Transfer Learning based Performance Comparison of the Pre-Trained Deep Neural Networks

Jayapalan Senthil Kumar, Syahid Anuar, Noor Hafizah Hassan
Razak Faculty of Technology and Informatics
Universiti Teknologi Malaysia (UTM)
54100 Kuala Lumpur, Malaysia

***Abstract*—Deep learning has grown tremendously in recent years, having a substantial impact on practically every discipline. Transfer learning allows us to transfer the knowledge of a model that has been formerly trained for a particular task to a new model that is attempting to solve a related but not identical problem. Specific layers of a pre-trained model must be retrained while the others must remain unmodified to adapt it to a new task effectively. There are typical issues in selecting the layers to be enabled for training and layers to be frozen, setting hyper-parameter values, and all these concerns have a substantial effect on training capabilities as well as classification performance. The principal aim of this study is to compare the network performance of the selected pre-trained models based on transfer learning to help the selection of a suitable model for image classification. To accomplish the goal, we examined the performance of five pre-trained networks, such as SqueezeNet, GoogleNet, ShuffleNet, Darknet-53, and Inception-V3 with different Epochs, Learning Rates, and Mini-Batch Sizes to compare and evaluate the network's performance using confusion matrix. Based on the experimental findings, Inception-V3 has achieved the highest accuracy of 96.98%, as well as other evaluation metrics, including precision, sensitivity, specificity, and f1-score of 92.63%, 92.46%, 98.12%, and 92.49%, respectively.**

***Keywords*—Transfer learning; deep neural networks; image classification; Convolutional Neural Network (CNN) models**

I. Introduction

The primary evolution of neural networks was stimulated by the desire to design a process that could imitate the human brain. The ability of conventional machine-learning approaches to explore natural data in its natural form was limited. Deep learning enables computational models with several processing layers to learn and represent data at multiple levels of abstraction, simulating how the brain receives and analyses multi-modal information, and so implicitly capturing intricate data structures [1]. A convolutional neural network (CNN) is one of the most popular deep learning models. It uses deep convolutional networks and non-linearity to discover local and spatial features, and patterns directly from raw data. As a result, a CNN learns features from data automatically, eliminating the necessity to manually extract them [2]. Image classification is a vital phenomenon in computer vision and other computer vision approaches, such as localisation, detection, and segmentation are built on top of it [3]. Deep neural networks (DNN) have recently been popular in the deep learning community for solving real-world issues, but the deep networks may face obstacles and hurdles throughout the training process, such as exploding/vanishing gradients and degradation [4]. The deep architecture presents the dedicated concern of training a CNN from scratch, which needs massive computational power, a long training time, and a substantial amount of training data. The specificity of features rises as we progress from lower-level CNN layers to higher-level layers, until the last classification layer becomes profoundly task specific. The image features extracted by the lower-level CNN layers can be used to retrain the model for a completely different task, avoiding the need to start over [5]. In this case, all of the layers of a pre-trained CNN model can be employed as fixed feature extractors, with the exception of the final classification layer. Using the knowledge gained from earlier training, the final layer can be customised and retrained for a new task. When the depth of a network goes beyond the limit, it endures the degradation problem, which results in a decline in accuracy [6]. The internal covariate shift, which is the variation in the dissemination of the input data to a layer during training, is another matter of concern.

Transfer learning is a machine learning technique in which knowledge gained from one type of problem is applied to another similar task or domain [7]. CNN models are normally trained either from scratch or by applying transfer learning. Training from scratch involves a substantial amount of data to learn millions of parameters. Because a sufficiently labelled dataset is required for many applications, CNNs rarely train from scratch. Instead, a large-scale dataset is commonly used to pre-train a CNN, which is subsequently used as a fixed feature extractor or as an initialisation for other particular tasks [2]. The initial few layers of CNN models are trained to recognise task features. In the first layer, pre-trained models learn simple patterns like shapes and diagonals, then combine these components in successive layers to learn multipart features [8]. The models create meaningful constructs in the final layer by exploiting patterns learned from earlier layers. The final few layers of the trained network can be replaced and retrained with new layers for the target activity during transfer learning. Although fine tuned learning experimental studies need some learning, they are still much quicker than learning from the scratch [9], [6].

A. Pre-Trained Deep Learning Architectures

The promotion of artificial neural networks (ANNs) is the deep neural network (DNN), which comprises numerous hidden layers between the input and output layers. A DNN is capable of expressing an object well through its deep architectures and excels at modelling complex nonlinear relationships [10]. In recent times, CNNs have played a critical role in image classification and object detection. In 1998

TABLE I. SUMMARY OF SELECTED PRE-TRAINED CNN MODELS

Pre-Trained Models	Time	Depth	Layers	Image Input Size	Parameters
SqueezeNet [19]	2016	18	68	227-by-227	1.24 M
GoogleNet [20]	2014	22	144	224-by-224	7.0 M
ShuffleNet [21]	2018	50	173	224-by-224	1.4 M
Darknet-53 [22]	2018	53	184	256-by-256	41.6 M
Inception-V3 [23]	2016	48	315	299-by-299	23.9 M

LeCun et al. [11] proposed the first multilayer CNN, which is a convolutional network with seven levels that is simple to use, called LeNet-5. The layers of CNNs have become significantly deeper as GPU technology continues to advance. From 1998 to 2018, a number of CNN frameworks were developed, including LeNet [12], AlexNet [13], VGG 16 and VGG 19 [14], ResNet's Inception ResNet [15], ResNeXt, and other frameworks including PolyNet [16], DenseNet [17]. Transfer learning at a deep level instead of utilising traditional machine learning approaches that benefit from handcrafted features, CNN learns the most representative features from raw data automatically [18]. A variety of CNN architecture modifications with a rapid growth in the number of layers have recently been demonstrated. In this study, five pre-trained models, namely SqueezeNet, GoogleNet, ShuffleNet, Darknet-53, and Inception-v3 have been selected for the performance comparison and a brief summary is provided in Table I.

