First vs. Second-Order Optimization in Shallow Neural Networks: A Systematic Study Across Problem Landscapes

Sezil Ağırbaş
Department of Computer Engineering
Hacettepe University
Ankara, Turkey
sezilagirbas25@hacettepe.edu.tr
Roketsan Missile Industries
Ankara, Turkey

Mehmet Önder Efe
Department of Computer Engineering
Hacettepe University
Ankara, Turkey
onderefe@gmail.com

***Abstract*—To ensure the accuracy and convergence of neural network training, it is essential to choose an appropriate optimization algorithm. First-order methods prevail in practical applications because of their computational efficiency, whereas second-order methods provide theoretical benefits in traversing intricate loss landscapes. This paper offers a systematic comparison of five optimization algorithms—Stochastic Gradient Descent (SGD), SGD with Momentum, Adam, Newton's method with diagonal Hessian approximation, and Limited-memory BFGS (L-BFGS)—across three distinct problem landscapes: convex (MNIST), non-convex bottleneck (Fashion-MNIST), and ill-conditioned (unnormalized CIFAR-10). Our experiments indicate that adaptive first-order methods exhibit strong performance in typical scenarios, attaining accuracies of up to 92.7% and 85.8% on convex and non-convex problems, respectively. Conversely, second-order methods demonstrate distinct, context-dependent benefits: Newton's method attains superior accuracy (23.9%) on ill-conditioned problems, markedly surpassing Adam (15.5%). Nonetheless, it performs poorly on non-convex landscapes (9.5% accuracy) because of vanishing gradients and saturating activations in bottleneck architectures, which compromise diagonal Hessian approximations. These findings offer evidence-based criteria for optimizer selection: Adam is a reliable default, momentum-based methods are crucial for non-convex optimization, and second-order methods are particularly advised for ill-conditioned situations lacking feature scaling.**

***Index Terms*—Optimization, Second-order methods, Gauss-Newton, Adam, Ill-conditioned problems, Loss landscape geometry**

I. Introduction

The training of neural networks is fundamentally a non-linear, non-convex optimization problem. As architectures grow in complexity, the efficiency of the optimizer determines not only the feasibility of the training process but also the generalization capability of the resulting model. Historically, first-order methods have dominated the field because they rely solely on the gradient of the loss function, making them computationally efficient for high-dimensional parameter spaces [1].

However, first-order methods are susceptible to the selection of learning rates and may encounter difficulties when dealing with ill-conditioned loss surfaces, where curvature exhibits considerable variance across different parameter directions [2]. In contrast, second-order methods, which utilize the Hessian matrix or its approximation to integrate curvature information, are theoretically advantageous in these situations. By adapting the step size to the local characteristics of the loss surface, these techniques may facilitate faster convergence.

Despite their theoretical advantages, second-order methods, for instance Newton's method, are rarely employed in deep learning applications, primarily because of the substantial expense associated with computing and inverting the complete Hessian matrix. Furthermore, these methods are susceptible to convergence at saddle points within non-convex landscapes [3], thereby diminishing their practical effectiveness. This has led to a reliance on more efficient alternatives. Diagonal Hessian estimates, for instance, streamline complexity from $O(n^2)$ to $O(n)$, and quasi-Newton methods such as L-BFGS approximate the necessary inverse Hessian by looking back at gradient history [4]. The Gauss-Newton method streamlines computations by employing outer products of gradients to approximate the Hessian; moreover, Levenberg-Marquardt damping introduces regularization, thereby promoting numerical stability and mitigating overshooting near saddle points [5].

The goal of this study is to bridge the gap between theory and practice by systematically comparing first-order and second-order optimization methods on shallow neural networks. We analyze five algorithms: SGD, SGD with Momentum, Adam, L-BFGS, and a diagonal Gauss-Newton method incorporating Levenberg-Marquardt damping, which uses exponential moving averages of squared gradients as curvature approximations. Shallow architectures facilitate a detailed examination of optimizer behavior, while remaining suitable for edge-computing and low-latency applications, where computational resources are limited.

The contributions of this work are three-fold:

1) **Rigorous Experimental Framework:** A fair evaluation environment with identical training samples, network initializations, and systematic hyperparameter tuning across all optimizers.
2) **Landscape Characterization:** Optimizer behavior analysis across convex (MNIST), non-convex bottleneck (Fashion-MNIST), and ill-conditioned (unnormalized CIFAR-10) landscapes.
3) **Empirical Guidelines:** Evidence-based optimizer selection criteria, demonstrating scenario-specific strengths and failure modes of diagonal Gauss-Newton methods.

