International Journal for Computers and Information, IJCI, Vol. 10 - 3, Oct. 2023 (Special Issue)

A Comparative Study of State-of-the-Art Algorithms for Plant Recognition and Classification on a Large Dataset

Sara Elhamouly[1],Eman Meselhy[2],Gamal Farouk[3]
Department of Computer Science, Faculty of Computers and Information, Menofia University, Menofia ,32511, Egypt
sara.elhamouly@ci.menofia.edu.eg[1], eman.mohamed@ci.menofia.edu.eg[2], gamal.elmetwali@ci.menofia.edu.eg[3]

Abstract*— Plant classification and recognition is a vital task for many applications including retail, agriculture, and food processing. Several state-of the-art algorithms were developed in order to address this challenge. While there are several papers that were published to propose classification algorithms for plants and fruits, these algorithms targeted small datasets with 10K or fewer images. This study compares the performance of several Deep Learning models in fruit and plant classification on a large plants and fruits dataset with more than 225k images. This study aims to guide researchers about the performance implications of using popular models at large-scale by testing the scalability and reliability of these models. Fruits-262 is a dataset of over 225k images representing 262 different classes of plants and fruits. To ensure fairness, we trained each model using the same runtime environment on the Fruits-262 dataset. The models were evaluated using four different evaluation metrics; accuracy, loss, validation accuracy and validation loss. We also considered the computational complexity, the training time and the model size in order to evaluate the efficiency, reliability and scalability. Our findings reveal that there are some models that can offer high classification accuracy, yet with high computational resources and long iterations. This paper explores some potential alternatives and highlights some interesting models for future research.

Keywords—*: Plants; Classification; Convolutional Neural Networks; Comparisons; Analysis*

1. INTRODUCTION

Artificial intelligence (AI) has revolutionized many fields and industries. It has become one of the most important technologies in recent years. AI applications provide powerful tools for solving complex problems, enhancing productivity and improving performance. Computer vision is one of the domains in which AI has made great strides. This subfield focuses on enabling computers to analyze and interpret visual data such as videos and images.

Computer vision techniques are used in several types of applications including fruit detection [1], fruit classification [2], fruit grading [3], yield estimation [4] and disease classification [5]. Image classification is a critical aspect of computer vision. This involves assigning labels or categories to images according to their content. This process is especially relevant when it comes to fruit classification and recognition since it can help with various agricultural processes such as sorting and grading fruits or assessing their quality [6].

Deep learning is a subset of AI that has been instrumental in the advancement of computer vision techniques. It uses artificial neural networks to learn complex patterns and representations based on large amounts of data [7]. Convolutional Neural Networks (CNNs), in particular, have become an effective methodology for image classification tasks because of their ability to learn hierarchical feature-representations from raw pixel data. Different CNN architectures with different performance levels and characteristics have been developed and proposed over the recent few years.

In this paper, we compare the performance of plant classification and recognition task for 28 state-of the-art algorithms from the following families: EfficientNet, BiT, ResNet, Inception-ResNet, PnasNet and MobileNet. We aim that by evaluating and contrasting the algorithms on a large dataset, we can provide valuable insight into their strengths, weaknesses, and ultimately inform the selection of algorithms appropriate for large-scale applications within the computer vision field and plant recognition domain.

This paper is structured where section 2 reviews the relevant literature while Section 3 discusses the methodology used in this paper. The results are presented in Section 4 and the limitations of this research are listed in Section 5. The paper concludes with the Conclusions and Future Work in Section 6.

2. RELATED WORK

Plant recognition and classification has been the focus of several researchers in the last decade. Recent papers can be divided into two large domains: feature-based methods and deep learning-based methods. These two types are discussed over the next subsections.

2.1. Feature-based Methods

The authors in [8] proposed a fruit classification system that uses image features like color, shape, and texture. The dimensions of the fruit images were reduced using Principal Component Analysis (PCA) [9] before being fed into classification algorithms like Fed-forward Neural Networks (FNN) or Support Vector Machines (SVMs). The authors used a small dataset that comprises 1,653 images from 18 different classes to test the model accuracy leading to a conclusion that SVM had the highest accuracy at 88.2%.

However, the dataset did not include fruits or plants that had been sliced, dried or partially covered.

In a recent study [12], the authors used Bayesian-optimized Support Vector Machine and hybrid feature-based Random Forest classifier to compare the performance and accuracy of five different CNN models in detecting leaf diseases of apples, corn, potatoes, tomatoes and rice plants. The authors used a medium-size dataset containing 54k images and they concluded that MobileNet-driven models provide best accuracy at 96% in disease detection. However, the authors didn't demonstrate whether MobileNet models will continue to show superior performance with large datasets that incorporate higher number of classes.

Sandler and others [45] compared the performance of MobileNet based models in conducting object detection and classification activities. They compared the performance of MobileNet on ImageNet and the large COCO dataset with 200k+ images. Their performance showed that MobileNetV2 was the most efficient model out of the selected models. However, their work was limited to 6 models only and they didn't incorporate large set of models.

In a different study [10], the authors employed another Random Forest approach (RF) combined with KNN for the classification of three fruit types: Strawberry, Apples and Oranges. The authors used Scale-Invariant Feature Transform (SIFT) to extract features like shape, color and scale. The results of their experiment were interesting and showed that fruits with different shapes are harder to classify than fruits with similar shapes. However, their experiment was limited to 137 fruit images only.

In [11], the combination of machine-learning algorithms and color space was used to classify Cape gooseberry ripeness. The researchers used 925 images for both model training and validation. They reported that SVM was the most accurate machine learning classifier (70.14%) out of twelve different classifiers. Their method was limited to a small dataset and small coverage of algorithms. Multilayer neural networks approach was used to classify a total of 100 tomato images into large, medium and small classes along with four grades in each class [3]. On such small dataset, the average accuracy was 90.7%.

There are also a few studies which use images other than RGB, such as Near Infrared (NIR), or multispectral images. In [13], the authors proposed an in-field leaf-spectroscopy-based Grapevine varieties classification method. They collected leaves from 20 different types of grapevines and measured the NIR spectrum of each leaf. The spectra from these leaves were fed into a combination of two machine-learning algorithms: Support vector machines (SVM), and Artificial Neural Networks (ANN). Their method produced an overall accuracy of 87.25%.

In [14], a classification of strawberry ripeness using multispectral imagery of 17 bands is also proposed. The PCA was used to reduce the size of the multispectral image's features.

These papers were limited to small-to-medium size datasets with fewer than 100k images or focused on small number of classes in their classification algorithm which made it difficult to generalize these models.

2.2. Deep-Learning Methods

Deep learning (DL), or deep neural networks, are derived from the human brain. DL-based methods use a large neural network with several layers, nodes and activation functions [15]. Deep learning methods have become popular in recent years and are widely used for image classification tasks. Many works [10, 16 and 17] have reported promising results using deep learning methods on the classification of fruit images.

