The Phase-Switch Hybrid Adam–SGD Optimizer for Non-Convex Deep Learning: Evaluation on the MNIST and CIFAR-10 Dataset

Harish Kunder
Department of Artificial Intelligence and Machine Learning, Alva's Institute of Engineering and Technology, Moodbidri, India | Visvesvaraya Technological University, Belagavi, India
researchkunder@gmail.com (corresponding author)

Manjunath Kotari
Department of Computer Science and Engineering, Alva's Institute of Engineering and Technology, Moodbidri, India | Visvesvaraya Technological University, Belagavi, India
mkotari@gmail.com

Received: 6 March 2026 | Revised: 25 March 2026 and 17 April 2026 | Accepted: 30 April 2026

 DOI: https://doi.org/10.48084/etasr.18573

ABSTRACT

Deep neural networks are highly likely to face non-convex optimization issues during training because of local minima, saddle points, and highly complex loss surfaces. Existing optimizers, such as Adam and Stochastic Gradient Descent (SGD), optimize faster or generalize better; however, they cannot optimize both properties effectively for non-convex optimization problems. This study proposes a phase switch hybrid optimization method to optimize and improve the training of deep neural networks. The proposed method uses Adam for faster convergence during the initial phase and SGD with momentum for better generalization during the latter phase. The hybrid optimization method combines the advantages of both Adam and SGD, enabling faster convergence and better generalization during training. The method is evaluated on the well-known datasets MNIST and CIFAR-10. The results obtained using the proposed method are better or comparable to existing methods on different metrics such as accuracy and loss minimization. The introduced method achieves improved performance, with accuracy gains of up to 0.4–0.6% on the MNIST dataset and up to 1–2% on the CIFAR-10 dataset compared to baseline optimizers, along with lower test loss in several learning rate settings. The study demonstrates that the proposed method is an effective solution for handling non-convex optimization problems and improves the robustness of training on different datasets.

Keywords-deep learning; Adam; SGD; MNIST; CIFAR-10; loss function; accuracy

I. INTRODUCTION

The availability of digital data has significantly influenced the way information is processed and analyzed in modern computing environments. Data are continuously generated from a wide range of sources, including healthcare systems, financial platforms, transportation networks, and online social services. These data sources produce both structured and unstructured information at a very large scale. Handling such data effectively requires computational methods capable of identifying patterns and extracting useful knowledge. Deep learning has become an important technique for analyzing complex datasets and identifying meaningful relationships within the data. Deep learning models learn automatically multiple levels of representation from raw inputs, which allows them to perform effectively in tasks such as image recognition, speech processing, natural language understanding, and time-series prediction. Because of these capabilities, deep learning methods are widely adopted in many real-world applications that involve large and complex datasets.

Deep learning models are trained by minimizing a loss function using the optimization technique. During the process, the parameters of the neural network are repeatedly updated so that the difference between predicted outputs and actual values becomes smaller. The effectiveness of the training process depends on the optimization algorithm used to update these parameters. Several optimization approaches have been developed for this purpose. Among the most commonly used techniques are adaptive optimization algorithms such as the Adam optimizer and gradient-based approaches such as Stochastic Gradient Descent (SGD). Adam adapts the learning

rate for individual parameters, which often results in faster convergence during training. On the other hand, SGD is valued for its simplicity and its ability to achieve good generalization performance. Even though these optimizers are widely used, their effectiveness may decrease when the loss surface becomes complex.

Due to the non-convex nature of problems, training a neural network is difficult. Non-convex objective functions often contain numerous local minima and saddle points. When optimization algorithms encounter such landscapes, they may get trapped in suboptimal regions instead of reaching the true global minimum. Consequently, the resulting model may not achieve the best possible performance. This challenge becomes significant as neural networks grow deeper and the parameter space becomes high-dimensional. Although many optimization strategies have been proposed, achieving reliable convergence in non-convex environments is an important research issue in deep learning.

To overcome the limitations of nonconvex problems, the present study introduces a hybrid optimization strategy intended to enhance the training process of deep learning models. The proposed method combines the strengths of adaptive optimization and traditional gradient-based techniques. By combining these two optimization strategies, the hybrid method aims to maintain fast convergence and to improve the stability of the learning process. The proposed hybrid optimizer is analyzed using various benchmark datasets. The experiments include image classification tasks using the MNIST and CIFAR-10 datasets. The performance of the hybrid algorithm is measured using accuracy, minimum loss, and variations in the loss value based on different learning rates. The experimental results indicate that the hybrid optimizer is capable of achieving competitive or improved performance when compared with commonly used optimization algorithms.

The proposed hybrid method works efficiently on the non-convex optimization in deep learning models. The method combines adaptive and gradient-based learning strategies to balance convergence speed and training stability. The empirical analysis is performed on multiple datasets to evaluate the effectiveness of the approach. Finally, the proposed optimizer is compared with widely used optimization algorithms to demonstrate its performance advantages. Many existing approaches rely on switching criteria, learned update rules, or additional computational overhead, while the proposed method focuses on a simple and structured phase-switch mechanism that is easy to implement and computationally efficient. The hybrid optimizer transition is performed at a predefined switching point with a reset of optimizer states. Therefore, the hybrid optimizer avoids residual effects from adaptive moment estimates and ensures stable convergence behavior in a non-convex optimization environment.

II. LITERATURE REVIEW

Optimization is one of the most important research areas in machine learning and deep learning due to the complexity involved in the minimization of complex non-convex objective functions during the training of neural networks. Many optimization algorithms have been proposed to improve the convergence rate and stability of deep learning models with better performance.

A. Foundations of Optimization in Machine Learning

The mathematical foundations of optimization have been extensively discussed in the literature. Authors in [1] presented a detailed theoretical framework for convex optimization and demonstrated how the latter can ensure a single global optimum solution. However, it has been observed that most machine learning problems are non- convex, making it a far more complex optimization problem. The advent of neural networks has been made possible by the invention of the backpropagation algorithm [2]. This algorithm enables efficient computation of gradients for multi-layer networks and has been a crucial part of modern deep learning algorithms. Later advancements on gradient-based learning were presented in [3], focusing on efficient gradient propagation and learning in neural networks.

