https://doi.org/10.26682/sjuod.2019.22.2.10

A NEW IMAGE CLASSIFICATION SYSTEM USING DEEP CONVOLUTION NEURAL NETWORK AND MODIFIED AMSGRAD OPTIMIZER

ARMAN I. MOHAMMED* and AHMED AK. TAHIR**

- Dept.Information Technology, Presidency, Duhok Polytechnic University, Kurdistan Region, Iraq

** Dept. Of Computer Science, College of Science, University of Duhok, Kurdistan Region, Iraq

(Received: June 11, 2019; Accepted for Publication: September 25, 2019)

ABSTRACT: A new deep Convolutional Neural Network (CNN) with six convolutional layers and one fully-connected layer is developed and trained by backpropagation using a new optimization algorithm called Fast-AMSgrad which is modified from AMSgrad. The aims are to speed up the training process while achieving acceptable accuracy. The application of the network using both, the Fast-AMSgrad and the AMSgrad algorithms to CIFAR-10 dataset for image classification reveals that the developed CNN performs better when trained with Fast-AMSgrad for both cases, with and without Batch Normalization (BN) layers. The training time is reduced by 50% when Fast-AMSgrad algorithm is used. Also the accuracy and loss values of the training and validation are improved when Fast-AMSgrad is used. The training and validation accuracies provided by Fast-AMSgrad with BN are (91.18% and 86.92%) at epoch number (50) and (*94.13*% and 86.758%) at epoch number (100), while the corresponding accuracies that are provided by AMSgrad with BN are (82.65% and 81.4%) at epoch (50) and (88.82% and 85.85%) at epoch (100). The overall test accuracy and classification metric measures indicate that the given architecture of CNN and optimization algorithm perform reasonably well.

KEYWORDS: Adam, AMSgrad, CNN, Deep neural networks, Image classification, Optimization algorithms.

1. INTRODUCTION

Image classification is the process of extracting features upon which the object(s) are classified in an image. These features could be edges, lines, ridges, any localized point of interest, or it could be texture or structure of any shape information that describe objects. The objects could be human face, car, buildings, train, airplane, face, defected tissues, etc.

Image classification has become an important tool in many applications that are based on computer vision and artificial intelligence such as medical imaging, security and authentication, military surveillance, office automation, etc.

The previous approach of image classification, the hand-grafted features utilizes methods and algorithms that aim the extraction of features or attributes that are effective for object discrimination in an image, [Wu et al, 2014]. Typical methods are the Scale Invariance Feature Transform (SIFT) which was developed by [Lowe, 1999] for object recognition and Histogram of Oriented Gradient (HOG) which was developed by [Dalal and Triggs, 2005] for object detection. However, the approach of hand-crafted features suffers some shortcomings especially when dealing with large scale images it needs the incorporation of more discriminate features and expert knowledge or ancillary data such as texture, geometry, etc. Whereas incorporating many features and data types will complicate the task of designing the feature extraction methods and may lead to tremendous computational complexity.

In recent years, the approach of deep convolutional neural networks (CNN) has opened up prospects for superior image classification and detection outperforming traditional methods [Krizhevsky et al, 2012: Zhang et al, 2017: Hoseini et al, 2018]. Deep CNN could learn rich highly abstract image features to represent complex objects effectively.

armanamedi@gmail.com

CNN can directly learn data representations from the samples of training dataset and detect data-driven features for specific tasks. In contrast to hand-crafted features, deep learning can extract and organize the discriminative information from large scale images and can be faster. Many structures of deep CNN have already been developed. Examples of the well known deep CNNs are AlexNet by [Krizhevsky et al, 2012], ZFNet by [Zeiler and Fergus, 2014] and VGG-19 by [Simonyan and Zisserman, 2015]. In addition, many optimization algorithms for CNN have been developed with the aim to improve the performance of CNN networks, [Kingma and Ba, 2015: Reddi et al, 2018: Ma and Yarats, 2019]. However, developing a CNN that performs well for image classification will remain as one of the most challenging issue and require a good coordination between the CNN architecture, optimization algorithm, training parameters, training data etc. Good performance can be achieved when the combination of these criteria provides good training and validation accuracy with least overfitting.

The objective of this paper is introducing a CNN with efficient architecture and optimization algorithm for image classification using CIFAR-10 dataset. The main goals are to reduce the training time and at the same time to achieve best validation and testing accuracy.

The remainder of this paper is organized as follows. In Section 2, a brief review of the related work is given. In Section 3, the proposed CNN architecture and the Fast-AMSgrad optimization algorithm are presented. In section 4, the results of applying the proposed CNN to CIFAR-10 dataset are presented and discussed. In section 5, the conclusions are given.