A system designer must incorporate their judgment and substantial feature engineering to resolve the question of what needs to be transferred. The challenge on how knowledge should be conveyed through is model selection and how to supplement it to enhance prediction performance [11]. When selecting a network to apply to a problem, different aspects of pre-trained models are important to consider. Network accuracy, speed, and size are the most important considerations. Choosing a network is usually a compromise between these factors. The primary goal of this study is to compare the network performance of the selected pre-trained models based on accuracy, speed, and size to help the selection of a suitable model for image classification.

The rest of the paper is organised as follows. In Section II, we give a description of the related works. In Section III, the methodology is described in detail together with the transfer learning steps used in MATLAB. In Section IV, the experimental results are shown, followed by the performance evaluation and performance comparison of the pre-trained deep neural networks. Finally, in Section V, we provide the conclusions and future work.

II. RELATED WORK

In several disciplines, traditional machine learning algorithms have been widely accepted. Deep learning as well as image processing techniques have been used. With the introduction of transfer learning as a new learning framework [24], by fine-tuning pre-trained CNN models that have already been trained on ImageNet, similar results can now be obtained on deep learning applications. These models require a smaller number of training examples than developed models, which necessitate a significant amount of effort to acquire a big number of training instances [25]. Transfer learning has been used in a variety of fields, including agriculture, where it has been used to identify weeds, classify land cover, identify plants, count fruits, and classify crop types. Transfer learning has become increasingly important in medical image processing, while pre-trained deep neural networks have made significant advances in the medical field, including the use of magnetic resonance imaging (MRI) scans, computerised tomography (CT) scans, and electrocardiograms (ECs) to detect life-threatening diseases, such as heart disease, cancer, and brain tumours. Shakil Ahmed et. al. [8] developed a transfer learning-based framework, which was tested against two well-known CNN models, Inception-V3 and VGG-16, using the Kimia Path24 dataset, which was created specifically for the classification and retrieval of histopathological images. Muhammed Talo [26] did the same kind of study with the same Kimia Path24 dataset. ResNet-50 and DenseNet-161, however, were used as well-known pre-trained CNN models. Rishav Singh et. al [11] presented a framework based on the concept of transfer learning to address and focus efforts on histopathology and unbalanced image classification, employing the widely used VGG-19 as a base model.

Samuel Kumaresan et. al. [18], suggested employing transfer learning to overcome the issue of a small dataset of welding defect X-ray pictures. They used two large pre-trained convolutional neural networks, VGG16 and ResNet50, to extract features from weld defect radiograph images that can be used to classify 14 different types of weld defects. The goal for Edna Chebet Too et. al. [6] was to fine-tune and explore the deep convolutional neural network for image-based plant disease classification. The models VGG 16, Inception V4, ResNet with 50, 101, and 152 layers, and DenseNet with 121 layers were assessed in an empirical comparison of the deep learning architectures. Jianping Ju et. al. [27] in an effort to address the actual demand for jujube fault detection, introduced a jujube sorting model in small data sets based on convolutional neural networks and transfer learning using the SE-ResNet50-TL and SE-ResNet50-CL models. Triplet loss function and Center loss function were used to replace SoftMax loss function and embedded SE module for the dry red date defect detection. Alper et. al. [28] recommended a new CNN architecture for the hazelnut variety classification and the model was compared with four pre-trained models: VGG16, VGG19, InceptionV3, and ResNet50. In recent years, the difficulty of layer selection when using transfer learning with fine-tuning has received substantial attention. With the widespread adoption of deep learning techniques, transfer learning with fine-tuning appeared to be the ultimate approach for transferring knowledge, allowing scientists and professionals to apply such deep learning methods more quickly to a variety of domain problems [29].

III. METHODOLOGY

Classification has been an effective mission, and it is important in the subject of computer vision, which seeks to classify images into predefined classes automatically. Prior to the boom of deep learning approaches, a lot of effort was invested into constructing scale-invariant features, feature representations, and image classification classifiers [30]. These well-crafted qualities, on the other hand, work against objects in natural images with complex scenes, varying colour, texture, and illumination, as well as constantly changing positions

and view parameters. Researchers have been working on sophisticated ways to increase image classification accuracy for decades. When the large-scale image dataset ImageNet was formed in 2009, Feifei Li [31] created the great-leap-forward advancement of image classification. The information about dataset, learning environment, gradient, learning rate, epoch, and mini-batch size employed in MATLAB, and the steps of transfer learning used in this study are explained under the methodology in the subsequent sections.

A. Dataset

The CIFAR-10 [32] dataset has 32 x 32 colour images that are divided into ten classes, each with 5,000 training images and 1,000 test images. Among the ten classes, five classes have been selected for the experimental process. From the training dataset, 3,000 training images are selected for each of the five classes, which includes the list as shown below:

Selected Classes = ('bird', 'cat', 'deer', 'dog', 'horse');

The most widely used split ratios are 70:30; 80:20; 65:35; 60:40 etc., in which the sample size suits the nature of the problem and the architecture implemented. There is no fixed law for dividing training and trial datasets when it comes to data splitting. Some scholars have traditionally used the 70:30 ratio to differentiate the datasets. As most widely used in MATLAB, the training set of images is split into training set and validation set by the 70:30 ratio.

[imdsTrain, imdsValidation] = splitEachLabel(imds, 0.7);

Besides that, image augmentation could be used at random on the training datasets with distinct values to help expand the dataset, preventing the network from overfitting and capturing the exact features of the training images. The following settings for image augmentation have been chosen, as indicated in most of the MATLAB examples: horizontal reflection, horizontal and vertical translation in the range [-30 30] pixels, and horizontal and vertical scaling in the range [0.9 1.1] with a random rate.

