Manchester Metropolitan University

e-space

Manchester Metropolitan University's Research Repository

Please cite the Published Version

Khan, W, Ali, S, Muhammad, USK, Jawad, M, Ali, M and Nawaz, R (2020) AdaDiffGrad: An Adaptive Batch Size Implementation Technique for DiffGrad Optimization Method. In: 14th International Conference on Innovations in Information Technology (IIT), 17 November 2020 - 18 November 2020, Al Ain, United Arab Emirates.

DOI: https://doi.org/10.1109/IIT50501.2020.9299013

Publisher: IEEE

Version: Accepted Version

Downloaded from: https://e-space.mmu.ac.uk/627193/

Additional Information: "(c) 2020 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, including reprinting/ republishing this material for advertising or promotional purposes, creating new collective works for resale or redistribution to servers or lists, or reuse of any copyrighted components of this work in other works."

Enquiries:

If you have questions about this document, contact openresearch@mmu.ac.uk. Please include the URL of the record in e-space. If you believe that your, or a third party's rights have been compromised through this document please see our Take Down policy (available from https://www.mmu.ac.uk/library/using-the-library/policies-and-guidelines)

AdaDiffGrad*: An Adaptive Batch Size Implementation Technique for DiffGrad Optimization Method.*

Wajid Khan
Department of Computer Science
COMSATS University, Abbottabad.
Email: fa17-r17-009@cuiatd.edu.pk

Sara Ali
Department of Robotics
National University of Sciences and Technology, Islamabad, Pakistan.
Email: sarababer@smme.nust.edu.pk

Muhammad U.S. Khan
Department of Computer Science
COMSATS University, Islamabad Campus.
Email: ushahid@cuiatd.edu.pk

Muhammad Jawad
Department of Electrical Engineering
COMSATS University, Lahore Campus.
Email: mjawad@cuilahore.edu.pk

Mazhar Ali
Department of Computer Science
COMSATS University, Islamabad Campus.
Email: mazhar@ciit.net.pk

Raheel Nawaz
School of Computing, Mathematics and Digital Technology, Manchester Metropolitan University, UK
Email: r.nawaz@mmu.ac.uk

***Abstract*-**Stochastic Gradient Descent is a major contributor to the success of the deep neural networks. The gradient provides basic knowledge about the function direction and its rate of change. However, SGD changes the step size equally for all parameters irrespective of their gradient behavior. Recently, several efforts have been made to improve the SGD method, such as AdaGrad, RMSprop, Adam, and diffGrad. The diffGrad is an appropriate and enhanced technique that uses fraction constant based on previous gradient information for gradient calculation. This fraction constant decreases the momentum resulting in slow convergence towards an optimal solution. This paper addresses the slow convergence problem of the diffGrad algorithm and proposed a new adaDiffGrad algorithm**.** In adaDiffGrad an adoptive batch size is implemented for the diffGrad to overcome the problem of slow convergence. The proposed model is experimented for image categorization and classification over CIFAR10, CIFAR100, and FakeImage dataset. The results are compared with the state of art models, such as Adam, AdaGrad, DiffGrad, RMSprop, and, SGD. The results show that adaDiffGrad outperforms other optimizers and improves the accuracy of the diffGrad.

Keywords: *Optimization, Gradient Descent, Adam, DiffGrad, Image Classification, Ada-Batch size, and ResNet*

I. INTRODUCTION

Deep neural networks are widely used in solving various problems including pattern recognition, classification, clustering, dimensionality reduction, computer vision, Natural Language Processing (NLP), regression, and predictive analysis. The neural network algorithms become famous due to the availability of resources such as GPU, memory, and storage, etc.

To find an optimal solution, most of the neural network models use gradient descent as an optimizer [1].The Gradient Descent and its variants batch [2], mini batch [3], and stochastic [4] are the commonly used optimizers, however, these optimizers require complete dataset for gradient calculation. If the dataset is large these optimizers suffer from slow convergence towards global minimal. Recently, many attempts have been made to overcome the slow convergence problem of gradient descent. Adam performs reasonably well compared to previous adaptive techniques [5]. However, it does not utilize past gradient information. To overcome the limitation of Adam, a novel optimizer diffGrad is proposed that used the difference between the present and past gradients. It uses gradient behavior to control the learning rate at the optimization stage resulting in an optimal solution. If the gradient difference is large this means that optimization is not stable and therefore the diffGrad allows a high learning rate. Conversely, if gradient difference is small it means that model is closed to an optimal solution. Then the diffGrad lowers the learning rate automatically. Although diffGrad outperforms all the previous optimizers in terms of accuracy, however, its performance is very slow for an optimal solution.