For example, [18] presented a lightweight CNN for the classification of Fruit-360 dataset. This paper showed that CNN performance is improved by adding additional features, such as RGB color and histogram. Their method achieved the highest accuracy of 93% in fruit classification. However, the dataset they used for their research contains 3 fruit classes only. Furthermore, [16] proposed an CNN model for classification of fruits on the Fruit-360 dataset. This model achieved a classification accuracy rate of 94.35%.

Fruits-360 dataset has become a popular reference for fruit classification and recognition with its medium size (69k images). [35] compared the performance and accuracy of fourteen different deep learning-based models on this dataset where MobileNet based models produced the best accuracy rate.

Yu et al. [32] leveraged deep learning using the fastest sensor in conjunction with a ResNet50 neural network to detect fruits for a strawberry-picking robot. The analysis results obtained in this study achieved an accuracy of 95.78%.

VGG-16 architectures have become popular in plant classification and disease detection [34]. Fuentes et al. [33] compared the three types of sensor families with respect to different architectures and achieved better results with VGG-16 architecture together with R-CNN, which is faster compared to CNN, together with the

On the other hand, [19] proposed a lightweight CNN model for fruit and plant classification on two different datasets with a comparison to a fine-tuned VGG-16 [20] model. This method had a classification accuracy between 99.49% for dataset-1 and 85.43% in dataset-2.

Deep learning methods developed toward video capturing as well where [21] proposed a classifier to categorize fruits in retail stores using video captured with the installed camera. The researchers used two convolutional networks; InceptionV3 [20] and MobileNet [22], to detect and classify vegetables and fruits on video. InceptionV3 achieved the highest accuracy (76%) for a ten-class dataset of fruits.

As discussed in the literature review, researchers tend to use smaller datasets in evaluating the performance of machine learning models. However. industries aim to deploy machine learning algorithms at large-scale where huge amounts of data are expected to be processed. Also, research papers that compares the performance of deep-learning models tend to focus on relatively small number of models in their comparisons. This paper focuses on addressing these gaps by comparing the performance of 28 models from the most popular model-families on a large dataset with 225k+ images.

3. METHODOLOGY

3.1. Selected Models and Architectures

This paper compares several AI and DL algorithms from the following families: EfficientNet, ResNet, Inception ResNet, MobileNet, NAS and BiT. Over the upcoming few subsections, these families are discussed in detail.

3.1.1. EfficientNet

EfficientNet is a family of Convolutional Neural Networks (CNNs), which are designed using compound scaling to balance depth, width and resolution [23].

1. **EfficientNet-B0**: This is the base model that serves as a basis for all other models with the architecture shown in Figure 1. This is the smallest and the least computationally expensive model in the family. EfficientNet-B0 offers a lower level of accuracy than the larger models.
2. **EfficientNet-B1** to **EfficientNet-B7**: These are enlarged versions of the base (B0) model, created by compound scaling. Compound scaling is a method that scales network width, depth and input resolution simultaneously to improve performance without increasing computational costs. The accuracy and network size increase as the model number (from B1 to B7) increases.

3.1.2. ResNet (Residual Networks)

ResNet is a CNN architecture that uses residual connections (skip connection) to solve the problem of vanishing gradients in deep networks [24]. ResNets come in different depths such as ResNet50, ResNet101 and ResNet152.

3.1.2. ResNet (Residual Networks)

ResNet is a CNN architecture that uses residual connections (skip connection) to solve the problem of vanishing gradients in deep networks [24]. ResNets come in different depths such as ResNet50, ResNet101 and ResNet152. The various ResNet algorithms are typically identified by the number of layers in the network. Some common ResNet architectures include:

1. **ResNet-18**: This architecture consists of 18 layers. This architecture includes 5 stages, each with 2 convolutional layers and a pooling layer, as well as 2 fully-connected layers. There are 8 blocks of residual layers, each with 2 convolutional layers.
2. **ResNet-34**: This model contains 34 layers. It includes 5 stages, each with 2 convolutional layers and a pooling layer, as well as 2 fully connected layers. The model has 16 blocks of residual layers, each with 2 convolutional layers.
3. **ResNet-50**: This model is an architecture that consists of 50 layers, and instead of using the usual 2-layer residual block, it uses bottleneck blocks. These blocks consist of 3 convolutional layers (1x1, 1x3, and 2x1) instead. ResNet-34 uses bottleneck blocks to reduce complexity and parameters. Figure 2 shows a sample architecture for the ResNet-50 model.
4. **ResNet-101**: ResNet-50 is a model with 101 layers. It follows a similar architecture, but has fewer bottleneck blocks. This change results in increased capacity and depth.
5. **ResNet-152**: This architecture consists of 152 layers. It uses bottleneck residual blocks again, but has an even greater number of blocks compared to ResNet-101, increasing the depth and capability of the network

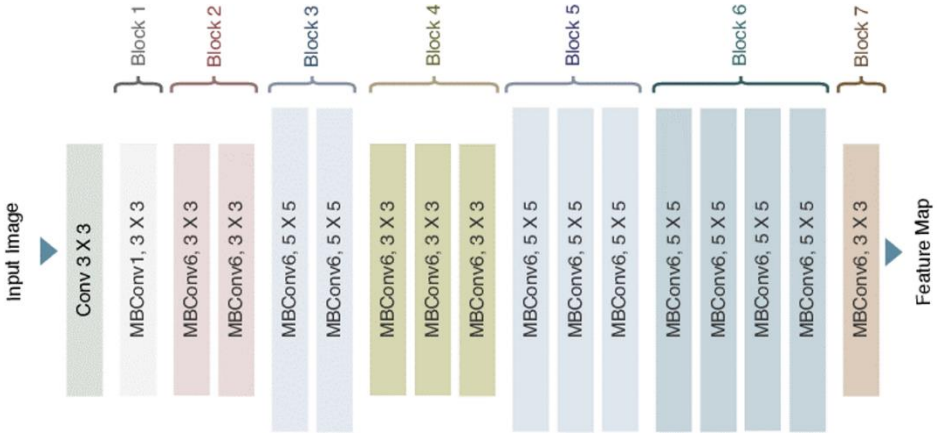

Fig.1 Architecture of EfficientNet-B0 Image taken from [39]

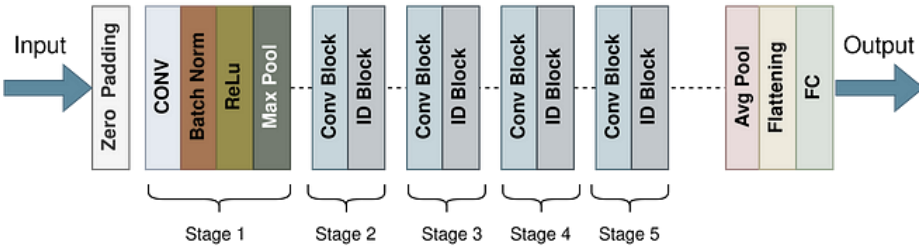

Fig. 2 Sample ResNet50 Model Architecture. Image taken from [40]

All ResNet architectures share the same design principles. For example, they use residual connections to allow the network to learn residual functions. Also, ResNet architectures are capable of performing at the highest level on a variety of computer vision tasks including object detection, image classification and segmentation. The choice of ResNet models depends on the balance required between accuracy and computational power for a particular task. ResNet models can be combined with other models as well. For example, [32] combined ResNet with Fast Fourier models to establish a new Fast Fourier Convolutional ResNet (FFC-ResNet).