B. Learning of the Pre-Trained Networks

- Environment – The networks are implemented in MATLAB R2021a. The size of input images is adjusted to match the layers of various models.
- Stochastic Gradient Descent with Momentum (SGDM) – Gradient descent [10] is a popular neural network optimisation approach that can tackle a variety of trivial issues. When the training dataset is huge, however, the simple gradient descent method may use a lot of processing resources, making the convergence process slow. Simultaneously, because the gradient descent approach considers all of the training data for each calculation, it may result in overfitting. To resolve this challenging dispute, SGDM has been considered in this study. Momentum [33] is a commonly used acceleration technique in the gradient descent method whereby the convergence process can be accelerated.
- Learning Rate (LR) – When it comes to CNN training, LR is a crucial parameter. The LR is frequently decreased by a factor of 0.1 or 0.5. In this study, fixed learning rates of LR-0.001 and LR-0.0001 have been chosen instead of reducing the LR by each epoch.
- Epoch – The complete pass of the training algorithm across the entire training set is referred to as an epoch. In this study, the selected epoch values are 10, 20, 30.
- Mini Batch Size – A mini-batch is a subset of the training set that is utilised to calculate the loss function's gradient and update the weights. Two batch sizes are selected as part of the experimentation process: 32 and 64.

C. Transfer Learning Flow in MATLAB

CNN's unique qualities, such as incremental feature extraction in subsequent layers, make it possible to use parts of a pre-trained model for a completely new task without retraining the entire network [34], [35]. The fundamental idea is to use the initial layers from a pre-trained model and just retrain the last few layers on new images. Transfer learning can be implemented by replacing the output layer with a new classifier and then, selecting one of the two approaches:

- Fixed feature extraction by freezing the initial layers or other layers of the convolutional base.
- Fine-tuning the weights and other parameters to retrain one or more convolution layers.

The entire experimentation process of image classification with the dataset CIFAR-10 has been done with MATLAB. The CIFAR-10 dataset is downloaded and provided as input data to the pre-trained model. Prior to loading the data, the entire dataset is divided into three main datasets comprising the training, validation, and testing datasets. To get good performance, deep neural networks require a vast amount of training data. Image augmentation, such as reflection, translation, and scaling, are used to increase the performance of deep networks in order to develop an effective image classifier with little training data. The pre-trained network is loaded, and the final layers are replaced with a new classification layer and a fully connected or convolutional layer. To fine-tune the model, the initial layers of each network are frozen and other parameters like the pool size, stride etc., are updated. The freezing layers are chosen according to the depth, size, and number of layers of the pre-trained network. Afterwards, the training process is initiated, followed by the classification of the validation, and test images. Finally, the classification accuracy is computed and performance of the networks are evaluated using confusion matrix. The transfer learning steps are illustrated in Fig. 1 which is then followed by the detailed steps performed in MATLAB to implement the entire transfer learning process.

First, training and validation sets are used to perform the training process to make sure that we have the best possible training model in the study. By checking the accuracy and loss of training and validation sets, we will be able to control the model's performance during training. Thus, the best possible model can be obtained by fine-tuning at the end of the training process. The results are evaluated by using the test set and these procedures were applied for each of the five models used in the study. All parameters are used in the same way for each model and the models used were evaluated using confusion

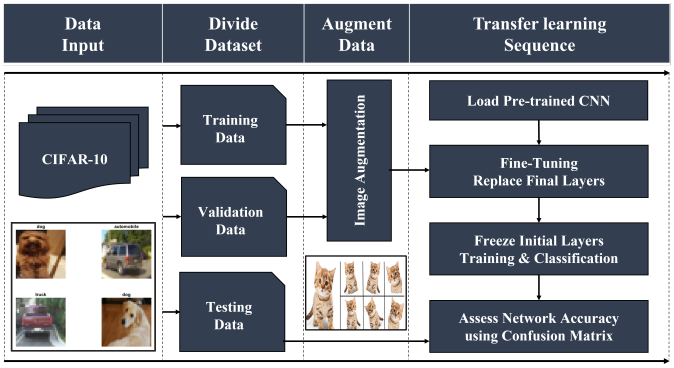


Fig. 1. Flow of Transfer Learning Sequence.

Transfer Learning Steps - MATLAB

Prepare Data
1) Downloading the data.
2) https://www.cs.toronto.edu/ kriz/cifar.html.
Load Data
3) Selection of classes from the downloaded data.
4) imds = imageDatastore(fullfile(rootFolder));
Load Pretrained Network
5) net = SqueezeNet | GoogleNet | ShuffleNet | Darknet-53 | Inception-v3;
6) Analyze the network.
Replace Final Layers
7) lgraph = layerGraph(net);
8) lgraph = replaceLayer(lgraph, learnableLayer.Name, newLearnableLayer);
Freeze Initial Layers
9) layers = lgraph.Layers;
10) layers(1:10) = freezeWeights(layers(1:10));
Train Network
11) Training options: ['MiniBatchSize', 'MaxEpochs', 'InitialLearnRate'];
12) net = trainNetwork(augimdsTrain, lgraph, options);
Classify Images
13) Validation, Testing.
Accuracy and Loss Plot
14) Validation.
Confusion Matrix
15) cm = confusionmat(trueLabels, predictedlLabels);
16) cm_chart = confusionchart(trueLabels, predictedlLabels);
Reset GPU

matrix to find out the performance of the classifier. The performance evaluation used in the study and the comparison on the performance of different models are presented in Table VIII and Table IX in the following section.

IV. Result and Discussion

The whole experimental process was carried out with a laptop and the experimental setup including the hardware, software, and its specifications are mentioned in Table II.

