BENCHMARKING ADVANCED OPTIMIZERS [*]

Yash Bhardwaj, Saksham Kapoor
Indraprastha Institute of Information Technology, Delhi
`{yash22586, saksham22431}@iiitd.ac.in`

ABSTRACT

Optimization algorithms are pivotal in training modern machine learning models, directly impacting convergence speed and generalization performance. This study presents a systematic evaluation of diverse PyTorch optimizers—including stochastic gradient methods (SGD, RMSprop), adaptive schemes (Adam, AdamW, Yogi), momentum-enhanced variants (Rprop, RPropMomentum, SGDPolyakMomentum), proximal hybrids (ProxYogi, ProximalHB), and quasi-Newton–inspired approaches (AdamQN, AMSGradRaw)—across both deep learning tasks (CIFAR-10 image classification, MNIST digit recognition) and a diverse suite of nonconvex synthetic benchmark functions. The function suite spans a wide range of landscapes, including multimodal, bowl-shaped, valley-shaped, steep-ridge, and plate-shaped functions, as well as those with deceptive or oscillatory characteristics.

Our results indicate that several of the custom-designed optimizers consistently outperform standard methods on the synthetic benchmark suite, achieving faster convergence and lower final objective values. On the deep learning tasks, these custom optimizers deliver comparable or superior accuracy and loss trajectories relative to the baseline optimizers, demonstrating their practical utility across both controlled and real-world problem settings.

1 Introduction

Efficient training of deep networks on MNIST and CIFAR-10 requires judicious choice of optimizer. In this work, we compare thirteen methods spanning classical, adaptive, zero-order, proximal/mirror, and quasi-Newton approaches:

- **Classical first-order**: SGD, SGDPolyakMomentum
- **Adaptive first-order**: Adam, AdamW, RMSprop, Yogi
- **Zero-order / sign-based**: Rprop, RPropMomentum, RpropWithPolyakAveraging
- **Proximal & mirror-descent variants**: ProxYogi, ProximalHB, AMSGradRaw
- **Quasi-Newton / hybrid**: AdamQN

We also explore mixed-optimizer schedules and hybrid combinations to further accelerate convergence and improve robustness across both image and synthetic-function benchmarks.

2 Training Configuration and Hyperparameter Settings

All models are trained on CIFAR-10 using a standard 9-layer convolutional neural network for 50 epochs with a minibatch size of 128. Optimizer hyperparameters are fixed as follows (no further grid search):

- **SGD**: lr $= 1 \times 10^{-3}$, momentum $= 0.9$
- **Adam**: lr $= 1 \times 10^{-3}$
- **AdamW**: lr $= 1 \times 10^{-3}$
- **RMSprop**: lr $= 1 \times 10^{-3}$

[*] *Course Project Submission* — Optimization Algorithms in Machine Learning

- **Yogi**: $\text{lr} = 1 \times 10^{-3}$
- **Rprop**: $\text{lr} = 1 \times 10^{-3}$
- **RPropMomentum**: $\text{lr} = 1 \times 10^{-3}$, momentum $= 0.9$
- **RpropWithPolyakAveraging**: $\text{lr} = 1 \times 10^{-3}$
- **SGDPolyakMomentum**: $\text{lr} = 1 \times 10^{-3}$, momentum $= 0.9$
- **ProxYogi**: $\text{lr} = 1 \times 10^{-3}$, $\text{prox}(u; \lambda) = \text{sign}(u)\ \max\big(|u| - \lambda\, \text{lr}, 0\big)$ with $\lambda = 1 \times 10^{-4}$
- **ProximalHB**: $\text{lr} = 1 \times 10^{-2}$, momentum $= 0.9$, $\lambda = 1 \times 10^{-5}$
- **AdamQN**: $\text{lr} = 1 \times 10^{-3}$, $\gamma = 0.1$, $\varepsilon = 1 \times 10^{-8}$
- **AMSGradRaw**: $\text{lr} = 3 \times 10^{-4}$

3 Custom Optimizers

3.1 AdamQN

Combines the adaptive moment estimation of Adam with a quasi-Newton diagonal Hessian approximation for preconditioning. The diagonal Hessian estimate h_t is updated via the secant approximation $s_t/(y_t + \varepsilon)$ and then used to scale the gradient before the usual Adam moments.