3.1.3. Inception ResNet

Inception ResNet is a family of hybrid architectures that combines ideas from both the Inception and ResNet architectures [25]. This architecture was introduced by researchers at Google. The key components of this model are:

Inception modules: are the basic building blocks of the Inception architecture. Inception modules are composed of parallel convolutional branches, each with a different filter size and operation, such as 3x3, 5x5, and 1x1 convolutions. These branches' outputs are then concatenated together to create the module's final output. This design allows the network to be wider and capture information on multiple scales at each layer.

Residual connections: The ResNet architecture's primary innovation is the use residual connections, also known as a skip connection. This allows the network to learn residual functions while reducing the problem of vanishing gradients in deep networks. In Inception ResNet, residual blocks are created by combining Inception modules with residual connections. This enhances the learning capability and convergence of the networks.

Inception-ResNet has two main variations:

1. **Inception ResNet-V1**: This variant was built by incorporating the residual connections in Inception-V3 architecture [25]. It has the same complexity and depth as Inception-V3, but with a faster convergence rate and slightly better performance.
2. **Inception ResNet-V2**: This version is based on a more advanced Inception-V4 architecture. Inception ResNet-V2 is more complex and has a deeper design than Inception ResNet-V1 [25]. This complexity results in better performance for image classification tasks.

Inception-ResNet has demonstrated the best performance in various computer vision tasks such as object detection, image classification and segmentation with an architecture as shown in Figure 3. These results were achieved while maintaining an appropriate balance between model size and computational resources.

3.1.4. MobileNet (Mobile Networks)

MobileNet is a lightweight CNN family designed for embedded and mobile devices where computational resources and power consumption are limited. MobileNets are based on depth-separable convolutions which reduces the complexity and number of parameters compared to conventional CNNs [26]. The success of MobileNets comes from a belief that the model size is not a necessity for achieving better results. This has shown tremendous success in various experiments as shown in [12] and [35]. There are several variations of MobileNet models including:

1. **MobileNet-V1**: The first MobileNet version, developed by Google researchers, is based upon the concept of depth-wise separate convolutions. The standard convolution is replaced by two separate operations, depth-wise (which applies convolution filters to each channel separately) and point-wise (which combines the outputs from depth-wise convolutions with 1x1 convolutions). This design reduces the computational complexity and number of parameters compared to conventional CNN architectures.

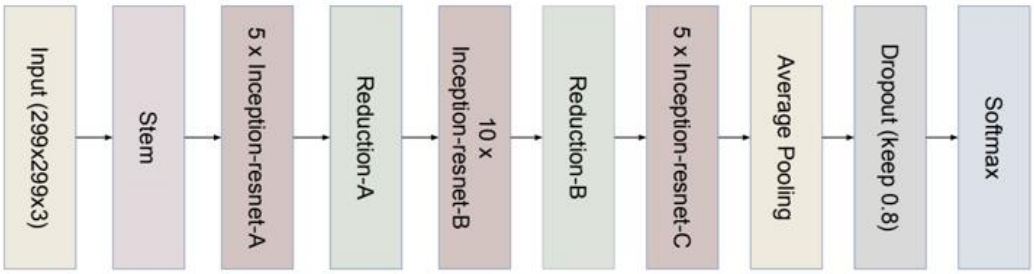

Fig.3 Inception ResNet Model Architecture. Image taken from [41]

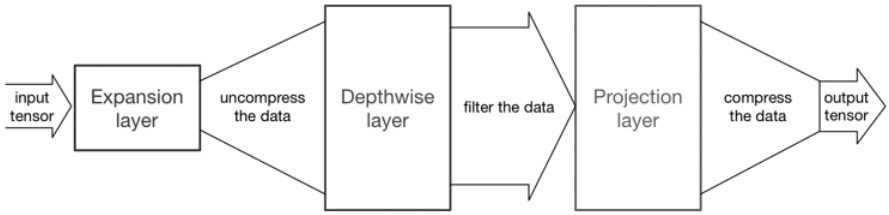

Fig.4 Mobile-V2 Building Blocks. Image taken from [42]

2. **MobileNet-V2**: MobileNet-V2 is the second version, which builds on MobileNet-V1 with inverted residual blocks. It also introduces linear bottlenecks and inverted residual blocks that are Rest-Net like structures, with input and output channels in reverse order. The linear bottleneck consists of a convolution layer 1x1, which reduces the channels without introducing nonlinearities. This helps preserve the information in the network. MobileNet-V2 is a model that improves efficiency and performance compared to MobileNet-V1 [26]. Figure 4 shows the main building blocks of MobileNet-V2
3. **MobileNet-V3**: The latest version of MobileNet and it combines architectural innovations from MobileNet-V2 with advanced search techniques such as Neural Architecture Search (NAS), NetAdapt, and other similar algorithms.

3.1.5. NASNet (Neural Architecture Search Networks)

NASNet models were invented by researchers at Google Brain while researching Neural Architecture Search (NAS) [27]. NAS offers automated methods to design high-performance neural networks. These automated methods use reinforcement learning to search for the optimal architecture by training and evaluating numerous candidate architectures on a given dataset.

Figure 5 illustrates the main building blocks of NAS.

There are several examples of NASNet models. For instance,

1. **NASNet-A** is the original NASNet discovered by NAS. It achieved the best possible performance on ImageNet dataset for image classification tasks. This model is computationally costly and therefore not suitable for devices with limited resources.
2. **NASNet-Mobile** is a smaller and more efficient version NASNet-A. It was designed specifically for mobile and embedded devices. It has a similar architecture to NASNet-A, but with reduced width and depth. This minimization results in a smaller number of parameters.
3. **PNASNet** uses more advanced version of NAS,called Progressive NAS (PNAS) to provide better results for large datasets.

3.1.6. BiT (Big Transfer Networks)

The BiT family is a collection of pre-trained CNNs that uses transfer learning in order to achieve high performance for various computer vision tasks [28]. BiT networks can classification, or segmentation.