2. RELATED WORK

Much of research works on convolutional neural network (CNN) has been done to improve the CNN performance for various types of applications including image classification. Many of deep learning networks of different architecture have already been designed and many optimization algorithms for updating the weights during the training of these networks have been developed. In 2012, Krizhevsky and others proposed one convolutional neural network for image classification called AlexNet [Krizhevsky et al, 2012]. This deep CNN contained five convolutional layers and three fully-connected layers and trained by backpropagation learning algorithm and SGD with Momentum, [Qian, 1999], as optimizer is used for update rules to minimize the loss function. This CNN is applied to a subset of ImageNet dataset containing 1000 categories and 1000 image for each category and achieved test top-1 error of 37.5% and top-5 error as 17.0%. In 2014, [Zeiler and Fergus, 2014] modified AlexNet to ZFNet by reducing the filter size, reducing the stride from 4 to 2 and increasing the size of activation map. In addition, they used deconvolution for visualizing the learnt features in order to illustrate how CNNs should be developed for image classification. The ZFNet was trained on a single GTX580 GPU and achieved top-5 error rate 14.8%. The field of CNNs continued to progress and very deep CNNs have also been developed to be used for large scale images. An example of very deep CNN is the VGG network, which has been developed by Visual Geometry Group (VGG) at Oxford University. The last version of this network VGG-19 was much deeper than previous ones, [Simonyan and Zisserman, 2015]. The VGG-19 contained 16 convolution layers, 3 fully-connected layers, five max pooling layers, ReLu activation and Batch Normalization layer. The network was trained using backpropagation with SGD optimization algorithm, [Qian, 1999]. The application of VGG-19 to ImageNet dataset with 1000 categories achieved top-5 error of 8%. Moreover, deeper CNN which are called very deep CNN have been developed to be used with larger scale datasets. For instance, a team from Google Inc., designed another CNN known as Inception v1 also called GoogleNet [Szegedy et al, 2015: Szegedy et al, 2016] The Inception v1 CNN consisted of 22 layers of inception modules in total and a ReLU activation function is used with each convolution operation. The dropout rate of 0.7 and SDG optimization method for updates were used. Another very deep CNN was developed by the Microsoft research team called Residual Network or ResNet [He et al, 2016]. The ResNet-152 consists of 152 convolution layers accompanied by ReLU activation function. The main idea of ResNet is to add the input to the output after few convolutional layers, this scheme is call skip or shortcut connections which solves the problem of gradient vanishing or exploding. The bottle neck design is also used within this network that was suggested in Network in Network or (NiN) [Lin et al, 2013] and GoogleNet by using 1x1 convolution layers before and after each convolution layer which are used for dimensionality reduction. The ResNet was applied to ImageNet dataset and achieved top-5 error rate of 3.57% and was the first CNN to beat the Human error rate which is 5% [Russakovsky et al, 2015].

Other networks such as Aggregated Residual Transformations for Deep Neural Networks (Xie et al, 2017) which was designed by UC San Diego and Facebook AI research and the Squeeze-and-Excitation (SENet) networks was introduced by [Hu et al, 2018] to work with very large scale datasets. Both of these networks succeeded to achieve top-5 error of less than 3%.

In order to increase the performance of CNN, special attention was given to the methods of optimization. The main purpose of optimization algorithms is to find the optimal minima of the gradient which indicates the process of learning in neural networks. More specifically, it is the process of finding or extracting features form training data and it has no tasks during the testing phase. Many algorithms of optimization have been developed. The most common of these algorithms are Gradient-Based learning, Newton's method, Stochastic Gradient Descent with Momentum (SGD with Momentum), Resilient Propagation (RPROP), Adaptive Subgradient Method (AdaGrad), Adadelta, The Root Mean Square Propagation Optimization (RMSProp), The Adaptive Moment Estimation (Adam), Adaptive Method Setup Gradient (AMSgrad), Adam with decoupled weight decay (AdamW) and AdaptAhead, [Riedmiller and Braun, 1993: Qian, 1999: Duchi, 2011: Zeiler, 2012: Tieleman and Hinton, 2012: Kingma, 2015: Reddi, 2018: Loshchilov and Hutter, 2019: and Hoseini et al, 2019]. In addition, the problem of overfitting and underfitting that may occur during the training mode and decreases the performance of CNN was investigated, [Hinton et al, 2012: Srivastava et al, 2014: Wu et al 2015]. This problem appears when a difference between the training accuracy and the validation accuracy occurs. This problem was solved to a good extend by dropout technique (dropout some of the convolution parameters). Also the research works included other specific problems in the deep CNN such as selecting and regulating the receptive field automatically, [Wei et al 2018]. The progress in the field of CNN covered also the preparation of many datasets of various volume and characteristics for CNN training. Examples are CIFAR-10, CIFAR-100 and ImageNet [Krizhevsky, 2009: Deng et al, 2009].

3. THE PROPOSED CNN ARCHITECTURE

Generally speaking, there are several criteria that must be considered during the design of the CNN as these criteria will have direct effect on the computational time of training process classification results. These criteria are the size of the dataset used for CNN training, the resolution of the dataset images and the number of classes. For instance, using large dataset with high resolution images and large number of classes will require CNN structure with more convolutional layers and this in turns needs more filters and more Batch Normalization and fully-connected layers, [Krizhevsky et al, 2012]. In addition, dropout layers are needed to reduce the effect of overfitting, [Hinton et al 2012].

In this work, taking the aforementioned details in the consideration one reasonable structure is suggested to be designed for image classification using the CIFAR-10 dataset. The suggested structure consists of 6 convolutional layers with 6 BN layers, 6 ReLU layers, 3 dropout layers, 3 pooling layers and one fully-connected layer as shown in figure 1. Using 6 convolutional layers will achieve acceptable accuracy. This CNN consists of (417,734) parameters for learning the features of CIFAR-10 dataset. The dropout rates of all three layers are (0.3, 0.4 and 0.5) respectively. This architecture is implemented in two modes, with BN and without BN for each the Fast-AMSgrad and AMSgrad. All four schemes were trained on Floydhub deep learning server with Xeon 2 Cores CPU, [Floyhub Server , https://www.floyhub.com/jobs, 2018].