II. Related Work

The development of neural network optimization has been motivated by the necessity to reconcile computational efficiency with convergence stability. This section covers significant advancements in first-order adaptive methods, second-order approaches, and the difficulties that loss landscape geometry brings.

A. First-Order Adaptive Methods

Classical Stochastic Gradient Descent (SGD) is memory-efficient, although it needs careful tuning of the learning rate and often takes a long time to converge on ill-conditioned problems [1]. Momentum-based acceleration [6] and Nesterov's accelerated gradient [7] addressed these issues by stacking up the velocity of past gradients.

Later, adaptive methods were developed to automate learning rates that were specific to each parameter. AdaGrad [8] worked well with sparse features, but it converged too quickly because of aggressive decay. RMSprop [9] and the Adam optimizer [10] reduced this problem by using exponential moving averages of squared gradients. Even though Adam is the de facto standard, recent research by Wilson et al. [11] shows that well-tuned SGD can provide better generalization for particular architectures. We build on this by testing these dynamics in shallow architectures. Recent refinements such as AdamW [12] have further optimized these adaptive mechanisms by decoupling weight decay, though standard Adam remains the primary baseline for general benchmarks.

B. Second-Order and Quasi-Newton Approaches

Second-order methods use Hessian information to take into account the curvature of the loss surface. Newton's method has quadratic convergence, but it costs too much computationally since Hessian inversion's complexity is $O(n^3)$ [2]. Hessian-free optimization [13] and quasi-Newton methods like L-BFGS [4] provide more tractable options because they use conjugate gradients or gradient history to get the approximation of the Hessian. Nevertheless, L-BFGS frequently encounters difficulties stemming from the noise present in stochastic mini-batches [14], as the approximation demands high-fidelity gradient differences. Martens [15] provides a comprehensive theoretical framework connecting these Gauss-Newton approximations to the Fisher information matrix, establishing the validity of second-order optimization in principle. Recent work has focused on diagonal approximations [16] and Kronecker-factored methods (K-FAC) [17] to make the most of the structure of neural networks.

C. Loss Landscape Geometry and the Saddle Point Problem

Dauphin and colleagues [3] observed that the principal challenge in non-convex optimization stems from the abundance of saddle points, rather than local minima. These points present a particular difficulty for Newton-style methods, which are inclined to traverse directions of negative curvature. Regularization strategies, including Levenberg-Marquardt damping [5], mitigate this issue by ensuring the Hessian approximation's positive definiteness.

Moreover, LeCun et al. [2] emphasized the importance of input normalization in preserving a beneficial condition number. In the absence of such scaling, the loss surface exhibits significant elongation, thereby impeding the efficacy of gradient-based methods. Tools such as PyHessian [18] have recently enabled granular analysis of these landscapes through Hessian eigenvalue computation, confirming the correlation between curvature properties and training dynamics. Li et al.'s recent visualization methods [19] furnish empirical support for the pathways optimizers take through intricate surfaces, thereby underscoring the significance of landscape-aware optimization.

D. Positioning of This Work

Although considerable research has been dedicated to individual optimizers, systematic comparisons across diverse problem geometries—encompassing convex, non-convex, and ill-conditioned landscapes—are still scarce. The majority of existing studies concentrate on deep architectures, where the application of second-order methods is often impractical. This research endeavors to fill this void by conducting a systematic comparison of first-order methods (SGD, SGD+Momentum, Adam) and second-order methods (diagonal Gauss-Newton, L-BFGS) across three deliberately constructed problem geometries, with the aim of identifying the scenario-specific benefits and drawbacks inherent in each approach. In particular, our analysis reveals a *damping dominance* phenomenon in diagonal second-order approximations under non-convex conditions, providing new insight into why such methods fail in the presence of vanishing gradients and saturating activations.

III. Methodology

This section details the neural network architectures, the mathematical formulation of the optimization methods, and the experimental design used to evaluate optimizer performance across distinct loss landscapes.