The BiT network architecture is based on ResNet, but the innovation comes from the pre-training method. Researchers pre-trained the models using a large dataset (ImageNet-21k containing over 14,000,000 images), and then fine-tuned their performance on smaller datasets. Figure 6 shows the model architecture for BiT-CNN.

The key principles that have led to BiT's success include:

1. **Scale**: BiT is trained on large datasets, which allows them to learn more features and generalize better across tasks.
2. **Architecture**: BiT uses the ResNet architecture which has been proven effective in a variety of an object identification tasks. Researchers used different depths (e.g., ResNet-50 ResNet-101 ResNet-152) in order to examine the impact of architectural choices on transfer learning.
3. **Fine-tuning**: BiT models can be fine-tuned using task-specific datasets by following a simple but effective strategy. This strategy uses a large batch size and extensive data augmentation to allow the models to quickly adapt to new tasks.

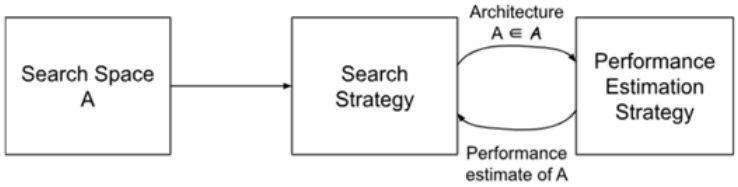

Fig.5 NAS Building Blocks. Image taken from [43]

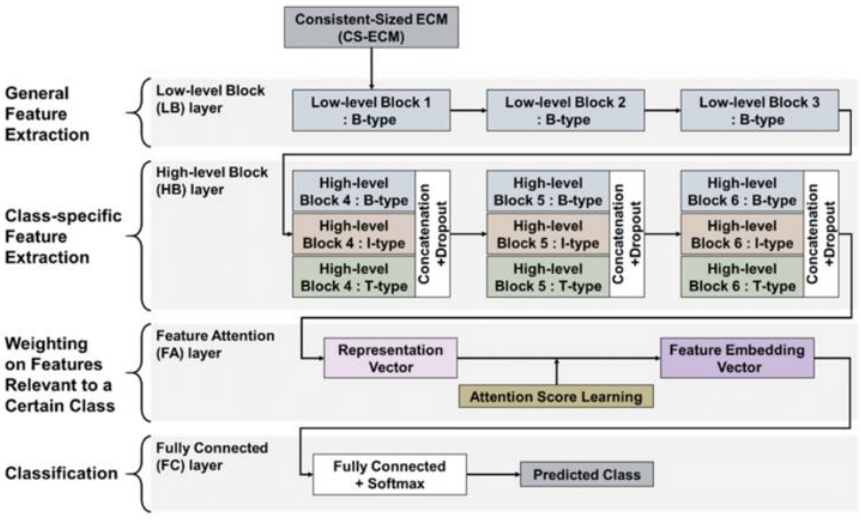

Fig.6 BIT-CNN Architecture. Image taken from [44]

This paper selected the most prominent algorithms from each of the families described earlier. Table 1 lists all the selected models and their respective families. The model names reflect important information about the model such as family name, model base, model version, size of the dataset that the model was pre-trained on (if any), fine-tuning parameters for the pre-trained model (if applicable), number of model layers, depth of the model, and whether the input images are resized to small, medium, large or specified width by height. All selected models were run in 5 different epochs and the performance for each epoch was captured. More details are discussed in the results section.

4. Results and Discussion

4.1.Dataset

Fruits-262 dataset was created to improve the process of fruit classification [29]. This dataset is selected due to its large size (225k+ images). Fruit-262 contains a vast majority of the popular and known plants and fruits. Due to the extensive and large number of images for each class (861 images/class), the leveraged models will offer high-accuracy for plant and fruit classification. Below are the statistics for the selected dataset.

1. **Total number of images:** 225,640 images
2. **Number of classes:** 262 fruits and plants
3. **Average number of images for each label/class:** 861 images
4. **Average image width:** 213 pixels
5. **Average image height:** 262 pixels

All the plant and fruit classes were investigated to ensure enough number of samples is available for each class.

Figure 7 shows the distribution of input images for sample classes in the dataset.

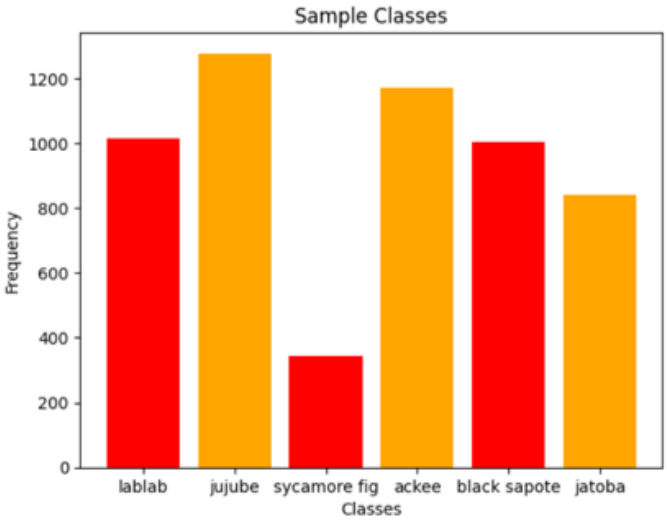

Fig.7 Input Data Distribution for Sample Classes in the Fruits-262 Dataset

4.2 Run-time Environment

All the models were run on Google Colab Pro to ensure fairness. Every model used 2 vCPUs with 32 GB RAM, 15 GB persistent storage and a GPU. In addition, TensorFlow was used to implement each model using Keras library.

4.3 Evaluation Metrics

In evaluating selected models, the dataset was divided into two categories: training data and validation data. Appendix A defines the terminologies of training and validation data. In general, TensorFlow models use 4 main metrics to evaluate and monitor their performance. These metrics have been widely used in several research papers such as [36] and [45].

1. **Accuracy** measures the percentage of predictions that are correct. The number of correct forecasts divided by the total number is used to calculate it. TensorFlow's accuracy is usually used to solve classification problems. The goal is to determine the correct input class or category. Accuracy is calculated as the ratio of correct predictions to the total number of predictions:

$$\text{Accuracy} = \frac{N_c}{N_t} \quad (1)$$

where N_C is the number of correct of predictions and N_t is the total number of predictions

2. **Loss** is also called the objective function or cost function. It measures the difference between the predictions of the model and the ground truth labels. The loss function quantifies errors in model predictions and is used to guide the training process. Training is aimed at minimizing the loss through the adjustment of the model parameters. TensorFlow uses the Mean Squared error (MSE) to solve regression problems, and Cross-Entropy loss for classification problems.

$$\text{Cross Entropy} = \left(-\frac{1}{N}\right) * \sum_{i=1}^{N} y_{\text{true}} * \log y_{pred} + (1 - y_{true}) * \log(1 - y_{pred}) \quad (2)$$

Equation cited from [46]

where N is the number of samples, y_{true} represents the true labels and y_{pred} represents the model's predicted probabilities.

