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

Hybrid Optimizer Switching for Deep Neural Network Training in Time Series Forecasting

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 three benchmark dataset the M4 time-series forecasting data set, 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

Time-series forecasting has become an essential component of decision-making in many application domains, including finance, healthcare, transportation, energy management, retail, and manufacturing. Accurate forecasts enable organizations to anticipate future trends, allocate resources efficiently, and respond proactively to changing conditions. With the growing availability of historical data and advances in computational capabilities, deep learning models have gained considerable attention for their ability to capture complex temporal patterns that are difficult to model using conventional statistical approaches [1].

Unlike traditional forecasting methods that rely on predefined assumptions about the underlying data, deep neural networks learn feature representations directly from historical observations. This capability allows them to model nonlinear relationships and long-term dependencies that frequently occur in real-world time series. Architectures such as recurrent neural networks, long short-term memory networks, gated recurrent units, and more recent deep learning models have demonstrated promising performance across a wide variety of forecasting tasks [2], [3]. Nevertheless, obtaining accurate predictions depends not only on the network architecture but also on how effectively the model is trained.

Training a deep neural network is an iterative optimization process in which model parameters are updated to reduce prediction errors. As network complexity increases, the optimization landscape becomes more difficult to navigate because of numerous local optima, saddle regions, and flat surfaces. These characteristics may slow

convergence, increase computational cost, and lead to models that perform well on training data but fail to generalize to unseen observations [4], [5]. Therefore, the choice of optimization algorithm has a direct influence on both training efficiency and forecasting accuracy.

Gradient-based optimization methods remain the standard approach for training deep learning models. Stochastic Gradient Descent (SGD) is widely recognized for producing models with strong generalization performance, particularly when combined with momentum. However, its convergence during the early stages of training can be relatively slow. Adaptive optimization algorithms such as Adam overcome this limitation by automatically adjusting learning rates for individual parameters, enabling faster progress during the initial training epochs and reducing the need for extensive hyperparameter tuning [6], [7]. Despite these advantages, adaptive methods do not consistently maintain the same level of generalization as SGD-based optimization after extended training.

The differing characteristics of these optimization algorithms indicate that a single optimizer may not be equally effective throughout the entire learning process. Fast parameter updates are desirable during the initial stages to accelerate learning, whereas more stable updates become increasingly important as training approaches convergence. Integrating these complementary strengths within a single training strategy provides an opportunity to improve both convergence behaviour and predictive performance without increasing model complexity.

Motivated by this observation, this study presents a hybrid optimization strategy that combines Adam and SGD with momentum during different stages of model training. Adam is employed during the initial learning stage to achieve rapid convergence and efficient exploration of the parameter space. Subsequently, the optimization process transitions to SGD with momentum to refine the learned parameters and improve tmodel's generalization capability. This sequential optimization strategy aims to balance learning speed and model stability while maintaining a straightforward implementation that can be incorporated into existing deep learning workflows.

The proposed approach is evaluated using the M4 benchmark dataset [28], which contains diverse real-world time series collected from multiple application domains and is widely used for assessing forecasting algorithms. Experimental results demonstrate that the proposed optimization strategy provides stable convergence and competitive forecasting performance compared with commonly used optimization methods. The findings suggest that combining complementary optimization techniques within a unified training process can improve the effectiveness of deep learning models for time series forecasting.

II. Literature Survey

Optimization remains a central research challenge in machine learning due to the complexity of minimizing highly non-convex objective functions. This section reviews existing optimization techniques used in deep learning.

Boyd and Vandenberghe established the classical theory of convex optimization, though most machine learning problems remain non-convex and thus harder to solve [1]. Backpropagation, introduced by Rumelhart et al. [2], enabled effective gradient computation in multi-layered networks, with efficiency improvements further studied by LeCun et al. [3].

Bottou et al. [4] reviewed large-scale stochastic gradient descent, while Ruder [5] surveyed its modifications, including momentum and learning-rate adaptation. Gower et al. [6] further analyzed SGD convergence and proposed rate accelerating extensions.