A. Network Architectures and Problem Design

We evaluate on three standard image classification benchmarks: MNIST (handwritten digits, 28×28 grayscale, 10 classes, 60K training samples), Fashion-MNIST (clothing items, same dimensions and class count, 60K training samples), and CIFAR-10 (natural images, 32×32 RGB, 10 classes,

50K training samples). Subsets of 5,000 samples (MNIST, Fashion-MNIST) and 3,000 samples (CIFAR-10) were used to ensure computational tractability for second-order methods while maintaining statistical significance, consistent with common practices in optimizer benchmarking studies [11].

We designed three experimental scenarios to characterize optimizer behavior across diverse landscapes:

1) **Convex-like Landscape (MNIST):** A two-layer network $[784 \to 128 \to 10]$ with ReLU activation, trained on normalized MNIST data. The relatively simple architecture and well-scaled inputs create a nearly convex loss landscape. Training utilized 5,000 samples with He initialization [20] to ensure optimal gradient flow.
2) **Non-Convex Landscape (Fashion-MNIST):** A bottleneck architecture $[784 \to 128 \to 32 \to 128 \to 10]$ with Tanh activation. The information bottleneck forces compressed representations, while the saturating Tanh function creates regions of vanishing gradients and numerous saddle points. Training utilized 5,000 samples with Xavier initialization [21].
3) **Ill-Conditioned Landscape (CIFAR-10):** A three-layer network $[3072 \to 256 \to 128 \to 10]$ with Sigmoid activation. Inputs were intentionally kept unnormalized (pixel values in $[0, 255]$) to create extreme differences in gradient magnitudes, resulting in a highly ill-conditioned Hessian. Training utilized 3,000 samples with Xavier initialization.

B. First-Order Optimization Methods

We evaluated three widely used first-order algorithms. Stochastic Gradient Descent (SGD) updates parameters according to Eq. (1):

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t) \tag{1}$$

where η is the learning rate. SGD with Momentum incorporates a velocity vector v with coefficient μ (typically 0.9) to accelerate convergence, as expressed in Eq. (2):

$$v_{t+1} = \mu v_t - \eta \nabla L(\theta_t), \quad \theta_{t+1} = \theta_t + v_{t+1} \tag{2}$$

Adam [10] utilizes adaptive per-parameter learning rates based on the first moment m_t and second moment v_t of the gradients. The bias-corrected estimates in Eq. (3) are used to compute the final update in Eq. (4):

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{3}$$

$$\theta_{t+1} = \theta_t - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{4}$$

C. Second-Order Optimization Methods

The core contribution of this benchmarking is the evaluation of a *Diagonal Gauss-Newton method with Levenberg-Marquardt Damping*. To avoid the $O(n^3)$ cost of full Hessian inversion, we implement a diagonal approximation using exponential moving averages of squared gradients:

1) **Curvature Estimation:** The diagonal Hessian is approximated via Eq. (5):

$$h_t = \beta h_{t-1} + (1-\beta) g_t^2 \tag{5}$$

2) **Bias Correction:** Similar to Adam, we apply bias correction in Eq. (6) to account for initialization at zero:

$$\hat{h}_t = h_t / (1-\beta^t) \tag{6}$$

3) **Damped Newton Update:** The final parameter update in Eq. (7) incorporates Levenberg-Marquardt damping:

$$\theta_{t+1} = \theta_t - \eta \cdot \frac{g_t}{\hat{h}_t + \lambda} \tag{7}$$

The damping factor λ regularizes the step size and ensures numerical stability near saddle points. Unlike Adam, which uses $\sqrt{\hat{v}_t}$ (an L_2 normalization effect), our Gauss-Newton implementation divides directly by $\hat{h}_t$, maintaining the quadratic scaling where the update is inversely proportional to curvature.

L-BFGS [4] was implemented via `scipy.optimize` using a limited-memory history ($m = 10$) to approximate the inverse Hessian. Consistent with its sensitivity to stochastic noise [14], L-BFGS was evaluated using full-batch gradients.

D. Experimental Setup and Hyperparameters

Implementation: All optimizers except L-BFGS (namely SGD, SGD+Momentum, Adam, and the diagonal Gauss-Newton method) were implemented from scratch in Python/NumPy to ensure full control over the update mechanics. Training was conducted for 20–25 epochs using a mini-batch size of 128, a consistent random seed (27), and cross-entropy loss with a softmax output layer. In contrast, L-BFGS was evaluated using full-batch gradients via the `scipy.optimize` library to maintain the deterministic requirements of the quasi-Newton update [14].

