$$

Also, to reduce the overfitting problem, dropout strategy was employed especially in the fully connected network layers. Dropout causes neurons to work together, reducing joint adaptation and improving generalization. One noteworthy characteristic of AlexNet is that it can improve recognition accuracy very efficiently despite decreasing the number of parameters. AlexNet's architecture is made up of approximately 650,000 neurons and 60 million parameters [26]. Regardless to the fact that AlexNet was run on two graphics processing units (GPUs), researchers nowadays implement AlexNet using a single GPU [27]. Several techniques such as overlapping, pooling, and multiple GPU training are used for training AlexNet to improve the accuracy.

In Fig. 2, AlexNet architecture is visualized to show how the feature maps are propagated through different layers and activation functions. For each layer (Convolution, Max Pool, and Fully Connected) a number of filters or neurons have been shown accordingly. For each convolutional layer except the first one, padding was kept such as the input and output height and width dimensions stay the same. Also, each convolutional layer uses ReLU as the activation function. To conduct the experiments in this literature, the input image's dimensions were 227 × 227 × 3, where 3 denotes the number of colour channels. Also, the problem focused on classifying two classes. The total number of

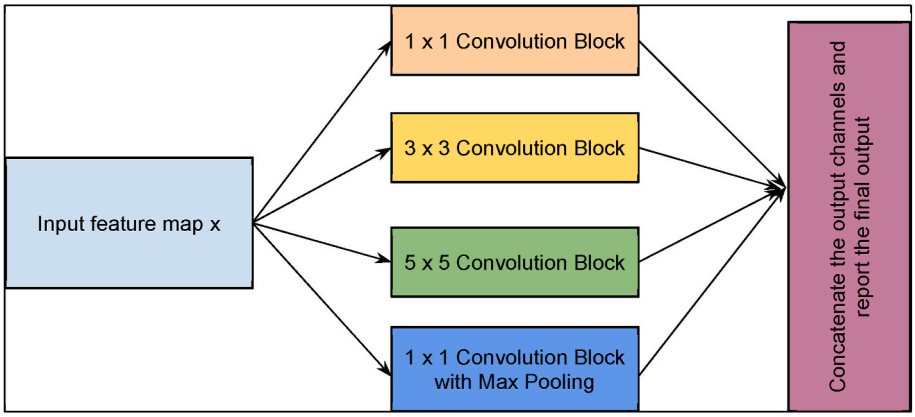

Fig. 4. General view of Inception Blocks.

Table 2
Summarized information of the feature maps of InceptionNetV1 architecture.

Layer	#Filters/ Neurons	Filter size	Stride	Size of feature map	Activation function
Image				227 × 227 × 3*	
CONV	64	7 × 7	2	114 × 114 × 64	ReLU
Max Pool		3 × 3	2	57 × 57 × 64	
CONV	192	3 × 3	1	57 × 57 × 192	ReLU
Max Pool		3 × 3	2	29 × 29 × 192	
Inception (3A)				29 × 29 × 256	
Inception (3B)				29 × 29 × 480	
Max Pool		3 × 3	2	15 × 15 × 480	
Inception (4A)				15 × 15 × 512	
Inception (4B)				15 × 15 × 512	
Inception (4C)				15 × 15 × 512	
Inception (4D)				15 × 15 × 528	
Inception (4E)				15 × 15 × 832	
Max Pool		3 × 3	2	8 × 8 × 832	
Inception (5A)				8 × 8 × 832	
Inception (5B)				8 × 8 × 1024	
Avg Pool		7 × 7	1	1 × 1 × 1024	
Dropout	rate = 0.4				
FC	2				SoftMax
Auxiliary Network 1					
Avg Pool		5 × 5		4 × 4 × 512	
CONV	128	1 × 1		4 × 4 × 128	
FC	1024			1024	ReLU
Dropout	rate = 0.7				
FC	2			2*	SoftMax
Auxiliary Network 2					
Avg Pool		5 × 5		4 × 4 × 528	
CONV	128	1 × 1		4 × 4 × 128	
FC	1024			1024	ReLU
Dropout	rate = 0.7				
FC	2			2*	SoftMax
Total number of parameters including auxiliary networks = 10,309,430					
Total number of trainable parameters including auxiliary networks = 10,309,430					

N.B.: CONV: Convolutional Layers, Max Pool: Max Pooling Layers, and FC: Fully Connected Layers.

Table 3
Summarized information of the Inception Layers.

Name	# (1 × 1) Filters	# (3 × 3) Reduce filters	# (3 × 3) Filters	# (5 × 5) Reduce filters	# (5 × 5) filters	# of filters in Max Pooled and then applied (1 × 1) convolution
Inception 3A	64	96	128	16	32	32
Inception 3B	128	128	192	32	96	64
Inception 4A	192	96	208	16	96	64
Inception 4B	160	112	224	24	64	64
Inception 4C	128	128	256	24	64	64
Inception 4D	112	144	288	32	64	64
Inception 4E	256	160	320	32	128	128
Inception 5A	256	160	320	32	128	128
Inception 5B	384	192	384	48	128	128

N.B.: CONV: Convolutional Layers, Max Pool: Max Pooling Layers, and FC: Fully Connected Layers.

parameters of the AlexNet architecture was 58,295,042 where among them 58,292,290 were trainable and the reaming were non-trainable parameter.

3.2.2. VGGNet

VGGNet proposed in Ref. [28], is an improved CNN architecture that achieved significant improvements compared to AlexNet, ZFNet etc. VGG16 (13 convolutional layers and 3 fully connected layers) and VGG19 (16 convolutional layers and 3 fully connected layers) are the two most common VGG architectures that are widely used in classification problems [29]. In VGGNet architecture, there are five blocks where each block starts with a convolutional layer and then moves on to max-pooling. Basic VGGNet architecture operates by taking input data and passing them through a convolution layer stack. In Fig. 3, a block-based diagram has been shown to understand the layers of this architecture. As previously mentioned, to conduct the experiments in this literature, the input image's dimensions were 227 × 227 × 3, where 3 denotes the number of RGB colour channels.

Also, the problem focused on classifying three classes. The scaler number attached to the convolution (CONV) layer denotes the number of consecutive layers of convolution and for each convolution layer

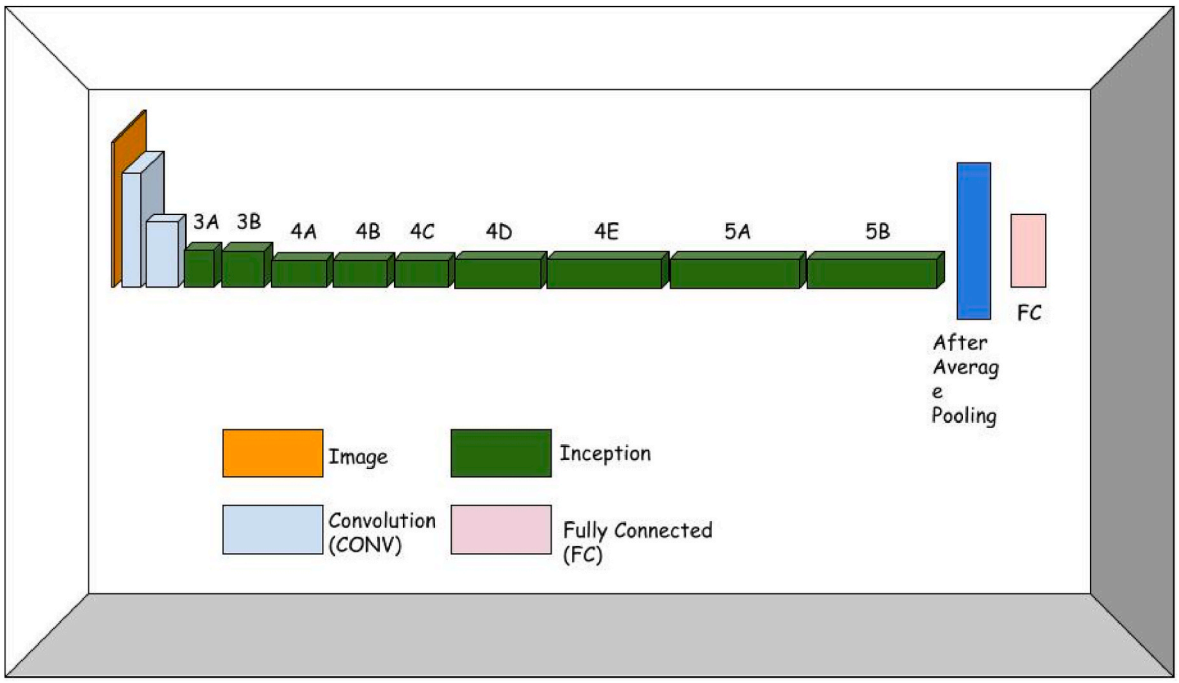

Fig. 5. Abstract view of InceptionNet architecture's main network.

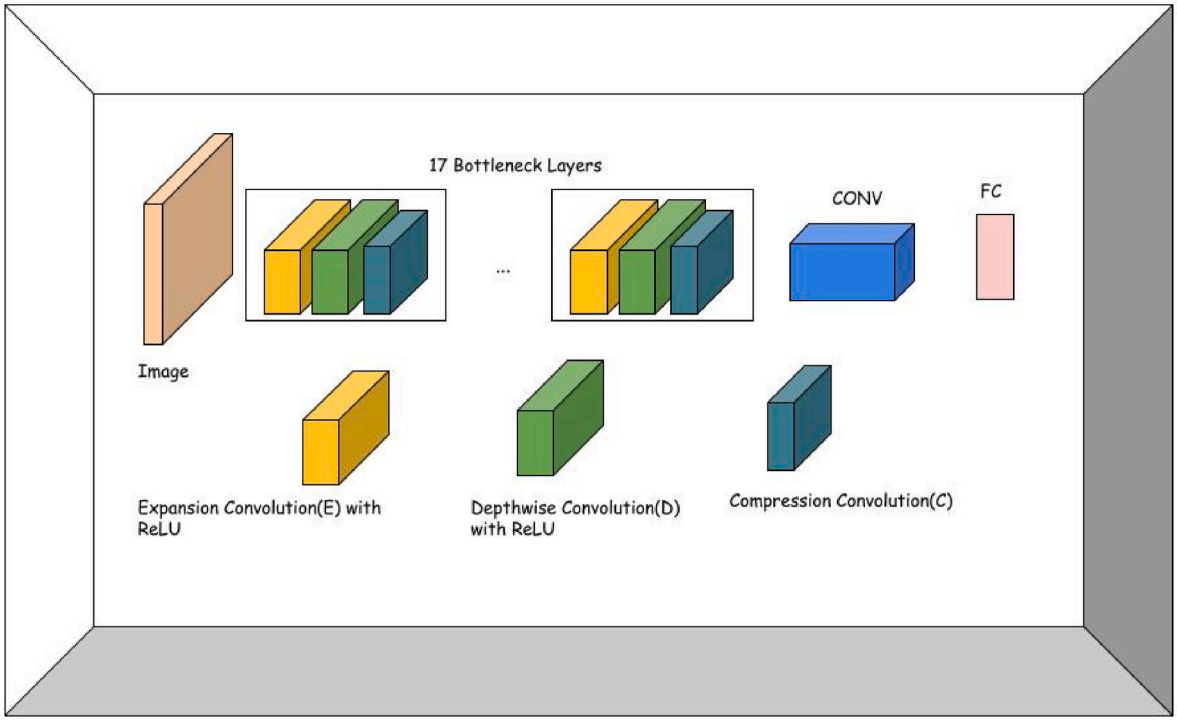

Fig. 6. An abstract pictorial representation of MobileNetV2 architecture's layers.

paddings are kept such as the input and output height and width dimension stays the same. The output of each CONV layer is passed through a ReLU activation function. The total number of parameters of the architecture is 134,268,738 among them 134,268,738 are trainable parameters.

3.2.3. *InceptionNetV1*

In this section, we will discuss the InceptionNetV1 architecture as shown in Fig. 4 that has been used to address the Phuti Karpas leaf detection problem [30]. Identifying the filter or kernel size is crucial to label the performance of CNN based architectures and often it is a difficult and experimental task for each dataset to properly find the concerned size. InceptionNet architecture or GoogleNet architecture shades light on this aspect and applies filters of multiple sizes. Its novelty lies in recognizing similar categories of images having multiple sizes, shapes, scales, and orientations of the main subject. In Tables 2 and 3, the summarized information of the convolutional layers, max pooling layers, and fully connected layers have been presented. Besides, there

Table 4
Summarized information of the feature maps of MobileNetV2 architecture.

