SMARAN: CLOSING THE GENERALIZATION GAP WITH PERFORMANCE DRIVEN OPTIMIZATION METHOD

Anonymous authors
Paper under double-blind review

ABSTRACT

Optimization methods have evolved significantly by introducing various learning rate scheduling techniques and adaptive learning strategies. Although these methods have achieved faster convergence, they often struggle to generalize well to unseen data compared to traditional approaches such as Stochastic Gradient Descent (SGD) with momentum. Adaptive methods such as Adam store each parameter's first and second moments of gradients, which can be memory-intensive. To address these challenges, we propose a novel SMARAN optimization method that adjusts the learning rate based on the model's performance rather than the objective function's curvature. This approach is particularly effective for minimizing stochastic loss functions, standard in deep learning models. Traditional gradient-based methods may get stuck in regions where the gradient vanishes, such as plateaus or local minima. Therefore, instead of only depending on the gradient, we use the model's performance to estimate the appropriate step size. We performed extensive experiments on standard vision benchmarks, and the generalization trends observed with SMARAN demonstrate compelling distinctions relative to adaptive and non-adaptive optimizers.

1 INTRODUCTION

A stochastic optimization problem is defined as

$$\min_{\mathbf{x}\in X} \mathbb{E}_{\xi}[f(\mathbf{x},\xi)] \tag{1}$$

where ξ is a random variable that introduces uncertainty in the objective function $f(\mathbf{x},\xi)$ and $\mathbf{x}$ is the decision variable belonging to the feasible domain X. Standard solution methods for this form of optimization problems include gradient-based approaches such as SGD and its variants Lecun et al. (1998); Graves et al. (2013); Krizhevsky et al. (2012). However, gradient-based methods rely on the gradient direction for updating the parameters, but the gradient itself is affected by the stochastic nature of the function; hence, a negative gradient direction may not always be the best search direction. In nonconvex settings, relying on gradient magnitude to identify the optimal point could be misleading. Flat regions, saddle points, and inflexion points exhibit the property of zero gradient. Finally, gradients give local neighborhood information; hence, one may get stuck at local optima instead of searching for a global solution. Then we have the problem of gradient explosion in steep regions. In addition, gradient-based optimizers lack step size adaptability based on landscape curvature, resulting in uniform step size scaling. In the literature, many variants are proposed to overcome these drawbacks. SGD with momentum Polyak (1964) and Nesterov Nesterov (1983) overcome the first problem by aggregating past gradients to determine the current update direction. Aggregation reduces the effect of the stochasticity in the gradient. Adaptive methods overcome the uniform scaling of the gradient along all coordinate directions. AdaGrad Duchi et al. (2011) was the first algorithm in this line of research. AdaGrad used the historical sum of squared gradients to adjust the learning rate of individual parameters, resulting in faster learning. However, the accumulated squared gradients grow monotonically, causing the learning rate to shrink and leading to premature convergence. Later methods, such as RMSProp Tieleman (2012) and Adam Kingma & Ba (2015), overcome this difficulty using an exponential moving average (EMA) of gradients. Adam is the most prominent optimizer used among the adaptive optimizers. It uses the EMA of

gradient (the first-moment estimate) for update direction and normalizes the learning rate with the EMA of gradient square, the second-moment estimate. Although Adam-based methods have the advantage of faster convergence, storing first and second-moment estimates for each parameter becomes memory-intensive. Also, there is no clear evidence that Adam-based methods perform better in generalization than SGD with momentum. To overcome these drawbacks of previous methods, we introduce a novel optimization approach based on the objective function value rather than gradient dependence. Main contributions of the paper are:

- A novel optimizer SMARAN, which is based on the concept of EMA but uses the objective function value to adjust the learning rate instead of gradients, unlike Adam-based methods.
- Theoretical regret analysis of our objective function in the online convex optimization framework, and provide bounds on the learning rate.
- We experimentally compared our algorithm with state-of-the-art methods. Experimental results demonstrate that SMARAN outperformed other methods in terms of generalization ability for vision tasks.

2 Related works

Progression from manually scheduled updates to gradient-based adaptivity marks the advancement of optimization methods. Classical methods such as Stochastic Gradient Descent (SGD) and its momentum-augmented variants Polyak (1964), including Nesterov Accelerated Gradient (NAG) Nesterov (1983), laid the foundation for this development. These methods primarily focused on exploiting gradient direction and aggregating past values to reduce noise from stochastic updates. Nevertheless, they fail to adapt the learning rate based on the loss landscape, which results in oscillations near the optimal point.

Learning rate scheduling schemes were introduced based on training steps to overcome these limitations. While step decay Ge et al. (2019) reduces the learning rate at predefined intervals, cosine annealing Loshchilov & Hutter (2017) and cyclic schedules Smith (2017) use periodic changes. Although these methods improve convergence, they lack responsiveness towards the loss landscape and model performance. Their performance is heavily dependent on manual hyperparameter tuning.

Recent works on adaptive learning rate modify the Polyak stepsize for stochastic nonconvex optimization Loizou et al. (2021b). Orvieto et al. (2022) demonstrates a polyak stepsize variant with decreasing stepsize that gives a convergence rate equivalent to gradient descent with proper initialization.

