diffGrad: An Optimization Method for Convolutional Neural Networks

Shiv Ram Dubey, *Member, IEEE*, Soumendu Chakraborty, Swalpa Kumar Roy, *Student Member, IEEE*, Snehasis Mukherjee, *Member, IEEE*, Satish Kumar Singh, *Senior Member, IEEE*, and Bidyut Baran Chaudhuri, *Life Fellow, IEEE*

***Abstract*— Stochastic gradient descent (SGD) is one of the core techniques behind the success of deep neural networks. The gradient provides information on the direction in which a function has the steepest rate of change. The main problem with basic SGD is to change by equal-sized steps for all parameters, irrespective of the gradient behavior. Hence, an efficient way of deep network optimization is to have adaptive step sizes for each parameter. Recently, several attempts have been made to improve gradient descent methods such as AdaGrad, AdaDelta, RMSProp, and adaptive moment estimation (Adam). These methods rely on the square roots of exponential moving averages of squared past gradients. Thus, these methods do not take advantage of local change in gradients. In this article, a novel optimizer is proposed based on the difference between the present and the immediate past gradient (i.e., diffGrad). In the proposed diffGrad optimization technique, the step size is adjusted for each parameter in such a way that it should have a larger step size for faster gradient changing parameters and a lower step size for lower gradient changing parameters. The convergence analysis is done using the regret bound approach of the online learning framework. In this article, thorough analysis is made over three synthetic complex nonconvex functions. The image categorization experiments are also conducted over the CIFAR10 and CIFAR100 data sets to observe the performance of diffGrad with respect to the state-of-the-art optimizers such as SGDM, AdaGrad, AdaDelta, RMSProp, AMSGrad, and Adam. The residual unit (ResNet)-based convolutional neural network (CNN) architecture is used in the experiments. The experiments show that diffGrad outperforms other optimizers. Also, we show that diffGrad performs uniformly well for training CNN using different activation functions. The source code is made publicly available at https://github.com/shivram1987/diffGrad.**

Manuscript received September 14, 2018; revised March 23, 2019, June 8, 2019, August 16, 2019, and November 3, 2019; accepted November 19, 2019. Date of publication December 23, 2019; date of current version October 29, 2020. This work was supported by IIIT Sri City. *(Corresponding author: Shiv Ram Dubey.)*

S. R. Dubey and S. Mukherjee are with the Computer Vision Group, Indian Institute of Information Technology at Sri City, Chittoor 517646, India (e-mail: shivram1987@gmail.com; snehasis.mukherjee@iiits.in).

S. Chakraborty is with the Indian Institute of Information Technology at Lucknow, Lucknow 226002, India (e-mail: soum.uit@gmail.com).

S. K. Roy is with the Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, Kolkata 700108, India (e-mail: swalparoy@gmail.com).

S. K. Singh is with the Computer Vision and Biometrics Laboratory, Indian Institute of Information Technology at Allahabad, Allahabad 211015, India (e-mail: sk.singh@iiita.ac.in).

B. B. Chaudhuri is with the Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, Kolkata 700108, India, and also with Techno India University, Kolkata 700091, India (e-mail: bidyutbaranchaudhuri@gmail.com).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2019.2955777

***Index Terms*— Adaptive moment estimation (Adam), difference of gradient, gradient descent, image classification, neural networks, optimization, residual network.**

I. Introduction

During the last few years, deep-learning-based techniques have gained more and more popularity in solving problems in different domains, especially where a data-driven approach is required [1]. Due to the availability of GPU-based high-end computational facilities and the huge amount of data, deep-learning-based approaches generally outperform the traditional hand-designed approaches to solve research problems in computer vision [2]–[5], image processing [6], [7], signal processing [8], [9], robotics [10], natural language processing [11], [12], and many other diverse areas of artificial Intelligence. Other applications where deep learning can be used include object tracking [13]–[15], face anti-spoofing and micro-expression recognition [16], [17], hyper-spectral image classification [18], and so on.

The deep neural network has different variants to deal with the different problems, such as convolutional neural networks (CNNs) and generative adversarial networks (GANs) for images, recurrent neural network (RNN), and long short-term memory network (LSTM) for temporal sequences of data, 3-D-CNN for videos, and so on. Research on CNN has observed a rapid growth in recent years, especially on various image-based problems. Different CNN architectures have been proposed for image-related problems such as AlexNet [2], VggNet [4], GoogLeNet [19], and ResNet [20] for image classification; R-CNN [21], Fast R-CNN [22], Faster R-CNN [23], and YOLO [24] for object detection; Mask R-CNN [25] and PANet [26] for instance segmentation; and RCCNet [27] for colon cancer nuclei classification.

In most neural networks, the basic approach that is usually followed for finding an optimal solution is stochastic gradient descent (SGD) optimization [28]. Here, initially, a measure of loss over the current parameter values is computed using a loss function defined for the specific problem being solved. Then, the gradient for each parameter (i.e., in each dimension) in the network is computed and the parameter values are updated in the opposite direction of the gradient by a factor proportional to the gradient. For SGD optimization, the above two steps are repeated until convergence or until a certain number of epochs or iterations are completed. Following are

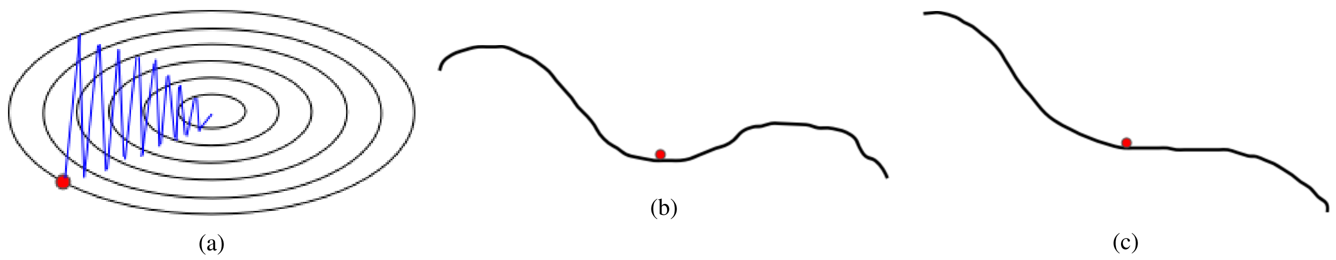


Fig. 1. (a) Optimization landscape for two parameters represented in two directions, i.e., horizontal and vertical directions. The loss changes slow in the horizontal direction and fast in the vertical direction. (b) Local minima scenario in 1-D optimization. (c) Saddle point scenario in 1-D optimization.

the four major drawbacks in the basic SGD approach: 1) if the loss changes quickly for a set of parameters and slowly for another set of parameters, then it leads to very slow learning along shallow dimensions and a jittering effect along steep dimensions [29] as depicted in Fig. 1(a); 2) if the loss function has a local minimum or a saddle point [30], then SGD gets stuck due to zero gradients in that region. These situations are illustrated in Fig. 1(b) and (c); 3) the gradients are usually computed over minibatches, so the gradients can be noisy; and 4) SGD takes the same step for each parameter irrespective of the iteration-wise gradient behavior for that parameter, which leads to poor optimization. Note that each parameter of the network is considered as a dimension in the optimization landscape. Therefore, both dimension and parameter are used interchangeably in this article.

Improving SGD-based optimization method for neural networks has recently become an active area of research interest. In order to address the above problems, several SGD variants have been proposed in the literature. For example, SGD with momentum (SGDM) is an extension of SGD by incorporating the past gradients in each dimension [31], [32]. SGDM maintains a momentum in each dimension as a function of previous momentum and current gradient in that dimension. The goal of SGDM is to develop high "velocity" in any dimension that has a consistent gradient. In SGDM, the jittering problem is reduced with high velocity in consistent gradient dimensions and the saddle point problem is reduced by using the past gradients, which provide some momentum, even when the current gradient is close to zero. SGDM optimization is further improved by Sutskever *et al.* [32] using Nesterov Accelerated Gradient (NAG), which guarantees a better convergence rate than SGD in certain situations. In NAG, the gradients are computed based on an approximation of the next position of the parameters, which are used to update the current moments and parameters. Thus, NAG updates both the moments and gradients, based on future dimensions.

Another widely used variant of the gradient descent method is AdaGrad [33], which is proposed to deal with sparse data. AdaGrad performs larger updates for infrequent parameters which leads to smaller magnitude of gradients and smaller updates for frequent parameters lead to a larger magnitude of gradients. Basically, AdaGrad divides the learning rate with the square root of the sum of the squares of the past gradients for all parameters. Apart from dealing with sparse data, AdaGrad is applied to other kinds of problems as well, such as training large-scale neural nets to recognize cats in Youtube videos [34] and training GloVe word embeddings, as infrequent words require larger updates than frequent ones [35]. However, AdaGrad accumulates the square of gradients which, in turn, may lower the learning rate drastically after some time and eliminate the learning process. Zeiler [36] extended AdaGrad to AdaDelta by removing the problem of a dying learning rate, which was caused by the monotonically increasing sum of square of gradients. AdaDelta also accumulates the square of past gradients, but considers only a few immediate past gradients instead of all past gradients. RMSProp is another attempt to correct the diminishing learning rate of AdaGrad similar to AdaDelta [37], i.e., by accumulating the gradient as an exponentially decaying average of squared gradients. The major difference between AdaDelta and RMSProp is that AdaDelta does not use any learning rate [36], whereas RMSProp uses a learning rate [37].

