Transfer Learning Based Network Performance Comparison of the Pre-Trained Deep Neural Networks using MATLAB

Senthil Kumar Jayapalan[1], Syahid Anuar[2]

[1,2]Razak Faculty of Technology and Informatics, Universiti Teknologi Malaysia, Kuala Lumpur, Malaysia
kjsenthil@graduate.utm.my[1], syahid.anuar@utm.my[2]

Article history

Received:
02 Dec 2021

Received in revised form:
15 Jan 2022

Accepted:
15 Feb 2022

Published online:
20 May 2022

*Corresponding author
syahid.anuar@utm.my

Abstract

Deep learning has grown tremendously in recent years, having a substantial impact on practically every discipline. The performance of the neural network will improve as the depth of the network is increased, but this progress will come at the cost of time and processing resources. Transfer learning allows us to transfer the knowledge of a model that has been formerly trained for a particular job to a new model that is attempting to solve a related but not identical problem. Specific layers of a pretrained model must be retrained while the others must remain unmodified in order to adapt it to a new task effectively. When faced with a challenge selecting which layers should be enabled for training and which should be frozen, this adaptation is commonly made employing fine-tuning procedures. Furthermore, similar to traditional deep neural network training, there is a typical issue with setting hyper-parameter values. All of these concerns have a substantial effect on training capabilities as well as classification performance. In this study, 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 in order to evaluate and compare the network's performance using confusion matrix. Based on the findings, Inception-V3 has achieved the highest accuracy of 96.98%, as well as other evaluation metrics including precision of 92.63%, sensitivity of 92.46%, specificity of 98.12%, and f1-score of 92.49 %, respectively.

Keywords: *Deep Learning, Transfer Learning, Convolutional Neural Network, Image Classification, Computer Vision.*

1. Introduction

The primary evolution of neural networks was stimulated by the desire to design a system that could imitate the human brain. The MCP model of a neuron, developed by McCulloch and Pitts, has made a significant contribution to the development of artificial neural networks [1]. The ability of conventional machine-learning approaches to analyze natural data in its raw form was limited. For decades, devising a feature extractor that turned raw data into an appropriate internal representation or feature vector from which the learning subsystem, frequently a classifier, could detect or classify patterns in the input needed precise engineering and extensive field expertise [2]. Representation learning is a set of techniques that allow a machine to be fed raw data and automatically learn the representations essential for detection or classification. Deep learning methods are representation-learning methods that have numerous levels of representation. Deep learning

* *Corresponding author. syahid.anuar@utm.my*

enables computational models with several processing layers to learn and represent data at multiple levels of abstraction, simulating how the brain receives and analyses multimodal information, and so implicitly capturing intricate data structures [3]. A CNN is one of the most popular deep learning models. It learns local and spatial features and patterns directly from raw data like as images, video, text, and sound using deep convolutional networks and non-linearity. As a result, a CNN learns features from data automatically, removing the need to manually extract them [4]. Through a sequence of successive convolutional layers, it can create complex features by combining simple features. The deeper layers learn to recognise complicated high-intensity features such as whole objects in an image, while the early layers learn to recognise low-intensity features such as edges and curves. Due to the accessibility of a huge number of annotated images, such as ImageNet, and significant processing power devices, such as GPUs or distributed huge-quantity clusters employing cloud computing, in these tasks, deep learning has a high rate of success. Its success had previously been limited by the need for big databases and extended training cycles. However, transfer learning and fine-tuning techniques were used to alleviate these issues. Transfer learning is a machine learning technique in which knowledge gained from one type of problem is applied to another similar task or domain [5].