In this paper, Ada-diffGrad is proposed to overcome the slow convergence problem of diffGrad optimizer by adaptively increasing the batch-size during training. The major contributions of this paper are summarized as follow:

1. This paper proposes a new adaDiffGrad optimization method for CNN by using adaptive

978-1-7281-8184-4/20/$31.00 ©2020 IEEE

Authorized licensed use limited to: Manchester Metropolitan University. Downloaded on January 30,2021 at 17:11:25 UTC from IEEE Xplore. Restrictions apply.

batch-size with diffGrad to overcome the slow convergence problem of an optimizer.
2. The proposed scheme utilizes the accumulation of the scheduler to adopt the batch size at specific intervals.
3. Keeping the ratio α/r constant, the proposed model adopts the learning rate α simultaneously to avoid the overshooting problem.

The rest of the paper is structured in the following manner: Section II presents related work, section III proposes AdaDiffGrad scheme, Section IV describes the experimental setup, section V indicates the experimental results and findings, and section VI concludes the paper.

II. RELATED WORK

This section provides an overview of related work regarding previous schemes that are used as optimizers. Furthermore, different optimizers and their variants are also discussed. It is also highlighted the major deficit of optimizers and their enhancement proposed by a new scheme.

Gradient Descent [6], uses the weights of all samples for parameter update. The large sample requires more time for updating parameters. Weights are calculated for GD as:

$$\Delta\omega = \alpha \sum_{\iota=1}^{N} \left(y^{(\iota)} - \theta(\omega^T x)^{(\iota)}\right) x^{(\iota)} \qquad (1)$$

Where α $used\ for$ learning rate, θ for a parameter value for ιth sample with ι=1,2,3……N. N is the sample size. While $y^{(\iota)}$ is target value and $(\omega^T x)^{(\iota)}$ is output value.

In SGD [7] weights can be updated after one sample or subset of samples.

$$\Delta\omega = \alpha\left(y^{(\iota)} - \theta(\omega^T x)^{(\iota)}\right) x^{(\iota)} \qquad (2)$$

SGD updates the parameters as

$$\theta_{t+1,\iota} = \theta_{t,\iota} - \alpha_t \times g_{t,\iota} \qquad (3)$$

Here $g_{t,\iota}$. is gradient for i^{th}

$$g_{t,\iota} = \frac{\partial(\mathcal{L}_{t,}\theta)}{\partial(\theta_{t,\iota})} \qquad (4)$$

Here,

$$\mathcal{L}_{t,}\theta = \frac{1}{N_b} \sum_{j=1}^{N_b} \mathcal{L}_{t,\theta,j} + \sigma R_{t,\theta} \qquad (5)$$

For which N_b is the number of training images in batch,$\mathcal{L}_{t,\theta}$ is cross-entropy, and $R_{t,\theta}$ is regularization loss.

SGDM [8] brings a new idea to use a gradient for calculating the moment for parameters as shown below

$$m_{t,\iota} = \beta m_{t,\iota-1,\iota} + g_{t,\iota} \qquad (6)$$

The parameters are updated using the following formula:

$$\theta_{t+1,\iota} = \theta_{t,\iota} - \alpha m_{t,\iota} \qquad (7)$$

Where $m_{t,\iota}$ is the moment gained and β is hyperparameter to control moment.

AdaGrad [9] modified the SGD approach to calculate the parameter by normalizing the learning rate α given as

$$\theta_{t+1,\iota} = \theta_{t,\iota} - \frac{\alpha_{t\,\times g_{t,\iota}}}{\sqrt{G_{t,\iota} + \epsilon}} \qquad (8)$$

Where $G_{t,\iota}$ is the sum of squares of gradients for t steps and ϵ is a small value added to avoid divide by zero error.