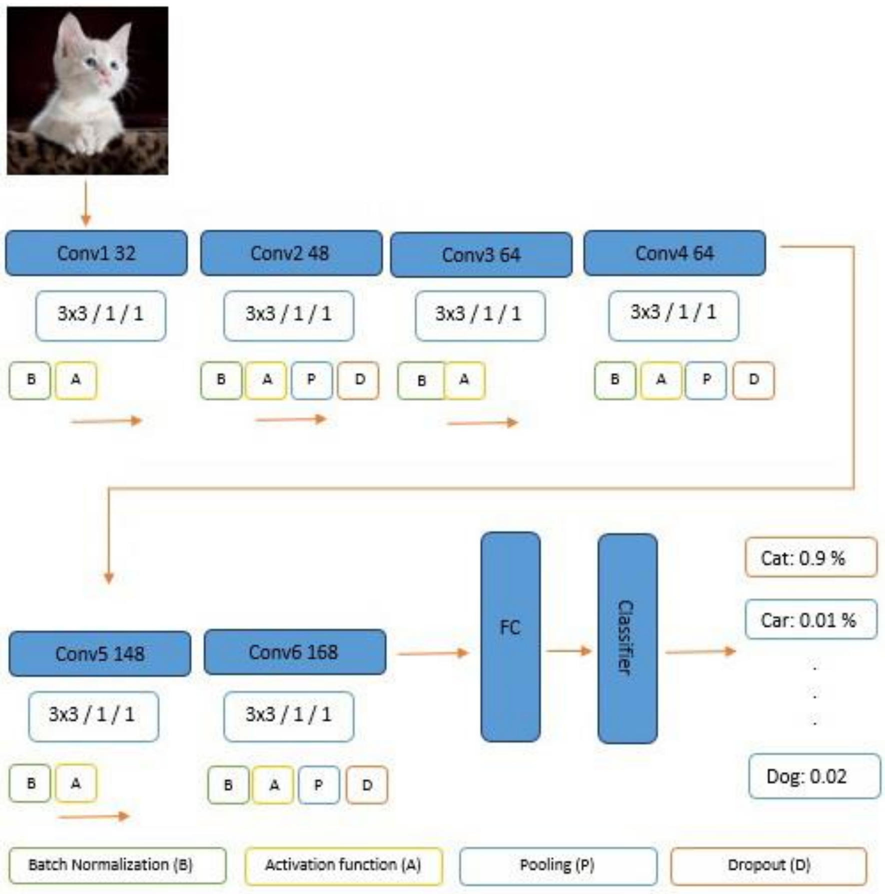

Fig. (1): CNN Architecture

4. THE PROPOSED FAST-AMSGRAD OPTIMIZER

Recently several optimization algorithms that are based on adaptive learning rate have been developed and used widely in training CNN for image classification. Examples of these optimizers are RMSProp, Adam, AMSgrad Adam W and Quasi-Hyperbolic Momentum (QHAdam), [Tieleman T. and Hinton G., 2012: Kingma and Ba, 2015: Reddi et al, 2018: Loshchilov and Hutter, 2019: Ma and Yarats, 2019]. However, it has been noticed by [Huang et al, 2017 and Johnson et al, 2017] that the adaptive learning rate optimizers may fail to convergence and may not able to find the optimal minima, especially for applications such as image classification, object recognition and machine translation. To overcome this problem, [Reddi et al, 2018] modified AMSgrad algorithm from Adam algorithm by changing the rule of computing the second moment and has achieved better performance as claimed by the authors. However, more recent work such as [Korzeniowski, 2018] proved that even after the modification, the performance of AMSgrad did not outrage Adam.

In this paper, a new optimization algorithm called Fast-AMSgrad is modified from AMSgrad. The modification is done by dividing the weight change Δw_t of AMSgrad by the square root of the corrected second moment. This means, instead of using $\sqrt{\hat{v}_t}$ in the denominator of the weight change equation, just $\hat{v}_t$ is used. This is equivalent to raising the power of the denominator to 2. Thus the pseudo code of Fast-AMSgrad becomes as shown in table (1) below:

Table (1): The Pseudo Code of the Developed Optimizer (Fast-AMSgrad) with Criteria Definitions

	Pseudo Code		***Parameter definition***
1	***Input***: *w, ϵ, ŋ, β1, β2*	w	weight
2	***Initialize***: $m_t = 0, v_t = 0, t = 0, \hat{v}_t = 0$	ŋ	Learning rate
3	$\boldsymbol{While}$ w not $converged$ $\boldsymbol{do}$	ϵ	Small value = 10^{-8}
4	$t = t + 1$	t	Iteration number
5	$g_t = \nabla f_t(w_t)$	g_t	Gradient at iteration t
6	$m_t = \beta_1 . m_{t-1} + (1-\beta_1) . g_t$	∇f_t	Computational gradient function
7	$v_t = \beta_2 . v_{t-1} + (1-\beta_2) . g_t^2$	m_t	First moment
8	$\hat{v}_t = max(\hat{v}_{t-1}, v_t)$	β1	Hyperparameter (Decay rate=0.9)
9	$\Delta w_t = -\frac{ŋ}{\hat{v}_t} . m_t$, $when$ $\hat{v}_t > 0$	β2	Hyperparameter (Decay rate=0.99)
		v_t	Second moment
10	$w_t = w_{t-1} + \Delta w_t$	$\hat{v}_t$	Bias correction of second moment
		Δw_t	Weight change
11	$\boldsymbol{End\ while}$		

It can be shown that the power of the denominator in the equation of calculating Δw_t controls the speed of the learning throughout the stages of the training. In doing so, the equations of weight change Δw_t in both Fast-AMSgrad and AMSgrad are reformulated to modulation functions without losing their mathematical meaning as shown below:

The weight change in AMSgrad according to [Reddi et al, 2018] is given in equation (1) below:

$$\Delta w_t = -\frac{ŋ}{\sqrt{\hat{v}_t}} . m_t \qquad (1)$$

The weight change in Fast-AMSgrad is chosen as in equation (2) below:

$$\Delta w_t = -\frac{ŋ}{\hat{v}_t} . m_t \qquad (2)$$

Where, Δw_t is the update in weight, ŋ is the learning rate, m_t is the first moment, $\hat{v}_t$ is the second moment.

Equations (1 and 2) can be re-written as follow:

For AMSgrad $\Delta w_t = -ŋ . M_1 . m_t$ (3)

For Fast-AMSgrad $\Delta w_t = -ŋ . M'_1 . m_t$ (4)

Where, $(M_1 \ and \ M'_1)$ are the first modulation functions in the two optimizers such that:

For AMSgrad $M_1 = \frac{1}{\sqrt{\hat{v}_t}}$ (5)

For Fast-AMSgrad $M'_1 = \frac{1}{\hat{v}_t}$ (6)

The difference between these two functions is shown graphically in figure 2.