Layer	#Filters/Neurons	Filter size	Stride	Size of feature map	Activation function
Image				227 × 227 × 3*	–
CONV	32	3 × 3	2	114 × 114 × 32	ReLU
Depthwise CONV	32	D 3 × 3		114 × 114 × 32	ReLU
Compressed CONV	16	1 × 1		114 × 114 × 16	–
Bottleneck 1	24	E 1 × 1, D 3 × 3, C 1 × 1	2	57 × 57 × 24	ReLU –
Bottleneck 2	24	E 1 × 1, D 3 × 3, C 1 × 1	1	57 × 57 × 24	ReLU –
Bottleneck 3	32	E 1 × 1, D 3 × 3, C 1 × 1	2	29 × 29 × 32	ReLU –
Bottleneck 4	32	E 1 × 1, D 3 × 3, C 1 × 1	1	29 × 29 × 32	ReLU –
Bottleneck 5	32	E 1 × 1, D 3 × 3, C 1 × 1	1	29 × 29 × 32	ReLU –
Bottleneck 6	64	E 1 × 1, D 3 × 3, C 1 × 1	2	15 × 15 × 64	ReLU –
Bottleneck 7	64	E 1 × 1, D 3 × 3, C 1 × 1	1	15 × 15 × 64	ReLU –
Bottleneck 8	64	E 1 × 1, D 3 × 3, C 1 × 1	1	15 × 15 × 64	ReLU –
Bottleneck 9	64	E 1 × 1, D 3 × 3, C 1 × 1	1	15 × 15 × 64	ReLU –
Bottleneck 10	96	E 1 × 1, D 3 × 3, C 1 × 1	1	15 × 15 × 96	ReLU –
Bottleneck 11	96	E 1 × 1, D 3 × 3, C 1 × 1	1	15 × 15 × 96	ReLU –
Bottleneck 12	96	E 1 × 1, D 3 × 3, C 1 × 1	1	15 × 15 × 96	ReLU –
Bottleneck 13	160	E 1 × 1, D 3 × 3, C 1 × 1	2	8 × 8 × 160	ReLU –
Bottleneck 14	160	E 1 × 1, D 3 × 3, C 1 × 1	1	8 × 8 × 160	ReLU –
Bottleneck 15	160	E 1 × 1, D 3 × 3, C 1 × 1	1	8 × 8 × 160	ReLU –
Bottleneck 16	320	E 1 × 1, D 3 × 3, C 1 × 1	1	8 × 8 × 320	ReLU –
CONV	1280	1 × 1		8 × 8 × 1280	
Batch Normalization				8 × 8 × 1280	
ReLU				8 × 8 × 1280	
Avg Pool				1 × 1 × 1280	
FC	2			2*	SoftMax
Total number of parameters including auxiliary networks = 2,260,546 Total number of trainable parameters including auxiliary networks = 2,226,434					

N.B.: E, D, and C are used to denote expansion, depthwise, and compressed convolution.

are some inception layers that applies some (1 × 1), (3 × 3), (5 × 5) filters and a max pooling over the input feature map separately and then concatenate the results increasing the number of channels and keeping the height and width same to the input image during returning the extracted feature map (see Fig. 4 and Table 2).

Fig. 5 presents an overview of Inception architecture. Only the major architecture's layers are presented. Inception architecture also maintains two additional very small architectures which are simultaneously trained. For simplicity, they are omitted.

InceptionNetV1 also applies two additional parallel auxiliary layers consisting of an average pooling layer (Avg Pool), a single convolutional layer, and two fully connected layers. All of these three networks are simultaneously trained, and weights are updated. These two auxiliary networks work over the outputs of inception 4A and inception 4D layers respectively.

3.2.4. MobileNetV2

MobileNetV2 is another very known architecture as shown in Fig. 6 that is very lightweight and suits well for mobile devices where it requires to solve extensive mathematical calculations [31]. This architecture maintains a bottleneck layer that applies inverted residuals technique which works on three steps. Firstly, it applies an expansion layer or convolution (E) to uncompress the data, then a depthwise layer or depthwise convolution (D) to filter the data and finally a projection layer or compression layer (C) to compress the data. Also, they apply the idea of skip connection to propagate residue among deep layers. To expand, (1 × 1) filter is used in the two-dimensional convolutional layer along with ReLU as the activation mechanism. In the depthwise layer, depthwise 2D convolution is used which is composed of (3 × 3) filters. Depthwise convolution is different from the traditional two-dimensional convolutional layers where the prior one applies different filters for each channel and then merges them. The projection layer applies traditional two-dimensional convolution consisting of (1 × 1) filters. Both depthwise CONV and Expansion CONV applies ReLU following Batch Normalization after two-dimensional convolution. Compressed CONV does not apply any activation and directly forwards the data after applying convolution (Table 4).

Each bottleneck layer consists of three modules, expansion convolution with ReLU, depthwise convolution with ReLU, and normal two-dimensional convolution without ReLU. Also, a skip connection or simple value concatenation is performed between input and output if the number of channels (3rd dimension) matches.

3.2.5. Baseline architecture

In this study, we have also tested a custom baseline architecture that is comparatively much simpler in design than the rest of the well-known architectures. The main goal was to understand how large-scale deep architectures perform in solving the Phuti Karpas leaf recognition

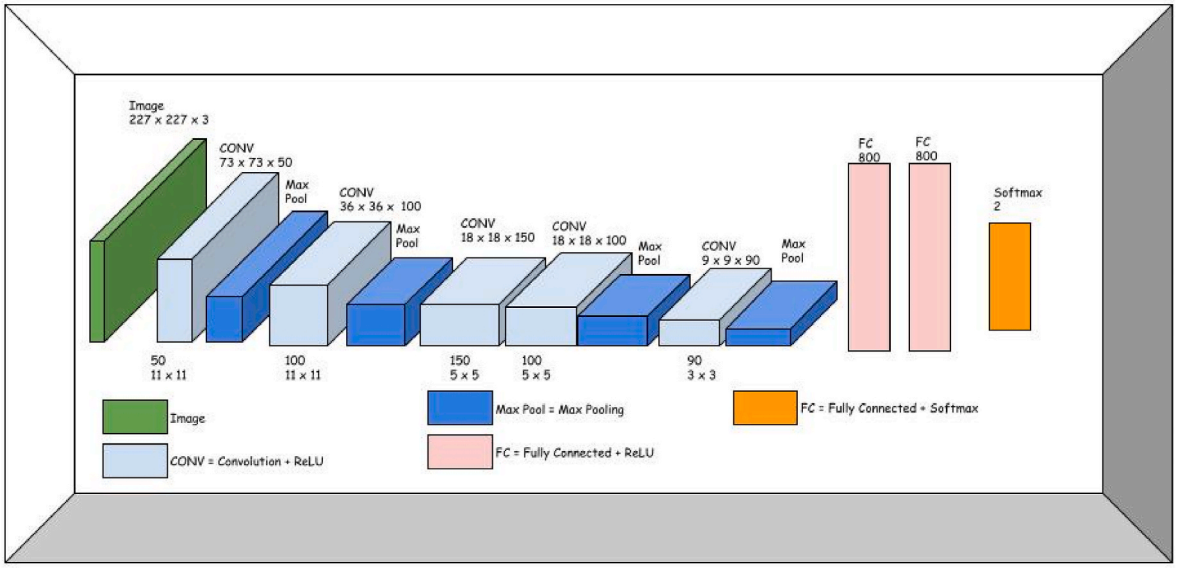

Fig. 7. Visualization of the layers of the custom baseline architecture.

problem than a very simple architecture. This approach helps to understand both the challenges and novelties that come with using deep large-scale architectures.

Our developed custom architecture is inspired by the AlexNet architecture. In the similar fashion it first gradually increases the number of channels in the feature maps with the combination of two convolution (CONV) and max pooling (Max Pool) blocks and a single convolution block. Then using similar types of two CONV-Max Pool blocks it starts decreasing the number of channels in the feature map. Then using two fully connected layers and a single SoftMax layer it generates the class predictions for an input image of ($227 \times 227 \times 3$) dimension. After each CONV layer, a batch normalization layer is added to regularize the feature maps. The novelty of this architecture is that it is much lighter in weight compared to the AlexNet in the number of parameters. In Fig. 7, we have shown a visualization of our baseline architecture. For each convolutional block we have shown the number of filters, the kernel size and the output feature maps. Each convolutional block uses ReLU activation function. Also, the output of convolutional block is passed through a batch normalization layer before passing thorough a max pooling layer. Dropout normalization of 0.5 is also applied over the output of fully connected layers. The total number of parameters in this architecture is 3,251,802.

In this section, we presented a discussion regarding the wide known image classification architectures and our proposed one custom baseline architecture. Classification of Phuti Karpas plants is crucial due to not having enough amount of trainable data. In the subsequent sections, we present our findings in terms of evaluating performance of different architectures in properly classifying Phuti Karpas plants. We believe, our contribution will pave a great path in identifying Phuti Karpas plants in nature under different circumstances and will be helpful in reviving this

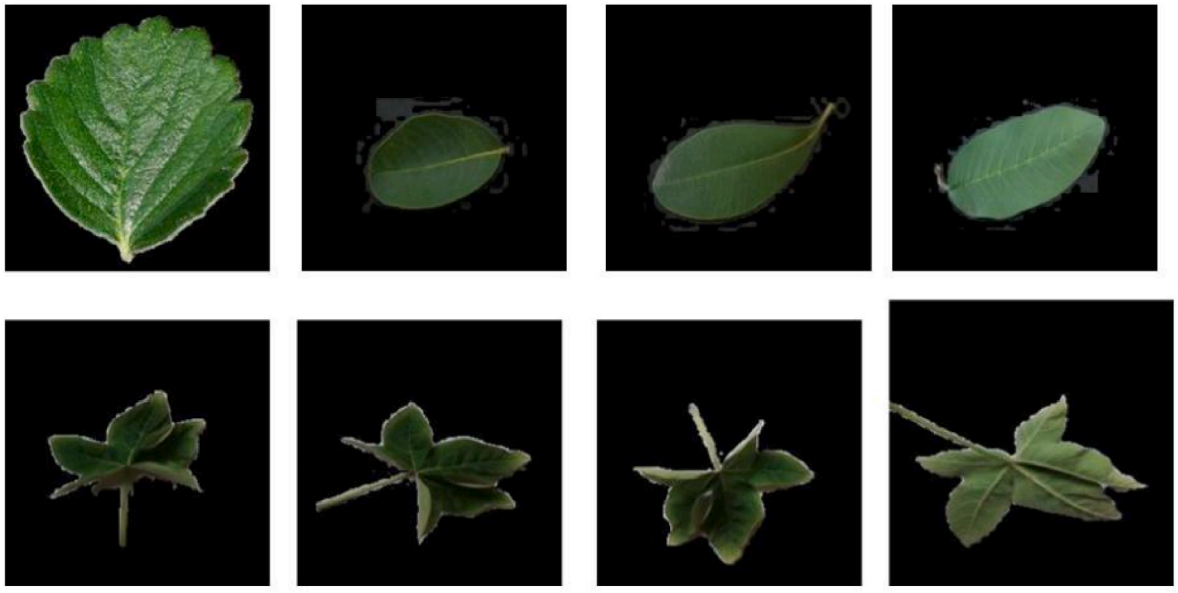

Fig. 8. Snapshot of the custom dataset used in this study. This study categorizes the instances within two classes – Phuti Karpas and Non-Phuti Karpas. In the image, the upper and lower row represent the non-Phuti and Phuti-karpas instances. In the upper row, the first image is of Strawberry, second image is of Alstonia Scholaris, third image is of Arjun and fourth image is of Basil. Amalgamation of all such images (13 classes each having 91 instances) are considered under a single class denoting Non-Phuti Karpas.

Table 5
Information about Batch size, Epoch, and training time.

Rank	Name	Average training time (ms) per batch (CPU)	Average training time (ms) per batch (GPU)	Average training time (ms) per epoch (CPU)	Average training time (ms) per epoch (GPU)
2nd	AlexNet	66	0.04	23000	8
4th	Inception	246	0.35	11000	70
5th	VGG16	2000	2	298000	380
3rd	MobileNetV2	216	0.3	41000	58
1st	Baseline	26	0.02	6000	4

endangered entity.

4. Dataset description, implementation, and performance evaluation

The primary goal of this study was to investigate the performance of CNN-based architectures to correctly recognize Phuti Karpas Plants. This study has tested five architectures in this regard and reported their results. The architectures were AlexNet, IncpetionNetV1(Inception), VGG16, MobileNetV2 and a simple custom developed baseline architecture. All the architectures were investigated in detail to see how they perform in recognizing the problem of Phuti Karpas plants.

This work has used a new dataset to address the problem. In the dataset, there was no separation between the training and testing data. Therefore, cross-validation has been used to analyze the performance of the architectures based on different metrics. All the experiments were conducted in a 64-bit machine having AMD Ryzen 9 5900x 12 Core processor × 24, Nvidia RTX 3090 GPU, 128 GB RAM and Ubuntu 20.04 LTS operating system. The architectures were implemented in python language using Keras, a deep learning API, and TensorFlow, an end-to-end open-source platform for machine learning [32].

4.1. Dataset collection and pre-processing

In the dataset, there were in total of 2354 instances, where 1171 images were of Phuti Karpas plants' and the remaining 1183 were of other different healthy leaves such as Alstonia Scholaris, Arjun, Basil, Catharanthus Roseus, Chinar, Gauva, Jamun, Jatropha, Lemon, Mango, Pomegranate, Pongamia Pinnata, and Strawberry. Except from Catharanthus Roseus and Strawberry leaves, all were collected from Kaggle [33]. Phuti Karpas plants' images were collected manually using a smartphone camera in a normal light condition found between 9 and 11 a.m. in the month of March. In total, 91 samples were collected from each type of leaves for the aforementioned 13 categories and compiled into one major category under Non-Phuti Karpas plants [33], provided a dataset of both healthy and diseased plants. Only the healthy plants were randomly chosen for each category. Images were collected using a smartphone device's camera. Catharanthus Roseus and Strawberry leaves were also manually collected and curated by us through capturing images using a smartphone device.

The main challenge of this study was to design a CNN-based architecture that would properly recognize the Phuti Karpas plants among the others. Therefore, there were two class labels that have been considered, a Phuti Karpas plant and a non-Phuti Karpas plant. Some images from the dataset have been shown in Fig. 8. The upper row shows images of non-Phuti Karpas instances and the bottom row shows various images of the Phuti karpas plants. Through pre-processing all the images were scaled having dimensions of (227 × 227 × 3) where 3 denotes the number of color channels in RGB sample space. Also, through pre-processing the backgrounds of images were uniformed to color black using built in python modules OpenCV.