A paradigm shift occurred with the coming of adaptive methods such as AdaGrad Duchi et al. (2011), RMSProp Tieleman (2012), and Adam Kingma & Ba (2015). They introduced a parameter-wise adaptation of the learning rate. AdaGrad uses accumulated squared gradients to normalize the learning rate, penalizing frequently updated directions but often resulting in premature convergence. RMSProp overcomes this drawback by using the EMA of squared gradients, promoting smoother adaptation. Adam combines first and second-order moment estimates of gradients for learning rate adaptation. This results in stable updates with rapid initial convergence. Empirical studies show that adaptive methods may overfit, resulting in inferior performance to SGD with momentum on specific benchmarks.Chen et al. (2020); Reddi et al. (2018)

More recent optimization approaches include AdamW Loshchilov & Hutter (2019), which decouples the weight decay from gradient updates. AMSGrad Reddi et al. (2018) controls the learning rate to become monotonically decreasing over iterations. Yogi Zaheer et al. (2018) prevents the second moment estimate of Adam from exploding by sign correction. AdaBound Luo et al. (2019) clips the bounds of the learning rate to avoid the exploding and vanishing problem. PAdam Chen et al. (2020) scales the learning rate by a tunable adaptivity parameter. RAdam Liu et al. (2020) rectifies the adaptive learning rate variance. AdaBelief Zhuang et al. (2020) replaces the uncentred second moment with a centred variance estimate around the gradient. DecGD Shao et al. (2025) decomposes the gradient into a product of the surrogate loss and its gradient, which allows the learning rate adaptation based on the loss vector.

Now that we have traversed the evolution of optimization methods from simple gradient heuristics to momentum-based adaptation, we introduce our novel optimizer, SMARAN, whose mechanism and theoretical insights are discussed in the next section.

3 METHODOLOGY

Let $f : \mathbb{R}^n \to \mathbb{R}$ denote the loss function to minimize and let $\mathbf{x} \in X \subseteq \mathbb{R}^n$ be the decision variable. We have three hyperparameters: the global learning rate η, the regularization coefficient λ, and the moment discount factor γ. We included the factor ϵ with a value 10^{-8} for the numerical stability of the division operations. The terms $\mathbf{m}_t$ and v_t represent the moments of normalized gradient loss and square loss at iteration t, respectively, with initial values set to zero.

The previous section identifies three key factors that drive optimization algorithm design: update direction, update magnitude, and adaptiveness across different landscapes. Following these insights and the gaps in optimizing a stochastic function, we incorporated these aspects into our method, illustrated in Algorithm 1. The algorithm finds the optimal point using gradient information, momentum, a loss-based scaling mechanism for adaptivity, and variable regularization to avoid overfitting. The algorithm blends normalization, regularization, and an adaptive learning rate.

Following the Normalized Gradient Descent Shor (1985), to estimate the update direction, we use the normalized gradient $\hat{g}_t$ instead of the complete gradient $\nabla f(\mathbf{x}_t)$

$$\hat{\mathbf{g}}_t = \frac{\nabla f(\mathbf{x}_t).}{\|\nabla f(\mathbf{x}_t)\| + \epsilon} \tag{2}$$

Since the loss function is stochastic, we take the exponential average over past gradients for smoothing purposes to reduce the effect of uncertainty. The first moment is

$$\mathbf{m}_t = \gamma \mathbf{m}_{t-1} + \hat{\mathbf{g}}_t. \tag{3}$$

Theorem 1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable loss function parameterized by $\mathbf{x}$, then the norm of the exponential average of the normalized gradients over some time step t, given by Eq. (3), is upper bounded as*

$$\|\mathbf{m}_t\| < \frac{1}{1-\gamma} \tag{4}$$

Proof. Expanding Eq. (3),

$$\mathbf{m}_t = \sum_{\tau=1}^{t} \gamma^{t-\tau} \hat{\mathbf{g}}_\tau$$

Using the triangle inequality,

$$\|\mathbf{m}_t\| \leq \sum_{\tau=1}^{t} |\gamma^{t-\tau}| . \|\hat{\mathbf{g}}_\tau\| \tag{5}$$

Since $\|\hat{\mathbf{g}}\| < 1$ due to ϵ factor in Eq. (2)

$$\|\mathbf{m}_t\| < \sum_{\tau=1}^{t} |\gamma^{t-\tau}| \tag{6}$$

$$= \frac{1-\gamma^t}{1-\gamma} \tag{7}$$

As $t \to \infty$, sum of geometric progression becomes

$$\frac{1-\gamma^t}{1-\gamma} \to \frac{1}{1-\gamma} > \|\mathbf{m}_t\|.$$

Moreover, the first moment is upper-bounded. Hence, this moment prevents gradient explosion, provided the multiplicative learning factor is not ∞. □

This approach is effective for high-dimensional or ill-conditioned landscapes.

Adapting the learning rate based on the curvature of the function is done using a performance-based factor instead of depending on the gradients. Our performance-based factor is

$$\frac{f(\mathbf{x}_t)}{\sqrt{v_t}+\epsilon} \tag{8}$$

where

$$v_t = \gamma v_{t-1} + f(\mathbf{x}_t)^2 \tag{9}$$

Theorem 2. *Let $f : \mathbb{R}^n \to \mathbb{R}_+$ be a continuous loss function parameterized by $\mathbf{x}$, and $\{\mathbf{x}_t\}$ for $t = 1, 2, ..., T$ be a finite parameter sequence generated by gradient descent updates in T iterations, then for every time step t,*

$$0 < \frac{f(\mathbf{x}_t)}{\sqrt{v_t}+\epsilon} < 1 \tag{10}$$

where

$$v_t = \sum_{\tau=1}^{t} \gamma^{t-\tau} f(\mathbf{x}_\tau)^2 \tag{11}$$

Proof. From Eq. (9), we have $v_t \geq f(\mathbf{x}_t)^2$ which implies,

$$\begin{aligned}
\sqrt{v_t} &\geq f(\mathbf{x}_t) \\
\sqrt{v_t}+\epsilon &> f(\mathbf{x}_t) \\
\frac{1}{\sqrt{v_t}+\epsilon} &< \frac{1}{f(\mathbf{x}_t)} \\
\frac{f(\mathbf{x}_t)}{\sqrt{v_t}+\epsilon} &< 1
\end{aligned}$$

