AdaInject: Injection-Based Adaptive Gradient Descent Optimizers for Convolutional Neural Networks

Shiv Ram Dubey, Senior Member, IEEE, S. H. Shabbeer Basha, Satish Kumar Singh, Senior Member, IEEE, and Bidyut Baran Chaudhuri, Life Fellow, IEEE

Abstract—The convolutional neural networks (CNNs) are generally trained using stochastic gradient descent (SGD)-based optimization techniques. The existing SGD optimizers generally suffer with the overshooting of the minimum and oscillation near minimum. In this article, we propose a new approach, hereafter referred as AdaInject, for the gradient descent optimizers by injecting the second-order moment into the first-order moment. Specifically, the short-term change in parameter is used as a weight to inject the second-order moment in the update rule. The AdaInject optimizer controls the parameter update, avoids the overshooting of the minimum, and reduces the oscillation near minimum. The proposed approach is generic in nature and can be integrated with any existing SGD optimizer. The effectiveness of the AdaInject optimizer is explained intuitively as well as through some toy examples. We also show the convergence property of the proposed injection-based optimizer. Furthermore, we depict the efficacy of the AdaInject approach through extensive experiments in conjunction with the state-of-the-art optimizers, namely AdamInject, diffGradInject, RadamInject, and AdaBeliefInject, on four benchmark datasets. Different CNN models are used in the experiments. A highest improvement in the top-1 classification error rate of 16.54% is observed using diffGradInject optimizer with ResNeXt29 model over the CIFAR10 dataset. Overall, we observe very promising performance improvement of existing optimizers with the proposed AdaInject approach.

Impact Statement—Adaptive moment based optimizers are among the popular gradient descent optimization techniques for the training of deep learning models. They try to control the step size based on the gradient behavior. However, the existing gradient descent optimization techniques either overshoot the “steep and narrow” valley (i.e., minimum) or oscillate near it, due to large step size caused by the exponential moving average of gradients used for parameter updates. The AdaInject optimization technique we introduce in this paper tackled this problem by incorporating the immediate parameter change weighted second order moment injection for the parameter updates. Using the proposed optimization technique, a significant improvement is observed in the performance of image classification using different CNN models. Moreover, the proposed AdaInject approach can be used with any existing adaptive moment based optimization technique. Hence, it can provide the alternative optimizers with better step size control to train different deep learning models for diverse applications.

Index Terms—Adaptive optimizers, convolutional neural networks (CNNs), deep learning, image recognition, parameter update history, second-order moment injection, stochastic gradient descent (SGD).

I. INTRODUCTION

DEEP learning has shown a great impact over the performance of the neural networks for a wide range of problems [1]. In the recent past, convolutional neural networks (CNNs) have shown very promising results for different computer vision applications, such as object recognition [2], [3], [4], [5], object detection [6], [7], face recognition [8], [9], image quality assessment [10], gesture recognition [11], COVID-19 grading [12], and many more. CNNs have also been used as basic building blocks in other networks, such as autoencoder [13], [14], [15], Siamese network [16], [17], generative adversarial networks [18], [19], etc.

The training of different types of deep neural networks is mainly performed with the help of stochastic gradient descent (SGD)-based optimization [20]. SGD optimizer updates the parameters of the network based on the gradient of objective function w.r.t. the corresponding parameters [21]. The vanilla SGD optimization suffers from the following three problems.

1) Zero gradient in local minimum and saddle regions leading to no update in the parameters.
2) A jittering effect along steep dimensions due to the inconsistent changes in the loss caused by the different parameters.
3) Noisy updates due to the gradient computed from the batch of data.

SGD with moment (SGDM) [22] considers the first-order moment (i.e., velocity) as an exponential moving average (EMA) of gradient for each parameter while training progresses [23]. The parameter is updated in SGDM based on the EMA of gradient, which resolves the problem of zero gradient.
Several SGD-based optimization techniques have been proposed in the recent past [24, 25, 26, 27, 28, 29, 30, etc.]. AdaGrad [24] controls the learning rate (LR) by dividing it with the root of the sum of the squares of the past gradients. However, it makes the LR very small after certain iterations and kills the parameter update. AdaDelta [25] resolves the diminishing LR issue of AdaGrad by considering only a few immediate past gradients. However, it is not able to exploit the global information. In another attempt to resolve the problem of AdaGrad, RMSProp [26] divides the LR by the root of the exponentially decaying average of squared gradients. In 2015, Kingma and Ba [27] proposed the adaptive moment-based Adam optimizer. Adam combines the ideas of SGD and RMSProp and uses first- and second-order moments. Adam computes the first-order moment as the EMA of gradients and uses it to update the parameter. Adam also computes the second-order moment as the EMA of the square of gradients and uses it to control the LR. Adam performs well in practice to train the CNNs [27]. However, it suffers from overshooting and oscillations near minimum and varying gradient variance due to batch-wise computation. DiffGrad [28] resolved the issues as posed by Adam by introducing a friction term in parameter update using the rate of change in gradients. Radam [29] resolved the variance issue as posed by Adam by rectifying the variance of gradients during parameter update. AdaBelief [30] uses the belief in gradients to compute the second-order moment. The belief in gradients is computed as the difference between the gradient and the first-order moment of the gradient. Other recently proposed and notable gradient descent optimizers are proportional integral derivative [31], Nesterov’s moment Adam [32], nostalgic Adam [33], YOGI [34], adaptive bound [35], adaptive and moment-bound [36], aggregated moment [37], Lamb [38], Adam projection [39], gradient centralization [40], AdaHessian [41], and AngularGrad [42].

The adaptive SGD optimization techniques have led to a promising performance on deep CNN models. The majority of the abovementioned adaptive gradient descent optimizers suffer due to the overshooting of the minimum and oscillation near minimum. However, it is evident that a robust online step size adaptation in optimization plays an important role in gradient descent optimization [43]. We resolve the abovementioned issues by injecting the second-order moment in first-order for the parameter update, which is weighted by the short-term parameter update history to incorporate the robust adaptation of step size. The major contributions of this work are summarized as follows.

1) We propose AdaInject for the adaptive optimizers by injecting the short-term parameter change weighted second-order moment in EMA of gradient used for parameter update.