Table 6
Information about Loss function, Optimizer, Learning Rate, and Early stopping callback used during training across different architectures.

Loss function	Categorical Cross Entropy
Optimizer	Stochastic Gradient Descent (SGD)
Learning Rate	For each architecture, the learning rate was 0.001 except for the Inception architecture. In Inception architecture, progressive learning rate was used where the initial learning rate was 0.01 and dropped as the power of 0.96 over the interval of 8 epochs. The rates were set based on general practice for each architecture.
Early stopping	Monitor metric = validation loss patience = 20

4.2. Performance evaluation and discussion

In this section, all the experimented architectures are evaluated on various metrics and aspects. The following subsections will contain a brief description of each of the addressed factors. The architectures are implemented in Python and using the Keras deep learning library. The experiments are conducted in a Google Colab environment with a 16 GB RAM and under both GPU and CPU processing capabilities to understand the associated processing factors.

In Tables 5 and 6, information related to the training setup has been presented. For each architecture, we have fixed a maximum of 100 epochs to run, and the batch size was fixed as 10. We have also used early stopping criteria where the monitoring metric and patience has been provided in Table 6.

Our training procedure was conducted in GPU. But we also experimented in the CPU environment so that we can get an estimation about the performance in low powered devices. In the GPU environment, all the architectures can perform very fast and so their differences are very insignificant. But in the CPU environment, it is found that the baseline architecture works fastest among the other architectures. The main reasoning is the number of numeric computations is comparatively lesser in this architecture than the others. Though the total number of parameters is the least in MobileNetV2, it contains a significant number of layers making it a very deep architecture which leads to a good number of numerical calculations during processing. The other architectures' timing variation is also representative of their number of layers and the parameters to train.

4.2.1. Accuracy

In this section, we will discuss the accuracy observed across different cross-validation datasets for each of the architectures. The summarized results have been presented in Table 7. A pictorial representation of the table's data has also been shown in Fig. 9.

From this experiment, we have observed that there is not any single architecture that always works best for each cross-validation set. But based on the average accuracy value, AlexNet has been found giving the best performance whereas MobileNetV2 is very close and falls behind by a little margin. The baseline architecture provides the third best performance exceeding the known architecture InceptionNet. InceptionNet and VGG16 lie in the fourth and fifth position respectively. Now, if we observe the head-to-head comparison to the best performance in each fold, we might summarize it as follows.

- MobileNetV2 has provided the best accuracy performance in six folds.
- AlexNet has provided the best accuracy performance in five folds.
- Baseline architecture has provided the best accuracy performance in three folds.
- Inception has provided the best accuracy performance in two folds.
- In the head-to-head comparison, VGG16 has never provided accuracy performance equivalent to the best in any fold.

Table 7
The Accuracy observed in each cross-validation fold during training across different architectures over the testing dataset. In each fold lies 80 % training data, 10 % testing data, and 10 % validation data.

Fold	AlexNet (%)	Inception (%)	VGG16 (%)	MobileNetV2 (%)	Baseline (%)
1	98.31	97.88	97.81	100[a]	98.31
2	99.15	99.58[a]	98	99.58[a]	98.73
3	99.15[a]	98.73	96.61	96.19	98.73
4	99.58[a]	98.73	98.73	98.73	99.58[a]
5	100[a]	98.31	99.58	100[a]	100[a]
6	99.15[a]	97.88	97.88	99.15[a]	99.15[a]
7	98.73	99.58[a]	98.73	99.58[a]	98.31
8	99.58	99.58	97.88	100[a]	99.15
9	98.31	98.31	98.31	99.58[a]	98.31
10	99.58[a]	99.15	99.58	97.88	98.73
Average	99.15[a]	98.77	98.31	99.07	98.9
Rank Summary	1st	4th	5th	2nd	3rd

[a] Denotes the best performance observed in that particular fold.

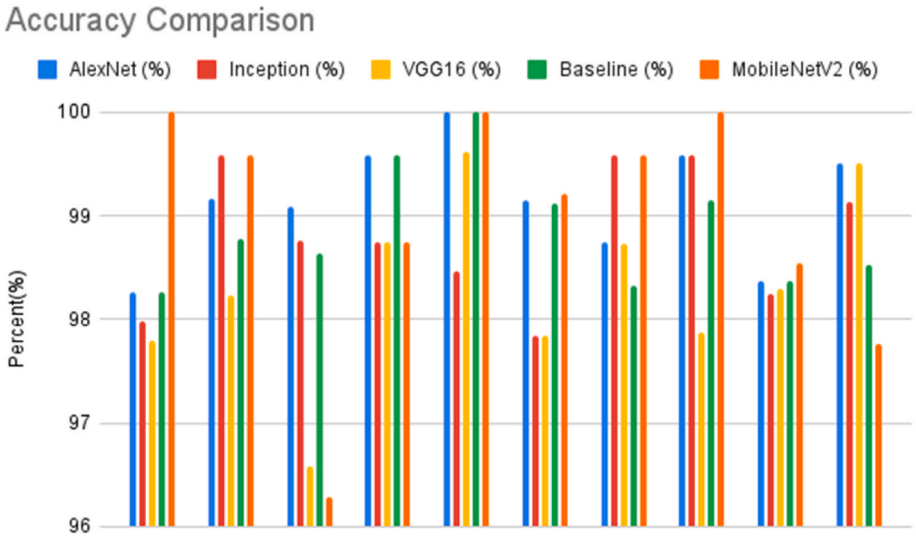

Fig. 9. Bar Chart Representation of the observed **accuracies** of each architecture in each fold.

Table 8
Precision values observed for each architecture across different cross-validation folds. (*) Operator denotes the best performance observed.

Fold	AlexNet (%)	Inception (%)	VGG16 (%)	MobileNetV2 (%)	Baseline (%)
1	98.26	97.98	97.8	100*	98.26
2	99.17	99.58*	98.23	99.58*	98.77
3	99.08*	98.76	96.58	96.28	98.64
4	99.58*	98.74	98.74	98.74	99.58*
5	100*	98.47	99.61	100*	100*
6	99.15	97.85	97.84	99.21*	99.12
7	98.74	99.58*	98.73	99.58*	98.33
8	99.59	99.59	97.87	100*	99.15
9	98.37	98.25	98.3	98.55*	98.37
10	99.5*	99.13	99.5	97.76	98.53
Average	99.14*	98.79	98.32	98.97	98.88
Rank Summary	1st	4th	5th	2nd	3rd

Therefore, it can be said that based on the average fold accuracy, AlexNet provides comparatively better performance than MobileNetV2, but in the head-to-head comparison, MobileNetV2 has given better performance. MobileNetV2 falls behind because in some of the epochs, the accuracy has reduced significantly whereas AlexNet has never fallen very behind in accuracy.

4.2.2. *Precision and recall*

This section will summarize the performance status observed in precision and recall of all the tested architectures across different cross-validation folds. Precision denotes the ratio of the number of true positives with respect to all the positive instances observed for a class and recall denotes the ratio of the positive instances over all the expected instances for a particular class. Similar to the discussion regarding accuracy, results observed in precision represent a similar performance pattern that is summarized in Table 8. Similar to previous discussion, a pictorial representation has also been presented in Fig. 10.

From Table 8, we can see that, there is not any single architecture that has performed best in each and every fold. However, we can get an overall recapitulated idea if we consider the average statistics across all

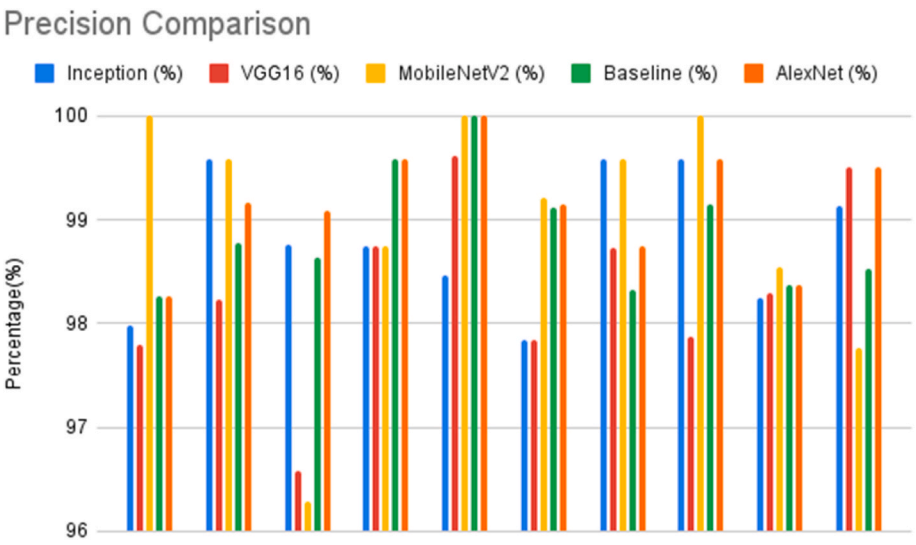

Fig. 10. Bar Chart Representation of the observed **precision scores** of each architecture in each fold.

Table 9
Recall values observed for each architecture across different cross-validation folds. (*) Operator denotes the best performance observed.

Fold	AlexNet (%)	Inception (%)	VGG16 (%)	MobileNetV2 (%)	Baseline (%)
1	98.35	97.78	97	100	98.35
2	99.15	99.57	98.31	99.58	98.72
3	99.22	98.68	96.58	96.03	98.84
4	99.58	98.73	99	98.73	99.58
5	100	98.17	99.54	100	100
6	99.15	97.8	97.95	99.1	99.2
7	98.73	99.57	98.73	99.57	98.3
8	99.57	99.57	97.89	100	99.17
9	98.24	98.41	98.3	99.6	98.24
10	99.64	99.13	99.64	97.83	98.91
Average	99.16**	98.74	98.29	99.04	98.93
Rank Summary	1st	4th	5th	2nd	3rd

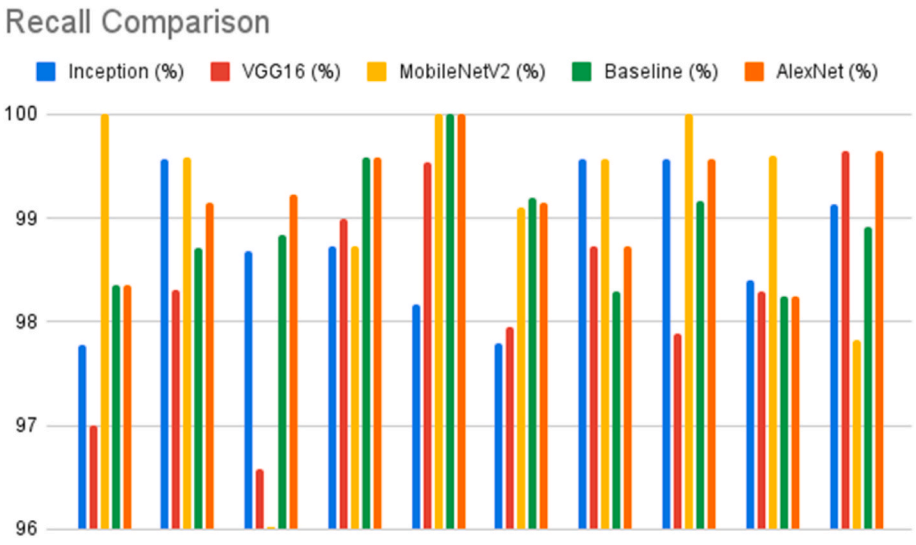

Fig. 11. Bar Chart Representation of the observed **recall scores** of each architecture in each fold.

the folds. Here AlexNet provides the best average performance observed across different epochs as well. Second best performance has been observed from MobileNetV2. MobileNetV2 demonstrated impressive performance in some folds, but due to exhibiting poor performance in some folds (E.g., Fold 3, Fold 10, etc.) the overall performance deteriorated. Similar to previous discussion, though our proposed customized lightweight baseline architecture did not exhibit the best performance but it provided a very competitive performance across each fold. Most noticeably, in Fold 3, where even MobileNetV2 degraded its performance radically, our custom architecture possessed an overall