Since $f : \mathbb{R}^n \to \mathbb{R}_+$, both numerator and denominator are positive,

$$0 \leq \frac{f(\mathbf{x}_t)}{\sqrt{v_t}+\epsilon} < 1 \tag{12}$$

This factor is 0 only when $f(\mathbf{x}_t) = 0$. □

Previous methods use the EMA of the gradient square for normalizing the learning rate because gradients give the curvature of the landscape. However, for a nonconvex setting, steep curvature results in slow learning, whereas in our approach, the optimizer adjusts the learning rate based on its recent losses. If recent losses are high, then the optimizer updates the parameters cautiously. In contrast, if recent losses are low and decreasing with each timestep, the learning rate increases, making convergence faster. Unlike other methods stuck at local minima or landscapes where the gradient vanishes, our optimizer searches for paths to find the global minimum. For constant loss, SMARAN's learning rate approaches $\sqrt{1-\gamma}$ regardless of the loss magnitude (proof given in Appendix A.2). In an ideal case, as $\mathbf{x} \to \mathbf{x}^*$(global minima), $f(\mathbf{x}) \to 0$ hence,

$$\lim_{\mathbf{x}_t \to \mathbf{x}^*} \frac{f(\mathbf{x}_t)}{\sqrt{v_t}+\epsilon} = 0$$

To avoid getting overfit, we included a weight decay regularization term in the update step.

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \left(\frac{f(\mathbf{x}_t)}{\sqrt{v_t}+\epsilon} \right) (\mathbf{m}_t + \lambda \mathbf{x}_t) \tag{13}$$

AdamW inspires the weight decay, but unlike AdamW, which uses a constant weight decay, ours is an adaptive weight decay controlled by an adaptive learning rate term. Since the learning rate scheduler is based on the objective function value over training data, if the optimizer tries to overfit the training data, the same proportion of regularization prevents the model from overfitting.

Finally, SMARAN's memory requirement is lower than the adaptive methods since the adaptive learning rate is a scalar quantity.

Algorithm 1 SMARAN

Input: Initial vector $\mathbf{x}_0 \in X \subseteq \mathbb{R}^n$, loss function $f(\mathbf{x})$
Parameter: η, λ, γ
Output: $\mathbf{x}_T$
Initialize: $\mathbf{m_0} = \mathbf{0}$, $v_0 = 0$

1: **for** $t = 1$ **to** T **do**
2: $\quad \hat{\mathbf{g}}_t = \frac{\nabla f(\mathbf{x}_t)}{\|\nabla f(\mathbf{x}_t)\| + \epsilon}$
3: $\quad \mathbf{m}_t = \gamma \mathbf{m}_{t-1} + \hat{\mathbf{g}}_t$
4: $\quad v_t = \gamma v_{t-1} + f(\mathbf{x}_t)^2$
5: $\quad \mathbf{x}_{t+1} = \mathbf{x}_t - \eta \left(\frac{f(\mathbf{x}_t)}{\sqrt{v_t} + \epsilon} \right) (\mathbf{m}_t + \lambda \mathbf{x}_t)$
6: **end for**
7: **return** $\mathbf{x}_T$

4 Convergence Analysis

We perform the SMARAN algorithm's convergence analysis and highlight a risk bound under a convex setting. First introduced in Duchi et al. (2011), convergence analysis in a convex setting was discussed in many of the later works on adaptive methods, including Adam Kingma & Ba (2015), AMSGrad Reddi et al. (2018), Adabound Luo et al. (2019), Adabelief Zhuang et al. (2020), and DecGD Shao et al. (2025).

4.1 Online Convex Optimization

Given the objective function $f_t : X \to \mathbb{R}$, the online convex optimization framework aims to minimize the regret $R(T)$

$$R(T) = \sum_{t=1}^{T} f_t(\mathbf{x}_t) - \min_{\mathbf{x} \in X} \sum_{t=1}^{T} f_t(\mathbf{x}) \tag{14}$$

Standard assumptions Duchi et al. (2011); Reddi et al. (2018); Hazan et al. (2016) of online convex optimization framework are as follows

Assumption 1. (1) The domain $X \subseteq \mathbb{R}^n$ is a bounded convex set; the diameter of X is assumed bounded. For some bound D, $\|x - y\| \leq D \; \forall \; x, y \in X$. (2) f_t is a convex function. (3) Gradient of f_t-∇f_t is assumed to be bounded. For some bound G, $\|\nabla f_t\| \leq G, \forall \; \mathbf{x}_t \in X$.

Theorem 3. *Under Assumption 1,* $\lambda \in (0, 1)$, $\eta_t = \frac{\eta}{\sqrt{t}}$, $\eta > 0$, *and* $V_t = \frac{f(\mathbf{x}_t)}{\sqrt{v_t} + \epsilon}$, *the regret bound of SMARAN is*

$$R(T) \leq \frac{GD^2\sqrt{T}}{\eta V} + G\lambda D^2 + G\eta(G + \lambda D)^2(2\sqrt{T} - 1) \tag{15}$$

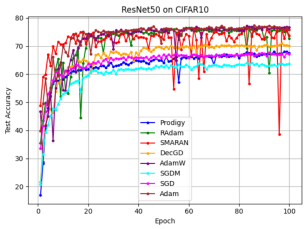
Proof of Theorem 3 is given in Appendix A.1. We conclude that, like previous adaptive methods Kingma & Ba (2015); Reddi et al. (2018); Luo et al. (2019); Zhuang et al. (2020); Shao et al. (2025), SMARAN also has an upper bound in $\mathcal{O}(\sqrt{T})$.

5 Experimental Results

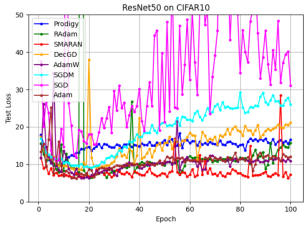
We highlight the results of an extensive evaluation of the SMARAN algorithm on different benchmark datasets for the vision task. We empirically demonstrate the generalization capability of SMARAN over state-of-the-art models. All the experiments are performed on NVIDIA RTX A6000 GPU with Python 3.12.7 and Pytorch 2.6.0 + cu124. Code for the proposed optimizer is available here.