3. **Validation Loss** is calculated based on a validation dataset that is separate from the training data. The validation dataset is used to monitor overfitting and evaluate the model on new data. The validation loss may be high for a model with low training loss. This indicates that the model has learned to memorize the training data but is not able to generalize it well.
4. **Validation Accuracy** is the accuracy metric that is calculated using the validation dataset. It is also used to monitor overfitting and evaluate the model performance when using unseen data. A high validation accuracy is a sign that the model generalizes and will perform well with new data.

$$\text{Validation Accuracy} = N_{cv} / N_{tv} \quad (3)$$

Where N_{cv} is the number of correct predictions on validation data and N_{tv} is the total number of correct predictions on validation data

When evaluating model performance, the following rules apply:

1. If **Validation Loss** is increasing and **Validation Accuracy** is decreasing, the model is cramming values and not learning.
2. If **Validation Loss** is increasing and Validation Accuracy is increasing, this can happen due to overfitting.
3. If **Validation Loss** is decreasing and **Validation Accuracy** is increasing, the model is learning and working fine.

There are other evaluation metrics that could be used such as F1-score, precision and recall. In the next sub-section, the experiment results will be presented and explained further.

4.4. Experiment Results

Running the models at large-scale for industrial purposes require continue retraining for the model based on incoming data. Therefore, the time spent for each epoch must be reasonable. While the term "reasonable" has subjectivity in its definition, the authors have identified that ~ 1 hour/epoch for such dataset. As a result, some algorithms were identified for their insufficient performance and limited scalability. These models are: efficientnet_b5, efficientnet_b6, efficientnet_b7 and pnasnet_large. Every epoch in these 3 models took over 90 minutes (over 7.5 hours for 5 epochs) and therefore, they can't provide good performance at large-scale operation. Therefore, these models will be excluded from future comparisons.

Table 2 shows the full time spent/epoch for each model (in minutes). It's worth noting that longer execution times doesn't imply higher accuracy.

To prove this assumption, pnasnet_large model was included in the list of models that were compared. Therefore, Table 3 shows the comparison results for the selected models on the Fruits-262 dataset when trained for 5 different iterations (epochs). The results indicate that efficientnetv2-l-21k-ft1k has the highest accuracy and validation accuracy scores (and lowest loss and validation loss) while nasnet_mobile has the lowest validation accuracy and highest validation loss scores.

The validation accuracy scores for all models are illustrated in Figure 8. Figure 8 shows that efficientnetv2-l-21k-ft1k model is the best model. This model is an EfficientNet-V2 model that is pre-trained on a 21k image dataset and fine-tuned on a 1k image datasets. All input images to the model are resized to large size. The model efficientnetv2-l-21k-ft1k consumes a lot of resources due to its need to resize all images to large size. The model size is ~ 428MB [30]. Therefore, there is a need for large-cluster to run such heavy processing.

A good compromise is offered by the efficientnetv2-m-21k-ft1k model. This model is an EfficientNet-V2 model that is pre-trained on a 21k image dataset and fine-tuned on a 1k image datasets. All input images to the model are resized to medium size. Due to the smaller resize, the model size is 191MB only (56% less size than efficientnetv2-l-21k-ft1k). In addition, the average epoch length for efficientnetv2-l-21k-ft1k is 40.5 minutes while the average epoch length for efficientnetv2-m-21k-ft1k is 30 minutes with around 25% time saving.

Table 1. Selected Models and their respective families

Model Family	Selected Model(s)	Notes
EfficientNet	efficientnet_b0	V1 of EfficientNet with base of 0
	efficientnet_b1	Similar to efficientnet_b0 with base of 1
	efficientnet_b2	Similar to efficientnet_b0 with base of 2
	efficientnet_b3	Similar to efficientnet_b0 with base of 3
	efficientnet_b4	Similar to efficientnet_b0 with base of 4
	efficientnet_b5	Similar to efficientnet_b0 with base of 5
	efficientnet_b6	Similar to efficientnet_b0 with base of 6
	efficientnet_b7	Similar to efficientnet_b0 with base of 7
	efficientnetv2-b0	V2 of EfficientNet with base of 0
	efficientnetv2-b1	Similar to efficientnetv2_b0 with base of 1
	efficientnetv2-b2	Similar to efficientnetv2_b0 with base of 2
	efficientnetv2-b3	Similar to efficientnetv2_b0 with base of 3
	efficientnetv2-s	EfficientNet V2 trained on 1k images resized to small size
	efficientnetv2-s-21k-ft1k	EfficientNet V2 trained on 21k image and fine tuned on 1k images. All images are resized to small size
	efficientnetv2-m-21k-ft1k	Similar to efficientnetv2-s-21k-ft1k but all images are resized to medium size
	efficientnetv2-l-21k-ft1k	Similar to efficientnetv2-s-21k-ft1k but all images are resized to a large size
ResNet	resnet_v1_50	ResNet model with 50 layers
Inception ResNet	inception_resnet_v2	Inception ResNet-V2 model
MobileNet	mobilenet_v2_100_224	V2 of MobileNet with images resized to 100 x 224
	mobilenet_v2_130_224	V2 of MobileNet with images resized to 130 x 224
	mobilenet_v2_140_224	V2 of MobileNet with images resized to 140 x 224
	mobilenet_v3_small_075_224	V3 of MobileNet with small depth and images resized to 75 x 224
	mobilenet_v3_small_100_224	V3 of MobileNet with small depth and images resized to 100 x 224
	mobilenet_v3_large_075_224	V3 of MobileNet with large depth and images resized to 75 x 224
	mobilenet_v3_large_100_224	V3 of MobileNet with large depth and images resized to 100 x 224
NASNet	nasnet_mobile	NASNet Mobile model
	pnasnet_large	Progressive NASNet model prepared for large dataset
BiT	bit_s-r50x1	BiT small model based on ResNet_v2_50

On the other hand, the pnasnet_large model had the highest growth potential among all selected model although each epoch took over 2 hours (the longest among all selected models).

5. LIMITATIONS

This paper compared the performance of 28 models from different families. Despite the large number of models, there are several models that were not included in this experiment. For example, AlexNet and GoogleNet families were not included in this paper. Also, four evaluation metrics were used to evaluate these models. Other evaluation metrics should be included and can provide additional insights such as F1-score, precision and recall.

