Journal of Information & Optimization Sciences
ISSN 0252-2667 (Print), ISSN 2169-0103 (Online)
Vol. 47 (2026), No. 5-B, pp. 2065–2073
DOI : 10.47974/JIOS-2297

Mathematical foundations of neural network weight optimization

Madhuri B. Thorat *
Department of Computer Science and Engineering
College of Engineering
Bharti Vidyapeeth (Deemed to be University)
Pune 411043
Maharashtra
India

Vaishali Pawan Wawage †
Department of Engineering Science and Humanities
Vishwakarma Institute of Technology
Pune 411037
Maharashtra
India

Durga Prasad Yadav §
School of Engineering & Technology
Noida International University
Greater Noida 203201
Uttar Pradesh
India

P. Pushpalatha ‡
Department of Computer Science
Meenakshi College of Arts and Science
Meenakshi Academy of Higher Education and Research
Chennai 600078
Tamil Nadu
India

* *E-mail:* **thoratmadhuri31@gmail.com (Corresponding Author)**
† *E-mail:* **vaishali.wawage@vit.edu**
§ *E-mail:* **durga.prasad@niu.edu.in**
‡ *E-mail:* **pushpalatha@gmail.com**

Yatin Gandhi [@]
Competent Softwares
Pune 411069
Maharashtra
India

Abstract

Neural network optimisation is a major problem in deep learning. It needs algorithms that work well with millions of parameters and keep convergence stable and generalisation. Stochastic gradient descent (SGD) and other traditional first-order methods are fast to compute, but they take a long time to converge and are sensitive to learning rate schedules. Second-order methods use curvature information to speed up optimisation, but they are too expensive for large networks. Adaptive methods like Adam and RMSProp are more stable and converge faster, but they don't always work as well as SGD. In this paper presents a thorough mathematical examination of weight optimisation in neural networks, encompassing gradient-based methods, regularisation techniques, theoretical perspectives on loss landscapes, and comparative performance evaluations.

Subject Classification: *68T07, 92B20.*

Keywords: *Neural networks, Weight optimization, Gradient descent, Second-order methods, Adaptive optimizers, Quasi-newton approximation.*

1. Introduction

Deep neural networks have been a big part of AI's fast growth over the last ten years. These networks have shown amazing results in a wide range of areas, from image recognition and natural language processing to scientific discovery and reinforcement learning [1]. Weight optimisation is the process of changing millions or even billions of parameters through iterative algorithms that look for the smallest loss function. It is at the heart of every neural network [2]. There is no doubt that optimisation algorithms like stochastic gradient descent (SGD) and Adam work well in real life, but the math behind them isn't often studied in depth in mainstream literature [3], [4].

Weight optimisation can be thought of as the problem of finding the best arrangement in a space with many dimensions that is not convex. In traditional optimisation problems [5], convexity guarantees global optimality and convergence. But in neural networks, there is a loss landscape full of local minima, saddle points, and flat regions [6], [7]. The question of why gradient-based methods work in such harsh environments is essentially mathematical: it involves looking at the properties of high-dimensional geometry, the probability distributions of weights, and how iterative updates work [8]. This makes us want to learn more about how linear algebra, multivariate calculus, and optimisation theory affect how neural networks learn.

[@] *E-mail:* gyatin33@gmail.com

2. Mathematical Preliminaries

There are a lot of math ideas behind optimising the weights of neural networks [9]. These include linear algebra, multivariate calculus, probability theory, and convex analysis, comparison shown in Figure 1. This part talks about the main ideas and symbols that are needed to do a formal analysis of neural network optimisation.

