Proceedings of the Kabarak University International Conference On Computing And Information Systems, 4th - 8th October 2021 Nakuru, Kenya.

Empirical Evaluation of Adaptive Optimization on the Generalization Performance of Convolutional Neural Networks

Stephen WANJAU[1], Geoffrey WAMBUGU[1], Aaron OIRERE[1]
[1] *Murang'a University of Technology, P.O. Box 75 - 10200, Murang'a, Kenya*
Tel: +254 740 910 882, Email: steve.kahara@gmail.com

Abstract: Recently, deep learning based techniques have garnered significant interest and popularity in a variety of fields of research due to their effectiveness in search for an optimal solution given a finite amount of data. However, the optimization of these networks has become more challenging as neural networks become deeper and datasets growing larger. The choice of the algorithm to optimize a neural network is one of the most important steps in model design and training in order to obtain a model that will generalize well on new, previously unseen data. In deep learning, adaptive gradient optimization methods are mostly preferred for supervised and unsupervised tasks. First, they accelerate the training of neural networks and since mini batches are selected randomly and are independent, an unbiased estimate of the expected gradient can be computed. This paper examined six state-of-the-art adaptive gradient optimization algorithms, namely, AdaMax, AdaGrad, AdaDelta, RMSProp, Nadam, and Adam on the generalization performance of convolutional neural networks (CNN) architecture that are extensively used in computer vision tasks. Experiments were conducted giving comparative analysis on the behaviour of these algorithms during model training on three large image datasets, namely, Fashion-MNIST, Kaggle Flowers Recognition and Scene classification. The results show that Adam, Adadelta and Nadam finds the global minimum faster in the experiments, have a better convergence curve, and higher test set accuracy in experiments using the three datasets. These optimization approaches adaptively tune the learning rate based only on the recent gradients; thus, controlling the reliance of the update on the past few gradients.

Keywords: Adaptive gradient methods; optimization; deep learning; convolutional neural networks; image processing.

1. Introduction

The design goal of any machine learning classifier and any learning algorithm in general, based on a training set of a finite size, is to ultimately provide a good generalization performance (Theodoridis, 2020). The generalization performance is quantified by the difference between the training error and the test error where good machine learning algorithms are those where the test error and the training error have close values (Hu & Zheng, 2019). Recently, deep learning algorithms have shown to have better generalization performance than traditional machine learning techniques in solving classification problems. However, explaining why these highly non-convex models trained by a specific optimization algorithm can generalize better has become a significant open question in deep learning (Wang, Meng, Chen, & Liu, 2021).

In the literature, different approaches have been employed to improve on the generalization performance of neural networks. Among these approaches, some explore the implicit regularization property of Stochastic Gradient Descent (SGD) (Hardt, Recht, & Singer, 2016; Zhang, Bengio, Hardt, Recht, & Vinyals, 2017); Neyshabur, Tomioka, Salakhutdinov, & Srebro, 2017). Another perspective relies on the geometry of loss function around a global minimum with the work of Keskar, Mudigere, Nocedal, Smelyanskiy, and Tang (2017) adopting this perspective to explain why small-batch SGD often converges to the solution generalizing better than large-batch SGD similar to the work of (Baldassi, et al., 2016) where discrete networks were considered.

Optimization is a significant component in deep learning. Optimization provides an approach to minimizing the loss function, often referred to as the objective function in stochastic nonconvex optimization (Kingma & Ba, 2015). Optimization considers different methods and algorithms used for learning the underlying mapping from input data to outputs by choosing the right set of parameters that will reduce the error during model training (Marin, Skelin, & Grujic, 2019). Through optimization, researchers seek to find a suitable model, which will generalize well on new, previously unseen data. Stochastic gradient descent (SGD) with mini-batches and its variant are undoubtedly the most prevalent methods for training deep neural networks, owing to its simplicity and greater performance than the alternatives (Wang & Srebro, 2019). For instance, the use of minibatch optimization algorithms makes the training process more stable as it reduces the variance of gradient estimate. These algorithms take smaller number of updates if a larger mini-batch size is used. Moreover, the backpropagation procedure on a larger mini-batch can utilize massive parallelization of linear algebra routines provided by advanced computational hardware such as graphical processing units and clusters (Hu & Zheng, 2019).

Stochastic optimization methods are the dominant techniques in the training of deep neural networks. In this paper, we explore the commonly used adaptive optimization algorithms and conduct empirical analysis of their effect on the training process and the final generalization performance of deep learning models. In particular, convolutional neural networks (CNNs) are considered. Convolutional neural networks are one of the popular deep learning models that have a wide range of applications in the field of computer vision. A comparative analysis on the validation accuracy, as well as the model loss (train and test) of each optimization algorithm in the generation of an optimization solution is done using three large image datasets, namely, Fashion MNIST, Kaggle Flowers and Scene classification.

The rest of the paper is structured as follows. Section 2 defines the problem statement giving clarity to the thrust of our paper. Section 3 highlights the objectives of the study and section 4 review literature on the concept of optimization, gives a theoretical background on nonconvex stochastic optimization with a focus on adaptive optimization algorithms used for training deep neural networks and highlights optimization challenges in deep learning. Section 5 describes the methodology used. Section 6 provides the results and discussion on the comparative analysis of the different adaptive optimization algorithms on the image classification problem on the training process and generalization performance. In section 7, conclusion is made and in section 8, recommendations for future work are provided.

2. Problem Statement

Optimizing neural networks has become more challenging as the neural networks become deeper and datasets growing larger (Marin, Skelin, & Grujic, 2019). The well-known challenges in training deep neural networks are local minima, saddle points and the explosion/vanishing gradient attributable to the concatenation of many layers (Tan & Lim, 2019). During network training, an optimization algorithm iteratively steps across the search space, updating the weights. The algorithm seeks possible values for the model in order to

obtain a set of weights that results in good performance. The choice of the best algorithm to optimize the neural network is one of the most important steps in model design and training so as to obtain a model that will generalize well on new, previously unseen data. Adaptive optimization methods such as AdaGrad, RMSProp and Adam are mostly preferred for supervised and unsupervised learning tasks (Reddi, Zaheer, Sachan, Kale, & Kumar, 2018; Liang, Ma, & Li, 2020). Numerous works have provided empirical evidence that adaptive optimization methods may suffer from poor generalization performance (Wang, Meng, Chen, & Liu, 2021). However, empirical evaluation on the generalization of adaptive optimization algorithms is still lacking. Additionally, adaptive methods have been studied in the convex settings but their analysis in the non-convex settings is still in the nascent stages (Duchi, Hazan, & Singer, 2011; Li, Zhao, Arora, Liu, & Haupt, 2016).

