The Effect of Adaptive Learning Rate on the Accuracy of Neural Networks

Jennifer Jepkoech[1]*, David Muchangi Mugo[2], Benson K. Kenduiywo[3], Edna Chebet Too[4]
University of Embu, P.O BOX 6 – 60100, Embu, Kenya[1, 2]
Jomo Kenyatta University of Science and Technology, P.O. Box 62 000 – 00200, Nairobi, Kenya[3]
Chuka University, P.O BOX 109-60400, Chuka, Kenya[4]
echebet@chuka.ac.ke

***Abstract*—Learning rates in gradient descent algorithms have significant effects especially on the accuracy of a Capsule Neural Network (CNN). Choosing an appropriate learning rate is still an issue to date. Many developers still have a problem in selecting a learning rate for CNN leading to low accuracies in classification. This gap motivated this study to assess the effect of learning rate on the accuracy of a developed (CNN). There are no predefined learning rates in CNN and therefore it is hard for researchers to know what learning rate will give good results. This work, therefore, focused on assessing the effect of learning rate on the accuracy of a CNN by using different learning rates and observing the best performance. The contribution of this work is to give an appropriate learning rate for CNNs to improve accuracy during classification. This work has assessed the effect of different learning rates and came up with the most appropriate learning rate for CNN plant leaf disease classification. Part of the images used in this work was from the PlantVillage dataset while others were from the Nepal database. The images were pre-processed then subjected to the original CNN model for classification. When the learning rate was 0.0001, the best performance was 99.4% on testing and 100% on training. When the learning rate was 0.00001, the highest performance was 97% on testing and 99.9% on training. The lowest performance observed was 81% accuracy on testing and 99% on training when the learning rate was 0.001. This work observed that CNN was able to achieve the highest accuracy with a learning rate of 0.0001. The best Convolutional Neural Network accuracy observed was 98% on testing and 100% on training when the learning rate was 0.0001.**

***Keywords*—CNN; ConvNet; learning rate; gradient descent**

I. Introduction

Deep learning has been used over time for plant leaf disease detection and classification. Some of the researchers who have used deep learning include [29,30,31,32,33,34,35,36,37,38,39]. Capsule neural networks (CNN) are a regularly used neural network structure that has significant effects on deep learning, particularly in computer vision studies. CNN's have attained superhuman levels in different computer task categories, for example, object detection, classification, incidence segmentation, semantic segmentation, and parsing. The learning rate is viewed as the absolute hyper-parameter to tune and remarkably influence model training with gradient descent algorithms [1, 2]. Studies have come up with several learning rate techniques including inverse square root decay, linear decay, exponential decay, and cosine decay [3, 4]. These learning rates have varying procedures that are based on an optimization problem. One of the limitations involves the selection of a suitable learning rate for a given application.

Practically, researchers have adopted a trial-and-error method for various learning rates alongside diverse hyper-parameters, which is a very tedious process [5]. This paper utilizes a regulator that adapts three learning rate schedules of 0.001, 0.0001, and 0.00001. Existing learning rate schedules adopt predefined parametric learning rate changes, which are fixed regardless of prevailing training dynamics. The predefined parametric learning rate changes have a limited flexibility and may not be improved for the training dynamics of various high dimensional and non-convex advancement issues [6]. The context for this work provides adaptive meta-learned learning rates that dynamically adjust to current training. The process of training a neural network using an algorithm, for example, the error back-propagation [1, 2, 3, 4] is normally time-consuming, especially when working on complex problems. These types of algorithms naturally have a learning rate parameter that controls the extents by which the weights can change based on an observed error that was noted on the training set.

Learning rate schedules can dramatically affect the accuracy of the results. Therefore, the process of choosing learning rates using training algorithms can be problematic especially when there is no guiding value for specific tasks. Various algorithms have been used to tune the learning rate parameters [6, 7, and 8], yet such strategies generally have failed to concentrate on refining the resulting accuracy. Most of the experts in neural networks use the highest learning rates that allow merging. However, when learning rates are set too high, it causes unwanted divergent behavior in the loss function. Hence when the highest learning rates are applied to complex and large problems, there is a negative effect on the training process and accuracy. On the other hand, when the learning rate is set too low, the training progress will be very slow because very small updates are made to the weights of the work [9]. So there is a need to balance and there is no better way to do that other than to test several learning rates and observe their performances. This work adopts the use of online training instead of batch training. This is because batch training needs more time compared to online training with no corresponding improvement inaccuracy [5]. This paper aims to investigate the effect of learning rate on the accuracy of CNN's as applied in plant disease detection. Since Tensor flow recommends a learning rate of 0.001, this works started by using that learning rate and observed a low percentage of 84%

*Corresponding Author

accuracy in testing. It is from there that this work focused on reducing the learning rate further to 0.0001 and then to 0.00001. A total of 24 experiments were conducted for plant leaf disease classification using the three learning rates and 0.0001 gave the best classification results of 99.4% accuracy on testing and 100% on training.

II. Related Work

Hyperparameters such as batch size need adjusting before capsule neural network training for image classification. Studies on the effect of batch size and learning rates on neural network accuracy have been conducted. The studies have tried to determine the more efficient network performances related to learning rates and the magnitude of batches.

According to [10] the default number batch size should be 32. The author noted that a large batch size and high learning rates speed up the process of network performance but reduce the number of updates needed to reach convergence. Batch sizes do not affect the performance of the neural network but influence the convergence time. Masters and Luschi [11] studied the effect of batch sizes on ImageNet, CIFAR10, and CIFAR100 datasets for two architectures of ResNet and AlexNet. The batch sizes ranged between 21 and 211. The results showed that the best accuracies were achieved from batch sizes that ranged from 2 and 32. The study concluded that large batch sizes are not efficient compared to small batch sizes. Radiuk [12] also studied the effect of batch size on network performance for the classification of images using CIFAR-10 and MNIST datasets for the LeNet architecture. The study used two learning rates of 0.0001 (CIFAR-10) and 0.001 (MNIST). The results showed that the highest accuracy was obtained from the largest batch size with a lower learning rate of 0.0001. This showed that batch size and learning rates affect the performance of neural networks.

Several studies have proposed improved update schedules for gradient descent algorithms [7, 8, 9, 13, 14, and 15]. In [7], the need for direct learning of the gradient descent updates through the use of the long short-term memory (LSTM) network was proposed. Hyper gradient tends to assume the learning rate derivative and subsequently updates it according to its gradient [8]. Z. XU [9] proposed a reinforcement learning-based framework that can auto-learn an adaptive learning rate schedule according to the existing information from historical training. This method puts into consideration the whole training history while presenting a comprehensive interpretation. Daniel [13] proposed the application of reinforcement learning (RL) with a focus on learning rate adaptation. This paper uses the learning rates as the action and the reward indicator include validation loss. Duchi et al. [14] used learning rate adaptation based on the weight and the total number of gradient squares and obtained some results. Kingma [15] used an exponentially decayed mean of historical gradients.

Neural Networks are models with progressive layers of neurons that have been in existence for quite a long time. They can be trained in both Supervised and Unsupervised [16] ways. In supervised training, a backpropagation algorithm was created in the 1970s [17]. This algorithm utilizes a gradient descent approach to compute the learning system of the neural network. A gradient descent approach is commonly used in neural networks to update parameters ($\lambda = \{1e^{-1},\ 1e^{-2},\ 1e^{-3}\}$). Such training is conducted to get to an optimum point where the loss is at its minimum and the expected and predicted values are almost similar [18]. Training a large neural network is a challenging task. Sebastian [19] established the Stochastic Gradient Descent (SGD) algorithm to accomplish an improved performance during the training time using variable learning rates. Such processes have been described as Adaptive Learning Rates/ Rate Scheduling [26]. Larger learning rates have also been used by [27], who used a learning rate of 0.4 and achieved 75% accuracy. The results show a low accuracy rate which most likely was caused by the high learning rate. Purnomo [28] used 0.01 and observed that this learning rate led to low accuracies

III. Convolutional Neural Networks (ConvNets)

When convolutional neural networks in Fig. 1 are applied to disease detection, models demonstrate great performance. The discussion below shows some great materials showing the use of convolutional neural networks in the detection and classification of plant diseases.

