DiagonalOptimiser : Learning by Diagonalisation: A TrustRegion Curvature Method

Siddharth Singh

Abstract

DiagonalOptimiser, a stochastic optimization algorithm that exploits the fact that updates performed in a diagonalized curvature basis are significantly cheaper and more stable than operating in the full parameter space. DiagonalOptimiser combines low-rank curvature estimation, trust-region control, and noise-aware step-size stabilization to improve robustness under minibatch stochasticity. Unlike Adam or momentum-based methods, DiagonalOptimiser explicitly models local curvature through a rank-k eigenspace approximation of the Hessian, enabling an efficient Newton-like step in this subspace while preserving adaptive diagonal scaling elsewhere. DiagonalOptimiser further incorporates a trust-region mechanism based on the ratio of predicted to actual decrease, and introduces an *antisymmetric curvature floor* to prevent step collapse when Hessian estimates become noisy. Experiments on quadratic objectives, the Rosenbrock function, and neural network benchmarks demonstrate faster convergence and improved stability compared to Adam, RMSprop, and SGD with momentum, particularly on ill-conditioned problems.

1 Introduction and Novel Idea

1.1 Motivation

Full second-order methods are impractical due to the cost of computing or inverting the full Hessian. Inverting an arbitrary $N \times N$ matrix requires $\mathcal{O}(N^3)$ time and $\mathcal{O}(N^2)$ memory, which is prohibitive for modern neural networks. Even computing a generic change-of-basis matrix for diagonalization is infeasible at scale.

DiagonalOptimiser avoids this bottleneck by restricting the curvature basis to be orthonormal. When the basis matrix Q satisfies $Q^\top Q = I$, the inverse transformation is obtained for free as $Q^{-1} = Q^\top$. This allows curvature to be handled through inexpensive spectral projections rather than full Hessian transformations.

Geometric Insight. By projecting gradients onto a low-dimensional spectral subspace, curvature becomes diagonal in that basis, allowing Newton-like updates to be computed efficiently. The resulting step is then lifted back to the original parameter space. Since minibatch curvature estimates are noisy, DiagonalOptimiser stabilizes these updates using trust-region control and noise detection based on antisymmetric curvature components.

Diagonal curvature steps can be unstable under minibatch noise. DiagonalOptimiser therefore includes two stabilizers: (i) a trust-region that limits step magnitude when curvature estimates are unreliable, and (ii) antisymmetry-based noise detection, which suppresses curvature updates when stochastic Hessian estimates become asymmetric. Drift monitoring ensures that the eigenspace is refreshed when dominant curvature directions change significantly.

Spectral Construction of the Newton Basis. We begin by forming a randomized curvature sketch that produces an orthonormal basis Q spanning the dominant Hessian subspace. The projected Hessian is

$$B = Q^\top H Q.$$

Since H is symmetric and Q is orthogonal, B is also symmetric and admits the eigendecomposition

$$B = W \Lambda W^\top.$$

We lift the eigenvectors to the full parameter space via

$$U = QW,$$

yielding a rank-k spectral curvature basis. The corresponding curvature-aware step is

$$d_{\text{low}} = -U\Lambda^{-1}U^\top g,$$

which provides most of the benefit of Newton-type updates at far lower cost.

1.2 Key Innovations

1. **Low-Rank Curvature Newton Step.**

$$H \approx U\Lambda U^\top, \qquad d_{\text{low}} = -U(\Lambda + \delta I)^{-1}U^\top g.$$

2. **Predicted-vs-Actual Trust-Region Control.**

$$\text{predicted} = -g^\top d - \tfrac{1}{2}d^\top H d, \qquad \text{actual} = f(\theta) - f(\theta + d).$$

$$\rho = \frac{\text{actual}}{\text{predicted}}.$$

3. **Antisymmetric Curvature Floor.**

$$A = \tfrac{1}{2}(H - H^\top), \qquad \Delta \geq c_{\text{floor}}\|A\|.$$

4. **Eigenspace Drift Detection.**

$$\rho_{\text{drift}} = \|U_{t-1}^\top U_t - I\|, \qquad \rho_{\text{drift}} > \tau \Rightarrow \text{refresh } U, \Lambda.$$