B. Gradient-Based Optimization Algorithms

Most deep learning models utilize gradient-based optimization algorithms. Authors in [4] proposed a comprehensive overview of optimization algorithms for large-scale machine learning and emphasized the importance of SGD for managing large-scale datasets. Authors in [5] also explored various gradient descent algorithms and their extensions, including momentum, scheduling, and adaptive learning rate methods. Similarly, authors in [6] proposed a theoretical analysis of SGD and extensions to improve its convergence rate for large-scale optimization problems. These works highlight the importance of gradient-based optimization algorithms for training deep neural networks.

C. Adaptive Optimization Methods

Adaptive optimization algorithms have become extremely popular due to their ability to adjust learning rates during training. Adam, one of the most widely used adaptive optimization algorithms, was introduced in [7]. It utilizes both momentum and adaptive learning rate estimation. Many studies have investigated its limitations, including [8], which analyzed the convergence characteristics of Adam and proposed an updated version of it. Authors in [9] extended their study on Adam and its convergence characteristics in stochastic optimization settings. Furthermore, authors in [10] proposed a distributed adaptive optimization method referred to as DADAM.

D. 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 in [11], discussing the limitations of conventional optimization techniques in handling large-scale parameter spaces. Authors in [12] demonstrated that saddle points are much denser than local minima in the loss function of neural network problems in high-dimensional space. These saddle points slow down the convergence of the conventional optimization algorithm because the gradients become extremely small in the vicinity of the saddle points.

Authors in [13] studied the non-convex min-max optimization problems, which frequently appear in the field of adversarial learning. Similarly, authors in [14] studied the conventional optimization algorithms for non-convex game-theoretic problems in the field of machine learning, while authors in [15] proposed second-order optimization techniques that can escape the saddle points effectively in the loss landscape.

E. 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. Authors in [16] researched different distributed learning frameworks that are used for non-convex optimization and emphasized the role of communication-efficient algorithms in distributed learning systems. Authors in [17] carried out a large-scale empirical study on different optimization methods used for Generative Adversarial Networks (GANs). It was found that the stability of GAN training is highly dependent on different optimization and regularization methods.

F. Alternative and Evolutionary Optimization Techniques

Apart from gradient-based optimization techniques, some heuristic optimization techniques have also been studied for the solution of complex optimization problems. Particle Swarm Optimization (PSO) is one such heuristic optimization technique that has been successfully applied to engineering optimization problems. The efficiency of PSO in exploring complex search spaces has been proven in previous studies; however, such techniques are computationally expensive in the context of training deep neural networks.

G. Recent Studies and Modern Optimization Research

Recent studies have tried to present a brief overview of the literature available on the optimization of deep learning techniques. Authors in [18] explored different optimization strategies for non-convex machine learning problems and discussed the issues encountered during optimization due to saddle points, flat regions, and local minima in the loss function. The authors highlighted the importance of hybrid optimization techniques that integrate different learning mechanisms. Similarly, authors in [19] examined optimization issues in deep learning functions and observed that optimizers like SGD, Adam, and RMSProp fail to escape flat regions in the loss function. Authors in [20] investigated optimization techniques theoretically by comparing first-order and second-order optimizers in terms of convergence and generalization. Authors in [21] divided the optimizers of deep learning algorithms into first-order and second-order optimizers. The method explained the advantages and disadvantages of both. Authors in [22] researched the optimization algorithms used in neural networks. Their method highlighted the significance of using hybrid optimization techniques. Authors in [23] studied the optimization techniques applied for solving multi-objective deep learning problems. They explained the importance of utilizing hybrid optimization techniques for solving the conflicting objectives.

H. Hybrid and Learned Optimization Strategies

Hybrid optimization techniques, which combine the advantages of different optimization algorithms, have been explored. Authors in [24] proposed the Hybrid Update-Based (HUB) optimizer, which utilizes update rules learned during training for dynamic optimizer switching. Authors in [25] evaluated the performance of various optimization algorithms for large-scale language modeling tasks. It was found that employing staged training strategies, where adaptive optimization algorithms can be used during initial training, can enhance generalization performance.

Authors in [26] investigated differentiable convex optimization layers, which involve incorporating structured optimization blocks within deep neural network architectures. Although they focused on the incorporation of convex optimization, it can be seen that the hybridization of different optimization techniques can be used for improving training robustness. Another hybrid optimization algorithm, called Foxtsage, employs both global search strategies, such as metaheuristic search, for escaping local minima, coupled with local optimization strategies such as SGD for fine-tuning parameters [27]. Authors in [28] proposed an ensemble machine learning approach that improves air quality index classification accuracy by combining multiple predictive models.

I. Datasets Used for Optimization Evaluation

Some commonly used benchmark datasets for testing optimization algorithms used in deep learning are the MNIST data set [29] and the CIFAR-10 dataset [31].

J. Research Gap

Although adaptive optimizers such as Adam achieve faster convergence and SGD provides better generalization, combining these advantages in a stable and computationally efficient manner remains a challenge [18, 19]. Existing hybrid optimization approaches rely on complex switching criteria, additional hyperparameters, or learned update mechanisms, which increase implementation complexity and computational cost [30]. An issue is the lack of a simple and reliable strategy that can transition between optimization phases without causing instability due to residual optimizer states, such as accumulated moments in adaptive methods. These residual effects can negatively influence the behavior of subsequent optimization phases and lead to suboptimal convergence. So, the proposed method introduces a structured phase-switch mechanism with a predefined transition point and explicit resetting of optimizer states. This ensures that each optimization phase operates independently by improving stability. Also, it makes the approach easier to implement while maintaining effective performance in a non-convex optimization environment [30].

