International Journal of Mathematics and Computer Research

ISSN: 2320-7167

Volume 14 Special Issue 03 March 2026, Page no. – 123-127

Impact Factor: 8.751

DOI: 10.47191/ijmcr/v14iSPC3.26

Optimization in Artificial Intelligence and Deep Learning — Algorithms, Theory, and Practice

Nikumbha Neha Ravindranath, Kartha Pearly Prasoon

Department of Mathematics, Dr. D. Y. Patil, Arts, Commerce & Science College, Pimpri, Pune, Maharashtra, India

ARTICLE INFO

Published Online:
16 March 2026

Corresponding Author:
Nikumbha Neha Ravindranath

ABSTRACT

The core of machine learning and deep learning models relates to optimization. The present paper summarizes the principal categories of optimization algorithms in AI and deep learning and describes the mathematical concepts underpinning them in non-technical language, and relates these mathematical concepts to training recipes and engineering considerations. We discuss first-order stochastic algorithms (SGD and variations), adaptive algorithms (AdaGrad, RMProp, Adam)(4)(5) momentum and acceleration, second-order concepts and approximations (Newton, quasi-Newton, natural gradient, K-FAC), and recent issues like learning-rate schedules, batch normalization behavior, sharp vs. flat minima, generalization, and training large models. Pseudo code, best practice hyper parameters, and practical traps are given to ensure that this paper serves as both a conceptual primer, as well as a practical guide.

KEYWORDS: Deep learning, Stochastic Gradient Descent (SGD), Adaptive methods, Adam optimizer, Momentum, Second-order optimization, Natural gradient, K-FAC, Learning-rate schedules, Regularization, Sharpness-aware minimization (SAM), Batch size scaling, Neural network optimization, Weight decay, Practical training recipes.

1. INTRODUCTION

Artificial intelligence (AI) and deep learning are about optimization, in which one aims to minimize a loss function in order to realize optimal model performance. The complexity of this issue consists in the fact that it is highly dimensional, non-convex, and stochastic, and therefore, the construction and analysis of optimization algorithms is difficult and necessary. The theoretical basis and practical uses of optimization algorithms that power the contemporary AI models, including classical gradient-based methods and adaptive and curvature-aware ones applied in deep neural networks, are the focus of this study. Past studies have done a thorough survey of the methods of optimization starting with the stochastic approximation suggested by Robbins and Monro (1951)(1), the momentum algorithms proposed by Polyak (1964)(2) and Nesterov (1983)(3), and then to adaptive optimizers like AdaGrad (Duchi et al., 2011)(4), RMSProp, and Adam (Kingma and Ba, 2015)(5). More recent advances like decoupled weight decay (Loshchilov and Hutter, 2019)(6), sharpness-sensitive optimisation (Foret et al., 2020)(9), and curvature-based methods, including K-FAC (Martens and Grosse, 2015), have enhanced the insight into convergence, scalability, and generalization during deep learning optimization. The study rationale is based on the necessity to bring theoretical knowledge and engineering practice together. Although there are numerous algorithms, in practice, practitioners do not know when and how to apply each optimizer to achieve the best results. It is then important to bridge the gap between mathematical intuition and implementation to create effective and stable AI systems. This paper is aimed at providing a detailed but easy-to-understand overview of optimization methods in artificial intelligence and deep learning. It will seek to describe the theory behind it, a comparison of algorithmic methods and practical training strategies balancing performance and generalization.

It is a method of exploration, which has a theoretical survey and empirical validation. The theoretical part compiles literature on the existing optimization models and their underlying assumptions, update rules, and convergence properties. The empirical part involves controlled benchmark experiments through the use of standardized architectures including ResNet and Transformer models and comparison of optimizers with consistent hyper parameter setups. To investigate the current advances in training efficiency, stochastic gradient-based methods were selected because of their magnitude and demonstrated effectiveness in large-scale learning, and adaptive and curvature-aware

methods were incorporated. The mean performance measures (accuracy, convergence rate, and variance) in more than one trial were statistically compared to guarantee reliability.

2. PROBLEM SETUP AND NOTATION

Let dataset $D=\{(x_i,y_i)\}$, $i=1,2,3,\ldots N$ Let model parameters be $\theta\in R^d$. We define a loss
$\ell(\theta; x,y)$ (e.g., cross-entropy). The empirical risk is

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N} l(\theta;\, x_i, y_i)$$

Training seeks min $L(\theta)$. Computing full gradient $\nabla L(\theta)$ is expensive for large N; stochastic gradients $g(\theta)=\nabla l\ (\theta;\ x_i)$ computed on mini batches are used.