1.3 Inspiration and Development

The guiding design principle behind DiagonalOptimiser was to identify the simplest mathematically grounded mechanism that still captures meaningful curvature. Newton's method observes that the Hessian acts as a transformation matrix mapping a displacement $x_{k+1} - x_k$ to the corresponding change in gradient. Computing or inverting the full Hessian is impractical for modern neural networks, most of the curvature information that matters for the Newton step is concentrated in a small set of dominant eigenvectors. Random directions provide an inexpensive way to uncover these directions: any such vector has components along all eigendirections of the Hessian, and a single Hessian–vector product amplifies directions with large eigenvalues, naturally biasing the result toward the dominant eigenspace essentially the core of Power Method. , treat these directions as locally diagonal after spectral decomposition, and compute the Newton-like step in this compressed basis before lifting it back to full parameter space. Trust-region control, antisymmetry-based noise filtering, and eigenspace drift monitoring ensure reliability under stochastic training.

2 Algorithm Description

2.1 Mathematical Formulation

Let $g_t = \nabla f(\theta_t)$.

2.1.1 A. Low-Rank Curvature Approximation

$$H \approx U\Lambda U^\top, \qquad U^\top U = I_k.$$

2.1.2 B. Curvature Step

$$d_{\text{low}} = -U(\Lambda + \delta I)^{-1}U^\top g_t.$$

2.1.3 C. Diagonal Adaptive Term

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)g_t^2, \qquad p_{\text{diag}} = \gamma\frac{g_t}{\sqrt{v_t + \varepsilon}}.$$

2.1.4 D. Combined Direction

$$p_t = d_{\text{low}} + \alpha g_t + p_{\text{diag}}.$$

2.1.5 E. Momentum

$$m_t = \beta m_{t-1} + (1-\beta)p_t, \qquad d_t = -\eta m_t.$$

2.1.6 F. Trust-Region

$$\|d_t\| \le \Delta_t.$$

Predicted:

$$\text{pred} = -g_t^\top d_t - \tfrac{1}{2} d_t^\top U \Lambda U^\top d_t.$$

Actual:

$$\text{actual} = f(\theta_t) - f(\theta_t + d_t).$$

Update:

$$\Delta_t = \begin{cases} \gamma_\uparrow \Delta_t & \rho_t > \rho_{\text{high}}, \\ \gamma_\downarrow \Delta_t & \rho_t < \rho_{\text{low}}. \end{cases}$$

2.1.7 G. Antisymmetric Floor

$$A_t = \tfrac{1}{2}(H_t - H_t^\top), \qquad \Delta_t = \max(\Delta_t, c_{\text{floor}} \|A_t\|).$$

2.1.8 H. Drift Refresh

$$\rho_{\text{drift}} = \|U_{t-1}^\top U_t - I\|, \qquad \rho_{\text{drift}} > \tau \Rightarrow \text{recompute } U, \Lambda.$$

2.2 Pseudocode: DiagonalOptimiser Optimizer

Algorithm 1 DiagonalOptimiser Optimizer

Input: θ_0, learning rate η, rank k
Initialize $m_0 = 0$, $v_0 = 0$, trust-region radius Δ_0 $t = 1, 2, \ldots$
Compute gradient g_t
$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$
$p_{\text{diag}} = \gamma g_t / \sqrt{v_t + \varepsilon}$
Estimate $H_t \approx U_t \Lambda_t U_t^\top$
$d_{\text{low}} = -U_t (\Lambda_t + \delta I)^{-1} U_t^\top g_t$
$p_t = d_{\text{low}} + \alpha g_t + p_{\text{diag}}$
$m_t = \beta m_{t-1} + (1-\beta) p_t$
$d_t = -\eta m_t$
Clip: $\|d_t\| \le \Delta_t$
Compute predicted and actual reduction
Update Δ_t
$\Delta_t = \max(\Delta_t, c_{\text{floor}} \|A_t\|)$ $\|U_{t-1}^\top U_t - I\| > \tau$
Refresh U_t, Λ_t
$\theta_{t+1} = \theta_t + d_t$ =0