One of the recent and popular variants of gradient descent is adaptive moment estimation (Adam). Adam computes adaptive learning rates for each parameter [38] by utilizing both first and second moments. Adam accumulates the exponentially decaying average of past gradients similar to SGDM as the first moment. It also accumulates the exponentially decaying average of the square of past gradients similar to AdaDelta and RMSProp as the second moment. The moment can be imagined as a ball rolling down a slope, where Adam behaves like a heavy ball with friction, which thus prefers flat minima in the error surface [39]. It is observed that Adam performs reasonably well in practice as compared to the other adaptive learning-methods. However, Adam does not utilize the change in immediate past gradient information, which is incorporated in the proposed diffGrad method. Very recently, Reddi *et al.* [40] proposed AMSGrad as improvement over Adam. AMSGrad considers the maximum of the past second moment (i.e., "long-term memory of past gradients) in the parameter update procedure. Consequently, AMSGrad imposes more friction to avoid the overshooting of the minimum. AMSGrad does not change the learning rate based on the recent gradient behavior and does not deal with the saddle point problem, either. On the other hand, the proposed diffGrad method controls the learning rate based on the changes in the gradient. Some of the recent works in stochastic gradient methods include the predictive local smoothness based SGD (PLS-SGD) [41], the sign of each minibatch SGD (signSGD) [42], and Nostalgic Adam (NosAdam) [43].

This article proposes a difference of gradient-based optimizer, which improves the well-known Adam [38] with the difference of gradients (diffGrad) over the iterations. The main contributions of this article are summarized as follows.

1) This article proposes a new diffGrad gradient descent optimization method for CNN by considering the local gradient change information between the current and immediate past iteration.
2) We show how the "short-term gradient behavior" can be utilized to control the learning rate in the optimization landscape in terms of the optimization stage, i.e., near or far from an optimum solution. If the change in gradient is large, it means that the optimization is not stable due to local optima, salient region or other factors, and diffGrad allows a high learning rate. If change in gradient is small, it means that the optimization is likely to be close to the optimum solution, and diffGrad lowers the learning rate automatically.
3) The proposed method also utilizes the accumulation of past gradients over iterations to deal with saddle points.
4) We conduct a convergence analysis of diffGrad in terms of the regret bound using the online learning framework. We also derive a proof for diffGrad convergence.
5) An empirical analysis is done by modeling the optimization problem as a regression problem to show the advantages of the proposed diffGrad optimization method over three synthetic complex nonconvex functions.
6) We conduct an experimental study on the proposed method and observe its improved performance for an image categorization task using the ResNet-based CNN architecture. We also experiment with different variants of diffGrad and different activation functions.

The remainder of this article is structured in the following manner. Section II presents the preliminaries in SGD optimization. Section III proposes the diffGrad optimization method. Section IV conducts the convergence analysis. Section V is devoted to the empirical analysis. Section VI presents the experimental setup. Section VII presents the experimental results, comparison, and analysis. Section VIII provides the concluding remarks.

II. Preliminaries

In SGD, all parameters are updated with the same learning rate α_t in the tth iteration as

$$\theta_{t+1,i} = \theta_{t,i} - \alpha_t \times g_{t,i} \tag{1}$$

where $\theta_{t,i}$ and $\theta_{t+1,i}$ are the previous and updated values for the ith parameter with $i = 1, 2, \ldots, d$, where d is the number of parameters, and $g_{t,i}$ is the gradient with respect to the parameter $\theta_{t,i}$ for a loss function $\$$, defined as

$$g_{t,i} = \frac{\partial(\$_{t,\theta})}{\partial(\theta_{t,i})} \tag{2}$$

where $\$_{t,\theta}$ is a loss function with respect to the parameters of the network (θ) in tth iteration. In this article, the cross entropy loss used for image categorization experiments is defined as

$$\$_{t,\theta} = \frac{1}{N_b}\sum_{j=1}^{N_b} \$_{t,\theta,j} + \sigma R_{t,\theta} \tag{3}$$

where N_b is the number of training images in the batch, $\$_{t,\theta,j}$ is the cross entropy data loss for jth training image in tth iteration, $R_{t,\theta}$ is the regularization loss in tth iteration, and σ is a regularization loss hyper-parameter. The cross-entropy data loss $\$_{t,\theta,j}$ for jth training image is computed as

$$\$_{t,\theta,j} = -\log\left(\frac{e^{S_{o_j}}}{\sum_{k=1}^{N_c} e^{S_k}}\right) \tag{4}$$

where N_c is the total number of classes in the data set, o_j is the actual class (i.e., ground truth class) for jth training image, and S_k is the computed class score for kth class for jth training image. The regularization loss $R_{t,\theta}$ is computed as

$$R_{t,\theta} = \sum_{i=1}^{d} (\theta_{t,i})^2. \tag{5}$$

In SGDM [32], the gradient in each dimension is incorporated to gain moment for the parameters having consistent gradient, as follows:

$$m_{t,i} = \beta m_{t-1,i} + g_{t,i} \tag{6}$$

$$\theta_{t+1,i} = \theta_{t,i} - \alpha m_{t,i} \tag{7}$$

where $m_{t,i}$ is the moment gained at tth iteration for ith parameter $\theta_{t,i}$ with $m_{t,i} = 0$ for $t = 0$ and β is a hyper-parameter to control the moment.

In AdaGrad [33], the basic SGD approach is modified by normalizing the learning rate α_t as

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\alpha_t \times g_{t,i}}{\sqrt{G_{t,i} + \epsilon}} \tag{8}$$

where ϵ is a small value to avoid division by zero and $G_{t,i}$ is the sum of the squares of the gradients of t steps for the ith parameter and given as

$$G_{t,i} = \sum_{t=1}^{t} (g_{t,i})^2 \tag{9}$$

where $g_{t,i}$ is given by (2). Over the iterations, the value of $G_{t,i}$ may become very large due to the positive accumulation of the square of the gradients and may decrease the effective learning rate α drastically, which in turn can cut down the learning process. This problem has been addressed in AdaDelta [36] and RMSProp [37] by leaking the accumulated square of gradients $G_{t,i}$ with a decay rate β. The $G_{t,i}$ in RMSProp is modified as

$$G_{t,i} = \beta G_{t-1,i} + (1-\beta)(g_{t,i})^2 \tag{10}$$

where $G_{t-1,i} = 0$ for $t = 1$.

Adam [38] is another widely used gradient descent optimization technique that computes the learning rates at each step based on two vectors known as the first- and second-order moments (i.e., mean and variance, respectively), which are recursively defined using the gradient and the square of the gradient, respectively. Basically, Adam is an improvement over RMSProp by incorporating first moment with RMSProp. Here, the first- and second-order moments are defined as

$$m_{t,i} = \beta_1 m_{t-1,i} + (1-\beta_1) g_{t,i} \tag{11}$$

$$v_{t,i} = \beta_2 v_{t-1,i} + (1-\beta_2) g_{t,i}^2 \tag{12}$$

where β_1 and β_2 are the decay rates for first and second moments, respectively, $m_{t-1,i}$ and $v_{t-1,i}$ are the mean and variance of the gradient of the previous steps, respectively. Both $m_{t-1,i}$ and $v_{t-1,i}$ are initialized with 0 at the first iteration, $t = 1$. It is observed that, initially, the value of the first moment is small and the value of the second moment is very small, thus leading to a very large step size. In order to solve this issue, Adam has incorporated a bias correction of the first and second-order moments as

$$\hat{m}_{t,i} = \frac{m_{t,i}}{\left(1-\beta_1^t\right)} \quad \text{and} \quad \hat{v}_{t,i} = \frac{v_{t,i}}{\left(1-\beta_2^t\right)} \tag{13}$$

where β_1^t is β_1 power t, β_2^t is β_2 power t, and $\hat{m}_{t,i}$ and $\hat{v}_{t,i}$ are the biased corrected first and second moments, respectively. Thus, the parameter update in Adam is incorporated as

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\alpha_t \times \hat{m}_{t,i}}{\sqrt{\hat{v}_{t,i}+\epsilon}}. \tag{14}$$

A problem arises in Adam when the value of the second moment decreases significantly. Due to low values of the second moment, the friction in the optimization landscape decreases, which leads to the situation where the update process overshoots an optimum solution due to a high learning rate and diverges. This problem is addressed in AMSGrad [40] by considering the maximum of second moments in current and past iterations. In AMSGrad, the first- and second-order moments are computed and bias-corrected similar to Adam. However, AMSGrad normalizes the learning rate α_t by the maximum of all $\hat{v}_{t,i}^{\max}$ values, instead of only $\hat{v}_{t,i}$. AMSGrad memorizes the highest of the second-order moment to give more priority to those steps which update the parameter in a more accurate direction. Thus, in AMSGrad, $\hat{v}_{t,i}^{\max}$ is defined as

$$\hat{v}_{t,i}^{\max} = \max\left(\hat{v}_{t-1,i}^{\max}, \hat{v}_{t,i}\right) \tag{15}$$

where max is the maximum operator and $\hat{v}_{t-1,i}^{\max} = 0$ for $t = 1$. Thus, the parameter update in AMSGrad is carried out using the following update rule:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\alpha_t \times \hat{m}_{t,i}}{\sqrt{\hat{v}_{t,i}^{\max}+\epsilon}}. \tag{16}$$

In practice, Adam is popular in various problems related to deep learning. Adam with the first-order moment hyper-parameter $\beta_1 = 0.9$, second-order moment hyper-parameter $\beta_2 = 0.999$, and learning rate $\alpha \in [10^{-2}, 10^{-4}]$ is a good starting choice for many models [38].