Key notation:

- η: learning rate (step size).
- m_t, v_t: first/second moment estimates (adaptive methods).
- B: minibatch size.
- $\|\cdot\|$: Euclidean norm.

3. METHODOLOGY

3.1 First Order Stochastic Method

i. Stochastic Gradient Descent (SGD)

Algorithm idea: take noisy gradient steps using minibatches:

$$\Theta_{t+1}=\theta_t-\eta_t\, g_t\ ,$$

where g_t is gradient on minibatch.

Why it works:

This type of noise in gradients tends to get around saddle points and certain sharp local minima; in suitable step-size decay, SGD can approach stationary points under relatively weak conditions.

Practical notes:

• In deep learning, most of the time use momentum (see next section).

• Common learning initial values: 0.1 -0.0001 based on model and scaling.

Rule To minimize noise scale, learners should use large batches, which may necessitate scaling learning rates.

ii. SGD with momentum (Polyak / Heavy-ball)(2)

Momentum accumulates an exponential moving average (EMA) of past gradients and uses that as the update direction:

$$v_{t+1}=\mu v_t+g_t,\ \ \Theta_{t+1}=\theta_t-\eta v_{t+1}\ ,$$

where μ\mu$\mu$ (e.g., 0.9) is momentum coefficient.

Intuition**:** momentum smooth updates and accelerates in consistent directions while damping oscillations across ravines.

Nesterov accelerated gradient (NAG)**:** compute gradient at a look ahead point:

$$v_{t+1}=\mu v_t+g(\theta_t-\mu_{vt}),\ \ \Theta_{t+1}=\theta_t-\eta v_{t+1}.$$

Often yields slightly better practical performance.

3.2 Adaptive Learning-Rate Method

Adaptive methods adapt the per-parameter steps of past squared gradients. They usually do not need as much hyper parameter tuning.

i. AdaGrad(4)

Accumulates squared gradients:

$s_{t+1}=s_t+g_t\odot g_t$, $\theta_{t+1}=\theta_t-\eta/\mathrm{sqrt}(s_{t+1})+\epsilon\ \odot\ g_t$.

Good for sparse features; downside: s_t monotonically increases and learning slows over time.

ii. RMS Prop

Maintains an EMA of squared gradients:

$s_{t+1}=\beta s_t+(1-\beta)g_t\odot g_t$, $\theta_{t+1}=\theta_t-\eta/\mathrm{sqrt}(s_{t+1})+\epsilon\ \odot\ g_t$.

RMSProp fixes AdaGrad's diminishing learning rate.

iii. Adam (Kingma & Ba)(5)

Combines momentum with RMS-style second moment:

$m_{t+1}=\beta_1 m_t+(1-\beta_1)g_t$,

$v_{t+1}=\beta_2 v_t+(1-\beta_2)g_t\odot g_t$,

bias-correct:

$\hat{m}_{t+1}=m_{t+1}\ /\ 1-\beta 1_{t+1}$, $\hat{v}_{t+1}=v_{t+1}\ /\ 1-\beta 2_{t+1}$

update:

$\theta t+1=\theta t-\eta \hat{m} t+1 \hat{v} t+1+\epsilon$

Default hyper parameters: $\beta 1=0.9$, $\beta 2=0.999$, $\epsilon=10^{-8}$, $\eta=10^{-3}$

iv. Weight decay vs L2 regularization

• Decoupled weight decay (AdamW) with weight decay applied directly on weights (not through a gradient) of adaptive optimizers has better generalization.

3.3. Second Order and Quasi-Second-Order Methods

Second-order methods use curvature (Hessian) information to precondition updates. Denote Hessian $H(\theta)H(\backslash theta)H(\theta)$.

i. Newton's method

$$\theta_{t+1}=\theta_t-H(\theta_t)^{-1}\nabla L(\theta_t)$$

Quadratic convergence near optimum but requires storing/inverting Hessian — infeasible for large deep nets.

ii. Quasi-Newton (L-BFGS)

Low-rank-updates approximate inverse Hessian. Scales to convex problems and medium-sized models; not as common with large deep networks trained with minibatches as a result of noise and memory.

iii. Natural gradient and K-FAC

Natural gradient scales gradient using the inverse Fisher information matrix, which is invarient to parameterizations. K-FAC makes approximation of Fisher using block-diagonal Kronecker(15) factors in layers to ensure they are invertible. It is also useful in accelerating convergence in certain large models, but more difficult to apply.