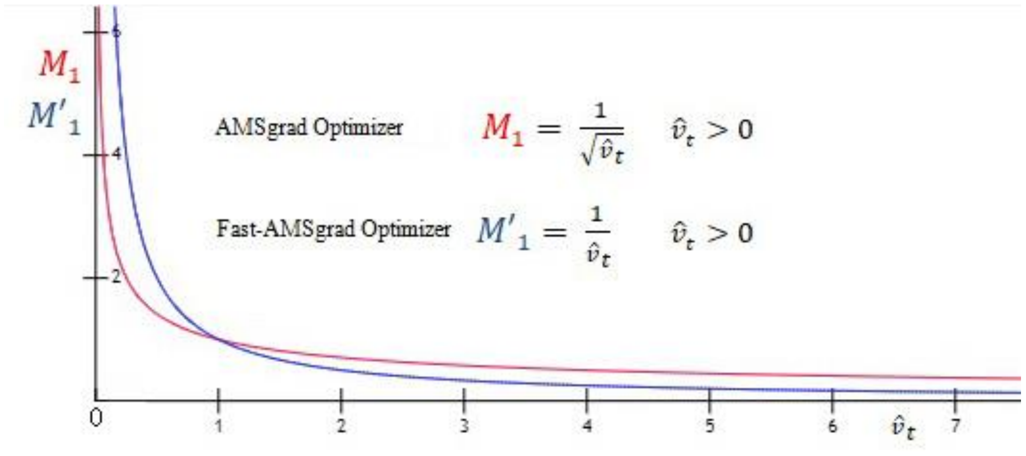

Fig. (2): The Modulation Functions for AMSgrad and Fast-AMSgrad

In this figure the red color represents the modulation function of AMSgrad, while the blue color represents the modulation function of Fast-AMSgrad.

Comparison of these two curves reveals the followings:

1- At $\hat{v}_t > 1$, looking at figure 2 it can be observed that at $\hat{v}_t > 1$ the modulation function of Fast-AMSgrad has lower value than that of the AMSgrad modulation function. Thus, it can be said that the changes on Δw_t that are made by the modulation function of Fast-AMSgrad are less than that made by the modulation function of AMSgrad. Keeping in mind that increasing Δw_t by large amount may lead to overpass the global minima, then it can be said that with Fast-AMSgrad optimizer there will be less chance to overpass the global minima than with AMSgrad. In fact the case when $\hat{v}_t > 1$ is expected to be the dominant case in AMSgrad and Fast-AMSgrad since at each iteration the maximum value of $\hat{v}_t$ from the current and the previous iteration is chosen. That is the value of $\hat{v}_t$ is increasing continuously from the first iteration to the last iteration.

2- At $\hat{v}_t = 1$, both functions have the same value. The point at $\hat{v}_t = 1$ represents the turning point, at which the value of the modulation function of Fast-AMSgrad equals the value of the modulation function of AMSgrad. Inspecting figure 2, it can be realized that at $\hat{v}_t = 1$ the curve of the modulation function of Fast-AMSgrad in blue color looks as a clockwise-rotation of the modulation function of AMSgrad in red color.

3- At $0 < \hat{v}_t < 1$ the situation is reversed, the values of Fast-AMSgrad modulation function (blue color) is higher than that of AMSgrad. This means, at $0 < \hat{v}_t < 1$ the change in Δw_t that is made by Fast-AMSgrad modulation function is higher than that made by the modulation function of AMSgrad. Keeping in mind that in both optimizers the value of $\hat{v}_t$ always increases with increasing the iteration, this means that the step size of the modulation function of Fast-AMSgrad is higher than that of the modulation function of AMSgrad. This in turns will allow Fast-AMSgrad modulation functions to help the process of training to continue without sticking in local minima.

4- In term of speed, since the step size of the modulation function of Fast-AMSgrad is higher than that of the modulation function of AMSgrad at $0 < \hat{v}_t < 1$, then the difference between the values of Δw_t from the current iteration to the previous iteration will be larger. This will speed up the process of training when $0 < \hat{v}_t < 1$. When $\hat{v}_t > 1$ the step size of the modulation functions for both optimizers from iteration to iteration will be almost the same but the values of Δw_t that are made by the Fast-AMSgrad optimizer are smaller than those made by the MASgrad optimizer. Thus, there will be more chance for the Fast-AMSgrad to find the local minima in less number of iterations. While in the AMSgrad there will be more chances that the global minima may be bypassed or found after quiet large number of iteration.

5. RESULTS AND DISCUSSIONS

In order to reveal the performance of the Fast-AMSgrad optimizer and the developed CNN and to show the effect of Batch Normalization (BN), four models of classification are implemented using CIFAR-10 dataset. These are Fast-AMSgrad with and without BN and AMSgrad with and without BN. The performance measures including the training, validation and their Loss values versus epoch numbers of the four models are given in table (2). The results are discussed in details in the following subsections. Also the testing accuracy and the evaluation metrics are calculated for each model.

armanamedi@gmail.com

Table (2): Performance Measures of the Proposed CNN with Different Optimizers

Method	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss	Time	Epoch No.
AMSgrad with BN	0.7633	0.6751	0.7857	0.6187	6:5:21	25
	0.8265	0.4952	0.8140	0.5528	12:10:42	50
	0.8882	0.3193	0.8585	0.4354	24:21:12	100
Fast AMSgrad with BN	0.8690	0.3844	0.8374	0.5046	6:8:8	25
	0.9118	0.2644	0.8692	0.4500	12:16:17	50
	0.9413	0.1937	0.8675	0.4974	24:32:33	100
AMSgrad without BN	0.6969	0.8685	0.7280	0.7909	3:4:7	25
	0.7851	0.6209	0.7889	0.6126	6:8:13	50
	0.8524	0.4278	0.8354	0.4997	12:16:25	100
Fast AMSgrad without BN	0.8463	0.4493	0.8289	0.5369	3:5:5	25
	0.8842	0.3443	0.8421	0.5380	6:10:9	50
	0.9074	0.2901	0.8568	0.5514	12:19:30	100

5.1 Training Accuracy

In the evaluation of training mode three criteria are considered instantly, accuracy, loss value and computation time. As mentioned previously, CIFAR-10 dataset is used. This dataset consists of 50 000 images for training and 10000 for validation and test. The image resolution is 32 by 32 pixels. The evaluation metrics training accuracy of the four models of classification versus epoch number is given in figure 3. According to this figure, the following results can be pointed out:

1- A comparison between Fast-AMSgrad without BN (Green color) and AMSgrad without BN (the violet color) shows that the accuracy of Fast-AMSgrad without BN is higher than that of AMSgrad without BN. Figure 3 also shows that the improvement of the accuracy from epoch to epoch in the case of Fast-AMSgrad is higher than that of AMSgrad without BN. For instance Fast-AMSgrad accuracy at the end of epoch (1) is 31.11% while for the AMSgrad without BN is 12.58%. This indicates that the Fast-AMSgrad can reach to the convergence state in less number of epochs (less time) than AMSgrad. Table 2 shows the training accuracy and loss, validation accuracy and loss, the time at epoch (25), (50) and (100). According to this table the training accuracy of Fast-AMSgrad at epoch number (50) is 88.42% while the training accuracy of AMSgrad is 78.51% and becomes 85.24% at epoch number (100). This means that by Fast-AMSgrad without BN the training time is reduced to half of that of the AMSgrad without BN and at the same time the achieved accuracy by Fast-AMSgrad is higher.

2- A comparison between Fast-AMSgrad with BN (Orange color) and AMSgrad with BN (Blue color) in figure 3, shows that the training accuracy of Fast-AMSgrad with BN is higher than that of AMSgrad with BN. Figure 3 also shows that the improvement of the accuracy from epoch to epoch in the case of Fast-AMSgrad with BN is higher than that of AMSgrad with BN. For instance Fast-AMSgrad accuracy at the end of epoch (1) is 39.97% while for the AMSgrad with BN is 26.81%. This indicates that the Fast-AMSgrad can reach to the convergence state in less number of epochs (less time) than AMSgrad. Table 2 shows the training accuracy and loss, validation accuracy and loss, the time at epoch (25), (50) and (100). According to this table the training accuracy of Fast-AMSgrad at epoch number (50) is 91.18% while the training accuracy of AMSgrad is 82.65% and at epoch number (100) the training accuracy by the Fast-AMSgrad with BN becomes 94.13% and for AMSgrad with BN becomes 88.82%. This means that by Fast-AMSgrad with BN the training time is reduced to half of that of the AMSgrad with BN and at the same time the achieved accuracy by Fast-AMSgrad is higher.

3- A comparison between Fast-AMSgrad without BN and AMSgrad with BN (the orange and blue colors) shows that even Fast-AMSgrad without BN performs better than AMSgrad with BN. That means, much of computation time can be saved by Fast-AMSgrad while the accuracy is remained better than of AMSgrad. The computation time for the Fast-AMSgrad without BN is (12 h and 19 min) while AMSgrad with BN is 24 h 21 min.

4- In general, the improvement in the accuracy of all the schemes of classification is high at the beginning epochs and this improvement becomes lower at the ending epochs.

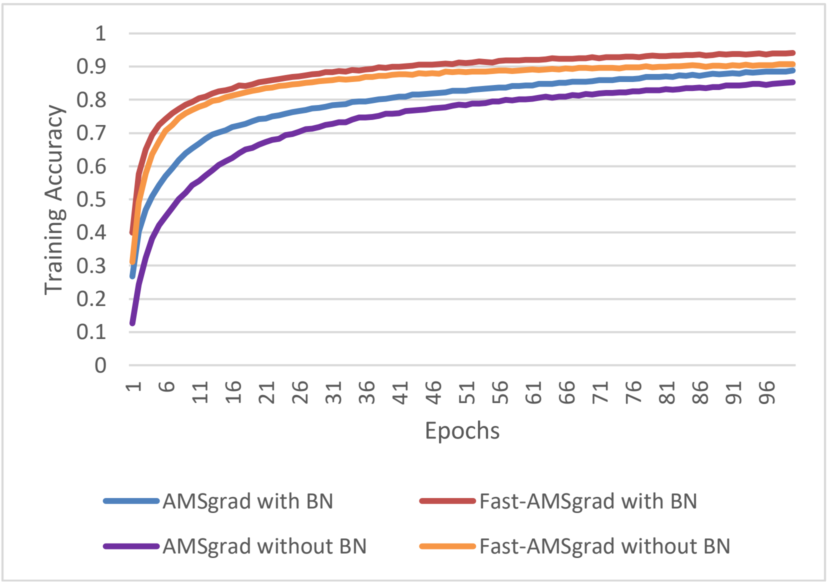

Fig. (3). The Training Accuracy for the Four Models of Classification for 100 Epochs.

5.2 Training Loss

Loss value is related inversely to the level of confidence of classification. For high level of confidence loss value must be small. The value of loss is measured from the average of the losses of the training samples within mini-batch in the current iteration. In this work, the size of mini-batch is 64 samples. However, the loss is an important metric that shows how good the model is. The main goal is to reduce the loss as much as possible. Figure 4 shows that the smallest loss value is found for Fast-AMSgrad with BN (0.1937) at epoch number (100) and the next smallest value is for the Fast-AMSgrad without BN (0.2901) at epoch number (100). While the loss values for the AMSgrad without and with BN are (0.4278 and 0.2901) respectively. This leads to the conclusion that Fast-AMSgrad is performing better than AMSgrad for both cases with and without BN.

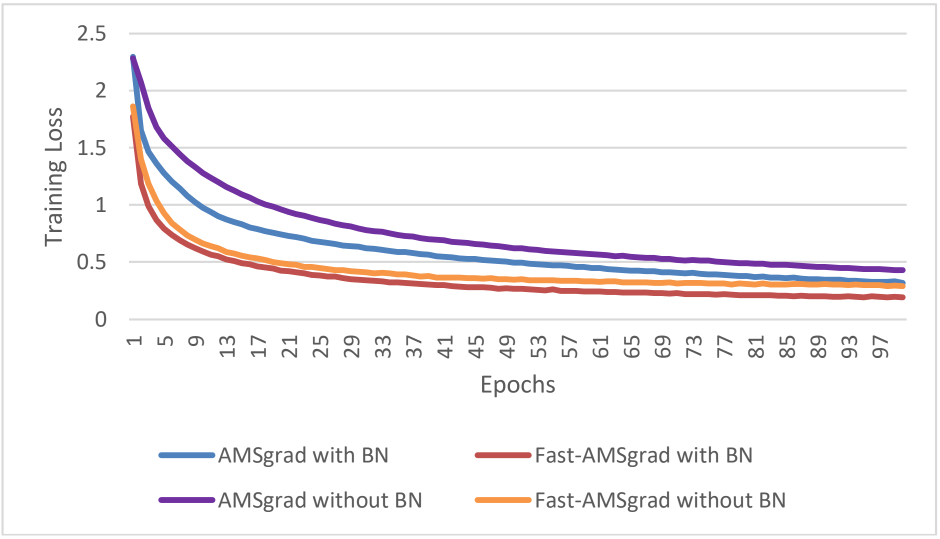

Fig. (4): The Loss Values for the Training of the Four models of Classification for 100 Epochs.