1.1 Research Background

Image classification is a fundamental phenomenon in computer vision. Other computer vision approaches such as localization, detection, and segmentation are built on top of it [6]. Deep neural networks (DNN) have recently been popular in the deep learning community for solving real-world issues like image classification, language translation, object identification, and speech recognition. Deeper neural networks, on the other hand, have been shown to have the unfavourable attribute of being prone to over-fitting [7]. Furthermore, the computing cost associated with training a deeper network is not insignificant. This makes deploying deeper models in a real world, such as interactive mobile applications, autonomous driving, and so on, a difficult process. The deep architecture also presents the dedicated concern of training a CNN from scratch, which necessitates massive computer power, a long training time, and a large amount of training data. The knowledge obtained by a CNN pertaining to a certain problem is transferrable to another problem, similar to human learning. 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 [8]. 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 customized and retrained for a new task. Deep networks may face obstacles and hurdles throughout the training process, such as exploding/vanishing gradients and degradation. When the depth of a network exceeds the maximum, it suffers from the degradation problem, which results in a decline in accuracy [9]. The internal covariate shift, which is the change in the distribution of the input data to a layer during training, is another matter of concern. However, a number of optimization techniques, such as skip connections, transfer learning, initialization strategies,

optimization strategies, batch Normalization, and layer-wise training, have been presented to address the issues and challenges successfully. Transfer learning is a technique that may be used to retrain a CNN model that has been trained on a big dataset.

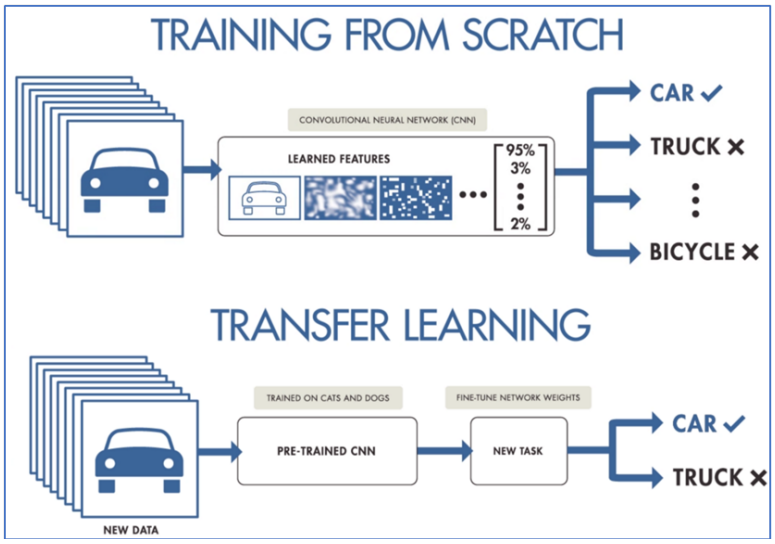

Figure 1. Transfer Learning Method

Transfer learning is frequently employed in a variety of circumstances. 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 [10]. The models create meaningful constructs in the final layer by exploiting patterns learned from earlier layers. Fine-tuning is a transfer learning concept. The initial few layers of deep learning are trained to recognise task features. During transfer learning, you can delete the last few layers of the trained network and retrain with new layers for the target job [9]. Although fine-tuned learning experimental studies need some learning, they are still much quicker than learning from the scratch [11].

Table 1. Summary of Different Pre-Trained CNN Models

Pre-Trained Models	Time	Depth	Layers	Image Input Size	Parameters
SqueezeNet [12]	2016	18	68	227-by-227	1.24 M
GoogleNet [13]	2014	22	144	224-by-224	7.0 M
ShuffleNet [14]	2018	50	173	224-by-224	1.4 M
Darknet-53 [15]	2018	53	184	256-by-256	41.6 M
Inception-V3 [16]	2016	48	315	299-by-299	23.9 M

The promotion of artificial neural networks (ANNs) is the deep neural network (DNN), which comprises many hidden layers between the input and output layers. DNN is capable of expressing an object well through its deep architectures and excels at modelling complex nonlinear interactions [17]. From 1998 to 2018, a number of CNN frameworks were developed, including LeNet, AlexNet, VGG 16, VGG 19, GoogleNet, DenseNet etc. [18]. A system designer must incorporate their judgement and substantial feature engineering to resolve the question of what needs

to be transferred. The challenge on how should knowledge be conveyed through is model selection and how to supplement it to enhance prediction performance [19]. 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 research is to compare the network performance of the selected pre-trained models as in Table 1 based on accuracy, speed, and size in order to help the selection of a suitable model for image classification.