3. Objectives

The main objective of this paper is to present an empirical evaluation of adaptive optimization on the generalization performance of CNNs. The specific objectives are:

(a) To review literature on the adaptive optimization approaches employed to improve on the generalization performance of deep learning networks.
(b) To analyse the impact of adaptive optimization on the generalization performance of convolutional neural networks
(c) To determine the best optimization algorithms that can be used to obtain a model that will generalize well on new, previously unseen data.

4. Literature Review

The crux of machine learning algorithms is to develop an optimized model capable of learning the parameters in the objective function and the constraints placed from the given dataset. Several effective optimization methods have been put forward to stimulate development of machine learning, consequently improving their performance and efficiency (Shone, Ngoc, Phai, & Shi, 2018). According to Zhou (2018), majority of neural network applications are naturally formulated as non-convex optimization due to the complex mechanism of the underlying model. In addition, neural networks have many symmetric configurations such as exchanging intermediate neurons, hence non-convex.

With the increasing interest in deep learning applications, researchers have deemed it necessary to deal with non-convex optimization progressively, more particularly because of the benefits hidden behind their complexity. By definition, a non-convex optimization is any problem where the objective or any of the constraints are non-convex (Jain & Kar, 2017) predominantly because such algorithms operate in high-dimensional spaces. The freedom to express the learning problem as a non-convex optimization problem gives immense modelling power to the algorithm designer (Mehdi, 2020).

Training a deep neural network can be described as an optimization problem with non-convex objective function. The non-convex deep neural networks have been found to have large amount of global minima (Choromanska, Henaff, Mathieu, Arous, & LeCun, 2015) with only a few able to guarantee satisfactory generalization property (Brutzkus, Globerson, Malach, & Shalev-Shwartz, 2018). During the training process, model parameters are iteratively updated in order to reduce the cost on the training data. Optimization methods are used to optimize noisy functions, i.e., track key parameters of interest in data streams, which can have changing dynamics over time. Stochastic methods are mainly employed in non-convex problems where robust results have been demonstrated (Reddi, et al., 2016; Jain & Kar, 2017; Mehdi, 2020). The adaptive nature of stochastic approximation methods such as stochastic gradient descent make them highly applicable in a range of applications,

particularly in machine learning (Reddi, Zaheer, Sachan, Kale, & Kumar, 2018). Various optimization algorithms exist in the literature for training neural networks and they vary in the way they update network parameters.

4.1 Nonconvex Stochastic Optimization

The literature on stochastic optimization is vast. In this work, the stochastic gradient descent and its variant for smooth non-convex problems are analysed by (Ghadimi, Lan, & Zhang, 2016). According to Huang and Chen (2019), stochastic gradient descent is an efficient method for solving the following optimization problem, which is fundamental to machine learning,

Eq. (1)

$$\min_{x \in R^d} f(x) + g(x)$$

Where $f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$ indicates the loss function, and $g(x)$denotes the regularization function. However, when sample size n is large, even the model optimization becomes computationally burdensome. This is because SGD computes gradient of one sample instead of all samples in each iteration, and therefore it has only per-iteration complexity of $O(d)$.

Moreover, regardless of its scalability, stochastic gradient is much noisier than the batch gradient due to the existence of variance in stochastic process (Ghadimi, Lan, & Zhang, 2016). Recently, adaptive gradient optimization algorithms have successfully been applied to reduce this variance. They adaptively tune the learning rate based only on the recent gradients; thus controlling the reliance of the update on the past few gradients. These algorithms leverage the curvature of the objective function that yields adaptive coordinate-wise learning rates, which leads to faster model convergence (Zhou, Karimi, Yu, Xu, & Li, 2020). The following section discusses the adaptive optimization techniques.

4.2 Adaptive Optimization Algorithms

In deep learning, training a model is computationally expensive, exhibits slow convergence, and normally takes plenty of time. Numerous optimization algorithms are increasingly being developed that produce the adaptive learning rate factor by global estimation of the gradient (Liu, Feng, Li, Wang, & Wu, 2021). Researchers become drained with setting up the learning rates in an effort to optimize the models. Hence, the adaptive optimization techniques were developed.

Adaptive optimization techniques do not require setting the learning rate since they are dynamically adjusted during model training (Liang, Ma, & Li, 2020). Researchers need to initialize the learning rate parameters and the adaptive optimization algorithms keep updating learning rates during model training (Reddi, et al., 2018). The fundamental issue for the success of adaptive optimization algorithms is to design better conditioners of the gradient (Wang, Meng, Chen, & Liu, 2021). The following six adaptive optimization techniques namely, AdaGrad, RMSProp, AdaDelta, Adam, AdaMax and Nadam algorithms described in the literature are considered. These algorithms have been successfully employed in a plethora of applications, achieving status of the art results (Reddi, et al., 2018).

4.2.1 Adagrad Optimizer

AdaGrad (Duchi, Hazan, & Singer, 2011) is the first algorithm designed that can independently adapt to the learning rate of all hyperparameters. Adagrad works on setting the learning rate by dividing the learning rate component by the square root of the cumulative sum of the current gradient and the previous gradient. In other words, Adagrad calculates the step size for each parameter by first summing the partial derivatives for the parameter seen so far during the search, then dividing the initial step size hyperparameter by the square root of the sum of the squared partial derivatives. According to (Liang, Ma, & Li, 2020) the main weakness of Adagrad is that the continuous accumulation of past gradients will make the learning rate very small, which will lead to the inability to effectively update the parameters, and will be difficult to obtain useful information.

4.2.2 RMSProp

Root Mean Squared Propagation (RMSProp) (Hinton, Srivastava, & Swersky, 2012) was proposed to achieve a rapid training process with an element-wise scaling term on learning rates. The technique tries to resolve the problem that gradients may vary widely in magnitudes. The method uses a decaying average of partial gradients in the adaptation of the step size for each parameter. In this way, the method focus is on the most recently observed partial gradients seen during the progress of the search, overcoming the limitation of AdaGrad method.