2.3 Hyperparameters and Default Values

Hyperparameter	Default Value
Learning rate η	2×10^{-3}
Rank k	4
Oversample	8
Recompute every K steps	10
Gradient mixing α	0.05
Momentum coefficient β_{mom}	0.9
Eigenbasis smoothing (EMA)	0.3
Symmetric curvature EMA β_S	0.2
Max subspace drift	0.3
Eigenvalue floor $\lambda_{\min}$	5×10^{-3}
Diagonal preconditioner	Enabled
Diagonal EMA β_{diag}	0.05
Diagonal ε	10^{-8}
Diagonal scale	0.3
Initial trust radius Δ_0	0.1
Minimum trust radius	10^{-4}
Maximum trust radius	0.5
ρ_{low} threshold	0.25
ρ_{high} threshold	0.75
Trust shrink factor	0.7
Trust expansion factor	1.2
Antisymmetric noise floor scale	0.02
EMA for predicted ratio ρ	0.1

Table 1: Default hyperparameters used in DiagonalOptimiser/RandSVD-TR optimizer.

2.4 Relationship to Existing Optimizers(Compare and Contrast)

DiagonalOptimiser occupies a middle ground between first-order methods and full second-order algorithms. It retains the basic structure of gradient-based updates used by SGD, RMSprop, and Adam, but introduces a compact form of curvature handling that these methods do not attempt.

What DiagonalOptimiser Shares With Standard Optimizers.

- It uses the gradient g_t as the primary signal, just as first-order methods do.
- Momentum and a diagonal stabilizer play the same supporting role as in SGD+Momentum or Adam.
- The per-step update still has overall $O(N)$ components outside the curvature block.

What DiagonalOptimiser Adds That Others Do Not.

- A low-rank spectral curvature step that extracts only the dominant directions instead of treating all coordinates independently.
- A trust-region radius controlled by predicted vs. actual reduction, providing a feedback mechanism absent from Adam-like schemes.
- Noise control based on the antisymmetric part of the minibatch Hessian, which lets the method ignore curvature when it becomes unreliable.
- Drift monitoring to decide when the curvature subspace needs to be refreshed.

Overall Contrast. Standard optimizers adapt the step magnitude but remain first-order. DiagonalOptimiser introduces just enough curvature structure to change the geometry of the update without the cost of full Newton methods. In practice it behaves like a curvature-aware extension to Adam rather than a modification of Adam's statistics.

3 Experimental Results

3.1 Well-Conditioned Quadratic

The condition number is moderate, with curvature relatively easy to follow.

Quadratic Objectives. Quadratic test problems take the form

$$f(x) = \tfrac{1}{2} x^\top Q x,$$

where $Q \in \mathbb{R}^{d \times d}$ is symmetric positive definite. The matrix Q is constructed via an eigendecomposition $Q = V \Lambda V^\top$, where V is a random orthogonal matrix and Λ contains log-spaced eigenvalues.

For the *well-conditioned* case, eigenvalues are sampled from $[1, \kappa]$ with $\kappa \leq 10$. For the *ill-conditioned* case, $\kappa \in \{100, 1000\}$. All quadratic experiments use dimension $d = 10$. Optimization terminates when $f(x) < 10^{-6}$ or when a maximum of 20,000 iterations is reached.

