GaussianAdam

A Noise-Driven Variant of Adam

Esraaj Sarkar Gupta

December 24, 2025

Project Codebase:
https://github.com/Esraaj-Sarkar-Gupta/GaussianAdam

Plaksha University

Abstract

While Adaptive Moment Estimation is the standard for training deep neural networks due to its stability and speed, it frequently converges to suboptimal local minima in non-convex loss landscapes. This report introduces GaussianAdam, a stochastic variant of the Adam optimizer that incorporates a time-decaying, positive folded Gaussian noise term directly into the learning rate. By treating the learning rate as a stochastic process, GaussianAdam facilitates a "jump-and-settle" dynamic – a high initial variance enables the traversal or high-energy barriers in the loss-surface to escape local basins, while a decay mechanism anneals the optimizer over time to a stable, deterministic trajectory (similar to Adam) to settle into a minimum.

This report evaluates GaussianAdam across multiple convex and non-convex benchmark functions, including ill-conditioned quadratics, the Rosenbrock function, the Rastrigin function, as well as a Multi-Layer Perceptron of MNIST. Experimental results demonstrate that GaussianAdam outperforms standard Adam in global optimization tasks – traversing a multimodal Rastrigin function where Adam stalls, while retaining comparable (or in some cases better) convergence speed and stability on convex, ill-curved loss surfaces. It demonstrates comparable accuracy to Adam on the MNIST Neural Network Training. The findings in this report suggest that positive-bias noise injection offers a robust method for improving exploration without sacrificing the convergence guarantees of Adaptive Moment Estimation.

1 Introduction

Among first order optimizers, Adaptive Moment Estimation, or Adam, is the most stable and widely deployed modern optimizer due to its robustness across architectures, resilience to gradient noise, and consistent performance without sensitive hyperparameter tuning. This is in contrast to other first order gradient descent based optimizers such as Stochastic Gradient Descent (SGD), Nesterov Accelerated Gradient (NAG), Root Mean Squared Propulsion (RMSProp), Adaptive Gradient Descent (AdaGrad), etc. which tend to either explode or stall under the same conditions.

Despite the stability and efficiency that Adam exhibits in smooth and curved optimization landscapes, it struggles to find global optimums in non-convex functions containing multiple local minima, periodic structures, flat regions or saddle points[1]. Gradient descent based optimizers, including adaptive optimizers are known to stall in such landscapes. In such circumstances, noise-driven methods (such as Stochastic Gradient Langevin Dynamics – SGLD) are necessary to escape suboptimal basins and explore the parameter space more effectively.

Although Adam excels as a local optimizer, it is not well suited for global optimization problems where exploration is critical. The aim of this report is to build a stochastic adaptive moment estimation based optimizer that is capable of performing on non-convex parameter spaces while preserving the stability and efficiency of Adam.

Inspiration
The idea of using a stochastic optimizer is, of course, not novel: plenty of noise-based samplers are already in deployment. One such popular candidate, mentioned previously, is Stochastic Gradient Langevin Dynamics (SGLD). SGLD is most used in Bayesian deep learning to approximate Bayesian posterior distributions as a lighter substitute to a full Monte-Carlo Markov Chain (MCMC) approach.

Although both methods mentioned above are based on Bayesian statistics and are typically slower and more complicated to tune, and are used as exploratory samplers, not optimizers, they both have structured Gaussian noise (such as Gaussian neighborhood sampling in Metropolis Hastings MCMC) as a part of their algorithm. In order to make an exploratory optimizer motivated by stochasticity[2], one may consider using a heuristic that samples a Gaussian distribution. Where the heuristic may be applied remains a task of simple trial-and-error.

Many possibilities were considered, such as adding / multiplying the noise directly to the parameters, the first or second moments, but best results were obtained from adding the noise directly to the learning parameter. It was taken into account that a noisy learning rate may prevent the optimizer from converging at any point. In order to solve this problem, it was decided that the variance of the Gaussian distribution that the noise term samples will be decreased with every step, necessitating the introduction of a new hyperparameter – the variance decay constant.

Future iterations of the optimizer may include a rolling hyperparameter.

2 Methodology and Algorithm

GaussianAdam attempts to tackle non-convex loss surfaces by adding controlled noise to the learning parameter of the standard Adam algorithm[3]. The Adam update functions unchanged for a function $f(\vec{\theta})$ where the gradient in step t is given by $g_t = \nabla_\theta f(\vec{\theta_t})$ are described below.