4.2.3 Adadelta Optimizer

Adadelta (Zeiler, 2012) works on exponential moving averages of the squared delta's. Delta refers to the difference between the current weight and the newly updated weight. Adadelta optimizer uses momentum techniques to deal with the monotonically decreasing learning rate problem. The technique removes the learning rate parameter and replaces it with delta. This leads to the learning rate being more stable, and the algorithm overall becomes more robust.

4.2.4 Adam

The Adaptive Momentum (Adam) optimization technique proposed by Kingma and Ba (2015) is the most popular method used in for optimizing many neural network models. Adam is a combination of RMSprop and momentum, that uses the squared gradient to scale the learning rate parameters like RMSprop and it works similar to the momentum by adding averages of moving gradients. It computes different parameters for individual parameters.

4.2.5 AdaMax

AdaMax (Luo, Xiong, Liu, & Sun, 2018) is variant of Adam that uses dynamic boundaries of learning rates. AdaMax is as fast as Adam and uses an exponentially weighted infinity norm instead of the second-order moment estimate (Yi, Ahn, & Ji, 2020) when scaling gradient.

4.2.6 NADAM

The Nesterov-accelerated Adaptive Moment Estimation (Nadam) is a blend of Adam and Nesterov accelerated algorithms. The idea behind this type of algorithm is to increase and decrease the decay factor β over time. A series of parameters $\beta_1, \beta_2, \ldots, \beta_t$ corresponding respectively to steps $1, 2, \ldots, t$ is considered for better clarity (Mustapha, Mohamed, & Ali, 2021). The application of the momentum step in step $t+1$ is applied once updating the step t as an alternative of$t+1$.

4.3 Optimization Challenges in Deep Learning

Numerical optimization methods systematically vary the inputs to a function so as to realize the minimum or maximum value of the function. Normally, this requires numerous

evaluations of the objective function (Fike, Jongsma, Alonso, & Weide, 2011). Optimization methods that utilize gradient information usually converge to the optimal solution in fewer iterations than methods that only rely on the function values and tends to drive parameters to certain kinds of global minima, which generalize well. However, optimization of deep neural networks comes with some challenges (Marin, Skelin, & Grujic, 2019). In the literature, the most vexing ones are local minima, saddle points, and vanishing gradients.

4.3.1 Local Minima

The convergence of the optimization algorithm should be ensured to avoid falling into local optimum (Hu & Zheng, 2019). Researchers have empirically demonstrated that different local optima, attained from training deep neural networks do not generalize in a similar manner for the unseen data sets, albeit achieving similar training loss (Wang, Keskar, Xiong, & Socher, 2018). Figure 1 shows local minima in deep learning training.

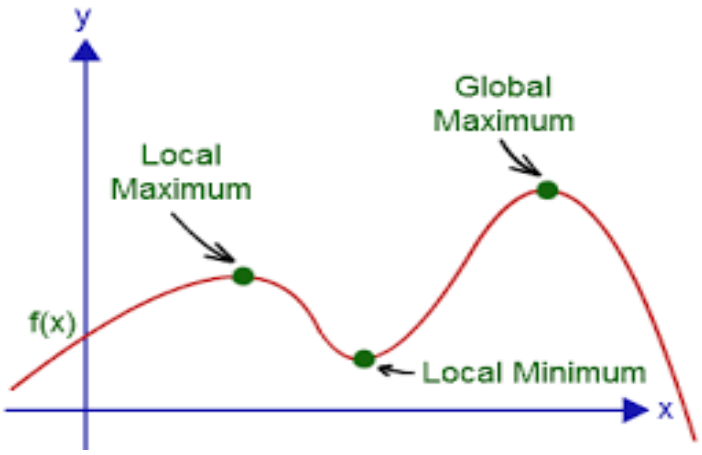


***Figure 1: Local Minima in non-convex functions (Source:** (MyRank, 2018))*

Experiments conducted with different batch sizes showed that sharp minima do correlate with higher generalization error. According to Keskar, et al. (2017) and Chaudhari, et al. (2019), the generalization ability of a model is associated with the spectrum of the Hessian matrix (Fike, Jongsma, Alonso, & Weide, 2011), a square matrix that sort out all the second partial derivatives of a given multivariable function, evaluated at the solution. Large eigenvalues of the hessian matrix often leads to poor model generalization (Wang, Keskar, Xiong, & Socher, 2018).

Give the objective function $f(x)$, if the value of $f(x)$, at x is smaller than the values of $f(x)$, at any other points in the vicinity of x, then $f(x)$ could be a local minimum. On the other hand, if the value of $f(x)$, at x is the minimum of the objective function over the entire domain, then $f(x)$ is the global minimum.

4.3.2 Saddle Points

A saddle point is any location where all gradients of a function vanish but which is neither a global nor a local minimum**.** Saddle points in higher dimensions problems are even more insidious since the likelihood that at least some the eigenvalues of the function's Hessian matrix are negative is quite high (Dauphin, et al., 2014). Figure 2 demonstrates sandle points in non-convex functions.

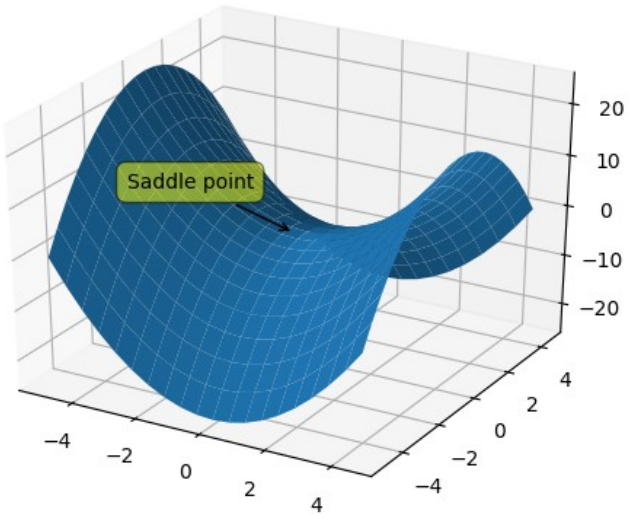


Figure 2: Saddle points in non-convex functions (source: (Zadeh, 2016))

Such saddle points are surrounded by high error plateaus that can dramatically slow down learning, and give the illusive impression of the existence of a local minimum (Jin, Ge, Netrapalli, Kakade, & Jordan, 2017). Previous works elucidates why gradient descent optimization avoids saddle points in the non-convex settings. For instance, (Dauphin, et al., 2014) proposed a saddle-free Newton method that scales gradients by the absolute value of the inverse Hessian. Their work demonstrated that near saddle points, they could achieve rapid escape by combining the best of gradient descent and Newton methods while avoiding the pitfalls of both.