III. METHODOLOGY

The proposed methodology aims to improve the convergence rate and performance of the network during training. It includes dataset preprocessing, model selection, hybrid optimization, and performance evaluation. The overall workflow of the proposed methodology is shown in Figure 1.

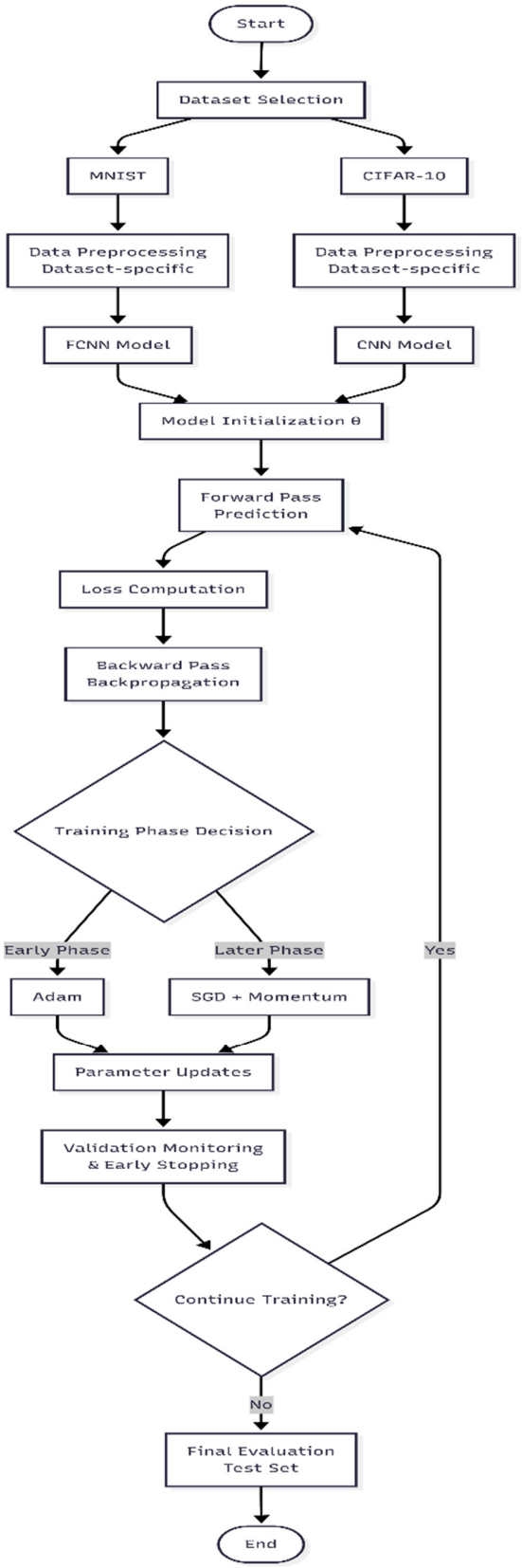

Fig. 1. Proposed methodology.

A. Overall Framework

Figure 1 shows the overall workflow of the proposed methodology. The overall workflow is divided into several steps, starting from dataset 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 consists of the following steps:

1) Step 1: Model Parameter Initialization

Training starts with the initialization of the model parameters θ. The initialization of the parameters plays a significant role in the stability of the training process and prevents symmetry issues during training. The weights are assigned random values using random distributions, while bias terms are assigned constant values.

2) Step 2: Data Preprocessing and Dataset Preparation

The MNIST [29] and CIFAR-10 [31] datasets need to be converted into proper formats before starting the training process. The preprocessing techniques used depend on the type of datasets. The preprocessing techniques improve the learning ability of the neural network and prevent instabilities in the training process.

3) Step 3: Model Selection

After preprocessing, the neural network architecture is selected based on the characteristics of the dataset to ensure appropriate feature learning. For the MNIST dataset, a Fully Connected Neural Network (FCNN) is used as the classifier. The dataset consists of grayscale images of handwritten digits with relatively simple patterns, for which a dense architecture is sufficient. The model includes an input layer, followed by hidden layers with nonlinear activation functions, and an output layer with softmax activation for classification. For the CIFAR-10 dataset, a Convolutional Neural Network (CNN) is used as the classifier. This dataset contains color images with higher variability and spatial complexity. To capture these features, convolutional layers are employed for feature extraction, followed by pooling operations and fully connected layers, with a softmax layer at the output. Using dataset-specific classifiers allows the proposed hybrid optimization method to be evaluated across both simple and complex model structures, providing a more reliable assessment of its performance.

4) Step 4: Forward Propagation

In this step, the processed data are passed through the different layers of the neural network. In each layer of the network, the data are transformed using operations such as matrix multiplication, convolution, or recurrence. The output of the last layer of the network represents the predictions $\hat{y}$.

5) Step 5: Loss Computation

The predicted output of the network is compared with the actual output using an appropriate loss function. Categorical cross-entropy loss is used for classification problems, while

Mean Squared Error (MSE) loss is used for forecasting problems.

6) Step 6: Gradient Backpropagation

The loss value is then used to calculate the gradients of the loss with respect to all model parameters. This is done by employing the backpropagation algorithm. The gradients provide information on how the model parameters should be updated in order to reduce the loss.

7) Step 7: Training Phase Determination

At this stage, the algorithm determines whether the training phase is in the early phase. The training phase determines the optimization technique used at this stage.

8) Step 8: Early Phase – Adaptive Optimization

In the early phase of training, adaptive optimizers such as Adam are used. Adam calculates the adaptive learning rates of the model parameters utilizing the first moment of the gradients and the second moment of the gradients.

9) Step 9: Later Phase

At a later stage, the optimization process is performed by employing SGD with momentum. This helps in stabilizing the training process and directs the optimization process towards flatter minima, which are more likely to generalize well.

10)Step 10: Parameter Updates

In all the mentioned optimization methods, the parameters are updated, and this process improves the model's prediction accuracy in a gradual manner by minimizing the loss function.

