An Improved Adaptive Optimization Technique for Image Classification

Nazmus Saqib
Electronics & Communication Engineering (ECE)
Khulna University of Engineering & Technology (KUET)
Khulna, Bangladesh
nsaqib1995@gmail.com

Fatema Tuz Zahra
Materials Science & Engineering (MSE)
Khulna University of Engineering & Technology (KUET)
Khulna,Bangladesh
fatematuzzahraa@gmail.com

***Abstract*—In deep learning, the optimization techniques are the most part dependent on gradient descent methods, such as SGD, ADAM; adopt the leading place in the area of optimization methods. Conventional methodologies which depend on stochastic gradients are non-adaptive because the prescribed parameter worth's usage should be tuned for every application. However, the generalization performance of the stochastic optimizers is far superior to the adaptive methods, whereas ADAM and its variants cannot maintain this without a fast convergence rate in deep neural networks. To improve this generalization performance, we need to diminish the oscillation of the weights, the general problem of the accuracy fall. To stable the generalization performance, we have introduced Mean-ADAM, a variance of ADAM, extends the updated weights by an external weight to diminish the oscillation and overcome a superior accuracy rate than all other adaptive gradient methods till the conclusion of the training. Therefore, we substantially improve the generalization performance, permit it to contend with the existing algorithms on different image classification datasets such as MNIST, CIFAR 10, CIFAR 100, ImageNet, etc. We have attained 82% at 150th epochs with CIFAR 10 and 99.49% with MNIST where the ADAM has indicated 76% and 99.43% individually. The graphical statistics and the quantitative results can indicate that the proposed method is more likely to perform in more diverse image datasets.**

***Index Terms*—Convolution neural network, MNIST dataset, Fashion MNIST dataset, CIFAR 10 dataset, CIFAR 100 dataset.**

I. INTRODUCTION

Computer vision is a collaborative field of numerous problems such as recognition of images, localization, segmentation, and object detection. Among these, image classification can be considered as a fundamental problem and is the basis for other computer vision problems. The convolutional neural network has exhibited fantastic accomplishments in issues of computer vision, particularly in image classification, image recognition, object detection, and so on. The design of these issues is generally conventional, non-linear, pooling, and fully connected layer. The traditional connected layers are profoundly gainful for local connections, shared weights, pooling and the utilization of numerous layers. The prevalent methodology in training the neural network advocates the use of the optimization method. The optimization of some scalar parameterized differentiable objective function requires maximization and minimization with respect to its parameters. Gradient descent is the most efficient first-order iterative optimization method to find the minimum of the function. However, when the dataset is numerous in quantity, Gradient Descent follows a very moderate process by updating the whole dataset only once and can not fit it into the memory. Regarding the issue, Stochastic Gradient Descent (SGD) turns out to be increasingly efficient due to update these scalar parameters for each perception, which prompts a prominent number of updates that bring about a quicker methodology than existing [1] .

These continuous parameter updates have high in variance and cause the loss function to vacillate to different intensities, which help to discover new and potentially better local minima. However, it eventually muddles the convergence to the exact minimum. Furthermore, it will continue overshooting due to the frequent fluctuations, shows a similar convergence pattern as gradient descent. Unfortunately, SGD is somewhat productive for enormous datasets. SGD has performed just a single parameter update on a mini-batch of training examples. When the gradient is reliably small and noisy, SGD turns out to be slower, every now, and again follows the inappropriate gradient and shows a similar convergence design as gradient descent.

In neural network training, the Momentum method can accelerate gradient descent by taking records of previous gradients in the update rule at each iteration. When the scholarship pace is relatively huge, Nesterov Accelerated Gradients (NAG) permits a larger decay rate than the Momentum Method, while forestalling oscillations. The Nesterov can figure the gradient, not concern with the present advance, can assume the future step by the previous momentum and then update the weights [14]. This incident encourages NAG to finish the activity faster than the momentum-based GD with fewer oscillations. In any case, for the real-time datasets like MNIST, CIFAR 10, CIFAR 100, and the vast majority of the highlights are inadequate, having zero values, thus the corresponding gradient is zero, and the parameter refreshes are additionally zero. However, for sparse datasets having highlights of different frequencies, a single learning rate for every weight update can have exponential regret.