2) We provide an intuitive explanation in support of the effectiveness of the proposed AdaInject in different optimization scenarios.

3) We show the effect of the proposed approach using toy examples. The convergence analysis is also conducted using regret bound, which shows the convergence property of the proposed approach.

4) We validate the superiority of the proposed injection concept with the recent state-of-the-art optimizers, including Adam [27], diffGrad [28], Radam [29], and AdaBelief [30], using a wide range of CNN models for image classification over four benchmark datasets.

5) The proposed concept is generic and can be easily integrated with any existing adaptive moment-based SGD optimizer.

The rest of this article is organized as follows. The proposed injection-based optimizers are presented in Section II. Intuitive explanation and empirical analysis are presented in Section III. Convergence analysis is presented in Section IV. Experimental analysis is presented in Section V. Finally, Section VI concludes this article.

II. PROPOSED INJECTION-BASED OPTIMIZERS

As per the conventions used in Adam [27], the aim of gradient descent optimization is to minimize the loss function $f(\theta) \in \mathbb{R}$, where $\theta \in \mathbb{R}^d$ is the parameter. The gradient $(g_t)$ at $t$th step is computed as $g_t = \nabla f^\top(\theta_{t-1})$. Adam computes the first-order moment ($m_t$) and second-order moment ($v_t$) as the EMA of $g_t$ and $g_t^2$, respectively, which can be written as

$$m_t = \beta_1 \times m_{t-1} + (1 - \beta_1) \times g_t$$

(1)

$$v_t = \beta_2 \times v_{t-1} + (1 - \beta_2) \times g_t^2$$

(2)

Algorithm 1: Adam Optimizer.

**Initialize:** $\theta_0, m_0 \leftarrow 0, v_0 \leftarrow 0, t \leftarrow 0$

**Hyperparameters:** $\alpha, \beta_1, \beta_2$

**While** $\theta_t$ not converged

$t \leftarrow t + 1$

$$g_t = \nabla f^\top(\theta_{t-1})$$

$$m_t \leftarrow \beta_1 \times m_{t-1} + (1 - \beta_1) \times g_t$$

$$v_t \leftarrow \beta_2 \times v_{t-1} + (1 - \beta_2) \times g_t^2$$

**Bias Correction**

$$\hat{m}_t \leftarrow \frac{m_t}{(1 - \beta_1^t)}$$

$$\hat{v}_t \leftarrow \frac{v_t}{(1 - \beta_2^t)}$$

**Update**

$$\theta_t \leftarrow \theta_{t-1} - \alpha \times \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$$

Algorithm 2: AdaInject (i.e., Adam + AdaInject) Optimizer.

**Initialize:** $\theta_0, s_0 \leftarrow 0, v_0 \leftarrow 0, t \leftarrow 0$

**Hyperparameters:** $\alpha, \beta_1, \beta_2, k$

**While** $\theta_t$ not converged

$t \leftarrow t + 1$

$$g_t = \nabla f^\top(\theta_{t-1})$$

If $t = 1$

$$s_t = \beta_1 \times s_{t-1} + (1 - \beta_1) \times g_t$$

Else

$$\Delta \theta \leftarrow \theta_{t-2} - \theta_{t-1}$$

$$s_t \leftarrow \beta_1 \times s_{t-1} + (1 - \beta_1) \times (g_t + \Delta \theta \times g_t^2) / k$$

$$v_t \leftarrow \beta_2 \times v_{t-1} + (1 - \beta_2) \times g_t^2$$

**Bias Correction**

$$\hat{s}_t \leftarrow s_t / (1 - \beta_1^t), \hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$$

**Update**

$$\theta_t \leftarrow \theta_{t-1} - \alpha \hat{s}_t / (\sqrt{\hat{v}_t} + \epsilon)$$

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
where $\beta_1$ and $\beta_2$ are the smoothing hyperparameters, typically set as $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The $g_t^2$ is computed as $g_t \times g_t$ as in the Adam. A bias correction is performed as $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$, $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ to avoid very large step size in the initial iterations. The parameter update rule in Adam [27] is given as

$$
\theta_t \leftarrow \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}
$$

(3)

where $\alpha$ is the LR and $\epsilon = 1e^{-8}$ is a small number for numerical stability to avoid division by zero. A detailed algorithm of Adam optimizer is summarized in Algorithm 1. The first-order moment $m_t$ is used to update the parameters in Adam, wherein the second-order moment $v_t$ is used to control the LR. It can be noted that Adam mainly relies on the gradients.

However, the SGDm considers only the momentum to update the parameters as follows:

$$
\theta_t \leftarrow \theta_{t-1} - \alpha m_t.
$$

(4)

In order to utilize the parameter update history information during optimization, we propose a novel concept named AdaInject. Basically, we inject the short-term parameter change weighted second-order moment into first-order moment to compute the injected moment using the EMA of $(g_t + \Delta \theta \times g_t^2) / k$

$$
s_t = \beta_1 \times s_{t-1} + (1 - \beta_1) \times (g_t + \Delta \theta \cdot g_t^2) / k
$$

(5)

where $\Delta \theta = \theta_{t-2} - \theta_{t-1}$ is the short-term change in parameter $\theta$ to utilize the parameter history information and $k$ is an injection controlling hyperparameter, typically set to 2 in the experiment. The injection of parameter history guided second-order moment helps the optimizers to perform the smaller updates near minimum (i.e., “steep and narrow” valley) to avoid the overshooting and oscillation, while reasonably large updates are used in the small curvature regions. This phenomenon is shown in Fig. 1 with a detailed analysis in the following section. We perform the bias correction of injected moment and second-order moment as $\tilde{s}_t \leftarrow s_t / (1 - \beta_1^t)$ and $\tilde{v}_t \leftarrow v_t / (1 - \beta_2^t)$, respectively.

