Anusandhanvallari
Vol 2024, No.1
February 2024
ISSN 2229-3388

Hybrid Switching Optimization Strategy for Efficient Training of Deep Neural Networks on the MNIST Dataset

Harish Kunder[1] & Manjunath Kotari[2]

[1]Department of Artificial Intelligence and Machine Learning, Alva's Institute of Engineering and Technology, Moodbidri, VTU Belagavi, India.

[2]Department of Computer Science and Engineering, Alva's Institute of Engineering and Technology, Moodbidri, VTU Belagavi, India.

Abstract: Deep neural networks often encounter non-convex optimization challenges during training due to the presence of local minima, saddle points, and complex loss surfaces. Existing optimization algorithms such as Adam and Stochastic Gradient Descent (SGD) offer complementary advantages—Adam provides faster convergence, while SGD tends to achieve better generalization. However, neither optimizer alone effectively balances both properties in non-convex settings. To address this limitation, this paper proposes a phase-switch hybrid optimization strategy that combines the strengths of Adam and SGD. The proposed method employs Adam during the initial phase of training to enable rapid convergence and efficient exploration of the loss landscape, and then switches to momentum-based SGD in the later phase to improve generalization and ensure stable convergence. The effectiveness of the proposed approach is evaluated on one benchmark dataset, MNIST dataset, under different learning rate settings. Experimental results demonstrate that the proposed method achieves performance that is superior or comparable to existing optimizers in terms of accuracy and loss minimization. These results indicate that the proposed hybrid optimization strategy provides a simple and effective solution for handling non-convex optimization problems in deep learning.

Index Terms: Adam, Deep learning, Hybrid optimizer, Neural networks, Non-convex optimization, Phase-switch strategy, Stochastic Gradient Descent.

I. INTRODUCTION

Deep learning is one of the most promising technologies in the field of artificial intelligence, and has shown great promise in computer vision, speech recognition, natural language processing, recommendation systems, and time-series forecasting. Deep neural networks are able to learn complex representations from large-scale data sets, which has led to significant improvements in the performance of modern machine learning systems. However, they are not only relying on large data sets and powerful hardware like Graphics Processing Units (GPUs), they also rely on efficient optimization algorithms which can facilitate stable and effective training of deep neural networks [1]. However, optimization is one of the most difficult problems in deep learning [2].

The aim of training a neural network is to reach the lowest possible value of a loss function using an optimal set of model parameters. This optimization problem can be formulated as (1)

$$\theta^* = \arg\min_{\theta} L(\theta; X, Y) \quad (1)$$

Here, θ are the trainable parameters, X is the input data, Y is the target labels, and L is the loss function. The optimization process aims at finding values for the parameters that lead to

low prediction error and good generalization.

Convex optimization problems have loss landscapes with only one global minimum, whereas deep neural networks have loss landscapes that are highly non-convex with multiple local minima, saddle points and flat regions, making optimization much harder [3]. This can be expressed as a property of the set of all matrices of a certain size shown in (2).

$$\nabla L(\theta_1) = 0, \qquad \nabla L(\theta_2) = 0, \qquad \theta_1 != \theta_2 \qquad (2)$$

where θ_1 and θ_2 are two different stationary points in the parameter space. However, saddle points are more common than local minima in practice and often lead to slower convergence during the training [4].

Most Deep Learning Models use Gradient Based Optimisation Algorithms that update model parameters based on (3)

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta L(\theta_t) \qquad (3)$$

where η is the learning rate. Despite its intuitive simplicity, the success of this update rule relies on the choice of appropriate optimization algorithms and hyperparameters like the learning rate, momentum, and batch size. In addition, the performance of optimizing different neural network architectures is not directly comparable for the same data set and determining an appropriate optimizer becomes more complex. Popular optimizers include SGD, Adam, RMSProp, and AdaGrad. Adam offers fast convergence, whereas SGD with momentum provides better generalization despite slower convergence.

This complementarity is used to propose a two-phase scheme called *Phase-Switch* that switches between the two methods in different epochs: Adam is used in the first to ensure a quick early convergence; and SGD with momentum is used subsequently to improve generalization.