ADAGRAD is a general algorithm for choosing a learning rate dynamically by adapting to the data and calculates a different learning rate for every feature [4]. However, after

978-1-7281-9331-1/20/$31.00

a couple of updates corresponding to dense features, the learning rate decays rapidly because the algorithm accumulates squared gradients, refuses to learn, and converges slowly for a large number of epochs. AdaGrad brings about decaying and diminishing the learning rate for bias parameters. Lately, Rmsprop conquers the decaying learning rate issue of AdaGrad and forestalls the rapid growth in velocity. This algorithm accumulates the previous gradients in weight, which increases the rapid growth of velocity and attempts to converge [3]. However, Rmsprop and momentum are differentiating in approach. While momentum accelerates the inquiry in the direction of local minima, Rmsprop blocks the pursuit in the direction of oscillations. Considering this issue, AdaDelta, additionally an augmentation of AdaGrad, which attempts to decrease AdaGrad's aggressive, monotonically lessening learning rate by confining the window of the past accumulated gradient to some fixed size of weight [5].

Considering reduction of the profoundly diminishing learning rates of AdaGrad, a major breakthrough occurs after the introduction of Adam as the most popular adaptive gradient method based optimizer. Adam was proposed as a blend of AdaGrad which works admirably on sparse gradients and Rmsprop which functions admirably in online and non-stationary backgrounds. Adam actualizes the exponential moving average of the gradients to scale the learning rate, keeping an exponentially decaying average of past gradients. The popular optimizer is computationally a productive technique for straightforward execution having a little memory prerequisite [1]. Adam is invariant to diagonal rescale of the gradients, appropriate for issues that are enormous in terms of data, proper for non-stationary objectives, fitting for issues with very noisy or sparse gradients. Adam in certain zones does not converge to an optimal solution, so for certain undertakings, for example, Cifar datasets state-of-the-art results are still only reached by applying SGD with momentum. To improve the convergence, Nadam consolidates both classical momentum and NAG, which characterizes momentum as a decaying sum over the previous updates; however, Adam characterizes it as a decaying mean over the previous gradients [6]. Using the previous gradients rather than the previous updates permits the algorithm to keep changing direction in any case, when the learning rate has tempered immediately toward the close of the training, bringing about progressively exact fine-grained convergence.

However, in simple one-dimensional convex settings, Adam can fail to converge where some mini-batches provide quite rarely large gradients, which are quite informative. Their impact ceases to exist rather quickly due to the exponential averaging, in this manner, prompting poor convergence [14]. The amount basically quantifies the adjustment in the reverse of the learning rate of the adaptive method to time. AmsGrad utilizes a littler learning rate in contrast with Adam and consolidates the instinct of gradually decaying the impact of past gradients on the learning rate as long as the quantity is positive semi-unequivocal [7]. AmsGrad provides a hypothetical assurance of convergence yet illustrates its better performance on training data [11]. Nonetheless, the speculation capability of AmsGrad on concealed data is similar to that of Adam while an impressive execution gap despite everything exists among AmsGrad and SGD.

Moreover, similar learning rate decay strategies that function admirably in SGD with momentum to adaptive gradient methods. However, it is tough to apply when adaptive gradient methods typically require a very littler base learning rate that will evaporate not long after several rounds of decay. To determine this little learning rate issue and to permit fast convergence hence; closing the generalization gap, Padam unifies Adam/AmsGrad and SGD with momentum by a partially adaptive parameter [8]. By controlling the degree of adaptiveness, the learning rate in Padam need not be as little as other adaptive methods to forestall the gradient update from denoting. This purposes a small learning rate quandary. Initially, the accuracy rate of Padam is extremely high. However, with the increase of the epochs, the accuracy rate becomes compatible with Adam.