11)Step 11: Convergence Monitoring

At every stage, the model monitors the convergence, and if there is improvement in the validation accuracy and loss, the training process is continued. If there is no improvement, the model is stopped at some point, known as early stopping.

12)Step 12: Final Parameter Estimation

In the final stage, the parameters are optimized, and the optimized parameters are given by θ^*. These parameters represent the knowledge learned by the neural network model.

13)Step 13: Testing on Unseen Data

The trained model is tested on unseen data, which were not utilized during training.

14)Step 14: Performance Evaluation

Depending on the model, performance is evaluated using various performance metrics such as accuracy, loss, and forecasting error. The switching point Ts is to be chosen based on the experiment. The optimizer state is reset during the switch to avoid interference from residual momentum or adaptive buffers. The hybrid strategy is provided in Algorithm 1. Figure 2 shows the detailed steps to be followed by the hybrid optimizer.

The architecture of the neural network models used in each case is provided in Table I. In the case of the MNIST dataset, an FCNN is used with 784 input neurons in the input layer, followed by two dense layers with 256 and 128 neurons each, and an output layer with 10 neurons. In the case of the CIFAR-10 dataset, a CNN is used with multiple convolutional layers with 32, 64, and 128 filters each, followed by a max pooling layer and a fully connected layer before the output layer.

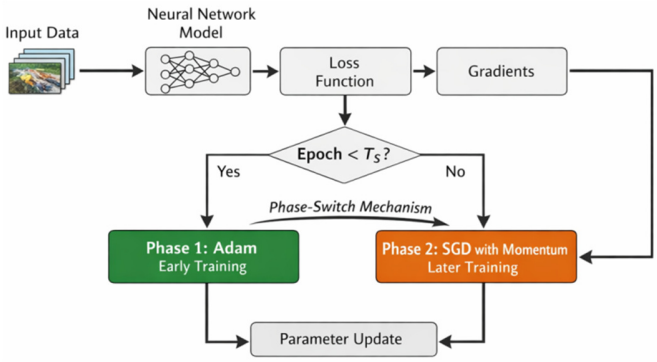

Fig. 2. Hybrid optimizer.

TABLE I. ARCHITECTURE OF THE LAYERS

Dataset	Model	Architecture details
MNIST	FCNN	Input layer (784) → Dense layer (256) → Dense layer (128) → Output layer (10), with ReLU and Softmax activations
CIFAR- 10	CNN	Convolution layer (32) → Convolution layer (64) → Max- Pooling layer → Convolution layer (128) → Fully Connected – Output layer

```
Algorithm 1: Phase-Switch Training Strategy
Input:
    Training set D = {(x_i, y_i)}
    Model f(·; θ) with parameters θ
    Optimizer_A = adaptive optimizer
    Optimizer_S = stochastic optimizer with momentum
    Total epochs T, Switching epoch Ts
    Learning rates ηA, ηS, Momentum μ
    Batch size B
Output:
    Optimized parameters θ*

Procedure:
1   Initialize θ
2   For epoch t = 1 … T do
3       Divide D into mini-batches of size B
4       For each mini-batch Mb do
5       Compute predictions ŷ = f(Mb.x; θ)
6           Compute loss L = ℓ(ŷ, Mb.y)
7           Compute gradient g = ∂L/∂θ
8           If t < Ts:
9               θ ← Optimizer_A(θ, g, ηA)
10          Else:
11              θ ← Optimizer_S(θ, g, ηS, μ)
12      End For
13  End For
```

14 Return θ*

The switching epoch (Ts) is not fixed and depends on the dataset, model architecture, and training environment. In the initial stages, Adam enables faster loss reduction, while in later stages, SGD improves generalization. In this study, a switching point around epoch 8 was selected empirically for the MNIST and CIFAR-10 datasets because it was observed that the reduction in loss begins to stabilize in early epochs. This value should not be considered universal, and different switching points may be more suitable for other datasets or models.

IV. RESULTS

Table II summarizes accuracy and minimum test loss values for the Adam, SGD, and proposed optimization algorithms using different learning rate values. It is demonstrated how the three optimizers perform when using different learning rate settings. When utilizing a high learning rate of 0.01, SGD shows the best results in terms of accuracy and test loss. SGD achieved an accuracy of 97.76% and a minimum test loss of 0.0690. Although the hybrid optimizer shows competitive results in terms of accuracy, its test loss is higher compared to SGD. Adam's performance is worse in this high learning rate setting.

However, when using lower learning rates of 0.002 and 0.001, the best results in terms of accuracy and test loss are shown by the hybrid optimizer. When utilizing a learning rate of 0.002, the hybrid method exhibits the best accuracy of 98.02% and a test loss of 0.0682. Similarly, when using a lower learning rate of 0.001, the hybrid method shows the best accuracy of 97.96% and a test loss of 0.0649. However, at the lowest learning rate of 0.0001, Adam again takes over by attaining an accuracy of 96.22. From these results, it can be concluded that Adam performs better at very small learning rates, whereas SGD performs better at relatively larger learning rates. However, the hybrid optimizer performs better at intermediate learning rates, and hence can be regarded as a powerful optimization algorithm.

TABLE II. ACCURACY AND MINIMUM TEST LOSS OF MNIST

Learning rate	Optimizer	Accuracy (%)	Minimum 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

Figure 3 illustrates the training loss trajectories for the three optimizers when the learning rate is set to 0.002. It is observed that Adam rapidly decreases the training loss in the initial epochs. This indicates that Adam is effective in reducing loss in the initial stages of optimization. However, in the case of SGD, a gradual reduction in loss is seen. This demonstrates that SGD converges slowly. The hybrid optimizer is an optimization method that brings together the best of both worlds in terms of optimization. In the initial stages of optimization, Adam optimization is followed. However, around epoch 8, SGD optimization is followed. In this way, the hybrid optimizer avoids the premature convergence of Adam optimization and also avoids the slow learning characteristics of SGD optimization.