RMSProp [10] and AdaDalta [11] calculate the $G_{t,\iota}$ with decay rate β as to avoid a sudden decrease in the learning rate.

$$G_{t,\iota} = \beta G_{t-1,\iota} + (1-\beta)\left(g_{t,\iota}{}^2\right) \qquad (9)$$

Adam [12] is also another popular optimizer which computesα based upon two factors 1st and 2nd order moments (mean and variance) as

$$m_{t,\iota} = \beta_1\left(m_{t-1,\iota} + (1-\beta_1) g_{t,\iota}\right) \qquad (10)$$

$$\upsilon_{t,\iota} = \beta_2 \upsilon_{t-1,\iota} + (1-\beta_2) g_{t,\iota}{}^2 \qquad (11)$$

Where β_1 and β_2 are decay rates and $m_{t,\iota}$ and $\upsilon_{t,\iota}$ are mean and variance of gradients.It was observed that the initial value of $\upsilon_{t,\iota}$was very small as compared to $m_{t,\iota}$.This problem was addressed as:

$$\widehat{m}_{t,\iota} = \frac{m_{t,\iota}}{(1-\beta_1^t)} \text{and } \hat{\upsilon}_{t,\iota} = \frac{\upsilon_{t,\iota}}{(1-\beta_2^t)}, \qquad (12)$$

Where $\widehat{m}_{t,\iota}$ and $\hat{\upsilon}_{t,\iota}$ is biased corrected 1st and 2nd order moments.

Now parameters are calculated using following formula:

$$\theta_{t+1,\iota} = \theta_{t,\iota} - \frac{\alpha_{t\,\times}\widehat{m}_{t,\iota}}{\sqrt{\hat{\upsilon}_{t,\iota}} + \epsilon} \qquad (13)$$

Another problem with Adam is when the second moment decreases significantly, the friction in the optimization landscape decreases, for this decrease the learning rate and divergence increase and process overshoot to an optimal solution. AMSGrad [13] addresses

Authorized licensed use limited to: Manchester Metropolitan University. Downloaded on January 30,2021 at 17:11:25 UTC from IEEE Xplore. Restrictions apply.

this problem by considering the maximum of second moments as:

$$\hat{v}_{t,\iota}^{\ max} = \max\left(\hat{v}_{t-1,\iota}^{\ max}, \hat{v}_{t,\iota}\right) \quad (14)$$

Thus, parameters updating for AMSGrad method areas

$$\theta_{t+1,\iota} = \theta_{t,\iota} - \frac{\alpha_t \times \widehat{m}_{t,\iota}}{\sqrt{\hat{v}_{t,\iota}^{\ max} + \epsilon}} \quad (15)$$

For Adam optimizer the default values for

$\beta_1 = 0.9$ and $\beta_2 = 0.999$ and $\alpha \in [10^{-2}, 10^{-4}]$

Another problem is the automatic adjustment of the learning rate of fractions for the first moment to avoid overshooting. DiffGrad [14] introduces DiffGrad fraction coefficient (DFC) to control the learning rate using the information of previous gradients. So, the equation of parameter update for DiffGrad is defined as:

$$\theta_{t+1,\iota} = \theta_{t,\iota} - \frac{\alpha_t \times \mathcal{E}_{t,\iota} \widehat{m}_{t,\iota}}{\sqrt{\hat{v}_{t,\iota}} + \epsilon} \quad (16)$$

Where $\mathcal{E}_{t,\iota}$ is DFC for DiffGrad calculated as:

$$\mathcal{E}_{t,\iota} = AbsSig\left(\Delta g_{t,\iota}\right) \quad (17)$$

AbsSig is a non-linear sigmoid function for x that bound the values between 0.5 and 1and calculated as

$$\mathcal{A}bsSig(x) = \frac{1}{1 + e^{-|x|}} \quad (18)$$

And $\Delta g_{t,\iota}$ is gradient difference and calculated as

$$\Delta g_{t,\iota} = g_{t-1,\iota} - g_{t,\iota} \quad (19)$$

In diffGrad, the DFC is used to control the gradient fluctuation near the optimum solution. The diffGrad also claims that the inclusion of DFC provides a high learning rate for large gradient change and low learning rate for low gradient change areas.

A. *Effects of Adaptive Batch*

Adabatch [15] explains the effects of adaptive batch size during training of optimizers. It explains the impacts of a batch size overtraining, learning rate, and works per epoch for fixed and adoptive batch size. The scheme also describes the relationship among batch size, learning rate, and performance. The batch size plays a vital role in model performance. If the batch size is small the training process takes time and convergence will be slow, while large batch size overshoots the training process and ignores the local minimal. The adaptive batch size is the solution to this problem. The Adabatch provides the adoptive batch size to ignore these problems to gain accurate and timely results. The AdaBatch also explains that adapting the learning rate allow the training process to enable a large batch size without losing accuracy. Here it explains the relationship between batch size and learning rate, and this analysis provides the basis for using adaptive batch size technique for DiffGrad optimizer.