2. Related Works

With the introduction of transfer learning as a new learning framework, 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 [20]. 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. [10] 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 [21] 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. Samuel Kumaresan et. al. [22], to overcome the issue of a small dataset of welding defect X-ray pictures, transfer learning was employed. Jianping Ju et. al. [23] to address the actual demand of jujube faults detection, introduced a jujube sorting model in small data sets based on convolutional neural networks and transfer learning to address the classification requirements of dry red date defect detection. 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 has emerged as the most common strategy for transferring knowledge in the context of deep learning, allowing researchers and practitioners to apply such deep learning methods more quickly to a variety of domain problems [24].

3. Materials and Methods

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 [25]. These well-crafted qualities, on the other hand, work against objects in natural images with a 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 created the great-leap-forward advancement of image classification [26]. The process of correctly training a classifier is time-consuming and necessitates huge datasets [27]. The learning environment, gradient, learning rate, epoch, and mini batch size employed in MATLAB, and the steps of transfer learning used in this study were explained under materials and methods in the subsequent sections.

3.1 Learning of Pre-Trained CNN's

- **Environment –** The networks are implemented in MATLAB R2021a. The size of input images is adjusted to match with the layers of various models.
- **Stochastic Gradient Descent with Momentum (SGDM) –** Gradient descent is a popular neural network optimization approach that can tackle a variety of trivial issues. SGDM have been considered in this study. Momentum [28] is a commonly used acceleration technique in gradient descent method whereas the convergence process can be accelerated.
- **Learning Rate (LR)** – When it comes to CNN training, LR is a crucial parameter. In this study, fixed LR-0.001 and LR-0.0001 have been chosen instead of reducing the LR by each epochs.
- **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 utilized 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.

3.2 Methodology

The entire experimentation process of image classification with the dataset CIFAR-10 has been done with MATLAB. The transfer learning steps are illustrated in Figure 2 below.

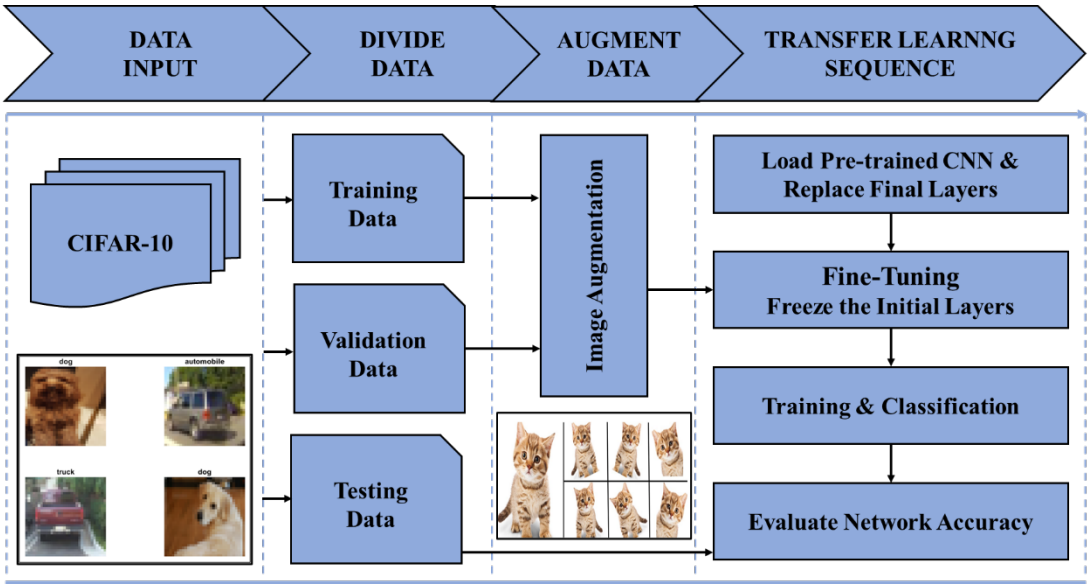

Figure 2. Flow of Transfer Learning Steps

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 refection, 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 a new fully connected classifier replaces the final classification layer. To fine-tune the model, the initial layers of each network are frozen as in MATLAB, allowing you to visually view the entire network layers. 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 training, validation, and test images. Finally, classification accuracy is computed and based on the accuracy that networks are evaluated using a confusion matrix. The detailed steps in the entire transfer learning process performed in MATLAB are presented below.