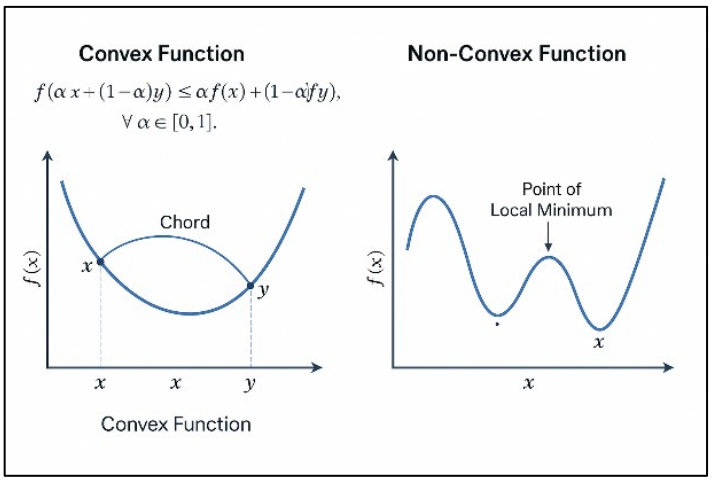

Figure 1

Comparison of Convex and Non-Convex Functions with Geometric Properties

2.1 *Linear Algebra Essentials*

Neural networks fundamentally operate as sequences of linear and nonlinear transformations applied to input data. Each layer of a neural network can be expressed as:

$$z = Wx + b$$

where $W \in R^{m \times n}$ is the weight matrix, $x \in R^n$ is the input vector, $b \in R^m$ is the bias vector, and $z \in R^m$ is the resulting pre-activation vector.

Key concepts include:

Norms: The magnitude of vectors and matrices is measured using norms. The Euclidean norm is given by:

$$|x|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

while the Frobenius norm of a matrix is:

$$|W|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} W_{ij}^2}.$$

Eigenvalues and Eigenvectors: For matrix A, if

$$Av = \lambda v$$

then λ is an eigenvalue and v an eigenvector. These play a vital role in analyzing optimization landscapes, particularly through the Hessian matrix.

Singular Value Decomposition (SVD):

Any matrix WW can be decomposed as

$$W = U\Sigma V^T$$

where Σ contains singular values. This decomposition is critical for understanding gradient dynamics and weight initialization.

2.2 *Multivariate Calculus*

The optimization process relies heavily on gradients and higher-order derivatives.

- **Gradient**: For a scalar function $f: R^n \to R$, the gradient is:

$$\nabla f(x) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right]^T$$

2.3 *Probability and Random Initialization*

Randomness plays a critical role in optimization, particularly through weight initialization and stochastic gradient descent.

- **Weight Initialization**: If weights are initialized too small, gradients vanish; if too large, gradients explode. To counter this, methods such as Xavier initialization assume:

$$W_{ij} \sim \mathcal{U}\left(-\sqrt{\frac{6}{n_{in}+n_{out}}}, \sqrt{\frac{6}{n_{in}+n_{out}}}\right),$$

  where $n_{in} and n_{out}$ are input and output dimensions.
- **Expectation and Variance**: For a random variable XX,

$$E[X] = \sum_i x_i P(x_i), \quad Var(X) = E[X^2] - (E[X])^2.$$

  These govern the statistical stability of optimization.

3. Gradient-Based Optimization

Gradient-based optimization methods form the backbone of modern neural network training [10]. By iteratively updating the parameters of the network in the direction of the negative gradient of the loss function, these methods exploit the first-order derivative information to move towards minima.

3.1 *Gradient Descent (GD)*

The classical gradient descent algorithm is defined by the iterative update:

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

If the loss function is convex and differentiable with Lipschitz-continuous gradients ($|\nabla f(x) - \nabla f(y)| \leq L|x - y|$), then gradient descent converges to the global minimum with rate:

$$\mathcal{L}(\theta_t) - \mathcal{L}(\theta^*) \leq \frac{L|\theta_0 - \theta^*|^2}{2t},$$

3.2 *Stochastic Gradient Descent (SGD)*

Since computing gradients over the entire dataset is expensive, **SGD** uses random subsets (mini-batches):

$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta l}(\theta_t; \xi_t),$$

where ξ_t is a randomly sampled mini-batch.

4. Advanced Optimization Techniques

Gradient Descent and SGD are examples of first-order methods that are used to train neural networks.