The authors in [59] used LeNet architecture [Le89] architecture with CNN for the classification of banana leaf disease. The images used were from Plantvillage which were from homogeneous backgrounds. The results obtained, according to the authors were good. The challenges experienced were that in some splits, the model took more time to converge, and practically, all the images cannot be from uniform backgrounds.

Researchers in [60] used the digital color image analysis discrimination method: The results were questionable because of the existence of other leaves or weeds. Segmentation is not fit to be used in the field because it will fail to effectively extract the leaf from its background hence inaccurate results. Alex Net and transfer learning were used by [61] to detect common rice plant anomalies using CNN. During the classification task, the approach never considered the specific class of diseases that may affect rice plants. The authors also used transfer learning on AlexNet which is a small and old CNN architecture. Authors in [62] used segmentation method in detecting soybean rust from multispectral images using CNN. The results were questionable because of the existence of other leaves or weeds. Segmentation is not fit to be used in the field because it will fail to effectively extract the leaf from its background hence inaccurate results. Author in [63] used segmentation with CNN and there was a lot of reliability on hand-crafted features such as color histograms, texture features, shape features, and SIFT that require expensive work and demand expert knowledge. The author in [64] used Gabor filter for feature extraction and Artificial Neural Network classifier for classification in real plant tomato leaf disease recognition. They used images from homogeneous backgrounds alone which practically is not true because there must be other plants and weeds in farms. The author in [65] used images from plant village alone and AlexNet architecture for plant disease detection and they lacked the accuracy of result because plant village dataset images have homogeneous

backgrounds while under normal conditions, images from the field have heterogeneous backgrounds.

The authors in [65] utilized the convolutional neural network to detect disease in plant leaves. In the research, 54306 images and 14 different species of plants were used which later represented 26 diseases together with healthy leaves. Furthermore, the authors used segmented, greyscale, and colored images for model training and the accuracy was 99.35%. However, when tested on another dataset, the accuracy fell to 31.4%. Real-Time captured images have heterogonous backgrounds while the images from the plant village dataset have a homogenous background. The author in [66] used convolutional neural networks to recognize 13 plants of different species and detect disease in their leaves. All the images utilized by the author were secondary images fetched from the internet. Here, 15 classes were considered and one class for the healthy leaves the accuracy was measured at 88%. However, it was discovered that most of the images from the website were mislabeled and differed greatly from those taken from the field which later introduced a mismatch error. The researcher in [67] used 87,848 images with 25 different plant species and included healthy plants. AlexNet, Over feat, GoogleNet and AlexNet were utilized in the identification of plant leaf diseases from images captured from the field. The aim was to be able to match the plant and disease combination when a leaf image was provided. The datasets here contained images from the laboratory and field as well. The accuracy was found to be 99.53%. CNN gives some results, but they have two major challenges, Pooling layers, and Translation Invariance.

In research done by [68], CNN was used in detection and classification. The technique used was transfer learning while the coffee disease was classified as coffee leaf rust. The study used an android profiler in determining resource consumption. The researchers obtained some results. However, they observed loss of data through Pooling.

IV. Capsule Neural Networks (CNNs)

Capsules comprise neuron clusters that have vector activities [21]. The activities are a representation of different pose parameters while their vector lengths show the existence of specific neuron elements. Most CNNs problems are generally associated with the pooling layers. For capsule networks, issues of pooling layers are corrected using the "routing by agreement" procedure [22]. The procedure involves adding neuron outputs to parent capsules in a subsequent layer, though, they typically have different coupling coefficients. The output of the parent capsules is based on the prediction of an individual capsule. If a prediction is consistent with the actual output of the parent capsule, then there is an increase in the coupling coefficient between the two capsule layers. Using capsule *I* that has u_i as its output, the prediction of an individual capsule *i* for parent capsule *j* is as shown in equation 1.

$$\hat{U}_{j|i} = W_{ij}\, u_i \tag{1}$$

where, $\hat{U}_{j|i}$ is considered as the prediction vector of the *jth* capsule output from a different layer resulting from capsule *i*, and where W_{ij} is used as a weighting matrix after being learned using the backward pass. The SoftMax equation can after that be computed from the coupling coefficients c_{ij} as shown in equation 2.

$$c_{ij} = \frac{exp(b_{ij})}{\sum_k exp(b_{ik})} \tag{2}$$

Where the log probability is represented by b_{ij}, which is programmed initially as 0 before initiating the "routing by agreement" process. The computation for adding a vector neuron to the parent capsule *j* is shown in equation 3.

$$s_j = \sum_i c_{ij}\, \hat{U}_{j|i} \tag{3}$$

Subsequently, there is a need to use a non-linear equation (4) to prevent capsule vectors from producing more than one output and creating the final output of an individual capsule.

$$v_j = \frac{\|s_j\|^2}{1+\|s_j\|^2}\frac{s_j}{\|s_j\|} \tag{4}$$

where, v_j is the output of capsule *j*, and s_j is the input vector. The "routing by agreement" process allows for an updating of the log probabilities considering the agreement set between v_j and $\hat{U}_{j|i}$. When the two capsule vectors are in agreement, then they will produce a larger inner output. Equation (5) shows an agreement a_{ij} that is necessary for the updating of coupling coefficients and log probabilities.

$$a_{ij} = v_j \,.\, \hat{U}_{j|i} \tag{5}$$

For equation 6, individual capsule k in the final layer is related to l_k, which is a loss function. The function adds high loss values on individual capsules that have long output instantiation parameters when there is a missing entity.

$$l_k = T_k\, max\,(0, m^+ - \|v_k\|)^2 + \lambda(1 - T_k)\, max(0, \|v_k\| - m^-)^2 \tag{6}$$

When class k exists, T_k is 1. Otherwise when class k is absent T_k is 0. Hyperparameters m^-, λ and m^+ are used before beginning the learning process, and they must be indicated.

V. Hyperparameters in Deep Learning

A. Learning Rate

While training neural networks, a hyperparameter that has a positive value ranging between 0.0 and 1.0 is utilized [5]. The parameter is known as the learning rate and can be configured. Learning rates help in taking control of the adaptation of the model to a given problem. The smaller the learning rate the more the epochs because the changes made to the weights will be small. Large learning rates attract few epochs due to high speed. When the learning rate is too high, model convergence is very fast while on the other hand when the learning rate is too small, the process might be stuck at some point. It is therefore important to carefully select the learning rate to get correct results. This, therefore, makes the learning rate the most important parameter when it comes to neural networks.

B. The Number of Layers that are Hidden

The topology or architecture of a network is controlled by the actual number of layers and the nodes that each hidden layer contains. During network configuration, the values of the parameters must be specified. Systematic experimentation is considered the most accurate way of configuring parameters for various modeling problems. It is only through running various experiments that the number of hidden layers required, can be determined.

C. Momentum

This is a technique utilized during the backpropagation stage to track preceding directions and store them as embedded processed data. This helps the model to learn and embed the direction of the previous weights and proceed towards the same direction in the next propagation.

D. Activation Function

The decision as to whether neuron activation should be done or not is done by the activation function. The action is completed through the calculation of weighted sum and bias addition. Non-linearity is introduced to the neuron output by activation functions. For good results, it is advisable to use ReLu activation for layers that are hidden and then use a sigmoid activation function in the final layer.

E. Mini-batch Size

While using a very large dataset, it is challenging to feed a neural network with all of it. Therefore, it is a good practice to subdivide data into smaller sizes or group them into batches. This helps because each time the algorithm trains itself, a batch of the same size will be trained. If the batch sizes are too big, however, it may result in a model that is overgeneralized and data won't fit well.