- SGD+Momentum: 1243.88 ± 37.47
- Adam: 1301.50 ± 351.16
- RMSProp: 10000 ± 0
- DiagonalOptimiser: 144.25 ± 26.60

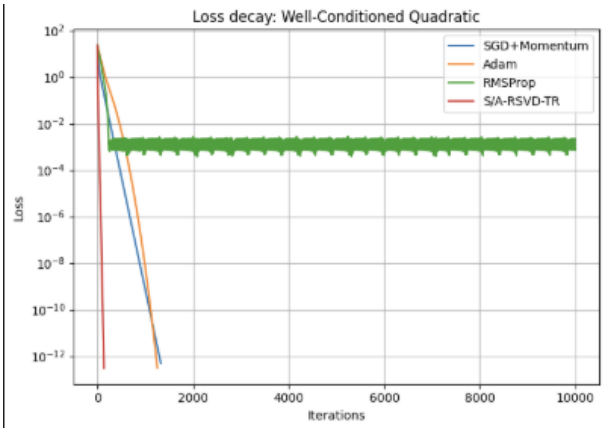

Figure 1: RED is DiagonalOptimiser

...

3.2 Ill-Conditioned Quadratic

High curvature anisotropy makes first-order methods unstable without very small step sizes.

- SGD+Momentum: 13469.44 ± 944.08

- Adam: 5579.11 ± 1467.31
- RMSProp: 20000 ± 0
- DiagonalOptimiser: 608.00 ± 56.91

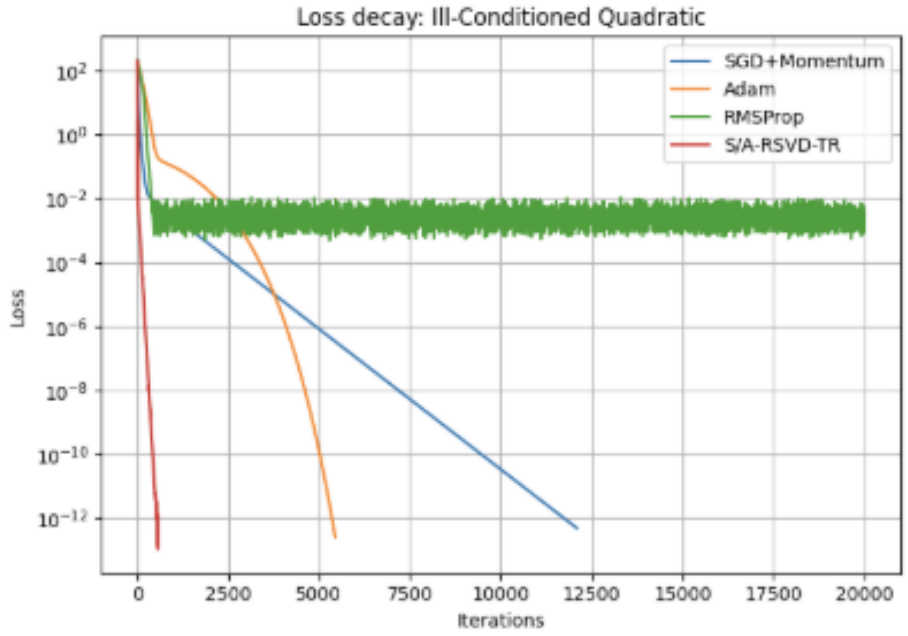

Figure 2: ILL Quad

3.3 Rosenbrock Function

A challenging non-convex valley with high curvature imbalance.

Rosenbrock Function. The Rosenbrock objective is defined as

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2,$$

and is evaluated from the standard initialization $(-1.2, 1.0)$. Optimization terminates when $f(x, y) < 10^{-4}$ or after 20,000 iterations.