5.3 Validation Accuracy

Validating the classification system represents one good indicator for evaluating the level of success of the network performance. For instance, validation accuracy can be used to detect the overfitting and underfitting states of the network during the training mode. When the accuracy of validation and training are close and reasonably high, it can be said that the training is acceptable. Vice versa, when there is a quit difference between the accuracy of validation and training, it can be said that the training has either overfitting or underfitting problem. Overfitting is the case when the network performs well for training data and badly for validation and testing. In order to evaluate the validation of the network, the accuracy and loss of the validation are measured for the four schemes of classifications.

The validation accuracy of the four schemes of classification against epoch number is given in figures 5. According to this figure, the following results can be pointed out:

1. The validation accuracy at epoch number (1) for AMSgrad with BN is (38.27%) and for AMSgrad without BN is (23.25%) the difference of accuracy among these two schemes is (15.02). While, the accuracy of Fast-AMSgrad with and without BN is (54.25%) and (46.25%) respectively, having the difference of (8) which is quite lower than the difference of AMSgrad schemes. This is a good indicator that the BN layer doesn’t have much impact on the new proposed optimizer while its effect on AMSgrad is very high.
2. At epoch number (50) the accuracy of Fast-AMSgrad with BN is (0.8692) which is the convergence state of this network and the training process can be terminated for this optimizer. While the accuracy of AMSgrad with BN at epoch number (50) is (81.4%) and at epoch number (100) is (85.85%). This assures the good performance (accuracy and speed) of the Fast-AMSgrad with BN over that of AMSgrad with BN and much faster to find the global minima.
3. The accuracy of Fast-AMSgrad without BN at epoch number (100) is (85.68%) that is very close to accuracy of AMSgrad with BN accuracy which is (85.85%). While the accuracy of AMSgrad without BN is (83.54) at epoch number (100).
4. It can be clearly seen from figure 5, that the new proposed optimizer provides smooth learning with less oscillation.

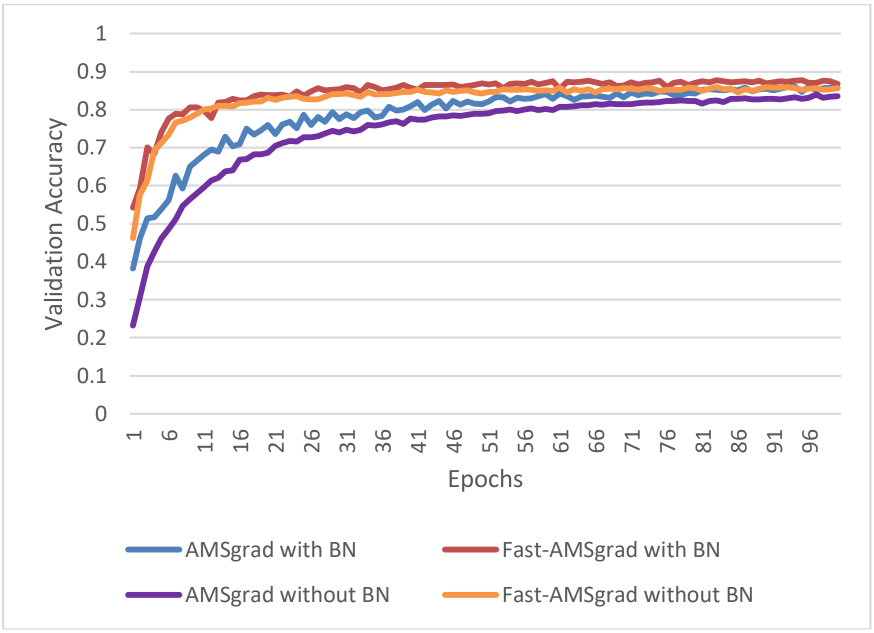

Fig. (5): The Validation Accuracy for the Four Models of Classification for 100 Epochs.

5.4 Validation Loss

The validation loss can also be used as an indicator to detect whether the CNN is overfitting or not. In addition, it shows the level of confidence of the predicted values made by CNN. As shown in figure 6 and in table 2, the validation loss values achieved by all four optimizers are decreasing with the number of epochs.

For Fast-AMSgrad with BN the validation loss values at epoch number (25) is (*0.5046*) and decreases to (0.4974) at epoch number (100).

For AMSgrad with BN the validation loss values at epoch number (25) is (0.6187) and decreases to (0.4354) at epoch number (100).

For Fast-AMSgrad without BN the validation loss values at epoch number (25) is (0.5369) and decreases to (0.5514) at epoch number (100).

For AMSgrad without BN the validation loss values at epoch number (25) is (0.7909) and decreases to (0.4997) at epoch number (100).

Since the loss values are decreasing with epoch number then this indicates that the training of the CNN with four optimizers is acceptable.