5.1 Experimental Setup

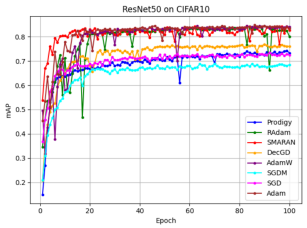
We perform experiments on an image classification task over multiple datasets and models. We use CIFAR-10 and CIFAR-100 Krizhevsky & Hinton (2009) datasets with $60,000$ color images of resolution 32×32. CIFAR-10 contains 10 object categories, and CIFAR-100 includes 100 categories.



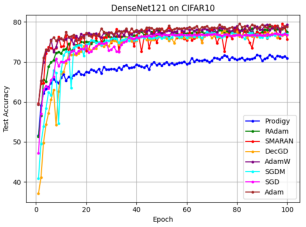
(a) Test accuracy of ResNet50 on CIFAR10



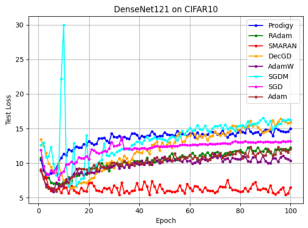
(b) Test Loss of ResNet50 on CIFAR10



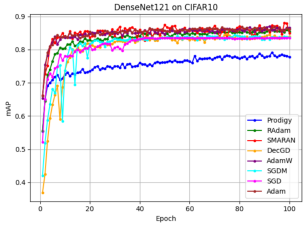
(c) mAP of ResNet50 on CIFAR10



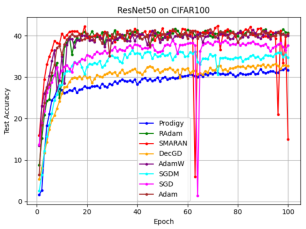
(d) Test accuracy of DenseNet121 on CIFAR10



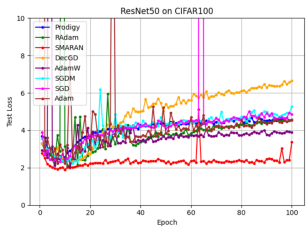
(e) Test Loss of DenseNet121 on CIFAR10



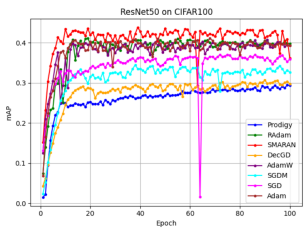
(f) mAP of DenseNet121 on CIFAR10



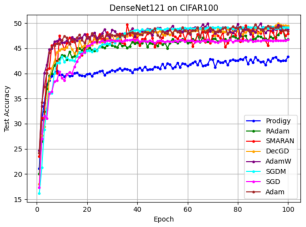
(g) Test accuracy of ResNet50 on CIFAR100



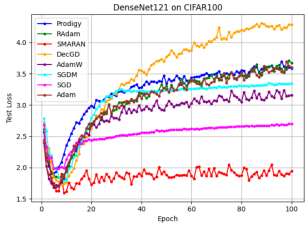
(h) Test Loss of ResNet50 on CIFAR100



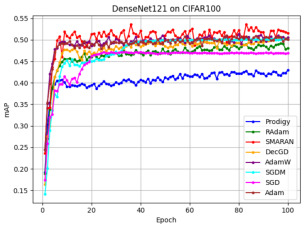
(i) mAP of ResNet50 on CIFAR100



(j) Test accuracy of DenseNet121 on CIFAR100



(k) Test Loss of DenseNet121 on CIFAR100

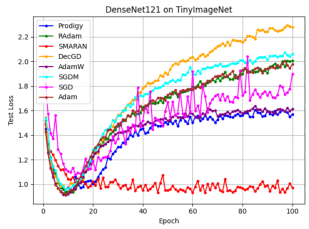


(l) mAP of DenseNet121 on CIFAR100

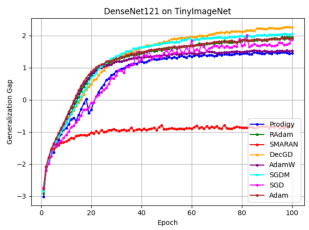
Figure 1: Experimental Results on CIFAR10 and CIFAR100 datasets.

We split the data into $50,000$ training samples and $10,000$ test samples. For the Tiny ImageNet dataset, we used $100,000$ training samples and $10,000$ test samples of size 64×64. Tiny ImageNet contains 200 unique categories.

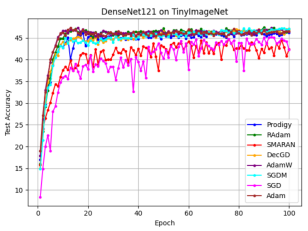
The architectures used for image classification include ResNet50 He et al. (2015) and DenseNet121 Huang et al. (2017). The architectures follow the standard configurations available in the PyTorch package. We use SGD, SGD with momentum, Adam, AdamW, RAdam, DecGD, and Prodigy Mishchenko & Defazio (2024) as optimizers for a comparative study with SMARAN for the image classification task. Unless otherwise stated, all optimizers are initialized with the default hyperparameter values as mentioned in the PyTorch official documentation. We use cross-entropy loss as an objective function. We train each optimizer with a list of learning rates in $[10^{-1}, 10^{-2}, 10^{-3}]$ to find the best-performing configurations. Fig. 1 compares different models with the best-performing configurations for the CIFAR10 and CIFAR100 datasets. SGD is configured with 0.1 learning rate 0 momentum, and weight decay. SGDM uses the same configuration with 0.9 momentum. Adam and AdamW follow a learning rate of 0.001 with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ and weight decay of 0