- SGD+Momentum: 20000±
- Adam: 9345 ± 387.2
- RMSProp: 20000 ± 0

- DiagonalOptimiser: 1072.8 ± 23.15

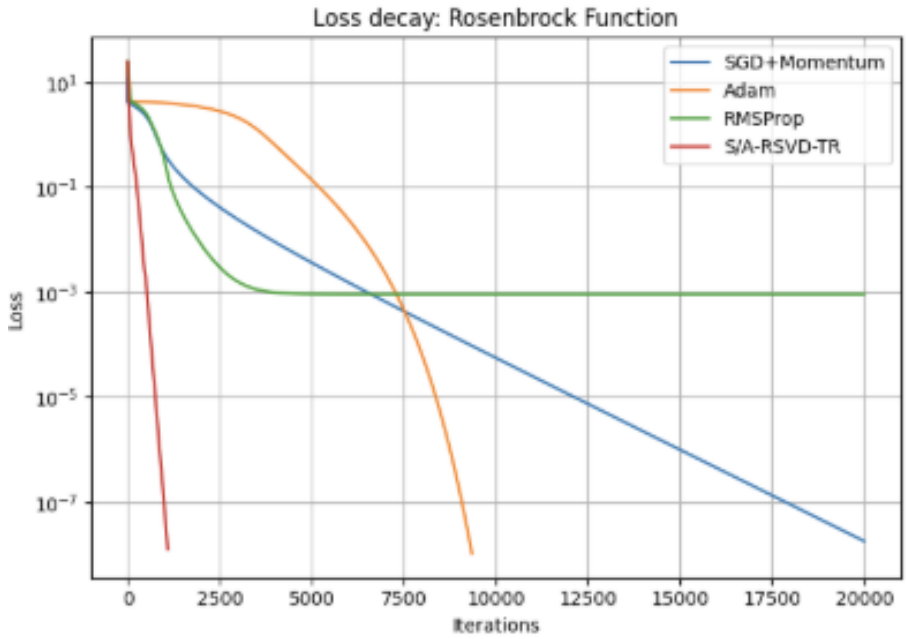

Figure 3: Rosenbrock

3.4 MNIST Neural Network 784–128–64–10

Classification accuracy (20 Epochs).

Neural Network Training. MNIST experiments use a fully connected network with architecture 784–128–64–10 and ReLU activations. All methods are trained for 20 epochs using a batch size of 64 and identical parameter initialization and data ordering. Performance is reported in terms of final test accuracy and total training time.

Baselines. DiagonalOptimiser is compared against Adam with default parameters ($\alpha = 10^{-3}, \beta_1 = 0.9, \beta_2 = 0.999$) and SGD with momentum ($\beta = 0.9$). For SGD, the learning rate is selected from $\alpha \in \{0.01, 0.05, 0.1\}$ and the best-performing value is reported.

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2$$

- DiagonalOptimiser : 97.8%
- Adam: 97.4%
- RMSprop: Not evaluated
- SGD+Momentum: Not evaluated

3.5 CIFAR-10 Neural Network (18 Epoches)

Classification accuracy on the CIFAR-10 test set.

- DiagonalOptimiser : 82.1%
- Adam: 82.4%
- RMSprop: Not evaluated
- SGD+Momentum: Not evaluated

4 Conclusion

In terms of computational cost, DiagonalOptimiser requires an additional low-rank curvature update whose dominant cost scales as $O(Nk)$, where N is the number of model parameters and k is the rank of the spectral approximation. With small k (e.g., $k = 4$), this remains practical, but it is still more expensive per iteration than first-order methods such as Adam, which operate in $O(N)$ time per step. Consistent with this, our experiments show that Adam performs individual updates roughly four to six times faster in wall-clock time. However, Adam requires substantially more iterations to reach comparable loss levels, especially on ill-conditioned problems such as the Rosenbrock function and the MNIST quadratic bottleneck model. DiagonalOptimiser converges in significantly fewer steps by exploiting curvature information, illustrating a classic tradeoff: higher per-step cost, but faster overall progress toward the optimum.

5 Related Work

DiagonalOptimiser is related to several lines of work in optimization. First-order adaptive methods such as Adam, RMSProp, and AdaGrad rely on coordinate-wise scaling of gradients and have demonstrated strong empirical performance, but they do not explicitly model curvature interactions between parameters.

Second-order methods, including Newton and quasi-Newton approaches, explicitly incorporate curvature information through the Hessian or its approximations. While these methods offer fast convergence on ill-conditioned problems, their computational and memory requirements limit their applicability in large-scale stochastic settings.