The parameter $(\theta)$ update of AdamInject optimizer is given as

$$
\theta_t \leftarrow \theta_{t-1} - \alpha \times \tilde{s}_t / \left( \sqrt{\tilde{v}_t} + \epsilon \right)
$$

(6)

where $\alpha$ is the LR and $\epsilon = 1e^{-8}$ is a small number for numerical stability to avoid the division by zero. We refer to Adam optimizer with the proposed second-order moment injection as AdamInject optimizer. A detailed algorithm of AdamInject optimizer is presented in Algorithm 2 with highlighted changes in blue color as compared to vanilla Adam optimizer, which is shown in Algorithm 1.

Basically, we use the proposed AdaInject concept with four existing state-of-the-art optimizers: 1) Adam [27]; 2) diffGrad [28]; 3) Radam [29]; and 4) AdaBelief [30], and propose the corresponding AdamInject (i.e., Adam + AdaInject), diffGradInject (i.e., diffGrad + AdaInject), RadamInject (i.e., Radam and AdaInject), and AdaBeliefInject (i.e., AdaBelief + AdaInject) optimizers, respectively. The algorithms for different optimizers (i.e., without and with AdaInject), such as diffGrad, diffGradInject, Radam, RadamInject, AdaBelief, and AdaBeliefInject, are provided in Supplementary. Though we test the proposed injection concept with four optimizers, it can be extended to any EMA-based gradient descent optimization technique. In the following section, we analyze the property of the proposed approach.

III. INTUITIVE EXPLANATION AND EMPIRICAL ANALYSIS

In this section, we present an intuitive explanation using a 1-D optimization landscape having three scenarios and an empirical analysis using three toy examples.

A. Intuitive Explanation

The existing gradient descent optimizers, such as Adam, diffGrad, Radam, etc., only consider the EMA of gradient for parameter update. However, the consideration of parameter history is important as the gradient behavior and required step size are different for different regions of loss optimization landscape [43], [30]. We explain the advantage of the proposed optimizer by considering three typical scenarios using a 1-D optimization curvature (i.e., S1, S2, and S3), as shown in Fig. 1. The bias correction step is ignored in the explanation for simplicity.

1) S1: This scenario depicts a flat region on the optimization landscape. An ideal optimizer is expected to perform large parameter updates in this scenario. The $|g_t|$ and $|\Delta \theta|$ in flat region are small. Thus, the EMA of gradient (i.e., $m_t$) as well as the EMA of the proposed injected gradient (i.e., $s_t$) are small. It leads to a small step size in both Adam and AdaInject due to the small value of $\sqrt{\tilde{v}_t}$ in the denominator.

2) S2: The “large gradient, small curvature” is another scenario in the optimization landscape. The gradient $|g_t|$ is higher in such regions. An ideal optimizer is expected to take the large parameter updates in such regions. The EMA of gradient (i.e., $m_t$) as well as squared gradient (i.e., $v_t$) are large. Moreover, the EMA of the proposed injected gradient (i.e., $s_t$) is also sufficiently large as $|\Delta \theta|$
is small. Hence, the SGD takes large step in this scenario. However, both the Adam and AdamInject take relatively smaller step due to large value of $\sqrt{v_t}$ in the denominator. But, we show experimentally that this problem can be reduced by considering AdaBelief concept [30] with the proposed injection idea (i.e., AdaBeliefInject).

3) Scenario 3: The third scenario is parameter updates near “steep and narrow” valley (i.e., minimum). It is expected for an ideal optimizer to decrease the step size for parameter updates in this scenario to avoid the overshooting as well as to reduce the oscillation near the valley. The proposed AdamInject optimizer is very beneficial in this scenario too. The gradient $|g_t|$ is large in this scenario, hence $m_t$ and $\sqrt{v_t}$ are also large. The SGD suffers due to large value of $m_t$. This problem is reduced to a certain extent in Adam due to large value of $\sqrt{v_t}$ in the denominator. In this scenario, $|\Delta \theta|$ is large, $\Delta \theta < 0$ when $g_t > 0$, and $\Delta \theta > 0$ when $g_t < 0$, leading to $|s_t| < |m_t|$ (note that $t$ is expected not to be the initial iterations near minimum, rather sufficiently large). Hence, the proposed AdamInject method reduces $s_t$ while enjoying the benefits of Adam (i.e., large $\sqrt{v_t}$ in denominator) leading to a reduced step size, which avoids the overshooting and oscillation near minimum to a greater extent. In order to show this effect using toy examples, we conduct an empirical study with the help of synthetic, nonconvex functions in the following section.

B. Empirical Analysis Using Toy Examples

We perform the empirical analysis using three synthetic, 1-D, nonconvex functions by following the protocol of diffGrad [28]. These functions are given as

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

(7)

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

(8)

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

(9)

where $-\infty < x < +\infty$ is the input. Functions $F1$, $F2$, and $F3$ are shown in Fig. 2 in the first column and in the first, second, and third rows, respectively, for $-1 < x < +1$. The parameter $x$ is initialized at $-1$. The experiment is performed by computing the regression loss as the objective function. The second column shows the parameter values at different iterations using Adam and AdamInject optimizers. Similarly, the third column illustrates the parameter values at different iterations using diffGrad and diffGradInject optimizers. It can be noticed that Adam overshoots the minimum for both $F1$ and $F2$ functions, whereas AdamInject is able to avoid the overshooting due to the small step size caused by the proposed parameter change weighted second-order moment injection in parameter update. In other cases, including Adam and AdamInject for $F3$ function...
optimizer, similar to Adam [27]. Let us represent the unknown sequence of convex cost functions as \( f_1(\theta), f_2(\theta), \ldots, f_T(\theta) \). We want to estimate parameter \( \theta_i \) at each iteration \( t \) and assess over \( f_i(\theta) \). The regret bound is commonly used in such scenarios to assess the algorithm where the information of the sequence is not known in advance. The sum of the difference between all the previous online guesses \( f_i(\theta_i) \) and the best fixed point parameter \( f_i(\theta^*) \) from a feasible set \( \chi \) of all the previous iterations are used to compute the regret bound. The regret bound is given as

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

where \( \theta_t = \arg\min_{\theta \in \chi} \sum_{i=1}^{T} f_i(\theta) \). The regret bound for the proposed injection-based AdamInject is noticed as \( O(\sqrt{T}) \), which is the same as Adam and is comparable to general convex online learning approaches. We provide the proof in the supplementary. We consider \( g_{t,i} \) as the gradient for the \( i \)th element in the \( T \)th iteration, \( g_{t:1,i} = [g_{1,i}, g_{2,i}, \ldots, g_{t,i}] \in \mathbb{R}^k \) is the gradient vector in the \( i \)th dimension up to \( T \) iterations, and \( \gamma \triangleq \frac{\beta_1}{dG^2} \).