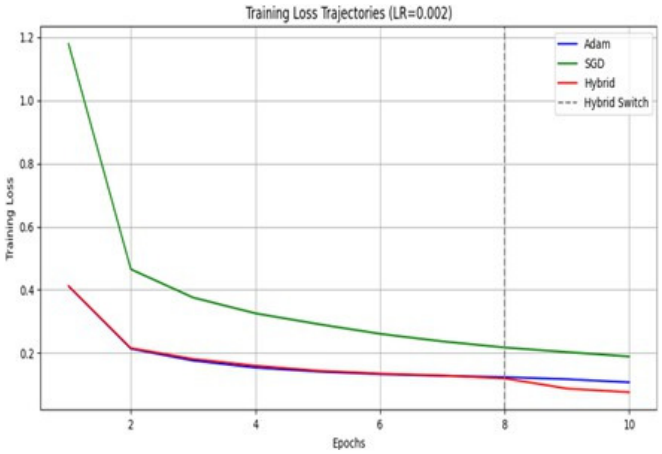

Fig. 3. Training loss trajectories for Adam, SGD, and hybrid optimizers on the MNIST dataset (learning rate = 0.002).

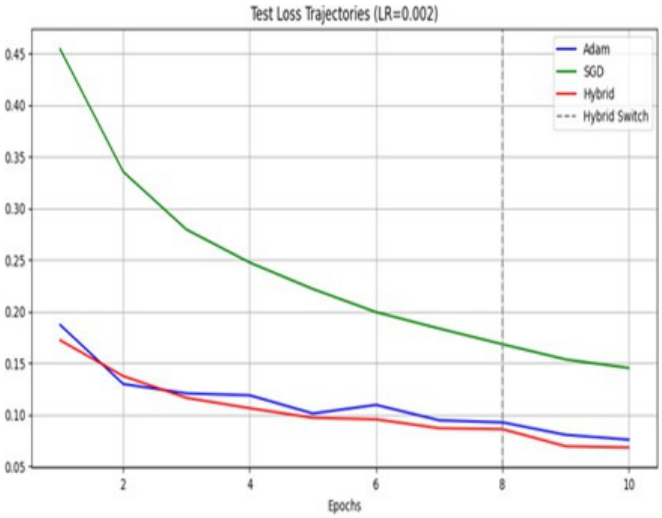

Fig. 4. Test loss trajectories on the MNIST dataset with a learning rate of 0.002.

The test loss curves for a learning rate of 0.002, as depicted in Figure 4, again confirm the observation from a different point of view. In the initial stages of training, Adam and the hybrid optimizer are successful in reducing test loss. However, in the later stages of training, after the switch point in the hybrid optimizer's strategy, the test loss is reduced steadily compared to all other optimizers. Thus, in the end, the test loss is minimized for the hybrid optimizer compared to all other optimization strategies. Figure 5 displays the relationship between the minimum test loss and the learning rate for each of

these three optimizers. It is emphasized how each optimization technique reacts to different learning rates during training. With a learning rate of 0.01, SGD has the lowest test loss, signifying that it can effectively utilize a higher learning rate for faster convergence.

The hybrid optimizer shows the most consistent performance across the broad range of learning rates. In specific cases, the Hybrid optimizer performs optimally in the intermediate range of learning rates from 0.001 to 0.002. This consistency suggests that the hybrid optimization technique can successfully combine the speed of adaptive optimization with the stability of SGD. Similarly, Figure 6 illustrates the relationship between the best test accuracy and the learning rate for all three optimizers. For the intermediate learning rates of 0.001 and 0.002, all three optimizers report competitive results with accuracy ranging between 97% and 98%. Among all optimizers, the hybrid optimizer always reports the highest accuracy values with slight improvements over AdamFor the high learning rate of 0.01, SGD performs best, implying that it benefits from larger step sizes during the optimization process. In contrast, Adam outperforms other optimizers for the smallest learning rate of 0.0001 due to its adaptive learning rate scheme. The hybrid optimizer performs well for all extreme learning rates while attaining its best results within the intermediate learning rate range. This again proves the flexibility of the hybrid optimization scheme, which has stable performance over a broad range of training settings.

Compared with the MNIST dataset, the CIFAR-10 dataset is more challenging because the images are complex in color with greater intra-class variations. The accuracy and minimum test loss obtained using Adam, SGD, and hybrid optimizers with different learning rate values are summarized in Table III. The hybrid optimizer is found to have the lowest loss range, indicating that it is the most stable compared to Adam and SGD. This is because the hybrid optimizer combines Adam's ability to converge quickly and SGD's ability to refine, thus providing a balanced result.

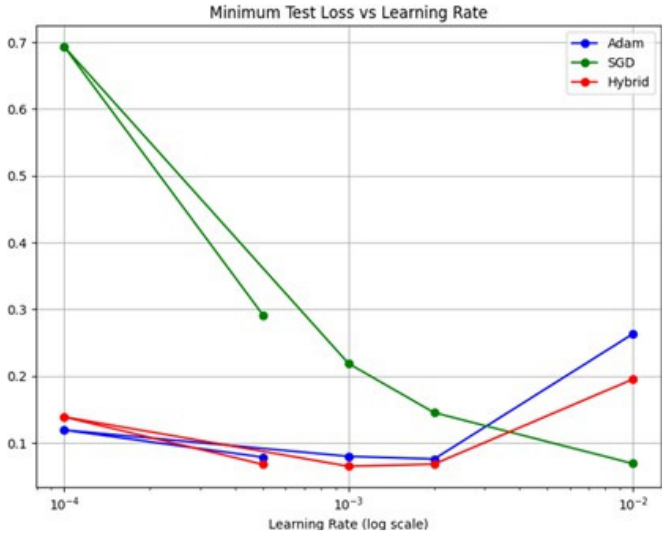

Fig. 5. Minimum test loss versus learning rate for Adam, SGD, and hybrid optimizers on the MNIST dataset.