A. Experimental Results

In the training process, each iteration involves a gradient estimation and a network parameter update. Training can be tracked in MATLAB to determine how quickly the network's accuracy improves, as well as if the network attempts to overfit the training data. Following the completion of the training, the results can be inspected to see the finalised validation accuracy and to discover how the training was proceeded by plotting the key metrics, which include training accuracy, validation

TABLE II. Experimental Setup

Hardware/Software	Specifications
Microprocessor	AMD Ryzen 7 5800H- Radeon Graphics@3.20 GHz
RAM	16.0 GB
GPU	NVIDIA GeForce RTX 3060 Laptop GPU
Dedicated Video RAM	6.0 GB
Deep Learning Framework	MATLAB R2021a – 64 bit
Programming Language	MATLAB
Operating System	Windows 10 Home Single Language

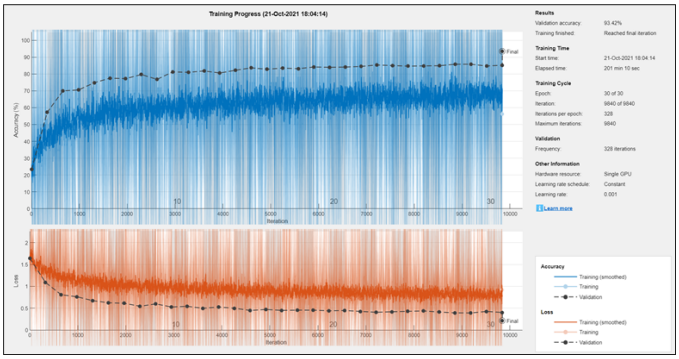


Fig. 2. Training Progress Sample – MATLAB.

accuracy, training loss, and validation loss. Fig. 2 depicts the sample of training progress accomplished with MATLAB, which primarily highlights the results for validation accuracy, training time, training cycle with iterations and epoch, validation, and about the hardware resources. The validation plots are portrayed in Fig. 3, and they contain the validation accuracy, which represents the classification accuracy, and the validation loss, which represents the validation loss across the entire validation set for the five pre-trained networks.

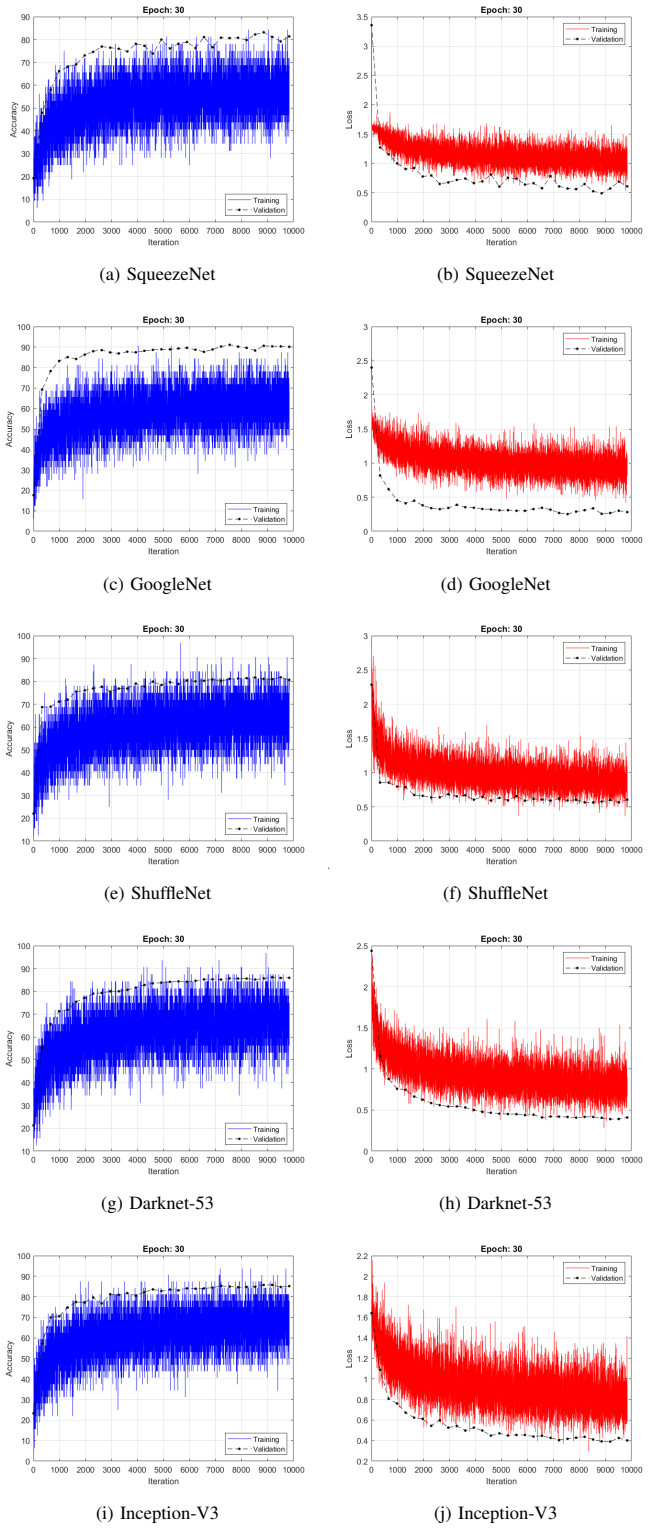
The experimental results are presented in the tables

TABLE III. SqueezeNet – Freezing Layers:[1-11]

Hyper Parameters	Validation Accuracy (%)	Test Accuracy (%)	Time (mins)
LR - 0.001	**Epoch - 30**		
Mini Batch Size- 64	81.87	79.10	21.20
Mini Batch Size- 32	81.40	78.90	20.15
LR - 0.0001	**Epoch - 30**		
Mini Batch Size- 64	72.73	70.78	21.19
Mini Batch Size- 32	78.36	76.28	20.17