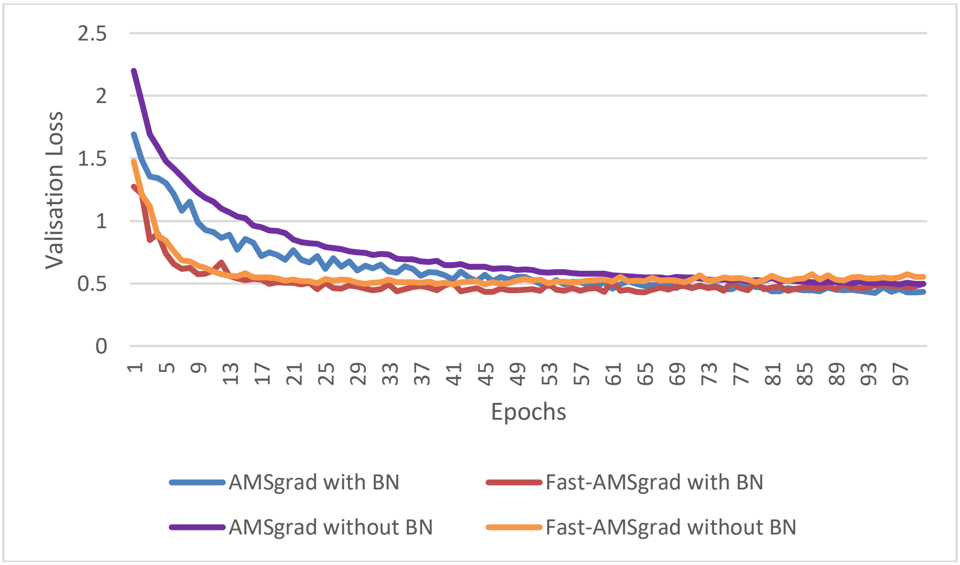

Fig. (6): The Loss Values for Validation for the Four Models of Classification for 100 Epochs.

To illustrate better, the differences between the performances of the four optimizations algorithms at different epoch number, tables 3 is given. It can be seen that the fastest model is Fast-AMSgrad with BN and Fast-AMSgrad without BN comes in second place, while AMSgrad without BN is performing poorly.

Table (3): Accuracy and speed of the CNN with four optimizers at different epoch number

Method	Training Accuracy	Validation Accuracy	Epoch Number	Time / h
AMSgrad with BN	0.8299%	0.8333%	52	12:66
Fast AMSgrad with BN	0.8423%	0.835%	18	4:41
AMSgrad without BN	0.8475%	0.832%	94	11:53
Fast AMSgrad without BN	0.8303%	0.8306%	20	2:46

5
.5 Testing Accuracy

In testing mode, 2000 images are used. The predicted labels with true labels are fed to confusion matrix, and then the classification metrics, classification accuracy, precision, Kapa and error measures are calculated, [Drăgulescu et al, 2015]. Table 4 shows the results. According to this table, Fast-AMSgrad performs better than AMSgrad for both cases, without and with BN. It can be seen that Fast-AMSgrad without BN performs even better than AMSgrad with BN. This result coincides with that achieved for the validation during the training mode. That is, with FAST-AMSgrad the CNN does not need BN layers and much of computation time can be saved for the CNN training while still performing better than AMSgrad with BN. In this table, the classification metrics are also shown. The error values for the Fast-AMSgrad with and without BN are 0.1414 and 0.1554 which are acceptable compared to previous works such as [Zeiler and Fergus, 2014: Krizhevsky et al, 2012]. The kappa coefficient values also indicate the confidentiality of the results.

Table (4): Performance measures for Testing Mode

Method	Overall Accuracy	Precision	Error	Kappa Coefficient
AMSgrad with BN	0.8415	0.8457	0.1585	0.8238
Fast AMSgrad with BN	**0.8585**	**0.8634**	**0.1414**	**0.8427**
AMSgrad without BN	0.818	0.8230	0.1820	0.7977
Fast AMSgrad without BN	**0.8445**	**0.8477**	**0.1554**	**0.8272**

6. CONCLUSIONS

The implementation of the CNN with and without BN has shown that with Fast-AMSgrad algorithm the training time is reduced to half of the time needed by AMSgrad while achieving better accuracy for training, validation and testing. The decrease in the validation loss values with increasing the epoch number indicate that the training was working properly. The division of the update weight by the second order moment has let the Fast-AMSgrad to be less affected by the use of BN since even when used without BN it has performed better than AMSgrad with BN and thus saving a lot of computation time. The kappa coefficient values for all schemes of classification were within the range of perfect agreement which means that the classification performance is good compared to just randomly assigning values. The error percents and precisions for all schemes were acceptable compared to the results of previous works. In particular the error percent provided by Fast-AMSgrad was good. These error percents assure that the architecture of CNN is adaptable for both optimization algorithms. According to the classification metrics of table 3, Fast-AMSgrad with BN was the best and Fast-AMSgrad without BN was the second best.

armanamedi@gmail.com

ACKNOWLEDGEMENTS

➢ This research work was implemented at the Computer Science Department / College of Science / University of Duhok as a part of the Master degree requirements, 2016-2019.

➢ The authors would like to express their sincere gratitude to the University of Duhok and the College of Science for their continuous support to make this work possible.

➢ The first author wants to use this opportunity to express his special thanks to the Duhok Polytechnic University for its invaluable support.

REFERENCES

Dalal N. and Triggs B., 2005, "Histograms of oriented gradients for human detection", in international Conference on computer vision & Pattern Recognition, ,CVPR'05, June, IEEE Computer Society ,Vol. 1, pp. (886-893).

Deng J., Dong W., Socher R., Li L.J., Li K., and Fei-Fei, L., 2009, "ImageNet: A large-scale hierarchical image database", IEEE Conference on Computer Vision and Pattern Recognition. (pp. 248-255).

Drăgulescu B., Bucos M., Vasiu R., 2015, "Predicting Assignment Submissions in a Multi-class Classification Problem", TEM Journal, Vol. 4, No. 3, Pp.(244-254).

Duchi J., Hazan E. and Singer Y., 2011, "Adaptive subgradient methods for online learning and stochastic optimization", Journal of Machine Learning Research, 12(Jul), pp. (2121-2159).

Floyhub Server, "Deep Learning Platform-Cloud GP", https://www.floyhub.com/jobs, Visiting Date, Oct-2018.