Moreover, L2 regularization and weight decay regularization are equivalent in SGD rather than Adam. Adam and other adaptive gradient methods have just actualized the L2 regularization, not the original weight decay and that's the reason Adam prompts more terrible outcomes than SGD with momentum on numerous popular image classification datasets. Regarding this issue, AdamW is proposed to improve regularization in Adam by decoupling the weight decay from the gradient-based update [9]. It has demonstrated that the generalization performance of Adam with decoupled weight decay is improved for various datasets yet for extreme learning rate the accuracy begins to fall like the other adaptive gradient methods.

Disregarding this negative impact of extreme learning rate, new variants of Adam and AmsGrad; AdaBound and AmsBound respectively have been proposed. These very recent optimizers can utilize dynamic bounds on learning rates in adaptive optimization algorithms, where the lower and upper bounds are introduced as zero and boundlessness separately, and both smoothly converge to a constant final step size [13]. In this framework, we can appreciate a rapid initial training process just as great final generalization ability. The authors ensured great final generalization ability but the accuracy rate in this method gradually falls after 300 epochs.

Considering the overall performances of the existing methods, we have proposed an improved philosophy based on the Adam optimization algorithm. The overall contribution represents the bulk of the study, can be summarized as follows:

a) The proposed algorithm improves the fluctuation of the updated weights which can increase the accuracy rate during the training on complex datasets.

b) The study introduces a small constant value in the algorithm section as an external weight within the positive and negative infinity which can stretch the updated weights towards the boundary with an enormous number.

c) A straightforward averaging operation of the updated weights will diminish the general oscillation of the weights

of the optimizers of any adaptive gradient methods. ADAM and all other gradient methods cannot maintain the consistency of fast convergence.

d) The proposed optimizer has considered as proficient evidence with the consistency of high convergence by lessening the loss and parallelly charge out of the most noteworthy accuracy rate till the end of the training.

II. PROPOSED METHODOLOGY

In this segment, we will portray the subtleties of our proposed study. Let, $f(\theta)$ be a noisy objective function: a stochastic scalar function that is differential with respect to parameters θ. The stochastic scalar function follows the range of "$f_1(\theta), f_2(\theta), f_3(\theta), \ldots\ldots f_t(\bigtriangledown\theta f_\theta(\theta))$", we denote the gradient, the vector of partial derivatives of f_t, with respect to θ evaluated at timestep t.

The algorithm with the updates of exponential moving averages of the gradient m_t and the squared gradient v_t where the hyper-parameters $\beta_1, \beta_2 \subset [0, 1)$ are controlling the exponential decay rates of the moving averages.

Algorithm 1 Mean-ADAM, our proposed algorithm for stochastic optimization. g_t^2 indicates the element-wise square $g_t * g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_2 and β_2 to the power t.

Require: α : Step size.
Require: $\beta_1, \beta_2 \subset [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ.
Require: θ_0: Initial parameter vector
$m_0 = 0$ (Initialize 1st moment vector)
$v_0 = 0$ (Initialize 2nd moment vector)
$t = 0$ (Initialize time step)
while θ_t is converged do **do**
    $t = t + 1$
    $g_t = \bigtriangledown_\theta f_t(\theta_{(t-1)})$ (Get gradients w.r.t stochastic objective at timestep t)
    $m_t = \beta_1.m_{(t-1)} + (1 - \beta_1).g_t$ (Update biased first moment estimate)
    $v_t = \beta_2.v_{(t-1)} + (1-\beta_2).g_t^2$ (Update biased second raw moment estimate)
    $s = \alpha.m_t/(v_t + \epsilon)$
    $\delta = 4 \times 10^{-8}$ ($\delta \geq 0$: Orientation of the angle of the gradient vector)
    $\theta_{t(max)} = max(\theta_{(t-1)} - s, [(\theta_{(t-1)} - s) + \delta])$ (The maximum weight)
    $\theta_{t(min)} = min(\theta_{(t-1)} - s, [(\theta_{(t-1)} - s) - \delta])$(The minimum weight)
    $\theta_{t(Avg)} = (\theta_{max} + \theta_{min})/2$ (The update attainment of the maximum and minimum parameter)
end while
return θ_t (Resulting parameters)

Due to this updating method, the variance of the weights might increase in the middle of the optimization procedure. Accordingly, ADAM can experience the ill effects of a severe generalization problem. This sudden increment can present a horrible performance based upon the multifaceted nature of the dataset. Complex datasets like ImageNet or CIFAR 100 are way complex for any regular neural network. Therefore, the presence of abrupt variance uprising in training is not alluring.