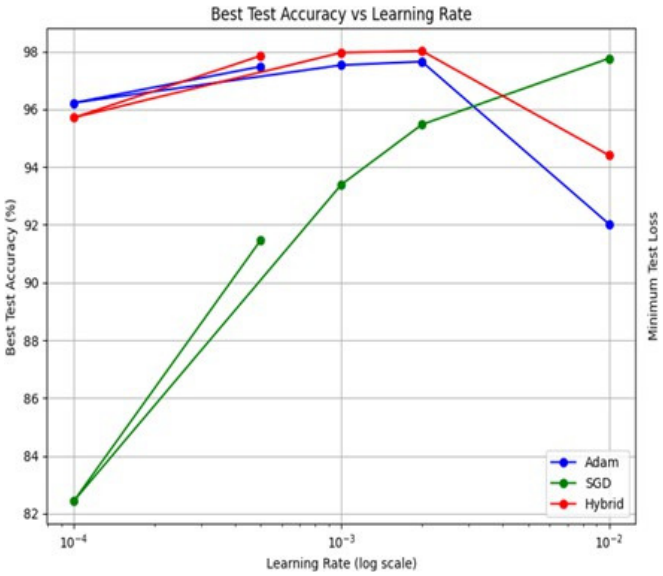

Fig. 6. Best test accuracy versus learning rate for Adam, SGD, and hybrid optimizers on the MNIST dataset.

Figure 7 shows the training loss curves of the three optimizers at a learning rate of 0.0001. Adam demonstrates a stable and consistent decrease in training loss, outperforming the others, while SGD converges slowly and remains at a higher loss level, indicating poor adaptability. The hybrid optimizer starts with a higher loss than Adam but gradually improves, and after the phase switch around epoch 8, its behavior becomes similar to Adam, reducing the gap in performance. Figure 8 demonstrates the minimum test loss for different learning rates with each optimizer. Adam optimizer achieves the best results for moderate learning rates, 0.001 and 0.002; however, its performance is worse for a higher learning rate of 0.01. The hybrid optimizer is more stable for all learning rates; though, it achieves lower performance compared to the Adam optimizer for optimal learning rates. SGD optimizer is more sensitive to learning rates, as it demonstrates better results for higher learning rates but shows unstable and higher loss for lower learning rates.

TABLE III. ACCURACY AND MINIMUM TEST LOSS ON CIFAR-10

Learning rate	Optimizer	Accuracy (%)	Minimum test loss
0.01	Adam	77.04	0.6625
	SGD	75.36	0.6978
	Hybrid	71.77	0.7907
0.001	Adam	83.61	0.4856
	SGD	54.49	1.2789
	Hybrid	82.91	0.5028
0.002	Adam	83.27	0.4907
	SGD	62.04	1.0802
	Hybrid	82.15	0.5158
0.0001	Adam	72.78	0.7866
	SGD	36.27	1.8566
	Hybrid	72.98	0.7735
0.0005	Adam	80.97	0.5453
	SGD	48.99	1.4274
	Hybrid	81.03	0.5482

In Figure 9, the optimization path using PCA for each optimizer with a learning rate of 0.0001 is visualized, providing a better understanding of the behavior of each optimizer in the search space. Adam's optimization path is smooth and stable, indicating that Adam's optimization process is consistent. On the other hand, the optimization path for SGD is scattered due to the instability of the optimizer at a low learning rate. The optimization path for the hybrid optimizer is similar to Adam's path at first due to the adaptive update, but then changes after the phase switch in the optimization process.

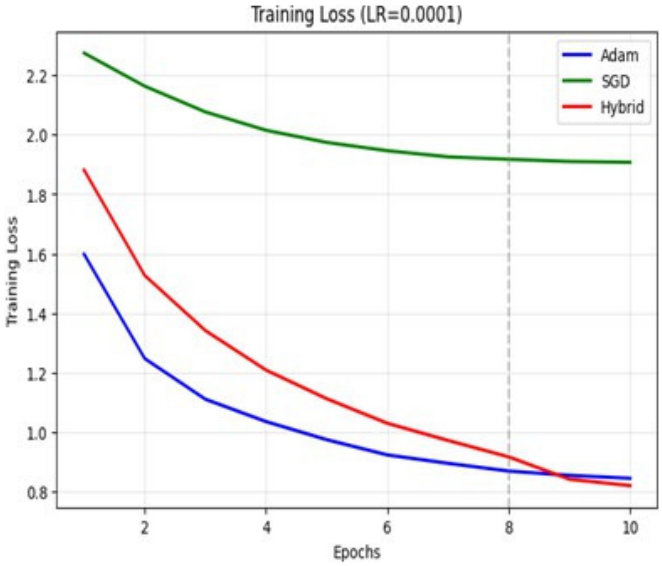

Fig. 7. Training loss trajectories for Adam, SGD, and hybrid optimizers on the CIFAR-10 dataset with a learning rate of 0.0001.

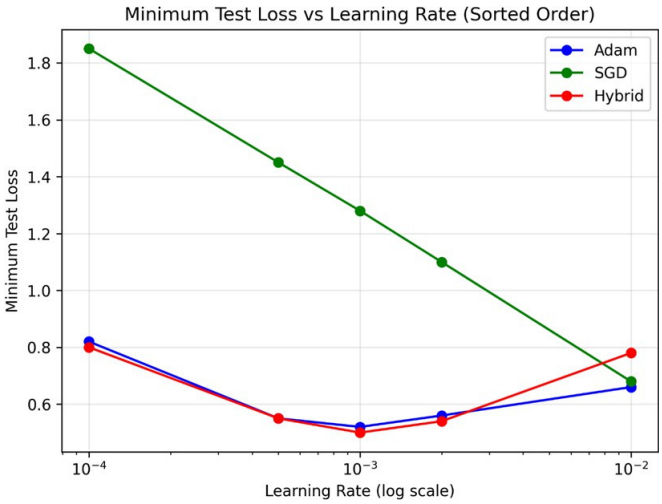

Fig. 8. Minimum test loss versus learning rate for Adam, SGD, and hybrid optimizers on the CIFAR-10 dataset.