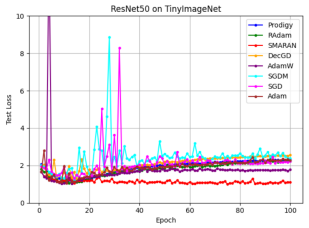
(a) Test loss of DenseNet121 on TinyImageNet



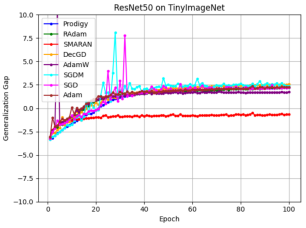
(b) Generalization gap of DenseNet121 on TinyImageNet



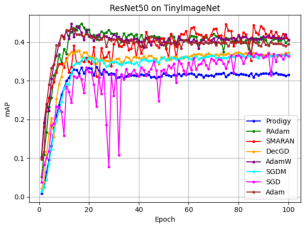
(c) mAP of DenseNet121 on TinyImageNet



(d) Test loss of ResNet50 on TinyImageNet



(e) Generalization Gap of ResNet50 on TinyImageNet



(f) mAP of ResNet50 on TinyImageNet

Figure 2: Experimental Results on Tiny Imagenet Dataset.

and 0.01 respectively. RAdam follows the same configuration as Adam. DecGD and Prodigy follow their respective default configurations Shao et al. (2025); Mishchenko & Defazio (2024). SMARAN use $\gamma = 0.9$ and $\lambda = 0.01$.

All models are trained for 100 epochs with a batch size of 128. According to their preprocessing schemes, only standard normalization is performed on the CIFAR10 and CIFAR100 images. For Tiny ImageNet, we normalize the data with a mean of $[0.480, 0.448, 0.398]$ and a standard deviation of $[0.277, 0.269, 0.282]$. We perform data augmentation by padding four pixels on all sides and cropping to a fixed resolution of 64×64. To augment orientation diversity, horizontal flipping is performed. All model weights are randomly initialized. We use three evaluation metrics for comparison: Test accuracy, Test loss, and Mean Average Precision (mAP) on CIFAR10 and CIFAR100 datasets as shown in Fig. 1. For Tiny ImageNet data, we plot the generalization gap along with Test loss and mAP in Fig. 2. Apart from the optimizers mentioned above, we also experimented with other baselines that work based on the loss function, such as Stochastic Polyak Step-size Loizou et al. (2021a), POlyak NOnmonotone Stochastic (PoNoS) Galli et al. (2023), and sign-based Lion Chen et al. (2023) optimizer. Results of the experiments are shown in Appendix A.3.

5.2 Discussion

According to the results shown in Fig. 1, the SMARAN algorithm outperforms all models regarding test loss. Moreover, when comparing the test accuracy, SMARAN outperforms the state-of-the-art optimizers. SMARAN's training curve tends to stabilize in regions where overfitting is prevalent. Where other optimizers overfit the data after certain epochs, SMARAN is a perfect fit on the data, with testing loss either decreasing, as in the case of CIFAR10, or remaining stable. The reason behind this behaviour is the variable regularization parameter in our optimizer formulation. One can also see the same phenomenon for AdamW; however, for AdamW, the regularization is fixed, whereas for SMARAN, the adaptive learning rate parameter controls the regularization. The optimizer tries to reduce the gap between training loss and test loss, resulting in better generalization. Figs. 2b and 2e show the generalization gap, the difference between the testing and training loss of DenseNet121 and ResNet50 on the TinyImageNet dataset. The results show that the generalization gap is closer to zero for SMARAN compared to other optimizers. Generalization gap plots of other datasets are provided in the Appendix A.3. Since the accuracy curve does not reflect the proportionate improvement displayed by the loss curve, we use mAP as a complementary evaluation metric. mAP provides class-wise prediction confidence and enables a reliable comparison across different optimizers. The mAP values shown in Figs. 1 and 2 suggest that SMARAN performance

is better than other methods. Figs. 1h and 1k show that SMARAN performs better with more categories such as CIFAR100 over CIFAR10. A similar trend is seen with the TinyImageNet dataset in Figs. 2a and 2d. The results of the experiments on loss-based optimizers show that SMARAN is performing better than other optimizers. The performance gap is relatively low regarding accuracy on DenseNet121 with CIFAR100, which shows a scope for further improvements for larger models with diverse datasets.

6 Conclusion

In this work, we introduced a novel optimization method, SMARAN, that adapts the learning rate based on loss value instead of depending on the gradients. Our method provides a bounded learning rate, resulting in stable training and better generalization. Our variable regularization mechanism constrains the model from overfitting after reaching the optimal test results. Even though the model is not adaptive coordinate-wise, such as Adam and other adaptive methods, the learning rate still adapts to the curvature of the loss landscape using the exponential average of historical loss. Additionally, SMARAN is memory efficient compared to Adam-type methods since its learning rate is a scalar. Experimental results demonstrate the algorithm's effectiveness for vision-based models over adaptive methods. Future work includes extending the SMARAN optimizer for other domains like text and video processing.

References

Jinghui Chen, Dongruo Zhou, Yiqi Tang, Ziyan Yang, Yuan Cao, and Quanquan Gu. Closing the generalization gap of adaptive gradient methods in training deep neural networks. In *IJCAI*, pp. 3267–3275, 2020.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V Le. Symbolic discovery of optimization algorithms. In *NeurIPS*, volume 36, pp. 49205–49233, 2023.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12(7):2121–2159, 2011.

Leonardo Galli, Holger Rauhut, and Mark Schmidt. Don't be so monotone: Relaxing stochastic line search in over-parameterized models. In *NeurIPS*, 2023.

Rong Ge, Sham M Kakade, Rahul Kidambi, and Praneeth Netrapalli. The step decay schedule: A near optimal, geometrically decaying learning rate procedure for least squares. In *NeurIPS*, pp. 14977–14988, 2019.