As for every optimizer, the majority of learning rates fail to train model, there is a valley shape for each optimizer. We have fixed a learning rate as 0.0001 which becomes too low, has no progresses and fails to reach the ultimate destination. On the other hand, an extreme learning rate causes instability, which never converges. Between the low and high learning rates, there is a band of "just right" learning rates that successfully trained. As a result, there occurs an unknown destination which decreases the accuracy level.

A. Significance of δ

Any real number is within the bound of positive infinity and the negative infinity. We can utilize this equivalent thought for the weights generated by the ADAM optimizer. We have stretched the updated weights towards positive and negative interminability. Pushing them towards the boundary with a large number can introduce a significant drop in the performance. Consequently, we have taken a small constant to perform weight stretching. Adding and subtracting the small constant with any single weight can push the weight towards the upper limit and lower limit, compared to the original weight. From the point onward, we have extracted the real maxima and real minima from the weights using the simple min-max operation. At that point, a simple average operation between the minimum weight and the maximum weight produces the final weight. There are some special significance of the constant number δ, a very small in number which can differ the angle of every gradient momentum and $\delta \geq 0$ makes the orientation of the angle of the gradient vector. As the maximum weight update becomes greater than the earlier one,

$$[(\theta_{(t-1)} - s) + \delta] > (\theta_{(t-1)} - s) \tag{1}$$

As the minimum weight update becomes smaller than the earlier one,

$$[(\theta_{(t-1)} - s) - \delta] < (\theta_{(t-1)} - s) \tag{2}$$

According to the gradient descent method, equations 1 and 2 are continually incompatible.There may be no hazard for any weight update of being always positive or negative. Any updated weight after an epoch can be greater or smaller than the earlier ones. For convex optimization, the weight update gradually decreases than the earlier ones with the passage of the epochs. As every real and synthetic dataset always creates a non-convex optimization scenario, the weight can be increased or decreased. We have already agreed that, $[(\theta_{(t-1)-s}) + \delta]$ and $[(\theta_{(t-1)-s}) - \delta]$ are always highest and lowest respectively. While the incident considers towards the convergence;

according to convergence rule, the distance between $\theta_{(t-1)}$ and s is gradually decreasing a lot which neglects weight update significantly in the end of the training. From this intuition, we are using the constant value δ . If we continue doping the value of δ, weight update occurs and results in an increment of the accuracy.

The value of δ is important because if the weight update fails to arise in any step, then it takes place and handiest depending on the value of δ Which enables the propose method in faster convergence. Moreover, if we suppose like this once more, that will be, $(s = \delta)$ and then the update attainment will be as follows:

$$\theta_{t(Avg)} = (\theta_{max} + \theta_{min})/2 = (2\theta_{[(t-1)-s]} - \delta)/2 \qquad (3)$$

This updated attainment approaches our equation to perform in a considerable position. Finally, to summarize, this algorithm is operating under a convex setting. Relatively, the proposed optimizer is optimizing the problem within the convex field. The upper bound and lower bound of the optimization setting can be derived from the original ADAM paper. Due to the averaging procedure, the upper bound becomes reasonably smaller than the original ADAM upper bound and the lower bound becomes greater than the original ADAM lower bound. Unfortunately, we can not ensure any theoretical guarantee for the proposed study. However, We have presented empirical evidence of its convergence capability using MNIST, Fashion MNIST, CIFAR 10, and CIFAR 100 datasets.