III. Proposed diffGrad Optimization

From the discussions in Section II, we can conclude that recent optimization techniques such as Adam and AMSGrad suffer from the problem of automatic adjustment of the learning rate. The main problem is with controlling friction for the first moment in order to avoid overshooting near an optimum solution. In this section, we propose a new gradient descent optimization technique referred to as diffGrad to address these issues of existing gradient descent optimization techniques. The proposed diffGrad optimization technique is based on the change in short-term gradients and controls the learning rate based on the need for dynamic adjustment of learning rate. This means that diffGrad follows the norm that the parameter update should be smaller in low gradient changing regions and vice versa. diffGrad computes the first- and second-order moments (i.e., $m_{t,i}$ and $v_{t,i}$, respectively) as well as the first- and second-order bias-corrected moments (i.e., $\hat{m}_{t,i}$ and $\hat{v}_{t,i}$, respectively) for the ith parameter at the tth iteration similar to Adam [38] using (11)–(13).

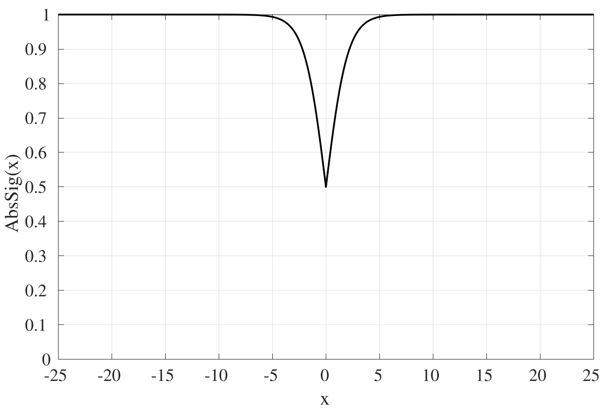


Fig. 2. Behavior of DFC in terms of the AbsSig(x) values. Here a DFC value of 1 means no friction and 0 means infinite friction. Note that the friction is roughly negligible if the difference in gradient is high. Here, negligible represents DFC ≈ 1 which means there is no friction, and in this case, diffGrad is the same as Adam. Moreover, the minimum DFC is 0.5 when there is no gradient change. Otherwise, the DFC, close to zero, will slow down the learning process heavily.

A diffGrad friction coefficient (DFC) is introduced in the proposed work to control the learning rate using information of the short-term gradient behavior. The DFC is represented by ξ and defined as

$$\xi_{t,i} = \text{AbsSig}(\Delta g_{t,i}) \tag{17}$$

where AbsSig is a nonlinear sigmoid function that squashes every value between 0.5 and 1, and is defined as

$$\text{AbsSig}(x) = \frac{1}{1+e^{-|x|}} \tag{18}$$

while $\Delta g_{t,i}$ is the change in gradient between immediate past and current iterations, given as

$$\Delta g_{t,i} = g_{t-1,i} - g_{t,i} \tag{19}$$

where $g_{t,i}$ is the computed gradient for the ith parameter at the tth iteration and defined in (2).

The behavior of DFC (i.e., ξ) is characterized in Fig. 2 in terms of AbsSig(x) which represents the friction with respect to the change in gradient. It can be observed from Fig. 2 that large changes in the gradient incur less friction, whereas small changes in the gradient incur more friction with at most 0.5 when there is no change in the gradient. Note that $|\Delta g_{t,i}| \in \mathbb{R}_0^+$, $\forall i \in [1, d]$, and $i \in \mathbb{I}^+$ lead to $\xi_{t,i} \in [0.5, 1]$ at any iteration t. The DFC imposes more friction when the gradient changes slowly and vice versa.

In the proposed diffGrad optimization method, the steps up to the computation of bias-corrected first-order moment $\hat{m}_{t,i}$ and bias-corrected second-order moment $\hat{v}_{t,i}$ are the same as

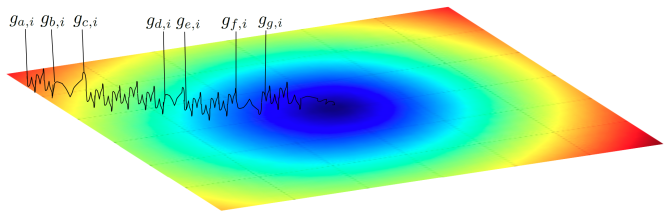

$g_{a,i}$ $g_{b,i}$ $g_{c,i}$ $g_{d,i}$ $g_{e,i}$ $g_{f,i}$ $g_{g,i}$

Fig. 3. Illustration of optimization landscape in order to understand the importance of short-term gradient change. The dark blue area represents the optimization goal to be reached.

those of Adam optimization [38]. The diffGrad optimization method updates the ith parameter at the tth iteration using the following update rule:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\alpha_t \times \xi_{t,i} \times \hat{m}_{t,i}}{\sqrt{\hat{v}_{t,i}} + \epsilon} \tag{20}$$

where $\theta_{t,i}$ is the weight for ith parameter at tth iteration, α_t is the learning rate for tth iteration, ϵ is a very small value of approximately 10^{-7} added to avoid division by zero, $\xi_{t,i}$ is the diffGrad friction coefficient, and $\hat{m}_{t,i}$ and $\hat{v}_{t,i}$ are the bias-corrected first-order moment and the bias-corrected second-order moment, respectively, as defined in (13).

The proposed optimizer introduces the DFC to control the oscillations. The difference in gradient reduces the learning rate by controlling the moving average near an optimum solution. The necessity of the friction coefficient is illustrated using an optimization landscape in Fig. 3. The dark blue area shows the optimization goal. As we move from dark red to dark blue, the depth of the optima tends to be reduced. Lighter shades between two colors show steep descent toward the optimization goal. Here, $g_{a,i} \rightarrow g_{g,i}$ shows the gradients at steps a to g. The color between $g_{a,i}$ and $g_{b,i}$ changes very rapidly. As a result, the learning rate increases due to the increase in the diffGrad friction coefficient. Hence, the frequency of vertical fluctuations also increases. There is a very steep descent between the pairs $g_{c,i} \rightarrow g_{d,i}$ and $g_{e,i} \rightarrow g_{f,i}$. Hence, we again see the frequent fluctuations in the vertical direction. The descents between $g_{b,i} \rightarrow g_{c,i}$ and $g_{f,i} \rightarrow g_{g,i}$ are very slow, which is evident from slow or no change in color. Here, the DFC reduces the frequency of fluctuations by decreasing the learning rate of the network. Hence, the introduction of DFC reduces redundant learning and increases the rate of convergence. It also helps in finding an optimum solution by reducing the vertical fluctuation near local optima.

The problem of an ever decreasing learning rate in Adagrad [33] has been solved by Adam [38] by introducing two moments on a gradient. The adaptive nature of these two moments is controlled during learning with respect to the slope of the descent of the probable solution toward a local optimum. However, a sudden decrease in the second moment adversely affects the Adam optimizer and the actual solution overshoots the local optima. This problem has been resolved in AMSGrad [40] by introducing a new parameter, $\hat{v}_{t,i}^{\max} = \max(\hat{v}_{t-1,i}^{\max}, \hat{v}_{t,i})$, which prevents the learning rate from overshooting. Adam and AMSgrad ignore the effect of the first moment in controlling the learning rate over the optimization landscape. The proposed diffGrad optimizer resolves this drawback by introducing a new parameter DFC (ξ), which not only allows the high learning rate for a high gradient changing surface but also reduces the learning rate for a low gradient changing surface and prevents the probable solution from overshooting. The improved performance of diffGrad shows that the optimizer effectively introduces the required friction ($\uparrow$ or $\downarrow$). Please note that the minimum DFC is considered as 0.5, as shown in Fig. 2, such that the optimization should not become stuck in local optima and saddle regions. Due to the high moment gained, a sufficient magnitude of the step size will be allowed by the DFC in the proposed diffGrad approach, so that it emerges from flat local optima and flat saddle regions.

IV. Convergence Analysis

The convergence property of Adam [38] is shown using the online learning framework proposed in [44]. We also use this technique to analyze the convergence of the proposed diffGrad optimizer. Consider $f_1(\theta), f_2(\theta), \ldots, f_T(\theta)$ as the unknown sequence of convex cost functions. Our aim is to predict parameter θ_t at each iteration t and evaluate over $f_t(\theta)$. For this type of problem, where the nature of the sequence is not known *a priori*, the algorithm can be evaluated based on the regret bound. The regret bound is computed by summing the difference between all the previous online guesses $f_t(\theta_t)$ and the best fixed point parameter $f_t(\theta^*)$ from a feasible set χ for all the previous iterations. Mathematically, the regret bound is given as

$$R(T) = \sum_{t=1}^{T} [f_t(\theta_t) - f_t(\theta^*)] \tag{21}$$

where $\theta^* = \arg\min_{\theta \in \chi} \sum_{t=1}^{T} f_t(\theta)$. We observe that diffGrad has an $O(\sqrt{T})$ regret bound. The proof is given in the Appendix. Our regret bound is comparable to general convex online learning methods. We have used the following definitions: $g_{t,i}$ refers to the gradient in the tth iteration for the ith element, $g_{1:t,i} = [g_{1,i}, g_{2,i}, \ldots, g_{t,i}] \in \mathbb{R}^t$ is the vector of gradients in the ith dimension over all iterations up to t, and $\gamma \triangleq (\beta_1^2/\sqrt{\beta_2})$.