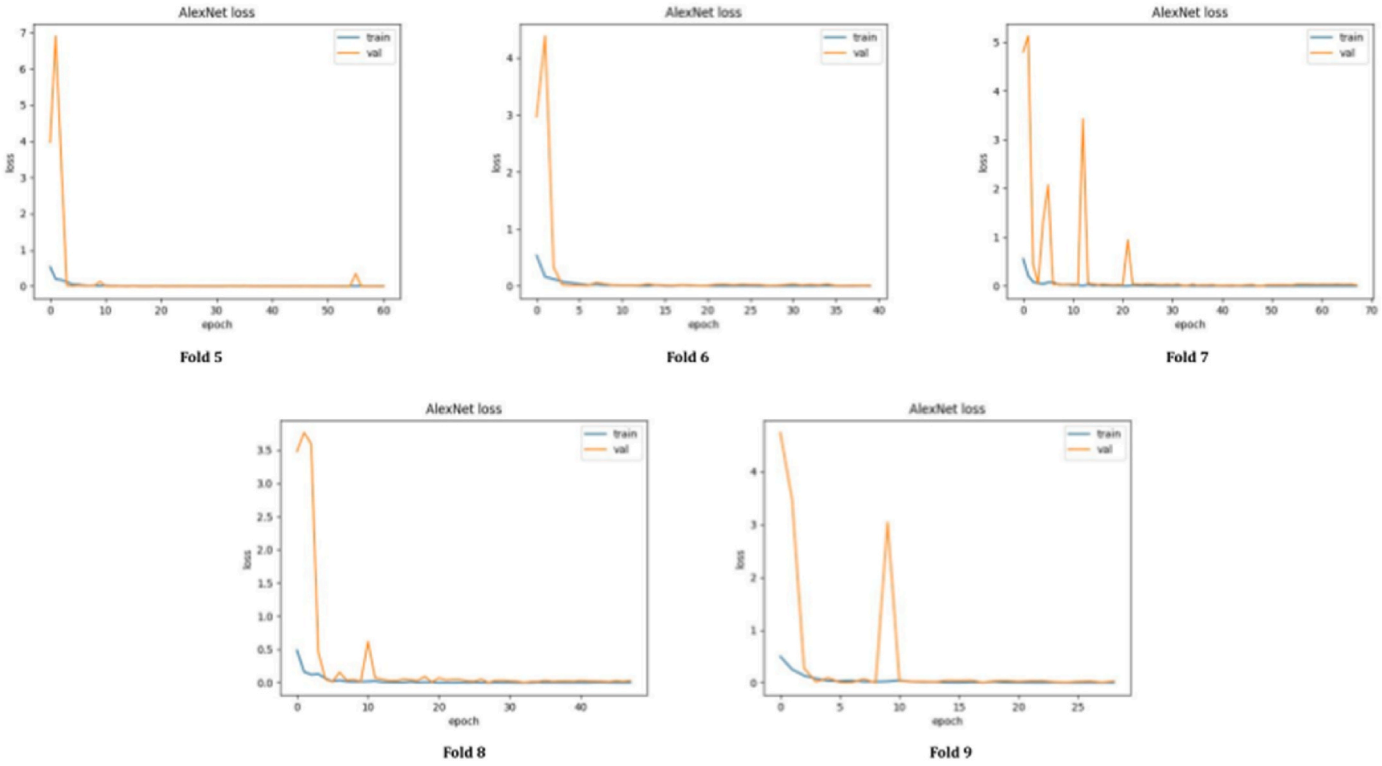

Fig. 12. Loss variation observed during training of **AlexNet** for fold 5, 6, 7, 8, and 9, respectively from left to right.

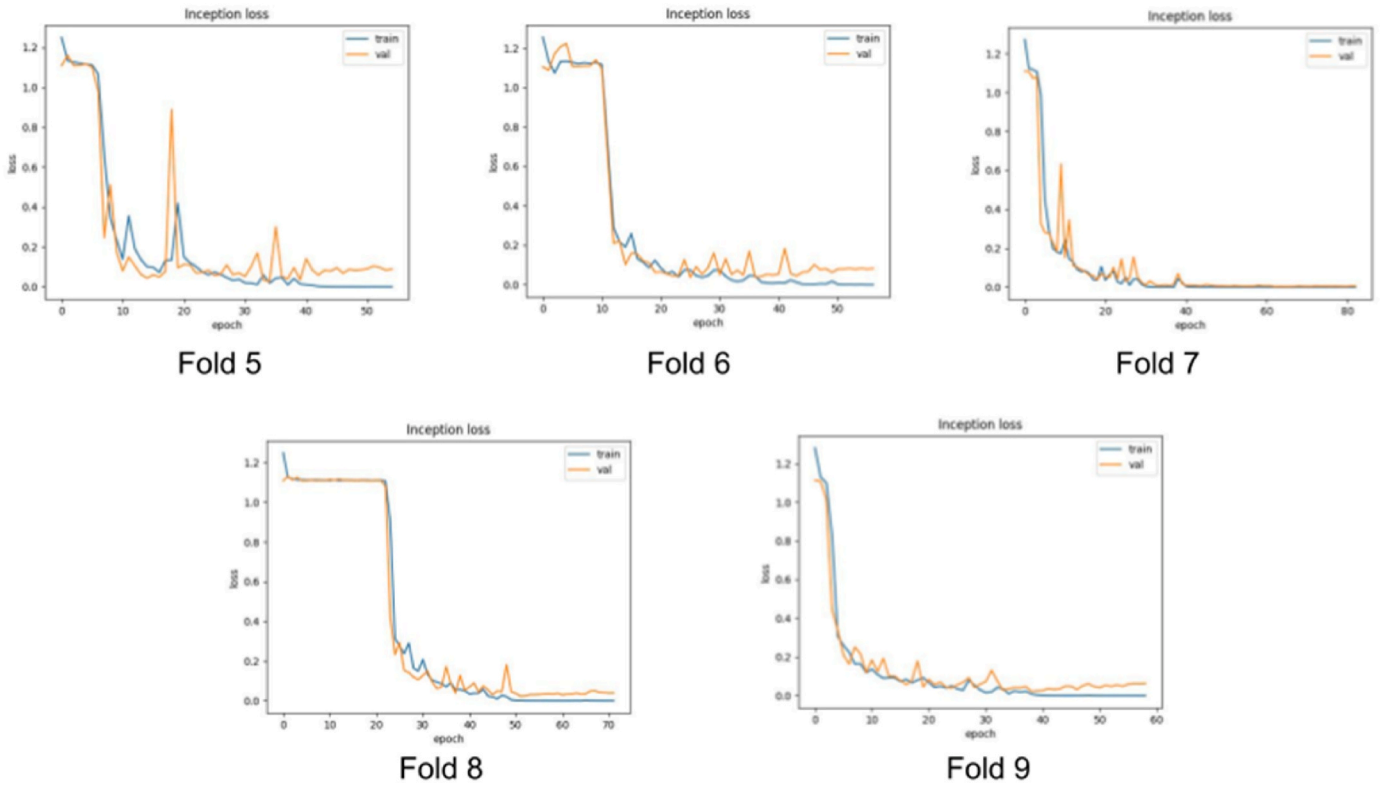

Fig. 13. Loss variation observed during training of **Inception** for fold 5, 6, 7, 8, and 9, respectively from left to right.

constant performance. In terms of the average statistics, Inception architecture comes at the fourth place. It also showed a good performance, however average result deteriorated due to exhibit poor performance across fold 6. VGG16 ranks in the fifth position demonstrating a weaker performance almost across all the folds.

We have also shown the results that have been observed from the recall across different folds. From Table 9, a summarized status regarding this can be seen. The results represent the pattern already

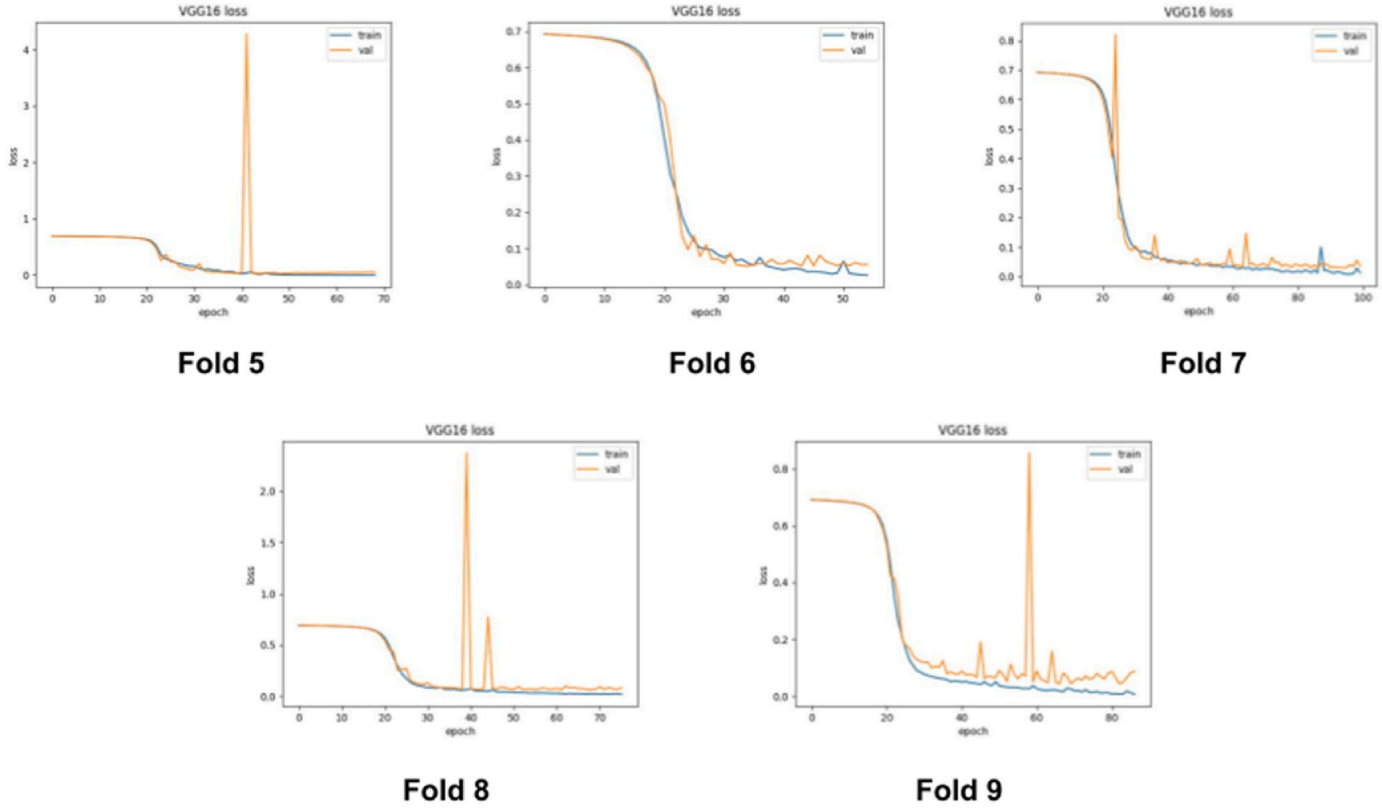

Fig. 14. Loss variation observed during training of **VGG16** for fold 5, 6, 7, 8, and 9, respectively from left to right.

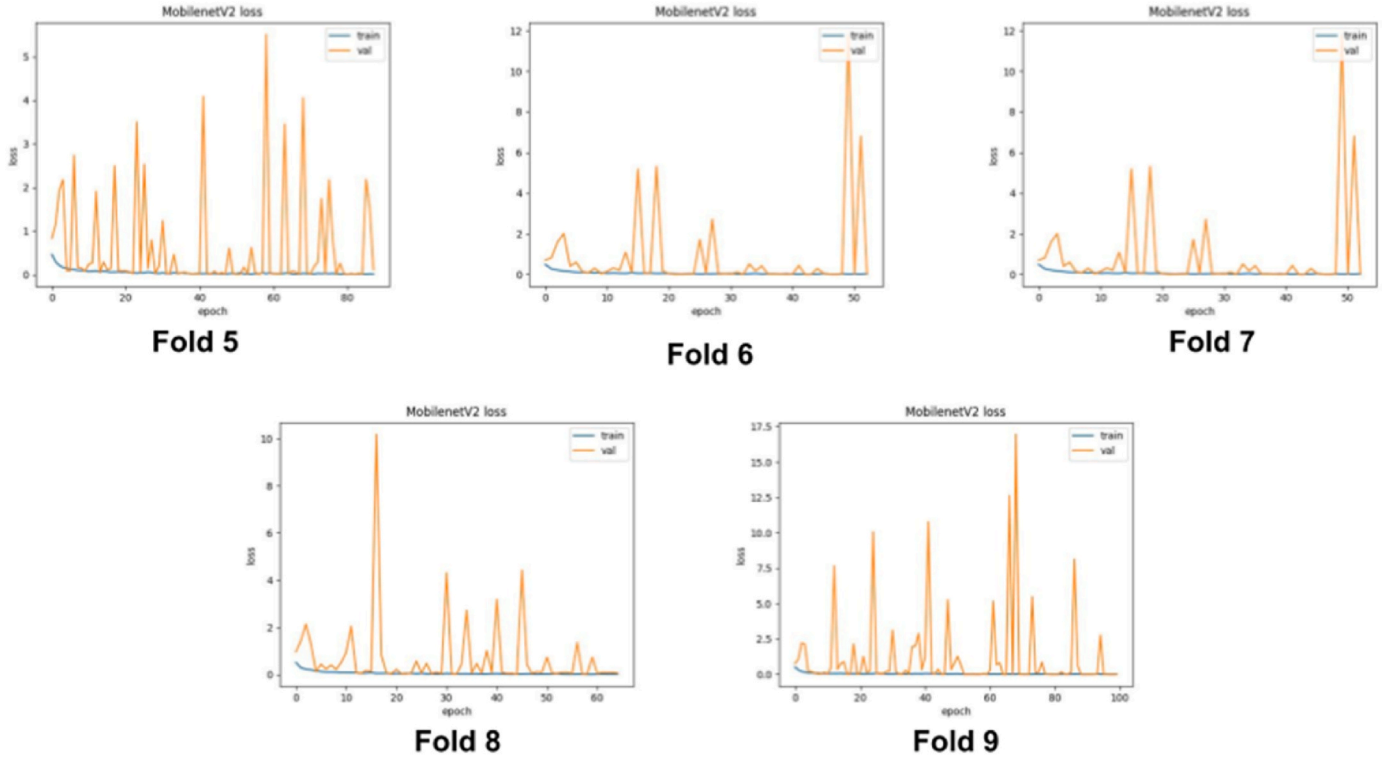

Fig. 15. Loss variation observed during training of **MobileNetV2** for fold 5, 6, 7, 8, and 9, respectively from left to right.