4.3.3 Vanishing Gradients

The vanishing gradient is the most insidious problem encountered in deep learning. The problem is mostly encountered when training deep learning models using gradient-based learning methods and backpropagation (Tan & Lim, 2019). As more layers using certain activation functions are added to a neural network, the gradients of the loss function approaches zero, making the network hard to train (Wang C.-F. , 2019). Figure 3 shows the vanishing gradient problem in training deep neural networks using the sigmoid activation function.

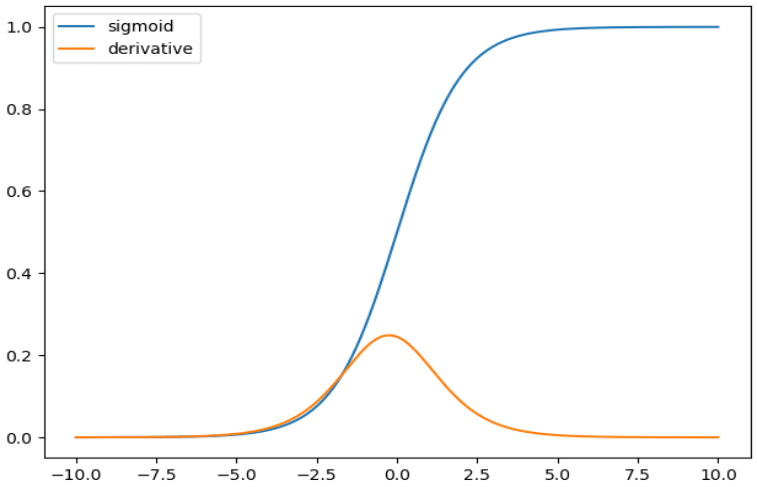


Figure 3: Vanishing gradient problem (Source: (Khan, 2020))

The idea is that the calculated partial derivatives is used to compute the gradient as one goes deeper into the neural network. Since the gradients control how much the network learns during training, if the gradients are very small or zero, then little to no training can take place, leading to poor predictive performance. In the literature, Rectified linear units (ReLUs) have been proposed for use during model training as they suffer less from the vanishing gradient problem compared to sigmoidal activation functions that are known to saturate in both directions (Brownlee, 2019).

5. Materials and Methods

This section presents the experimental design of the study. The setup and settings used in the experiments, the deep learning architecture used, the hyper-parameter setting, and the datasets used are described.

5.1 Experiments

The aim of this experimental study was to compare the effect of adaptive optimizers on the training process and final generalization performance of classification models. Convolutional neural networks, deep learning architectures were train on three datasets. For implementation, a simulation experiment environment (Vieira, Koch, Sobral, Westphall, & de Souza Leao, 2019) was built on a 64-bit computer based on Win10 operating system platform with Intel® i7-9750 Hz 2.60 GHz CPU, 8 GB RAM, NVIDIA GeForce RTX 2060 6G GDDR6 GPU, and 10.2 CUDA. The experiments were written in Python language and the Scikit-learn and NumPy were used for data pre-processing. The neural networks was built based on the Tensorflow and Keras deep learning frameworks.

5.2 CNN Model Architectures

The network architecture plays a critical role in the performance of deep learning models. In this study, two well-known CNN model architectures were considered and further fine-tuned to ensure effective training process (Gong, Wang, Guo, & Lazebnik, 2014). The output layer for each model was fine-tuned to reflect the classification output expected from the chosen dataset. Model 1 architecture was based on the VGG-16 architecture developed by the Visual Geometry Group from the University of Oxford in 2014. VGG-16 (Simonyan & Zisserman, 2014) served as the mainstream CNN model for feature representation and classification and was used to win the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2014.

The architecture consists of 16 layers, where convolutional layers (13) with 3x3 filters and 2x2 max pooling layers are stacked and dense layers incorporated before the output layer. The model contains a set of weights that are used to train the images. Table 1 shows the summary of the fine-tuned VGG-16 model that was used to train on the Kaggle Flowers recognition dataset.

Table 1: Architecture of the fine-tuned VGG-16 CNN Model

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 224, 224, 32)	896
conv2d_2 (Conv2D)	(None, 224, 224, 64)	18496
max_pooling2d_1 (MaxPooling2	(None, 112, 112, 64)	0

dropout_1 (Dropout)	(None, 112, 112, 64)	0
conv2d_3 (Conv2D)	(None, 112, 112, 128)	73856
conv2d_4 (Conv2D)	(None, 110, 110, 256)	295168
max_pooling2d_2 (MaxPooling2	(None, 36, 36, 256)	0
dropout_2 (Dropout)	(None, 36, 36, 256)	0
flatten_1 (Flatten)	(None, 331776)	0
dense_1 (Dense)	(None, 256)	84934912
re_lu_1 (ReLU)	(None, 256)	0
dropout_3 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 256)	65792
re_lu_2 (ReLU)	(None, 256)	0
dense_3 (Dense)	(None, 5)	1285

Total params: 85,390,405
Trainable params: 85,390,405
Non-trainable params: 0

Model 2 architecture, AlexNet, won the ImageNet large-scale visual recognition challenge in 2012 (Krizhevsky, Sutskever, & Hinton, 2012). The architecture is capable of achieving high accuracies on very challenging datasets. AlexNet has eight layers with learnable parameters comprising of 5 convolution layers with a combination of max-pooling layers and 3 fully connected layers. The model has an image input image of size 227×227x3. Images need to be converted to 227×227x3 before using it for training the network. If the input image is grayscale, it is converted to an RGB image by replicating the single channel to obtain a 3-channel RGB image. The authors introduced padding to prevent the size of the feature maps from reducing drastically. Table 2 shows the architecture of AlexNet.