III. EXPERIMENTS

To exactly assess the proposed methodology, it is contrasted and other proposed baselines. We examined distinctive famous machine learning models, including logistic regression, multilayer completely associated neural networks and deep convolutional neural networks. Utilizing huge models and datasets, we demonstrate Mean-ADAM can fathom practical deep learning problems.

Mean-ADAM functions to overcome the oscillation problem of the weights at the expense of presenting a new constant value. We will show that, our proposed algorithm still enjoys a faster convergence rate with a good SGD with momentum and much better than existing adaptive gradient methods such as ADAM, AMSGRAD, etc.

A. Baseline Algorithms

We compare our proposed algorithms against several state-of-the-art algorithms including: 1) SGD with momentum 2) ADAM 3) AMSGRAD 4) PADAM 5) NADAM (Nesterov ADAM) and 6) ADADELTA.

B. Datasets/prameters Settings/ CNN Architectures

For the former, we have experimented with different architectures on different popular datasets for image classification: GoogleNet, ResNet56, VGGNet [16] and Inception V1 [17] on the CIFAR 10 [18]. The goal is to classify images into one of 10 classes for CIFAR 10 and 100 classes for CIFAR 100.

The data sets contain a set of 50000 $32 - 32$ RGB images for training and 10000 images for testing set which are chosen, these architectures have given their superior performance on several image classification bench-marking tasks.

In particular, the model used for these experiments include three convolution layers with max pooling followed by a fully connected layer with 32 hidden units which has represented a test on CIFAR 10, CIFAR 100, MNIST and Fashion MNIST task for 150^{th} epochs. We could attach the long outcomes for the other architectures, but because of the paucity of space we can't append those outcomes.

C. System specification & Implementation Environment

For these experiments, we have used the TensorFlow library with its high level API tf.keras in order to build and train deep learning models with the configuration of GE Force GTX1060 with 6GB GDDR5 as the GPU with 4GB RAM which needs about four days to train each comparison. We adopt different architectures to evaluate the performance of our proposed algorithms.

For all experiments, we have used a fixed multi-stage learning rate decaying scheme: we decay the learning rate by 0.0001 at the 50^{th} and 150^{th} epochs. To choose the best base learning rate for all algorithms we have performed a grid search on validation set, the first order and the second order momentum parameter $(\beta_1, \beta_2 \subset [0, 1))$ for all adaptive gradient methods. In term of choice the constant number, we recommend to fix any value bound to infinity. We avoid the use of any weight decay regularization.

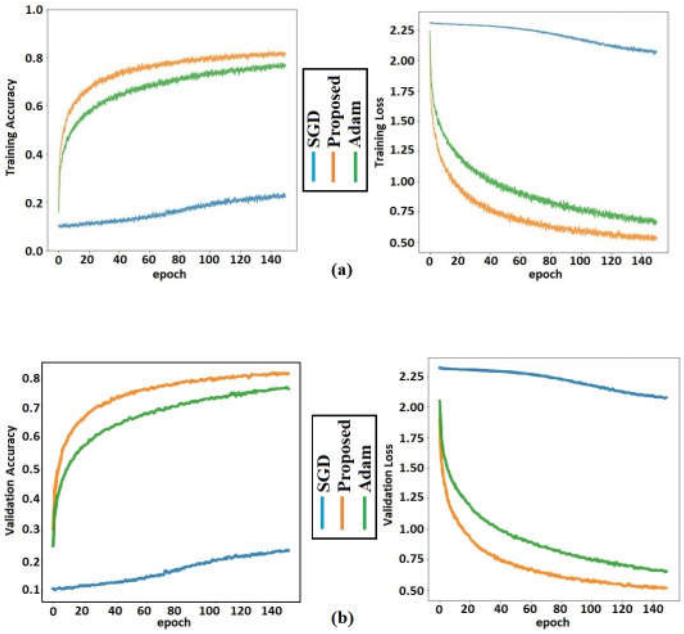


Fig. 1. *Comparison with SGD and ADAM for CIFAR 10 dataset according to the training accuracy and loss (a) and validation accuracy and loss(b).*