6. CONCLUSION AND FUTURE WORK

As the world evolves toward Artificial General Intelligence (AGI), there is a need to generalize algorithms and models to serve multiple purposes efficiently. In this paper, the authors explored popular machine learning algorithms that tend to perform well on small to medium size datasets. Some of these models didn't perform well on a large scale.

In conclusion, this paper compared the performance of several deep-learning models from several families on the large Fruits-262 dataset. These models tend to classify images with high-accuracy when applied on small-to-medium size datasets.

Through an extensive literature review, the authors discussed the latest developments in plant and fruit classification, the selected models, and performance comparison studies.

In the methodology section, the authors outlined the data pre-processing, augmentation techniques and training strategies employed by each model. The performance of these models was then evaluated using metrics like accuracy, loss, validation accuracy and validation loss. The scores for validation accuracy were also visualized.

The study identified a few models with faster execution and superior results that could be used at large scale. Several models didn't retain high levels of accuracy when applied to large datasets (e.g., resnet, mobilenet and bit families). Our findings revealed that efficientnetv2-l-21k-ft1k has the best accuracy and validation accuracy. Due to its large size, efficientnetv2-m-21k-ft1k can be used as an alternative model given its lower size and faster execution time. The pnasnet_large was also identified as having a higher potential for higher numbers of iterations, despite its relatively long learning time.

In the future, this comparison needs to be implemented with higher number of epochs (iterations) with investigation to the models that showed higher potential (e.g., pnasnet_large). Furthermore, there is a need to explore the underlying reasons that prevented models like resnet and mobilenet from performing at the same accuracy offered at smaller datasets. Finally, the comparison results shown don't reflect an ideal model that can offer best accuracy with low consumption of resources. Therefore, the development of a new model that achieves this balance is highly needed.

Table 2. Execution Times in minutes for selected models over 5 epochs

Model	1st Epoch	2nd Epoch	3rd Epoch	4th Epoch	5th Epoch
efficientnet_b0	14.95	14.36667	14.31667	14.36667	14.31667
efficientnet_b1	19.75	19.88333	19.63333	19.55	19.56667
efficientnet_b2	24.48333	24.18333	24.25	24.21667	24.16667
efficientnet_b3	36.83333	36.53333	36.5	36.56667	36.45
efficientnet_b4	67.01667	66.43333	66.4	66.33333	65.55
efficientnet_b5	**90+**	**90+**	**90+**	**90+**	**90+**
efficientnet_b6	**90+**	**90+**	**90+**	**90+**	**90+**
efficientnet_b7	**90+**	**90+**	**90+**	**90+**	**90+**
efficientnetv2-b0	24.71667	23.45	23.2	23.18333	23.21667
efficientnetv2-b1	27.28333	26.55	26.78333	26.61667	26.95
efficientnetv2-b2	31.71667	32.35	32.21667	31.73333	32.7
efficientnetv2-b3	43.83333	43.45	42.81667	43.4	43.41667
efficientnetv2-s	58.78333	58.3	58.33333	58.45	58.7
efficientnetv2-s-21k-ft1k	16.45	16.21667	16.21667	16.2	16.16667
efficientnetv2-m-21k-ft1k	30.25	29.93333	29.93333	29.93333	29.85
efficientnetv2-l-21k-ft1k	41.11667	40.63333	40.61667	40.56667	40.46667
resnet_v1_50	30.25	29.15	29.03333	28.53333	28.88333
inception_resnet_v2	67.9666	67.4166	67.56666	68.6833	67.6833
mobilenet_v2_100_224	24.48333	25.23333	25.3	24.15	24.45
mobilenet_v2_130_224	13.83333	13.31667	13.23333	13.2	13.4
mobilenet_v2_140_224	13.86667	13.76667	13.73333	13.53333	13.43333
mobilenet_v3_small_075_224	11.61667	11.26667	11.26667	11.31667	11.43333
mobilenet_v3_small_100_224	11.46667	11.53333	11.61667	11.91667	11.7
mobilenet_v3_large_075_224	12.13333	11.85	11.88333	12.01667	12.13333
mobilenet_v3_large_100_224	12.48333	12.46667	12.55	12.66667	12.56667
nasnet_mobile	26.08333	25.23333	24.73333	25.13333	25.61667
pnasnet_large	**124.2**	**123.8667**	**123.5833**	**123.7167**	**122.0167**
bit_s-r50x1	30.68333	30.31667	30.28333	30.28333	30.33333

Table 3. Model Evaluation Metrics (for 5 epoch-run) - with highlights for the most accurate and least accurate

Model	Accuracy	Loss	Validation Accuracy	Validation Loss
efficientnet_b0	0.6271	2.4530	0.6581	2.3527
efficientnet_b1	0.6246	2.4513	0.6570	2.3453
efficientnet_b2	0.6213	2.4764	0.6497	2.3818
efficientnet_b3	0.6408	2.3967	0.6639	2.3140
efficientnet_b4	0.6259	2.4882	0.6461	2.4170
efficientnetv2-b0	0.5263	2.8541	0.5720	2.6892
efficientnetv2-b1	0.5335	2.8349	0.5815	2.6620
efficientnetv2-b2	0.5251	2.8746	0.5645	2.7255
efficientnetv2-b3	0.5671	2.7012	0.6042	2.5694
efficientnetv2-s	0.6021	2.5822	0.6257	2.5010
efficientnetv2-s-21k-ft1k	0.7796	1.8407	0.8038	1.7428
efficientnetv2-m-21k-ft1k	0.8123	1.7155	0.8272	1.6419
efficientnetv2-l-21k-ft1k	0.8208	1.6835	0.8362	1.6070
resnet_v1_50	0.5738	2.6063	0.6156	2.4574
inception_resnet_v2	0.5285	2.7369	0.5650	2.6074
mobilenet_v2_100_224	0.6017	2.4797	0.6314	2.3816
mobilenet_v2_130_224	0.6428	2.3377	0.6711	2.2513
mobilenet_v2_140_224	0.6514	2.3042	0.6760	2.2242
mobilenet_v3_small_075_224	0.6017	2.5138	0.6275	2.4327
mobilenet_v3_small_100_224	0.6031	2.5107	0.6249	2.4346
mobilenet_v3_large_075_224	0.6394	2.3872	0.6583	2.3222
mobilenet_v3_large_100_224	0.6639	2.2711	0.6847	2.1958
nasnet_mobile	0.4481	3.0582	0.4854	2.9341
pnasnet_large	0.5404	2.6998	0.5536	2.6547
bit_s-r50x1	0.6307	2.3735	0.6749	2.2246