F. Epochs

Epochs represent the number of times the dataset will be trained by the used algorithm during training. The number varies with data or task one is facing and there is no predefined number of epochs in any neural network. The idea is to introduce a condition that stops the epochs when the error is near zero or just starts with a lower number of epochs.

G. Dropout

Dropout allows the removal of some nodes in cases where the neural network is very heavy and cannot train well. The action is performed during the training stage and helps remove redundancies that may occur due to congestion.

VI. Proposed Work Methodology

In deep learning there are generally two basic parameters; hyper-parameters and machine learnable parameters (MLP). While training a particular dataset in any model, algorithms used in that model can estimate MLP on their own. On the other hand, Hyperparameters are assigned by data scientists or engineers in form of values. These values help in tuning the model and control how algorithms can learn. Learning rate is denoted by 'α'. In this work, the learning rate used is known as adaptive learning rate whereby the increase or decrease in learning rate is based on the gradient value of cost function (*CF*). Equation 7 below was used in the calculation of learning rates.

$$\propto_n = \frac{\propto_0}{\sqrt{s_n}} \tag{7}$$

where, the initial learning rate is denoted by *0* while the momentum factor (MF) is denoted by s_n. The number of epochs is denoted by *n.* MF was calculated using Equation 8 below:

$$s_n = [\gamma s_{n-1} + (1- \gamma) \frac{\partial CF}{\partial \beta}]n \tag{8}$$

where, γ is the hyperparameter and s_n is exponentially weighted gradient average. Here values of all the gradients were considered including those from previous epochs. The major contribution of this work is to demonstrate the effect of learning rates in the classification accuracy of a CNN. This has been achieved through the performance of the experiments using three learning rates with various class sizes. This work also assesses the accuracy of the original CNN model when different plant species are used with different learning rates. The learning rates that have been used in the experiments are 0.001, 0.0001, and 0.00001, respectively.

A. The Data

This work has used two sets of data. One set comprised of images from the PlantVillage dataset while the other dataset comprised of images from the Nepal database. The algorithm used for this work is routing by agreement with a convolution of 256 filters, a kernel size of 9, and a ReLU activation function [23]. The model has 32 channels and a kernel size of 9 with a vector dimension of 16. It contains a decoder network with 3 dense layers (512; 1024; shape). Images from the PlantVillage database were resized for use in this research. The routing by agreement algorithm has been used by Hinton et al. [22] for lung cancer screening. Mobiny and Van Nguyen [24] used the algorithm to detect movements in movies. In this work, each capsule attempted to predict the output of the parent capsules, and when the prediction conformed to the actual output of the parent capsule, then there it was assumed that there was an increase in the coupling coefficient between the two capsules as outlined by Gogola et al. [25]. This work has also tested the same dataset on a convolutional neural networks (ConvNet) model and the results were recorded.

The general procedure for disease classification involved several stages, such as image acquisition, data pre-processing, and data classification. Thereafter, training and validation of the dataset were performed using different learning rates in the normal CNN model [22] for classifications. The overall workflow diagram for the techniques adopted is presented in Fig. 1.

B. Image Acquisition

This phase involved the acquisition of images from the PV dataset while other images were from the Nepal database. The images were initially collected under different conditions that were either controlled, wild, or uncontrolled before being put under laboratory conditions.

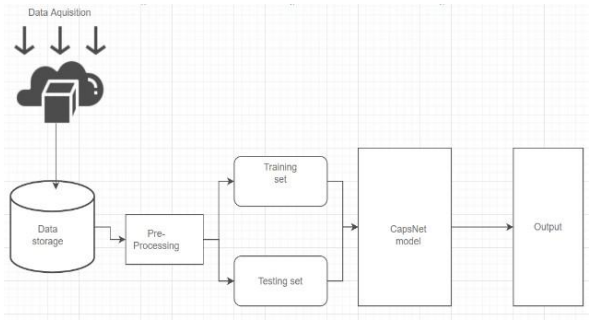

Fig. 1. The General Flow Diagram.

C. Pre-processing

Pre-processing techniques that were used include resizing and checking dimensionality. For this work, each image from the dataset was checked to find out if they were of the same squared shape. The images that were not of the squared shape were cropped to get the center square part of the image for good classification. All images were then resized to 28 x28 pixels. Image resizing was done using Photoshop where the large images were reduced in size and unneeded pixel information was discarded. In cases where the images were too small, Photoshop was used to enlarge, create a pixel and add new pixel information. Resizing was done to reduce the number of parameters and increase the processing speed.

D. Training and Testing Data Sets

In regards to splitting, the entire dataset was divided into two subsets one used to train the model and the other used to test the model. The testing subset was used to make predictions that were compared with the original one to check the model accuracy. The major objective of splitting was to be able to evaluate the model based on new data; data that had not been used to train the model. In all the experiments, the test set was 0.3 of the total number of images while the training set was 0.7 of the total images used.

E. Experimental Results

The first six experiments were done using four disease classes as shown in Table I. The total number of images that were used for this experiment was 10295 and 30% of that was used for testing while 70% was used for training. In the first and second experiments, the learning rate that was used was 0.0001 and the accuracy observed on training was 99.9% while that of testing was 99% in the Capsule neural network (CNN) model. While the ConvNet model displayed 100% on training and 98.75% on testing. The levels of accuracies and losses have been represented by Fig. 2 and Fig. 3 for CNN, 4 and 5 for ConvNet.

TABLE I. THE CLASSES THAT WERE USED FOR THE FIRST, SECOND AND THIRD EXPERIMENTS

Disease	Plant	No.of images
Esca Black Measles	Grape	2573
healthy	Grape	2370
Leaf blight Isariopsis Leaf Spot	Grape	2450
Huanglongbing Citrus greening	Orange	2902

Fig. 2. Training and Testing Accuracy for CNN using Learning Rate of 0.0001 with 4 Disease Classes.

Fig. 3. Training and Testing Loss for CNN using Learning Rate of 0.0001 with 4 Disease Classes.

Fig. 4. Training and Testing Accuracy for ConvNet using Learning rate of 0.0001 with 4 Disease Classes.

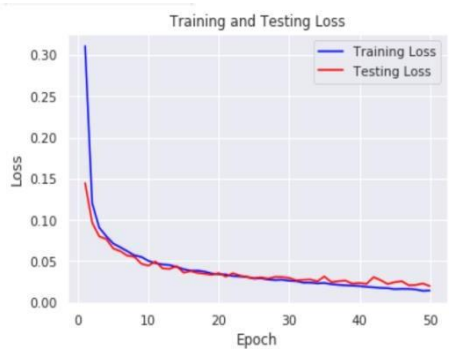

Fig. 5. Training and Testing Loss for ConvNet using Learning Rate of 0.0001 with 4 Disease Classes.

In the third and fourth experiments, the highest accuracy reached on training was 94% while the testing accuracy was 97.0% for the capsule neural network (CNN) while the Convolutional neural network (ConvNet) had 97% accuracy on testing and 99% accuracy on training. The learning rate for this particular experiment was 0.00001. Fig. 6 and 7 show the graphs for training and testing accuracy and loss for CNN while Fig. 8 and 9 show training and testing accuracy and loss for ConvNet.

Fig. 6. Training and Testing Accuracy for CNN using Learning Rate of 0.00001 with 4 Disease Classes.

Fig. 7. Training and Testing Loss for CNN using Learning Rate of 0.00001 with 4 Disease Classes.

Fig. 8. Training and Testing Accuracy for ConvNet using Learning rate of 0.00001 with 4 Disease Classes.

Fig. 9. Training and Testing Loss for ConvNet using Learning rate of 0.00001 with 4 Disease Classes.

The fifth and sixth experiment was performed using 10295, and 30% of that was used for testing. The highest accuracy reached on training was 99.9% while the testing accuracy was 81% for Capsule neural network (CNN). The testing accuracy for convolutional neural network (ConvNet) was 96.5 while that of training was 99.0%. The learning rate for this particular experiment was 0.001. Fig. 10 shows the graph for training and testing accuracy for CNN, while Fig. 11 shows training and testing loss for CNN. Fig. 12 and 13 show training and testing accuracy and loss for ConvNet, respectively.