TABLE IV. GoogleNet – Freezing Layers:[1-10]

Hyper Parameters	Validation Accuracy (%)	Test Accuracy (%)	Time (mins)
LR - 0.001	**Epoch - 30**		
Mini Batch Size- 64	89.29	88.58	57.28
Mini Batch Size- 32	90.16	88.82	53.37
LR - 0.0001	**Epoch - 30**		
Mini Batch Size- 64	83	83.94	58.21
Mini Batch Size- 32	85.98	86.10	53.51



(a) SqueezeNet (b) SqueezeNet

(c) GoogleNet (d) GoogleNet

(e) ShuffleNet (f) ShuffleNet

(g) Darknet-53 (h) Darknet-53

(i) Inception-V3 (j) Inception-V3

Fig. 3. Validation Plots - Batch-32.

TABLE V. SHUFFLENET – FREEZING LAYERS:[1-15]

Hyper Parameters	Validation Accuracy (%)	Test Accuracy (%)	Time (mins)
LR - 0.001	**Epoch - 30**		
Mini Batch Size- 64	88.93	87.18	97.21
Mini Batch Size- 32	88.07	85.08	111.29
LR - 0.0001	**Epoch - 30**		
Mini Batch Size- 64	79.33	78.26	98.70
Mini Batch Size- 32	85.02	82.54	113.11

TABLE VI. DARKNET-53 – FREEZING LAYERS:[1-14]

Hyper Parameters	Validation Accuracy (%)	Test Accuracy (%)	Time (mins)
LR - 0.001	**Epoch - 30**		
Mini Batch Size- 64	Error@19/30 (Out of Memory)		
Mini Batch Size- 32	89.89	86.62	207.13
LR - 0.0001	**Epoch - 30**		
Mini Batch Size- 64	Error@19/30 (Out of Memory)		
Mini Batch Size- 32	90.58	86.76	203.30

such as Table III-SqueezeNet, Table IV-GoogleNet, Table V-ShuffleNet, Table VI-Darknet-53 and Table VII-Inception-V3, including the hyperparameters such as mini-batch size, learning rate (LR), epoch as well as the validation accuracy and testing accuracy with the elapsed time to complete the training progress. The experimental findings made it possible to emphasise the following outcomes,

- Epoch-30 was chosen for further comparison based on the experimental findings, and the results were quite promising.
- When it comes to mini batch sizes, batch 32 has shown to be more promising than batch 64. Also, with Darknet-53, batch 64 displayed an error due to a lack of RAM (out of memory); hence batch 32 was chosen for further evaluation and comparison.
- With the exception of Darknet-53, LR-0.001 yielded favourable results when compared to LR-0.0001. For the subsequent studies, LR-0.001 findings were chosen for the other four networks, and LR-0.0001 results for Darknet-53.
- Out of the five pre-trained networks, Inception-V3 produced the best results, with the most layers, while SqueezeNet produced the unpleasant results, with the

TABLE VII. INCEPTION-V3 – FREEZING LAYERS:[1-41]

Hyper Parameters	Validation Accuracy (%)	Test Accuracy (%)	Time (mins)
LR - 0.001	**Epoch - 30**		
Mini Batch Size- 64	92.78	91.60	165.16
Mini Batch Size- 32	93.42	92.46	201.10
LR - 0.0001	**Epoch - 30**		
Mini Batch Size- 64	80.29	76.54	165.80
Mini Batch Size- 32	87.53	83.86	206.70

fewest layers.

B. Performance Evaluation using Confusion Matrix

The ratio between the number of right predictions made and the total number of predictions produced is known as classification accuracy [18]. The learning performance of the pre-trained deep neural networks is assessed using a standard confusion matrix method. A confusion matrix is a summary of classification problem prediction outcomes. It provides insight into correct and incorrect classifications, as well as the types of errors made, for each specific class. In image classification, the confusion matrix is primarily used to compare the classification to the actual measurement value in order to intuitively and accurately describe the accuracy of model classification [36]. The confusion matrix can be used to directly identify the performance of deep CNN models, and the evaluation metrics are listed below:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

where ACC stands for accuracy, which is defined as the percentage of correctly classified samples when a measured value is compared to a known value.

$$PREC = \frac{TP}{TP + FP} \tag{2}$$

where PREC is the precision used to determine the model's ability to correctly classify positive values.

$$SENS = \frac{TP}{TP + FN} \tag{3}$$

where SENS is the sensitivity, also known as recall, which is the frequency with which the model correctly predicts positive values. It's used to figure out how well the model can predict positive values.

$$SPEC = \frac{TP}{TN + FP} \tag{4}$$

where SPEC denotes the specificity with which the model's ability to predict negative values.

$$F1 - Score = \frac{2 * PREC * SENS}{PREC + SENS} \tag{5}$$

whereas the harmonic mean of the precision and sensitivity is the F1-score, also known as the balanced F-score or F-measure.

In MATLAB, the predicted class is represented by the rows, while the true class is represented by the columns. The diagonal cells relate to accurately classified observations. The off-diagonal cells correspond to observations that were inaccurately classified. The number of accurately and inaccurately classified observations for each predicted class are displayed as percentages of the total number of observations in the respective predicted class in a column-normalized column summary. The number of accurately and inaccurately classified observations for each true class are displayed as percentages of the total number of observations for that true class in a row-normalized row summary.