observed and discussed prior to the accuracy and precision where AlexNet provides the best performance and second to it is found from MobileNetV2. A pictorial representation has also been shown in Fig. 11. Similar to the previous discussion, we can see that, the performance of all the architectures is quite competitive. Though, in terms of average statistics, AlexNet holds the first position, MobileNetV2 exhibits

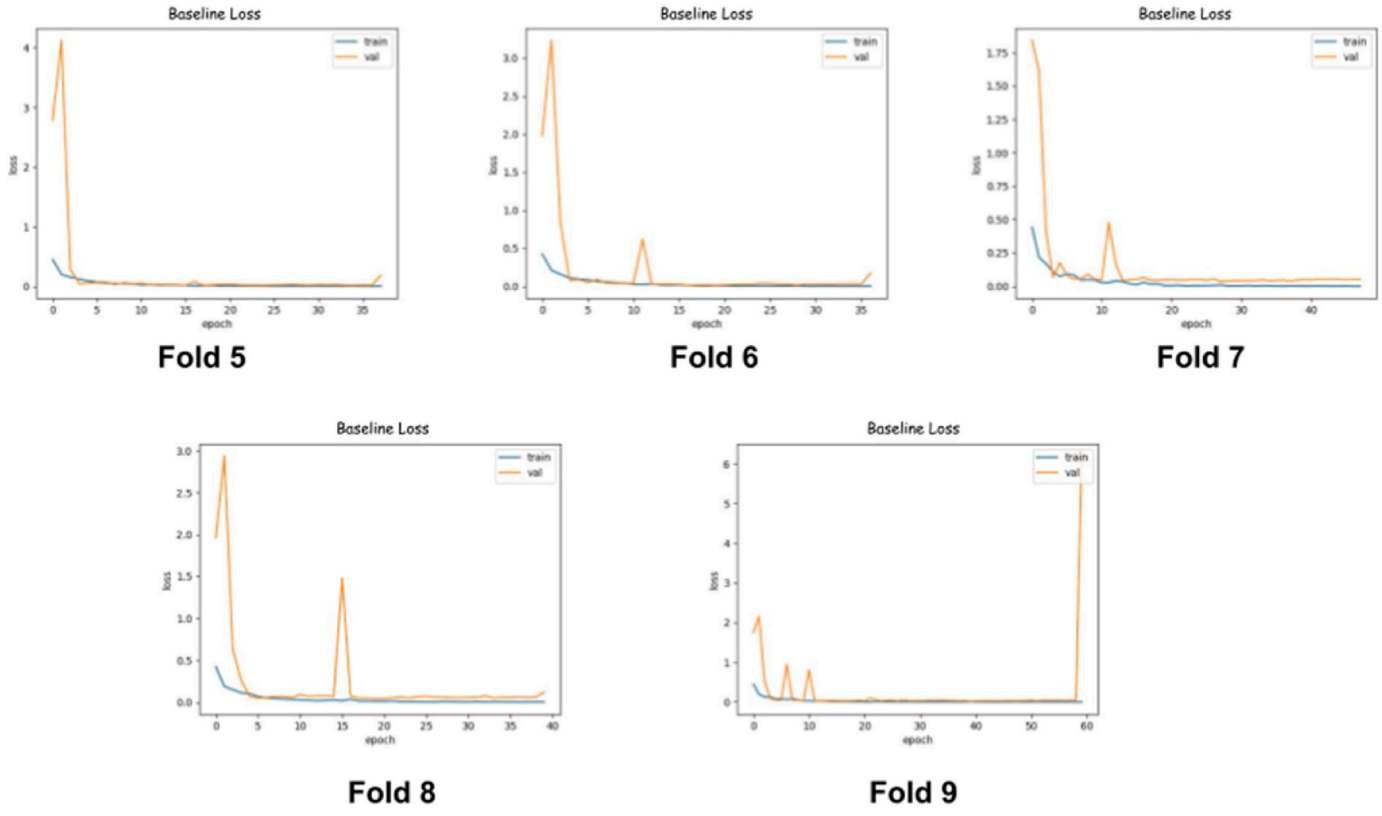

Fig. 16. Loss variation observed during training of **Custom Baseline** for fold 5, 6, 7, 8, and 9, respectively from left to right.

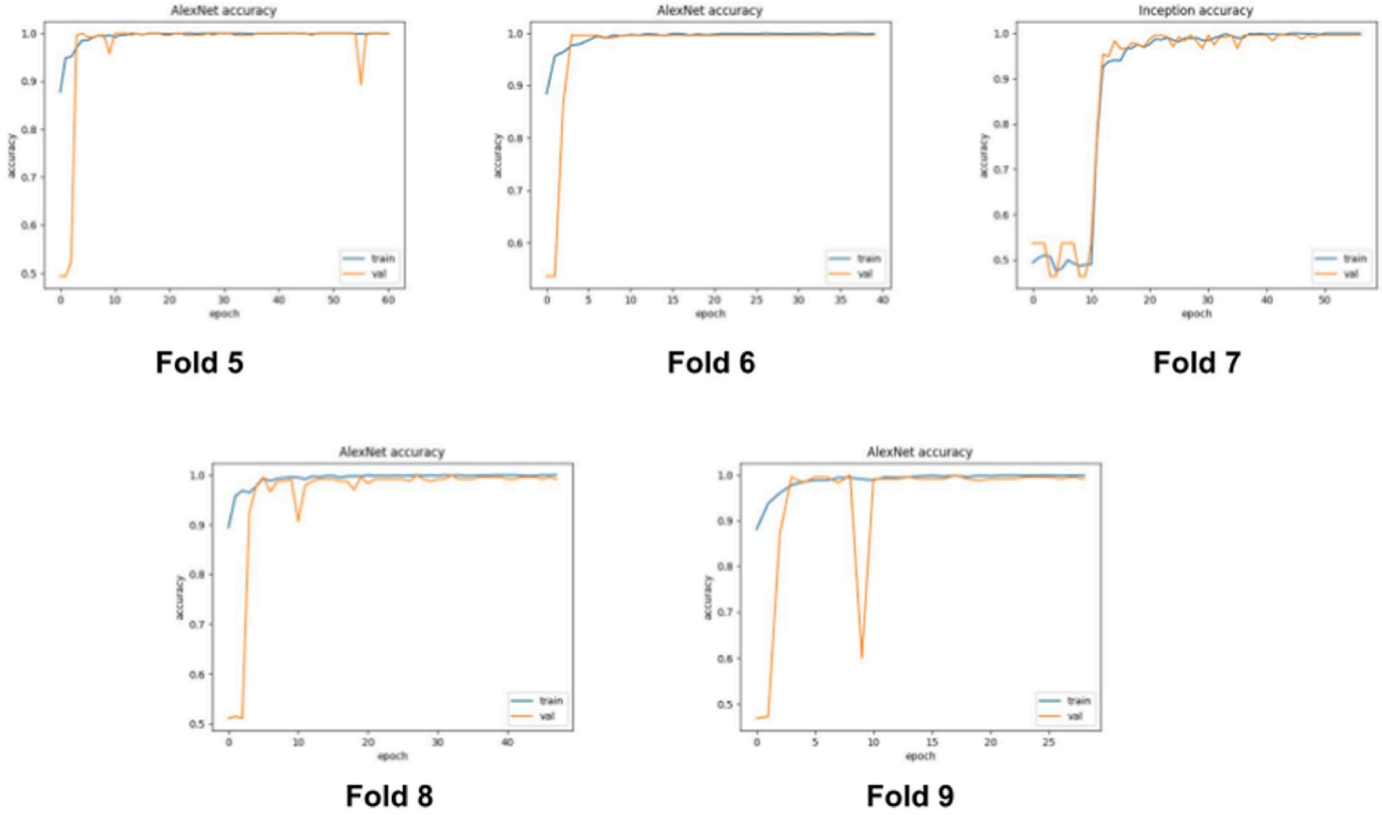

Fig. 17. Accuracy variation observed during training of **AlexNet** for fold 5, 6, 7, 8, and 9, respectively from left to right.

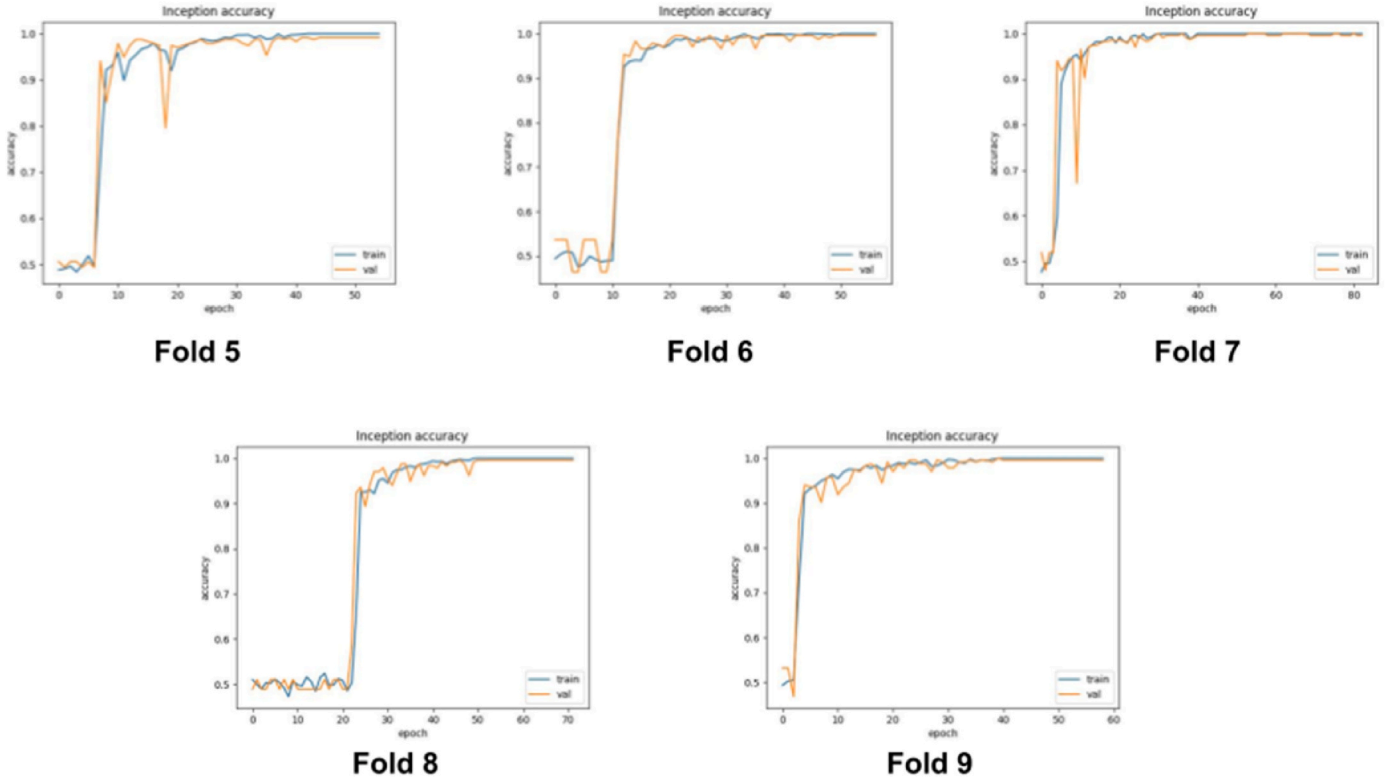

Fig. 18. Accuracy variation observed during training of **Inception** for fold 5, 6, 7, 8, and 9, respectively from left to right.