Table 2: Architecture of the fine-tuned AlexNet Model

Layer	Size of Feature Map	#filters or neurons	Filter Size	Stride	Padding	Activation
Input	227x227x3	-	-	-	-	-
Convolution 1	55x55x96	96	11x11	4	-	relu
Maxpooling 1	27x27x96	-	3x3	2	-	-
Convolution 2	27x27x256	256	5x5	1	2	relu
Maxpooling 2	13x13x256	-	3x3	2	-	-
Convolution 3	13x13x384	384	3x3	1	1	relu
Convolution 4	13x13x384	384	3x3	1	1	relu
Convolution 5	13x13x256	256	3x3	1	1	relu
Max pooling 3	6x6x256	-	3x3	2	-	-
Dropout 1	6x6x256	rate=0.5	-	-	-	-

5.3 Hyperparameter settings

The definitive solution of the gradient descent optimization algorithms substantially is contingent on the choice of the hyperparameters (Liang, Ma, & Li, 2020). The hyperparameters of optimizers used for training each model on the three datasets are shown in table 3 which are recommended in practice (Kingma & Ba, 2015) and employs a highly tuned learning rate schedule.

Table 3: Hyperparameters for Adaptive Optimizers

Optimizer	*CNN Model 1*	*CNN Model 2*
Adagrad	lr = 0.05, $\epsilon = 10^{-7}$	lr = 0.05, $\epsilon = 10^{-7}$
Adam	lr = 0.0005, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$	lr = 0.0005, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$
Adadelta	lr = 0.0001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$	lr = 0.0001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$
Nadam	lr = 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$	lr = 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$
AdaMax	lr = 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$	lr = 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$
RMSProp	lr = 0.001, $\rho = 0.9$, $\epsilon = 10^{-7}$	lr = 0.0001, $\rho = 0.9$, $\epsilon = 10^{-7}$

The study used 50 epochs for model training while the grid search method was used to tune the step size for each optimizer. The models were trained with mini-batches of size 40. All the experiments utilized the ReduceLROnPlateau schedule with patience of 8 epochs and a decay factor of 0.5.

5.4 Datasets

In this study, three large image datasets, namely, MNIST, Kaggle Flowers and Scene classification were used. The original training data were split into two parts: training data and validation data, where 30% of original training data were used for validation and the rest for training.

5.4.1 Fashion-MNIST Dataset

Fashion-MNIST (Xiao, Rasul, & Vollgraf, 2017) dataset comprises of 70,000 of 28X28 grayscale images of fashion products (clothes and shoes) from 10 different categories.

5.4.2 Kaggle Flowers Recognition Dataset

This dataset contains 4,242 images of labelled flowers. The pictures are divided into five classes: daisy, tulip, rose, sunflower and dandelion. For each class, there are about 800 photos. Photos are not in high resolution, 320x240 pixels.

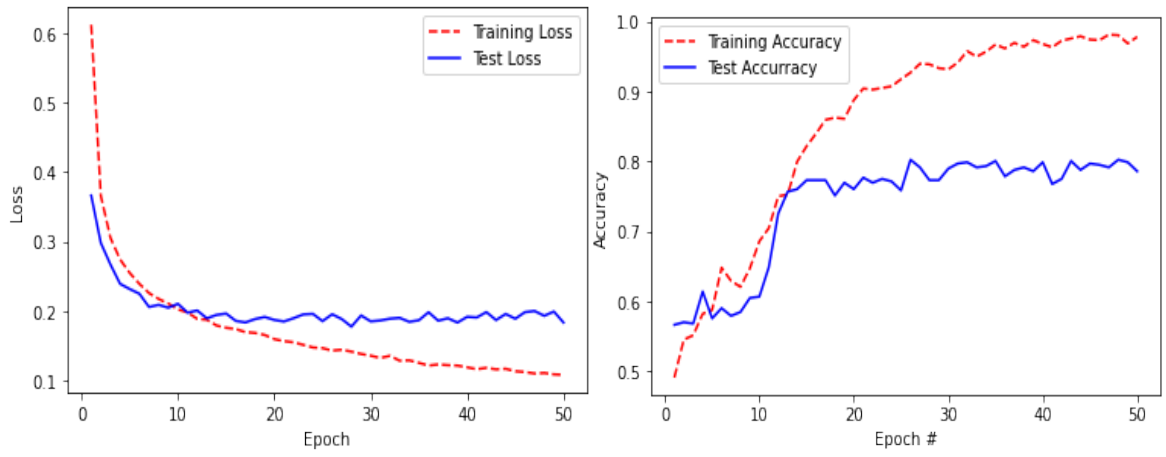
5.4.3 Scene Classification Dataset

The 15 Scene classification dataset (Lazebnik, Schmid, & Ponce, 2006), is a popular dataset with 15 natural and indoor categories. The categories include office, kitchen, living room, bedroom, store, industrial, tall building, inside city, street, highway, coast, open country, mountain, forest, and suburb. The dataset contains 4,485 images, whose average image size is 300×250 pixels. The main sources of the pictures include the COREL collection, personal photographs, and Google image search. The objective is to classifying a scene image into one of the predefined scene categories by comprehending the entire image.

6. Results and Discussion

Training the deep learning models simply meant determining good values for all the weights and the bias from the labelled examples. The goal is to find a set of weights and biases that have low loss, on average, across all examples. Adaptive gradient methods widely used in deep learning, namely, AdaGrad, RMSProp, AdaDelta, Adam, AdaMax and Nadam were used to adapt weights and learning rate of the networks, thus reduce the losses and provide the most accurate results possible.