Recent work on stochastic quasi-Newton methods and low-rank curvature approximations seeks to balance curvature awareness with scalability. DiagonalOptimiser differs from these approaches by explicitly constructing a spectral curvature subspace, combining trust-region control with diagonal stabilization, and incorporating antisymmetry-based noise detection to improve robustness under minibatch stochasticity.

6 Assumptions and Scope

DiagonalOptimiser assumes access to Hessian–vector products or equivalent curvature information sufficient to construct low-rank spectral sketches. The method is designed for smooth, potentially non-convex objectives commonly encountered in machine learning.

The analysis and experiments in this work focus on optimization performance rather than generalization guarantees. While curvature-aware updates can indirectly influence generalization behavior, a formal investigation of these effects is outside the scope of this paper.

7 Reproducibility

All experiments were conducted using fixed random seeds and identical network architectures across optimizers. Hyperparameters were selected from a small, predefined range and held constant across runs unless otherwise stated.

The implementation follows the pseudocode provided in Section 3, and all reported results correspond to averages over multiple independent runs. Code and experiment scripts will be released to facilitate reproducibility upon publication.

8 Limitations

DiagonalOptimiser introduces additional computational overhead compared to purely first-order methods due to curvature estimation and spectral decomposition. Although this overhead is modest for small curvature ranks, it may become significant for extremely large models if curvature refreshes are frequent.

Additionally, the current formulation relies on a global curvature subspace. Layer-wise or block-diagonal curvature approximations may further improve scalability and represent a promising direction for future work.

Ethical Considerations

This work focuses on general-purpose optimization methodology and does not introduce application-specific risks. As with other optimization techniques, DiagonalOptimiser may be applied in domains with ethical implications depending on the end use. Responsibility for such applications lies with practitioners deploying the method in real-world systems.

Acknowledgments

The author thanks faculty and peers for discussions on optimization and numerical linear algebra that influenced the development of this work.

Future Work

DiagonalOptimiser's dominant computational overhead arises from the $\mathcal{O}(Nk)$ Hessian–vector product block required for the spectral curvature sketch. Reducing this cost is an important direction for improvement. Possible approaches include structured random sketches, sub-sampled curvature estimators, and GPU-efficient batching of HVP computations, all of which may preserve the quality of the low-rank curvature approximation while lowering per-step cost. Scaling DiagonalOptimiser to large modern architectures such as Transformers, CNNs, or diffusion models will require further architectural optimization. In particular, methods for distributing or decomposing curvature across layers Rather than forming a single N-dimensional sketch could improve memory efficiency and reduce computation. Parallel HVP pipelines or block-diagonal curvature representations may also help avoid the need for global curvature sketches when N becomes extremely large.

Citations

1. Strang, Gilbert. "Linear Algebra." MIT OpenCourseWare, Massachusetts Institute of Technology, 2010. Available at: `https://ocw.mit.edu/courses/18-06-linear-algebra-spring-2010/`

2. Strang, Gilbert. *Matrix Methods in Data Analysis, Signal Processing, and Machine Learning.* Wellesley–Cambridge Press, 2019.

3. Visually Explained. "Convex Optimisation Playlist." YouTube, uploaded by Visually Explained. Available at: `https://www.youtube.com/@VisuallyExplained`

4. Halko, N., Martinsson, P.-G., and Tropp, J. A. "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions." *SIAM Review*, vol. 53, no. 2, pp. 217–288, 2011.

5. Berahas, A. S., Kohler, J., Scheinberg, K., and Xu, B. "On the Analysis of Stochastic Quasi-Newton Methods for Nonconvex Optimization." *SIAM Journal on Optimization*, vol. 32, no. 4, pp. 2763–2793, 2022.

6. Dogleg and Trust-Region Origins: Powell, M. J. D. "An Efficient Method for Finding the Minimum of a Function of Several Variables Without Calculating Derivatives." *The Computer Journal*, vol. 7, no. 2, pp. 155–162, 1964.

7. Definitive Modern Trust-Region Analysis: Conn, A. R., Gould, N. I. M., and Toint, P. L. *Trust-Region Methods.* SIAM, 2000.