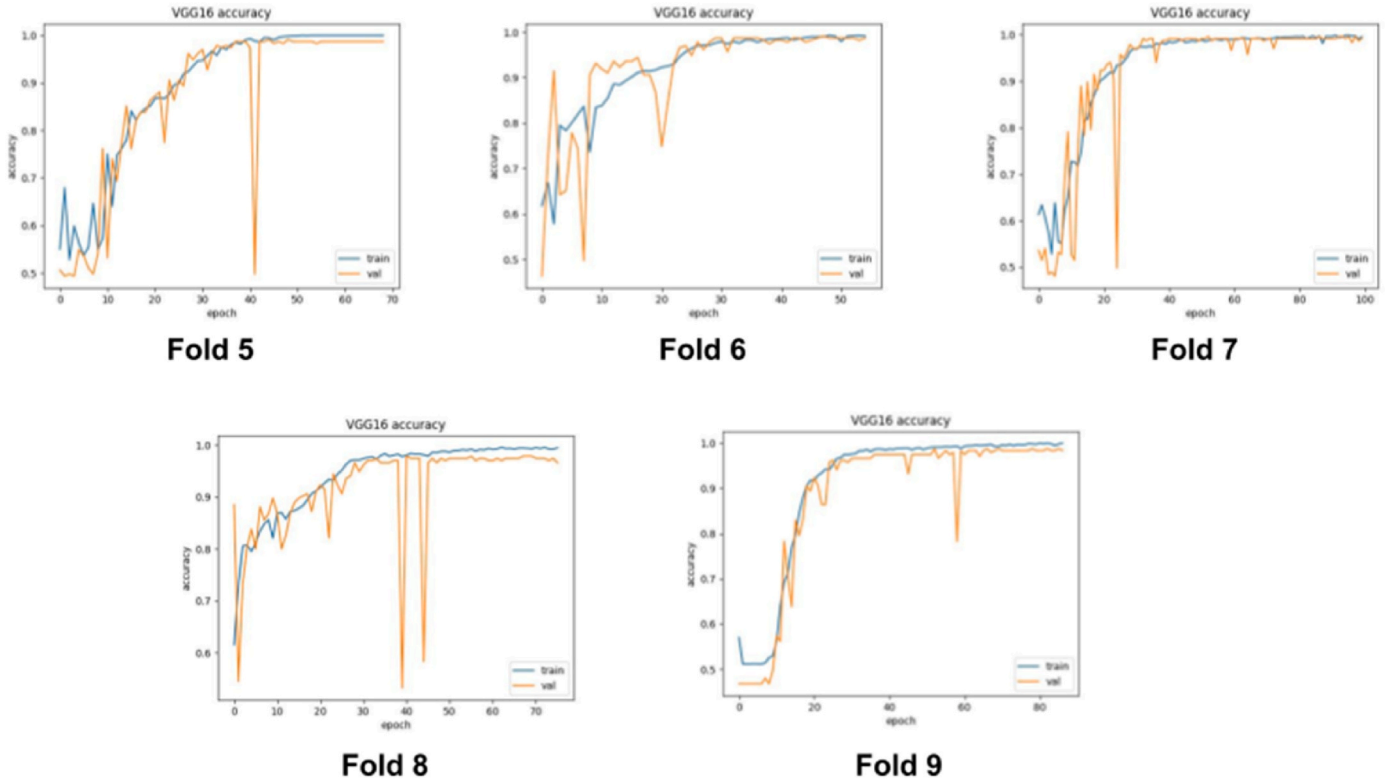

Fig. 19. Accuracy variation observed during training of **VGG16** for fold 5, 6, 7, 8, and 9, respectively from left to right.

impressive performance in some folds but just like the prior discussion due to performing worse in some folds the average value deteriorated. Our custom baseline architecture holds a constant performance across all the folds in terms of recall values also. Inception architecture holds its place in the fourth position. It also presented a competitive performance as already discussed but falls a bit behind in folds such as – 1st, 5th and

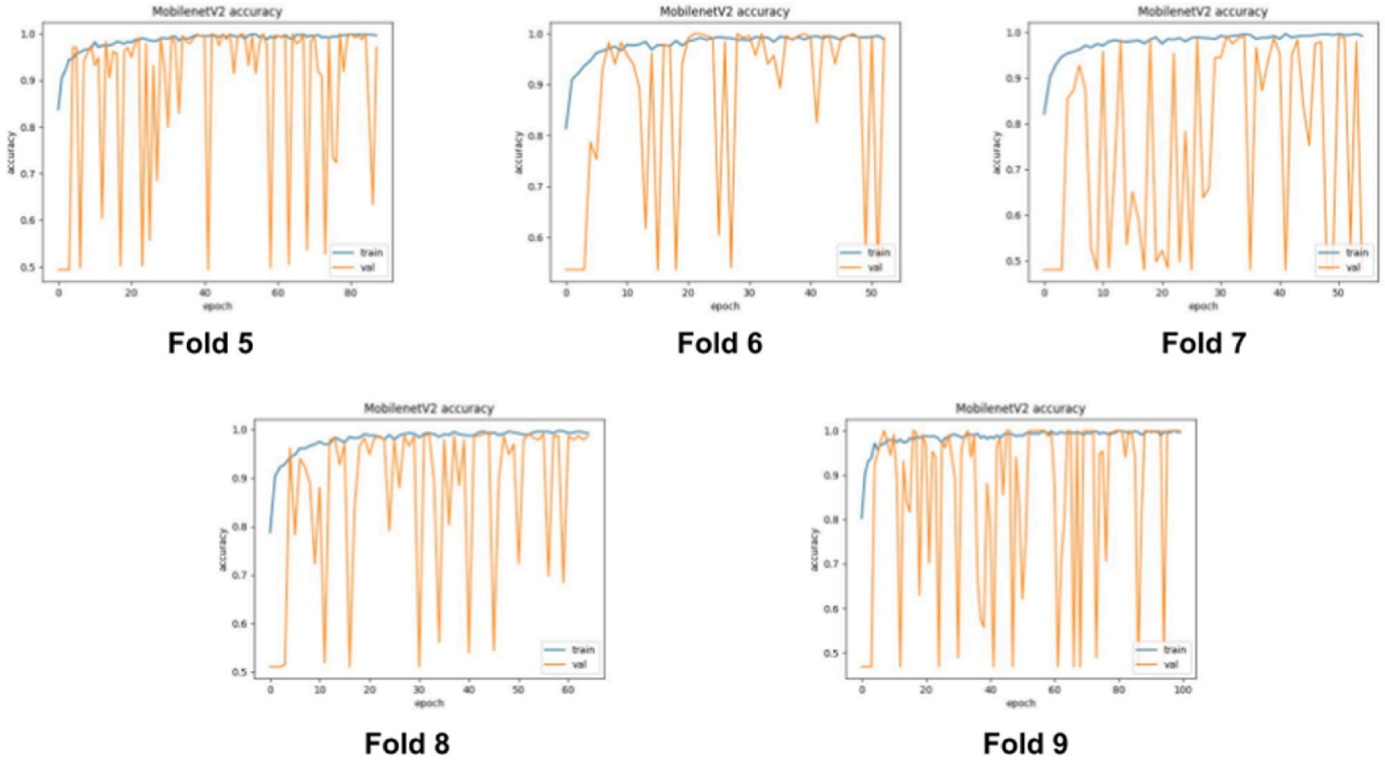

Fig. 20. Accuracy variation observed during training of **MobileNetV2** for fold 5, 6, 7, 8, and 9, respectively from left to right.

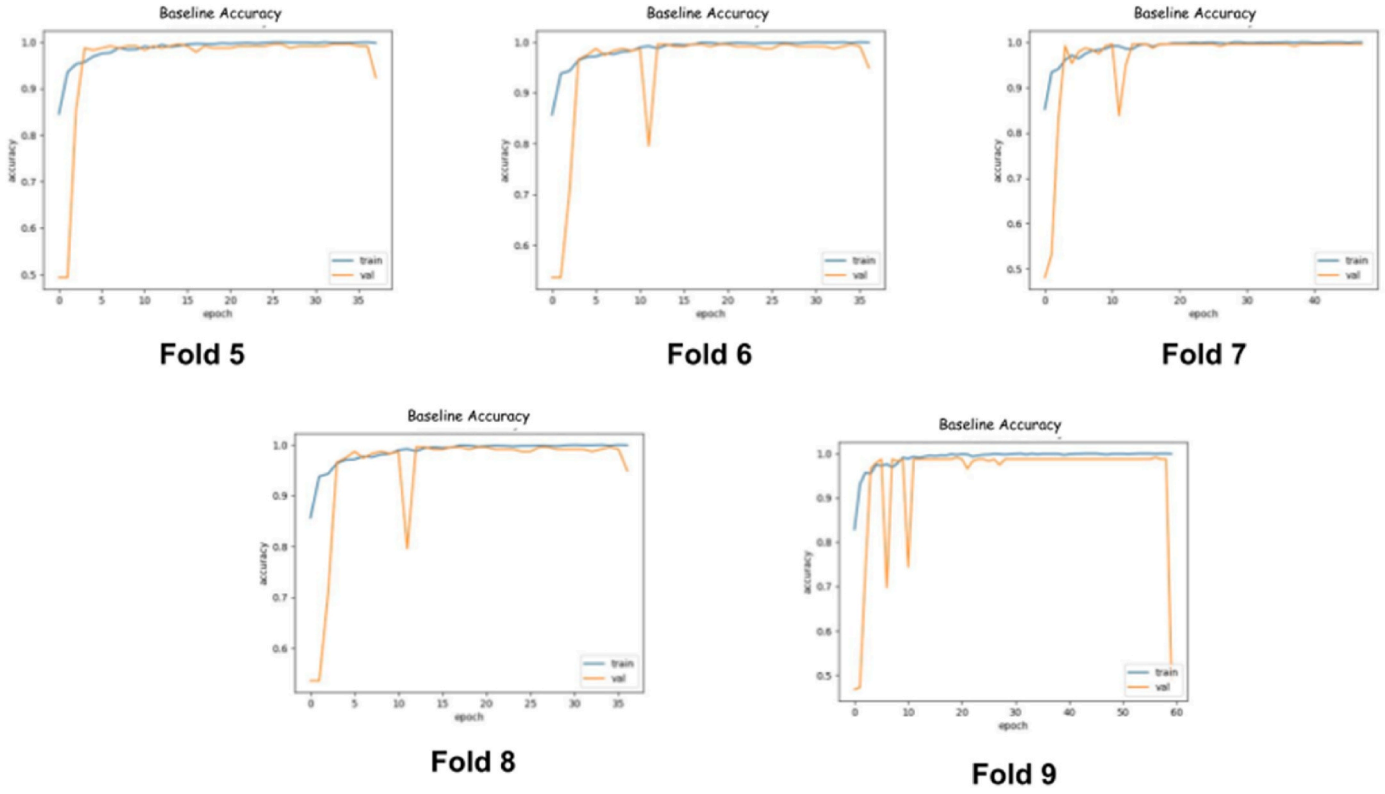

Fig. 21. Accuracy variation observed during training of **Custom Baseline** for fold 5, 6, 7, 8, and 9, respectively from left to right.

6th. VGG16 showed a weaker performance in almost all the folds except the 10th fold.

4.2.3. Loss and accuracy variation during training

In this sub-section, we will observe the training status to understand how the loss and accuracy are varied over epochs in different folds. Figs. 12-16 and Figs. 17-21 illustrates loss and accuracy variations

Table 10
Trained Model's size of different architectures found in hdf5 (Hierarchical data format version 5).

Rank	Name	Size (Megabytes)	Total number of parameters
4th	AlexNet	466.4	58,295,042
3rd	Inception	41.5	10,309,430
5th	VGG16	537.1	134,268,738
1st	MobileNetV2	9.4	2,260,546
2nd	Baseline	13.1	3,251,802

during training of AlexNet, Inception, VGG16, MobileNetV2, and baseline, respectively. This will help us to understand the architecture's learning behaviour along with the convergence pattern. For ease of understanding, we have provided the results for five folds for each of the architectures. The patterns were found similar in the remaining folds.

The change in the training loss was found stable in all the architectures. Main distinguishing pattern was observed in the validation loss. During training of AlexNet, we have seen that the loss variation to a very minimum level. It converged very early and stayed stable during the remaining epochs. Only in epoch 7, there were some abrupt changes. The change in loss was also found mostly stable in the Inception architecture. A bit of instability was observed only in fold 5 where some sudden spike is found. VGG16 provided mainly two types of loss curves. One is seen in fold 5, 8 and 9 where it mostly stays stable with some intermediary spikes. The second type can be observed in fold 6 and 7 where it suddenly reduces loss and stays mostly in that way. The loss changing behaviour of MobileNetV2 was very unstable, it showed a significant number of spikes through the epochs in all the folds. The behaviour of baseline architecture was very similar to the AlexNet mostly because the prior one was designed being inspired by it. It was found very stable during the learning, but a sudden shift can only be observed only in fold 9 in later epochs.

Similar to the previous discussion, we shall mainly focus on the accuracy over the validation data or unseen data. The change in accuracy over the training data was seen stable in all the folds for all the architectures over the epochs. The accuracy variation of AlexNet was found stable. There were a small number of intermediary spikes (fold 7 and 9), but mostly the change was found stable. The change in accuracy of Inception architecture over epochs in different folds was found very stable overall. VGG16's accuracy variation was found unstable mostly in the earlier epochs with significant abrupt changes. But with gradual progression it started to converge and remained overall stable at the end of the epochs. This behaviour was found overall in the folds. MobileNetV2's behaviour was found very chaotic with a lot of abrupt shifts in the accuracy which is similar to its corresponding abrupt shifting in the loss in the same folds. A possible reason can be the number of data was