**Theorem 1:** Assume that the gradients for function \( f_i \) (i.e., \( ||g_{t,i}||_2 \leq G \) and \( ||g_{t,i}|| \leq G_\infty \) are bounded for all \( \theta \in \mathbb{R}^d \). Let also consider that the bounded distance is generated by the proposed optimizer between any \( \theta_t \) (i.e., \( ||\theta_n - \theta_m||_2 \leq D \) and \( ||\theta_n - \theta_m|| \leq D_\infty \) for any \( m,n \in \{1,\ldots,T\} \)). Let \( \gamma \triangleq \frac{\beta_1}{dG^2} \), \( \beta_1, \beta_2 \in [0,1) \), satisfy \( \frac{\beta_1}{\beta_2} < 1 \), \( \alpha_t = \frac{\beta_1}{\beta_2} \), and \( \beta_{1,t} = \beta_1 \lambda^{t-1}, \lambda \in (0,1) \) with \( \lambda \) around 1, e.g., \( 1 - 10^{-8} \). For all \( T \geq 1 \), the proposed injection-based AdamInject shows the following guarantee as derived in the supplementary:

\[
R(T) \leq \frac{D^2}{\alpha(1-\beta_1)} \sum_{i=1}^{d} \sqrt{T}v_{T,i}^2 + \frac{2\alpha(1+\beta_1)G_\infty}{(1-\beta_1)\sqrt{1-\beta_2(1-\gamma)^2}} \sum_{i=1}^{d} ||g_{1:T,i}||_2^2 + \frac{d}{1-\beta_1(1-\lambda)} \sum_{i=1}^{d} ||g_{1:T,i}||_2^2 + 4D_\infty G_\infty^2 \sum_{i=1}^{d} ||g_{1:T,i}||_2^2.
\]

(11)

The aggregation terms over the dimension (i.e, \( d \)) can be very small as compared to the corresponding upper bounds, such as \( \sum_{i=1}^{d} ||g_{1:T,i}||_2 \ll \sqrt{d}G_\infty \sqrt{T} \), \( \sum_{i=1}^{d} ||g_{1:T,i}||_2^2 \ll \sqrt{d}G_\infty^2 \sqrt{T} \), and \( \sum_{i=1}^{d} \sqrt{T}v_{T,i}^2 \ll \sqrt{d}G_\infty \sqrt{T} \). The adaptive methods, such as the proposed optimizers and Adam, show the upper bound as \( O(\log d_{\infty}) \), which is better than \( O(\sqrt{dT}) \) of nonadaptive optimizers. By following the convergence analysis of Adam [27], we also use the decay of \( \beta_{1,t} \).

We show the convergence of average regret of AdamInject in the following corollary with the help of the abovementioned theorem and \( \sum_{i=1}^{d} ||g_{1:T,i}||_2 \ll \sqrt{d}G_\infty \sqrt{T} \), \( \sum_{i=1}^{d} ||g_{1:T,i}||_2^2 \ll \sqrt{d}G_\infty^2 \sqrt{T} \), and \( \sum_{i=1}^{d} \sqrt{T}v_{T,i}^2 \ll \sqrt{d}G_\infty \sqrt{T} \).

**Corollary 1:** Assume that the gradients for function \( f_i \) (i.e., \( ||g_{t,i}||_2 \leq G \) and \( ||g_{t,i}|| \leq G_\infty \) are bounded for all \( \theta \in \mathbb{R}^d \). Let also consider that the bounded distance is generated by the proposed optimizer between any \( \theta_t \) (i.e., \( ||\theta_n - \theta_m||_2 \leq D \) and \( ||\theta_n - \theta_m|| \leq D_\infty \) for any \( m,n \in \{1,\ldots,T\} \)). For all \( T \geq 1 \), the proposed injection-based AdamInject optimizer shows the

Fig. 3. Optimization illustration for Rastrigin (upper row) and Rosenbrock (lower row) functions using Adam (left-hand side column) and AdamInject (right-hand side column).

and diffGrad and diffGradInject for all three functions, the effect of the proposed optimizer can be easily observed in terms of the smooth parameter updates and less oscillations near minimum by accumulating the injected momentum in an accurate direction. It is noticed that AdamInject is more effective with Adam than diffGrad, as diffGrad utilizes the short-term gradient change as friction coefficient. These results confirm that the proposed parameter change guided second-order moment injection leads to accurate and precise parameter updates, especially near “steep and narrow” valley.

In order to demonstrate the effect of the proposed optimizer on 2-D optimization, we consider nonconvex Rastrigin and Rosenbrock functions, which are widely used to show the optimization characteristics. The Rastrigin function has one global minimum at (0.0, 0.0). However, the Rosenbrock has one global minimum at (1.0, 1.0). The optimization trajectories using Adam and AdamInject optimizers under the same experimental setup are shown in Fig. 3. It can be noticed that Adam is not able to converge over the Rastrigin function due to the presence of several local minima, whereas the AdamInject is able to converge over the Rastrigin function due to the improved parameter updates caused by the second-order moment injection. It is also observed that the Adam optimizer takes more steps to reach the minimum over the Rosenbrock function due to irregular parameter updates caused by long, narrow, parabolic-shaped flat valley. However, the AdamInject optimizer is able to tackle this issue and reaches the minimum in less number of steps over the Rosenbrock function.

IV. CONVERGENCE ANALYSIS

We use the online learning framework to show the convergence property of the proposed injection-based AdamInject

1[Online]. Available: https://github.com/jettify/pytorch-optimizer
following guarantee:

\[
R(T) = O\left(\frac{1}{\sqrt{T}}\right).
\]  

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

V. EXPERIMENTAL ANALYSIS

In this section, first we describe the experimental setup. Then, we present the detailed results using different optimizers. Finally, we analyze effects of the hyperparameters.