Fig. 10. Training and Testing Accuracy for CNN using Learning Rate of 0.001 with 4 Disease Classes.

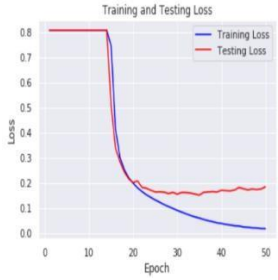

Fig. 11. Training and Testing Loss for CNN using Learning Rate of 0.001 with 4 Disease Classes.

Fig. 12. Training and Testing Accuracy for ConvNet using Learning Rate of 0.001 with 4 Disease Classes.

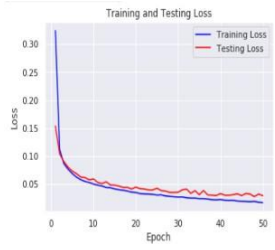

Fig. 13. Training and Testing Loss for ConvNet using Learning Rate of 0.001 with 4 Disease Classes.

The next experiments were done using six classes as shown in Table II. The total number of images that were used for this experiment was 12265 and 30% of that was used for testing. The learning rates that were used for the experiments were 0.0001, 0.00001, and 0.001, respectively. In the seventh and eighth experiments, the accuracy observed on training was 99.9% while that of testing was 99% for Capsule neural network (CNN) while the training and testing accuracy for the convolutional neural network (ConvNet) was 100% and 96.2%, respectively. The levels of accuracy and loss for CNN have been represented in Fig. 14 and Fig. 15, respectively while those of ConvNet have been represented in Fig. 16 and 17.

TABLE II. THE CLASSES THAT WERE USED FOR THE FOURTH, FIFTH AND SIXTH EXPERIMENTS

Disease	**Plant**	**No.of images**
Esca Black Measles	Grape	2573
healthy	Grape	2370
Leaf blight Isariopsis Leaf Spot	Grape	2450
Huanglongbing Citrus greening Early_blight Healthy	Orange Tomato Tomato	2902 1000 970

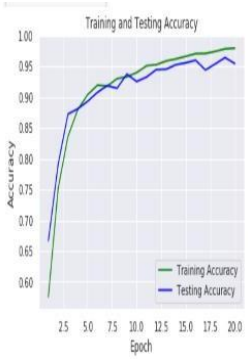

Fig. 14. Training and Testing Accuracy for CNN using Learning Rate of 0.0001 with 6 Disease Classes.

Fig. 15. Training and Testing Loss for CNN using Learning Rate of 0.0001 with 6 Disease Classes.

Fig. 16. Training and Testing Accuracy for ConvNet using Learning rate of 0.0001 with 6 Disease Classes.

Fig. 17. Training and Testing Loss for ConvNet using Learning Rate of 0.0001 with 6 Disease Classes.

In the ninth and tenth experiments, the learning rate used was 0.001 and the highest accuracy reached on training was 99.1% while the testing accuracy was 97.5% while using Convolutional Neural network (ConvNet). Training and testing accuracies were 100% and 98% for Capsule Neural Network (CNN) respectively. Fig. 18 shows the graph for training and testing accuracy, while Fig. 19 shows training and testing loss for ConvNet. Fig. 20 and 21 show training and testing accuracies for CNN.

Fig. 18. Training and Testing Accuracy for ConvNet using Learning Rate of 0.001 with 6 Disease Classes.

Fig. 19. Training and Testing Loss for ConvNet using Learning Rate of 0.001 with 6 Disease Classes.

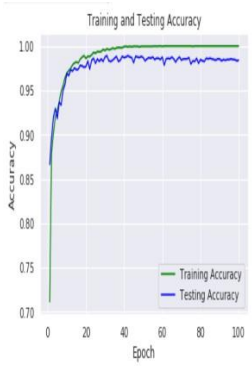

Fig. 20. Training and Testing Accuracy for CNN using Learning Rate of 0.001 with 6 Disease Classes.

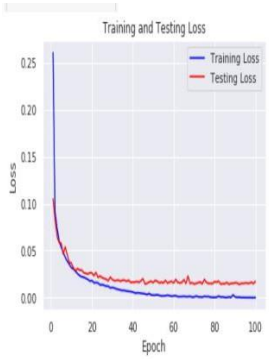

Fig. 21. Training and Testing Loss for CNN using Learning Rate of 0.001 with 6 Disease Classes.

In the eleventh and twelfth experiment, the learning rate was 0.0001 and the highest accuracy reached on training was 100% while the testing accuracy was 98.84% for Capsule Neural Network (CNN) while training and testing accuracies for Convolutional Neural Network (ConvNet) were 99.9% and 80%, respectively. Fig. 22 shows the graph for training and testing accuracy while Fig. 23 shows training and testing loss for ConvNet while Fig. 24 and 25 shows training and testing accuracies and losses for CNN.

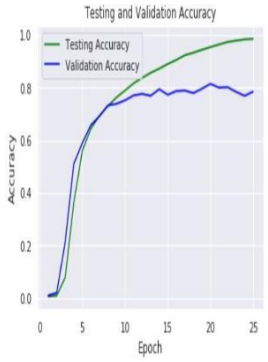

Fig. 22. Training and Testing Accuracy for ConvNet using Learning Rate of 0.0001 with 8 Disease Classes.

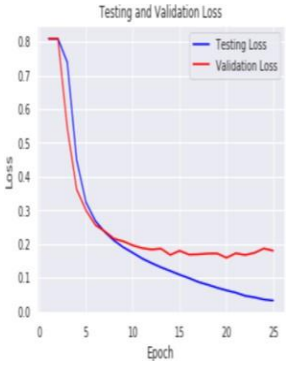

Fig. 23. Training and Testing Loss for ConvNet using Learning Rate of 0.0001 with 8 Disease Classes.

Fig. 24. Training and testing accuracy for CNN using learning rate of 0.0001 with 8 disease classes

Fig. 25. Training and Testing Loss for CNN using Learning Rate of 0.0001 with 8 Disease Classes.

The next experiments were done using 10 classes as shown in Table III below. The total number of images that were used for this experiment was 12751 and 30% of that was used for testing. The learning rate that was used for the thirteenth and fourteenth experiment was 0.0001 and the accuracy observed on training was 99.9% while that of testing was 93% for Capsule Neural Network (CNN). The level of accuracy and loss for training and testing using Convolutional Neural network (ConvNet) was 100% and 97.5%, respectively. The levels of accuracies and losses for CNN have been represented by Fig. 26 and Fig. 27 below while those of ConvNet have been represented by Fig. 28 and 29, respectively.

TABLE III. THE CLASSES THAT WERE USED FOR THE SEVENTH, EIGHTH AND NINTH EXPERIMENTS

Disease	Plant	No.of images
Healthy	Coffee	145
Miner	Coffee	400
Rust	Coffee	943
phoma	Coffee	1000
Cercospora	Coffee	870
Common_rust	Corn	2000
Healthy	Corn	1270
Cercospora_leaf_spot	Corn	1870
Gray_leaf_spot	Blueberry	2276
Healthy	Cherry	1977

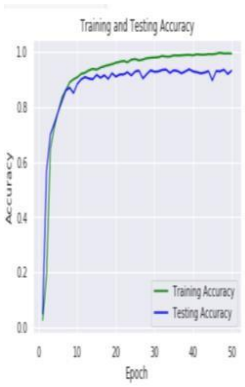

Fig. 26. Training and Testing Accuracy for CNN using Learning Rate of 0.0001 with 10 Disease Classes.

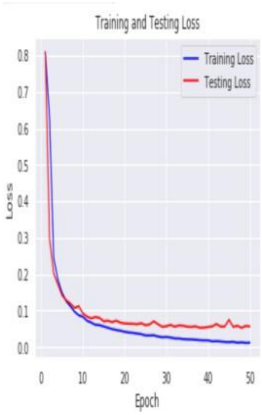

Fig. 27. Training and Testing Loss for CNN using Learning Rate of 0.0001 with 10 Disease Classes.