The important contributions of this work are:

- A hybrid Adam–SGD-with-momentum optimization method.
- Switching mechanism based on epochs for stable and reproducible training. A quick overview of existing optimizers and their drawbacks. Experimental results on MNIST with FCN, CNN and LSTM models [21].

II. Review

Minimizing highly non-convex objective functions is one of the most important research directions in the field of machine learning and deep learning, and optimization has become an essential component of the algorithms used in these applications. In recent years, several different optimization algorithms have been suggested to speed up convergence, stabilize computed models and enhance model performance. This paper reviews the current state of the art optimization methods for deep learning.

A. Foundations of Optimization in Machine Learning

Mathematical theories of optimization have been widely addressed in the classical literature. Boyd and Vandenberghe

provided a comprehensive theory of convex optimization, demonstrating how convex optimization could guaranty a unique optimal solution [1]. Nevertheless, it is noted that most problems in machine learning are non-convex, hence making the problem much harder to solve.

The creation of neural networks was enabled by the development of the backpropagation technique by Rumelhart et al. [2]. Backpropagation proved critical in enabling the effective computation of gradients in multi-layered networks. Recent advances in gradient descent learning have been made by LeCun et al. [3], where efficient gradient descent is involved in neural networks.

B. Gradient-Based Optimization Algorithms

Most deep learning models are based on gradient-based optimization techniques. Bottou et al. presented an extensive review of optimization algorithms in large-scale machine learning with a special focus on stochastic gradient descent that is required in the case of massive data sets [4]. Ruder reviewed various gradient descent techniques along with their modifications, such as momentum, learning schedule, and adaptation of the learning rate [5].

The stochastic gradient descent was theoretically analyzed by Gower et al., and extensions were suggested to accelerate its convergence rate in large-scale optimization problems [6].

C. *Adaptive Optimization Methods*

There has been a tremendous increase in popularity in adaptive optimization algorithms because of their ability to adapt learning rates throughout the training process. Adam is one of the most famous adaptive optimization algorithms, which was proposed by Kingma and Ba [7]. Adam is a learning optimization algorithm that employs moment estimation as well as adaptive learning rate estimation.

Although Adam has proved to be very popular, many scholars have endeavored to find out its shortcomings. The convergence properties of Adam were examined by Reddi et al. [8], and several improvements to Adam were suggested. Chen et al. [9] further explored the research on Adam and its convergence properties in stochastic optimization scenarios. Nazari et al. proposed an adaptive optimization technique called DADAM [10].

Other research has delved deeper into enhancements to the optimization technique used for adaptability. Chen et al. [27] conducted a literature review on adaptive gradient algorithms, while Loshchilov and Hutter [24] suggested AdamW, Recent advancements in optimization methods have also focused on improving both convergence speed and stability.

Challenges in Non-Convex Optimization

In the case of deep learning optimization problems, the problems are non-convex in nature, i.e., the loss function may have many local minima, saddle points, and flat regions. An extensive study of non-convex optimization techniques in the field of machine learning was provided by Jain and Kar, discussing the limitations of conventional optimization techniques in handling large-scale parameter spaces [11].

Dauphin et al. proved that saddle points are much denser than local minima in the loss function of neural network problems in high-dimensional space [12]. These saddle points lower the convergence of conventional optimization algorithms because the gradients become extremely small in the vicinity of the saddle points. Razaviyayn et al. studied non-convex min-max optimization problems, which frequently appear in adversarial learning [13]. Nouiehed et al. studied conven-tional optimization algorithms for non-convex game-theoretic problems [14]. Xu et al. proposed second-order optimization techniques that can effectively escape saddle points [15].

D. Distributed and Large-Scale Optimization

As the models used in deep learning are becoming more complex, distributed optimization methods have become a necessity for handling such complex training processes. Chang et al. researched distributed learning frameworks [16]. Kurach et al. carried out a large-scale study on optimization methods used for generative adversarial networks (GANs) [17].

E. *Alternative and Evolutionary Optimization Techniques*

In addition to gradient-based optimization techniques, heuristic optimization techniques have also been studied.

Particle swarm optimization (PSO) is one such technique. The efficiency of PSO in exploring complex search spaces has been demonstrated by Biswas et al.; however, such techniques are computationally expensive in deep learning contexts [18].