B. *Learning Rate*

For supervised learning, the data set is divided into three portions training, validation, and testing. The training part consists of consecutive forward and backward propagation. Let a training data consist of sample data $\{(x, z^*)\}$, which are $x \in \mathbb{R}^\iota$as inputs and $z^* \in \mathbb{R}^j$as output. During training weights $\omega$is used to solve the optimization problem.

$$arg\omega^{min}\mathcal{L} \quad (20)$$

Weights updated for ιth iteration are calculated as

$$\omega_{\iota+1} = \omega_\iota - \frac{\alpha}{r}\Delta\omega_\iota \quad (21)$$

$$\omega_{\iota+\tilde{p}} = \omega_\iota - \frac{\tilde{\alpha}}{\beta r}\sum_{\iota=1}^{\tilde{p}} \Delta\widetilde{\omega_\iota} \quad (22)$$

Where large batch size βr is inserted and updated matrix $\widetilde{\omega_\iota}$ are calculated with batch sizeβr and learning rate$\tilde{\alpha}$. We assume that $\Delta\omega_\iota \approx \Delta\omega_\iota$ are equal for both equations and that $1/\beta$ is learning rate decay. Equation (3) shows the relationship between learning rate and batch size that increasing batch size can mimic the learning rate decay and this experiment shows that effective learning rate for adaptive batch size will increase the performance.

C. *Work per Epoch*

A fixed batch size requires a fixed number of flops per iteration throughout the training process. As this technique involves adaptive batch size flops per iteration also increases. But the flops per epoch remains fixed for all processes if the computation is a linear function of batch size r. This point is briefly explained as a fully connected layer with weight $\omega \in \mathbb{R}^{\iota j}$ with input $\mathcal{X} = [\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3, \ldots \mathcal{X}_n] \in \mathbb{R}^{j \times r}$ and gradient vector $\mathcal{V} = [\mathcal{V}_1, \mathcal{V}_2, \mathcal{V}_3, \ldots . \mathcal{V}_n]$. Most of computation expensive operations during training are matrix multiplication.

$$\mathcal{Y} = \omega\, \mathcal{X} \quad (23)$$

and,

$$\mathcal{U} = \omega^T \mathcal{V} \quad (24)$$

Authorized licensed use limited to: Manchester Metropolitan University. Downloaded on January 30,2021 at 17:11:25 UTC from IEEE Xplore. Restrictions apply.

Operations require $\mathcal{O}(\iota j r)$ flops per iteration and $\mathcal{O}(\iota j r p)$ flops for epoch. As batch size increase to βr, then flops per iterations approach to$(\iota j \beta r)$.However, increasing batch size by β also decreases the number of iterations by a factor of β.Therefore the flops per epoch remain fixed$\mathcal{O}(\iota j r p)$. The larger batch size does not affect the flops per epoch, but it will perform better in terms of time and hardware usage.

III. PROPOSED METHODOLOGY

The DiffGrad performs well to find the optimal solution by using past gradient information for two consecutive iterations. It overcomes the problem of Adam and previous optimizers by using gradient descent. It uses a gradient to control the learning rate to avoid the overshooting problem. However, DiffGrad models use static batch-size towards convergence that forces the user to choose between accuracy and efficiency. The adaptive batch size tends for convergence as well as improves efficiency and performance. Our approach is to resolve the trade-off between small and large batch size to adaptively increase the batch size during training. The proposed technique resolves the problem of slow convergence by integrating AdaBatch [15] batch size with DiffGrad. The proposed method increases the batch size after specific epochs and controls the learning rate α to keep the ratio $(\frac{\alpha}{r})$ constant. This ratio enforces the model from overshooting.

IV.EXPERIMENTAL SETUP

This section presents the overall process during the experimental setup i.e. the datasets details, parameter settings, and system specification uses during the experiment.