Transfer Learning Steps - MATLAB

1. Prepare Data
- Downloading the data

2. Load Data
- imds = imageDatastore(fullfile(rootFolder))

3. Load Pretrained Network
- net = SqueezeNet | GoogleNet | ShuffleNet | Darknet-53 | Inception-v3;

4. Replace Final Layers
- lgraph = layerGraph(net);
- lgraph = replaceLayer(lgraph, learnableLayer.Name, newLearnableLayer);

5. Freeze Initial Layers
- layers = lgraph.Layers;
- layers(1:10) = freezeWeights(layers(1:10));

6. Train Network
- Training options: {‘MiniBatchSize’, 'MaxEpochs', 'InitialLearnRate'};
- net = trainNetwork(augimdsTrain, lgraph, options);

7. Classify Images
- Training, Validation, Testing

8. Accuracy and Loss Plot
- Validation

9. Confusion Matrix
- cm = confusionmat(trueLabels, predictedlLabels);
- cm_chart = confusionchart(trueLabels, predictedlLabels);

10. Reset GPU

4. Results and Discussion

4.1 Dataset

The CIFAR-10 [29] dataset has 32 x 32 colour images that are divided into ten classes, each with 5000 training images and 1000 test images. CIFAR-10 contains a total of 50000 training images and 10,000 testing images. Among the ten classes,

five classes have been selected for the experimental process which includes the list as shown below,

Selected Classes = {‘bird’,’cat’,’deer’,’dog’,’horse’};

The major developments in image and video processing rely not only on the development of new learning algorithms and the use of powerful hardware, but also on very large-scale public datasets [30]. The most widely used split ratios are 70:30; 80:20; 65:35; 60:40 etc. whatever 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 are split into training set and validation set by 70:30 ratio,

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

Furthermore, image augmentation was performed on the training datasets at random using different values to enhance the dataset size. The images in each batch were randomly subjected to the following operations during the training phase, as specified in most of the MATLAB examples: horizontal reflection, horizontal and vertical translation with a random value in the range [-30 30] pixels, and horizontal and vertical scaling with a random rate in the range [0.9 1.1].

4.2 Experimental Details

The whole experimental process was carried out with a laptop and the experimental setup including the hardware, software, and its specifications are mentioned in the Table 2 below,

Table 2. 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

4.3 Experimental Results

The tables Table 3 – Darknet-53 and Table 4 – Inception-V3 below are presented with the experimental results, including the hyper-parameters such as mini-batch size, learning rate, epochs as well as the validation accuracy and testing accuracy with the elapsed time to complete the training progress. Out of the five selected pre-trained networks, results of two of the networks are presented in this section which is then followed by the experimental findings.

Table 3. Darknet-53 – Freezing Layers: [1-14]

Hyper Parameters	Validation Accuracy (%)	Test Accuracy (%)	Elapsed 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	203.3

Table 4. Inception-V3 – Freezing Layers: [1-41]

Hyper Parameters	Validation Accuracy (%)	Test Accuracy (%)	Elapsed Time (mins)
LR – 0.001	**Epoch – 30**		
Mini Batch Size - 64	92.78	91.6	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	165.8
Mini Batch Size - 32	87.53	83.86	206.7

The experimental findings made it possible to emphasize 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 worst results, with the fewest layers.

4.4 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 [22]. 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 [31].

The confusion matrix can be used to directly identify the performance of deep CNN models, and the evaluation metrics are listed below,

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

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.

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

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

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

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.

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

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

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

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 on the confusion matrix plot, the predicted class (Output Class) is represented by the rows, while the true class is represented by the columns (Target Class). 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. On the next pages, the confusion matrices for LR-0.001 and LR-0.0001 are presented and discussed.