TABLE I COMPARISON OF EXISTING OPTIMIZATION METHODS

Method	Strength	Limitation
SGD	Strong generalization	Slow convergence
Adam	Fast convergence	Poor generalization
AdamW	Better regularization	Still adaptive bias
SWATS	Adam-to-SGD transition	Heuristic switching
Hybrid Methods	Balance convergence and accuracy	No structured switching strategy
Proposed	Adam (early) + SGD (later)	Requires switching epoch

I. Research Gap

Despite the numerous optimization algorithms that have been proposed, several challenges remain unresolved. Adaptive optimizers ensure faster convergence but often suffer from poor generalization performance. In contrast, SGD and other conventional optimizers provide better generalization but require careful hyperparameter tuning and slower convergence.

F. Hybrid and Learned Optimization Strategies

Kashyap [19] classified optimizers according to their characteristics. Abdulkadirov et al. [20] further emphasized the importance of hybrid optimization methods in neural networks.

Recent IEEE-based studies [28][29] also indicate that combining adaptive and stochastic optimization strategies can significantly improve training efficiency and generalization performance.

Zhou et al. [26] further explored hybrid optimization by combining adaptive and SGD-based methods.

G. Comparative Analysis of Optimization Algorithms

III. METHODOLOGY

The new way of optimizing things is really good at getting to the answer and it also works well for many different situations. This process has a lot of steps: we get the data ready we start with a model we go through the data we see how wrong we are we go back and fix the model we use a way to make the model better and then we see how well it is working. We use Adam first because it is good at looking and finding the best path quickly. Then after a while we switch to something called SGD with momentum because it is better at making sure the model is stable and works well. You can see all the steps, in Figure 1.

A. Overall Framework

A fully connected network with constant weights and biases is first initialized and then the magnitude of the pixel intensities in the MNIST data is normalized to the range [0, 1]. For each iteration, the input samples are passed forward and class probabilities are calculated, and the prediction error is calculated by categorical cross entropy loss; the gradients are calculated by back propagation. In the first part of the training, Adam is used for fast convergence, and in the second part of the training, SGD with momentum is used to optimize to flatter, better generalizing minima when the predefined switching epoch is reached. The training process is repeated until convergence or until the maximum number of epochs is reached, and then the unseen MNIST test set is used to evaluate the model with the accuracy of classification and the minimum test loss.

B. Optimization Problem Formulation

Let the training dataset be represented as (4)

Table I compares existing optimization methods and highlights their limitations in non-convex problems. While SGD

$$D = \{(x_i, y_i)\}_{i=1}^{N} \quad (4)$$

provides good generalization but slow convergence, adaptive methods like Adam achieve faster convergence but may lead to poor generalization. Recent hybrid approaches attempt to combine these benefits but often lack a structured mechanism. The proposed phase-based optimizer addresses this by using Adam for fast convergence in the initial stage and SGD for improved generalization in the later stage, achieving a better balance between performance and stability.

where x_i denotes an input image and y_i is its corresponding class label.

The objective of training is to determine the optimal network parameters that minimize the empirical loss over the training dataset.