Adaptive Moment Estimation – Adam

The first (momentum) and second (velocity or RMS) moment updates

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

Bias correction

$$\hat{m}_t = \frac{m_t}{(1-\beta_1^t)} \quad (3)$$
$$\hat{v}_t = \frac{v_t}{(1-\beta_2^t)} \quad (4)$$

Parameter Update

$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \quad (5)$$

where,

$$\begin{aligned} \beta_1 &\in (0,1) \text{ is the momentum decay,} \\ \beta_2 &\in (0,1) \text{ is the second moment decay,} \\ \alpha &> 0 \text{ is the learning rate,} \\ \epsilon &\ll 1 \text{ is a very small number for divisional stability.} \end{aligned}$$

The GaussianAdam algorithm differs in the addition of a single step-dependent stochastic parameter ξ_t sampled from a right-tailed Gaussian distribution directly to the learning rate α. Effectively, the learning rate is transformed to be a function of time.

$$\alpha \mapsto \alpha_t = \alpha_0 + \xi_t \quad (6)$$

The GaussianAdam algorithm builds on the existing Adam method by introducing a new "noisy" term, along with a hyperparameter γ – the decay of the variance of the Gaussian noise.

Gaussian Adaptive Moment Estimation – GaussianAdam

The first (momentum) and second (velocity or RMS) moment updates

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

Bias correction

$$\hat{m}_t = \frac{m_t}{(1-\beta_1^t)}$$
$$\hat{v}_t = \frac{v_t}{(1-\beta_2^t)}$$

Noise Term Sampling and Updates

$$\xi_t = |\mathcal{N}(0, \sigma_t^2)| \tag{7}$$
$$\sigma_{t+1} = (1-\gamma)\,\sigma_t \tag{8}$$

Parameter Update

$$\theta_{t+1} = \theta_t - \alpha_0(1+\xi_t)\frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon} \tag{9}$$

where,

$$\begin{aligned}
\beta_1 &\in (0,1) \text{ is the momentum decay,}\\
\beta_2 &\in (0,1) \text{ is the second moment decay,}\\
\sigma_0 &> 0 \text{ is the initial noise standard deviation}\\
\gamma &\in (0,1) \text{ is the standard deviation decay,}\\
\alpha_0 &> 0 \text{ is the base learning rate,}\\
\epsilon &\ll 1 \text{ is a very small number for divisional stability.}
\end{aligned}$$

Remark: An earlier formulation of Eq. 7 and Eq. 9 was the inclusion of the base learning rate α_0 directly in the mean of a two tailed Gaussian distribution (as opposed to the right-tailed Gaussian that GaussianAdam uses now) being sampled from.

$$\xi_t \sim \mathcal{N}(\alpha_0, \sigma_t^2) \tag{10}$$
$$\theta_{t+1} = \theta_t - \xi_t\frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon} \tag{11}$$

After testing it was found that this formulation was outperformed by that of the formulation outlined in Eq. 9. The older formulation lead to negative or excessively large learning rates. The idea was to leave the expected value of the learning rate $E[\alpha_t] = E[\alpha_0 + \xi_t] = \alpha_0$. Retrospectively, it's fairly obvious that all this did was add directionless noise to the optimizer that did nothing but increase convergence time.

Such a formulation gives the optimizer a sufficient degree of noise as it starts allowing to explore the parameter space more aggressively, which then slowly begins to decay as each step progresses. With a well-chosen standard deviation decay γ, the variance decays enough to mimic pure Adam as it approaches the optimum. Table 1 demonstrates the decay of noise with step size for different decay rates.

The orange line – termed the expectation curve represents the expected value of the right-tailed Gaussian variable

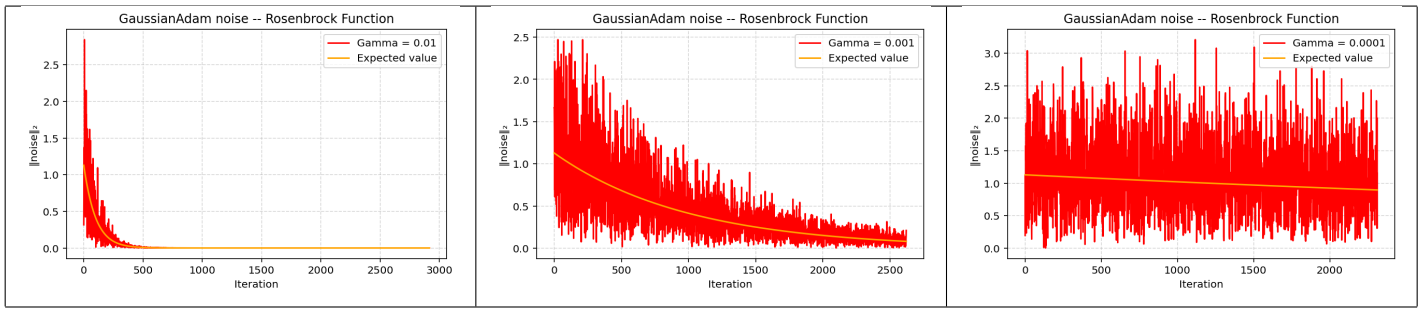

Table 1: The decay of sampled noise as the variance decay parameter is adjusted.

as the variance decays with step size. This value is given by the equation

$$E[\xi_t] = \sigma_t \sqrt{\frac{2d}{\pi}} \tag{12}$$

where d is the dimension of the noise vector.