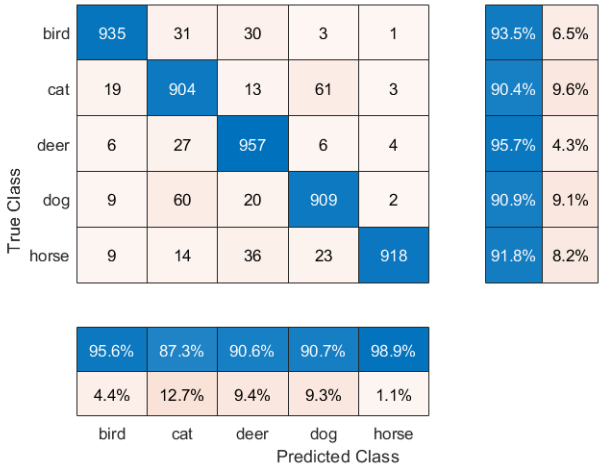
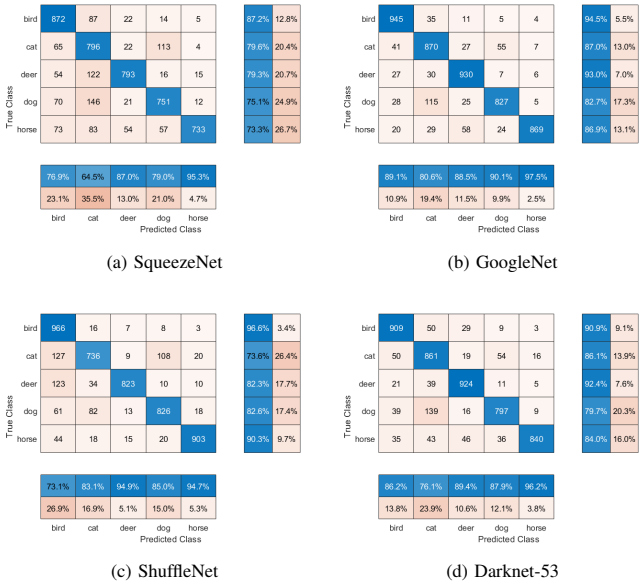


Fig. 4. Confusion Matrix of Inception-v3 - LR-0.001.



(a) SqueezeNet (b) GoogleNet

(c) ShuffleNet (d) Darknet-53

Fig. 5. Confusion Matrices for LR-0.001|Batch-32.

The Fig. 4 demonstrates the confusion matrix for the pre-trained network Inception-V3 and the Fig. 5 represents the confusion matrices for networks such as (a) SqueezeNet, (b) GoogleNet, (c) ShuffleNet and (d) Darknet-53 for the mini-batch 32 and LR-0.001. The Fig. 6 describes the confusion matrix for the pre-trained network Darknet-53 and the Fig. 7 represents the confusion matrices for networks such as (a) SqueezeNet, (b) GoogleNet, (c) ShuffleNet and (d) Inception-V3 for the mini-batch 32 and LR-0.0001. When it came to learning rates, among the selected five pre-trained networks LR-0.0001 was outperformed by LR-0.001. Under LR-0.001, almost all of the networks performed well in classifying the images and prediction, however under LR-0.0001, most of the networks struggled to predict the positive values. Based on the confusion matrices, the following inferences were discovered,

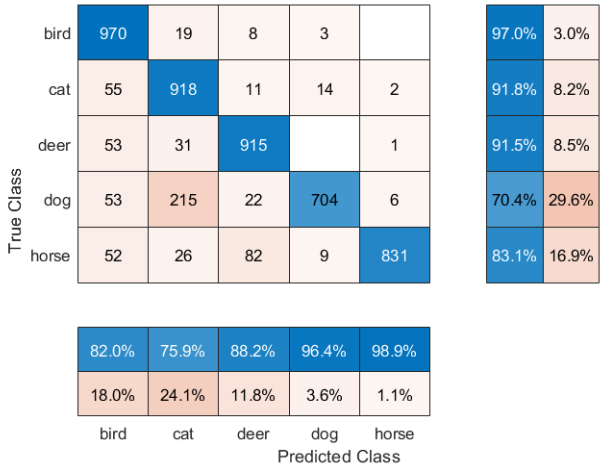
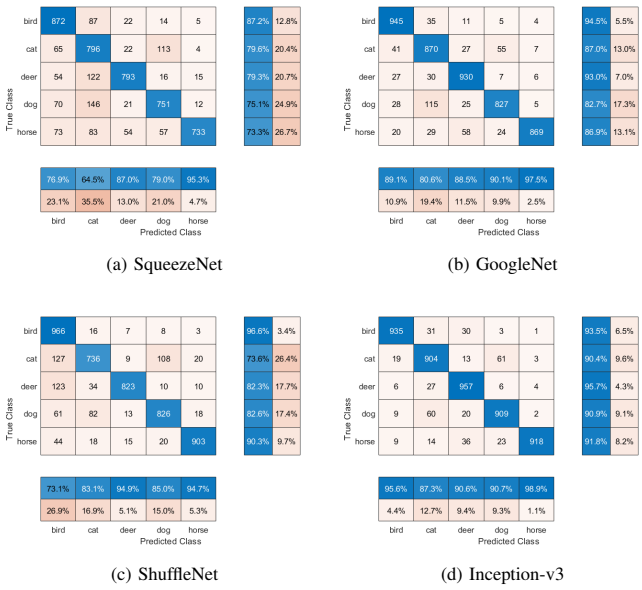


Fig. 6. Confusion Matrix of Darknet-53 - LR-0.0001.



(a) SqueezeNet (b) GoogleNet

(c) ShuffleNet (d) Inception-v3

Fig. 7. Confusion Matrices for LR-0.0001|Batch-32.