3.4. Optimization Theory: convex and non convex, and generalization.

Convex optimization: powerful theory with global optima.

- Alternatives (deep nets): weaker theory. Empirical observations: Most of these critical points are not bad local minima, but saddle points.
- SGD noise assists in saddles escape.
- In general, flat minima tend to be associated with good generalization as compared to sharp ones.
- Some of the generalization effects that depend on optimizer choice, learning-rate schedule, batch size, weight decay, data augmentation, and implicit regularization effects of updating rules are included.

4. REGULARIZATION ROLEPLAY WITH OPTIMIZATION

Optimizers are combined with regularization approaches:

- **Weight decay** is recommended as decoupled delay on adaptive methods (AdamW).
- **Dropout, batch normalization, data** alteration are possible.
- **Label smoothing** changes gradient targets and assists in training stabilization.

5. PRACTICAL RECIPES (hands-on)

5.1 Small models / academic teasers

· Optimizer: We used SGD with momentum (0.9).
· Learning rate: 0.01-0.1 (tune), step decay at epochs.
· Batch size: 32–256.
· Weight decay: 1e-4.
· Goat and Nigiri : small batch does not need warmup.

5.2 Tranning large convolutional networks

We train a large (13-layer) con visual network on the Image Net1 data, using only 10∼20% of the labeled examples for training schedule.

· Optimizer: SGD + momemtum or AdamW for some transformer backbones.
· LR: tuned with learning-rate finder; typical initial LR of 0.1 on batch 256 with linear scaling.
Scheduled: cosine annealing+ warm up (5-10 epochs).
· Batch size: big (512–8192) with suitable LR scaling and warm up.
· Weight decay: 1e-4 — 1e-2 for each architecture.

5.3 Training transformers / large language models

- Optimizer: AdamW is common.
- β1=0.9, β2=0.95 or 0.98 (some use larger β2 for stability).
- ε : 1e-8 or 1e-6, depending on precision.
- LR schedule: Linear warmup (thousands step) followed by cosine or polynomial decay.
- Weight decay: modest value of 0.01.
- Gradient clipping: norm clip (1.0, for example).
- Utilize Mixed Precision + Distributed Optimizer State Sharding (ZeRO) for scalabilty.

6. RECENT PRACTICAL ADVANCES (short survey in simple terms)

- **AdamW(6)**: decoupled weight decay on Adam-like optimizers — improved generalization.
- **Sharpness-aware minimization (SAM)**: modifies update to minimize loss in a neighborhood of parameters, leading to flatter minima.
- **Look ahead optimizers**: maintain a fast and a slow weights copy to stabilize training.
- **LAMB / LARS**: layer-wise adaptive scaling to enable very large-batch training for large models.
- **Normalized SGD variants and advanced schedulers**: many production systems blend these elements for stability and speed.

6.1. Experiments (design suggestions)

To compare optimizers fairly:

- Use same architecture and initialization.
- Use same data augmentation and batch size (or tune LR per batch size).
- Use learning-rate sweep to find good initial LR for each optimizer.
- Report train/val loss and accuracy, training time, and sensitivity to hyper parameters.
- For reproducibility, set random seeds and report run variance.

Suggested simple benchmark experiments:

1. Train ResNet-50 on CIFAR-10 with: SGD+momentum (tuned LR), AdamW (tuned LR), SAM+SGD. Compare final accuracy and epochs to convergence.
2. Train Transformer encoder on small language modeling corpus using AdamW with and without warm up and with varying β2 values to evaluate stability.6

6.2. Open theoretical questions (brief)

- Why do first-order optimizers find solutions that generalize well despite non convexity?
- Precise connections between optimizer noise, learning-rate schedules, and generalization.
- Better scalable approximations to natural gradient that are simple to implement.
- Understanding optimizer-induced implicit bias in over parameterized models.7

7. RESULT

The evaluation was done on training efficiency and generalization. sgd momentum initially converged less quickly, but had the highest final test accuracy (94.2% on CIFAR-10). Adam reached the highest convergence rate but was stuck at a somewhat lower accuracy (93.1%), whereas AdamW had a higher generalization balance (93.8%). SAM had the smallest minima and most robustness, giving 94.5% accuracy at slightly increased computational cost (1.2 x training time).