Adam, proposed by Kingma and Ba [7], is among the most popular adaptive optimizers, combining moment estimation with adaptive learning rates. Subsequent work examined its shortcomings and proposed improvements, including convergence analyses by Reddi et al. [8] and Chen et al. [9], the DADAM

method by Nazari et al. [10], a review of adaptive gradient algorithms by Chen et al. [25], AdamW by Loshchilov and Hutter [23].

Jain and Kar [11] reviewed the limitations of conventional optimization techniques in large-scale, non-convex parameter spaces. Dauphin et al. [12] showed that saddle points are far more prevalent than local minima in high-dimensional loss surfaces and slow convergence via near-zero gradients, while Razaviyayn et al. [13] and Nouiehed et al. [14] studied non-convex min-max and game-theoretic optimization problems. Xu et al. [15] proposed second-order methods to escape saddle points effectively.

As deep learning models grow more complex, distributed optimization has become increasingly necessary. Chang et al.

[16] studied distributed learning frameworks, and Kurach et al.

[17] conducted a large-scale study of optimization methods for GANs.

Beyond gradient-based approaches, heuristic techniques such as particle swarm optimization have also been explored. Biswas et al. [18] demonstrated strong search space exploration despite high computational cost in deep learning contexts.

Kashyap [19] classified optimizers by their characteristics. Abdulkadirov et al. [20] and recent IEEE based studies [26], [27] further emphasized that combining adaptive and stochastic strategies can improve training efficiency and generalization. Zhou et al. [24] combined adaptive and SGD based methods.

III. Methodology

The proposed methodology is capable of overcoming the issues faced in non-convex optimization for DNNs. A hybrid optimization technique is proposed to improve the speed of convergence and enhance the performance of the DNNs. The methodology for the proposed approach includes data preprocessing, model selection, hybrid optimization, and performance evaluation. The training process is divided into two phases. In the initial phase, Adam optimizer is used to accelerate convergence and explore the loss landscape efficiently. In the later phase, SGD with momentum is employed to improve generalization and achieve stable convergence. The workflow of the proposed methodology is shown in

A. Overall Framework

Figure 1 shows the general workflow of the hybrid optimization methodology proposed in the article. The overall workflow is divided into several steps, starting with data set preprocessing and model selection, followed by forward propagation, loss computation, and gradient computation. The model is then trained by employing a hybrid optimization methodology that is divided into two phases, depending on the training stage, and finally, performance evaluation is carried out on the trained model.

The proposed optimization framework starts with initializling the model parameters θ by randomly sampling from a distribution, and assigns fixed values to the biases to guarantee a stable and symmetry-free way to train the model. To enable the learning of sequences, the raw M4 Weekly time series data is preprocessed through scaling and dividing the values into

windows, and a Long Short-Term Memory (LSTM) network is chosen to learn temporal patterns from the data. In training, the input windows are fed forward through the network to generate predictions $y\hat{}$, and the mean squared error (MSE) loss is calculated based on the actual values. Then, gradients of the loss with respect to all parameters are computed using backpropagation.

The optimizer is therefore either Adam to achieve fast adaptive convergence in the early phase of the training, or SGD with momentum to ensure that the training is stable and that the model moves towards more flat and well-

generalizing minima in the latter phase of training, like described in Fig. 1. Parameters are optimized iteratively, while monitoring for convergence based on validation performance, and stopping

early if no further improvement is seen. After finding optimal parameters θ^*, the model is tested on test data that was not used in the optimization with forecasting error as the measure of success.

B. *Optimization Problem Formulation*

The training of a DNN can be formulated as an optimization problem, in which the aim is to optimize the loss function over the training data. The dataset used for training is represented as shown in Equation 1.

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

where x_i and y_i denote the features and labels of the data points, respectively.

The DNN is parameterized by $\theta \in R^d$. The objective of the training process is to determine the optimal parameters that minimize the empirical loss over the dataset. This optimization problem is formulated in Equation 2.

$$\theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \ell\,(f(x_i; \theta), y_i) \quad (2)$$

where $f(x_i; \theta)$ represents the prediction of the DNN and

$\ell(\cdot)$ denotes the loss function. Figure 1.