- In terms of LR – 0.001, Inception-V3 outperformed the rest of the pre-trained networks in the model's ability to predict positive values followed by GoogleNet and Darknet-53. With Inception-V3, all five classes were correctly classified with an overall accuracy of above 90%. Among the classes deer class made the highest score whereas 957 out of 1000 images were classified correctly. When it comes to prediction, the horse class scored high of correctly predicting 98.9% of the positive values. For all the five classes, GoogleNet scored 80% or higher, with the horse class getting the highest prediction score, predicting 97.5% of positive values. Darknet-53 scored the highest among the networks a prediction score of 96.2% in the horse class but got a very least score of only 76.1% in the cat class. ShuffleNet had a mediocre performance, scoring 94.7% in the horse class and a very low prediction score of 73.1% in the bird class. SqueezeNet had the lowest classification performance of all the networks, with a prediction score of 95.3% in the horse category and 64.5% in the cat category.
- In terms of LR – 0.0001, Darknet-53 outperformed the rest of the pre-trained networks in the model's ability to predict positive values followed by Inception-V3 and GoogleNet. If compared with LR 0.001, Darknet-53 scored the maximum classification accuracy under LR 0.0001 with the highest score of 97% among all the networks. With prediction, scored the highest of 98.9% in horse class and 75.9% in cat class. The bird class received the highest classification score of 93% in Inception-V3, with 930 out of 1000 images correctly classified, whereas the model struggled to predict the positive values of the bird class, accounting for 77.1%. In GoogleNet among the five classes, horse class got the highest prediction score of 91.1% and cat class got the lowest score of 81.7%. ShuffleNet had an average classification compared to other networks whereas horse class got the highest prediction score of 91.5% and cat class got the lowest score of 76.1%. SqueezeNet had the lowest performance among all other networks with the lowest prediction score of 66.0% for the bird class whereas got the highest score of 90.7% for the horse class.

TABLE VIII. EVALUATION RESULTS OF THE PRE-TRAINED NETWORKS

Pre-Trained Models	ACC (%)	PREC (%)	SENS (%)	SPEC (%)	F1-Score (%)
LR - 0.001			**Epoch - 30**		
SqueezeNet	91.56	80.53	78.90	94.73	79.16
GoogleNet	95.53	89.16	88.82	97.21	88.85
ShuffleNet	94.03	86.15	85.08	96.27	85.13
Darknet-53	94.65	87.15	86.62	96.65	86.68
Inception-V3	**96.98**	**92.63**	**92.46**	**98.12**	**92.49**
LR - 0.0001			**Epoch - 30**		
SqueezeNet	90.51	77.89	76.28	94.07	76.14
GoogleNet	94.44	86.14	86.10	96.53	86.06
ShuffleNet	93.02	82.76	82.54	95.63	82.57
Darknet-53	**94.70**	**88.29**	**86.76**	**96.69**	**86.70**
Inception-V3	93.54	84.55	83.86	95.96	83.91

The results of the classification metrics evaluation of the five pre-trained networks for both the Learning Rates of 0.001 and 0.0001 are summarized in Table VIII. According to the evaluation results it is evident that the networks performed well on the LR-0.001 in compared to LR-0.0001 except for Darknet-53. Darknet-53, in compared to the other four networks, showed promising results with a LR-0.0001, whilst the other networks' performances were on the decline.

C. Pre-Trained Networks Performance Comparison

The performance comparison of the five pre-trained networks, encompassing both LRs, is shown in Table IX. Based on the performance comparison of the pre-trained networks, Inception-V3 has achieved the highest accuracy of 96.98%, as well as other metrics such as precision, sensitivity, specificity,

TABLE IX. PERFORMANCE COMPARISON OF THE PRE-TRAINED NETWORKS

Pre-Trained Models	ACC (%)	PREC (%)	SENS (%)	SPEC (%)	F1-Score (%)
SqueezeNet	91.56	80.53	78.90	94.73	79.16
GoogleNet	95.53	89.16	88.82	97.21	88.85
ShuffleNet	94.03	86.15	85.08	96.27	85.13
Darknet-53	94.70	88.29	86.76	96.69	86.70
Inception-V3	**96.98**	**92.63**	**92.46**	**98.12**	**92.49**

and f1-score. The other networks produced somewhat lower results than Inception-V3, but altogether, all five pre-trained networks attained an accuracy of 90% or higher.

Transfer learning using a pre-trained CNN model is a better option for classification with the availability of only small datasets. In many of the previous studies, different pre-trained CNN models were compared using medical images and other relevant datasets. The results showed that the performance of the pre-trained models were mainly based on the dataset. With the limited computing resources, only five classes of the benchmark CIFAR-10 dataset are selected, but still managed to accomplish the aim of this study in the selection of a suitable model for image classification, Inception-V3. Even though one of the CNN model darknet-53 produced an error (out of memory) over batch-64, the current study shows that pre-trained networks with the highest number of layers (Darknet-53 and Inception-V3) provided the maximum scores in the prediction of classes with best accuracy. The current study proves that transfer learning can be useful for various computer vision problems, especially for the ones with small datasets. With the availability of proper datasets, deep CNN models have the capabilities to take medical imaging technology further, providing a higher level of automation in medical imaging, including image processing and analysis.

V. CONCLUSIONS

In this study, we experimented with the performance of five pre-trained networks, such as SqueezeNet, GoogleNet, ShuffleNet, Darknet-53, and Inception-V3 with different epochs, learning rates, and mini-batch sizes. We performed the entire training process in MATLAB R2021a where we can view the complete network architecture of the CNN models, which helped us in the selection of freezing the initial layers. The final layers of the pre-trained CNN models are replaced either with a fully connected layer or convolutional layer and a new classifier replacing the classification layer. The initial layers are frozen to keep the weights intact and after the training, each model was evaluated using a confusion matrix. The experimental findings show that each pre-trained network produced different results with different hyper-parameters in the prediction of positive values. The results demonstrate that all the five pre-trained networks yielded promising results over the mini batch size-32, and epoch-30. In terms of LR, Darknet-53 delivered impressive results with LR-0.0001, achieving a maximum accuracy of 94.70%. Overall, the Inception-V3 model with LR-0.001 achieved the highest accuracy of 96.98%, as well as other evaluation metrics including precision, sensitivity, specificity, and f1-score of 92.63%, 92.46%, 98.12%, and 92.49%, respectively.