Alex Graves, Abdel-Rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP*, pp. 6645–6649, 2013.

Elad Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2015.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, pp. 4700–4708, 2017.

D.P. Kingma and L.J. Ba. Adam: A method for stochastic optimization. In *ICLR*, pp. 13, 2015.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, Ontario, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, pp. 1097–1105, 2012.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *ICLR*, April 2020.

Nicolas Loizou, Sharan Vaswani, Issam Hadj Laradji, and Simon Lacoste-Julien. Stochastic polyak step-size for sgd: An adaptive learning rate for fast convergence. In *AISTATS*, pp. 1306–1314, 2021a.

Nicolas Loizou, Sharan Vaswani, Issam Hadj Laradji, and Simon Lacoste-Julien. Stochastic polyak step-size for sgd: An adaptive learning rate for fast convergence. In *AISTATS*, pp. 1306–1314, 2021b.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

Liangchen Luo, Yuanhao Xiong, and Yan Liu. Adaptive gradient methods with dynamic bound of learning rate. In *ICLR*, 2019.

Konstantin Mishchenko and Aaron Defazio. Prodigy: An expeditiously adaptive parameter-free learner. In *ICML*, volume 235, pp. 35779–35804, 2024.

Yurii Nesterov. A method for solving the convex programming problem with convergence rate $\mathcal{O}(1/k2)$. *Dokl Akad Nauk SSSR*, $269:543, 1983$.

Antonio Orvieto, Simon Lacoste-Julien, and Nicolas Loizou. Dynamics of sgd with stochastic polyak stepsizes: Truly adaptive variants and convergence to exact solution. In *NeurIPS*, pp. 26943–26954, 2022.

Boris Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4:1–17, 12 1964.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *ICLR*, 2018.

Zhou Shao, Hang Zhou, and Tong Lin. A new adaptive gradient method with gradient decomposition. *Machine Learning*, 114(7):155, May 2025.

Naum Zuselevich Shor. *Minimization methods for non-differentiable functions*. Springer Berlin, Heidelberg, 1985.

Leslie N. Smith. Cyclical learning rates for training neural networks. In *WACV*, pp. 464–472, 2017.

Tijmen Tieleman. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2):26, 2012.

Manzil Zaheer, Sashank Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar. Adaptive methods for nonconvex optimization. In *NeurIPS*, pp. 9815 – 9825, 2018.

Juntang Zhuang, Tommy Tang, Sekhar Tatikonda, Nicha C Dvornek, Yifan Ding, Xenophon Papademetris, and James S Duncan. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. In *NeurIPS Workshop: Deep Learning through Information Geometry*, 2020.

A Appendix

A.1 Proof of Theorem 3

Proof. The potential function is defined as

$$\phi = \|\mathbf{x}_{t+1} - \mathbf{x}^*\|^2 \tag{16}$$

$$\begin{aligned}\|\mathbf{x}_{t+1}-\mathbf{x}^*\| &= \|\mathbf{x}_t-\eta_t V_t(\mathbf{m}_t+\lambda\mathbf{x}_t)-\mathbf{x}^*\|^2\\ &= \|\mathbf{x}_t-\mathbf{x}^*\|^2+\eta_t^2V_t^2\|\mathbf{m}_t+\lambda\mathbf{x}_t\|^2-\eta_tV_t\langle\mathbf{m}_t+\lambda\mathbf{x}_t,\mathbf{x}_t-\mathbf{x}^*\rangle\\ &= \|\mathbf{x}_t-\mathbf{x}^*\|^2+\eta_t^2V_t^2\|\mathbf{m}_t+\lambda\mathbf{x}_t\|^2-\eta_tV_t\langle\mathbf{m}_t,\mathbf{x}_t-\mathbf{x}^*\rangle-\eta_tV_t\lambda\langle\mathbf{x}_t,\mathbf{x}_t-\mathbf{x}^*\rangle\end{aligned}$$

rearranging the terms,

$$\langle\mathbf{m}_t,\mathbf{x}_t-\mathbf{x}^*\rangle=\frac{\|\mathbf{x}_t-\mathbf{x}^*\|^2-\|\mathbf{x}_{t+1}-\mathbf{x}^*\|^2+\eta_t^2V_t^2\|\mathbf{m}_t+\lambda\mathbf{x}_t\|^2-\eta_tV_t\lambda\langle\mathbf{x}_t,\mathbf{x}_t-\mathbf{x}^*\rangle}{\eta_tV_t}$$

substitute $\mathbf{m}_t$ from Eq. 3,

$$\begin{aligned}\langle\gamma\mathbf{m}_{t-1}+\hat{\mathbf{g}}_\mathbf{t},\mathbf{x}_t-\mathbf{x}^*\rangle &= \frac{\|\mathbf{x}_t-\mathbf{x}^*\|^2-\|\mathbf{x}_{t+1}-\mathbf{x}^*\|^2+\eta_t^2V_t^2\|\mathbf{m}_t+\lambda\mathbf{x}_t\|^2-\eta_tV_t\lambda\langle\mathbf{x}_t,\mathbf{x}_t-\mathbf{x}^*\rangle}{\eta_tV_t}\\ \langle\hat{\mathbf{g}}_\mathbf{t},\mathbf{x}_t-\mathbf{x}^*\rangle &= \frac{\|\mathbf{x}_t-\mathbf{x}^*\|^2-\|\mathbf{x}_{t+1}-\mathbf{x}^*\|^2+\eta_t^2V_t^2\|\mathbf{m}_t+\lambda\mathbf{x}_t\|^2+\eta_tV_t\lambda\langle\mathbf{x}_t,\mathbf{x}_t-\mathbf{x}^*\rangle}{\eta_tV_t}\\ &\quad-\gamma\langle\mathbf{m}_{t-1},\mathbf{x}_t-\mathbf{x}^*\rangle\end{aligned}$$