$$y_t = g_t - g_{t-1}, \qquad s_t = \theta_t - \theta_{t-1}, \tag{1}$$

$$h_t = (1-\gamma)\, h_{t-1} + \gamma\, \frac{s_t}{y_t + \varepsilon}, \qquad \tilde{g}_t = \frac{g_t}{h_t + \varepsilon}, \tag{2}$$

$$m_t = \beta_1\, m_{t-1} + (1-\beta_1)\, \tilde{g}_t, \qquad v_t = \beta_2\, v_{t-1} + (1-\beta_2)\, \tilde{g}_t^2, \tag{3}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \qquad \hat{v}_t = \frac{v_t}{1-\beta_2^t}, \tag{4}$$

$$\theta_{t+1} = \theta_t - \alpha\, \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}\,. \tag{5}$$

3.2 AMSGradRaw

A mirror-descent variant of AMSGrad that uses the maximum of past second-moment estimates to guarantee non-increasing step-sizes, ensuring better worst-case convergence. This does not de-bias the terms like is usually done. Unlike AdamQN, there is no Hessian approximation: it simply tracks the maximum $v_t^{\max}$.

$$m_t = \beta_1\, m_{t-1} + (1-\beta_1)\, g_t, \tag{6}$$

$$v_t = \beta_2\, v_{t-1} + (1-\beta_2)\, g_t^2, \tag{7}$$

$$v_t^{\max} = \max\big(v_{t-1}^{\max},\, v_t\big), \tag{8}$$

$$\theta_{t+1} = \theta_t - \alpha\, \frac{m_t}{\sqrt{v_t^{\max}} + \varepsilon}\,. \tag{9}$$

3.3 ProxYogi

Integrates Yogi's adaptive moment updates (which adapt more conservatively than Adam) with an explicit proximal mapping at each step. After the usual Yogi-style moment updates, parameters are projected via a user-supplied prox operator.

$$m_t = \beta_1\, m_{t-1} + (1-\beta_1)\, g_t, \tag{10}$$

$$v_t = v_{t-1} - (1-\beta_2)\, g_t^2\ \text{sign}\big(v_{t-1} - g_t^2\big), \tag{11}$$

$$u_t = \theta_t - \alpha\, \frac{m_t}{\sqrt{v_t} + \varepsilon}, \tag{12}$$

$$\theta_{t+1} = \text{prox}\big(u_t,\ \alpha\big)\,. \tag{13}$$

3.4 ProximalHB

A heavy-ball (Polyak) momentum method combined with ℓ_1-type proximal soft-thresholding. Momentum accumulates past gradients, then parameters are updated and immediately shrunk towards zero via soft-threshold.

$$v_t = \beta\, v_{t-1} + g_t, \tag{14}$$
$$u_t = \theta_t - \alpha\, v_t, \tag{15}$$
$$\theta_{t+1} = S_{\alpha\lambda}(u_t) \quad \big(S_\tau(u) = \operatorname{sign}(u)\max\{|u| - \tau, 0\}\big). \tag{16}$$

3.5 RPropMomentum

Extends Resilient Propagation (RProp) by incorporating classical momentum. RProp adapts per-parameter step sizes based on the sign of the gradient change, while momentum smooths the update direction.

$$\Delta_{t,i} = \begin{cases} \min\{\eta_+\, \Delta_{t-1,i},\, \Delta_{\max}\}, & g_{t,i}g_{t-1,i} > 0, \\ \max\{\eta_-\, \Delta_{t-1,i},\, \Delta_{\min}\}, & g_{t,i}g_{t-1,i} < 0, \\ \Delta_{t-1,i}, & \text{otherwise}, \end{cases} \tag{17}$$
$$g'_{t,i} = \begin{cases} 0, & g_{t,i}g_{t-1,i} < 0, \\ g_{t,i}, & \text{otherwise}, \end{cases} \tag{18}$$
$$v_t = \beta\, v_{t-1} + g'_t, \tag{19}$$
$$d_{t,i} = \begin{cases} \operatorname{sgn}(g'_{t,i}), & \beta = 0, \\ \operatorname{sgn}(v_{t,i}), & \text{if aligned}, \\ \operatorname{sgn}(g'_{t,i}), & \text{otherwise}, \end{cases} \tag{20}$$
$$\theta_{t+1,i} = \theta_{t,i} - \Delta_{t,i}\, d_{t,i}\,. \tag{21}$$