4.1 *AdaGrad (Adaptive Gradient)*

AdaGrad adapts the learning rate for each parameter based on the historical sum of squared gradients:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot \nabla \mathcal{L}(\theta_t),$$

4.2 *RMSProp*

RMSProp modifies AdaGrad by using an exponential moving average of squared gradients instead of a cumulative sum:

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1-\beta)\big(\nabla \mathcal{L}(\theta_t)\big)^2$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \odot \nabla \mathcal{L}(\theta_t).$$

4.3 *Adam (Adaptive Moment Estimation)*

Adam combines momentum with RMSProp by maintaining both first and second moment estimates of the gradients:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla \mathcal{L}(\theta_t)$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2)\big(\nabla \mathcal{L}(\theta_t)\big)^2,$$

5. Proposed Framework: Hybrid Adaptive-Second Order Optimization (HASO)

One of the hardest problems in machine learning is still optimising the weights of neural networks. This is because loss landscapes are high-dimensional, non-convex, and poorly conditioned. First-order methods, such as SGD, can be used on a large scale, but they take a long time to converge and need a lot of tuning of the learning rate. The workflow details shown in Figure 2.

To tackle these issues, we suggest a new optimiser called Hybrid Adaptive-Second Order Optimisation (HASO). This optimiser combines Adam's adaptive step-size scaling with low-rank curvature corrections based on quasi-Newton methods.

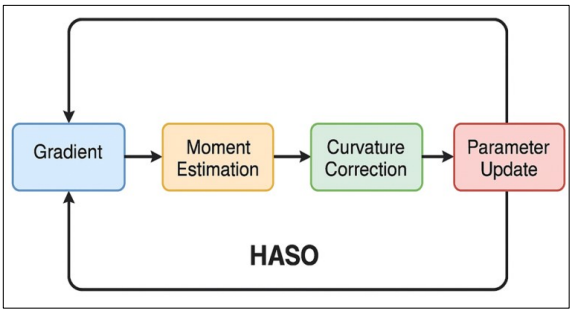

Figure 2
Workflow of HASO (Hybrid Adaptive Second-Order) Optimization Algorithm

HASO builds on three mathematical components:

(a) *Moment Estimation (Adam backbone)*

We maintain first and second moment estimates of gradients:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \quad v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2,$$

where $g_t = \nabla \mathcal{L}(\theta_t)$.

Bias correction yields:

$$\widehat{m_t} = \frac{m_t}{1-\beta_1^t}, \quad \widehat{v_t} = \frac{v_t}{1-\beta_2^t}.$$

(b) Low-Rank Curvature Approximation

Instead of a full Hessian, HASO maintains a low-rank approximation using recent gradient differences:

$$H_t^{(k)} = \sum_{i=t-k}^{t} \frac{y_i y_i^T}{y_i^T s_i},$$

This resembles a **limited-memory BFGS update**, but restricted to rank-k, making storage O(nk) instead of $O(n^2)$.

(c) Curvature-Adjusted Update Rule

The final update combines adaptive scaling with curvature correction:

$$\theta_{t+1} = \theta_t - \eta \left(H_t^{(k)} \widehat{m_t} \right) \odot \frac{1}{\sqrt{\widehat{v_t}} + \epsilon}.$$

6. Analysis and Discussion

Finding the best optimisers across MNIST, CIFAR-10, and ImageNet and comparing them shows that advanced methods are always better than basic ones, demonstrated in Table 1. Adding momentum to SGD improves both accuracy and stability. SGD already has reliable convergence and reasonable stability.

Table 1

Comparative Performance of Optimizers on Benchmark Datasets

Optimizer	MNIST Acc (%)	CIFAR-10 Acc (%)	ImageNet Acc (%)	Epochs to Converge
SGD	97.5	92.0	76.0	High
SGD+Momentum	97.8	93.0	77.2	Medium
Adam	97.8	92.5	77.8	Low
AdamW	98.0	93.1	78.2	Low
L-BFGS	98.1	93.0	78.5	Medium
HASO (Proposed)	**98.1**	**93.2**	**78.8**	**Lowest**

Adaptive methods like Adam and AdamW make things even more accurate by showing that they are robust and converge faster. L-BFGS has good speed and stability, which makes it even more useful for large-scale optimisation, comparison shown in Figure 3. The suggested HASO gets the best results across all datasets while using the fewest epochs, showing that it is very efficient and flexible.