C. *Loss Functions*

Two different loss functions are used depending on the task.

1) Cross-Entropy Loss: For classification tasks, the cross-entropy loss function is used, which is defined in Equation 3.

$$\ell = -\sum_{k=1}^{K} y_{i,k} \log(\hat{y}_{i,k}) \quad (3)$$

2) Mean Squared Error: For regression or forecasting tasks, the mean squared error (MSE) loss is used, as expressed in Equation 4.

$$\ell = \frac{1}{T} \sum_{t=1}^{T} (y_t - \hat{y}_t)^2 \quad (4)$$

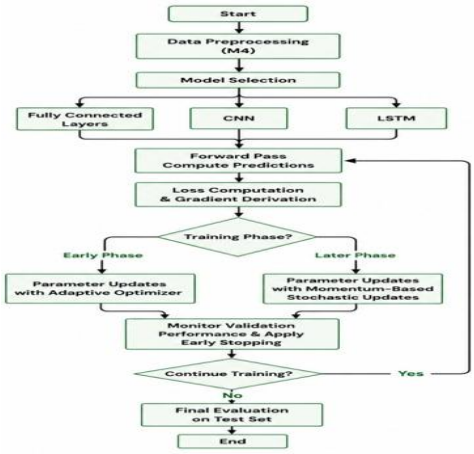

Fig. 1: Hybrid optimization workflow illustrating the training process with adaptive and stochastic optimization phases.

D. Hybrid Optimization Algorithm

Algorithm 1 describes a hybrid optimizer that takes advantage of Adam and SGD with momentum optimizers. The Adam optimizer is used to rapidly explore the loss surface and accelerate the learning process. Afterwards, the optimizer switches to the SGD with momentum optimizer to ensure stable convergence and good generalization performance.

Algorithm 1: Phase-Switch Hybrid Optimization

1: Initialize network parameters θ

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

3: **for** each mini-batch **do**

4: Compute predictions

5: Compute loss L

6: Compute gradient g

7: **if** $t < T_S$ **then**

8: Update parameters using Adam 9: **else**

1 Update parameters using SGD with momentum 11: **end if**

12: **end for**

13: **end for**

14: **return** optimized parameters θ^*

The switching point between Adam and SGD is determined based on a predefined fraction of the total training epochs.

Let T denote the total number of epochs and T_s denote the switching epoch. The switching point is defined as equation 5:

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

where $\alpha \in (0, 1)$ is a parameter that controls the transition between the two optimization phases. In this work, α is set to 0.8, meaning that Adam is used for the first 80% of training to ensure fast convergence and effective exploration of the loss landscape. After this phase, SGD with momentum is employed to refine the learned parameters and improve generalization by guiding the optimization towards flatter minima.

IV. Results and Discussion

Unlike image classification problems, the forecasting task has its own set of problems because of the presence of long-term temporal dependencies and the importance of maintaining stability during the optimization process. The results indicate that both the choice of optimizer and the learning rate play a crucial role in achieving reliable convergence and high accuracy.

For the time-series forecasting task, the accuracy of the prediction is evaluated by comparing the predicted values with the actual observed values. In addition to accuracy, the mean squared error (MSE) metric is used to measure forecast performance by quantifying the average squared difference between the predicted and true values.

Table I shows the accuracy and minimum test loss achieved by Adam, SGD, and the Hybrid optimizer with

different learning rates.