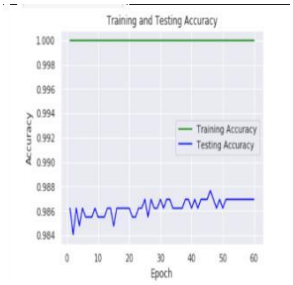

Fig. 28. Training and Testing Accuracy for ConvNet using Learning Rate of 0.0001 with 10 Disease Classes.

Fig. 29. Training and Testing loss for ConvNet using Learning Rate of 0.0001 with 10 Disease Classes.

In the fifteenth and sixteenth experiments, the highest accuracy reached on training was 91% while the testing accuracy was 91% for Capsule Neural Network (CNN), while Convolutional Neural Network (ConvNet) showed 97.75% for testing and 100% for training. The learning rate (LR) for this particular experiment was 0.00001. Fig. 30 and 31 show the graphs for training and testing accuracy and loss using CNN. Fig. 32 and 33 shows training and testing accuracy and loss observed from ConvNet.

In the seventeenth and eighteenth experiments, the highest accuracy reached on training was 99.9% while the testing accuracy was 81% while using Capsule Neural Network (CNN). On using Convolutional Neural Network (ConvNet) the testing accuracy was 97.75% while that of training was 100%. The learning rate for this particular experiment was 0.001. Fig. 34 shows the graph for training and testing accuracy while Fig. 35 shows training and testing loss for CNN while Fig. 36 and 37 show training and testing accuracies and losses for ConvNet.

Fig. 30. Training and Testing Accuracy for CNN using Learning Rate of 0.00001 with 10 Disease Classes.

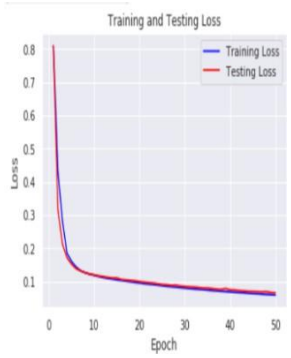

Fig. 31. Training and Testing Loss for CNN using Learning Rate of 0.00001 with 10 Disease Classes.

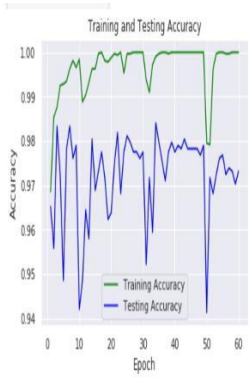

Fig. 32. Training and Testing Accuracy for ConvNet using Learning Rate of 0.00001 with 10 Disease Classes.

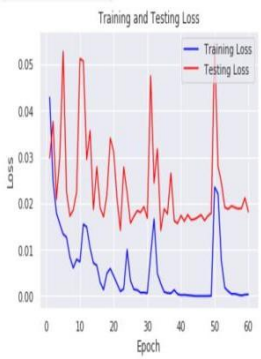

Fig. 33. Training and Testing Loss for ConvNet using Learning Rate of 0.00001 with 10 Disease Classes.

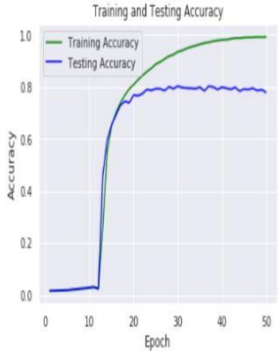

Fig. 34. Training and Testing Accuracy for CNN using Learning Rate of 0.001 with 10 Disease Classes.

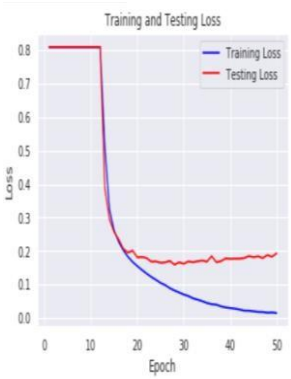

Fig. 35. Training and Testing Loss for CNN using Learning Rate of 0.001 with 10 Disease Classes.

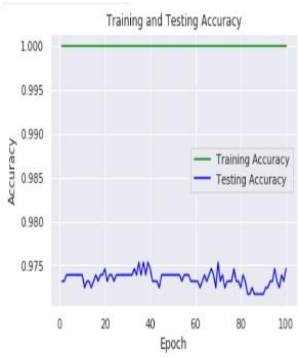

Fig. 36. Training and Testing Accuracy for ConvNet using Learning Rate of 0.001 with 10 Disease Classes.

Fig. 37. Training and Testing Loss for ConvNet using Learning Rate of 0.001 with 10 Disease Classes.

A total of 21121 images and 30% of that were used for testing for the next experiments. Table IV shows the classes that were used for the experiments. In the nineteenth and twentieth experiments, the highest accuracy reached on training was 99.9% while the testing accuracy was 97.4% while using Capsule Neural Network (CNN). On using Convolutional Neural Network (ConvNet) testing accuracy observed was 98.04% while that of testing was 100%. The learning rate (LR) for this particular experiment was 0.0001. Fig. 38 shows the graph for training and testing accuracy while Fig. 39 shows training and testing loss for CNN. Fig. 40 and 41 show training and testing accuracies and losses for ConNet.

TABLE IV. THE CLASSES THAT WERE USED FOR THE TENTH, ELEVENTH AND TWELFTH EXPERIMENTS

Disease	Plant	Number of images
Healthy	Coffee	145
Miner	Coffee	400
Rust	Coffee	943
phoma	Coffee	1000
Cercospora	Coffee	870
Common_rust	Corn	2000
Healthy	Corn	1270
Cercospora_leaf_spot Gray_leaf_spot	Corn	1870
Healthy	Blueberry	2276
Healthy	Cherry	1977
Black_rot	Grape	1000
Esca_(Black_Measles)	Grape	2573
Healthy	Grape	2370
Leaf_blight_(Isariopsis_Leaf_Spot)	Grape	977
Haunglongbing_(Citrus_greening)	Orange	800
Healthy	Soybean	650

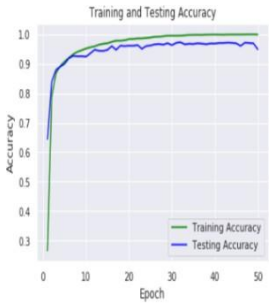

Fig. 38. Training and Testing Accuracy for CNN using Learning Rate of 0.0001 with 16 Disease Classes.

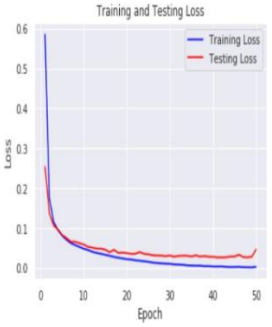

Fig. 39. Training and Testing Loss for CNN using Learning Rate of 0.0001 with 16 Disease Classes.

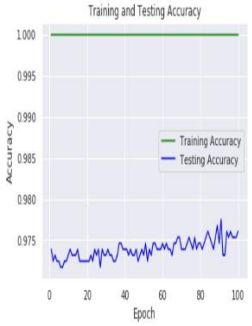

Fig. 40. Training and Testing Accuracy for ConvNet using Learning Rate of 0.0001 with 16 Disease Classes.

Fig. 41. Training and Testing Loss for ConvNet using Learning Rate of 0.0001 with 16 Disease Classes.

In 21st and 22nd experiments, the highest accuracy reached on training was 100% while the testing accuracy was 97.3% for Capsule Neural Network (CNN). The training and testing accuracies for Convolutional Neural Network (ConvNet) were 100% and 98.04%, respectively. The learning rate for this particular experiment was 0.00001. Fig. 42 shows the graph for training and testing accuracy while Fig. 43 shows training and testing loss for CNN while Fig. 44 and 45 show training and testing accuracies and losses for ConvNet.