Hyperparameter Selection: Learning rates (η) and damping coefficients (λ) were selected via a systematic grid search over $\eta \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 0.5\}$ and $\lambda \in \{0.01, 0.1, 1.0\}$ for the Newton method, selecting the configuration that maximized validation accuracy. Table I summarizes the optimal configuration for each landscape. Notably, the ill-conditioned CIFAR-10 experiment required significantly lower learning rates (by a factor of 10^3–10^4) compared to the normalized datasets. This is because the unnormalized pixel values ($[0, 255]$) generate extreme gradient magnitudes that lead to immediate divergence when standard learning rate scales are applied.

IV. Results and Discussion

This section presents the comparative performance analysis of the investigated optimizers across three distinct experimental landscapes and interpretation of the results. All experiments utilized consistent random seeds to ensure reproducibility.

A. Overall Performance Summary

The final test accuracies are summarized in Table II. The results reveal that optimizer efficacy is highly sensitive to the problem geometry and data conditioning.

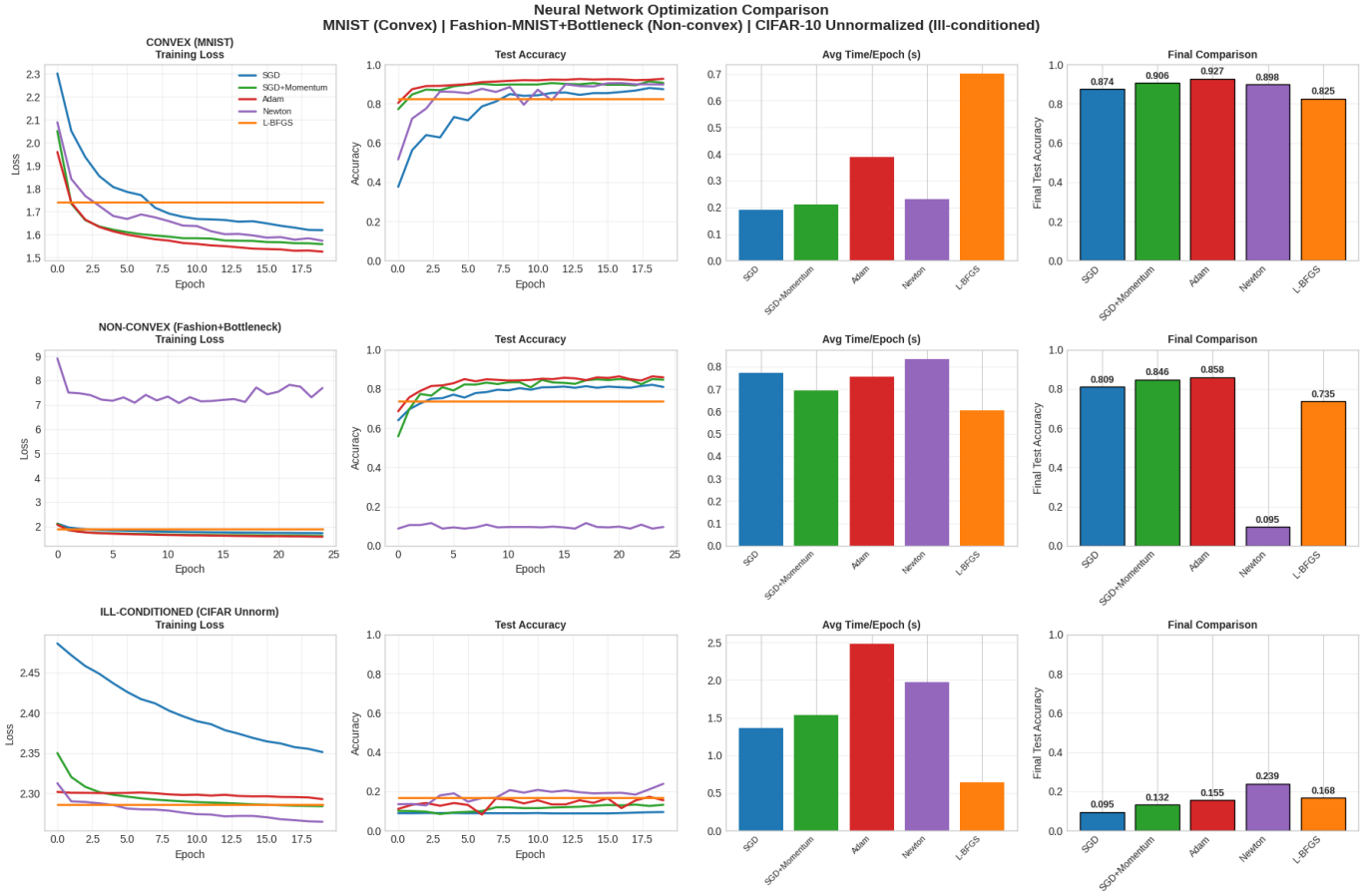

Fig. 1. Training dynamics and convergence behavior across three experimental landscapes. The middle row (Fashion-MNIST) clearly illustrates the catastrophic stagnation of the Gauss-Newton method, while the bottom row (CIFAR-10) demonstrates its superior performance in ill-conditioned scenarios.

TABLE I
HYPERPARAMETER CONFIGURATION FOR EACH EXPERIMENT

Optimizer	**MNIST**	**Fashion**	**CIFAR-10**
SGD	η=0.5	η=0.05	η=10^{-5}
SGD+Mom.	η=0.1	η=0.05	η=10^{-5}
Adam	η=0.003	η=0.002	η=10^{-4}
Newton	η=0.1, λ=0.1	η=0.5, λ=0.01	η=0.01, λ=1.0
L-BFGS	iter=50	iter=20	iter=10

Note: Momentum μ=0.9 for SGD+Mom.; β=0.9 for Newton across all experiments.

TABLE II
TEST ACCURACY COMPARISON ACROSS PROBLEM TYPES

Optimizer	**MNIST**	**Fashion**	**CIFAR-10**
SGD	0.874	0.809	0.095
SGD+Momentum	0.906	0.846	0.132
Adam	**0.927**	**0.858**	0.155
Newton (Proposed)	0.898	0.095	**0.239**
L-BFGS	0.825	0.735	0.168

Bold indicates best performance.