A. Experimental Setup

We use a wide range of CNN models (i.e., VGG16 [44], ResNet18, ResNet50, ResNet101 [2], SENet18 [45], ResNeXt29 [3], and DenseNet121 [4]) to test the suitability of the proposed AdaInject concept for optimizers. We follow the publicly available Pytorch implementation\(^2\) of these CNN models. For the ResNeXt29 model, we set the cardinality as 4 and bottleneck width as 64. We train all the CNN models using all the optimizers under the same experimental setup. The training is performed for 100 epochs with a batch size (BS) of 64 for CIFAR10/100 and FashionMNIST (FMNIST) and 256 for the TinyImageNet dataset. The LR is set to 0.001 for the first 80 epochs and 0.0001 for the last 20 epochs. Different computers are used for the experiments, including Google colabatory.\(^3\) We performed a random crop and random horizontal flip over training data. The normalization is performed for both training and test data.

In order to demonstrate the efficacy of the proposed AdaInject-based optimizers experimentally, we use four benchmark object recognition dataset: 1) CIFAR10[46]; 2) CIFAR100 [46]; 3) FMNIST [47]; and 4) TinyImageNet\(^4\). We use CIFAR and FMNIST datasets directly from the PyTorch library. The CIFAR10 dataset consists of a total 60000 images of dimension \(32 \times 32 \times 3\) from ten object classes with 6000 images per class. The training set contains 50 000 images with 5000 images per class, and the test set contains 10 000 images with 1000 images per class in CIFAR10. The CIFAR100 dataset contains all the images of CIFAR10, but is categorized into 100 classes. Thus, the CIFAR100 dataset contains 50 000 training images and 10 000 test images with 500 and 100 images per class, respectively. The FMNIST dataset contains 70 000 labeled fashion images of dimension \(28 \times 28\) from ten categories. The training and test sets consist of 60 000 and 10 000 images, respectively. The TinyImageNet dataset [48] is a subset of the large-scale visual recognition ImageNet challenge [49]. This dataset consists of the images from 200 object classes with 100 000 images in the training set (i.e., 500 images in each class) and 10 000 images in the validation set (i.e., 50 images in each class).

B. Experimental Results

We compare the performance using four recent state-of-the-art adaptive gradient descent optimizers (i.e., Adam [27], diffGrad [28], Radam [29], and AdaBelief [30]), without and with the proposed injection approach. We consider VGG16 [44], ResNet18, ResNet50, ResNet101 [2], SENet18 [45], ResNeXt29 [3], and DenseNet121 [4] CNN models. The experimental results over the CIFAR10 dataset are given in Table I in terms of the error rate. It is observed that the performance of all CNN models is improved with AdaInject-based optimizers as compared to its performance with corresponding vanilla optimizers. The RadamInject optimizer leads to best performance using the DenseNet121 model with 5.10\% error rate in classification. The highest improvement is reported by the ResNeXt29 model using diffGradInject. Moreover, the performance of the ResNeXt29 model is also significantly improved using AdaBeliefInject. In general, we observe better performance gain by heavy CNN models.

The results over the CIFAR100 dataset are given in Table II. The best performance of 77.26\% accuracy is achieved by the RadamInject optimizer using the ResNeXt29 model. The performance of ResNeXt29 is improved significantly using the proposed injection for optimizers with highest improvement by AdamInject. The results due to the proposed injection-based optimizers are improved using all the CNN models except RadamInject using VGG16 and AdaBeliefInject using ResNet101. Note that Radam does not use second-order moment if rectification criterion is not met and AdaBelief reduces the second-order moment. These could be the possible reasons that the performance of RadamInject and AdaBeliefInject is marginally down in some cases. A very similar trend is also noticed over the FMNIST dataset in Table III where the performance using the proposed approach is improved in all the cases. The best accuracy of 95.44\% is observed for the AdaBeliefInject optimizer using the DenseNet121 model. An outstanding improvement in top-1 error is perceived for the ResNeXt29 model over the FMNIST dataset using the optimizers with the proposed AdaInject concept. The performance of other models is also significantly improved due to the proposed injection approach.

We also perform the experiments over the TinyImageNet dataset using VGG16, ResNet18, and SENet18 models and show the results in terms of the classification accuracy in % in Table IV for different optimizers with and without the proposed injection concept. It is observed from this experiment that the proposed approach is able to improve the performance of the existing optimizers over the large-scale dataset as well. These results confirm that the proposed injection updates the parameter in an optimal way by utilizing the short-term parameter update information with second-order moment.

C. Effect of Injection Hyperparameter (i.e., \(k\))

In the previous results, we use the value of the injection hyperparameter (i.e., \(k\)) as 2. We show a performance comparison by considering the value of \(k\) as 1, 2, 3, 4, 5, 10, 20, and 50 in Table V. The results are presented using the AdamInject optimizer for VGG16 and ResNet18 models over the CIFAR10, CIFAR100, FMNIST, and TinyImageNet datasets. It is noticed that \(k = 2\) is better suitable for the VGG16 model on CIFAR10, MNIST, and TinyImageNet datasets. Moreover, the accuracy

\(^2\)[Online]. Available: https://github.com/kuangliu/pytorch-cifar

\(^3\)[Online]. Available: https://colab.research.google.com/

\(^4\)[Online]. Available: http://cs231n.stanford.edu/tiny-imagenet-200.zip
TABLE I