Figure 1(a) illustrates the worst accuracy and highest loss and 1(b) illustrates the worst validation accuracy and highest validation loss despite more generalization solution. Since we

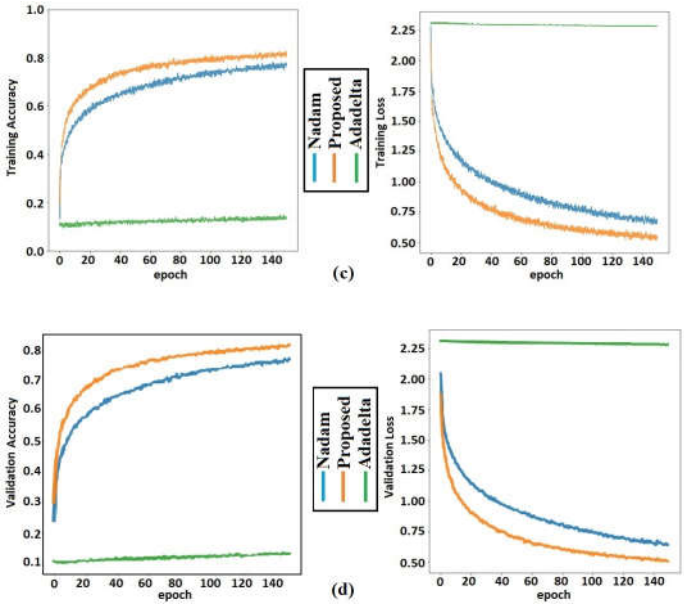
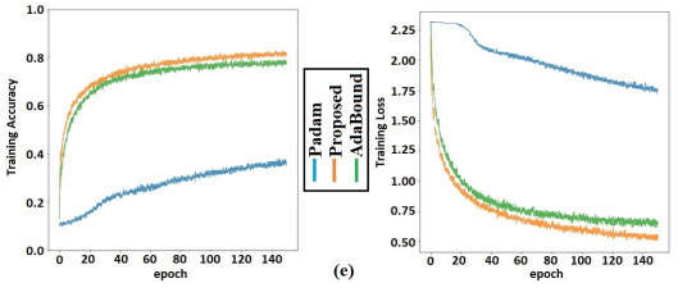


Fig. 2. *Comparison with NADAM and ADADELTA for CIFAR 10 dataset according to the training accuracy and loss (c) and validation accuracy and loss(d).*

kept fixed learning rate for these preparation situations where at the hour of SGD preparing, we did not utilize any force. The proposed calculation and ADAM have performed together well, however after twentieth ages the proposed has contacted the zenith of the accuracy and validation accuracy with the least loss.

ADADELTA has also performed the least gradual increasing training and validation accuracy rate with the passage of epochs. Meanwhile, the proposed algorithm also performs the best graphical figure with the comparison of the NADAM and ADADELTA.



According to the fig 3(e) and 3(f), PADAM has gradually increased the training and validation accuracy, whereas ADABOUND has shown must better accuracy rate with the very beginning of the epoch as we have also fixed a constant learning rate despite its dynamic bounds of learning rate theorem. However, the proposed method has achieved the highest accuracy rate similarly.

IV. EXPERIMENT RESULTS

The Table I has shown the accuracy rate for the MNIST, Fashion MNIST and CIFAR 100 dataset and the Table II has

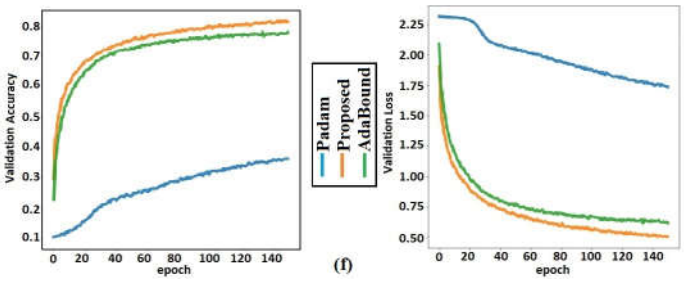


Fig. 3. *Comparison with PADAM and ADABOUND for CIFAR 10 dataset according to the training accuracy and loss (e) and validation accuracy and loss(f).*

illustrated the detail data for the CIFAR 10 architecture. The proposed has shown the highest accuracy rate among all others.