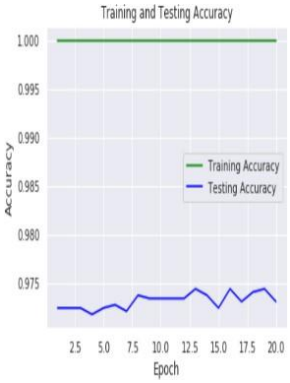

Fig. 42. Training and Testing Accuracy for CNN using Learning Rate of 0.00001 with 16 Disease Classes.

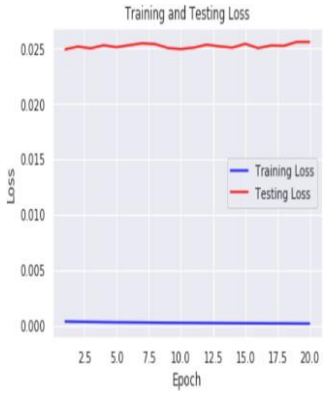

Fig. 43. Training and Testing loss using Learning Rate of 0.00001 with 16 Disease Classes.

Fig. 44. Training and Testing Accuracy for ConvNet using Learning Rate of 0.00001 with 16 Disease Classes.

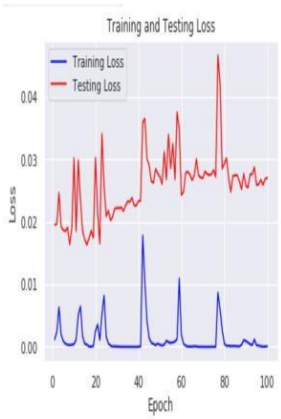

Fig. 45. Training and Testing Loss for ConvNet using Learning Rate of 0.00001 with 16 Disease Classes.

The total number of images that were used for the 23rd and 24th experiment was 15220 and 30% of that was used for testing. The highest accuracy reached on training was 99.9% while the testing accuracy was 84.5% for Capsule Neural Networks (CNN). The testing accuracy reached for Convolutional Neural Network (ConvNet) was 98.40% while that of training was 100%. The learning rate for this particular experiment was 0.001. Fig. 46 shows the graph for training and testing accuracy while Fig. 47 shows training and testing loss. Fig. 48 and 49 show training and testing accuracies and losses for ConvNet.

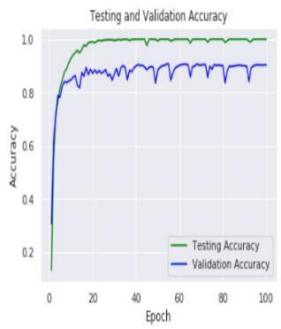

Fig. 46. Training and Testing Accuracy for CNN using Learning Rate of 0.001 with 16 Disease Classes.

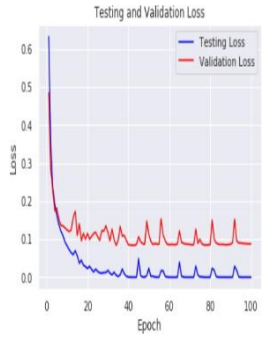

Fig. 47. Training and Testing Loss for CNN using Learning Rate of 0.001 with 16 Disease Classes.

Fig. 48. Training and Testing Accuracy for ConvNet using Learning Rate of 0.001 with 16 Disease Classes.

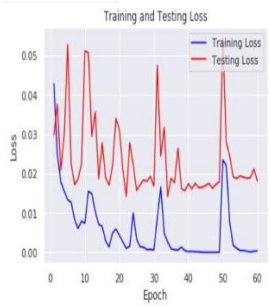

Fig. 49. Training and Testing Loss for ConvNet using Learning Rate of 0.001 with 16 Disease Classes.

From Table V, the highest accuracy of 0.99 was observed on recognition when the learning rate of 0.0001. On the other hand, the lowest accuracy of 0.81 was observed on recognition when the learning rate was 0.001. It was also noted that when images from other databases were added to those from PlantVillage, accuracy dropped from 0.99 to 0.97; hence it was concluded that CNN works best with PlantVillage datasets when it comes to disease detection. The best accuracy in testing while using ConvNet was 98.8%. It was however noted that there was a loss of data due to pooling which may have led to lower accuracies in both testing and training. There was also overfitting which shows that ConvNets are affected by dataset size, unlike the CNN which was not affected by dataset size at all. It was also noted that when images from other databases were added to those from PlantVillage, accuracy dropped from 0.99 to 0.97; hence it was concluded that CNN works best with PlantVillage datasets when it comes to disease detection.

TABLE V. RESULTS ANALYSIS FOR BOTH TRAINING AND TESTING IN CNN AND CONVNET

Dataset Size	Learning Rate	Convolutional Neural Network(ConvNet)		Capsule Neural Network(CNN)	
		Training accuracy (%)	Testing accuracy (%)	Training accuracy (%)	Testing accuracy (%)
10295	0.001	99	96.5	99	81
	0.0001	100	98	99.9	99
	0.00001	99	96.5	94	97.0
12265	0.001	99.9	80	100	98.84
	0.0001	100	96.2	99.9	99
	0.00001	99.1	97.5	100	98
12751	0.001	100	97.75	99	81
	0.0001	100	97.5	99	93
	0.00001	100	97.8	91	91
21121	0.001	100	98.4	99.9	84.5
	0.0001	100	98.0	99	97.4
	0.00001	100	98.04	99.9	97.3

The values chosen for learning rates can speed up the training processes of neural networks [40]. The author in [42] studied the effect of fine-tuning the learning rate together with the batch size and proposed the adjustment of learning rates relative to the batch size. This work used large and small batch size datasets of 21121and 10295 plant images. The author in [44] characterized the functions of learning rates on training and testing accuracies of neural network models. With this is the critical hyperparameter of the gradient descent learning rate [45]. Neural networks, according to several studies, do not learn when very large learning rates are used [46, 47, 48, and 49]. These studies further state that the use of very small learning rates leads to slow optimization and poor accuracy results. This study found that the model was able to learn with learning rates of 0.00001, 0.0001, and 0.001. However, the learning time for the 0.0001 learning was slower compared to the 0.001 learning rate. In support of this, [50] notes that very high learning rates need constant training that may end up consuming more time than is necessary and fail to achieve the expected accuracy. On the other hand, very low learning rates result in gradient decline, as well as lead to an increase in the number of reiterations [52].

From Table V, it was also observed that the training rate was relatively low when the learning rate was at 0.00001 while the training rate was high when the learning rate was either 0.001 or 0.0001. To strike a balance, the learning rate of 0.0001 for both training and testing was able to give a perfect fit.

VII. DISCUSSION

This section was used to examine the importance and performance of learning rates in neural networks' training for optimum test accuracies. Studies such as [40, 41, 42, 43], have recognized the influence of learning rates in achieving high accuracies. Neural networks' training processes are usually affected by learning rates. When the training is done many times, the learning rate can be affected and the system may fail to generate high accuracies as expected. The author in [40] established the triangle cyclic learning rates that included TRI2, TRI, and TRIEXP. However, this method does not require the use of specific learning rates but periodically varies the learning rate at certain intervals. Our study on the other hand used specific learning rates of 0.00001, 0.0001 and 0.001.