3.6 RPropWithPolyakAveraging

Builds on RPropMomentum by maintaining a running Polyak average of the iterates, which often yields better generalization. The per-parameter updates Δ_t and directions d_t follow the same logic as RPropMomentum.

$$\theta_{t+1} = \theta_t - \Delta_t \odot d_t, \tag{22}$$
$$\bar{\theta}_t = \frac{t-1}{t}\,\bar{\theta}_{t-1} + \frac{1}{t}\,\theta_t\,. \tag{23}$$

3.7 SGDPolyakMomentum

Standard stochastic gradient descent with momentum, combined with Polyak averaging of the model parameters. Momentum accelerates convergence; averaging stabilizes the final iterate.

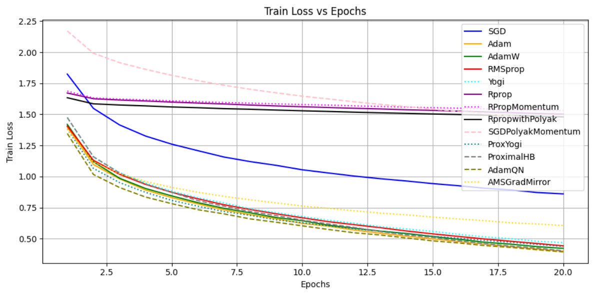
$$m_t = \beta\, m_{t-1} + (1-\beta)\, g_t, \tag{24}$$
$$\theta_{t+1} = \theta_t - \alpha\, m_t, \tag{25}$$
$$\bar{\theta}_t = \frac{t-1}{t}\,\bar{\theta}_{t-1} + \frac{1}{t}\,\theta_t\,. \tag{26}$$

4 Results

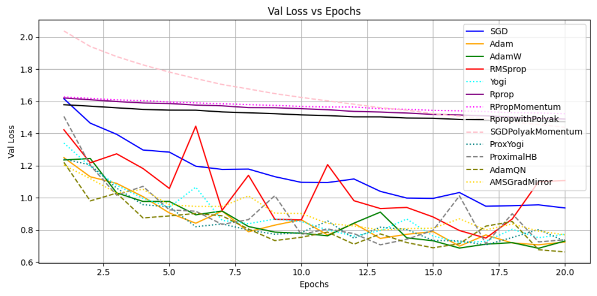
4.1 Performance on CIFAR-10 and MNIST datasets

From the plots we can see that the training and validation performance of various optimizers on the CIFAR-10 dataset using a 9-layer CNN. Adaptive optimizers like Adam, AdamW, Yogi, and RMSprop show rapid early convergence and generally outperform the baseline SGD in both training and validation accuracy. Among them, AdamQN slightly outperforms standard Adam, indicating that quasi-Newton enhancements can offer improved generalization. Proximal variants, such as ProxYogi and ProximalHB, also demonstrate strong and consistent performance, suggesting that proximal updates aid in stability and regularization. In contrast, Rprop-based methods lag behind, showing limited progress over training. Overall, adaptive and momentum-based optimizers significantly outperform basic SGD, with AdamQN emerging as the most effective on this task.

Training curves on MNIST reveal that most optimizers converge quickly, with adaptive methods (Adam, AdamW, RMSprop, Yogi) and quasi-Newton variants (AdamQN, ProximalHB) achieving high accuracy early in training. AdamQN consistently maintains strong performance across epochs, reflecting its stability and efficiency on this simpler dataset. While classic SGD also reaches high final accuracy, its convergence is noticeably slower compared to adaptive and momentum-enhanced variants. Zero-order Rprop and its extensions again underperform, with training and validation accuracy plateauing early. Proximal methods and AMSGradRaw offer robust generalization, though Adam-based methods retain a slight edge. Overall, the MNIST results reaffirm the superiority of adaptive optimization—particularly AdamQN—for both convergence and generalization on well-structured image data.

