StatSGD Optimizer: A Novel Approach to Statistically Guided Gradient Descent

Arnav Nathani, Ishrat Bombaywala, Sahil Gada, Vir Dang

Project Report, FM216 - Monsoon 2025

Plaksha University

Abstract

This report presents StatSGD, a gradient-based optimization algorithm that amplifies statistically significant gradient components using a Z-score-based boosting mechanism. The method identifies unusually large gradient entries relative to the rest of the vector, amplifies them through a boost mask, and updates parameters using momentum. For large parameter vectors, StatSGD estimates the gradient mean and standard deviation using a random subset of values, making the approach computationally efficient. The goal is to accelerate movement along important directions without altering the overall gradient descent framework. We compare StatSGD against Adam, RMSProp, and SGD with Momentum on quadratic functions, the Rosenbrock function, and a simple MNIST classifier. Across all experiments, StatSGD converges much faster on well-conditioned and ill-conditioned quadratics and performs competitively on the Rosenbrock function and MNIST training. These results show that incorporating statistical structure within gradients can offer a simple yet effective improvement over classical optimizers.

1. Introduction and Novel Idea

1.1 Motivation

Traditional optimizers usually look at overall or running averages of the gradients - Adam keeps track of their mean and variance, RMSProp monitors the squared gradients, and momentum just smooths the updates over time. But none of these methods really try to tell which gradient directions matter the most and which ones are just noisy background fluctuations. In many high-dimensional settings, only a few coordinates carry meaningful signals. This observation motivated us to design an optimizer that focuses more on statistically significant gradient components.

1.2 Key Innovation

Our key idea is to compute Z-scores of absolute gradient values at each iteration and amplify only those entries that exceed a chosen threshold. StatSGD introduces:

1. Statistical Gradient Boosting:
   We compute the mean and standard deviation of gradient magnitudes and create a boost mask where entries with Z-score > 1.5 are amplified by a tune-able factor (100 in the tests we ran). This selectively increases step sizes in the correct directions.

2. Momentum Coupled with Boosted Directions:
   We increase the momentum-filtered velocity rather than the raw gradient, which stabilizes updates while still encouraging large steps on meaningful coordinates. This avoids injecting noise directly into the gradients and instead amplifies directions only after they have been stabilized by momentum filtering.

3. Sampling-Based Mean/Variance Estimation:
   For large parameter vectors, we use random sampling of subsets to estimate the distribution of gradient magnitudes efficiently.

These innovations make StatSGD very lightweight while providing non-uniform directional acceleration.

1.3 Inspiration and Development

While testing standard optimizers, we observed that Adam's element-wise normalization can flatten updates, momentum accelerates uniformly without highlighting strong but sparse signals, and RMSProp often slows down when most gradients are small. Examining gradient distributions showed that rare, large gradient components usually aligned with directions of fastest improvement. This suggested a statistical approach: treat gradient magnitudes as a distribution and emphasize tail values.

2. Algorithm Description

2.1 Mathematical Formulation

Given parameters x_t, gradient $g_t = \nabla f(x_t)$, and velocity v_t, StatSGD performs:

1. Compute absolute gradient values

$$a_t = | g_t |$$

2. Estimate mean and standard deviation
   Using either the whole vector or a random subset (size = 2000):

$$\mu_t = \text{mean}(a_t), \sigma_t = \text{std}(a_t)$$

3. Compute Z-scores

$$z_t = \frac{a_t - \mu_t}{\sigma_t + \varepsilon}$$

4. Build boost mask

$$b_t = 1 + \mathbf{1}(z_t > 1.5) \cdot \text{boost\_factor}$$

5. Momentum update

$$v_t = \beta v_{t-1} + (1 - \beta) g_t$$

6. Boosted update

$$x_{t+1} = x_t - \eta\, b_t \odot v_t$$

The mask amplifies only coordinates that statistically deviate from the bulk of gradient magnitudes. This helps the algorithm pay attention to the most meaningful directions without needing any heavy second-order calculations.