B. Convex-like Landscape (MNIST)

On the well-conditioned MNIST dataset, all optimizers achieved reasonable performance. The proposed diagonal Gauss-Newton method demonstrated competitive accuracy (89.8%), reaching 90% of its final value within 3 epochs. This suggests that in well-behaved landscapes, second-order curvature information provides effective step-size calibration comparable to adaptive first-order methods.

C. Non-Convex Landscape: The Saddle Point Trap (Fashion-MNIST)

The bottleneck architecture revealed a critical failure mode for second-order approximations. While Adam and SGD+Momentum successfully navigated the landscape (85.8% and 84.6%, respectively), the Gauss-Newton method failed catastrophically (9.5%).

This failure is attributed to vanishing gradients amplified by the information bottleneck. As gradients g_t approach zero in saturating regions, the curvature estimate $\hat{h}_t$ gradually diminishes due to the exponential moving average. Despite Levenberg-Marquardt damping, when both $g_t \approx 0$ and $\hat{h}_t \approx 0$,

TABLE III
EPOCHS TO REACH 90% OF FINAL ACCURACY

Optimizer	MNIST	Fashion	CIFAR-10
SGD	7	3	0*
SGD+Momentum	1	2	7
Adam	1	2	2
Newton	3	0*	19
L-BFGS	0*	0*	0*

**0 indicates single-pass convergence (L-BFGS) or immediate stagnation.*

the update $\Delta\theta = \eta \cdot g_t/(\hat{h}_t + \lambda)$ becomes dominated by the damping term λ, leading to near-zero updates that fail to escape flat regions. Momentum-based methods, however, accumulate velocity to traverse these plateaus.

D. Ill-Conditioned Landscape: Second-Order Resilience (CIFAR-10)

In the ill-conditioned CIFAR-10 experiment, the proposed method achieved the highest accuracy (23.9%), significantly outperforming Adam (15.5%). The unnormalized data causes extreme variation in gradient scales. While first-order methods oscillate along narrow valleys, the diagonal Gauss-Newton approximation effectively normalizes the varying scales by dividing directly by the curvature $\hat{h}_i$ rather than its square root. This quadratic scaling proves more effective than Adam's L_2-style normalization in severely ill-conditioned scenarios.