Theorem 1: Consider the bounded gradients for function f_t (i.e., $||g_{t,\theta}||_2 \leq G$ and $||g_{t,\theta}||_\infty \leq G_\infty$) for all $\theta \in R^d$. Also, assume that diffGrad produces the bounded distance between any θ_t (i.e., $||\theta_n - \theta_m||_2 \leq D$ and $||\theta_n - \theta_m||_\infty \leq D_\infty$ for any $m, n \in \{1, \ldots, T\}$). Let $\gamma \triangleq (\beta_1^2/\sqrt{\beta_2})$, $\beta_1, \beta_2 \in [0, 1)$ satisfy $(\beta_1^2/\sqrt{\beta_2}) < 1$, $\alpha_t = (\alpha/\sqrt{t})$, and $\beta_{1,t} = \beta_1 \lambda^{t-1}$, $\lambda \in (0, 1)$ where λ is typically very close to 1, e.g., $1 - 10^{-8}$. For all $T \geq 1$, the proposed diffGrad optimizer shows the following guarantee:

$$\begin{aligned} R(T) \leq\ & \frac{D^2}{2\alpha(1-\beta_1)} \sum_{i=1}^{d} (1 + e^{-|g_{1,i}|}) \sqrt{T \hat{v}_{T,i}} \\ & + \frac{\alpha(1+\beta_1) G_\infty}{(1-\beta_1)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d} ||g_{1:T,i}||_2 \\ & + \sum_{i=1}^{d} \frac{D_\infty^2 G_\infty \sqrt{1-\beta_2}}{2\alpha(1-\beta_1)(1-\lambda)^2}. \end{aligned} \tag{22}$$

Note that the additive term over the dimension (d) can be much smaller than its upper bound $\sum_{i=1}^{d} ||g_{1:T,i}||_2 \ll dG_\infty\sqrt{T}$ and $\sum_{i=1}^{d} (1+e^{-|g_{1,i}|})(T\hat{v}_{T,i})^{1/2} \ll d(1+E_\infty)G_\infty\sqrt{T}$, where E_∞ is the upper bound over the exponential function and $E_\infty \gg \sum_{i=1}^{d} e^{-|g_{1,i}|}$. In general, $O(\log d\sqrt{T})$ is achieved by adaptive methods such as diffGrad and Adam which is improved over the $O(\sqrt{dT})$ of nonadaptive methods. The proposed diffGrad method also uses the decay of $\beta_{1,t}$ for the theoretical analysis, similar to Adam.

Finally, by using the above theorem and $\sum_{i=1}^{d} ||g_{1:T,i}||_2 \ll dG_\infty\sqrt{T}$ and $\sum_{i=1}^{d} (1+e^{-|g_{1,i}|})(T\hat{v}_{T,i})^{1/2} \ll d(1+E_\infty)G_\infty\sqrt{T}$, the convergence of average regret of diffGrad can be shown as described in the following corollary.

Corollary 1: Consider the bounded gradients for function f_t (i.e., $||g_{t,\theta}||_2 \le G$ and $||g_{t,\theta}||_\infty \le G_\infty$) for all $\theta \in R^d$. Also, assume that diffGrad produces the bounded distance between any θ_t (i.e., $||\theta_n - \theta_m||_2 \le D$ and $||\theta_n - \theta_m||_\infty \le D_\infty$ for any $m, n \in \{1, \ldots, T\}$). For all $T \ge 1$, the proposed diffGrad optimizer shows the following guarantee:

$$\frac{R(T)}{T} = O\left(\frac{1}{\sqrt{T}}\right). \tag{23}$$

Thus, $\lim_{T\to\infty}(R(T)/T) = 0$.

V. Empirical Analysis

In order to justify the purpose of introducing the difference of gradient-based friction in diffGrad optimization, we have conducted an empirical analysis in this section. We have modeled the optimization problem as a regression problem over three 1-D nonconvex functions. We have performed the optimization over these functions by using the widely used Adam [38] and the proposed diffGrad optimization methods.

The following nonconvex functions, represented by $F1$, $F2$, and $F3$, respectively, are used for this empirical analysis:

$$F1(x) = \begin{cases} (x+0.3)^2, & \text{for } x \le 0 \\ (x-0.2)^2 + 0.05, & \text{for } x > 0 \end{cases} \tag{24}$$

$$F2(x) = \begin{cases} -40x - 35.15, & \text{for } x \le -0.9 \\ x^3 + x\sin(8x) + 0.85, & \text{for } x > -0.9 \end{cases} \tag{25}$$

$$F3(x) = \begin{cases} x^2, & \text{for } x \le -0.5 \\ 0.75 + x, & \text{for } -0.5 < x \le -0.4 \\ -7x/8, & \text{for } -0.4 < x \le 0 \\ 7x/8, & \text{for } 0 < x \le 0.4 \\ 0.75 - x, & \text{for } 0.4 < x \le 0.5 \\ x^2, & \text{for } 0.5 < x \end{cases} \tag{26}$$

where x is the input for this function with $-\infty < x < +\infty$. Functions $F1$, $F2$, and $F3$ are shown in Fig. 4(a)–(c), respectively, for $-1 < x < +1$.

It can be observed that function $F1$ has one global minimum and one local minimum, whereas functions $F2$ and $F3$ have one global minimum and two local minima. In this experiment, for both Adam [38] and the proposed diffGrad optimization methods, the following are the hyper-parameter settings: the decay rate for the first moment (β_1) is 0.95; the decay rate for the second moment (β_2) is 0.999; the learning rate (η) is 0.1 for each iteration; both the first-order moment (m) and second-order moment (v) are initialized to 0; and the parameter θ is initialized to -1 in order to show the advantages of the proposed method. In the proposed diffGrad optimization method, the previous gradient value at the first iteration (i.e., g_0) is set to zero. We run Adam [38] and the proposed diffGrad optimization for 300 iterations for all functions. The regression loss, as well as parameter value θ, are recorded for both optimization methods at each iteration and analyzed.

Fig. 4(b) and (c) depicts the regression loss and the parameter value (θ), respectively, after each iteration for function $F1$. It is discovered from these plots that Adam overshoots the global minimum due to the high moment gained so far. It can be observed in Fig. 4(c) that Adam overshoots the global minimum at $\theta = -0.3$ and becomes stuck in the local minimum at $\theta = 0.2$. This problem is addressed by DFC of the proposed diffGrad optimization method, which controls the momentum while reaching toward the global minimum and the overshoot does not occur. Moreover, diffGrad is able to reach a zero loss, as opposed to Adam, which saturates with reasonable loss. The same behavior is also observed for the function $F2$, as shown in Fig. 4(e) and (f).

Both Adam and diffGrad are able to achieve the global minimum for function $F3$, as shown in Fig. 4(h) and (i). Note that one local minimum is present before the global minimum in function $F3$ [see Fig. 4(g)]. Both Adam and diffGrad accumulate enough momentum to cross the local minimum. It can be seen in Fig. 4(i) that Adam oscillates with higher frequency and amplitude around the global minimum, as compared to diffGrad. Thus, better stability is obtained by diffGrad.

VI. Experimental Setup for Classification

This section presents the setup and settings used in the experiments in terms of the deep architecture used, the hyper-parameter setting, and the applied data set.

A. Deep Architecture Used

The experiments are conducted for an image categorization problem. CNN is generally used for processing the images. The popular CNN architectures for image categorization problems are AlexNet [2], VGGNet [4], GoogleNet [19], and ResNet [20]. The ResNet-based CNN architecture introduced by He *et al.* [20] is one of the most accurate models which won the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [45] in 2015. The ResNet architecture is made with the residual unit. A residual unit considers the learning module as the residual of output with respect to the input by implementing a direct connection between input and output of that unit. The residual unit facilitates the training of deeper ResNet architecture which was not feasible in the earlier CNN architectures. The PyTorch implementation of ResNet, publicly available through GitHub,[1] is used for our experiments. The depth of ResNet is considered as 50 in this article. Following are the details of ResNet50 architecture for the CIFAR