| CNN models | Classification error (%) using different optimizers without and with AdaInject | Adam | Adad Inject | diffGrad | diffGradInject | Radam | RadamInject | AdaBelief | AdaBeliefInject |
|------------|--------------------------------------------------------------------------------|------|-------------|----------|---------------|-------|-------------|-----------|----------------|
| VGG16      | 7.45                                                                        | 7.20 (↑ 3.36) | 7.24 | 7.04 (↑ 2.76) | 7.06 | 6.88 (↑ 2.55) | 7.29 | 7.07 (↑ 3.02) |
| ResNet18   | 6.46                                                                        | 6.20 (↑ 4.02) | 6.51 | 6.10 (↑ 6.30) | 6.18 | 5.87 (↑ 3.02) | 6.37 | 6.30 (↑ 1.10) |
| SENet18    | 6.61                                                                        | 6.29 (↑ 4.84) | 6.44 | 6.21 (↑ 3.57) | 6.05 | 5.83 (↑ 3.64) | 6.59 | 6.23 (↑ 5.46) |
| ResNet50   | 6.17                                                                        | 5.89 (↑ 4.54) | 6.19 | 5.78 (↑ 7.43) | 5.86 | 5.29 (↑ 9.73) | 5.90 | 5.78 (↑ 2.03) |
| ResNet101  | 6.90                                                                        | 6.01 (↑ 12.90)| 6.45 | 5.69 (↑ 11.78)| 6.29 | 5.76 (↑ 8.43) | 6.37 | 6.03 (↑ 5.34) |
| ResNetX29  | 6.79                                                                        | 6.16 (↑ 9.28) | 6.83 | 5.70 (↑ 16.54)| 6.00 | 5.67 (↑ 5.50) | 6.43 | 5.99 (↑ 6.84) |
| DenseNet121| 6.30                                                                        | 5.63 (↑ 10.63)| 5.90 | 5.43 (↑ 7.97) | 5.25 | 5.10 (↑ 2.86) | 6.05 | 5.64 (↑ 6.78) |

The results with the proposed approach are highlighted in bold. The improvement in the error due to the proposed injection concept is also mentioned.

The highest increase for an optimizer is also highlighted in bold. The symbols ↑ and ↓ represent the improvement and degradation in %, respectively, in the Top-1 error. We follow the same convention in the results reported in Tables II and III also. These results are computed as the average over three independent trials.

TABLE II

| CNN models | Classification error (%) using different optimizers without and with AdaInject | Adam | Adad Inject | diffGrad | diffGradInject | Radam | RadamInject | AdaBelief | AdaBelief Inject |
|------------|--------------------------------------------------------------------------------|------|-------------|----------|---------------|-------|-------------|-----------|----------------|
| VGG16      | 32.71                                                                       | 31.81 (↑ 2.75) | 31.81 | 30.80 (↑ 3.18) | 29.31 | 30.07 (↑ 2.59) | 31.08 | 30.04 (↑ 3.35) |
| ResNet18   | 28.91                                                                       | 27.28 (↑ 5.64) | 26.50 | 26.23 (↑ 1.02) | 26.78 | 25.84 (↑ 3.51) | 27.28 | 26.31 (↑ 3.56) |
| SENet18    | 29.15                                                                       | 28.74 (↑ 1.41) | 28.60 | 27.64 (↑ 3.36) | 27.66 | 26.63 (↑ 3.72) | 26.90 | 26.52 (↑ 1.14) |
| ResNet50   | 28.12                                                                       | 25.44 (↑ 9.53) | 24.94 | 24.18 (↑ 3.05) | 25.05 | 24.13 (↑ 3.67) | 24.47 | 24.25 (↑ 0.90) |
| ResNet101  | 25.78                                                                       | 23.98 (↑ 6.98) | 26.58 | 24.17 (↑ 9.07) | 25.74 | 23.83 (↑ 7.42) | 24.12 | 24.24 (↑ 0.50) |
| ResNetX29  | 28.78                                                                       | 24.96 (↑ 13.27) | 25.47 | 24.53 (↑ 3.69) | 24.66 | 22.74 (↑ 7.79) | 24.61 | 23.63 (↑ 3.98) |
| DenseNet121| 26.40                                                                       | 24.33 (↑ 7.84) | 24.14 | 23.66 (↑ 1.99) | 25.17 | 23.06 (↑ 3.48) | 24.68 | 24.06 (↑ 2.51) |

These results are computed as the average over three independent trials. The boldface represents the better results.

TABLE III

| CNN models | Classification error (%) using different optimizers without and with AdaInject | Adam | Adad Inject | diffGrad | diffGradInject | Radam | RadamInject | AdaBelief | AdaBelief Inject |
|------------|--------------------------------------------------------------------------------|------|-------------|----------|---------------|-------|-------------|-----------|----------------|
| VGG16      | 5.13                                                                         | 5.04 (↑ 2.72) | 5.13 | 5.03 (↑ 1.95) | 5.11 | 5.07 (↑ 0.78) | 5.12 | 4.97 (↑ 2.93) |
| ResNet18   | 4.76                                                                         | 4.74 (↑ 0.42) | 4.82 | 4.65 (↑ 3.55) | 4.78 | 4.67 (↑ 2.30) | 4.95 | 4.75 (↑ 4.04) |
| SENet18    | 5.14                                                                         | 4.95 (↑ 3.70) | 5.11 | 4.79 (↑ 6.26) | 5.08 | 4.79 (↑ 5.71) | 5.06 | 4.91 (↑ 2.96) |
| ResNet50   | 5.10                                                                         | 4.76 (↑ 6.67) | 4.93 | 4.77 (↑ 3.25) | 4.98 | 4.84 (↑ 2.81) | 5.10 | 4.78 (↑ 6.27) |
| ResNet101  | 4.94                                                                         | 4.65 (↑ 5.87) | 5.05 | 4.73 (↑ 6.34) | 4.91 | 4.64 (↑ 5.50) | 5.21 | 4.69 (↑ 9.98) |
| ResNetX29  | 6.16                                                                         | 5.59 (↑ 9.25) | 5.92 | 5.16 (↑ 12.84) | 5.78 | 5.37 (↑ 7.09) | 5.25 | 4.90 (↑ 6.67) |
| DenseNet121| 4.88                                                                         | 4.69 (↑ 3.89) | 4.77 | 4.70 (↑ 1.47) | 4.89 | 4.68 (↑ 4.29) | 4.68 | 4.56 (↑ 2.56) |

These results are computed as the average over three independent trials. The boldface represents the better results.