E. Convergence Speed Analysis

Table III details the efficiency of each method in reaching 90% of its final accuracy.

F. Optimizer Performance Across Problem Geometries

Our empirical findings demonstrate that the effectiveness of the optimizer is directly related to the geometric properties of the loss landscape. No single algorithm dominates across all scenarios, underscoring the necessity of aligning optimization strategies with specific landscape topologies. Adam demonstrates the most consistent performance (Fig. 1), achieving top results on convex (92.7%) and non-convex (85.8%) problems through its hybrid momentum and adaptive scaling mechanisms.

In contrast, the diagonal Gauss-Newton method exhibits highly specialized behavior. Its superior performance in the ill-conditioned CIFAR-10 scenario (Table II, 23.9% vs. Adam's 15.5%) confirms that when curvature varies dramatically across the parameter space, second-order information provides a more accurate descent direction. The direct curvature normalization ($1/\hat{h}_t$) in Gauss-Newton allows for more aggressive steps in flat directions compared to the square-root scaling used by Adam.

G. Analysis of Newton's Failure in Non-Convex Landscapes

The Gauss-Newton method's poor performance on the Fashion-MNIST bottleneck architecture, which only achieved

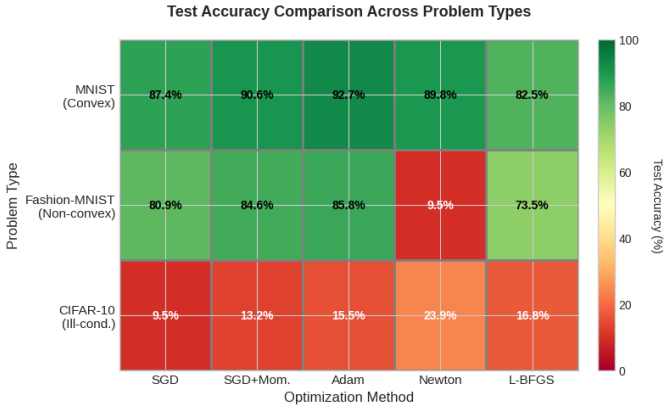

Fig. 2. Comparative performance summary across problem geometries. The heatmap illustrates the scenario-specific advantages of each optimizer, with Newton excelling in ill-conditioned scenarios while failing in non-convex landscapes.

9.5% accuracy, highlights a key limitation of diagonal second-order approximations. The training loss curves we observed (Fig. 1, middle row) clearly show this issue:

1) **Vanishing Gradients and Damping Dominance:** The saturating Tanh activations generate near-zero gradients, causing the curvature estimate $\hat{h}_t$ to diminish. In this case, the damping term λ is what mostly controls the update $\Delta\theta \approx \eta \cdot g_t/\lambda$. This is why we see very small updates and horizontal accuracy lines in our results.
2) **Lack of Negative Curvature Sensing:** Diagonal Hessian approximations also cannot pick up on off-diagonal information or negative eigenvalues. This stops the optimizer from finding the right descent directions to get out of saddle points, which is why the Fashion-MNIST accuracy plots show an immediate halt.

In contrast, momentum-based methods use accumulated velocity to traverse these flat areas, as shown by their steady upward accuracy trajectories in Fig. 1.

H. Practical Guidelines for Optimizer Selection

Based on our comparative analysis (Fig. 2), we propose the following guidelines:

- **Adam** is the robust default for scenarios with unknown conditioning.
- **SGD with Momentum** is essential for navigating non-convex landscapes where escape from saddle points is a priority.
- **Diagonal Gauss-Newton methods** should be prioritized for severely ill-conditioned problems (e.g., unnormalized data), provided the architecture avoids excessive bottlenecking.
- **L-BFGS** offers a stable quasi-Newton alternative for deterministic, full-batch optimization on smaller datasets.

I. Limitations

This study concentrated on shallow architectures to isolate second-order dynamics, with direct relevance to edge-computing environments. Contemporary deep architectures

utilizing batch normalization may alleviate certain instances of ill-conditioning noted herein. We acknowledge that the unnormalized CIFAR-10 scenario is an intentionally extreme case designed to stress-test optimizer robustness; while it may favor second-order methods, it reflects real-world scenarios where preprocessing is unavailable, such as embedded systems with limited capabilities. Recent work such as AdaHessian [22] has extended diagonal Hessian estimates to deep architectures through spatial averaging. Our findings on the damping dominance phenomenon (Section IV-G) provide theoretical grounding for why such extensions become necessary as network depth increases.