A. Dataset

The dataset used for this experiment is CIFAR10, CIFAR100, and FakeImages dataset. All these three datasets are available online for experiments and educational purposes use. Both CIFAR10 and CIFAR100 datasets contains 60k images for training and 10k images for testing. On the other hand, FakeImages dataset is used for fake image detection purposes. This dataset contains 3000 images of random people's faces. Among these 2500 images are used for training and 500 for testing purposes. All images are divided into two classes real and fake equally for both the training and testing phase.

B. System Specification

The experiment was performed on Google Colaboratory that provides free Jupyter notebook environment and resources for computation and experiments. The instance that was used for experimental possesses Tesla k80 GPU processing, 12GB memory, and 100GB storage. The experiment was performed using python language with TensorFlow, Torch, and Cuda libraries that are already available in the notebook environment.

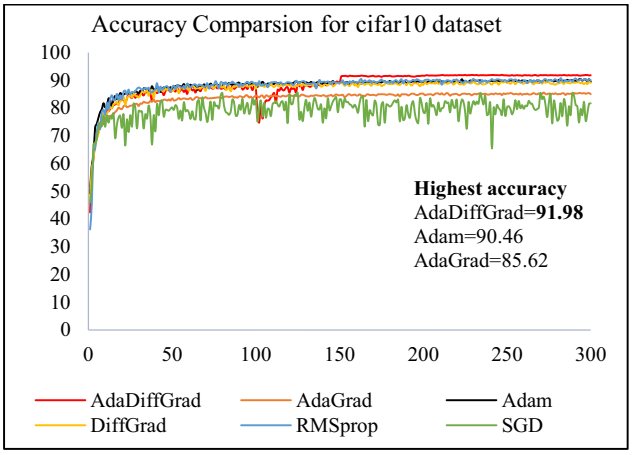
C. Parameters setting

This section provides information about the parameter setting used during the experiment. The optimization techniques Adam, AdaGrad, DiffGrad, RMSprop, SGD, and AdaDiffGrad were used with ResNet neural network. The number of epochs for all optimizer is 300 with 128, 256 and 512 batch size for each dataset. The batch size for all 300 epochs was static for all other optimizers except AdaDiffGrad. The batch size and learning rate for the first 100 epochs were fixed for AdaDiffGrad. After 100 epochs batch size is doubled and the learning rate is half for the next 50 epochs. And this process is repeated for 300 epochs for AdaDiffGrad. So, after every 50 epochs batch size is adopted to double and learning is cut to half for the whole experiment. The source code of the whole experiment is available online for demonstration and verification.

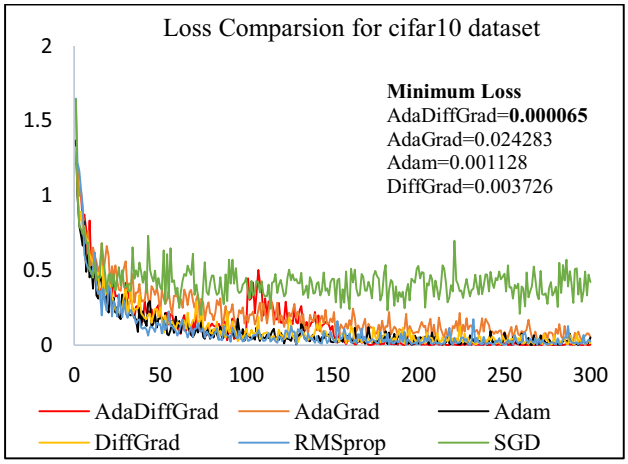
V. EXPERIMENTAL RESULTS

The accuracy validation of AdaDiffGrad optimizer is compared with state of art optimizers such as Adam, AdaGrad, DiffGrad, RMSprop, and SGD. In figure 1 (a) shows that the accuracy comparison of these methods for CIFAR10 dataset. The graph indicates that AdaDiffGrad outperforms some state of art algorithms in terms of accuracy. The proposed method achieves the highest accuracy of 91.98% while Adam, AdaGrad, DiffGrad, RMSprop, and SGD models achieve 90.46%, 85.62%, 89.71%, 90.8%, and 85.5% respectively. Similarly, AdaDiffGrad convergence to training loss to 0.000065 which is minimum among all the above describes models as indicates in figure 1(b). For CIFAR100 dataset AdaDiffGrad gain the highest accuracy of **67.03%** while approaching the lowest training loss of **0.06** among compared optimizers. For the FakeImage dataset AdaDiffGrad also achieved the highest accuracy of **93.17%** while Adam, AdaGrad, DiffGrad, RMSprop and SGD optimizers approach to 91.70%, 90.24%, 89.26% and 85% respectively. This comparison is also shown in Table 1 where the highest accuracy achieved by optimizers is highlighted in bold format. To achieve the best results, the experiment is conducted using the default parameters for every optimizer. The CIFAR10 dataset is experimented with three different batch sizes(128, 256, 215) to find the suitable batch size for the experiment. The result indicates that a suitable batch size for all optimizers is 128 because most of the optimizer achieves the highest accuracy with this batch size.