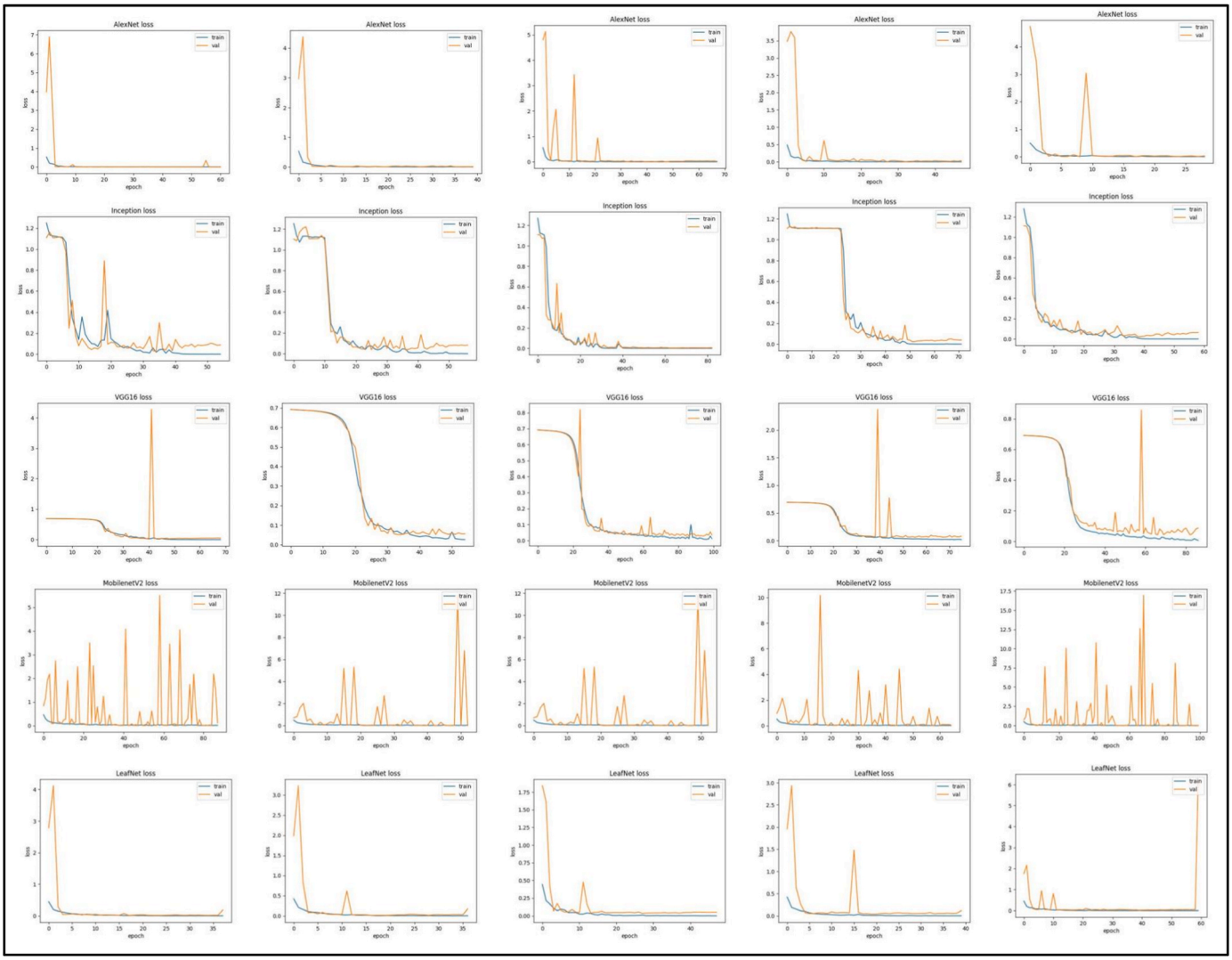

Fig. 22. Loss variation during training of AlexNet, Inception, VGG16, MobileNetV2, and baseline respectively. Each row represents charts for each architecture respectively. For fold 5, 6, 7, 8, and 9 results have been shown from left to right order in each row.

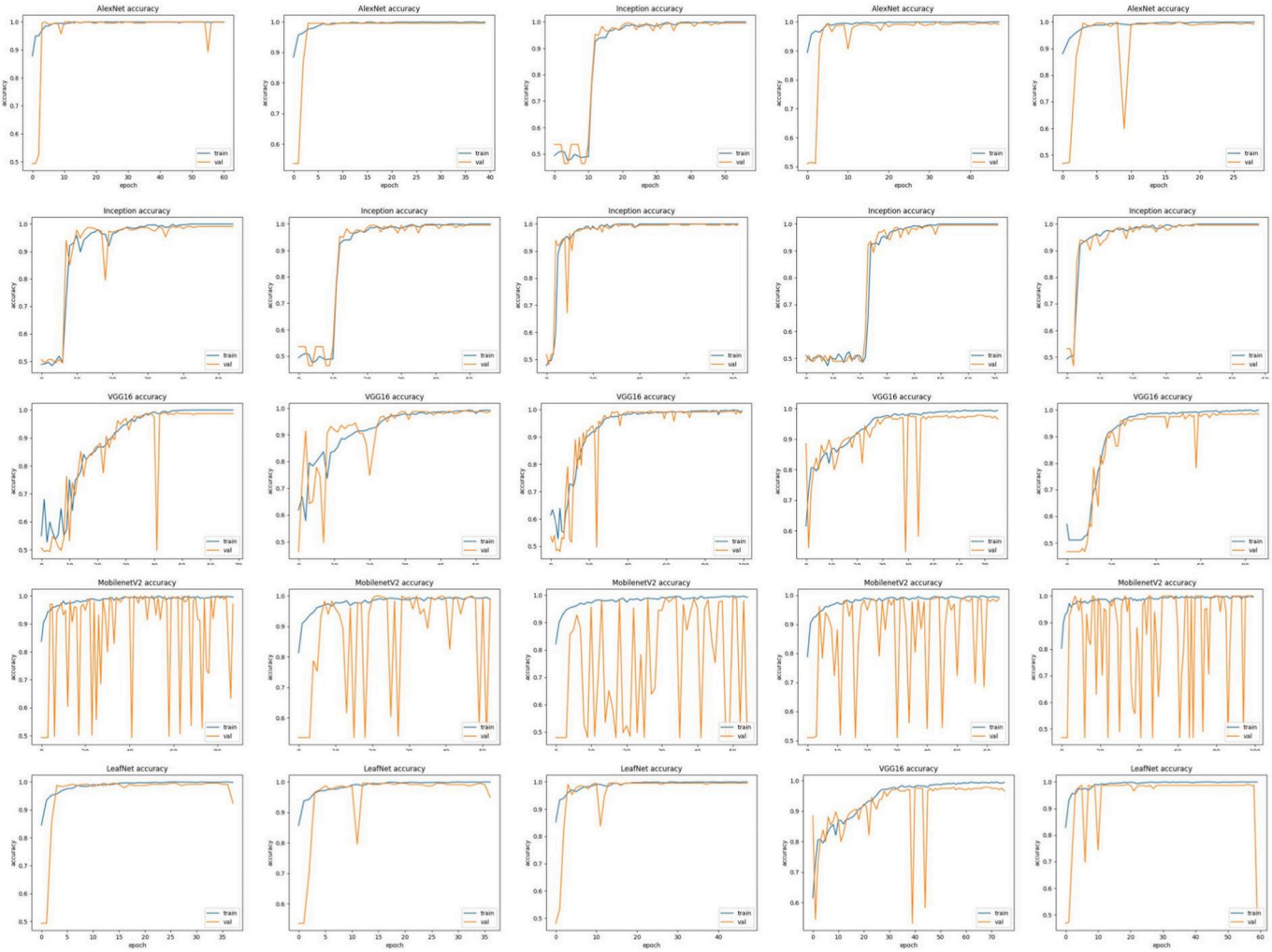

Fig. 23. Accuracy variation during training of AlexNet, Inception, VGG16, MobileNetV2, and baseline. Each row represents charts for their corresponding architecture. Results for fold 5–9 have been shown from left to right order in each row.

not enough to train this architecture in a stable manner. Though overall, the accuracy achieved by MobileNetV2 was found pretty impressive and very close to the best accuracy observed in this problem.

Baseline architecture's accuracy convergence behaviour was found very similar to the AlexNet, mostly being stable. Only in the fold 10 some abrupt changes can be observed in the later epochs which should be related to the change in the loss as discussed.

4.2.4. Model parameters and size

In this section, we shall provide the size of the trained weight files for each of the architectures to understand the required memory usage. From Table 10, we can get an idea about the required model size to store a trained model in the application device for each architecture. It is found that, VGG16 requires the most and MobileNetV2 requires the least amount of memory. This number is directly proportional to the number of architecture's parameters.

4.3. Deployment as an android application

To test the proposed model, an Android application is also designed with Android Studio using a desktop computer with AMD Ryzen5 3600 (4 GHz), Windows 10 Pro (64-bit) operating system, 16.0 GB RAM, GPU (Dedicated 6 GB NVIDIA GeForce GTX1660ti), and 1 TB Solid State Driver. The app is finally tested in Xiaomi Mi A1 based on stock Android 9. A snapshot of the proposed architecture is shown in Fig. 24.

So, in summary, the identification of the Phuti Karpas tree presents significant challenges due to the complexity and diversity of plant species. Traditional identification methods relying on visual inspections are inadequate, as they fail to account for variations influenced by environmental factors and seasonal changes. This study leverages advanced computational techniques through Convolutional Neural Networks (CNNs) to enhance the automatic classification of the Phuti Karpas tree. Various architectures, including AlexNet, Inception, VGG16, and MobileNetV2, were implemented and evaluated using a custom dataset of leaf images. Experiments were conducted in a Google Colab environment with both CPU and GPU processing, measuring key performance metrics such as accuracy, precision, and recall across multiple cross-validation folds.

Our findings reveal that AlexNet achieved the highest average accuracy, while MobileNetV2 displayed competitive efficiency with a significantly smaller model size. The baseline architecture, despite its fast training times, did not match the performance of AlexNet or MobileNetV2. The study underscores the effectiveness of deep learning approaches in plant species classification, providing valuable insights into feature extraction and model performance. Furthermore, deploying the trained model as an Android application enables real-time identification of the Phuti Karpas plant, offering practical implications for agricultural practices and conservation efforts. Overall, this research addresses critical challenges in automatic plant identification, demonstrating how advanced machine learning techniques can improve

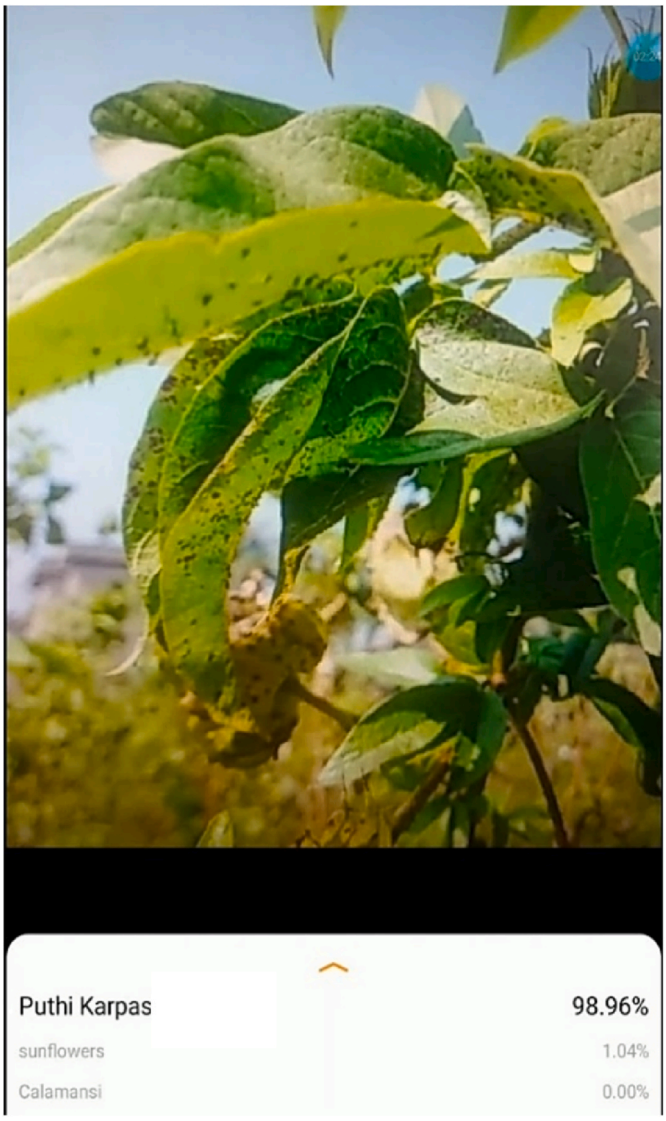

Fig. 24. A snapshot of the developed Android application to recognize a Phuti Karpas plant in real-time.

accuracy and efficiency in plant taxonomy.

5. Discussion and summary

This section presents a summary of all the architectures that have been tested in this study based on different metrics along with their pros and cons. Table 11 summarizes the pros and cons of all the architectures already been addressed. Based on the points, the users would be able to decide which architecture would be useful for their application and device that are going to be used.

6. Limitation and future directions

The core novelty of this work is providing a curated collection of data for Phuti Karpas plants and the evaluation of widely used CNN-based architectures in detecting the aforementioned plant from a wide range of samples of different plants. Through a robust evaluation, we have presented the limitations and constrained for all the architectures in terms of accuracy, precision, recall, learning pattern and the underlying challenges while deploying as a mobile application.

This work bears a number of limitations that can be addressed in the later iterations of this work. For example, a crucial factor is the limitation of dataset. Though the resource is quite scarce but a wide incorporation of more data could be useful in improving the performance of the currently used deep architectures. As our work is a pioneering direction in formulating a deep-learning based approach in detecting Phuti Karpas we have used a set of widely acknowledged deep learning based architectures for the evaluation and a custom lightweight baseline architecture. In the follow-up extension of this work, it would be a great opportunity to explore different architectures as well including hyper tunning the parameters or developing a grid-search alike parameter finding algorithm in this context. Experimenting with

Table 11
Summarized discussion regarding the pros and cons of all the tested architectures.