[1]https://github.com/kuangliu/pytorch-cifar

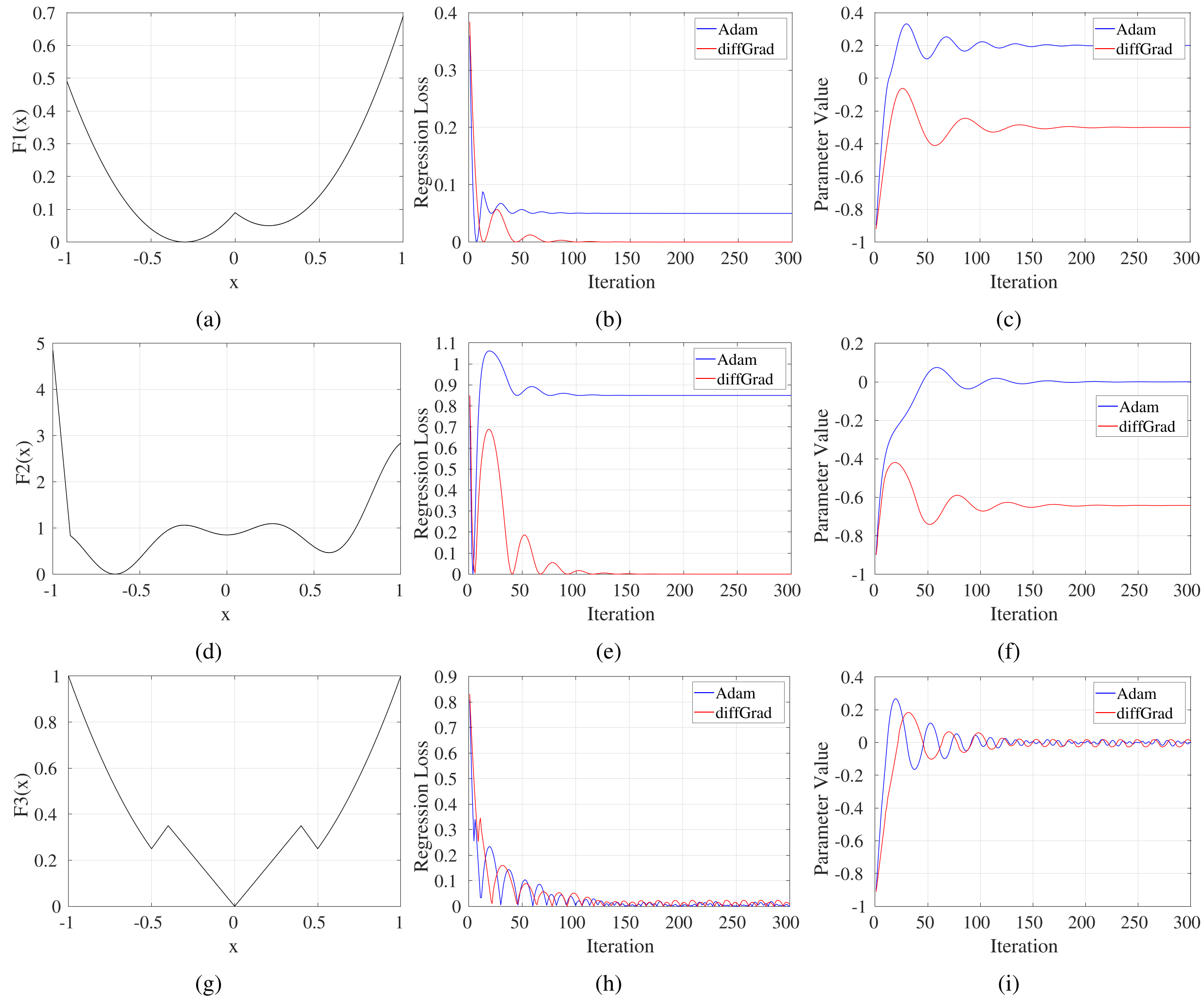


Fig. 4. Comparison of empirical results of the Adam [38] and the proposed diffGrad optimization techniques over three synthetic nonconvex functions. (a), (d), and (g) Non-convex synthetic functions $F1$, $F2$, and $F3$, respectively. (b), (e), and (h) Regression loss over functions $F1$, $F2$, and $F3$, respectively, at each iteration. (c), (f), and (i) Parameter value θ after each iteration during optimization of functions $F1$, $F2$, and $F3$, respectively.

data sets: Conv: {3 × 3, 64}, BatchNorm, Bottleneck (three times): {Conv: [1 × 1, 64], BatchNorm, Conv: [3 × 3, 64], BatchNorm: [1 × 1, 256], BatchNorm}, Bottleneck (four times): {Conv: [1 × 1, 128], BatchNorm, Conv: [3 × 3, 128], BatchNorm, Conv: [1 × 1, 512], BatchNorm}, Bottleneck (six times): {Conv: [1 × 1, 256], BatchNorm, Conv: [3 × 3, 256], BatchNorm, Conv: [1 × 1, 1024], BatchNorm}, Bottleneck (three times): {Conv: [1 × 1, 512], BatchNorm, Conv: [3 × 3, 512], BatchNorm, Conv: [1 × 1, 2048], BatchNorm}, AveragePooling: {4×4}. For more information regarding the used ResNet50 architecture, the above-mentioned GitHub page may be consulted.

B. Hyper-Parameter Setting

The hyper-parameter settings in the experiments are illustrated in this section. SGDM is used as the optimization technique. The batch sizes of 32, 64, and 128 are opted. The number of epochs is 100 with a learning rate of 10^{-3} for the first 80 epochs and 10^{-4} for the last 20 epochs. For all the optimizers, the default settings of PyTorch are used, excluding the moment coefficient for SGDM, which is set to 0.9.

C. Data Set Used

In order to conduct the image categorization experiments, the CIFAR10 and CIFAR100 data sets[2] [46] are used in this article. Both the CIFAR10 and the CIFAR100 data sets consist of the same 60 000 images, including 50 000 images for training and 10 000 images for validation. In the CIFAR10 data set, all images are divided into 10 categories including "airplane," "automobile," "bird," "cat," "deer," "dog," "frog," "horse," "ship," and "truck." On the other hand, in the CIFAR100 data set, the same set of images is partitioned into 100 categories. The dimension of all images is 32×32×3. Note that the images are pre-processed to make the RGB values zero-centered with unit standard deviation across each color channel. The data augmentation is done only over the training images by the process of flipping and cropping. The images are randomly flipped horizontally (i.e., with respect to the vertical axis) with a probability of 0.5. In the case of cropping, at first the images are scaled to 40×40×3 size by zero padding. Then, the images of size 32 × 32 × 3 are cropped randomly from the upscaled images.

[2]https://www.cs.toronto.edu/ kriz/cifar.html

TABLE I

COMPARISON RESULTS IN TERMS OF THE "VALIDATION CLASSIFICATION ACCURACY" OVER THE CIFAR10 AND CIFAR100 DATABASES AMONG SGDM, ADAGRAD, ADADELTA, RMSPROP, AMSGRAD, ADAM, AND PROPOSED DIFFGRAD OPTIMIZATION METHODS. COMPARISON IS MADE FOR THE BATCH SIZES (N_b) OF 32, 64, AND 128. BEST RESULTS AMONG DIFFERENT OPTIMIZATION TECHNIQUES ARE HIGHLIGHTED IN BOLD

Optimizer	CIFAR10 Database			CIFAR100 Database		
	N_b=32	N_b=64	N_b=128	N_b=32	N_b=64	N_b=128
SGDM	92.95	92.07	90.15	73.3	70.37	67.25
AdaGrad	92.04	92.01	91.3	71.01	70.8	68.38
AdaDelta	93.66	93.48	93.54	73.46	74.09	74.12
RMSProp	92.82	92.32	92.26	65.31	66.8	62.65
AMSGrad	93.73	93.4	93.51	73.06	72.45	72.86
Adam	93.78	93.81	93.72	71.82	73.31	73.72
diffGrad	**93.9**	**94.05**	**94.08**	**75.4**	**75.27**	**75.05**

TABLE II

COMPARISON RESULTS IN TERMS OF THE "VALIDATION CLASSIFICATION ACCURACY" OVER THE CIFAR10 AND CIFAR100 DATABASES AMONG DIFFERENT VARIANTS OF DIFFGRAD. COMPARISON IS MADE FOR THE BATCH SIZES (N_b) OF 32, 64, AND 128. BEST RESULTS AMONG DIFFERENT OPTIMIZATION TECHNIQUES ARE HIGHLIGHTED IN BOLD

Optimizer	CIFAR10 Database			CIFAR100 Database		
	N_b=32	N_b=64	N_b=128	N_b=32	N_b=64	N_b=128
diffGrad	93.9	94.05	94.08	75.4	75.27	75.05
diffGrad1	**94.37**	94.08	94.15	75.31	75.24	75.05
diffGrad2	94.17	94.18	94	**75.55**	75.51	75.06
diffGrad3	94.27	94.08	**94.19**	74.32	75.13	**75.45**
diffGrad4	94.04	**94.33**	93.9	74.83	74.82	75.25
diffGrad5	93.99	93.97	93.88	74.95	**75.55**	74.86

TABLE III

PERFORMANCE OF THE PROPOSED DIFFGRAD OPTIMIZATION TECHNIQUE WITH RESNET50 MODEL FOR DIFFERENT ACTIVATION FUNCTIONS, NAMELY RELU, LRELU, ELU, AND SELU. THE VALIDATION CLASSIFICATION ACCURACY OVER CIFAR10 IS REPORTED FOR THE BATCH SIZES (N_b) OF 32, 64, AND 128. THE BEST RESULTS AMONG DIFFERENT ACTIVATION'S ARE HIGHLIGHTED IN BOLD

Optimizer	CIFAR10 Database		
	N_b=32	N_b=64	N_b=128
ResNet50(ReLU)	93.9	94.05	94.08
ResNet50(LReLU)	**94.31**	**94.28**	**94.16**
ResNet50(ELU)	94.16	94.16	94.1
ResNet50(SELU)	93.48	93.34	93.38

VII. CLASSIFICATION EXPERIMENTS AND ANALYSIS

The image categorization experiments over the CIFAR10 and CIFAR100 data sets are conducted to test the performance improvement of the proposed diffGrad gradient descent optimization method. The ResNet50 is used to demonstrate the suitability of diffGrad for the CNN model. In this section, at first the results of the proposed diffGrad method are presented, then the results are compared with other state-of-the-art optimization methods, and finally, the stability of the proposed diffGrad optimization is tested over different activation functions. The results are computed in terms of the average top-1 validation classification accuracy.

A. Validation Results Comparison

The validation classification accuracy due to the diffGrad method is compared with state-of-the-art optimization techniques such as SGDM [32], AdaGrad [33], AdaDelta [36], RMSProp [37], AMSGrad [40], and ADAM [38]. Table I depicts the validation classification accuracy for different optimization techniques. The results are presented over the CIFAR10 and CIFAR100 data sets. The best results among the different optimization techniques are highlighted in bold. It is observed from Table I that the proposed diffGrad optimization technique outperforms all other optimization techniques over both the CIFAR10 and the CIFAR100 data sets for all tested batch sizes of 32, 64, and 128, respectively. The proposed diffGrad method utilizes all the positive characteristics of Adam. Moreover, the effect of the proposed difference of the gradient-based friction technique prevents the network from noisy oscillation near the minimum solution. It leads to more accurate results as compared to other optimization techniques.