The findings by [11] match our results regarding the relationship between learning rates and dataset sizes. The author in [11] proposed the use of smaller dataset sizes. However, our results do not agree with [12], who note that the use of larger learning rates, results in an increased accuracy result for the neural network. This is because from our results smaller learning rate of 0.0001 had the highest accuracies ranging between 93.7% and 99%, unlike the larger learning rate of 0.001 that had learning rates ranging from 80% and 84.5%. Similarly, [51] performed simulations where learning rates of 0.001 and 0.01 resulted in the best accuracy percentages. Smaller learning rates produced higher accuracy values and more time taken for the test training. Conversely, larger learning rates result in reduced accuracy percentages and a fast training process. Concerning larger dataset size, our findings agree with [12] that the higher the size, the better the neural performance. This work proposes the use of large dataset sizes. This is supported by [12] on the relationship between learning rates and batch sizes. This work highlights that larger dataset sizes require higher learning rates. The reason being that a larger dataset size of 21121 had the highest accuracy of 84.5% for the larger learning rate of 0.001. The author in [10] recommends a default batch size value of 32. The author in [53] was able to achieve the highest accuracy values using a batch size of 16. Other studies used ten classes

of image datasets to achieve the best performance [54, 55, and 56]. This work used the batch size of 10 and CNN algorithm and the best performance (99% and 97.4%) was achieved for a dataset of 12,265 and 9899 images (11 classes and 4 classes, respectively) using the smaller learning rate of 0.0001. Larger learning rates have also been used by [58], who used a learning rate of 0.4 and achieved a 75% accuracy. The author in [58] used two learning rates of 0001 and 0.01 to achieve the best performances and observed that the lower learning rate gave a higher accuracy as compared to its higher counterpart. These learning rate values have been supported by [59] whose study also produced the highest accuracy from the use of correct input parameters. The study concludes that CNN works best with a learning rate of 0.0001 when all other things are kept constant. Another observation according to this work was that other databases do not work well with CNN just like the PalntVillage datasets [57]. The introduction of coffee from a different database lowered the accuracy levels from 99% to 97%.

VIII. Conclusion

In conclusion, this work was able to observe that a large learning rate tended to move in the "correct" direction, which led to overshooting or surface error that could have interfered with the accuracy hence made the training process consume more time. This could be because of the constant "unlearning" and overshooting problems that require backtracking. Under normal circumstances, the failure to backtrack can result in failed fluctuations and poor accuracy percentages [20]. In this work, it was observed that small learning rates prevented instabilities and overcorrections, and allowed for a smooth path over the error landscape to reach a minimum. A lower learning rate further resulted in a smoother path and therefore, to significantly improve the testing accuracy, one can reduce the learning rate.

From research done by [9], there are no predefined learning rates but should be between 0 and 1.0 so there is a need to balance and there is no better way to do that other than to test several learning rates and observe their performances. This work adopts the use of online training instead of batch training. This is because batch training needs more time compared to online training with no corresponding improvement inaccuracy [5].

Nonetheless, there is a limit to the times one can decrease the learning rate. To avoid wasting time at such points, one should avoid repeating the same steps while taking the same path that results in the same minimum. Learning rate affects the testing and training accuracies of CNN and therefore researchers have to explore different learning rates before settling on one. When the learning rate was high at 0.001, the recognition rate was low at 84% and the model experienced a lot of losses. But when the learning rate was relatively low at 0.0001, recognition rates were high at 99% and minimal loss was observed. For CNN, to show good results, at both training and recognition, this work suggests the use of a 0.0001 learning rate. There is a need to further investigate the effect of batch sizes on test accuracies using adjusted learning rates ranging between 0 and 1.

References

[1] Bengio, Yoshua. "Practical recommendations for gradient-based training of deep architectures." In Neural networks: Tricks of the trade, pp. 437-478. Springer, Berlin, Heidelberg, 2012.

[2] Goodfellow, Ian, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. Deep learning. Vol. 1, no. 2. Cambridge: MIT Press, 2016.

[3] Schaul, Tom, Sixin Zhang, and Yann LeCun. "No more pesky learning rates." In International Conference on Machine Learning, pp. 343-351. PMLR, 2013.

[4] Zeiler, Matthew D. "Adadelta: an adaptive learning rate method." arXiv preprint arXiv:1212.5701 (2012).

[5] Bergstra, James, and Yoshua Bengio. "Random search for hyper-parameter optimization." Journal of machine learning research 13, no. 2 (2012).

[6] Li, Hao, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. "Visualizing the loss landscape of neural nets." arXiv preprint arXiv:1712.09913 (2017).

[7] Andrychowicz, Marcin, Misha Denil, Sergio Gomez, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. "Learning to learn by gradient descent by gradient descent." arXiv preprint arXiv:1606.04474 (2016).

[8] Baydin, Atilim Gunes, Robert Cornish, David Martinez Rubio, Mark Schmidt, and Frank Wood. "Online learning rate adaptation with hypergradient descent." arXiv preprint arXiv:1703.04782 (2017).

[9] Z. Xu, A. M. Dai, J. Kemp, L. Metz, and M. Sun. (2019). "Learning an Adaptive Learning Rate Schedule". 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

[10] I. Goodfellow, Y. Bengio, and A. Courville. " Deep Learning". The MIT Press, 2016.

[11] Masters, Dominic, and Carlo Lucchi. "Revisiting small batch training for deep neural networks." arXiv preprint arXiv:1804.07612 (2018).

[12] Radiuk, Pavlo M. "Impact of training set batch size on the performance of convolutional neural networks for diverse datasets." Information Technology and Management Science 20, no. 1 (2017): 20-24.

[13] Daniel, Christian, Jonathan Taylor, and Sebastian Nowozin. "Learning step size controllers for robust neural network training." In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30, no. 1. 2016.

[14] Duchi, John, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." Journal of machine learning research 12, no. 7 (2011).

[15] Kingman, D. P., and J. Ba. "Adam: A Method for Stochastic Optimization. Conference paper." In 3rd International Conference for Learning Representations. 2015.

[16] Schmidhuber, Jürgen. "Deep learning in neural networks: An overview." Neural networks 61 (2015): 85-117.

[17] Leung, Henry, and Simon Haykin. "The complex backpropagation algorithm." IEEE Transactions on signal processing 39, no. 9 (1991): 2101-2104.

[18] Sangati, Federico, and Stefania Costantini. "International Journal of Innovative Technology and Exploring Engineering (IJITEE)."

[19] Ruder, Sebastian. "An overview of gradient descent optimization algorithms." arXiv preprint arXiv:1609.04747 (2016).

[20] Darken, Christian, Joseph Chang, and John Moody. "Learning rate schedules for faster stochastic gradient search." In Neural networks for signal processing, vol. 2. 1992.

[21] Sabour, Sara, Nicholas Frosst, and Geoffrey E. Hinton. "Dynamic routing between capsules." arXiv preprint arXiv:1710.09829 (2017).

[22] Hinton, Geoffrey E., Sara Sabour, and Nicholas Frosst. "Matrix capsules with EM routing." In International conference on learning representations. 2018.

[23] Dou, Zi-Yi, Zhaopeng Tu, Xing Wang, Longyue Wang, Shuming Shi, and Tong Zhang. "Dynamic layer aggregation for neural machine translation with routing-by-agreement." In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, no. 01, pp. 86-93. 2019.

[24] Mobiny, Aryan, and Hien Van Nguyen. "Fast capsnet for lung cancer screening." In International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 741-749. Springer, Cham, 2018.

[25] Gogola, Ewa, Alexandra A. Duarte, Julian R. de Ruiter, Wouter W. Wiegant, Jonas A. Schmid, Roebi de Bruijn, Dominic I. James et al. "Selective loss of PARG restores PARylation and counteracts PARP inhibitor-mediated synthetic lethality." Cancer cell 33, no. 6 (2018): 1078-1093.

[26] Bache, Kevin, Dennis DeCoste, and Padhraic Smyth. "Hot swapping for online adaptation of optimization hyperparameters." arXiv preprint arXiv:1412.6599 (2014).

[27] Chen, Yie-Ruey, Jing-Wen Chen, Shun-Chieh Hsieh, and Po-Ning Ni. "The application of remote sensing technology to the interpretation of land use for rainfall-induced landslides based on genetic algorithms and artificial neural networks." IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 2, no. 2 (2009): 87-95.

[28] Ariyadi, Rachmad, Mauridi Hery Purnomo, Nana Ramadijanti, and Bima Sena Bayu Dewantara. "Pengenalan Rasa Lapar Melalui Suara Tangis Bayi Umur 0-9 Bulan Dengan Menggunakan Neural Network (Sub Judul: Penapisan Dengan Transformasi Wavelet Kontinyu)." eepis final project (2010).