V. Conclusion

This paper systematically compared first and second-order optimization methods for training shallow neural networks on three different types of landscapes: convex, non-convex bottleneck, and ill-conditioned. We have defined the operational limits and scenario-specific advantages of each algorithm by isolating these specific challenges.

Our experiments show that the effectiveness of an optimizer is directly related to problem geometry. The key findings of this study are:

- **Robustness of Adaptive First-Order Methods:** Adam consistently achieved top results on convex (92.7%) and non-convex (85.8%) problems, confirming its role as a strong general-purpose default.
- **Second-Order Resilience in Ill-Conditioned Scenarios:** The diagonal Gauss-Newton method with Levenberg-Marquardt damping achieved the best accuracy (23.9%) on unnormalized CIFAR-10, surpassing Adam by 8.4 percentage points, demonstrating that direct curvature normalization outperforms L_2-style scaling under severe ill-conditioning.
- **Vulnerability of Diagonal Hessian Approximations:** The Gauss-Newton method failed catastrophically (9.5%) on non-convex bottleneck architectures due to damping dominance, where simultaneously vanishing gradients and curvature estimates render the optimizer unable to escape saddle points.

Based on these results, we give practitioners the following evidence-based guidelines: Adam is the best general-purpose optimizer; momentum-based methods are necessary for dealing with complex, non-convex landscapes; and second-order methods are specifically recommended for severely ill-conditioned problems where traditional adaptive scaling is not enough.

Future work will be on developing hybrid optimization strategies that can switch between first-order momentum and quasi-Newton methods as needed. This kind of framework could potentially combine Adam's strength in early-stage exploration and Newton's accuracy for fine-tuning in poorly conditioned minima. Additionally, extending these experiments to deeper architectures and normalized data variants would further validate the generality of the observed phenomena.

References

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

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

[3] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio, "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2014, pp. 2933–2941.

[4] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical Programming*, vol. 45, no. 1–3, pp. 503–528, 1989.

[5] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of Applied Mathematics*, vol. 2, no. 2, pp. 164–168, 1944.

[6] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.

[7] Y. Nesterov, "A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$," *Doklady AN USSR*, vol. 269, pp. 543–547, 1983.

[8] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[9] T. Tieleman and G. Hinton, "Lecture 6.5-RMSprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Networks for Machine Learning*, vol. 4, no. 2, pp. 26–31, 2012.

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learning Representations (ICLR)*, 2015.

[11] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht, "The marginal value of adaptive gradient methods in machine learning," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 4148–4158.

[12] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2019.

[13] J. Martens, "Deep learning via Hessian-free optimization," in *Proc. 27th Int. Conf. Machine Learning (ICML)*, 2010, pp. 735–742.

[14] N. N. Schraudolph, J. Yu, and S. Günter, "A stochastic quasi-Newton method for online convex optimization," in *Proc. 11th Int. Conf. Artificial Intelligence and Statistics (AISTATS)*, 2007, pp. 436–443.

[15] J. Martens, "New insights and perspectives on the natural gradient method," *Journal of Machine Learning Research*, vol. 21, no. 146, pp. 1–76, 2020.

[16] R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer, "A stochastic quasi-Newton method for large-scale optimization," *SIAM Journal on Optimization*, vol. 26, no. 2, pp. 1008–1031, 2016.

[17] J. Martens and R. Grosse, "Optimizing neural networks with Kronecker-factored approximate curvature," in *Proc. 32nd Int. Conf. Machine Learning (ICML)*, 2015, pp. 2408–2417.

[18] Z. Yao, A. Gholami, K. Keutzer, and M. W. Mahoney, "PyHessian: Neural networks through the lens of the Hessian," in *Proc. IEEE Int. Conf. Big Data*, 2020, pp. 581–590.

[19] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2018, pp. 6389–6399.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, 2015, pp. 1026–1034.

[21] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artificial Intelligence and Statistics (AISTATS)*, 2010, pp. 249–256.

[22] Z. Yao, A. Gholami, S. Shen, M. Mustafa, K. Keutzer, and M. W. Mahoney, "AdaHessian: An adaptive second order optimizer for machine learning," in *Proc. AAAI Conf. Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 10665–10673.