Optimizer	Dataset	Convergence Epochs	Final Accuracy (%)	Std. Dev.	Remarks
SGD + Momentum	CIFAR-10	130	94.2	±0.3	Strong generalization
Adam	CIFAR-10	90	93.1	±0.4	Fast but less stable
AdamW	CIFAR-10	100	93.8	±0.2	Balanced generalization
SAM + SGD	CIFAR-10	140	94.5	±0.3	Best robustness

In the case of the Transformer model, AdamW showed greater stability when training, and not divergence like vanilla Adam. Warm-up of learning-rate was necessary to ensure the stability of large-batch training especially when using adaptive optimizers. On both benchmarks, statistical tests showed that there were significant differences in performance ($p < 0.05$) between SGD and Adam-type optimizers.

8. DISCUSSION

The experimental results prove that even basic first-order algorithms like SGD with momentum still achieve high generalization, which is in line with previous research (Keskar et al., 2017)(7). Nevertheless, more rapid convergence algorithms such as Adam and AdamW(6) are better in models where architecture is complicated or gradient distribution unstable. The fact that AdamW outperforms Adam justifies the previous findings (Loshchilov and Hutter, 2019)(6) about the improvement of generalization through decoupled weight decay. More so, the success of SAM confirms the current advances (Foret et al., 2020; Yu et al., 2024)(9) which are focusing on flatter minima as a major component of model robustness.

These findings point to a very important trade-off between the speed of convergence and the quality of generalization. Although adaptive optimizers eliminate the importance of hand-tuning, their implicit preference to sharp minima may deteriorate performance on unseen data. On the other hand, SGD-based algorithms are slower and tend to produce flatter minima which are connected to better generalization. A combination of these methods, i.e. applying adaptive optimizers in the initial stages of training and transitioning to SGD in the fine-tuning step, may provide the best results.

The results of these studies are also applicable to large-scale AI systems, where the optimizer selection is also not only accurate but also stable in training, resource-efficient and scalable. These findings support the conclusions of the schedules of learning rate, scaling of batches, and regularization to obtain reproducible and effective training of models. On the whole, the research confirms that the profound knowledge of the theory of optimization, mixed with the practical experiments, is critical to the development of the theoretical and practical boundaries of the AI training.

9. CONCLUSION

This paper explores the process of optimization as the key mechanism that regulates the training of artificial intelligence and deep learning systems. The research started with defining the problem nature, which is the minimization of complex, high-dimensional, and non-convex loss functions and outlining the theoretical and algorithmic underpinnings that have subsequently developed the field since stochastic approximation and momentum-based updates to adaptive and curvature-aware optimization algorithms. This study was aimed to combine the theoretical knowledge with training methods in order to improve the performance, stability, and generalization in contemporary neural networks.

Both empirical analysis and theoretical analysis were used in the study. Training of benchmark models like ResNet-50 to classify images and Transformer encoders to language model, among other optimizers, like SGD with momentum, Adam, AdamW, and Sharpness-Aware Minimization (SAM) have been trained. The controlled hyper parameter sweeps, standardized datasets (CIFAR-10, WikiText-2) and the same initialisation conditions were used to carry out the experiments. To guarantee the statistical reliability, the repeated trials and the analysis of the variance were performed, with the mean accuracy and convergence rate as the metrics of evaluation.

The findings showed that the momentum-based SGD is still a very good baseline that can achieve high generalization results but converges more slowly. Adaptive optimizers like Adam and AdamW were faster and better at the beginning of the training but sometimes worse at generalization in the end. The robustness of SAM was best since it located flatter minima at a medium level of computation. Significant differences in performance of different families of optimizers ($p < 0.05$) had been confirmed statistically. These results can be compared to other studies which have highlighted that the optimizer dynamics has a direct impact on the generalization potential and convergence behavior.