TABLE I
LOSS AND ACCURACY COMPARISON WITH DIFFERENT METHODS ON MNIST, FASHION MNIST, CIFAR 100 DATASET

Methods	**Accuracy**		
	MNIST	***Fashion MNIST***	***CIFAR 100***
ADAM	99.43%	92.41%	88.31%
NADAM	99.01%	90.81%	87.19%
PADAM	99.20%	88.84%	83.40%
SGD	96.30%	81.21%	78.20%
PROPOSED	**99.49%**	**93.62%**	**89.01%**

TABLE II
LOSS AND ACCURACY COMPARISON WITH DIFFERENT METHODS ON CIFAR 10 DATASET

Methods	**Epochs**	**Loss**	**Accuracy**	**Val. Loss**	**Val. Acc.**
SGD	50^{th}	2.27	0.74	2.29	0.12
	150^{th}	2.07	0.81	2.27	0.13
ADAM	50^{th}	0.93	0.67	0.89	0.69
	150^{th}	0.66	0.76	0.65	0.77
NADAM	50^{th}	0.94	0.66	0.90	0.62
	150^{th}	0.66	0.76	0.69	0.78
ADADELTA	50^{th}	2.29	0.11	2.29	0.12
	150^{th}	2.28	0.13	2.27	0.13
PADAM	50^{th}	2.03	0.23	1.98	0.28
	150^{th}	1.77	0.36	1.63	0.41
ADABOUND	50^{th}	0.78	0.72	0.68	0.76
	150^{th}	0.64	0.77	0.53	0.77
PROPOSED	50^{th}	0.72	0.77	0.63	0.54
	150^{th}	**0.93**	**0.82**	**0.77**	**0.82**

During the analysis of Table II, the most surprising result has occurred for the SGD. The validation accuracy of the SGD is so poor where SGD is known to learn a more generalized solution. The generalization performance of an optimizer mostly depends on the experimental dataset, network architecture, and the parameter. For example, most complex open-source datasets out here are ImageNet. Its common practice that people use ADAM as the optimizer on other complex vision takes like scene purring, scene understanding tasks. In Table I, we observe SGD with it's improved training accuracy where the training loss is bad enough to corrupt the overall

learning. This incident occurs due to the weight initialization/ structural feature of the used feed-forward network. ADADELTA has additionally performed like SGD due to this equivalent explanation. Be that as it may, the proposed strategy has demonstrated the best outcome among every other technique.

V. CONCLUSION & FUTURE WORKS

We have proposed Mean-ADAM, a modification of the ADAM, has accomplished the most noteworthy accuracy rate for different popular datasets than other existing algorithms. The concept of the adaptive gradient methods is not very ancient project. There are numerous researchers pursuing deep learning approaches via various deep learning networks. Optimizer is a vital part of these architectures to improve the major accuracy rate. It is intriguing to precise how Mean-ADAM performs with a constant accuracy rate from the earliest starting point to keep going for such celebrated neural networks such as ResNet, VGGNet, Inception V1, and so on. Mean- ADAM optimization method has stretched the updated weights by an external weight to decrease the oscillation and overcome a better accuracy rate than all other adaptive gradient methods till the last of the training. As δ is the fixed parameter, there is no reason to decrease the external weight. However, if we decrease the external weight gradually there will be several effects that can be occurred such as better accuracy, better generalization, low amount perturbation and invariant of dataset. Therefore, these can be our future research topics according to these optimization techniques.

Moreover, we have an affection for training complex datasets such as ImageNet. However, at this time this is impossible for spending impediments. We will compute for the complex datasets using this algorithm over the long run.