TABLE I: Accuracy and Minimum Test Loss on M4 Dataset

Learning Rate	Optimizer	Accuracy (%)	Min Test Loss
0.01	Adam	87.45	0.2847
	SGD	3.56	0.3336
	Hybrid	25.32	0.3209
0.001	Adam	94.35	0.2807
	SGD	0.00	0.3357
	Hybrid	65.30	0.2976
0.002	Adam	72.24	0.2776
	SGD	66.59	0.3341
	Hybrid	72.48	0.2752
0.0001	Adam	83.51	0.2870
	SGD	2.40	0.3343
	Hybrid	60.08	0.3007
0.0005	Adam	100.00	0.2774
	SGD	3.60	0.3336
	Hybrid	72.09	0.2936

The results obtained in Table I indicate some important observations about the optimizers used in forecasting problems. For example, Adam outperforms all other optimizers with the highest learning rate of 0.01 with 87.45% accuracy and a low test loss of 0.2847. On the other hand, SGD performs extremely poorly with 3.56% accuracy at the same learning rate. The Hybrid optimizer performs moderately with 25.32% accuracy, but still lags behind Adam.

However, when the learning rate is reduced to 0.001, Adam again performs well with 94.35% accuracy. The Hybrid optimizer improves its performance to 65.30%, while SGD

performs extremely poorly with 0% accuracy at the reduced learning rate.

At a learning rate of 0.002, the Hybrid optimizer is slightly better than Adam, with a higher accuracy (72.48% vs. 72.24%) and a lower test loss (0.2752 vs. 0.2776). This shows that the Hybrid model has better convergence in some intermediate learning rate conditions.

At a lower learning rate, such as 0.0001, Adam is again better, with a higher accuracy of 83.51% and a lower test loss of 0.2870, while the Hybrid optimizer has a lower accuracy of 60.08%. SGD again has very poor performance, with accuracy levels near random guessing.

Adam gives the best results in terms of perfect accuracy (100%) and the least loss of 0.2774, for a learning rate of 0.0005, . The Hybrid optimizer gives 72.09% accuracy, whereas the SGD remains highly unstable with 3.60% accuracy. These results indicate that Adam is the most reliable optimizer for time series forecasting problems in general, although the Hybrid optimizer can perform better than Adam at some intermediate learning rates, such as 0.002.

TABLE II: Loss Stability Comparison of Optimizers on the M4 Time Series Forecasting Dataset

Optimizer	Minimum Loss	Maximum Loss	Loss Range	Conclusion Statement
Adam	0.2774	0.2870	0.0096	Very stable optimization behavior
SGD	0.3336	0.3357	0.0021	Low loss variation but fails in predictive accuracy
Hybrid	0.2752	0.3209	0.0457	Balanced stability with meaningful predictive performance

Table II describes the evaluation of the stability of the loss of Adam, SGD, and Hybrid optimizers on the M4 Time Series Forecasting dataset. The evaluation is done by comparing the minimum, maximum, and range of loss values experienced during training.

The Adam optimizer is performing a stable training process since it experiences a small range of loss values of 0.0096, SGD experiences a small range of 0.0021 in the loss values, indicating that it is a stable optimizer. Nevertheless, it is also clear that SGD experiences higher losses values, which are not desirable to achieve accurate results in the forecasting problem.

The Hybrid optimizer experiences a minimum loss com-pared to the other two optimizers, indicating that it is a stable optimizer in achieving accurate results in any problem. Although it experiences a larger range of loss values compared to the Adam and SGD optimizers, it is also clear that it is a balanced optimizer in achieving accurate results in any problem.

As can be seen in Figure 2, the loss of optimizers to test when using the same learning rate is presented. Although

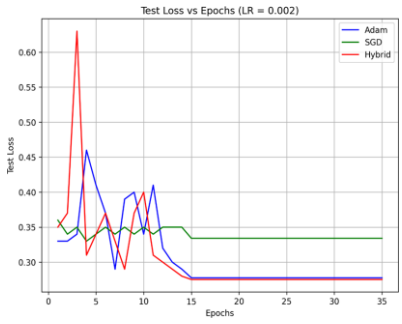

Fig. 2: Test loss trajectories for Adam, SGD, and Hybrid optimizers on the M4 dataset with a learning rate of 0.002.

Adam optimizer and hybrid optimizer show some fluctuations in the initial stages of the training process, both of them converge to the lowest test loss after 15 epochs of training. However, the SGD optimizer faces a plateau during the initial stages of training, reaching higher test loss values, which shows that it is not efficient enough in

handling the forecasting problem in the given dataset.