He K., Zhang X., Ren S. and Sun J., 2016, "Deep residual learning for image recognition", in Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR). (pp. 770-778).

Hinton G. E., Srivastava N., Krizhevsky A., Sutskever I., SalakhutdinovR.R., 2012, "Improving neural networks by preventing co-adaptation of feature detectors", arXiv:pp. (1207.0580).

Hoseini1 F., Shahbahrami A. and Bayat P., 2018, "An Efficient Implementation of Deep Convolutional Neural Networks for MRI Segmentation", Journal of Digital Imaging, Vol. 31, No. 5, pp. (738-747).

Hoseini1 F., Shahbahrami A. and Bayat P., 2019, "AdaptAhead Optimization Algorithm for Learning Deep CNN Applied to MRI Segmentation", Journal of Digital Imaging, Society of imaging informatics in medicine, Springer, Vol. 32, issue 1, Pp. (105-115).

Huang G., Liu Z., Van Der Maaten L. and Weinberger K.Q., 2017, "Densely connected convolutional networks", in Proceedings of the IEEE conference on computer vision and pattern recognition. (pp. 4700-4708).

Hu J., Shen L. and Sun G., 2018, "Squeeze-and-excitation networks", in Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7132-7141).

Johnson M., Schuster M., Le Q. V., Krikun M., Wu Y., Chen, Z., … Dean, J. (2017). Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. Transactions of the Association for Computational Linguistics, Vol. 5, (pp. 339–351).

Kingma, D. P., and Ba, J. L., 2015, "Adam: A Method for Stochastic Optimization", in Proceedings of the International Conference on Learning Representations (ICLR), pp. (1-15).

Korzeniowski F., 2018, "Experiments with AMSGrad" Retrieved December 24, 2018, from https://fdlm.github.io/post/amsgrad/.

Krizhevsky A., 2009, "Learning Multiple Layers of Features from Tiny Images", Chapter 3, Object Classification Experiments, pp. (32-35).

Krizhevsky A., Sutskever I. and Hinton G.E., 2012, "ImageNet classification with deep convolutional neural networks", Proceedings of the 25th International Conference on neural information processing systems (NIPS), Lake Tahoe, December, pp. (1097-1105).

Lin M., Chen Q. and Yan S., 2013, "Network In Network", arXiv preprint: (pp. 1312.4400).

Lowe, D.G. (1999) Object Recognition from Local Scale-Invariant Features. Proceedings of the 7th IEEE International Conference on Computer Vision, Kerkyra, 20-27 September, pp. (1150-1157).

Loshchilov I. and Hutter F., 2019, “Decoupled Weight Decay Regularization”, Proceedings of the International Conference on Learning Representations (ICLR), pp. (1-8).

Ma J. and Yarats D., 2019, “Quasi-Hyperbolic Momentum And Adam For Deep Learning”, International Conference on Learning Representations (ICLR), pp. (1-38).

Qian, N. 1999, “On the momentum term in gradient descent learning algorithms”, Neural Networks, ELSEVIER, Vol. 12, Issue 1, (pp. 145–151).

Reddi S. J., Kale S. and Kumar S., 2018, “On the Convergence of Adam And Beyond”, Proceedings of the International Conference on Learning Representations (ICLR), pp. (1-23).

Riedmiller M. and Braun H., 1993, “A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm”, in: Ruspini, H., (Ed.) Proc. of the International Congress on Nanoscience and Nanotechnology (ICNN 93), San Francisco, pp. (586-591).

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang Z. · Karpathy A., Khosla A., Bernstein M., Berg A. C. and Fei-Fei, L., 2015, “ImageNet Large Scale Visual Recognition Challenge”, Springer International Journal of Computer Vision, Vol. 115, Issue 3, pp. (211–252).

Simonyan, K. and Zisserman A., 2015, “Very deep convolutional networks for large-scale image recognition”, International Conference on Learning Representations (ICLR), (pp. 1409.1556).

Srivastava N., Hinton G. E., Krizhevsky A., Sutskever I., SalakhutdinovR., 2014, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”, Journal of Machine Learning Research 15, pp. (1929-1958).

zegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V. and Rabinovich, A., 2015, “Going deeper with convolutions”, in Proceedings of the IEEE conference on computer vision and pattern recognition, (pp. 1-9).

Szegedy C., Vanhoucke V., Ioffe S., Shlens J., and Wojna Z., 2016, “Rethinking the Inception Architecture for Computer Vision”, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, Nov. 2016, pp. (2818-2826).

Tieleman T. and Hinton G., 2012, “Lecture 6.5-rmsprop: Divide the Gradient by a Running Average of Its Recent Magnitude”, COURSERA: Neural Networks for Machine Learning, 4, pp. (26-31).

Wei Z., Sun Y., Lin J. and Liu S., 2018, “Learning adaptive receptive fields for deep image parsing networks”, Journal of Computational Visual Media, Vol. 4, No. 3, pp. (231-244).

Wu W., Xia R., Xiang W., Hui B., Chang Z., Liu Y., and Zhang Y., 2014, “Recognition of Airport Runways in FLIR Images Based on Knowledge” , IEEE Geosci. Remote Sens. Lett., vol. 11, no. 9, pp. (1534–1538)

Wu H. and X. Gu X., 2015, “Towards dropout training for convolutional neural networks”, Neural Networks, 71, pp. (1–10).

Xie S., Girshick R., Dollár P., Tu Z. and He K., 2017, “Aggregated residual transformations for deep neural networks”, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (pp. 1492-1500).

Zeiler M. D., 2012, “Adadelta: An Adaptive Learning Rate Method”, arXiv preprint arXiv, pp. (1212-5701).

Zeiler M.D., Fergus R., 2014, “Visualizing and Understanding Convolutional Networks”,in Fleet D., Pajdla T., Schiele B., Tuytelaars T., (eds) Computer Vision – European Conference on

Zhang P., Niu X., Dou Y., and Xia F., 2017, “Airport Detection on Optical Satellite Images Using Deep Convolutional Neural Networks”, IEEE Geoscience and Remote Sensing Letters, Vol. 14, No. 8, pp. (1183–1187).