[29] Satti, Vijay, Anshul Satya, and Shanu Sharma. "An automatic leaf recognition system for plant identification using machine vision technology." International journal of engineering science and technology 5, no. 4 (2013): 874.

[30] Brahimi, Mohammed, Kamel Boukhalfa, and Abdelouahab Moussaoui. "Deep learning for tomato diseases: classification and symptoms visualization." Applied Artificial Intelligence 31, no. 4 (2017): 299-315.

[31] Llorca, Charmaine, May Elsbeth Yares, and Christian Maderazo. "Image-based pest and disease recognition of tomato plants using a convolutional neural network." In Proceedings of international conference technological challenges for better world. 2018.

[32] Ferentinos, Konstantinos P. "Deep learning models for plant disease detection and diagnosis." Computers and Electronics in Agriculture 145 (2018): 311-318.

[33] Mohanty, S. P., and D. P. Hughes. "Salathé Marcel (2016). Using deep learning for image-based plant disease detection." Frontiers in Plant Science 7: 1419.

[34] Kaur, Sukhvir, Shreelekha Pandey, and Shivani Goel. "Plants disease identification and classification through leaf images: A survey." Archives of Computational Methods in Engineering 26, no. 2 (2019): 507-530.

[35] Chaki, Jyotismita, and Ranjan Parekh. "Plant leaf recognition using shape based features and neural network classifiers." International Journal of Advanced Computer Science and Applications 2, no. 10 (2011).

[36] Sladojevic, Srdjan, Marko Arsenovic, Andras Anderla, Dubravko Culibrk, and Darko Stefanovic. "Deep neural networks based recognition of plant diseases by leaf image classification." Computational intelligence and neuroscience 2016 (2016).

[37] Lowe, Amy, Nicola Harrison, and Andrew P. French. "Hyperspectral image analysis techniques for the detection and classification of the early onset of plant disease and stress." Plant methods 13, no. 1 (2017): 1-12.

[38] Kamilaris, Andreas, and Francesc X. Prenafeta-Boldú. "Deep learning in agriculture: A survey." Computers and electronics in agriculture 147 (2018): 70-90.

[39] Barbedo, Jayme GA. "Factors influencing the use of deep learning for plant disease recognition." Biosystems engineering 172 (2018): 84-91.

[40] Smith, Leslie N., and Nicholay Topin. "Super-convergence: Very fast training of neural networks using large learning rates." In Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications, vol. 11006, p. 1100612. International Society for Optics and Photonics, 2019.

[41] Smith, Leslie N. "Cyclical learning rates for training neural networks." In 2017 IEEE winter conference on applications of computer vision (WACV), pp. 464-472. IEEE, 2017.

[42] Goyal, Priya, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. "Accurate, large minibatch sgd: Training imagenet in 1 hour." arXiv preprint arXiv:1706.02677 (2017).

[43] Zulkifli, Hafidz. "Understanding learning rates and how it improves performance in deep learning." Towards Data Science 21 (2018): 23.

[44] Bache, Kevin, Dennis DeCoste, and Padhraic Smyth. "Hot swapping for online adaptation of optimization hyperparameters." arXiv preprint arXiv:1412.6599 (2014).

[45] Theodoridis, Sergios. "Machine Learning: A Bayesian and Optimization Perspective. NET Developers Series." (2015).

[46] Jastrzębski, Stanisław, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. "Three factors influencing minima in sgd." arXiv preprint arXiv:1711.04623 (2017).

[47] Kurita, Keita, Paul Michel, and Graham Neubig. "Weight poisoning attacks on pre-trained models." arXiv preprint arXiv:2004.06660 (2020).

[48] Blier, Léonard, Pierre Wolinski, and Yann Ollivier. "Learning with random learning rates." In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 449-464. Springer, Cham, 2019.

[49] Surmenok, Pavel. "Estimating an optimal learning rate for a deep neural network." Towards Data Science (2017).

[50] Pal, Dipan K., and Marios Savvides. "Non-parametric transformation networks." arXiv preprint arXiv:1801.04520 (2018).

[51] Wilson, D. Randall, and Tony R. Martinez. "The need for small learning rates on large problems." In IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222), vol. 1, pp. 115-119. IEEE, 2001.

[52] Ariyadi, Rachmad, Mauridi Hery Purnomo, Nana Ramadijanti, and Bima Sena Bayu Dewantara. "Pengenalan Rasa Lapar Melalui Suara Tangis Bayi Umur 0-9 Bulan Dengan Menggunakan Neural Network (Sub Judul: Penapisan Dengan Transformasi Wavelet Kontinyu)." eepis final project (2010).

[53] Kandel, Ibrahem, and Mauro Castelli. "The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset." ICT express 6, no. 4 (2020): 312-315.

[54] Sabour, Sara, Nicholas Frosst, and Geoffrey E. Hinton. "Dynamic routing between capsules." arXiv preprint arXiv:1710.09829 (2017).

[55] Xi, Edgar, Selina Bing, and Yang Jin. "Capsule network performance on complex data." arXiv preprint arXiv:1712.03480 (2017).

[56] Pal, Dipan K., and Marios Savvides. "Non-parametric transformation networks." arXiv preprint arXiv:1801.04520 (2018).

[57] Chen, Yie-Ruey, Jing-Wen Chen, Shun-Chieh Hsieh, and Po-Ning Ni. "The application of remote sensing technology to the interpretation of land use for rainfall-induced landslides based on genetic algorithms and artificial neural networks." IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 2, no. 2 (2009): 87-95.

[58] Arif, Nursida, and Projo Danoedoro. "An Analyze of A Backpropagation Neural Network in The Identification of Critical Land Based on ALOS Imagery." In Proc. Int. Conf. on 34th Asian Conference on Remote Sensing, vol. 1, pp. 589-593. 2013.

[59] Amara, Jihen, Bassem Bouaziz, and Alsayed Algergawy. "A deep learning-based approach for banana leaf diseases classification." Datenbanksysteme für Business, Technologie und Web (BTW 2017)-Workshopband (2017).

[60] Clement, Damien, Monna Arvinen-Barrow, and Tera Fetty. "Psychosocial responses during different phases of sport-injury rehabilitation: a qualitative study." Journal of athletic training 50, no. 1 (2015): 95-104.

[61] Atole, Ronnel R., and Daechul Park. "A multiclass deep convolutional neural network classifier for detection of common rice plant anomalies." International Journal of Advanced Computer Science and Applications 9, no. 1 (2018): 67-70.

[62] Cui, Di, Qin Zhang, Minzan Li, Glen L. Hartman, and Youfu Zhao. "Image processing methods for quantitatively detecting soybean rust from multispectral images." Biosystems engineering 107, no. 3 (2010): 186-193.

[63] Barbedo, Jayme GA. "Factors influencing the use of deep learning for plant disease recognition." Biosystems engineering 172 (2018): 84-91.

[64] Fuentes, Alvaro, Sook Yoon, Sang Cheol Kim, and Dong Sun Park. "A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition." Sensors 17, no. 9 (2017): 2022.

[65] Mohanty, Sharada P., David P. Hughes, and Marcel Salathé. "Using deep learning for image-based plant disease detection." Frontiers in plant science 7 (2016): 1419.

[66] Vogelmeier, Claus F., Gerard J. Criner, Fernando J. Martinez, Antonio Anzueto, Peter J. Barnes, Jean Bourbeau, Bartolome R. Celli et al. "Global strategy for the diagnosis, management, and prevention of chronic obstructive lung disease 2017 report. GOLD executive summary." American journal of respiratory and critical care medicine 195, no. 5 (2017): 557-582.

[67] Ferentinos, Konstantinos P. "Deep learning models for plant disease detection and diagnosis." Computers and Electronics in Agriculture 145 (2018): 311-318.

[68] Syamsuri, Burhanudin, and Gede Putra Kusuma. "Plant Disease Classification using Lite Pretrained Deep Convolutional Neural Network on Android Mobile Device." International Journal of Innovative Technology and Exploring Engineering 9 (2019).