VI. FUTURE WORK

We presented a transfer learning-based performance comparison between the selected five pre-trained networks in this study. The freezing of network layers was selected based on the network depth, size, and number of layers. Only the initial layers were frozen; however, different sets of layers can be frozen. In the future, focus will be given to freeze multiple set of layers and to compare the results of frozen and non-frozen layers of the pre-trained networks. For further evaluation and comparison, different datasets and other pre-trained deep neural networks can also be explored.

ACKNOWLEDGMENT

The authors are grateful to Universiti Teknologi Malaysia (UTM) for supporting the work with cost center number Q.K130000.2656.17J22.

REFERENCES

[1] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational intelligence and neuroscience*, vol. 2018, 2018.

[2] H. Altaheri, M. Alsulaiman, and G. Muhammad, "Date fruit classification for robotic harvesting in a natural environment using deep learning," *IEEE Access*, vol. 7, pp. 117 115–117 133, 2019.

[3] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural computation*, vol. 29, no. 9, pp. 2352–2449, 2017.

[4] K. Goutam, S. Balasubramanian, D. Gera, and R. R. Sarma, "Layerout: Freezing layers in deep neural networks," *SN Computer Science*, vol. 1, no. 5, pp. 1–9, 2020.

[5] U. A. Khan, M. A. Martinez-Del-Amor, S. M. Altowaijri, A. Ahmed, A. U. Rahman, N. U. Sama, K. Haseeb, and N. Islam, "Movie tags prediction and segmentation using deep learning," *IEEE Access*, vol. 8, pp. 6071–6086, 2020.

[6] E. C. Too, L. Yujian, S. Njuki, and L. Yingchun, "A comparative study of fine-tuning deep learning models for plant disease identification," *Computers and Electronics in Agriculture*, vol. 161, pp. 272–279, 2019.

[7] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[8] S. Ahmed, A. Shaikh, H. Alshahrani, A. Alghamdi, M. Alrizq, J. Baber, and M. Bakhtyar, "Transfer learning approach for classification of histopathology whole slide images," *Sensors*, vol. 21, no. 16, p. 5361, 2021.

[9] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Frontiers in plant science*, vol. 7, p. 1419, 2016.

[10] C. Lu and W. Li, "Ship classification in high-resolution sar images via transfer learning with small training dataset," *Sensors*, vol. 19, no. 1, p. 63, 2019.

[11] R. Singh, T. Ahmed, A. Kumar, A. K. Singh, A. K. Pandey, and S. K. Singh, "Imbalanced breast cancer classification using transfer learning," *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 18, no. 1, pp. 83–93, 2020.

[12] Y. LeCun *et al.*, "Lenet-5, convolutional neural networks," *URL: http://yann. lecun. com/exdb/lenet*, vol. 20, no. 5, p. 14, 2015.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

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

[15] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[16] X. Zhang, Z. Li, C. Change Loy, and D. Lin, "Polynet: A pursuit of structural diversity in very deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 718–726.

[17] G. Huang, S. Liu, L. Van der Maaten, and K. Q. Weinberger, "Condensenet: An efficient densenet using learned group convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2752–2761.

[18] S. Kumaresan, K. J. Aultrin, S. Kumar, and M. D. Anand, "Transfer learning with cnn for classification of weld defect," *IEEE Access*, vol. 9, pp. 95 097–95 108, 2021.

[19] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[21] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6848–6856.

[22] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[23] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[24] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *International conference on artificial neural networks*. Springer, 2018, pp. 270–279.

[25] M. Boulares, T. Alafif, and A. Barnawi, "Transfer learning benchmark for cardiovascular disease recognition," *IEEE Access*, vol. 8, pp. 109 475–109 491, 2020.

[26] M. Talo, "Automated classification of histopathology images using transfer learning," *Artificial Intelligence in Medicine*, vol. 101, p. 101743, 2019.

[27] J. Ju, H. Zheng, X. Xu, Z. Guo, Z. Zheng, and M. Lin, "Classification of jujube defects in small data sets based on transfer learning," *Neural Computing and Applications*, pp. 1–14, 2021.

[28] A. Taner, Y. B. Öztekin, and H. Duran, "Performance analysis of deep learning cnn models for variety classification in hazelnut," *Sustainability*, vol. 13, no. 12, p. 6527, 2021.

[29] G. Vrbančič and V. Podgorelec, "Transfer learning with adaptive fine-tuning," *IEEE Access*, vol. 8, pp. 196 197–196 211, 2020.

[30] X. Feng, Y. Jiang, X. Yang, M. Du, and X. Li, "Computer vision algorithms and hardware implementations: A survey," *Integration*, vol. 69, pp. 309–320, 2019.

[31] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[32] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[33] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48.

[34] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" *arXiv preprint arXiv:1411.1792*, 2014.

[35] A. Brodzicki, J. Jaworek-Korjakowska, P. Kleczek, M. Garland, and M. Bogyo, "Pre-trained deep convolutional neural network for clostridioides difficile bacteria cytotoxicity classification based on fluorescence images," *Sensors*, vol. 20, no. 23, p. 6713, 2020.

[36] H. Pan, Z. Pang, Y. Wang, Y. Wang, and L. Chen, "A new image recognition and classification method combining transfer learning algorithm and mobilenet model for welding defects," *IEEE Access*, vol. 8, pp. 119 951–119 960, 2020.