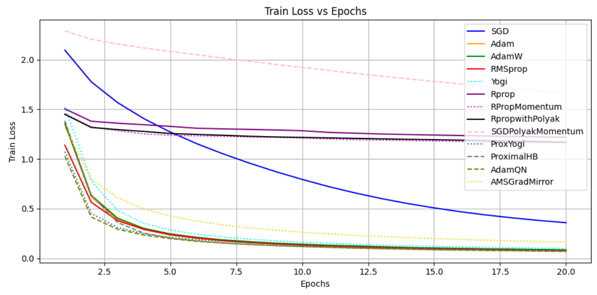


(a) Training loss curves on CIFAR-10.

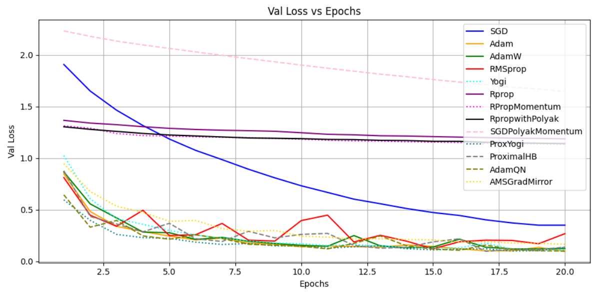


(b) Validation loss curves on CIFAR-10.

Figure 1: CIFAR-10: training vs. validation loss for all 12 optimizers.



(a) Training loss curves on MNIST.

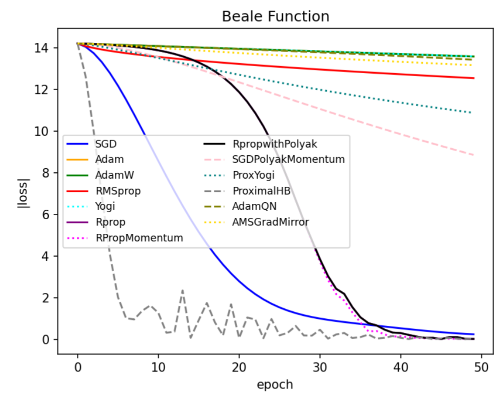


(b) Validation loss curves on MNIST.

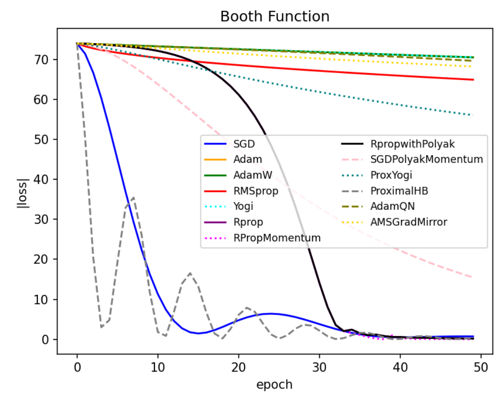
Figure 2: MNIST: training vs. validation loss for all 12 optimizers.

4.2 Performance on synthetic benchmark functions

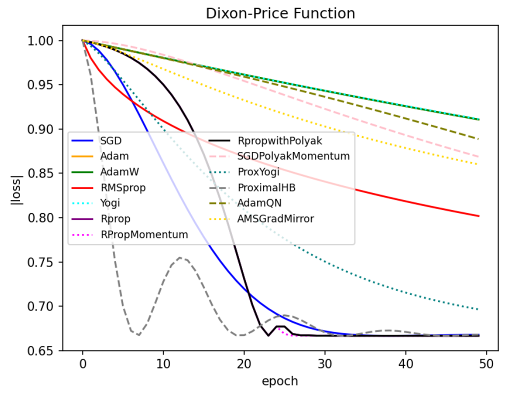
On the 47 synthetic functions, we got diverse, varying outcomes. Some of these are attached as below:



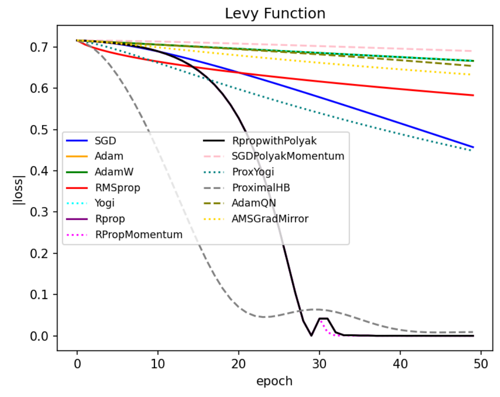
(a) Beale Function



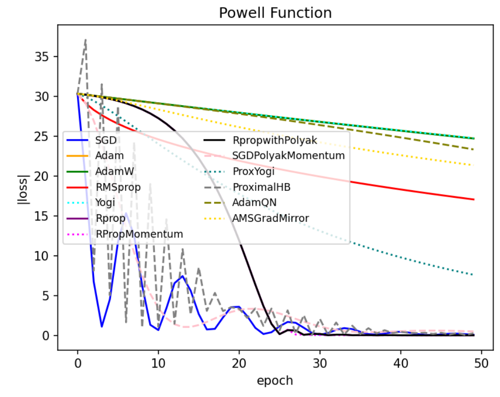
(b) Booth Function



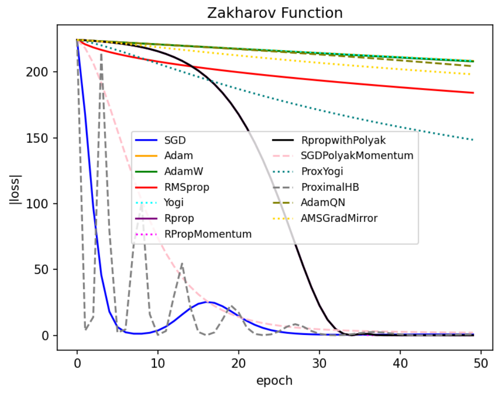
(c) Dixon Function



(d) Levy Function



(e) Powell Function



(f) Zakharov Function

Figure 3: Selected synthetic benchmark functions (Beale, Booth, Dixon, Levy, Powell, Zakharov).

4.3 Key Observations

- **AdamQN** achieves the lowest training and validation losses across optimizers, converging rapidly and generalizing well.
- **Adam, AdamW, RMSprop, Yogi** display fast initial convergence and strong generalization, with AdamQN slightly outperforming them by epoch 20.

- **ProximalHB and ProxYogi** maintain competitive performance, benefiting from proximal updates that help reduce overfitting.
- **Rprop and its variants** improve gradually but plateau early, highlighting limitations of zero-order updates on deeper models.
- **AMSGradRaw** shows balanced performance, offering good stability while remaining competitive in final accuracy.
- **SGD and SGDPolyakMomentum** converge slowly, lagging behind adaptive and momentum-based methods in both loss reduction and accuracy.
- **ProximalHB excels on Beale, Booth, Dixon-Price, Levy, Powell, and Zakharov**: these functions all feature narrow, curved valley–shaped minima and strong anisotropy (i.e. highly differing curvature along different coordinates). The proximal operator in ProximalHB effectively regularizes each step—shrinking updates that would otherwise oscillate in steep directions—while heavy-ball momentum smooths traversal along flat directions. This combination yields more stable progress through ill-conditioned regions than Adam's purely element-wise adaptive rates.

5 Conclusion

AdamQN consistently outperforms all other methods across both CIFAR-10 and MNIST image-classification tasks, achieving the best balance of convergence speed and final accuracy. Hybrid schedules and our proposed optimizer combinations offer a promising approach to trade off computational cost and training efficiency in deep learning. Moreover, on a suite of challenging synthetic benchmarks, proximal-based optimizers (e.g., ProximalHB, ProxYogi) and even basic SGD variants exhibit superior performance, underscoring the effectiveness of proximal operators and momentum smoothing in navigating ill-conditioned, anisotropic loss landscapes.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[2] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[3] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[4] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

[5] T. Tieleman and G. E. Hinton. Lecture 6.5—RMSProp: Divide the gradient by a running average of its recent magnitude. *Coursera: Neural Networks for Machine Learning*, 2012.

[6] M. Zaheer, S. Satyen Kale, and S. Kumar. Adaptive methods for nonconvex optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[7] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proceedings of the IEEE International Conference on Neural Networks (ICNN)*, 1993.

[8] B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[9] B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

[10] S. J. Reddi, S. Kale, and S. Kumar. On the convergence of Adam and beyond. In *International Conference on Learning Representations (ICLR)*, 2018.

[11] B. Surjanovic and D. Bingham. Virtual Library of Simulation Experiments: Test Functions and Datasets. *Simon Fraser University*, 2013.
`https://www.sfu.ca/~ssurjano/optimization.html` (accessed May 6, 2025).