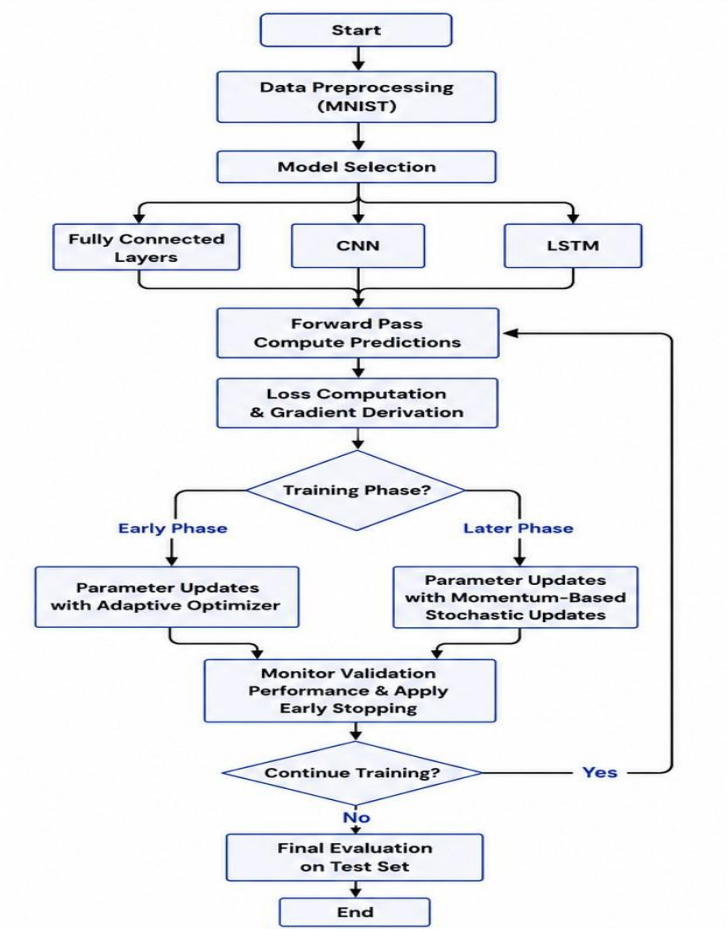


Figure 1. Overall workflow of the proposed phase-switch hybrid optimization framework.

Algorithm 1 Phase-Switch Hybrid Optimization

1: Initialize network parameters θ

2: **for** epoch $t = 1$ to T **do**

3: Forward propagation

4: Compute cross-entropy loss

5: Compute gradients using backpropagation

6: **if** $t < T_s$ **then**

7: Update parameters using Adam

8: **else**

9: Update parameters using SGD with momentum

10: **end if**

11: **end for**

12: **Return** optimized parameters θ^*

TABLE II TRAINING CONFIGURATION

Parameter	Value
Dataset	MNIST
Model	Fully Connected Network
Batch Size	128
Learning Rate	0.0001–0.01
Momentum	0.9
Epochs	10
Optimizer	Adam → SGD

SGD with momentum is used during the remaining epochs to refine the learned parameters and improve generalization.

D. Training Configuration

The proposed optimizer is evaluated using a fully connected neural network trained on the MNIST dataset. Mini-batch gradient optimization is employed throughout training. The experimental settings are summarized in Table II.

IV. RESULTS AND DISCUSSION

To evaluate the proposed Hybrid optimizer, the optimization ability of three optimization algorithms, namely Adam, SGD, and the proposed Hybrid optimizer is compared in the MNIST

C. Hybrid Phase-Switch Optimization

The proposed optimizer combines the complementary strengths of Adam and SGD with momentum. Adam is utilized during the early training stage because its adaptive learning rates accelerate convergence and efficiently explore the optimization landscape. As training progresses, the optimizer switches to SGD with momentum, which provides more stable parameter updates and encourages convergence toward flatter minima, thereby improving generalization performance. Algorithm 1 summarizes the complete optimization procedure.

The switching epoch is defined as (5)

$$T_s = \alpha T, \tag{5}$$

where T denotes the total number of training epochs and $\alpha \in (0, 1)$ specifies the transition point between the two optimization phases. In this work, α is fixed at 0.8, indicating that Adam is employed for the first 80% of training, while

handwritten digit classification task. The input layer comprised 784 neurons, followed by two dense layers with 256 and

128 neurons, respectively, and using ReLU and Softmax activations, respectively, and an output layer with 10 neurons and using Softmax activation. Experiments were conducted for a small number of epochs to highlight the convergence and stability properties, rather than the best generalization performance, and each experiment was repeated three times with different initializations, and the average across the three experiments reported.

Table III provides a summary of the performance over the learning rate. In the case of a high learning rate (0.01), SGD gives the best results with accuracy of 97.76% and the least test loss (0.0690), whereas Adam gives the worst results. The Hybrid optimizer has the best accuracy in the intermediate learning rates of 0.002 and 0.001, achieving 98.02

In Table IV we present the range of test loss values obtained by each optimizer for the different learning rates tested.