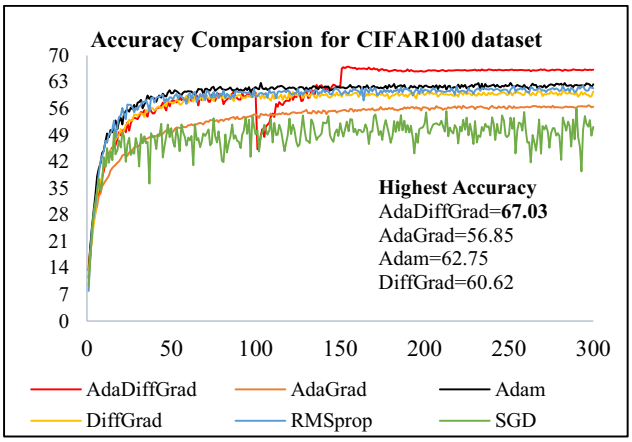
Authorized licensed use limited to: Manchester Metropolitan University. Downloaded on January 30,2021 at 17:11:25 UTC from IEEE Xplore. Restrictions apply.



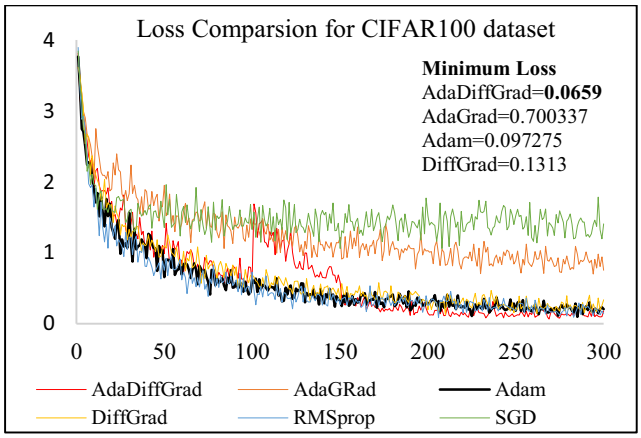
(a)



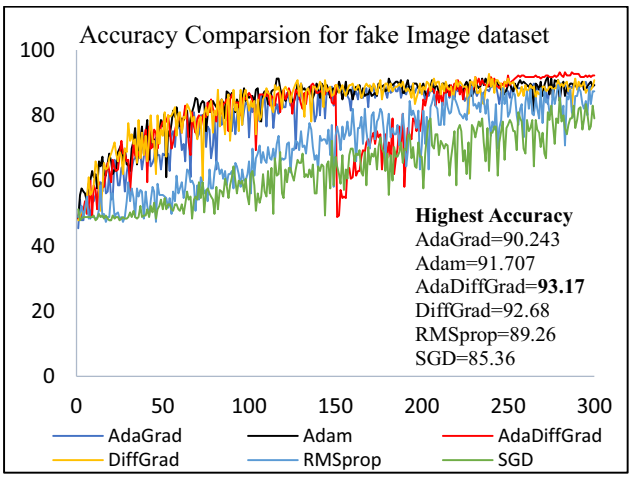
(b)



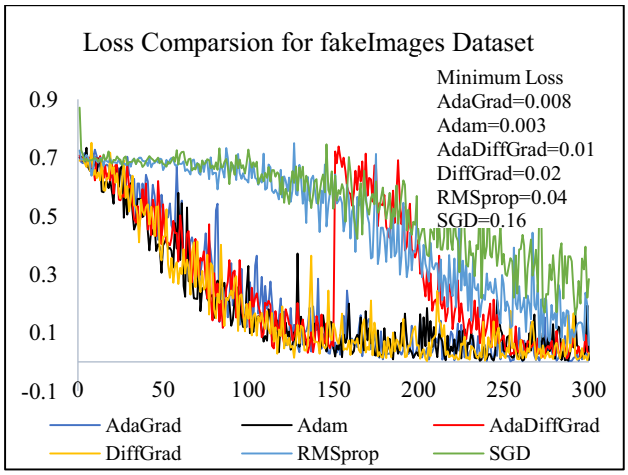
(c)