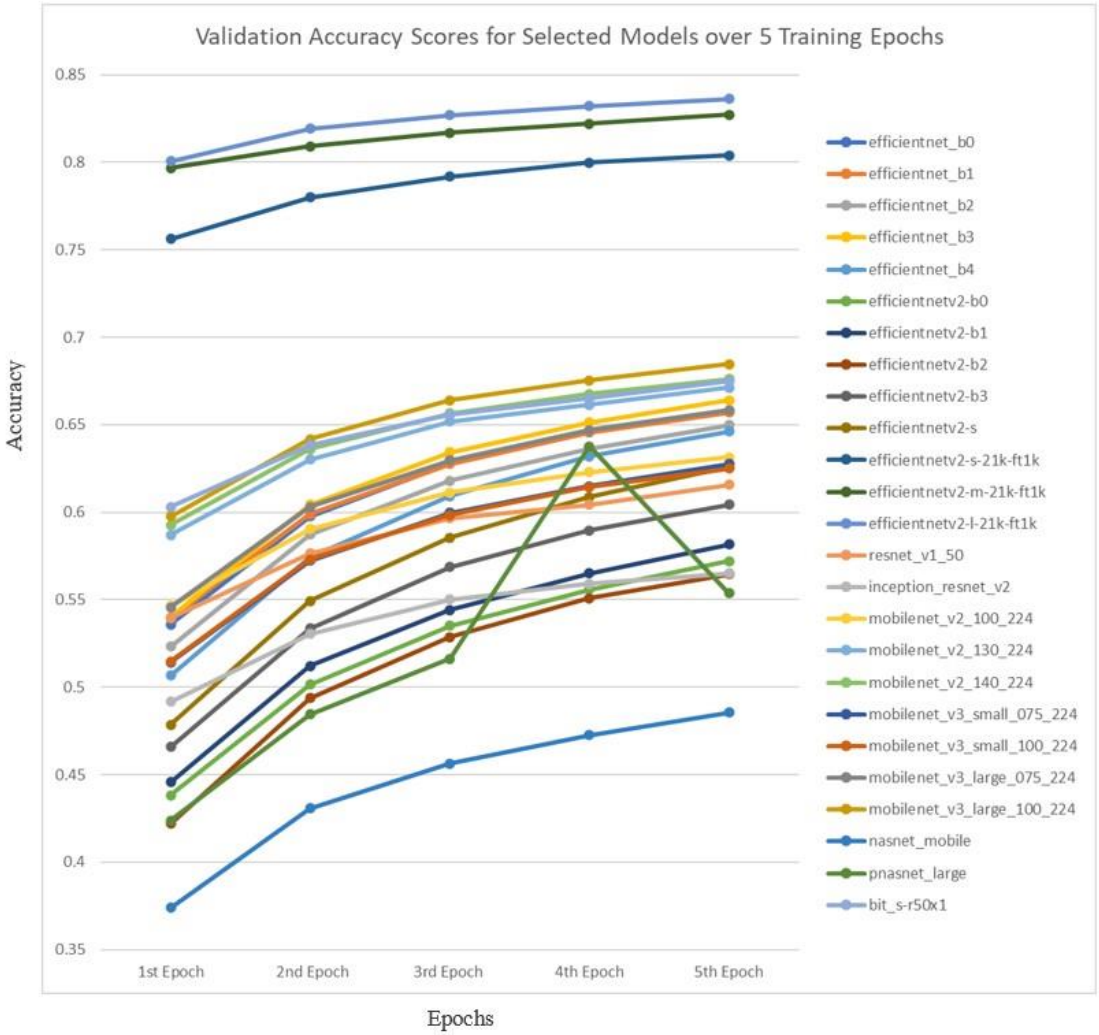

Figure. 8 Validation Accuracy Scores across Selected Models over 5 Training Epochs on the Fruits-262 Dataset

References

[1] A. Koirala, K. B. Walsh, Z. Wang, and C. McCarthy, "Deep learning for real-time fruit detection and orchard fruit load estimation: benchmarking of 'MangoYOLO'," Precision Agriculture, vol. 20, no. 6, pp. 1107-1135, 2019/12/01 2019.

[2] Y. Zhang, S. Wang, G. Ji, and P. Phillips, "Fruit classification using computer vision and feedforward neural network," Journal of Food Engineering, vol. 143, pp. 167-177, 2014/12/01/ 2014.

[3] L. Liu, Z. Li, Y. Lan, Y. Shi, and Y. Cui, "Design of a tomato classifier based on machine vision," PLOS ONE, vol. 14, no. 7, p. e0219803, 2019.

[4] J. Shook, T. Gangopadhyay, L. Wu, B. Ganapathysubramanian, S. Sarkar, and A. K. Singh, "Crop yield prediction integrating genotype and weather variables using deep learning," PLOS ONE, vol. 16, no. 6, p. e0252402, 2021.

[5] E. Khan, M. Z. U. Rehman, F. Ahmed, and M. A. Khan, "Classification of Diseases in Citrus Fruits using SqueezeNet," in 2021 International Conference on Applied and Engineering Mathematics (ICAEM), 30-31 Aug. 2021 2021, pp. 67-72.

[6] F. Wang et al., "Residual Attention Network for Image Classification," in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 21-26 July 2017 2017, pp. 6450-6458.

[7] T. B. Shahi, C. Sitaula, A. Neupane, and W. Guo, "Fruit classification using attention-based MobileNetV2 for industrial applications," (in eng), PLoS One, vol. 17, no. 2, p. e0264586, 2022.

[8] Y. Zhang, S. Wang, G. Ji, and P. Phillips, "Fruit classification using computer vision and feedforward neural network," Journal of Food Engineering, vol. 143, pp. 167-177, 2014/12/01/ 2014.

[9] H. Abdi and L. J. Williams, "Principal component analysis," WIREs Computational Statistics, vol. 2, no. 4, pp. 433-459, 2010/07/01 2010.

[10] H. M. Zawbaa, M. Hazman, M. Abbass, and A. E. Hassanien, "Automatic fruit classification using random forest algorithm," in 2014 14th International Conference on Hybrid Intelligent Systems, 14-16 Dec. 2014 2014, pp. 164-168.

[11] W. Castro, J. Oblitas, M. De-La-Torre, C. Cotrina, K. Bazán, and H. Avila-George, "Classification of Cape Gooseberry Fruit According to its Level of Ripeness Using Machine Learning Techniques and Different Color Spaces," IEEE Access, vol. 7, pp. 27389-27400, 2019.

[12] A. K. Singh, S. V. N. Sreenivasu, U. S. B. K. Mahalaxmi, H. Sharma, D. D. Patil, and E. Asenso, "Hybrid Feature-Based Disease Detection in Plant Leaf Using Convolutional Neural Network, Bayesian Optimized SVM, and Random Forest Classifier," Journal of Food Quality, vol. 2022, p. 2845320, 2022/02/10 2022.