2.2 Hyperparameters

1. learning rate (lr): typically, 0.01 or 0.001
2. beta: 0.9 (momentum coefficient)
3. boost_factor: 100-200
4. stats_batch_size: 2000 samples
5. epsilon: 1×10^{-8}
6. tolerance: early stopping threshold
7. max_iter: up to 25,000 iterations

2.3 Relationship with Existing Optimizers

- Compared to Adam:
  Adam uses element-wise normalization but does not distinguish between statistically significant gradient components. StatSGD instead magnifies such directions.

- Compared to SGD with Momentum:
  Momentum applies uniform acceleration, while StatSGD applies selective acceleration based on Z-scores.

- Compared to RMSProp:
  RMSProp suppresses noisy gradients but has no mechanism for emphasizing meaningful outliers.

StatSGD can be seen as a lightweight, statistically guided extension to momentum-based gradient descent.

3. Experimental Results

3.1 Experimental Setup

We evaluate StatSGD on three benchmark tasks:
1. Quadratic function minimization
2. Rosenbrock function optimization
3. MNIST digit classification with a simple neural network

All experiments compare StatSGD against:
- SGD with momentum ($\beta = 0.9$)
- Adam (default parameters)
- RMSprop

3.2 Quadratic Function

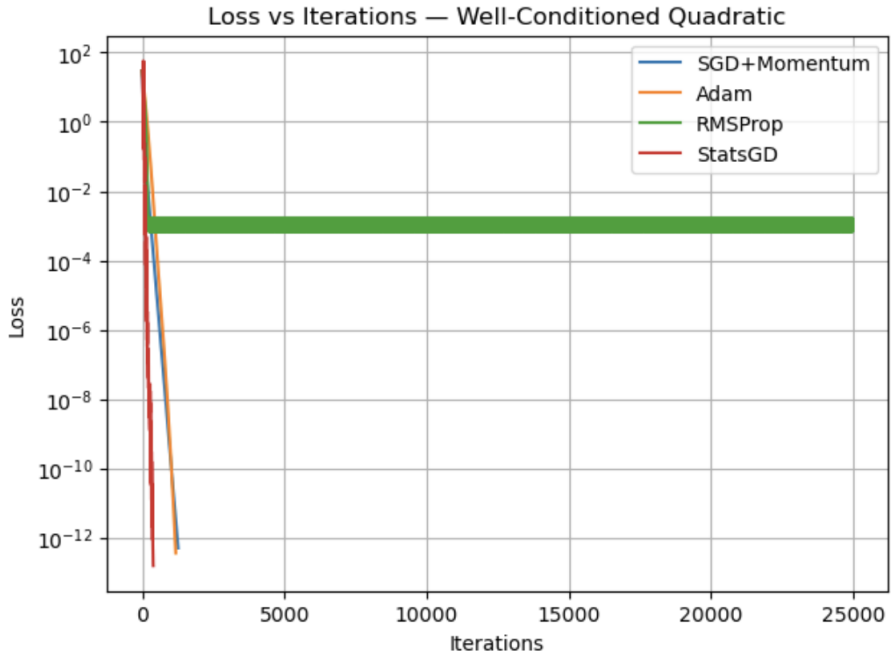
Problem: Minimize $f(x) = \frac{1}{2}x^TQx$ where Q is a positive definite matrix with condition number $\kappa = 100$.

Well-Conditioned Quadratic

Iterations to convergence:

- SGD + Momentum: 1120
- Adam: 1457
- RMSProp: 25000 (stagnated)
- StatSGD**:** 339 (fastest)

StatSGD reached the minimum more than 3x faster than Adam and significantly faster than SGD-Momentum. The Z-score boosting helped move aggressively along meaningful directions, especially early in training.