	Pros	Cons
AlexNet	• AlexNet has provided the best average accuracy in the phuti karpas leaf recognition problem. Based on precision and recall it also stands in the first position among others. • Its loss and accuracy changing behaviour was found mostly very stable. • Its processing time was also found very moderate.	• The major con of this architecture is its required size to store the model's weights. It is heavy weighted and ranks fourth based on lightness in weight among the architectures.
Inception	• This architecture ranks third among the architectures based on lightness in storing the weights. • This architecture provides stable change in training in loss and accuracy variation.	• This architecture has ranked fourth in the average accuracy, precision and recall over all the architectures with a difference of 0.38 %, 0.37 % and 0.42 % from the AlexNet in the metrics respectively. • This architecture also takes a good amount of time during training and ranks fourth though the difference is not much significant compared to the MobileNetV2.
VGG16	• This architecture provides stable performance during training while loss and accuracy variation.	• This architecture ranks fifth in the average accuracy, precision and recall over all the architectures having a difference of 0.84 %, 0.82 % and 0.87 % from AlexNet in the metrics respectively. • It is the most weighted architecture among all the architectures tested. • It also takes the most amount of time during training.
MobileNetV2	• MobileNetV2 has provided the second best performance in accuracy, precision and recall for our addressed problem. The difference is very insignificant and close to the AlexNet. • A major advantage of this architecture is it is very lightweight and ranks first among all the architectures that have been tested based on the lightness in weight.	• As this architecture is very deep, the processing time to receive the labelling verdict for an input image is high. • Its change in loss and accuracy during training was also found very unstable.
Baseline	• This architecture's processing speed is the fastest among all the tested architectures. • It is also the second lightest in weight among all the architectures. • Its learning curve for loss and accuracy during training was found mostly stable with a very few abrupt changes.	• It ranks third in the average accuracy, precision and recall metric among the architectures with a difference of 0.16 %, 0.26 % and 0.23 % respectively from AlexNet.

ensembled methods or transfer-learning based approaches can also be a potential direction for the extension of this work. Preprocessing is also an important part of the current version of this work which is a bit challenging in real-life settings, more generalized dataset or approaches can be investigated in this regard the make the solutions more robust.

7. Conclusion

Research studies on plant recognition and classification have grown exponentially. It has become an operational priority to monitor the growth of trees, production of fruits, disease, and pest control for plant management. In the past few years, research on plant recognition and identification using the Machine Learning and Image classification methods has increased significantly. Even though prevalent methods such as object-based image analysis and UAV imagery methods can work effectively on plant and tree recognition, Machine Learning methods such as CNN can deal effectively with large datasets providing sufficient accuracy for recognition and classification problems. In this study, different existing Convolutional Neural Network architectures along with a new architecture have been used to address the Phuti Karpas plant detection problem. For this study, a raw dataset was collected consisting of several images of the Phuti Karpas plant for training. Also, a custom dataset was prepared containing various Non-Phuti karpas instances for a fairground comparison. Full dataset was pre-processed so that they contain similar size and similar type of background. Then the experiment was conducted using the architectures, AlexNet, Inception, VGG16, MobilNetV2, and baseline over different metrics. AlexNet has provided the best average accuracy, precision and recall score in 10-fold cross validation though it poses some disadvantages such as being extremely heavy weighted architecture. MobileNetV2 has provided the second-best performance in the accuracy, precision and recall metrics and falls behind for a very small margin. It is a very light weighted architecture though its processing time is quite significant and possesses severe instability during training observed over the validation dataset. baseline architecture, inspired by AlexNet, performs neither best nor worst and provides a moderate performance in all the metrics. Inception and VGG16 have been found to perform worse comparatively than the others due to being very deep and heavy parameterized respectively. This study has tried to investigate all the important metrics so that it can guide the users about their usage considering their resource requirements and constraints. As an ongoing work, the future extensions will focus on evaluating other architectures along with focusing on identifying other types of Karpas plants in a robust manner.

CRediT authorship contribution statement

Redwan Ahmed Rizvee: Visualization, Resources, Project administration, Methodology, Investigation, Data curation, Conceptualization. **Omar Farrok:** Resources, Project administration, Investigation, Data curation. **Mahamudul Hasan:** Writing – review & editing, Writing – original draft, Validation, Supervision, Project administration, Methodology, Data curation, Conceptualization. **Faisal Farhan:** Visualization, Validation, Conceptualization. **Md Hafanul Islam:** Writing – original draft, Visualization, Validation, Supervision, Resources. **Md Khalid Hasan:** Writing – review & editing, Writing – original draft, Supervision. **Abidur Rahman:** Validation, Supervision, Software. **Maheen Islam:** Writing – original draft, Supervision, Software. **Md Sawkat Ali:** Supervision, Software, Project administration, Conceptualization. **Taskeed Jabid:** Software, Project administration, Conceptualization. **Mohammad Rifat Ahmmad Rashid:** Writing – original draft, Visualization, Supervision, Conceptualization. **Mohammad Manzurul Islam:** Data curation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that has been used is confidential.

References

[1] Saleem G, Akhtar M, Ahmed N, Qureshi WS. Automated analysis of visual leaf shape features for plant classification. Comput Electron Agric Feb. 2019;157:270–80. https://doi.org/10.1016/j.compag.2018.12.038.

[2] Anubha Pearline S, Sathiesh Kumar V, Harini S. A study on plant recognition using conventional image processing and deep learning approaches. J Intell Fuzzy Syst 2019;36(3):1997–2004. https://doi.org/10.3233/JIFS-169911.

[3] Dyrmann M, Karstoft H, Midtiby H. Plant species classification using deep convolutional neural network. Biosyst Eng Sep. 2016;151:72–80. https://doi.org/10.1016/j.biosystemseng.2016.08.024.

[4] Mehdipour Ghazi M, Yanikoglu B, Aptoula E. Plant identification using deep neural networks via optimization of transfer learning parameters. Neurocomputing 2017;235(Sep). https://doi.org/10.1016/j.neucom.2017.01.018.

[5] Haralick RM, Dinstein I, Shanmugam K. Textural features for image classification. IEEE Transactions on Systems, Man and Cybernetics 1973;SMC-3(6):610–21. https://doi.org/10.1109/TSMC.1973.4309314.

[6] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE computer society conference on computer vision and pattern recognition, vol 2016; Dec. 2016. p. 770–8. https://doi.org/10.1109/CVPR.2016.90. December.

[7] Zhu Y, et al. TA-CNN: two-way attention models in deep convolutional neural network for plant recognition. Neurocomputing Nov. 2019;365:191–200. https://doi.org/10.1016/j.neucom.2019.07.016.

[8] Neto JC, Meyer GE, Jones DD, Samal AK. Plant species identification using Elliptic Fourier leaf shape analysis. Comput Electron Agric 2006;50(2):121–34. https://doi.org/10.1016/J.COMPAG.2005.09.004.

[9] Ahmed Nisar, Khan MUG, Asif S. (2) (PDF) an automatic leaf based plant identification system. Science International-Lahore. 2016;28. https://www.researchgate.net/publication/299183724_An_Automatic_Leaf_Based_Plant_Identification_System. [Accessed 19 September 2021].

[10] Rashad MZ, el-Desouky BS, Khawasik MS. Plants images classification based on textural features using combined classifier. Int J Comput Sci Inf Technol Aug. 2011;3(4):93–100. https://doi.org/10.5121/IJCSIT.2011.3407.

[11] Olsen A, Han S, Calvert B, Ridd P, Kenny O. In situ leaf classification using histograms of oriented gradients. In: 2015 international conference on digital image computing: techniques and applications, DICTA 2015; 2015. https://doi.org/10.1109/DICTA.2015.7371274.

[12] Charters J, Wang Z, Chi Z, Tsoi AC, Feng DD. EAGLE: a novel descriptor for identifying plant species using leaf lamina vascular features. In: 2014 IEEE international conference on multimedia and expo workshops, ICMEW 2014; Sep. 2014. https://doi.org/10.1109/ICMEW.2014.6890557.

[13] Larese MG, Namías R, Craviotto RM, Arango MR, Gallo C, Granitto PM. Automatic classification of legumes using leaf vein image features. Pattern Recogn Jan. 2014;47(1):158–68. https://doi.org/10.1016/J.PATCOG.2013.06.012.

[14] Sünderhauf N, McCool C, Upcroft B, Perez T. Fine-grained plant classification using convolutional neural networks for feature extraction. In: Working notes of CLEF 2014 conference; 2014.

[15] Wäldchen J, Mäder P. Plant species identification using computer vision techniques: a systematic literature review. Arch Comput Methods Eng 2018;25:507–43.

[16] Naresh YG, Nagendraswamy HS. Classification of medicinal plants: an approach using modified LBP with symbolic representation. Neurocomputing Jan. 2016;173:1789–97. https://doi.org/10.1016/J.NEUCOM.2015.08.090.

[17] Gogul I, Sathiesh Kumar V. Flower species recognition system using convolution neural networks and transfer learning. In: 2017 fourth international conference on signal processing, communication and networking (ICSCN). IEEE; 2017.

[18] Csillik O, Cherbini J, Johnson R, Lyons A, Kelly M. Identification of citrus trees from unmanned aerial vehicle imagery using convolutional neural networks. Drones Dec. 2018;2(4):1–16. https://doi.org/10.3390/drones2040039.

[19] Safonova A, Tabik S, Alcaraz-Segura D, Rubtsov A, Maglinets Y, Herrera F. Detection of fir trees (Abies sibirica) damaged by the bark beetle in unmanned aerial vehicle images with deep learning. Remote Sens Mar. 2019;11(6):643. https://doi.org/10.3390/rs11060643.

[20] Lee SH, Chan CS, Remagnino P. Multi-organ plant classification based on convolutional and recurrent neural networks. IEEE Trans Image Process Sep. 2018;27(9):4287–301. https://doi.org/10.1109/TIP.2018.2836321.

[21] Zhang B, Zhao L, Zhang X. Three-dimensional convolutional neural network model for tree species classification using airborne hyperspectral images. Rem Sens Environ 2020;247(Sep). https://doi.org/10.1016/j.rse.2020.111938.

[22] Momeny M, Jahanbakhshi A, Jafarnezhad K, Zhang YD. Accurate classification of cherry fruit using deep CNN based on hybrid pooling approach. Postharvest Biol Technol 2020;166(Aug). https://doi.org/10.1016/j.postharvbio.2020.111204.

[23] Singh UP, Chouhan SS, Jain S, Jain S. Multilayer convolution neural network for the classification of Mango leaves infected by anthracnose disease. IEEE Access 2019;7:43721–9. https://doi.org/10.1109/ACCESS.2019.2907383.

[24] He H, Pan J, Lu N, Chen B, Jiao R. Short-term load probabilistic forecasting based on quantile regression convolutional neural network and Epanechnikov kernel density estimation. Energy Rep Dec. 2020;6:1550–6. https://doi.org/10.1016/j.egyr.2020.10.053.
[25] Liu Z. Soft-shell shrimp recognition based on an improved AlexNet for quality evaluations. J Food Eng 2020;266(Feb). https://doi.org/10.1016/j.jfoodeng.2019.109698.
[26] Lu S, Lu Z, Zhang YD. Pathological brain detection based on AlexNet and transfer learning. J Comput Sci Jan. 2019;30:41–7. https://doi.org/10.1016/j.jocs.2018.11.008.
[27] Wang SH, et al. Alcoholism identification based on an Alexnet transfer learning model. Front Psychiatr 2019;10(Apr). https://doi.org/10.3389/fpsyt.2019.00205.
[28] Prasetyo E, Suciati N, Fatichah C. Multi-level residual network VGGNet for fish species classification. J King Saud Univ Comput Inf Sci 2021. https://doi.org/10.1016/j.jksuci.2021.05.015.
[29] Wei J, et al. Analyzing the impact of soft errors in VGG networks implemented on GPUs. Microelectron Reliab 2020;110(Jul). https://doi.org/10.1016/j.microrel.2020.113648.
[30] Szegedy Christian, et al. Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2016.
[31] Sandler Mark, et al. MobileNetV2: inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2018.
[32] Gulli Antonio, Pal Sujit. Deep learning with Keras. Packt Publishing Ltd.; 2017.
[33] Chouhan Siddharth Singh, Kaul Ajay, Singh Uday Pratap, Jain, Sanjeev. Plant leaves for image classification. Version 2. Retrieved 5 Jan. 2023, from, https://www.kaggle.com/datasets/csafrit2/plant-leaves-for-image-classification.
[34] Gulzar Yonis. Fruit image classification model based on MobileNetV2 with deep transfer learning technique. Sustainability 2023;15(3):1906.
[35] Amri Emna, et al. Advancing automatic plant classification system in Saudi Arabia: introducing a novel dataset and ensemble deep learning approach. Modeling Earth Systems and Environment 2024;10(2):2693–709.
[36] Gulzar Yonis. Enhancing soybean classification with modified inception model: a transfer learning approach. Emir J Food Agric 2024;36:1–9.
[37] Gulzar Yonis, et al. Harnessing the power of transfer learning in sunflower disease detection: a comparative study. Agriculture 2023;13(8):1479.
[38] Alkanan Mohannad, Gulzar Yonis. Enhanced corn seed disease classification: leveraging MobileNetV2 with feature augmentation and transfer learning. Frontiers in Applied Mathematics and Statistics 2024;9:1320177.
[39] Gulzar Yonis, et al. Adaptability of deep learning: datasets and strategies in fruit classification. In: BIO web of conferences. EDP Sciences; 2024. vol. 85.
[40] Huang Mengxing, et al. Efficient click-based interactive segmentation for medical image with improved Plain-ViT. IEEE J Biomed Health Informatics 2024. https://doi.org/10.1109/JBHI.2024.3392893.
[41] Huang Mengxing, et al. An interpretable approach using hybrid graph networks and explainable AI for intelligent diagnosis recommendations in chronic disease care. Biomed Signal Process Control 2024;91:105913.
[42] Nizamani Abdul Haseeb, et al. Advance brain tumor segmentation using feature fusion methods with deep U-Net model with CNN for MRI data. J King Saud Univ-Comput Information Sci 2023;35(9):101793.
[43] Bhatti Uzair Aslam, et al. MFFCG–Multi feature fusion for hyperspectral image classification using graph attention network. Expert Syst Appl 2023;229:120496.
[44] Bhatti Uzair Aslam, et al. Deep learning with graph convolutional networks: an overview and latest applications in computational intelligence. Int J Intell Syst 2023;1(2023):8342104.