TABLE IV

| CNN models | Accuracy (%) using different optimizers without and with AdaInject | Adam | Adad Inject | diffGrad | diffGradInject | Radam | RadamInject | AdaBelief | AdaBelief Inject |
|------------|-----------------------------------------------------------------|------|-------------|----------|---------------|-------|-------------|-----------|----------------|
| VGG16      | 44.05                                                           | 44.58 (↑ 1.20) | 46.00 | 47.18 (↑ 2.57) | 45.92 | 46.38 (↑ 1.00) | 47.88 | 48.25 (↑ 0.77) |
| ResNet18   | 50.58                                                           | 51.90 (↑ 2.61) | 52.04 | 52.37 (↑ 0.63) | 52.12 | 52.50 (↑ 0.73) | 52.05 | 52.74 (↑ 1.33) |
| SENet18    | 48.04                                                           | 49.52 (↑ 3.08) | 49.51 | 50.28 (↑ 1.56) | 50.73 | 51.01 (↑ 0.55) | 51.76 | 51.94 (↑ 0.35) |

These results are computed as the average over three independent trials. The boldface represents the better results.

using $k = 2$ is also either best or second-best for the ResNet18 model on CIFAR10, MNIST, and TinyImageNet datasets. It is also evident that the results on the fine-grained CIFAR100 dataset are best using $k = 5$ for both VGG16 and ResNet18 models. It is suggested to consider the value of $k \in \{2, 3, 4, 5\}$. The original selection of the value of $k$ as 2 is also justified from this analysis.

D. Effect of BS and LR

In the previous experiments, the BS and LR were set to 64 and 0.001, respectively. In this experiment, we analyze the impact of BS and LR as detailed in Table VI. The results are reported for VGG16 and ResNet18 models on the CIFAR10, CIFAR100, MNIST, and TinyImageNet datasets. The BSs are considered as...


### TABLE V

| Model  | Dataset | k=1 | k=5 | k=10 | k=20 | k=50 |
|--------|---------|-----|-----|------|------|------|
| VGG16  | CIFAR10 | 92.68 | 92.80 | 92.78 | 92.90 | 92.61 | 91.83 | 90.75 |
|        | MNIST   | 80.04 | 80.61 | 80.54 | 80.32 | 80.45 | 80.21 | 79.38 |
|        | TinyImageNet | 65.66 | 65.54 | 65.41 | 65.27 | 65.31 | 65.04 | 64.05 |
| ResNet18 | CIFAR100 | 93.71 | 93.80 | 93.85 | 93.76 | 93.91 | 93.41 | 92.84 |
|        | MNIST   | 95.15 | 95.26 | 95.19 | 95.35 | 95.18 | 95.16 | 94.84 |
|        | TinyImageNet | 49.09 | 51.90 | 48.47 | 50.43 | 51.17 | 50.11 | 49.37 |

Note that best and second-best results are highlighted in bold and underline, respectively.

### TABLE VI

| Model  | Dataset | BS | LR | 32 | 64 | 128 | 0.0001 | 0.001 | 0.01 |
|--------|---------|----|----|----|----|-----|--------|------|------|
| VGG16  | CIFAR10 | 92.46 | 92.80 | 92.45 | 91.16 | 92.80 | 92.45 | 66.92 | 68.19 | 66.97 |
|        | MNIST   | 70.00 | 68.19 | 68.03 | 66.92 | 68.19 | 66.97 | 66.92 | 68.19 | 66.97 |
|        | TinyImageNet | 41.67 | 44.58 | 42.69 | 45.18 | 44.58 | 39.33 | 45.18 | 44.58 | 39.33 |
| ResNet18 | CIFAR100 | 93.94 | 94.13 | 93.71 | 92.36 | 94.13 | 93.65 | 94.13 | 94.13 | 93.65 |
|        | MNIST   | 70.05 | 74.16 | 72.76 | 70.79 | 74.16 | 67.31 | 74.16 | 72.76 | 67.31 |
|        | TinyImageNet | 49.63 | 51.90 | 50.18 | 49.30 | 51.90 | 45.96 | 49.30 | 51.90 | 45.96 |

Note that best results are highlighted in bold.

32, 64, and 128, respectively. It is evident from the results that the BS as 64 is better suitable with the proposed AdamInject optimizer in all the cases. The LR is considered as 0.0001, 0.001, and 0.01, respectively. Note that the LR is divided by 10 once in all the cases after 80 epochs of training for a fair comparison. It is noticed that the proposed optimizer performs best for 0.001 LR in almost all the cases. This analysis confirms the suitability of original BS (i.e., 64) and LR (i.e., 0.001) choices used for the experiments.

### VI. CONCLUSION

In this article, we presented a novel and generic injection-based EMA of gradients for parameter update by utilizing the parameter change information along with the second-order momentum. The proposed injection approach leads to an accurate and precise update by performing smaller updates near minimum to avoid the overshooting as well as oscillation and reasonably higher updates in the small curvature regions. The effect of the proposed injection-based optimizers was observed using toy examples. The convergence property of the proposed optimizer was also analyzed. The object recognition results for different CNN models over benchmark datasets using four optimizers showed the improvement of the proposed injection concept. It is noticed that the injection hyperparameter as 2 yields better results in majority of the cases using the AdamInject optimizer. It is also noted that the BS as 64 and LR as 0.001 are better suitable with the proposed AdamInject optimizer. The intuitive explanation, empirical, convergence, and experimental analyses are evident that the proposed injection-based optimizers lead to better optimization of CNNs by avoiding the overshooting of the minimum and reducing the oscillation near minimum to a greater extent.

### REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, pp. 436–444, 2015.
[2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 770–778.
[3] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 5987–5995.
[4] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 2261–2269.
[5] J. Kantipudi, S. R. Dubey, and S. Chakraborty, “Color channel perturbation attacks for fooling convolutional neural networks and a defense against such attacks,” IEEE Trans. Artif. Intell., vol. 1, no. 2, pp. 181–191, Oct. 2020.
[6] R. Girshick, “Fast r-CNN,” in Proc. IEEE Int. Conf. Comput. Vis., 2015, pp. 1440–1448.
[7] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-CNN: Towards real-time object detection with region proposal networks,” in Proc. 28th Int. Conf. Neural Inf. Process. Syst., 2015, pp. 91–99.
[8] A. Krizhevsky, I. Sutskever, and G. Hinton, “ImageNet classification with deep convolutional neural networks,” in Proc. Conf. NeurIPS, 2012, pp. 1097–1105.
[9] J. M. Tyagi, Z. Yang, and R. Ranzato, and L. Wolf, “DeepFace: Closing the gap to human-level performance in face verification,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2014, pp. 1701–1708.
[10] Z. Pan, F. Yuan, X. Wang, L. Xu, S. Xiao, and S. Kwong, “No-reference image quality assessment via multi-branch convolutional neural networks,” IEEE Trans. Artif. Intell., early access, Jan. 27, 2022, doi: 10.1109/TAI.2022.3146804.
[11] J. Zou and L. Cheng, “A transfer learning model for gesture recognition based on the deep features extracted from CNN,” IEEE Trans. Artif. Intell., vol. 2, no. 5, pp. 447–458, Oct. 2021.
[12] C. De Vente et al., “Automated COVID-19 grading with convolutional neural networks in computed tomography scans: A systematic comparison,” IEEE Trans. Artif. Intell., vol. 3, no. 2, pp. 129–138, Apr. 2022.
[13] K. Zeng, J. Yu, R. Wang, C. Li, and D. Tao, “Coupled deep autoencoder for single image super-resolution,” IEEE Trans. Cybern., vol. 47, no. 1, pp. 27–37, Jan. 2017.
[14] Z. Chen, K. Yin, M. Fisher, S. Chaudhuri, and H. Zhang, “BAE-Net: Branched autoencoder for shape co-segmentation,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2019, pp. 8489–8498.
[15] G. Dewangan and S. Maurya, “Fault diagnosis of machines using deep convolutional beta-variational autoencoder,” IEEE Trans. Artif. Intell., vol. 3, no. 2, pp. 287–296, Apr. 2022.
[16] X. Huang and J. Shen, “Triplet loss in siamese network for object tracking,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 472–488.
[17] H. Fan and H. Ling, “Siamese cascaded region proposal networks for real-time visual tracking,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 7944–7953.
[18] Y.-J. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” in Proc. IEEE Int. Conf. Comput. Vis., 2017, pp. 2242–2251.
[19] K. K. Babu and S. R. Dubey, “PCSGAN: Perceptual cyclic-synthesized generative adversarial networks for thermal and NIR to visible image transformation,” Neurocomputing, vol. 413, pp. 41–50, 2020.
[20] L. Bottou, “Large-scale machine learning with stochastic gradient descent,” in Proc. COMPSTAT, 2010, pp. 177–186.
[21] H. Robbins and S. Monro, “A stochastic approximation method,” Ann. Math. Statist., vol. 22, pp. 400–407, 1951.
[22] B. T. Polyak, “Some methods of speeding up the convergence of iteration methods,” USSR Comput. Math. Math. Phys., vol. 4, no. 5, pp. 1–17, 1964.
[23] I. S. J. J. Martens, G. Dahl, and G. Hinton, “On the importance of initialization and momentum in deep learning,” in Proc. Int. Conf. Mach. Learn., 2013, pp. 1139–1147.
[24] 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, 2011.
S. H. Shabbeer Basha received the Ph.D. degree in computer science and engineering from the Indian Institute of Information Technology, Sri City, India. He currently is a Lead Engineer with PathPartner Technology Private Limited, Bengaluru, India, where he is involved in R&D activities on neural network compression and deep learning. His research interests include computer vision, deep learning, deep model compression, unsupervised domain adaptation, transfer learning, and multiskil learning.

Satish Kumar Singh (Senior Member, IEEE) was with the Jaypee University of Engineering and Technology Guna, India, from 2005 to 2012. Since 2013, he has been an Associate Professor with the Indian Institute of Information Technology Allahabad, Prayagraj, India, where he is heading the Computer Vision and Biometrics Lab. His research interests include image processing, computer vision, biometrics, deep learning, and pattern recognition.

Dr. Singh has been proactively offering his volunteer services to IEEE for the last many years in various capacities. He is the Section Chair IEEE Uttar Pradesh Section (2021–2022) and a Member of the IEEE India Council (2021). He was the Vice Chair, Operations, Outreach, and Strategic Planning of the IEEE India Council (2020) and the IEEE Uttar Pradesh Section (2019 and 2020). Prior to that, he was a Secretary of the IEEE UP Section (2017 and 2018), a Treasurer of the IEEE UP Section (2016 and 2017), and a Joint Secretary of the IEEE UP Section (2015) and Convener Web and Newsletters Committee (2014 and 2015). He is also a Technical Committee Affiliate of IEEE SPS IVMSP and MMSP and the Chair of the IEEE Signal Processing Society Chapter of Uttar Pradesh Section.

Bidyut Baran Chaudhuri (Life Fellow, IEEE) received the Ph.D. degree from the Indian Institute of Technology Kanpur, Kanpur, India, in 1980.

From 1981 to 1982, he was a Leverhulme Postdoctoral Fellow with Queen’s University, U.K. In 1978, he joined Indian Statistical Institute, where he worked as an INAE Distinguished Professor and a J C Bose Fellow with Computer Vision and Pattern Recognition Unit. He has pioneered the first workable OCR system for printed Indian scripts Bangla, Assamese, and Devnagari. He also developed computerized Bharti Braille system with speech synthesizer and has done statistical analysis of Indian language. He is currently a Pro-Vice Chancellor (Academic) with Techno India University, Kolkata, India. His research interests include pattern recognition, image processing, computer vision, and deep learning.

Prof. Chaudhuri 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, a Ram Lal Wadhwa Gold Medal, the Jawaharlal Nehru Fellowship, the J C Bose Fellowship, and the Om Prakash Bhasin Award. He is a Fellow of INSA, NASI, INAE, IAPR, and The World Academy of Sciences (TWAS).

Shiv Ram Dubey (Senior Member, IEEE) received the Ph.D. degree in computer vision and image processing from the Indian Institute of Technology (IIT) Allahabad, Prayagraj, India, in 2016.

From 2012 to 2013, he was a Project Officer with the Indian Institute of Technology, Madras. He was a Research Scientist, from June 2016 to December 2016, and an Assistant Professor, from December 2016 to July 2021, with IIT, Sri City. Since July 2021, he has been with IIT Allahabad, where he is currently an Assistant Professor of Information Technology.

His research interests include computer vision and deep learning.