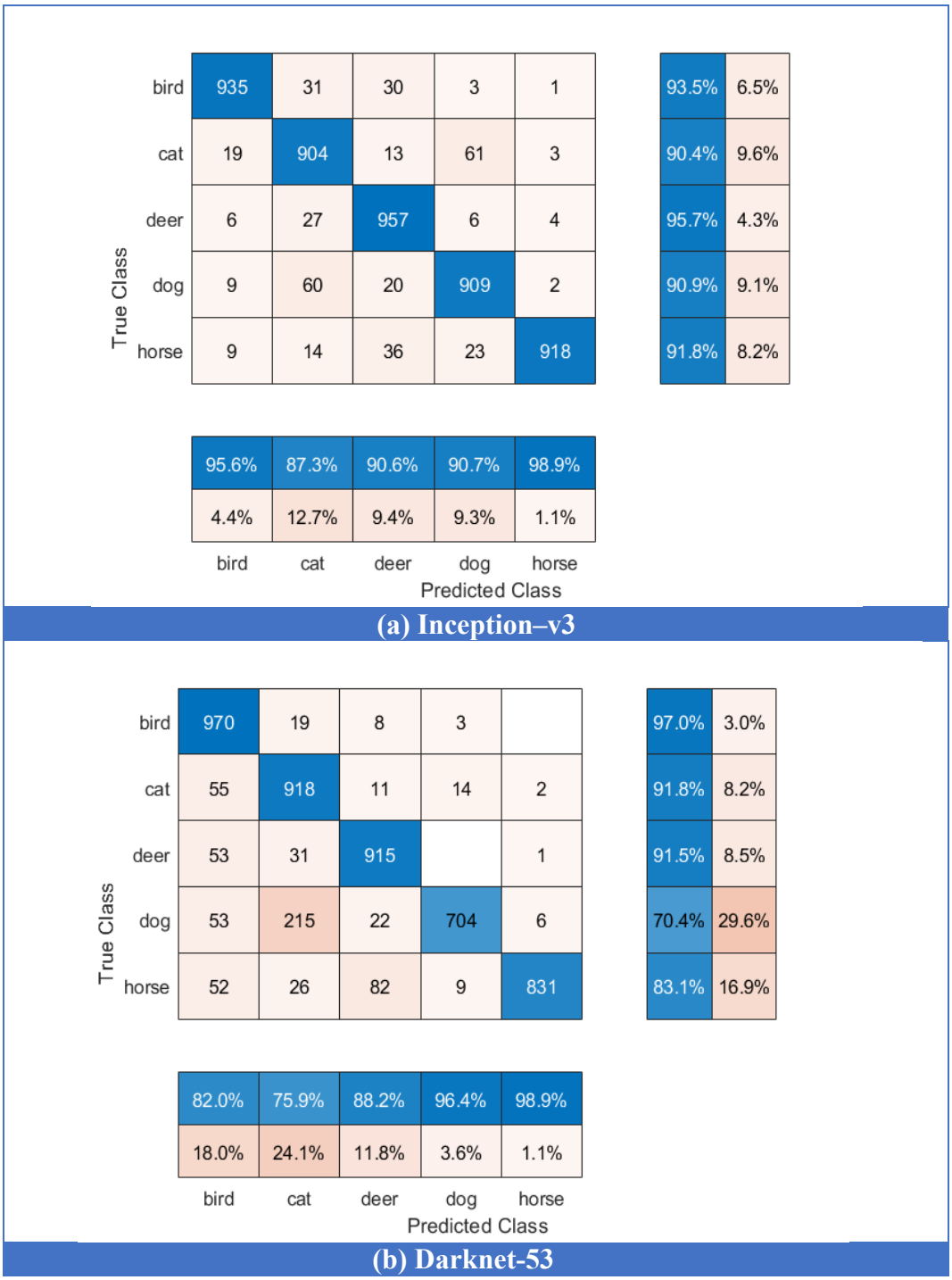

Figure 3. Confusion Matrices for (a) LR-0.001 and (b) LR-0.0001

The above Figure 3 demonstrates the confusion matrices for the pre-trained networks in which (a) represents Inception-V3, and (b) Darknet-53. When it came to LRs, among the 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, 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,