(d)



(e)



(f)

Fig. 1: The results comparison among Adam[12], AdaGrad[9], DiffGrad [14], SGD[7], RMSprop [10] and proposed AdaDiffGrad. (a, c, e) shows the accuracies comparison of optimizer for Cifar10, Cifar100, and FakeImage Dataset, respectively. Whereas, (b, d, f) represents the training Loss for respective optimizers during the training phase.

Authorized licensed use limited to: Manchester Metropolitan University. Downloaded on January 30,2021 at 17:11:25 UTC from IEEE Xplore. Restrictions apply.

Optimizer	***CIFAR10 dataset***			***CIFAR100 dataset***	***FakeImages dataset***
	$N_b = 128$	$N_b = 256$	$N_b = 512$	$N_b = 128$	$N_b = 128$
Adam	90.46	90.17	88.84	62.75	91.70
AdaGrad	85.62	84.39	84.74	56.85	90.24
DiffGrad	89.71	88.51	87.56	60.62	92.68
SGD	85.5	87.69	88.95	55.97	85.36
RMSprop	90.8	90.54	90.27	62.03	89.26
AdaDiffGrad	***91.98***	***91.73***	***91.12***	***67.03***	***93.17***

Table: 1 shows the comparison results in terms of accuracy over CIFAR10, CIFAR100 and FakeImages dataset among Adam, AdaGrad, DiffGrad, SGD, RMSprop and proposed AdaDiffGrad optimizers. The comparison is based upon three different batch sizes i.e. ($N_b = 128, 256, 512$). The best accuracy among all describes models are highlighted in bold.

For further results 128 batch size is adopted for CIFAR100 and FakeImages datasets. From Figure 1and Table 1, it is indicated that AdaDiffGrad outperforms most state of art optimizers.

VI.CONCLUSION

In this paper, a new optimization method AdaDiffGrad is proposed, which is an enhancement to DiffGrad. The AdaDiffGrad scheme overcomes the slow convergence issues of diffGrad. The proposed scheme added adaptive batch size to DiffGrad to improve its accuracy and performance. The adaptive batch size enhances the convergence rate as well as improves the efficiency of the model. The proposed scheme is tested with ResNet50 model for image classification over CIFAR10, CIFAR100, and FakeImages dataset. The results of AdaDiffGrad are compared with state-of-the-art optimizers such as Adam, AdaGrad, DiffGrad, RMSprop, and SGD. The results indicate that the adaDiffGrad achieves the best accuracy among the described models and outperforms them.

References:

[1]. Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010* (pp. 177-186).Physica-Verlag HD.

[2]. Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.

[3]. Hinton, G,Srivastava., &Swersky. (2012). Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. Cited on, 14(8).

[4]. Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In Proceedings of COMPSTAT'2010(pp.177-186). Physica-Verlag HD.

[5]. Shrestha, A., & Mahmood, A. (2019). Review of deep learning algorithms and architectures. *IEEE Access*, *7*, 53040-53065.

[6]. Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

[7]. Shamir, O., & Zhang, T. (2013, February). Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *International conference on machine learning* (pp. 71-79).

[8]. Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013, February). On the importance of initialization and momentum in deep learning. In *International conference on machine learning* (pp. 1139-1147).

[9]. Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, *12*(Jul), 2121-2159.

[10]. Hinton, Srivastava, &Swersky,. (2012). Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, *14*(8).

[11]. Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

[12]. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[13]. Reddi, S. J., Kale, S., & Kumar, S. (2019). On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*.

[14]. Dubey, S. R., Chakraborty, S., Roy, S. K., Mukherjee, S., Singh, S. K., & Chaudhuri, B. B. (2019). diffGrad: An Optimization Method for Convolutional Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*.

[15]. Devarakonda, A., Naumov, M., & Garland, M. (2017). Adabatch: Adaptive batch sizes for training deep neural networks. *arXiv preprint arXiv:1712.02029*.

Authorized licensed use limited to: Manchester Metropolitan University. Downloaded on January 30,2021 at 17:11:25 UTC from IEEE Xplore. Restrictions apply.