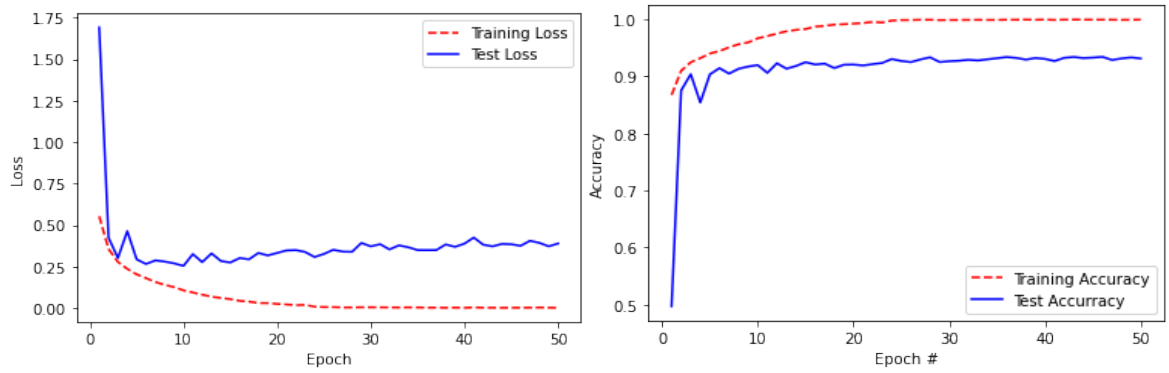
The observations regarding the influence of the adaptive optimization algorithms used on the behaviour and final generalization performance of the CNN model are based on the empirical evaluations on the two CNN model architectures, each trained on the three datasets. Figure 4 shows the loss and training accuracy of the CNN model 1 using the Adam optimizer on the Scene classification dataset after training the model using the Scene dataset.



Loss (b) Accuracy

Figure 4: Loss and Accuracy for CNN Model 1 using Adam optimizer on the Scene recognition dataset

From the results, it can be seen that the models performed very well since the training loss and the testing loss were very close. The loss function is used to optimize the deep learning algorithm. The loss is calculated on both the training and validation set and its interpretation is based on how well the model is performing on these two sets. The model loss is the sum of errors made for each example in training or validation sets. The loss value implies how poorly or well a model behaves after each iteration of optimization. Figure 5 presents the CNN model 2 loss and training accuracy optimized using the Adadelta algorithm on the Fashion-MNIST dataset.



Loss (b) Accuracy

Figure 5: Loss and Accuracy for CNN Model 2 based on Adadelta optimizer on Fashion-MNIST dataset

The accuracy metric is used to measure the algorithm's performance in an interpretable way. Accuracy of each model was determined after the model parameters were fully optimized and calculated in the form of a percentage. Accuracy is the measure of how perfect the model's prediction is compared to the true data. In the study, the validation loss and classification accuracy result were compared based on the different optimization algorithms. The results are presented over the Fashion-MNIST, Kaggle Flowers Recognition and Scene Classification datasets. The final results are reported in Tables 4, 5 and 6.

Table 4: Performance of the models trained using the different optimization algorithms on Fashion-MNIST dataset.

	CNN Model 1				CNN Model 2			
	Loss		Accuracy (%)		Loss		Accuracy (%)	
Optimization method	Train set	Test set	Train set	Test set	Train set	Test set	Train set	Test set
Adamax	0.759	0.920	95.81	91.62	0.186	0.936	98.92	91.62
RMSProp	0.167	2.867	97.43	93.43	2.473	2.867	97.68	92.43
Nadam	1.384	1.592	96.78	92.51	0.171	1.629	99.12	92.51
Adadelta	0.942	0.951	98.08	95.43	1.149	1.519	99.66	93.15
Adam	0.467	**0.694**	99.92	**95.88**	1.154	1.124	99.76	**95.67**
Adagrad	0.824	0.931	99.37	94.92	0.194	**0.931**	99.59	94.92

As shown in table 4, using CNN model 1, the Adam optimizer had the lowest loss on the test set while Adadelta reported a better classification result while trained on the Fashion-MNIST dataset. This demonstrates the ability of Adadelta to adapt learning rates instead of accumulating all past gradients. On the other hand, Adagrad algorithm had the lowest loss on the test set and the best training accuracy obtained from the Adam optimizer. Adagrad, which uses a per-dimension learning rate based on squared past gradients, achieves significant performance on the generalization error in comparison with other algorithms in sparse settings (Reddi, et al., 2018).

Table 5: Performance of the models trained using the different optimization algorithms on Kaggle Flower Recognition dataset.

	CNN Model 1				CNN Model 2			
	Loss		Accuracy (%)		Loss		Accuracy (%)	
Optimization method	Train set	Test set	Train set	Test set	Train set	Test set	Train set	Test set
Adamax	0. 993	1.793	98.78	92.58	1.138	1.926	100.00	83.68
RMSProp	1.727	2.211	99.62	91.24	4.747	9.545	98.54	81.84
Nadam	1.7.95	1.770	97.46	91.62	1.538	**1.516**	99.56	**84.56**
Adadelta	0.667	**0.937**	98.67	**93.19**	1.411	2.218	99.89	83.65
Adam	0.511	1.324	99.52	92.56	1.156	1.532	100.00	82.93
Adagrad	1.479	1.142	99.44	91.53	1.937	2.105	99.32	82.20

On the Kaggle Flower recognition test set, the CNN Model 1, achieved 93.19 % accuracy, while optimized by the Adadelta algorithm. CNN Model 2 achieved 84.56% while optimized on the Nadam algorithm.

Table 6: Performance of the models trained using the different optimization algorithms on Scene Classification dataset.

	CNN Model 1				CNN Model 2			
	Loss		Accuracy (%)		Loss		Accuracy (%)	
Optimization method	Train set	Test set	Train set	Test set	Train set	Test set	Train set	Test set

Adamax	0.465	0.964	94.24	75.63	0.773	1.436	99.33	94.16
RMSProp	2.147	1.219	94.38	75.37	8.619	2.211	96.91	93.98
Nadam	0.745	**0.755**	96.78	**76.44**	1.593	1.022	98.45	95.16
Adadelta	0.668	0.961	94.37	75.46	0. 815	**0.931**	98.66	**98.68**
Adam	0.981	1.029	98.49	75.55	1.324	1.382	99.22	97.54
Adagrad	0.422	0.852	97.72	74.87	1.061	1.231	98.28	98.12

Table 6 gives the results of the models while tested on the Scene classification dataset. The CNN Model 1, achieved 76.44 % accuracy when optimized by the Nadam algorithm whereas CNN Model 2 achieved 98.68% while optimized on the Adadelta algorithm.