TABLE III ACCURACY AND MNIST MINIMUM TEST LOSS

Learning Rate	Optimizer	Accuracy (%)	Min Test Loss
0.01	SGD	97.76	0.0690
	Hybrid	94.41	0.1953
	Adam	92.03	0.2632
0.002	Hybrid	98.02	0.0682
	Adam	97.65	0.0758
	SGD	95.47	0.1452
0.001	Hybrid	97.96	0.0649
	Adam	97.53	0.0797
	SGD	93.39	0.2187
0.0001	Adam	96.22	0.1192
	Hybrid	95.71	0.1388

	SGD	82.43	0.6931
0.0005	Hybrid	97.85	0.0677
	Adam	97.48	0.0785
	SGD	91.47	0.2914

TABLE IV THE LOSS STABILITY COMPARISON OF OPTIMIZERS ON MNIST

Optimizer	Min Loss	Max Loss	Loss Range	Conclusion
Adam	0.0758	0.2632	0.1874	Stable, but slightly less consistent than Hybrid.
SGD	0.0690	0.6931	0.6241	Highly unstable, poor convergence at low learning rates.
Hybrid	0.0649	0.1953	0.1304	Lowest loss variation, most consistent convergence.

The Hybrid optimizer has the smallest range of loss values indicating that its performance is not highly sensitive to the choice of learning rate, whereas the SGD optimizer has the largest range of loss values.

loss. The Hybrid optimizer optimizes Adam's trajectory up to epoch 8, but then changes to SGD-style optimizations and can reach a lower final loss than Adam or SGD alone.

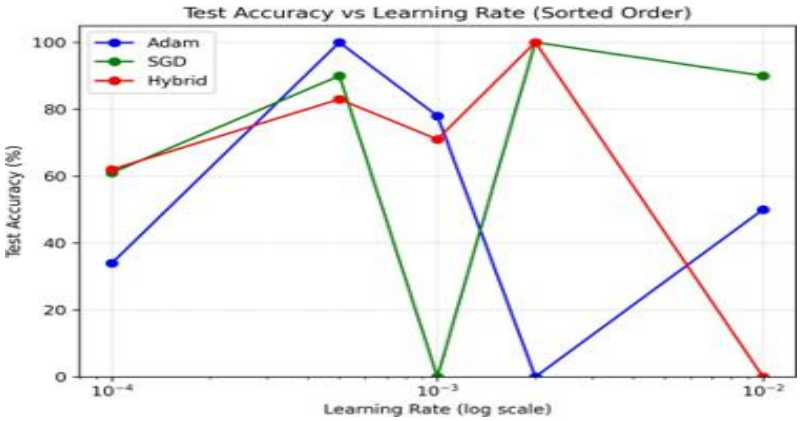


Figure 3. MNIST: best test accuracy as a function of learning rate for Adam, SGD and Hybrid optimizers.

The accuracy as a function of learning rate is shown in Figure 3. While the learning rate is in the range of 0.001–0.002, the accuracy reported by all three optimizers is within 97–98

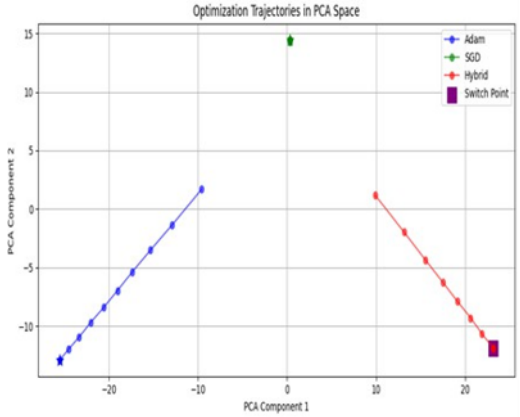
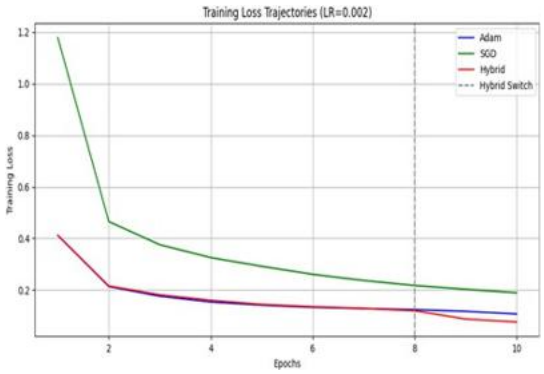


Figure 4. Principal Component Analysis (PCA) based optimization trajectories of Adam, SGD, and the Hybrid optimizer on the MNIST dataset

Optimization trajectories in reduced parameter space using PCA are visualized in Figure 4. The Hybrid optimizer's trajectory is very similar to Adam's in the early part, and then similar to SGD's in the later part, confirming the intended two phase trajectory of the optimizer: fast proximal exploration at the beginning coupled with stable refinement to flatter minima in the end.



igure 2. Training loss trajectories for Adam, SGD, and Hybrid optimizers on the MNIST dataset (learning rate =0.002).