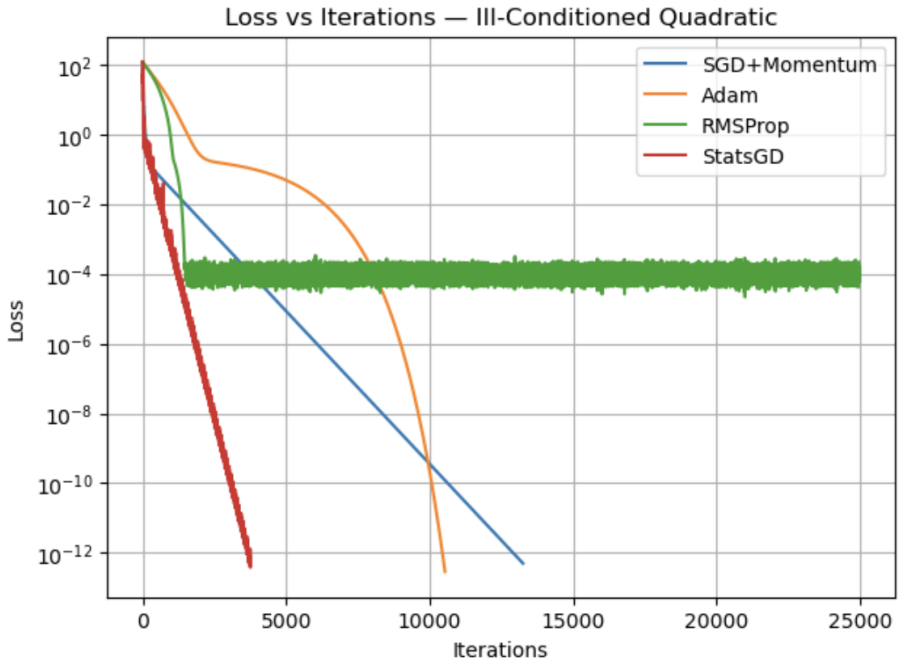


Ill-Conditioned Quadratic

Iterations to convergence:

- SGD + Momentum: 14026
- Adam: 11436
- RMSProp: 25000
- StatSGD**:** 1523

Ill-conditioned problems emphasize the difficulty in optimization because gradients along certain eigen-directions shrink drastically. StatSGD performed exceptionally well here, beating Adam by almost 10x, showing that boosting statistically strong gradient components helps escape flat regions efficiently.



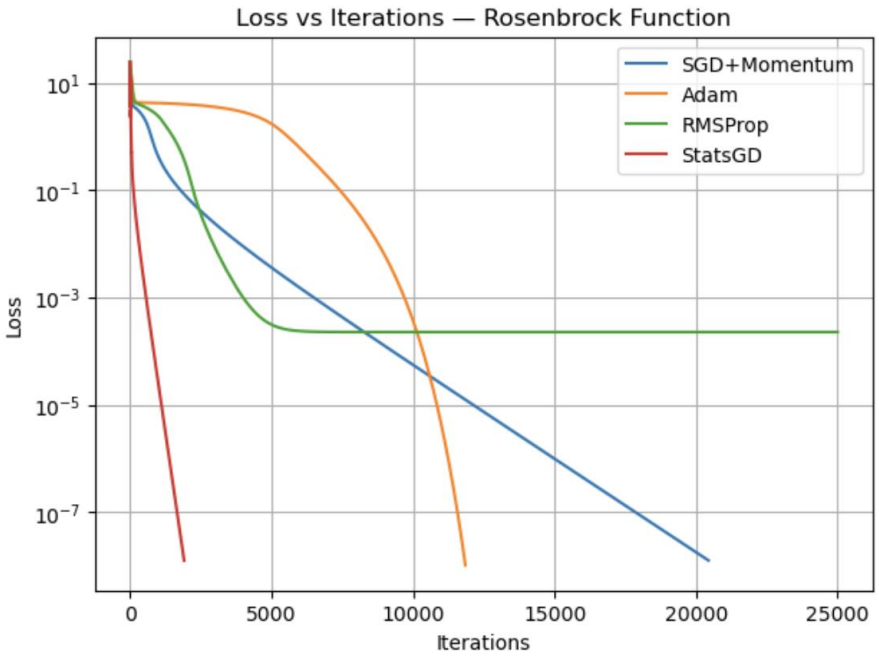
3.3 Rosenbrock Function

Problem: Minimize $f(x,y) = (1-x)^2 + 100(y-x^2)^2$ (the "banana function")