[13] S. Gutiérrez, J. Tardaguila, J. Fernández-Novales, and M. P. Diago, "Support Vector Machine and Artificial Neural Network Models for the Classification of Grapevine Varieties Using a Portable NIR Spectrophotometer," PLOS ONE, vol. 10, no. 11, p. e0143197, 2015.

[14] C. Liu et al., "Application of multispectral imaging to determine quality attributes and ripeness stage in strawberry fruit," (in eng), PLoS One, vol. 9, no. 2, p. e87818, 2014.

[15] J. Schmidhuber, "Deep learning in neural networks: an overview," (in eng), Neural Netw, vol. 61, pp. 85-117, Jan 2015.

[16] J. L. Joseph, V. A. Kumar, and S. P. Mathew, "Fruit classification using deep learning," 2021: Springer, pp. 807-817.

[17] H. Mureşan and M. Oltean, "Fruit recognition from images using deep learning," Acta Universitatis Sapientiae, Informatica, vol. 10, no. 1, pp. 26-42, 2018.

[18] J. L. Rojas-Aranda, J. I. Nunez-Varela, J. C. Cuevas-Tello, and G. Rangel-Ramirez, "Fruit Classification for Retail Stores Using Deep Learning," in Pattern Recognition, vol. 12088: © Springer Nature Switzerland AG 2020., 2020, pp. 3-13.

[19] M. S. Hossain, M. Al-Hammadi, and G. Muhammad, "Automatic Fruit Classification Using Deep Learning for Industrial Applications," IEEE Transactions on Industrial Informatics, vol. 15, no. 2, pp. 1027-1034, 2019.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

[21] F. Femling, A. Olsson, and F. Alonso-Fernandez, "Fruit and Vegetable Identification Using Machine Learning for Retail Applications," in 2018 14th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), 26-29 Nov. 2018 2018, pp. 9-15.

[22] S. Chakraborty, F. M. J. M. Shamrat, M. M. Billah, M. A. Jubair, M. Alauddin, and R. Ranjan, "Implementation of Deep Learning Methods to Identify Rotten Fruits," in 2021 5th International Conference on Trends in Electronics and Informatics (ICOEI), 3-5 June 2021 2021, pp. 1207-1212.

[23] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2019: PMLR, pp. 6105-6114.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2016, pp. 770-778.

[25] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," 2017, vol. 31, 1 ed.

[26] A. G. Howard et al., "Mobilenets: Efficient convolutional neural networks for mobile vision applications," arXiv preprint arXiv:1704.04861, 2017.

[27] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," 2018, pp. 8697-8710.

[28] A. Kolesnikov et al., "Big transfer (bit): General visual representation learning," 2020: Springer, pp. 491-507.

[29] M.-D. Minuț and A. Iftene, "Creating a dataset and models based on convolutional neural networks to improve fruit classification," 2021: IEEE, pp. 155-162.

[30] EfficientNetV2 ImageNet21k-FT1k-Large. TensorFlow Hub. Accessed on May 10, 2023. Available from (https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet21k_ft1k_l/classification/2

[31] EfficientNetV2 ImageNet21k-FT1k-Medium. TensorFlow Hub. Accessed on May 10, 2023. Available from https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet21k_ft1k_m/classification/2.

[32] Y. Yu, K. Zhang, L. Yang, and D. Zhang, "Fruit detection for strawberry harvesting robot in non-structural environment based on Mask-RCNN," Computers and Electronics in Agriculture, vol. 163, p. 104846, 2019/08/01/ 2019.

[33] A. Fuentes, S. Yoon, S. C. Kim, and D. S. Park, "A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition," Sensors, vol. 17, no. 9, p. 2022, 2017.

[34] R. Sujatha, J. M. Chatterjee, N. Z. Jhanjhi, and S. N. Brohi, "Performance of deep learning vs machine learning in plant leaf disease detection," Microprocessors and Microsystems, vol. 80, p. 103615, 2021/02/01/ 2021.

[35] V. L. Narla and G. Suresh, "Multiple Feature-Based Tomato Plant Leaf Disease Classification Using SVM Classifier," 2023: Springer, pp. 443-455.

[36] R. Siddiqi, "Comparative performance of various deep learning based models in fruit image classification," 2020, pp. 1-9.

[37] K. Ramamurthy, A. R. Varikuti, B. Gupta, and N. Aswani, "A deep learning network for Gleason grading of prostate biopsies using EfficientNet," (in eng), Biomed Tech (Berl), vol. 68, no. 2, pp. 187-198, Apr 25 2023.

[38] A. Zhou et al., "Multi-head attention-based two-stream EfficientNet for action recognition," Multimedia Systems, vol. 29, no. 2, pp. 487-498, 2023.

[39] T. Ahmed and N. H. N. Sabab, "Classification and understanding of cloud structures via satellite images with EfficientUNet," SN Computer Science, vol. 3, pp. 1-11, 2022.

[40] W. Yu and M. Nishio, "Multilevel Structural Components Detection and Segmentation toward Computer Vision-Based Bridge Inspection," Sensors, vol. 22, no. 9, p. 3502, 2022.

[41] Sharma, Vibhaakar, Swathi Gangaraju, and Vishal K. Sharma. "Masked face recognition." (2019): 1-7. Accessed on May 10. Available from http://cs230.stanford.edu/projects_winter_2021/reports/70747149.pdf.

[42] Badgujar, S. (2021, July 30). Creating mobilenetsv2 with tensorflow from scratch. Medium. https://medium.com/analytics-vidhya/creating-mobilenetsv2-with-tensorflow-from-scratch-c85eb8605342. Accessed on May 10.

[43] Gupta, A. (2021, May 5). Neural Architecture Search. Medium. https://ag7982.medium.com/neural-architecture-search-nas-49d378fa04fe. Accessed on May 10.

[44] H. Lee and M. Shin, "Learning Explainable Time-Morphology Patterns for Automatic Arrhythmia Classification from Short Single-Lead ECGs," Sensors, vol. 21, no. 13.

[45] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," 2018, pp. 4510-4520.

[46] The cross-entropy cost function. (2023, June 6). Y Combinator Research. https://eng.libretexts.org/@go/page/3752. Accessed on May 10.

Appendix A

Fruits-262 dataset was divided into training data and validation data.

- **Training data** are used to fit parameters (weights, biases, etc.) into the model. Models learn from examples by adjusting their weights and biases in order to minimize the loss function. The model's primary source of data is its training data, which allow it to identify patterns, relationships and generalizations, all of which can be used to predict data that has not yet been seen.

- **Validation data** are separate examples that are not used during the training process. These data are used to assess the performance of the model during training and to check for overfitting. Overfitting occurs when the model becomes overly sensitive to noise and random fluctuations within the training data. This may lead to a poor generalization of the data. If necessary, hyperparameters can be adjusted to prevent overfitting.