With a learning rate of 0.002 the training loss is displayed in Figure 2. In the beginning of training, the learning-rate mechanism of the Hybrid optimizer causes it to converge rapidly, while SGD is slow to converge and has the largest

V. Conclusion

The proposed Phase-Switch Hybrid optimization strategy is able to overcome non-convex optimization issues in deep learning by merging the fast early convergence of Adam with stable, well generalizing updates of SGD with momentum. The Hybrid optimizer demonstrated the highest test loss (best overall accuracy) on the MNIST handwritten digit classification task at a learning rate of 0.002, and performed best over different learning rates when compared against Adam and SGD individually. The results show that switching by epochs, using a structured epoch-based switching mechanism, can improve convergence speed and generalization. An adaptive switching criterion and evaluation on larger-scale data and current architectures like Transformers will be investigated in the future.

REFERENCES

[1] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[3] Y. LeCun, L. Bottou, G. B. Orr, and K.R. Mu¨ller, "Efficient backpropagation," in *Neural Networks: Tricks of the Trade*, 1998.

[4] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Rev.*, vol. 60, no. 2, pp. 223–311, 2018.

[5] S. Ruder, "An overview of gradient descent optimization algorithms," 2016.

[6] R. M. Gower et al., "SGD: General analysis and improved rates," in *Proc. ICML*, 2019.

[7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.

[8] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of Adam and beyond," in *Proc. ICLR*, 2018.

[9] X. Chen et al., "On the convergence of Adam-type algorithms," in *Proc. ICLR*, 2019.

[10] P. Nazari et al., "DADAM: Distributed adaptive gradient methods," 2019.

[11] P. Jain and P. Kar, "Non-convex optimization for machine learning," 2017.

[12] Y. N. Dauphin et al., "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization," in *Proc. NeurIPS*, 2014.

[13] M. Razaviyayn, M. Hong, and Z.-Q. Luo, "A unified convergence analysis of block successive minimization methods for nonsmooth optimization," *IEEE Signal Process. Mag.*, 2020.

[14] M. Nouiehed et al., "Solving a class of non-convex min-max games using first-order methods," in *Proc. NeurIPS*, 2019.

[15] P. Xu et al., "Second-order optimization methods for non-convex machine learning," in *Proc. SDM*, 2020.

[16] T. Chang et al., "Distributed learning in the nonconvex world," *IEEE Signal Process. Mag.*, 2020.

[17] K. Kurach et al., "A large-scale study on regularization and normalization in GANs," in *Proc. ICML*, 2019.

[18] K. Biswas et al., "Particle swarm optimization in engineering applications," *Eng. Optim.*, 2020.

[19] R. Kashyap, "A survey of deep learning optimizers," arXiv:2211.15596, 2022.

[20] R. Abdulkadirov, P. Lyakhov, and N. Nagornov, "Survey of optimization algorithms in neural networks," 2023.

[21] MNIST Dataset. [Online]. Available: https://www.kaggle.com/datasets/hojjatk/mnist-dataset

[22] N. S. Keskar and R. Socher, "Improving generalization performance by switching from Adam to SGD," arXiv:1712.07628, 2017.

[23] X. Tian et al., "Improved deep learning optimization algorithm based on variable speed integral control," in *Proc. Chinese Control Conf.*, 2021.

[24] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. ICLR*, 2019.

[25] M. Zaheer et al., "Adaptive methods for nonconvex optimization," in *Proc. NeurIPS*, 2018.

[26] P. Zhou et al., "MAS: Mixing adaptive and stochastic gradient descent optimization methods," arXiv:2011.08042, 2020.

[27] X. Chen et al., "Adaptive gradient methods in deep learning: A review," *IEEE Access*, 2023.

[28] V. Fotopoulos et al., "Optimization in machine learning: A comprehensive survey," *IEEE Access*, 2022.

[29] A. Abdulkadirov et al., "Review of optimization algorithms in deep learning," *IEEE Access*, 2023.