- 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 5. Evaluation Results of the Pre-Trained Networks

CNN Models	Precision (%)	Sensitivity (%)	Specificity (%)	F1-Score (%)	Accuracy (%)
LR – 0.001			**Epoch – 30**		
SqueezeNet	80.53	78.9	94.73	79.16	91.56
GoogleNet	89.16	88.82	97.21	88.85	95.53
ShuffleNet	86.15	85.08	96.27	85.13	94.03
Darknet-53	87.15	86.62	96.65	86.68	94.65
Inception-V3	**92.63**	**92.46**	**98.12**	**92.49**	**96.98**
LR – 0.0001			**Epoch – 30**		
SqueezeNet	77.89	76.28	94.07	76.14	90.51
GoogleNet	86.14	86.1	96.53	86.06	94.44
ShuffleNet	82.76	82.54	95.63	82.57	93.02
Darknet-53	**88.29**	**86.76**	**96.69**	**86.70**	**94.70**
Inception-V3	84.55	83.86	95.96	83.91	93.54

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 the above Table 5. 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.

4.5 Pre-trained Networks Performance Comparison

The performance comparison of the five pre-trained networks, encompassing both LRs, is shown in the Table 6 below.

Table 6. Performance Comparison of the Pre-Trained Networks

CNN Models	Precision (%)	Sensitivity (%)	Specificity (%)	F1-Score (%)	Accuracy (%)
SqueezeNet	80.53	78.9	94.73	79.16	91.56
GoogleNet	89.16	88.82	97.21	88.85	95.53
ShuffleNet	86.15	85.08	96.27	85.13	94.03
Darknet-53	88.29	86.76	96.69	86.70	94.70
Inception-V3	**92.63**	**92.46**	**98.12**	**92.49**	**96.98**

According to the 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, 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.

5. Conclusion

Transfer learning is a method of learning that involves applying previously learned knowledge to new problems. Small datasets can be used to train a deep model through transfer learning and fine-tuning. In this study, the performance of five pre-trained networks such as SqueezeNet, GoogleNet, ShuffleNet, Darknet-53 and Inception-V3 have been experimented with different epochs, LRs and mini batch sizes. The experiment was carried out entirely in MATLAB R2021a, and the pre-trained networks were evaluated using a confusion matrix based on the results. The experimental findings shows that each pre-trained network produced different results with different hyper-parameters in the prediction of positive values of the five classes. All the five networks yielded promising results over the mini batch size-32, and epoch-30. Darknet-53 made an error with the mini batch size-64 due to lack of GPU memory. In order to continue with the evaluation procedure utilizing the confusion matrix, epoch-30 and mini batch size-32 were chosen for the performance comparison. In terms of LR, Darknet-53 delivered impressive results, with LR-0.0001 achieving the maximum accuracy of 94.70%, as shown in Table 5. Overall, the Inception-V3 model had the highest accuracy with LR-0.001 at 96.98%, as well as other measures including 92.63% precision, 92.46% sensitivity, 98.12% specificity, and 92.49% f1-score respectively.

Future Work – In this research, presented a transfer learning performance comparison between the selected five pre-trained networks. The freezing of networks layers was selected based on the network depth, size, and number of layers. Only the initial layers were frozen; however different set of layers can be frozen. In the future research, 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, may also choose between different datasets and other pre-trained deep neural networks.

References

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," Bull. Math. Biol., vol. 52, no. 1, pp. 99–115, 1990.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015.

[3] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep Learning for Computer Vision: A Brief Review," Comput. Intell. Neurosci., vol. 2018, 2018, doi: 10.1155/2018/7068349.

[4] 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. 117115–117133, 2019, doi: 10.1109/ACCESS.2019.2936536.

[5] S. J. Pan and Q. Yang, "A survey on transfer learning," IEEE Trans. Knowl. Data Eng., vol. 22, no. 10, pp. 1345–1359, 2009.