B. Experiments With diffGrad Variants

In this section, we modify the DFC of diffGrad and analyze the performance over the CIFAR10 data set. The DFC is basically given as DFC= $1/(1+e^{-(|g_{t-1}-g_t|)})$ with the range DFC $\in [0.5, 1]$. We modify the DFC by removing the absolute value, which is given as DFC1= $1/(1+e^{-(g_{t-1}-g_t)})$. Note that DFC1 $\in [0, 1]$. We also generate another version called DFC2, given by DFC2 $= 9/(1+e^{-(0.5|g_{t-1}-g_t|)}-4)$ with the range DFC2 $\in [0.5, 5]$. We even use the mean (μ) and standard deviation (ν) of absolute gradients of the batch with DFC and consider the following three more scenarios: 1) DFC3 $= 1/(1+e^{-(\nu|g_{t-1}-g_t|-\mu)})$ with the range DFC3 $\in [0.5, 1]$; 2) DFC4 $= 1/(1+e^{-(\nu^2|g_{t-1}-g_t|-\mu)})$ with the range DFC4 $\in [0.5, 1]$; and 3) DFC5 $= 1/(1+e^{-(\sqrt{\nu}|g_{t-1}-g_t|-\mu)})$ with the range DFC5 $\in [0.5, 1]$. We define numbered diffGrad accordingly, i.e., diffGrad1 using DFC1. The validation classification accuracy for diffGrad using DFC, diffGrad1 using DFC1, diffGrad2 using DFC2, diffGrad3 using DFC3, diffGrad4 using DFC4, and diffGrad5 using DFC5 are presented in Table II. The results are computed over the CIFAR10 and CIFAR100 data sets with batch sizes 32, 64, and 128. It can be noticed from this result that diffGrad3 performs better in both data sets for higher batch size. Moreover, for the higher batch sizes (64 and 128), the best results are shown by the mean and standard deviation-based diffGrad variants. It is due to the fact that the mean and standard deviation of the absolute gradient tends to be more accurate for higher batch size. From Tables I and II, it is clear that all the diffGrad variants outperform the state-of-the-art optimization techniques.

C. Performance Analysis With Activation Functions

In other experiments, the Rectified Linear Unit (ReLU) [2] is used as the default activation function in the framework of ResNet [20]. In this experiment, we have considered four activation functions, namely ReLU [2], Leaky ReLU (LReLU) [47], Exponential Linear Unit (ELU) [48], and Scaled ELU (SELU) [49]. The performance of the proposed diffGrad optimization method is computed over the CIFAR10 data set for different activation functions. We have used the same experimental setup as for the ResNet50 model by replacing all the activation functions with ReLU, LReLU, ELU, and SELU, one by one. The rest of the experimental setup is same as the setup of the previous experiments. The leaky factor in LReLU is considered as 0.1. The validation classification accuracy is compared in Table III. It is clear from this result that the LReLU activation is better suited to the proposed diffGrad optimization. The ELU performs better than ReLU over CIFAR10 with ResNet50 using the proposed method.

VIII. Conclusion

In this article, a new SGD optimization method diffGrad is proposed. diffGrad incorporates the difference of gradients of current and immediate past iteration (i.e., short term gradient change information) with Adam optimization techniques to control the learning rate based on the optimization stage. The proposed diffGrad allows a high learning rate if the gradient change is more (i.e., the optimization is far from the optimum solution), and a low learning rate if the gradient changes minimally (i.e., the optimization is near to the optimum solution). The local optima and saddle point scenarios are handled by the moment gained due to past gradients. The regret bound analysis provides a guarantee of convergence. An empirical analysis over three synthetic, nonconvex functions reveals that the proposed diffGrad optimization method controls the update step in order to avoid the overshooting of global minimum and oscillation around global minimum. The proposed diffGrad optimization method is also tested with the ResNet50 model for an image categorization task over the CIFAR10 and CIFAR100 data sets. The results are compared with the state-of-the-art SGD optimization techniques such as SGDM, AdaGrad, AdaDelta, RMSProp, AMSGrad, and Adam. It is observed that the diffGrad outperforms all other optimizers. Moreover, for large batch sizes, the mean and standard deviation of the absolute gradient of a batch can be used with diffGrad for better results. The LReLU is better suited with the proposed diffGrad.

Appendix

A. Convergence Proof

Theorem 2: Let the bounded gradients for function f_t (i.e., $||g_{t,\theta}||_2 \leq G$ and $||g_{t,\theta}||_\infty \leq G_\infty$) for all $\theta \in R^d$. Also, assume that diffGrad produces the bounded distance between any θ_t (i.e., $||\theta_n - \theta_m||_2 \leq D$ and $||\theta_n - \theta_m||_\infty \leq D_\infty$ for any $m, n \in \{1, \ldots, T\}$). Let $\gamma \triangleq (\beta_1^2/\sqrt{\beta_2})$, $\beta_1, \beta_2 \in [0, 1)$ satisfy $(\beta_1^2/\sqrt{\beta_2}) < 1$, $\alpha_t = (\alpha/\sqrt{t})$, and $\beta_{1,t} = \beta_1 \lambda^{t-1}$, $\lambda \in (0, 1)$ with λ is typically close to 1, e.g., $1 - 10^{-8}$. For all $T \geq 1$, the proposed diffGrad optimizer shows the following guarantee:

$$R(T) \leq \frac{D^2}{2\alpha(1-\beta_1)} \sum_{i=1}^{d} (1 + e^{-|g_{1,i}|}) \sqrt{T \hat{v}_{T,i}} + \frac{\alpha(1+\beta_1)G_\infty}{(1-\beta_1)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d} ||g_{1:T,i}||_2 + \sum_{i=1}^{d} \frac{D_\infty^2 G_\infty \sqrt{1-\beta_2}}{2\alpha(1-\beta_1)(1-\lambda)^2}. \quad (27)$$

Proof: Using Lemma 10.2 of Adam [38], we can write as

$$f_t(\theta_t) - f_t(\theta^*) \leq g_t^T(\theta_t - \theta^*) = \sum_{i=1}^{d} g_{t,i}(\theta_{t,i} - \theta_{,i}^*).$$

We can write the following from the diffGrad update rule described in (20), ignoring $\epsilon 0$:

$$\theta_{t+1} = \theta_t - \frac{\alpha_t \xi_t \hat{m}_t}{\sqrt{\hat{v}_t}} = \theta_t - \frac{\alpha_t \xi_t}{(1-\beta_1^t)} \left(\frac{\beta_{1,t}}{\sqrt{\hat{v}_t}} m_{t-1} + \frac{(1-\beta_{1,t})}{\sqrt{\hat{v}_t}} g_t \right) \quad (28)$$

where $\beta_{1,t}$ is the first-order moment at tth iteration and β_1^t is the tth power of the initial first-order moment.

For ith dimension of parameter vector $\theta_t \in R^d$, we can write

$$(\theta_{t+1,i} - \theta_{,i}^*)^2 = (\theta_{t,i} - \theta_{,i}^*)^2 - \frac{2\alpha_t \xi_{t,i}}{1-\beta_1^t} \times \left(\frac{\beta_{1,t}}{\sqrt{\hat{v}_{t,i}}} m_{t-1,i} + \frac{(1-\beta_{1,t})}{\sqrt{\hat{v}_{t,i}}} g_{t,i} \right) \times (\theta_{t,i} - \theta_{,i}^*) + \alpha_t^2 \xi_{t,i}^2 \left(\frac{\hat{m}_{t,i}}{\hat{v}_{t,i}} \right)^2. \quad (29)$$

The above equation can be reordered as

$$g_{t,i}(\theta_{t,i} - \theta_{,i}^*) = \frac{(1-\beta_1^t)\sqrt{\hat{v}_{t,i}}}{2\alpha_t \xi_{t,i}(1-\beta_{1,t})} \left((\theta_{t,i} - \theta_{,i}^*)^2 - (\theta_{t+1,i} - \theta_{,i}^*)^2 \right) + \frac{\beta_{1,t}}{1-\beta_{1,t}} (\theta_{,i}^* - \theta_{t,i}) m_{t-1,i} + \frac{\alpha_t (1-\beta_1^t) \xi_{t,i}}{2(1-\beta_{1,t})} \frac{(\hat{m}_{t,i})^2}{\sqrt{\hat{v}_{t,i}}}. \quad (30)$$

Further, it can be written as

$$g_{t,i}(\theta_{t,i} - \theta_{,i}^*) = \frac{(1-\beta_1^t)\sqrt{\hat{v}_{t,i}}}{2\alpha_t \xi_{t,i}(1-\beta_{1,t})} \left((\theta_{t,i} - \theta_{,i}^*)^2 - (\theta_{t+1,i} - \theta_{,i}^*)^2 \right) + \sqrt{\frac{\beta_{1,t}}{\alpha_{t-1}(1-\beta_{1,t})} (\theta_{,i}^* - \theta_{t,i})^2 \sqrt{\hat{v}_{t-1,i}}} \times \sqrt{\frac{\beta_{1,t}\alpha_{t-1}(m_{t-1,i})^2}{(1-\beta_{1,t})\sqrt{\hat{v}_{t-1,i}}}} + \frac{\alpha_t (1-\beta_1^t) \xi_{t,i}}{2(1-\beta_{1,t})} \frac{(\hat{m}_{t,i})^2}{\sqrt{\hat{v}_{t,i}}}. \quad (31)$$