substitute $\hat{\mathbf{g}}_t$ from Eq. 2 and for simplicity assume $\epsilon\approx 0$,

$$\begin{aligned}\langle\nabla f_t(\mathbf{x}_t),\mathbf{x}_t-\mathbf{x}^*\rangle = \|\nabla f_t(\mathbf{x}_t)\|\Bigg(&\frac{\|\mathbf{x}_t-\mathbf{x}^*\|^2-\|\mathbf{x}_{t+1}-\mathbf{x}^*\|^2}{\eta_tV_t}+\eta_tV_t\|\mathbf{m}_t+\lambda\mathbf{x}_t\|^2+\\ &\lambda\langle\mathbf{x}_t,\mathbf{x}_t-\mathbf{x}^*\rangle-\gamma\langle\mathbf{m}_{t-1},\mathbf{x}_t-\mathbf{x}^*\rangle\Bigg)\end{aligned}\tag{17}$$

From Eq. 14, regret is defined as

$$\begin{aligned}R(T) &= \sum_{t=1}^{T}f_t(\mathbf{x}_t)-\min_{\mathbf{x}\in X}\sum_{t=1}^{T}f_t(\mathbf{x}) &(18)\\ &= \sum_{t=1}^{T}f_t(\mathbf{x}_t)-\sum_{t=1}^{T}f_t(\mathbf{x}^*) &(19)\\ &= \sum_{t=1}^{T}(f_t(\mathbf{x}_t)-f_t(\mathbf{x}^*)) &(20)\\ &\le \sum_{t=1}^{T}\langle\nabla f_t(\mathbf{x}_t),\mathbf{x}_t-\mathbf{x}^*\rangle &(21)\end{aligned}$$

Therefore

$$\begin{aligned}R(T)\le\|\nabla f_t(\mathbf{x}_t)\|\Bigg(&\frac{\|\mathbf{x}_t-\mathbf{x}^*\|^2-\|\mathbf{x}_{t+1}-\mathbf{x}^*\|^2}{\eta_tV_t}+\eta_tV_t\|\mathbf{m}_t+\lambda\mathbf{x}_t\|^2+\\ &\lambda\langle\mathbf{x}_t,\mathbf{x}_t-\mathbf{x}^*\rangle-\gamma\langle\mathbf{m}_{t-1},\mathbf{x}_t-\mathbf{x}^*\rangle\Bigg)\end{aligned}\tag{22}$$

Simplifying each term in Eq. 22 and substitute $\eta_t = \eta/\sqrt{t}$.

$$\sum_{t=1}^{T} \|\nabla f_t(\mathbf{x}_t)\| \left(\frac{\|\mathbf{x}_t - \mathbf{x}^*\|^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^*\|^2}{\eta_t V_t} \right) \tag{23}$$

$$= \sum_{t=1}^{T} \|\nabla f_t(\mathbf{x}_t)\| \sqrt{t} \left(\frac{\|\mathbf{x}_t - \mathbf{x}^*\|^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^*\|^2}{\eta V_t} \right) \tag{24}$$

$$\leq \frac{G}{\eta V} \sum_{t=1}^{T} \sqrt{t} \left(\|\mathbf{x}_t - \mathbf{x}^*\|^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^*\|^2 \right) \tag{25}$$

$$\leq \frac{G}{\eta V} \left(\|\mathbf{x}_1 - \mathbf{x}^*\|^2 + \sum_{t=2}^{T} \sqrt{t} \|\mathbf{x}_t - \mathbf{x}^*\|^2 - \sum_{t=2}^{T} \sqrt{t-1} \|\mathbf{x}_t - \mathbf{x}^*\|^2 \right) \tag{26}$$

$$= \frac{GD^2}{\eta V} \left(n + \sum_{t=2}^{T} (\sqrt{t} - \sqrt{t-1}) \right) \tag{27}$$

$$\leq \frac{GD^2\sqrt{T}}{\eta V} \tag{28}$$

Similarly,

$$\sum_{t=1}^{T} \|\nabla f_t(\mathbf{x}_t)\| \left(\lambda \langle \mathbf{x}_t, \mathbf{x}_t - \mathbf{x}^* \rangle \right) \tag{29}$$

$$\leq G\lambda D^2 \tag{30}$$

$$\tag{31}$$

For the remaining part,

$$\sum_{t=1}^{T} \|\nabla f_t(\mathbf{x}_t)\| \left(\eta_t V_t \|\mathbf{m}_t + \lambda \mathbf{x}_t\|^2 - \gamma \langle \mathbf{m}_{t-1}, \mathbf{x}_t - \mathbf{x}^* \rangle \right) \tag{32}$$

$$\leq \sum_{t=1}^{T} \|\nabla f_t(\mathbf{x}_t)\| \left(\eta_t V_t \|\mathbf{m}_t + \lambda \mathbf{x}_t\|^2 \right) \tag{33}$$

$$\leq G\eta \sum_{t=1}^{T} \frac{\|\mathbf{m}_t + \lambda \mathbf{x}_t\|^2}{\sqrt{t}} \tag{34}$$

$$\leq G\eta (G + \lambda D)^2 \sum_{t=1}^{T} \frac{1}{\sqrt{t}} \tag{35}$$

$$\leq G\eta (G + \lambda D)^2 (2\sqrt{T} - 1) \tag{36}$$

From Eq. 22, 28, 31 and 36 we write

$$R(T) \leq \frac{GD^2\sqrt{T}}{\eta V} + G\lambda D^2 + G\eta (G + \lambda D)^2 (2\sqrt{T} - 1) \tag{37}$$