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

[7] K. Goutam, S. Balasubramanian, D. Gera, and R. R. Sarma, "LayerOut: Freezing Layers in Deep Neural Networks," SN Comput. Sci., vol. 1, no. 5, pp. 1–9, 2020, doi: 10.1007/s42979-020-00312-x.

[8] U. A. Khan et al., "Movie Tags Prediction and Segmentation Using Deep Learning," IEEE Access, vol. 8, pp. 6071–6086, 2020, doi: 10.1109/ACCESS.2019.2963535.

[9] E. C. Too, L. Yujian, S. Njuki, and L. Yingchun, "A comparative study of fine-tuning deep learning models for plant disease identification," Comput. Electron. Agric., vol. 161, no. October 2017, pp. 272–279, 2019, doi: 10.1016/j.compag.2018.03.032.

[10] S. Ahmed et al., "Transfer learning approach for classification of histopathology whole slide images," Sensors, vol. 21, no. 16, pp. 1–12, 2021, doi: 10.3390/s21165361.

[11] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," Front. Plant Sci., vol. 7, p. 1419, 2016.

[12] 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 <0.5MB model size," pp. 1–13, 2016.

[13] C. Szegedy et al., "Going deeper with convolutions," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.

[14] 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.

[15] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 2018.

[16] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2016-Decem, pp. 2818–2826, 2016, doi: 10.1109/CVPR.2016.308.

[17] C. Lu and W. Li, "Ship classification in high-resolution SAR images via transfer learning with small training dataset," Sensors (Switzerland), vol. 19, no. 1, 2019, doi: 10.3390/s19010063.

[18] G. Habib and S. Qureshi, "Optimization and acceleration of convolutional neural networks: A survey," J. King Saud Univ. - Comput. Inf. Sci., no. xxxx, 2020, doi: 10.1016/j.jksuci.2020.10.004.

[19] 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 Trans. Comput. Biol. Bioinforma., vol. 18, no. 1, pp. 83–93, 2021, doi: 10.1109/TCBB.2020.2980831.

[20] M. Boulares, T. Alafif, and A. Barnawi, "Transfer Learning Benchmark for Cardiovascular Disease Recognition," IEEE Access, vol. 8, pp. 109475–109491, 2020, doi: 10.1109/ACCESS.2020.3002151.

[21] M. Talo, "Automated classification of histopathology images using transfer learning," Artif. Intell. Med., vol. 101, no. August, p. 101743, 2019, doi: 10.1016/j.artmed.2019.101743.

[22] S. Kumaresan, K. S. J. Aultrin, S. S. Kumar, and M. D. Anand, "Transfer Learning with CNN for Classification of Weld Defect," IEEE Access, vol. 9, pp. 95097–95108, 2021, doi: 10.1109/ACCESS.2021.3093487.

[23] 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 Comput. Appl., vol. 2, 2021, doi: 10.1007/s00521-021-05715-2.

[24] G. Vrbančič and V. Podgorelec, "Transfer learning with adaptive fine-tuning," IEEE Access, vol. 8, pp. 196197–196211, 2020, doi: 10.1109/ACCESS.2020.3034343.

[25] X. Feng, Y. Jiang, X. Yang, M. Du, and X. Li, "Computer vision algorithms and hardware implementations: A survey," Integration, vol. 69, no. June, pp. 309–320, 2019, doi: 10.1016/j.vlsi.2019.07.005.

[26] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," 2009 IEEE Conf. Comput. Vis. Pattern Recognit., pp. 248–255, 2010, doi: 10.1109/cvpr.2009.5206848.

[27] I. Kandel, M. Castelli, and A. Popovič, "Musculoskeletal images classification for detection of fractures using transfer learning," J. Imaging, vol. 6, no. 11, 2020, doi: 10.3390/jimaging6110127.

[28] 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.

[29] A. Krizhevsky, G. Hinton, and others, "Learning multiple layers of features from tiny images," 2009.

[30] S. Pouyanfar et al., "A Survey on Deep Learning," ACM Comput. Surv., vol. 51, no. 5, pp. 1–36, 2018, doi: 10.1145/3234150.

[31] 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. 119951–119960, 2020, doi: 10.1109/ACCESS.2020.3005450.