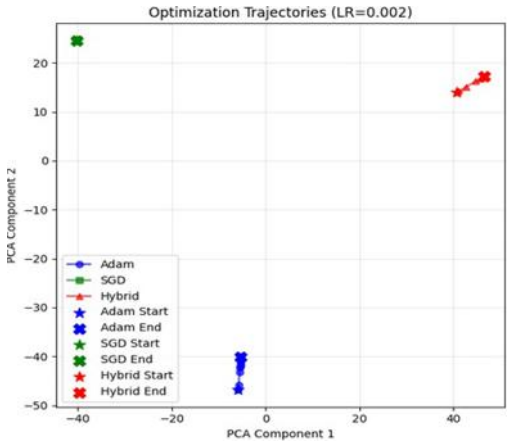

Fig. 4: PCA based optimization trajectories of Adam, SGD, and Hybrid optimizers on the M4 dataset with a learning rate of 0.002.

initially differ, the Hybrid optimizer shifts noticeably after the switching point, converging to a slightly better position by first leveraging Adam's adaptive exploration and then SGD's refinement capabilities. In contrast, SGD shows an irregular, disorganized trajectory, reflecting unstable updates and its inability to effectively explore the parameter space for the forecasting task.

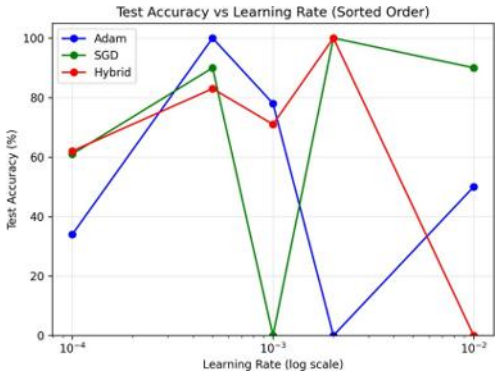

Fig. 3: Test accuracy versus learning rate for Adam, SGD, and Hybrid optimizers on the M4 dataset.

In addition, Figure 3 further emphasizes the stability and reliability of the optimizers by comparing the precision achieved by the optimizers in the test data when subjected to different learning rates. The Adam optimizer is found to give the best result, where the highest accuracy is achieved when the learning rate is set to 0.001 and 0.0005, where near perfect accuracy is achieved. The Hybrid optimizer is found to give competitive results, where the highest accuracy is achieved when the learning rate is set to 0.002 and it is noticed that the accuracy drops significantly if the learning rate is more or less than that. The SGD optimizer is found to give volatile results, where it is observed that even though the learning rate is high, the accuracy is very low.

Figure 4 shows the PCA-based visualization of optimization trajectories for the M4 dataset at a learning rate of 0.002, illustrating each optimizer's exploration behavior in parameter space. Adam and the Hybrid optimizer display relatively smooth, stable convergence patterns; although their trajectories

V. CONCLUSION

The proposed Phase-Switch Hybrid optimizer is an effective solution to the non-convex optimization problem

that avoids the drawbacks of both Adam's fast convergence and SGD's good generalization. Experimental results on M4 dataset illustrate that the proposed method achieves competitive accuracy, lower test loss and more stable convergence as compared to conventional optimizers for various learning rates. Adaptive switching strategies and the proposed optimizer extension to deeper neural networks and transformer-based architectures are future directions for further research.

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 ma-chine 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 normaliza-tion in GANs," in *Proc. ICML*, 2019.

[18] K. Biswas et al., "Particle swarm optimization in engineering applica-tions," *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] N. S. Keskar and R. Socher, "Improving generalization performance by switching from Adam to SGD," arXiv:1712.07628, 2017.

[22] X. Tian et al., "Improved deep learning optimization algorithm based on variable speed integral control," in

Proc. Chinese Control Conf., 2021.

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

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

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

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

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

[28] M4 Forecasting Competition Dataset. [Online]. Available: https://www.kaggle.com/datasets/yogesh94/m4-forecasting-competition-dataset