REFERENCES

[1] H. Robbins and S. Monro, "A stochastic approximation method", *Ann. Math. Statist. Vol. 22, No. 3, pp. 400-407, Sep. 1951.*

[2] D. P. Kingma and L. J. Ba, "ADAM: A method for stochastic optimization", *in Int. Conf. Learn. Rep., San Diego, USA, May 7-9, 2015.*

[3] T. Tieleman and G. Hinton, "RmsProp: Divide the gradient by a running average of its recent magnitude", COURSERA: Neural Networks for machine learning.(Feb 5,2016).

[4] J. C. Duchi, E. Hazan and Y. Singer, "Adaptive Subgradient methods for online learning and stochastic optimization", *in Conf. on Learn. Theor., Haifa, Israel, Jun. 27-29, 2010.*

[5] M. D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method", *CoRR,abs/1212.5701,2012, arXiv preprint arXiv:1212.5701v1.*

[6] T. Dozat, "Incorporating Nesterov Momentum into ADAM", *under review on Int. Conf. Learn. Rep., San Juan, Puerto Rico, May. 2-4, 2016.*

[7] S. J. Reddi, S. Kale and S. Kumar, "On the Convergence of Adam and Beyond". *in Int. Conf. Learn. Rep., Vancouver, BC, Canada, Apr. 30-May. 3, 2018.*

[8] J. Chen and Q. Gu, "PADAM: closing the generalization gap of adaptive gradient methods in training deep neural networks" *in Int. Conf. Learn. Rep., New Orelans, LA, USA, May. 6-9, 2019, arXiv preprint arXiv: 1806.06763, 2018.*

[9] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization", *in Int. Conf. Learn. Rep., New Orleans, LA, USA, May. 6-9, 2019.*

[10] H. Huang, C. Wang and B. Dong, "Nostalgic Adam: Weighting More of the Past Gradients when Designing the Adaptive Learning Rate", *in Int. Jt. Conf. on Artif. Intell., Macao, China, Aug. 10-16, 2019, arXiv preprint arXiv:1805.0557,2018.*

[11] S. Wang, J. Sun and Z. Xu, "HyperAdam: A learnable task-Adaptive Adam for Network Training", *in Innov. App. Artif. Intell. Conf., Honolulu, Hawaii, USA, Jan. 27 – Feb. 1, 2019, arXiv preprint arXiv:1811.0899, 2018.*

[12] T. T. Phuong and L. T. Phong, "The convergence proof of AMSGRAD and a newer version", *pp. 61706-61716, arXiv preprint arXiv:1904.0359 , 2019.*

[13] L. Luo, Y. Xiong, Y. Liu and X. Sun, "Adaptive Gradient Methods with dynamic bound learning rate", *in Int. Conf. Learn. Rep., New Orleans, LA, USA, May. 6-9, 2019, arXiv preprint arXiv: 1902.0984, 2019.*

[14] I. Sutskever, J. Martens, G. Dahl and G. Hinton, "On the importance of initialization and momentum in deep learning", *in Int. Conf. Mach. Lear., PMLR 28(3):1139-1147, Atlanta, Georgia, USA, June. 17-19, 2013.*

[15] N. S. Keskar and R. Socher, "Improving Generalization Performance by Switching from Adam to SGD", *pp. 65-71,2016, arXiv preprint arXiv: 1712.07628,2017.*

[16] Simonyan, Karen Zisserman, Andrew. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv 1409.1556.*

[17] Szegedy, Christian Liu, Wei Jia, Yangqing Sermanet, Pierre Reed, Scott Anguelov, Dragomir Erhan, Dumitru Vanhoucke, Vincent Rabinovich, Andrew. (2015). Going deeper with convolutions. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 1-9. 10.1109/CVPR.2015.7298594.*

[18] Krizhevsky, Alex Sutskever, Ilya Hinton, Geoffrey. (2012). ImageNet Classification with Deep Convolutional Neural Networks.*Neural Information Processing Systems. 25. 10.1145/3065386.*