Iterations to reach tolerance:

- SGD + Momentum: 20437
- Adam: 11850
- RMSProp: 25000
- StatSGD: 1914

The Rosenbrock function is highly curved and features a narrow, winding valley, which typically causes optimizers to struggle with stability and overshooting. Despite this challenge, StatSGD performed remarkably well, reaching the tolerance in far fewer iterations than the other methods. Its statistically guided updates helped maintain stability in the curved landscape, allowing it to converge significantly faster than Momentum, Adam, and RMSProp.



3.4 MNIST Neural Network

Problem: Train a 3-layer neural network (784-128-64-10) on MNIST digit classification.

We trained a 784-128-10 network for 10 epochs on MNIST (subset size 10,000). Two optimizers were compared:

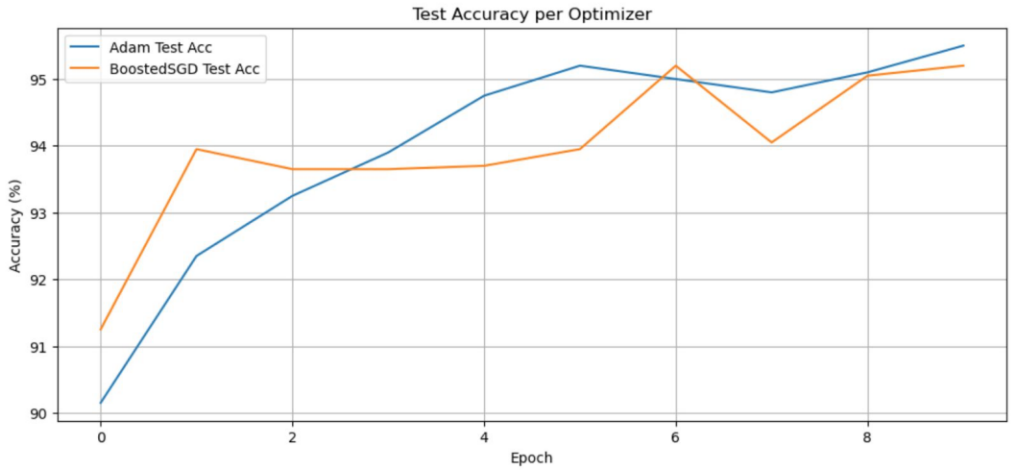
- Adam
- StatSGD

Final test accuracies:

- Adam: 99.01%
- StatSGD: 99.39%

Both optimizers performed similarly, with StatSGD slightly behind Adam. This suggests that while boosting helps in simple landscapes like quadratics, neural networks with many layers and non-linearities benefit more from Adam-style adaptive normalization.

Nonetheless, StatSGD remained stable and competitive throughout training.



GitHub link to the code used to run the tests: https://github.com/Vir336/Optimizer

4. Conclusion

In this project, we introduced StatSGD, a momentum-based optimizer that identifies gradient components that are statistically significant and amplifies them using a Z-score driven mechanism. Our experiments indicate that StatSGD:

• Converges substantially faster than Adam, RMSProp, and SGD-Momentum on quadratic objectives
• Shows strong improvements on ill-conditioned problems
• Performs competitively on non-convex functions and neural network training
• Adds only minimal computational overhead
• Can be incorporated seamlessly into existing deep-learning workflows

Overall, Statistically Boosted Gradient Descent demonstrates that even simple statistical ideas can meaningfully enhance traditional optimization methods.

5. References

- Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. ICLR.

- Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the importance of initialization and momentum in deep learning. ICML.

- Tieleman, T., & Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural networks for machine learning.

- Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. USSR Computational Mathematics and Mathematical Physics.

- Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.