Based on Young's inequality, $ab \leq a^2/2 + b^2/2$ and fact that $\beta_{1,t} \leq \beta_1$, the above equation can be reordered as

$$g_{t,i}(\theta_{t,i} - \theta_{,i}^*) \leq \frac{1}{2\alpha_t \xi_{t,i}(1-\beta_1)} \times \left((\theta_{t,i} - \theta_{,i}^*)^2 - (\theta_{t+1,i} - \theta_{,i}^*)^2 \right) \sqrt{\hat{v}_{t,i}} + \frac{\beta_{1,t}}{2\alpha_{t-1}(1-\beta_{1,t})} (\theta_{,i}^* - \theta_{t,i})^2 \sqrt{\hat{v}_{t-1,i}} + \frac{\beta_1 \alpha_{t-1}(m_{t-1,i})^2}{2(1-\beta_1)\sqrt{\hat{v}_{t-1,i}}} + \frac{\alpha_t \xi_{t,i}}{2(1-\beta_1)} \frac{(\hat{m}_{t,i})^2}{\sqrt{\hat{v}_{t,i}}}. \quad (32)$$

From (17) and Fig. 2, it is clear that $0.5 \leq \xi_{t,i} \leq 1$. Therefore, $\xi_{t,i}$ can be removed from the last term of the above equation, and it still satisfies the inequality. Then,

$$\begin{aligned} g_{t,i}\left(\theta_{t,i} - \theta^*_{,i}\right) \leq{}& \frac{1}{2\alpha_t \xi_{t,i}(1-\beta_1)} \\ &\times \left(\left(\theta_{t,i} - \theta^*_{,i}\right)^2 - \left(\theta_{t+1,i} - \theta^*_{,i}\right)^2\right)\sqrt{\hat{\upsilon}_{t,i}} \\ &+ \frac{\beta_{1,t}}{2\alpha_{t-1}(1-\beta_{1,t})}\left(\theta^*_{,i} - \theta_{t,i}\right)^2 \sqrt{\hat{\upsilon}_{t-1,i}} \\ &+ \frac{\beta_1 \alpha_{t-1}(m_{t-1,i})^2}{2(1-\beta_1)\sqrt{\hat{\upsilon}_{t-1,i}}} + \frac{\alpha_t}{2(1-\beta_1)} \frac{(\hat{m}_{t,i})^2}{\sqrt{\hat{\upsilon}_{t,i}}}. \end{aligned} \tag{33}$$

We use Lemma 10.4 of Adam [38] and derive the regret bound by aggregating it across all the dimensions for $i \in \{1, \ldots, d\}$ and all the sequence of convex functions for $t \in \{1, \ldots, T\}$ in the upper bound of $f_t(\theta_t) - f_t(\theta^*)$ as

$$\begin{aligned} R(T) \leq{}& \sum_{i=1}^{d} \frac{1}{2\alpha_1 \xi_{1,i}(1-\beta_1)}\left(\theta_{1,i} - \theta^*_{,i}\right)^2 \sqrt{\hat{\upsilon}_{1,i}} \\ &+ \sum_{i=1}^{d}\sum_{t=2}^{T} \frac{1}{2(1-\beta_1)}\left(\theta_{t,i} - \theta^*_{,i}\right)^2 \left(\frac{\sqrt{\hat{\upsilon}_{t,i}}}{\alpha_t \xi_{t,i}} - \frac{\sqrt{\hat{\upsilon}_{t-1,i}}}{\alpha_{t-1}\xi_{t-1,i}}\right) \\ &+ \frac{\beta_1 \alpha G_\infty}{(1-\beta_1)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d} ||g_{1:T,i}||_2 \\ &+ \frac{\alpha G_\infty}{(1-\beta_1)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d} ||g_{1:T,i}||_2 \\ &+ \sum_{i=1}^{d}\sum_{t=1}^{T} \frac{\beta_{1,t}}{2\alpha_t(1-\beta_{1,t})}\left(\theta^*_{,i} - \theta_{t,i}\right)^2 \sqrt{\hat{\upsilon}_{t,i}}. \end{aligned} \tag{34}$$

By utilizing the assumptions that $\alpha = \alpha_t \sqrt{t}$, $||\theta_t - \theta^*||_2 \leq D$ and $||\theta_m - \theta_n||_\infty \leq D_\infty$, we can write as

$$\begin{aligned} R(T) \leq{}& \frac{D^2}{2\alpha(1-\beta_1)} \sum_{i=1}^{d} \frac{\sqrt{T\hat{\upsilon}_{T,i}}}{\xi_{1,i}} \\ &+ \frac{\alpha(1+\beta_1)G_\infty}{(1-\beta_1)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d} ||g_{1:T,i}||_2 \\ &+ \frac{D_\infty^2}{2\alpha} \sum_{i=1}^{d}\sum_{t=1}^{t} \frac{\beta_{1,t}}{(1-\beta_{1,t})}\sqrt{t\hat{\upsilon}_{t,i}} \\ \leq{}& \frac{D^2}{2\alpha(1-\beta_1)} \sum_{i=1}^{d} \frac{\sqrt{T\hat{\upsilon}_{T,i}}}{\xi_{1,i}} \\ &+ \frac{\alpha(1+\beta_1)G_\infty}{(1-\beta_1)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d} ||g_{1:T,i}||_2 \\ &+ \frac{D_\infty^2 G_\infty \sqrt{1-\beta_2}}{2\alpha} \sum_{i=1}^{d}\sum_{t=1}^{t} \frac{\beta_{1,t}}{(1-\beta_{1,t})}\sqrt{t}. \end{aligned} \tag{35}$$

It is shown in Adam [38] that $\sum_{t=1}^{t}(\beta_{1,t}/(1-\beta_{1,t}))\sqrt{t} \leq (1/(1-\beta_1)(1-\gamma)^2)$. Thus, the regret bound can be written as

$$\begin{aligned} R(T) \leq{}& \frac{D^2}{2\alpha(1-\beta_1)} \sum_{i=1}^{d} \frac{\sqrt{T\hat{\upsilon}_{T,i}}}{\xi_{1,i}} \\ &+ \frac{\alpha(1+\beta_1)G_\infty}{(1-\beta_1)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d} ||g_{1:T,i}||_2 \\ &+ \sum_{i=1}^{d} \frac{D_\infty^2 G_\infty \sqrt{1-\beta_2}}{2\alpha(1-\beta_1)(1-\lambda)^2}. \end{aligned} \tag{36}$$

We know $\xi_{1,i} = 1/(1+e^{-|g_{0,i}-g_{1,i}|}) = 1/(1+e^{-|g_{1,i}|})$ as $g_{0,i} = 0$. Therefore, the regret bound for diffGrad is as follows:

$$\begin{aligned} R(T) \leq{}& \frac{D^2}{2\alpha(1-\beta_1)} \sum_{i=1}^{d} (1+e^{-|g_{1,i}|})\sqrt{T\hat{\upsilon}_{T,i}} \\ &+ \frac{\alpha(1+\beta_1)G_\infty}{(1-\beta_1)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d} ||g_{1:T,i}||_2 \\ &+ \sum_{i=1}^{d} \frac{D_\infty^2 G_\infty \sqrt{1-\beta_2}}{2\alpha(1-\beta_1)(1-\lambda)^2}. \end{aligned} \tag{37}$$

□

ACKNOWLEDGMENT

The authors would like to thank NVIDIA Corporation for the support of GeForce Titan X Pascal GPU donated to Computer Vision Group, IIIT Sri City, Chittoor, India. They would also like to thank the anonymous Associate Editor and Reviewers for their valuable comments to improve the quality of this article. They would also like to thank Dr. Shrijita for editing this article.

REFERENCES

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[3] X. Lan, S. Zhang, P. C. Yuen, and R. Chellappa, "Learning common and feature-specific patterns: A novel multiple-sparse-representation-based tracker," *IEEE Trans. Image Process.*, vol. 27, no. 4, pp. 2022–2037, Apr. 2018.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556

[5] R. Shao, X. Lan, and P. C. Yuen, "Joint discriminative learning of deep dynamic textures for 3D mask face anti-spoofing," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 4, pp. 923–938, Apr. 2019.

[6] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2015.

[7] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2017.

[8] D. Yu and L. Deng, "Deep learning and its applications to signal and information processing [exploratory DSP]," *IEEE Signal Process. Mag.*, vol. 28, no. 1, pp. 145–154, Jan. 2011.

[9] X.-L. Zhang and J. Wu, "Deep belief networks based voice activity detection," *IEEE Trans. Audio, Speech Language Process.*, vol. 21, no. 4, pp. 697–710, Apr. 2013.

[10] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[11] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proc. 25th Int. Conf. Mach. Learn.*, Jul. 2008, pp. 160–167.

[12] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017.

[13] X. Lan, A. J. Ma, P. C. Yuen, and R. Chellappa, "Joint sparse representation and robust feature-level fusion for multi-cue visual tracking," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 5826–5841, Dec. 2015.

[14] S. Zhang, X. Lan, H. Yao, H. Zhou, D. Tao, and X. Li, "A biologically inspired appearance model for robust visual tracking," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2357–2370, Oct. 2017.

[15] X. Lan, M. Ye, R. Shao, B. Zhong, P. C. Yuen, and H. Zhou, "Learning modality-consistency feature templates: A robust RGB-infrared tracking system," *IEEE Trans. Ind. Electron.*, vol. 66, no. 12, pp. 9887–9897, Dec. 2019.