Based on the results obtained, the following observations are made:

1) In terms of accuracy, the best test set results, were obtained using the Nadam optimization algorithm and the Adaptive Momentum Estimation (Adam) and its variant Adadelta.
2) The most stable, not essentially the best, performance on validation data, especially on the loss, among the optimizers show Adagrad and Adadelta optimization algorithms.
3) RMSProp optimization algorithm, in all the cases, has considerably larger validation loss than other optimizers that consistently keeps growing. Remarkably, notwithstanding the great discrepancy between RMSProp and others optimizers' losses, its validation, and finally test set accuracy remained equitably well and comparable with others.

Adam and its variant computes adaptive learning rates for each parameter by combining the ideas of momentum and adaptive gradient and keeps an exponentially decaying average of past gradients. They are mainly referred to as first-order adaptive optimization algorithms given their super-fast convergence speed in solving large scale optimization tasks (Tao, Xia, & Li, 2019). They iteratively update parameters by moving them to the direction of the negative gradient of the cost function with non-fixed learning rate. These methods address the problem of rapid decay of learning rate by scaling down the gradient by the square roots of exponential moving average of past squared gradients (Reddi, et al., 2018). The results of the analysis shows that the effective learning rate potentially increase over time in a fairly quick manner.

7. Conclusion

In this paper, we explored the different adaptive optimization algorithms commonly used for training deep learning model architectures and compared their effect on the model's generalization performance on image classification problem. Experiments were conducted using two convolutional neural network models and the performance of each adaptive optimization method was compared while trained and tested on three large image datasets. The empirical evaluations demonstrate that the chosen optimization algorithm can positively affect model's generalization performance. The Adam, Adadelta and Nadam optimization algorithms were observed to be the best-performing algorithms on new data in our experiments. Adaptive optimization algorithms consider the cumulative changes of each parameter in their respective iterative optimization processes. They scale coordinates of the gradient by square roots of some form of averaging of the squared coordinates in the past gradients and automatically adjust the learning rate on a parameter basis. The theoretical background complemented with experimental results of the deep learning process is beneficial to anyone who seeks more in-depth insight into the fields of optimization of deep learning.

8. Recommendations for Future work

In future, it would be interesting to expand the experimental evaluations to examine the extent to which other lower-performing optimizers would compare with the adaptive optimizers in generalizing model performance. Further, future experiments can also focus on experimental evaluations on different neural network architectures and problems from different domains.

References

Baldassi, C., Borgs, C., Chayes, J., Ingrosso, A., Lucibello, C., Saglietti, L., & Zecchina, R. (2016). Unreasonable effectiveness of learning neural networks: From accessible states and robust ensembles to basic algorithmic schemes. *Proceedings of the National Academy of Sciences, 113*(48), pp. E7655–E7662.

Brownlee, J. (2019, January 8). *A Gentle Introduction to the Rectified Linear Unit (ReLU).* Retrieved July 16, 2021, from Machine Learning Mastery: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

Brutzkus, A., Globerson, A., Malach, E., & Shalev-Shwartz, S. (2018). Sgd learns over-parameterized networks that provably generalize on linearly separable data. *International Conference on Learning Representations.*

Chaudhari, P., Choromanska, A., Soatto, S., LeCun, Y., Baldassi, C., Borgs, C., . . . Zecchina, R. (2019, December 20). Entropy-SGD: biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment, 2019*(124018).

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., & LeCun, Y. (2015). The loss surfaces of multilayer networks. In *Artificial intelligence and statistics* (pp. 192-204).

Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., & Bengio, Y. (2014, January). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Advances in Neural Information Processing Systems, 4*, 2933-2941.

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research, 12*(7), 257-269.

Fike, J., Jongsma, S., Alonso, J., & Weide, E. (2011). Optimization with Gradient and Hessian Information Calculated Using Hyper-Dual Numbers. *29th AIAA Applied Aerodynamics Conference* (pp. 1-19). Honolulu, Hawaii: American Institute of Aeronautics and Astronautics.

Ghadimi, S., Lan, G., & Zhang, H. (2016). Mini-batch stochastic approximation meth-ods for nonconvex stochastic composite optimization. *Mathematical Programming, 155*(1-2), 267-305.

Gong, Y., Wang, L., Guo, R., & Lazebnik, S. (2014). Multi-scale orderless pooling of deep convolutional activation features. *European Conference on Computer Vision (ECCV)*, 392-407.

Hardt, M., Recht, B., & Singer, Y. (2016). Train faster, generalize better: Stability of stochastic gradient descent. *Proceedings of The 33rd International Conference on Machine Learning*, *48*, pp. 1225-1234. New York, USA.

Hinton, G., Srivastava, N., & Swersky, K. (2012). Neural networks for machine learning. *Lecture 6a: Overview of mini-batch gradient descent*.

Hu, J., & Zheng, W. (2019). An Adaptive Optimization Algorithm Based on Hybrid Power and Multidimensional Update Strategy. *IEEE Access, 7*, 19355-19369. doi:10.1109/ACCESS.2019.2897639

Huang, F., & Chen, S. (2019). Mini-Batch Stochastic ADMMs for Nonconvex Nonsmooth Optimization. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, arXiv:1802.03284v3.

Jain, P., & Kar, P. (2017). Non-convex Optimization for Machine Learning. *Foundations and Trend in Machine Learning, 10*(3-4), 142-336. doi:10.1561/2200000058

Jin, C., Ge, R., Netrapalli, P., Kakade, S., & Jordan, M. (2017). How to Escape Saddle Points Efficiently. *Proceedings of the 34th International Conference on MachineLearning (PMLR70).* Sydney, Australia.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. (2017). On large-batch training for deep learning: Generalization gap and sharp minima. *International Conference on Learning Representations (ICLR).* Toulon, France.

Keskar, N., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. (2017). On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. *5th International Conference on Learning Representations, ICLR 2017.* Toulon,France.

Khan, A. (2020, November 1). *Vanishing Gradients*. Retrieved from in2techs.com: https://in2techs.com/vanishing-gradients/

Kingma, D., & Ba, J. (2015). Adam: A method for stochastic optimization. *Proceedimgs of the 3rd Int. Conf. Learning Representations (ICLR 2015)* (p. 13). San Diego, CA: Ithaca, NY: arXiv.org.

Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, (pp. 1097-1105). Lake Tahoe, Nevada.

Lazebnik, S., Schmid, C., & Ponce, J. (2006). Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)* (pp. 2169-2178). New York, NY, USA: IEEE.

Li, X., Zhao, T., Arora, R., Liu, H., & Haupt, J. (2016). Stochastic variance reduced optimization for nonconvex sparse learning. *Proceedings of The 33rd International Conference on Machine Learning*, *48*, pp. 917-925. New York City, NY, USA.

Liang, D., Ma, F., & Li, W. (2020). New Gradient-Weighted Adaptive Gradient Methods With Dynamic Constraints. *IEEE Access, 8*, 110929-110942. doi:10.1109/ACCESS.2020.3002590

Liu, Z., Feng, R., Li, X., Wang, W., & Wu, X. (2021). Gradient-Sensitive Optimization for Convolutional Neural Networks. *Computational Intelligence and Neuroscience,, 2021*, 16.

Luo, L., Xiong, Y., Liu, Y., & Sun, X. (2018). Adaptive gradient methods with dynamic bound of learning rate. *arXiv Preprints*, arXiv:1902.09843.

Marin, I., Skelin, A., & Grujic, T. (2019). Empirical Evaluation of the Effect of Optimization and Regularization Techniques on the Generalization Performance of Deep Convolutional Neural Network. *Applied Sciences, 10*, 30. doi:10.3390/app10217817

Mehdi, E. (2020, July 28). *Non-Convex Optimization in Deep Learning*. Retrieved from medium.com: https://medium.com/swlh/non-convex-optimization-in-deep-learning-26fa30a2b2b3

Mustapha, A., Mohamed, L., & Ali, L. (2021). Comparative study of optimization techniques in deep learning: Application in the ophthalmology field. *Journal of Physics: Conference Series, 1743*, 13. doi:10.1088/1742-6596/1743/1/012002

MyRank. (2018, February 17). *Maxima and Minima*. Retrieved from myrank.co,in: https://blog.myrank.co.in/maxima-and-minima/

Neyshabur, B., Tomioka, R., Salakhutdinov, R., & Srebro, N. (2017). Geometry of Optimization and Implicit Regularization in Deep Learning. *arXiv Preprints*, arXiv:1705.03071 .

Reddi, S., Hefny, A., Sra, S., Póczos, B., & Smola, A. (2016). Stochastic variance reduction for nonconvex optimization. *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016*, (pp. 314-323). New York City, NY, USA.

Reddi, S., Zaheer, M., Sachan, D., Kale, S., & Kumar, S. (2018). Adaptive Methods for Nonconvex Optimization. *32nd Conference on Neural Information Processing Systems (NIPS 2018)*, (pp. 9815-9825). Montréal, Canada.

Shone, N., Ngoc, T., Phai, V., & Shi, Q. (2018, February). A Deep Learning Approach to Network Intrusion Detection. *IEEE Transactions on Emerging Topics in Computational Intelligence, 2*(1), 41-50. doi:10.1109/TETCI.2017.2772792

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv Preprints*, arXiv:1409.1556.

Tan, H. H., & Lim, K. H. (2019). Vanishing Gradient Mitigation with Deep Learning Neural Network Optimization. *7th International Conference on Smart Computing & Communications (ICSCC)* (pp. 1-4). Curtin Malaysia, Miri, Malaysia: IEEE Xplore. doi:10.1109/ICSCC.2019.8843652

Tao, Z., Xia, Q., & Li, Q. (2019, September 25). *A new perspective in understanding of Adam-Type algorithms and beyond.* Retrieved from openreview.net: https://openreview.net/forum?id=SyxM51BYPB

Theodoridis, S. (2020). Chapter 7 - Classification: a Tour of the Classics. In S. Theodoridis (Ed.), *Machine Learning: A Bayesian and Optimization Perspective* (2nd ed., pp. 301-350). Academic Press.

Vieira, K., Koch, F. L., Sobral, J. B., Westphall, C. B., & de Souza Leao, J. L. (2019). Autonomic intrusion detection and response using big data. *IEEE Systems Journal, 14*(2), 1984-1991.

Wang, B., Meng, Q., Chen, W., & Liu, T.-Y. (2021). The Implicit Regularization for Adaptive Optimization Algorithms on Homogeneous Neural Networks. *Proceedings of the 37th International Conference on MachineLearning (PMLR 119)* , (p. arXiv:2012.06244v3). Vienna, Austria. Retrieved from https://arxiv.org/pdf/2012.06244.pdf

Wang, C.-F. (2019, January 8). *The Vanishing Gradient Problem*. Retrieved from towardsdatascience.com: https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484

Wang, H., Keskar, N., Xiong, C., & Socher, R. (2018). Identifying Generalization Properties in Neural Networks. *arXiv preprint* , arXiv:1809.07402.

Wang, W., & Srebro, N. (2019). Stochastic Nonconvex Optimization with Large Minibatches. *Proceedings of the 30th International Conference on Algorithmic Learning Theory*, *98*, pp. 1-26. Chicago, Illinois, USA.

Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-mnist: A novel image dataset for benchmarking machine learning. *arXiv Preprints*, arXiv:1708.07747.

Yi, D., Ahn, J., & Ji, S. (2020). An Effective Optimization Method for Machine Learning Based on ADAM. *Applied Sciences, 10*(1073), 20. doi:10.3390/app10031073

Zadeh, R. (2016, November 16). *The hard thing about deep learning*. Retrieved from https://www.oreilly.com: https://www.oreilly.com/radar/the-hard-thing-about-deep-learning/

Zeiler, M. D. (2012). ADADELTA: An adaptive learning rate method. *arXiv Preprints*, arXiv:1212.5701. Retrieved from https://arxiv.org/abs/1212.5701

Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2017). Understanding deep learning requires rethinking generalization. *International Conference on Learning Representations (ICLR 2017).* Toulon, France.

Zhou, Y., Karimi, B., Yu, J., Xu, Z., & Li, P. (2020). Towards Better Generalization of Adaptive Gradient Methods. *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, (p. 12). Vancouver, Canada.