Due to the stochastic nature of deep learning training, the results may vary slightly across different runs because of random initialization and data shuffling. In this study, consistent performance trends were observed, and the results from a representative run are reported for clarity. Future work will include reporting average results over multiple runs, along with standard deviations, to further improve statistical reliability.

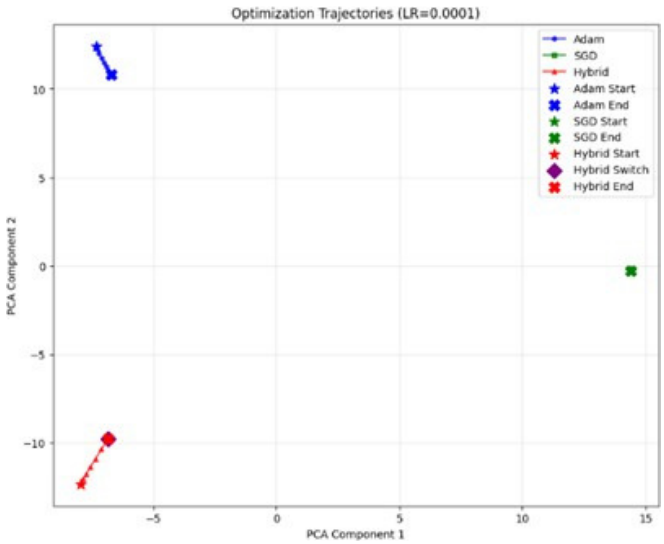

Fig. 9. PCA-based optimization trajectories of Adam, SGD, and hybrid optimizers on the CIFAR-10 dataset with a learning rate of 0.0001.

Although only Adam and SGD are considered in this study, they represent the fundamental categories of adaptive and gradient-based optimization methods. Many modern optimizers, such as AdamW, RMSProp, and Nadam, are extensions of these base algorithms. Therefore, the proposed hybrid strategy is expected to generalize to these variants. The former does not introduce a high additional computational cost compared to standard optimization methods. Since it utilizes existing Adam and SGD update rules without incorporating extra computations, the overall training complexity remains similar. The only additional step is the phase transition, which has a negligible impact on training time. Therefore, the method maintains computational efficiency while improving optimization performance.

The loss curve indicates that Adam reduces the training loss faster during the initial epochs, such as within the first 3–5 epochs, and demonstrates fast early convergence. The SGD takes more time to reach its minimum loss, usually requiring around 10–15 epochs, which reflects its slower convergence behavior.

The hybrid optimizer combines fast early convergence and generalization and achieves near-minimum loss in approximately 6–8 epochs. This allows it to converge faster than SGD while maintaining better stability than Adam in later stages. After the transition point, the hybrid method's loss curve becomes smoother, with fewer fluctuations, suggesting improved stability during training.

Overall, these observations indicate that the proposed approach provides a good balance between convergence speed and training stability.

V. CONCLUSION

This study proposed and evaluated a Phase-Switch Hybrid optimization strategy that can effectively deal with the complexities of non-convex optimization in deep learning models. The proposed hybrid optimization strategy utilizes the high convergence rate of the Adam optimization algorithm during the initial stages of training and the high precision of Stochastic Gradient Descent (SGD) during the later stages of training. This allows for faster convergence rates and high precision during model training.

The proposed optimization strategy has been evaluated on the MNIST and CIFAR-10 datasets. The results showed that Adam performs well as a reliable optimization algorithm for different learning configurations, especially when a high learning rate is used. SGD performs well only under certain conditions and has a high rate of convergence failure. The findings suggest that the Phase-Switch Hybrid optimizer offers a promising solution for addressing non-convex optimization problems in deep learning models. By adapting different optimization strategies during different phases of training, the proposed method improves the convergence rate of the model while maintaining competitive results on different datasets.

DECLARATION OF COMPETING INTERESTS

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

ACKNOWLEDGMENT

The authors sincerely thank the Principal and the management of Alva's Institute of Engineering and Technology for their constant support and guidance during the course of this research work. The authors extend their sincere gratitude to Dr. Gopalakrishna N. Kini, Professor, Department of Computer Science and Engineering, MIT Manipal; Dr. Jyothi Shetty, Professor and Head, Department of Computer Science and Engineering, NMAMIT Nitte; and Dr. Sarika Hegde, Professor, Department of Computer Science and Engineering, NMAMIT Nitte, for their constant guidance and support during the research process.

DATA AVAILABILITY

The MNIST and CIFAR-10 datasets used in this study were collected from [29] and [31], respectively.

AI USE AND DECLARATION OF GENERATIVE AI USE

During the preparation of this work, the authors used ChatGPT (OpenAI) to assist in language refinement. After using this tool, the authors carefully reviewed and edited the content as needed and take full responsibility for the content of the publication.

REFERENCES

[1] S. Boyd and L. Vandenberghe, *Convex Optimization*, 1st ed. Cambridge, UK: Cambridge University Press, 2004.

[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," Defense Technical Information Center, Fort Belvoir, VA, ADA164453, Sep. 1985. https://doi.org/10.21236/ADA164453.

[3] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient BackProp," in *Neural Networks: Tricks of the Trade*, Vol. 1524, G. B. Orr and K.-R. Müller, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 9–50.

[4] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization Methods for Large-Scale Machine Learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, Jan. 2018, https://doi.org/10.1137/16M1080173.

[5] S. Ruder, "An Overview of Gradient Descent Optimization Algorithms." arXiv, 2016, https://doi.org/10.48550/ARXIV.1609.04747.

[6] R. Gower *et al.*, "SGD: General Analysis and Improved Rates," in *36th International Conference on Machine Learning*, Long Beach, CA, USA, 2019, Art. no. PMLR 97.