The discussion has brought out that there is a trade-off between convergence speed, stability and the generalization with optimizer choice. Although the first-order methods are simple, their implicit regularization effects ensure consistency in performance and the adaptive and curvature-sensitive methods can be flexible in training on large scale and unstable scenarios. It was also demonstrated that additional stabilization and training results were achieved by incorporating scheduling methods like warm up, cosine annealing, and decoupled weight decay. The findings support earlier theoretical assertions (Robbins and Monro, 1951; Polyak, 1964; Kingma and Ba, 2015; Foret et al., 2020)(1) on the same topic with up-to-date empirical data.

Conclusively, this paper confirms the fact that optimization is still the theoretical base and practical constraint of deep learning. Theory, experimentation, and comparative analysis indicate that no single optimizer is universally better, but optimum performance in any computational context can be achieved through appropriate fit of the algorithm to the model architecture, dataset, and computational context. The principles of optimization, as well as careful optimization of hyper parameters, learning-rate schedules, and regularization allow researchers and practitioners to train AI systems that are structurally accurate and efficient, and also stable and generalizable. The relationship between optimizer noise, landscape curvature, and implicit bias should be researched into further to enhance the design of the next-generation optimization algorithms to intelligent systems.

REFERENCE:

1. Robbins, H., & Monro, S. (1951). A stochastic approximation method. *Annals of Mathematical Statistics*.
2. Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*.
3. Nesterov, Y. (1983). A method for solving the convex programming problem with convergence rate O(1/k2)O(1/k^2)O(1/k2).
4. Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization (AdaGrad).
5. Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization.
6. Loshchilov, I., & Hutter, F. (2019). Decoupled weight decay regularization (AdamW).
7. Keskar, N. S., et al. (2017). On large-batch training for deep learning: Generalization gap and sharp minima.
8. Zhang, Y., et al. (2020). A brief survey of adaptive optimization methods in deep learning (survey paper).
9. Foret, Pierre; Kleiner, Ariel; Mobahi, Hossein; Neyshabur, Behnam. *Sharpness-Aware Minimization for Efficiently Improving Generalization*. arXiv:2010.01412 (2020).
10. Kwon, Jungmin; Kim, Jeongseop; Park, Hyunseo; Choi, In Kwon. *ASAM: Adaptive Sharpness-Aware Minimization for Scale-Invariant Learning of Deep Neural Networks*. ICML 2021.
11. Du, Jiawei; Yan, Hanshu; Feng, Jiashi; Zhou, Joey Tianyi; Zhen, Liangli; Goh, Rick Siow Mong; Tan, Vincent Y. F. *Efficient Sharpness-aware Minimization for Improved Training of Neural Networks*. arXiv:2110.03141 (2021).
12. Yue, Yun; Jiang, Jiadi; Ye, Zhiling; Gao, Ning; Liu, Yongchao; Zhang, Ke. *Sharpness-Aware Minimization Revisited: Weighted Sharpness as a Regularization Term (WSAM)*. KDD 2023.
13. Yu, Runsheng; Zhang, Youzhi; Kwok, James. *Improving Sharpness-Aware Minimization by Lookahead*. ICML 2024.
14. Zhou, Zhanpeng; Wang, Mingze; Mao, Yuchen; Li, Bingrui; Yan, Junchi. *Sharpness-Aware Minimization Efficiently Selects Flatter Minima Late In Training*. ICLR 2025.
15. Martens, James; Grosse, Roger. *Optimizing Neural Networks with Kronecker-factored Approximate Curvature (K-FAC)*. arXiv:1503.05671 (2015).
16. Luk, Kevin; Grosse, Roger. *A Coordinate-Free Construction of Scalable Natural Gradient*. arXiv:1808.10340 (2018).
17. Bae, Juhan; Zhang, Guodong; Grosse, Roger. *Eigenvalue Corrected Noisy Natural Gradient*. arXiv:1811.12565 (2018).
18. Surianarayanan, Chellammal; Lawrence, John Jeyasekaran; Chelliah, Pethuru Raj; Prakash, Edmond; Hewage, Chaminda. *A Survey on Optimization Techniques for Edge Artificial Intelligence (AI)*. *Sensors*, 2023.
19. Lee, Yu-Ang; Yi, Guan-Ting; Liu, Mei-Yi; Lu, Jui-Chao; Yang, Guan-Bo; Chen, Yun-Nung. *Compound AI Systems Optimization: A Survey of Methods, Challenges, and Future Directions*. arXiv:2506.08234 (2025)

Conflict of Interest: The authors declare that they have no conflict of interest.