[16] S. P. T. Reddy, S. T. Karri, S. R. Dubey, and S. Mukherjee, "Spontaneous facial micro-expression recognition using 3D spatiotemporal convolutional neural networks," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, to be published.

[17] C. Nagpal and S. R. Dubey, "A performance evaluation of convolutional neural networks for face anti spoofing," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Jul. 2019, pp. 1–8.

[18] S. K. Roy, G. Krishna, S. R. Dubey, and B. B. Chaudhuri, "HybridSN: Exploring 3-D-2-D CNN feature hierarchy for hyperspectral image classification," *IEEE Geosci. Remote Sens. Lett.*, to be published.

[19] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[21] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.

[22] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1440–1448.

[23] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[24] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 779–788.

[25] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2961–2969.

[26] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8759–8768.

[27] S. H. S. Basha, S. Ghosh, K. K. Babu, S. R. Dubey, V. Pulabaigari, and S. Mukherjee, "RCCNet: An efficient convolutional neural network for histological routine colon cancer nuclei classification," in *Proc. 15th Int. Conf. Control, Autom., Robot. Vis. (ICARCV)*, Nov. 2018, pp. 1222–1227.

[28] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. COMPSTAT*, 2010, pp. 177–186.

[29] R. Sutton, "Two problems with back propagation and other steepest descent learning procedures for networks," in *Proc. 8th Annu. Conf. Cognit. Sci. Soc.*, 1986, pp. 823–832.

[30] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio, "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2933–2941.

[31] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Netw.*, vol. 12, no. 1, pp. 145–151, 1999.

[32] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. Int. Conf. Mach. Learn.*, Feb. 2013, pp. 1139–1147.

[33] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, Feb. 2011.

[34] J. Dean *et al.*, "Large scale distributed deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1223–1231.

[35] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Oct. 2014, pp. 1532–1543.

[36] M. D. Zeiler, "ADADELTA: An adaptive learning rate method," 2012, *arXiv:1212.5701*. [Online]. Available: https://arxiv.org/abs/1212.5701

[37] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning: Lecture 6A overview of mini-batch gradient descent course," Video Lectures Coursera, Mountain View, CA, USA, 2012.

[38] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: https://arxiv.org/abs/1412.6980

[39] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, G. Klambauer, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," 2017, *arXiv:1706.08500*. [Online]. Available: https://arxiv.org/abs/1706.08500

[40] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–23.

[41] J. Li, H. Liu, B. Zhong, Y. Wu, and Y. Fu, "Predictive local smoothness for stochastic gradient methods," 2018, *arXiv:1805.09386*. [Online]. Available: https://arxiv.org/abs/1805.09386

[42] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "signSGD: Compressed optimisation for non-convex problems," 2018, *arXiv:1802.04434*. [Online]. Available: https://arxiv.org/abs/1802.04434

[43] H. Huang, C. Wang, and B. Dong, "Nostalgic Adam: Weighting more of the past gradients when designing the adaptive learning rate," 2018, *arXiv:1805.07557*. [Online]. Available: https://arxiv.org/abs/1805.07557

[44] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proc. Int. Conf. Mach. Learn.*, 2003, pp. 928–936.

[45] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[46] A. Krizhevsky, "Learning multiple layers of features from tiny images," M.S. thesis, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, 2009.

[47] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. Int. Conf. Mach. Learn.*, Jun. 2013, pp. 1–6.

[48] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," 2015, *arXiv:1511.07289*.[Online]. Available: https://arxiv.org/abs/1511.07289

[49] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 971–980.

Shiv Ram Dubey (S'14–M'17) received the Ph.D. degree in computer vision and image processing from the Indian Institute of Information Technology at Allahabad (IIIT Allahabad), Allahabad, India, in 2016.

From August 2012 to February 2013, he was a Project Officer with the Computer Science and Engineering Department, IIT Madras, Chennai, India. He has been with the Indian Institute of Information Technology (IIIT) at Sri City, Chittoor, India, since June 2016, where he is currently an Assistant Professor of computer science and engineering. His research interest includes computer vision, deep learning, image processing, biometrics, medical imaging, convolutional neural networks, image feature description, content-based image retrieval, image-to-image transformation, face detection and recognition, facial expression recognition, texture, and hyperspectral image analysis.

Dr. Dubey was a recipient of several awards, including the Indo-Taiwan Joint Research Grant from Department of Science and Technology (DST)/Global Innovation and Technology Alliance (GITA), Government of India, the Best Ph.D. Award at Ph.D. Symposium, the IEEE-CICT2017 at IIITM Gwalior, the Early Career Research Award from SERB, Government of India, and NVIDIA GPU Grant Award Twice from NVIDIA. He received the Outstanding Certificate of Reviewing Award for *Information Fusion*, (Elsevier 2018) and the Best Paper Award at the IEEE UPCON 2015, a prestigious conference of the IEEE UP Section.

Soumendu Chakraborty received the Bachelor of Engineering (B.E.) degree in information technology from the University Institute of Technology, The University of Burdwan, Bardhaman, India, in 2005, and the M.Tech. degree in computer science and engineering from GLA University, Mathura, India, in 2013, and the Ph.D. degree from the Indian Institute of Information Technology at Allahabad, Allahabad, India, in 2018.

He has 12 years of teaching and research experience. He is currently working as an Assistant Professor with the Indian Institute of Information Technology at Lucknow, Lucknow, India. His research interests include computer vision, machine learning, image processing, biometric systems, image steganography, and pattern recognition.

Swalpa Kumar Roy (S'15) received the bachelor's degree in computer science and engineering from the West Bengal University of Technology, Kolkata, India, in 2012, and the master's degree in computer science and engineering from the Indian Institute of Engineering Science and Technology at Shibpur, Howrah, India, in 2015. He is currently pursuing the Ph.D. degree with the Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, Kolkata.

He was a Project Linked Person with the Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, from July 2015 to March 2016. He is currently an Assistant Professor with the Department of Computer Science and Engineering, Jalpaiguri Government Engineering College, Jalpaiguri, India. His research interests include computer vision, deep learning, remote sensing, texture feature description, and fractal image coding.

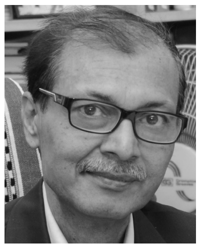
Snehasis Mukherjee (M'16) received the bachelor's degree in mathematics from the University of Calcutta, Kolkata, India, the master's degree in computer applications from Vidyasagar University, Midnapore, India, and the Ph.D. degree in computer science from the Indian Statistical Institute, Bengaluru, India, in 2012.

He did his Post-Doctoral Research works with the National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA. He is currently working as an Assistant Professor with the Indian Institute of Information Technology at SriCity (IIIT SriCity), Chittoor, India. He has written several peer-reviewed research articles (in reputed journals and conferences). His research area includes computer vision, machine learning, and image and video processing.

Satish Kumar Singh (M'11–SM'14) received the B.Tech., M.Tech., and Ph.D. degrees in 2003, 2005, and 2010, respectively.

He is currently an Associate Professor with the Indian Institute of Information Technology at Allahabad, Allahabad, India. He has over 13 years of experience in academic and research institutions. He has authored over 45 publications in reputed international journals and conference proceedings. His current research interests are in the areas of digital image processing, pattern recognition, multimedia data indexing and retrieval, watermarking, and biometrics.

Dr. Singh is a member of various professional societies, like the Institution of Electronics and Telecommunication Engineers. He is also an Executive Committee Member of the IEEE Uttar Pradesh Section from 2014. He is serving as the Secretary for the Signal Processing Society Chapter and the Uttar Pradesh Section. He is also serving as an editorial board member and a reviewer for many international journals.



Bidyut Baran Chaudhuri (LF'15) received the Ph.D. degree from IIT Kanpur, Kanpur, India, in 1980.

He was a Leverhulme Post-Doctoral Fellow with Queen's University at Belfast, Belfast, U.K., from 1981 to 1982. He joined the Indian Statistical Institute, Bengaluru, India, in 1978, where he worked as an Indian National Academy of Engineering (INAE) Distinguished Professor and a J. C. Bose Fellow with the Computer Vision and Pattern Recognition Unit. He is currently affiliated to Techno India University, Kolkata, India, as the Pro-Vice Chancellor (Academic). He pioneered the first workable optical character recognition (OCR) system for printed Indian scripts Bangla, Assamese, and Devnagari. He also developed a computerized Bharati Braille system with a speech synthesizer and has done statistical analysis of the Indian language. He has published about 425 research articles in international journals and conference proceedings. He also has authored/edited seven books in these fields. His research interests include pattern recognition, image processing, computer vision, natural language processing (NLP), signal processing, digital document processing, and deep learning.

Dr. Chaudhuri is a fellow of Indian National Science Academy (INSA), National Academy of Sciences India (NASI), INAE, International Association of Pattern Recognition (IAPR), and The World Academy of Sciences (TWAS). He received the Leverhulme Fellowship Award, the Sir J. C. Bose Memorial Award, the M. N. Saha Memorial Award, the Homi Bhabha Fellowship, the Dr. Vikram Sarabhai Research Award, the C. Achuta Menon Award, the Homi Bhabha Award: Applied Sciences, Ram Lal Wadhwa Gold Medal, the Jawaharlal Nehru Fellowship, the J. C. Bose Fellowship, and the Om Prakash Bhasin Award. He acted as the General Chair and the Technical Co-Chair at various international conferences. He is an Associate Editor of three international journals.