[7] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *International Conference on Learning Representations*, Banff, AB, Canada, Apr. 2014.

[8] S. J. Reddi, S. Kale, and S. Kumar, "On the Convergence of Adam and Beyond," in *6th International Conference on Learning Representations*, Vancouver, BC, Canada, Feb. 2018.

[9] X. Chen, S. Liu, R. Sun, and M. Hong, "On the Convergence of a Class of Adam-Type Algorithms for Non-Convex Optimization," in *7th International Conference on Learning Representations*, New Orleans, LA, USA, May 2019.

[10] P. Nazari, D. A. Tarzanagh, and G. Michailidis, "DADAM: A Consensus-Based Distributed Adaptive Gradient Method for Online Optimization," in *7th International Conference on Learning Representations*, New Orleans, LA, USA, May 2019.

[11] P. Jain and P. Kar, *Non-convex Optimization for Machine Learning*. Hanover, MA, USA: Now Publishers Inc., 2017.

[12] Y. N. Dauphin *et al.*, "Identifying and Attacking the Saddle Point Problem in High-Dimensional Non-Convex Optimization," in *Proceedings of the 28th International Conference on Neural Information Processing Systems*, Cambridge, MA, USA, Dec. 2014, vol. 2, pp. 2933–2941.

[13] M. Razaviyayn, T. Huang, S. Lu, M. Nouiehed, M. Sanjabi, and M. Hong, "Nonconvex Min-Max Optimization: Applications, Challenges, and Recent Theoretical Advances," *IEEE Signal Processing Magazine*, vol. 37, no. 5, pp. 55–66, Sep. 2020, https://doi.org/10.1109/MSP.2020.3003851.

[14] M. Nouiehed, M. Sanjabi, T. Huang, and J. D. Lee, "Solving a Class of Non-Convex Min-Max Games Using Iterative First Order Methods," in *33rd Conference on Neural Information Processing Systems*, Vancouver, BC, Canada, 2019.

[15] P. Xu, F. Roosta-Khorasani, and M. W. Mahoney, "Second-Order Optimization for Non-Convex Machine Learning: An Empirical Study." arXiv, 2017, https://doi.org/10.48550/ARXIV.1708.07827.

[16] T.-H. Chang, M. Hong, H.-T. Wai, X. Zhang, and S. Lu, "Distributed Learning in the Nonconvex World: From Batch Data to Streaming and Beyond," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 26–38, May 2020, https://doi.org/10.1109/MSP.2020.2970170.

[17] K. Kurach, M. Lucic, X. Zhai, M. Michalski, and S. Gelly, "A Large-Scale Study on Regularization and Normalization in GANs," in *Proceedings of the 36th International Conference on Machine Learning*, Long Beach, CA, USA, 2019.

[18] G. B. Fotopoulos, P. Popovich, and N. H. Papadopoulos, "Review Non-Convex Optimization Method for Machine Learning." arXiv, 2024, https://doi.org/10.48550/ARXIV.2410.02017.

[19] R. K. Sharma, C. Singh, and A. Shing, "Navigating Complexity: Optimization Challenges in Non-Convex Deep Learning Objective," *International Journal of Novel Research and Development*, vol. 9, no. 9, pp. 628–639, Sep. 2024.

[20] J. Wang and A. Choromanska, "A Survey of Optimization Methods for Training DL Models: Theoretical Perspective on Convergence and Generalization." arXiv, 2025, https://doi.org/10.48550/ARXIV.2501.14458.

[21] R. Kashyap, "A Survey of Deep Learning Optimizers -- First and Second Order Methods." arXiv, 2022, https://doi.org/10.48550/ARXIV.2211.15596.

[22] R. Abdulkadirov, P. Lyakhov, and N. Nagornov, "Survey of Optimization Algorithms in Modern Neural Networks." Computer Science and Mathematics, Apr. 20, 2023, https://doi.org/10.20944/preprints202304.0648.v1.

[23] S. Peitz and S. S. Hotegni, "Multi-Objective Deep Learning: Taxonomy and Survey of the State of the Art," *Machine Learning with Applications*, vol. 21, Sep. 2025, Art. no. 100700, https://doi.org/10.1016/j.mlwa.2025.100700.

[24] G. Dai, W. Wu, Z. Wang, J. Fu, S. Zhang, and T. Huang, "HUB: Guiding Learned Optimizers with Continuous Prompt Tuning," in *International Conference on Learning Representations*, Vienna, Austria, May 2024, https://doi.org/10.48550/ARXIV.2305.16823.

[25] R. Zhao, D. Morwani, D. Brandfonbrener, N. Vyas, and S. Kakade, "Deconstructing What Makes a Good Optimizer for Language Models." arXiv, 2024, https://doi.org/10.48550/ARXIV.2407.07972.

[26] C. Katyal, "Differentiable Convex Optimization Layers in Neural Architectures: Foundations and Perspectives." arXiv, 2024, https://doi.org/10.48550/ARXIV.2412.20679.

[27] S. A. Aula and T. A. Rashid, "Foxtsage vs. Adam: Revolution or Evolution in Optimization?" Jan. 05, 2025, https://doi.org/10.36227/techrxiv.173609892.25586054/v1.

[28] A. Fahim, A. M. Osman, Z. Tarek, and A. M. Elshewey, "Enhancing Air Quality Index Classification Based on Ensemble Machine Learning Techniques," *Engineering, Technology & Applied Science Research*, vol. 15, no. 6, pp. 29325–29333, Dec. 2025, https://doi.org/10.48084/etasr.13875.

[29] J. Khodabakhsh, "MNIST Dataset." Kaggle, May 2019, [Online]. Available: https://www.kaggle.com/datasets/hojjatk/mnist-dataset.

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

[31] "CIFAR-10 - Object Recognition in Images." Kaggle, 2014, [Online]. Available: https://www.kaggle.com/c/cifar-10.