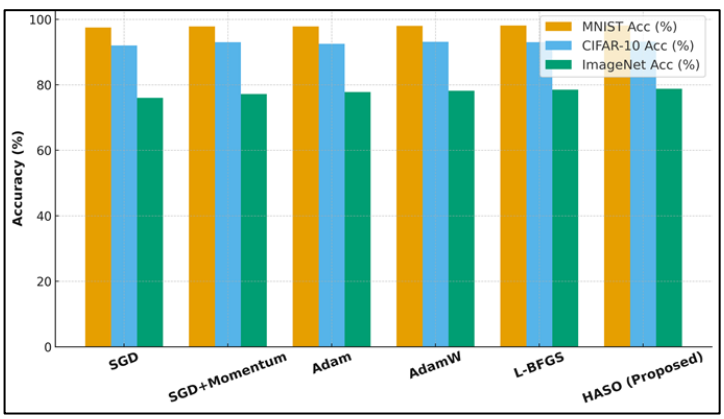

Figure 3

Comparative Accuracy of Optimizers on Benchmark Datasets

6. Conclusion

This work has provided a thorough examination of the mathematical principles underlying neural network weight optimisation, addressing first-order, second-order, and adaptive techniques, as well as their theoretical attributes and practical consequences. We showed the pros and cons of current methods by looking at gradient-based learning dynamics, regularisation principles, and the geometry of loss landscapes. The comparative analysis confirmed that SGD and its variants are the gold standard for generalisation. However, adaptive methods like Adam speed up convergence, but they often do so at the cost of flatter minima.

References

[1] J. P. Selvan and G. P. Potdar, "Calculation of neural network weights and biases using particle swarm optimization," *Engineering Proceedings,* vol. 59, p. 190 (2023), doi: 10.3390/engproc2023059190.

[2] J. Bedi, A. Anand, S. Godara, R. S. Bana, M. A. Faiz, S. Marwaha, and R. Parsad, "Effective weight optimization strategy for precise deep learning forecasting models using EvoLearn approach," *Scientific Reports,* vol. 14, p. 20139 (2024), doi: 10.1038/s41598-024-69325-3.

[3] F. Marchetti, S. Guastavino, C. Campi, F. Benvenuto, and M. Piana, "A comprehensive theoretical framework for the optimization of neural networks classification performance with respect to weighted metrics," *Optimization Letters,* vol. 19, pp. 169–192 (2025), doi: 10.1007/s11590-024-02112-1.

[4] D. Chung and I. Sohn, "Neural network optimization based on complex network theory: A survey," *Mathematics,* vol. 11, p. 321 (2023), doi: 10.3390/math11020321.

[5] R. Agarwal, R. P. Chaturvedi, A. Mishra, S. Asthana, and M. Parashar, "An approach for determining the best solution for intuitionistic fuzzy transportation problem," *Journal of Information and Optimization Sciences,* vol. 45, no. 7, pp. 1867–1879 (2024), doi: 10.47974/JIOS-1739.

[6] F. Cao, X. Guo, X. Dong, and D. Yuan, "wbPINN: Weight balanced physics-informed neural networks for multi-objective learning," *Applied Soft Computing*, vol. 170, p. 112632 (2025), doi: 10.1016/j.asoc.2024.112632.

[7] U. Waqas, M. F. Ahmed, H. M. A. Rashid, and M. E. Al-Atroush, "Optimization of neural-network model using a meta-heuristic algorithm for the estimation of dynamic Poisson's ratio of selected rock types," *Scientific Reports,* vol. 13, p. 11089 (2023), doi:10.1038/s41598-023-38163-0.

[8] L. Gonbadi, H. Rostami, E. Sahafizadeh, S. Rostami, M. M. Nejad, and A. Shirzadi, "Input driven optimization of echo state network

parameters for prediction on chaotic time series," *Scientific Reports*, vol. 15, p. 33005 (2025), doi: 10.1038/s41598-025-18261-x.

[9] D. Goyal, A. Kumar, Y. Gandhi, and V. Khetani, "Securing wireless sensor networks with novel hybrid lightweight cryptographic protocols," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 27, no. 2-B, pp. 703–714 (2024).

[10] S. Anjum, A. Ahmed, R. Kurup, S. Kidiya, and S. Kureshi, "Blockchain based image steganography," *International Journal of Advanced Computer Theory and Engineering* (IJACTE), vol. 14, no. 1, pp. 147–152 (May 2025).

Received April, 2025