□

A.2 Proof of constant learning rate for constant loss

Given

$$V_t = \frac{f(\mathbf{x}_t)}{\sqrt{v_t} + \epsilon} \tag{38}$$

Assume $f(\mathbf{x}_t) = c$ be constant and for simplicity assume $\epsilon \approx 0$. Then from Eq. 9,

$$v_t = \gamma v_{t-1} + c^2 \tag{39}$$

solution for the above recurrence relation is

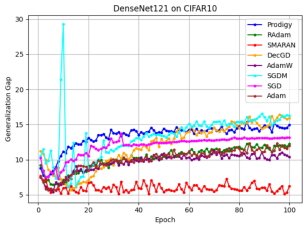
$$v_t = \frac{c^2}{1-\gamma} \tag{40}$$

substitute into Eq.38, we get

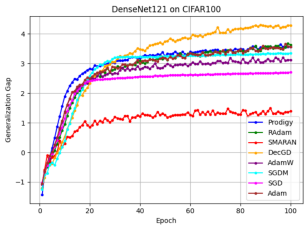
$$V_t = \frac{c}{\sqrt{\frac{c^2}{1-\gamma}}} \tag{41}$$

$$= \sqrt{1-\gamma} \tag{42}$$

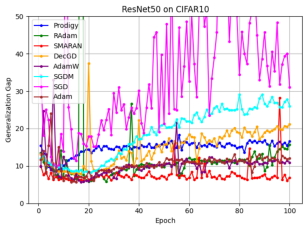
A.3 Additional Results



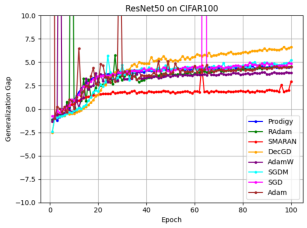
(a) Generalization gap of DenseNet121 on CIFAR10



(b) Generalization gap of DenseNet121 on CIFAR100

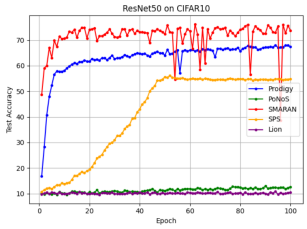


(c) Generalization gap of ResNet50 on CIFAR10

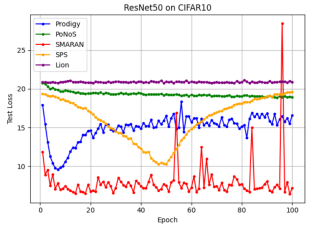


(d) Generalization gap of ResNet50 on CIFAR100

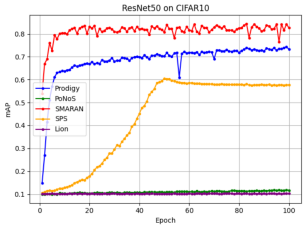
Figure 3: Generalization gaps on CIFAR10 and CIFAR100.



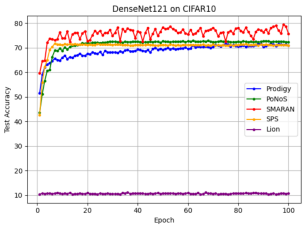
(a) Test accuracy of ResNet50 on CIFAR10



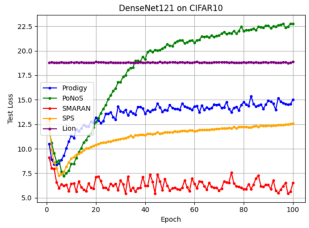
(b) Test Loss of ResNet50 on CIFAR10



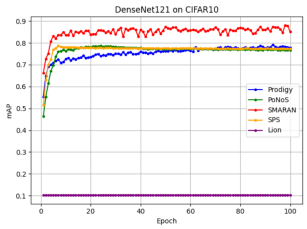
(c) mAP of ResNet50 on CIFAR10



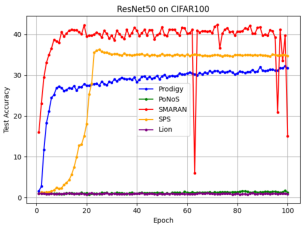
(d) Test accuracy of DenseNet121 on CIFAR10



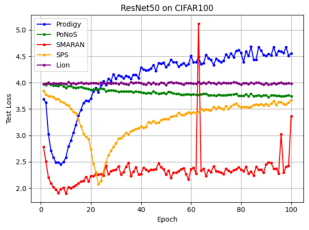
(e) Test Loss of DenseNet121 on CIFAR10



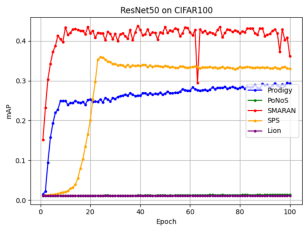
(f) mAP of DenseNet121 on CIFAR10



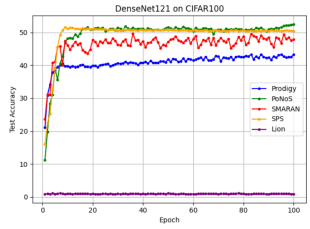
(g) Test accuracy of ResNet50 on CIFAR100



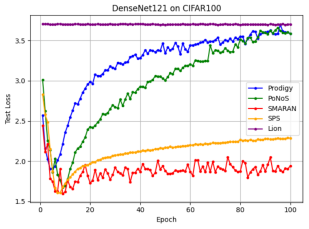
(h) Test Loss of ResNet50 on CIFAR100



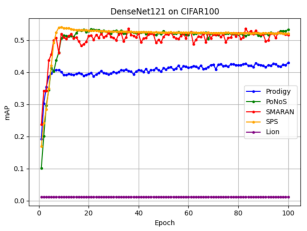
(i) mAP of ResNet50 on CIFAR100



(j) Test accuracy of DenseNet121 on CIFAR100



(k) Test Loss of DenseNet121 on CIFAR100



(l) mAP of DenseNet121 on CIFAR100

Figure 4: Experimental Results on CIFAR10 and CIFAR100 datasets for loss based models.