Algorithm 1 GaussianAdam

Require: Initial parameters θ_0, learning rate α_0, $\beta_1, \beta_2 \in (0, 1)$, standard deviation decay $\gamma \in (0, 1)$, initial noise standard deviation σ_0, small constant ϵ

Initialize $m_0 \leftarrow 0$, $v_0 \leftarrow 0$, $t \leftarrow 0$, $\sigma \leftarrow \sigma_0$

for $t = 1, 2, 3, \ldots$ **do**

  Compute gradient $g_t = \nabla_\theta f(\theta_t)$

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

  $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$

  $\hat{m}_t \leftarrow \dfrac{m_t}{1 - \beta_1^t}$

  $\hat{v}_t \leftarrow \dfrac{v_t}{1 - \beta_2^t}$

  Sample noise term: $\xi_t \sim |\mathcal{N}(0, \sigma^2)|$

  $\sigma \leftarrow (1 - \gamma)\, \sigma$ ▷ standard deviation decay

  Effective learning rate: $\alpha_t \leftarrow \alpha_0 (1 + \xi_t)$

  Update parameters:

$$\theta_{t+1} \leftarrow \theta_t - \alpha_t \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

end for

3 Benchmark Testing Results

In this section, first performance of GaussianAdam is assessed on standard benchmarks for evaluating gradient-descent based optimizers and compared against the performance of other well-known gradient-descent algorithms.

All the algorithms are given a common learning rate of $\alpha_0 = \alpha = 0.02$ in the following benchmarks unless and otherwise mentioned. All other gradient-descent algorithms are given their default hyperparameter values (would be included in an Appendix).

GaussianAdam is given the following default hyperparameters:

$$\begin{aligned} \beta_1 &= 0.9 \text{ (Adam default)} \\ \beta_2 &= 0.999 \text{ (Adam default)} \\ \gamma &= 0.001 \\ \sigma_{(t=0)} &= 1 \text{ (Initial noise standard deviation)} \end{aligned} \tag{13}$$

3.1 Quadratic Functions

In order to evaluate the baseline convergence speed and numerical stability, a standard convex quadratic objective is introduced:

$$f(x) = \frac{1}{2} x^{\mathrm{T}} Q x \tag{14}$$

where Q is a symmetric positive definite matrix. A useful function $\kappa(Q)$ is defined as the ratio of the largest eigenvalue to the smallest eigenvalue of matrix Q.

$$\kappa = \frac{\lambda_{\max}(Q)}{\lambda_{\min}}. \tag{15}$$

3.1.1 Well-Conditions Quadratic Function

For any such quadratic functions, where $\kappa(Q)$ is small is said to be a well conditioned quadratic. Here we consider a quadratic with $\kappa = 10$.

Such functions are widely used are fundamental benchmarks since they isolate the pure convergence behavior of an algorithm, without complications from non-linearity, curvature or non-convexity. An example run of various gradient-descent based algorithms on such a function have been shown in Fig. 1 and Table 2.

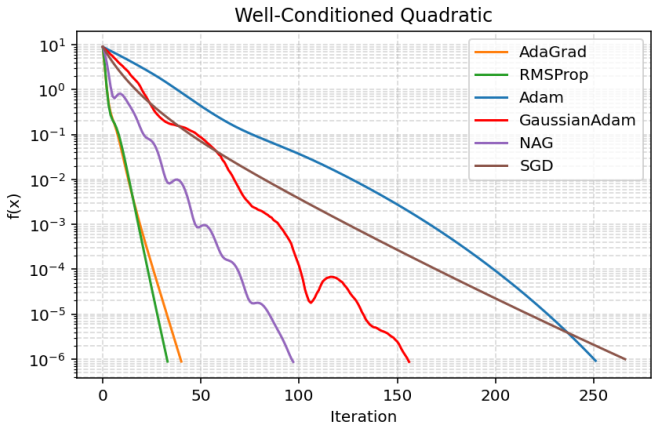

Figure 1: Well Conditioned Quadratic – Example Number of Iterations for Convergence

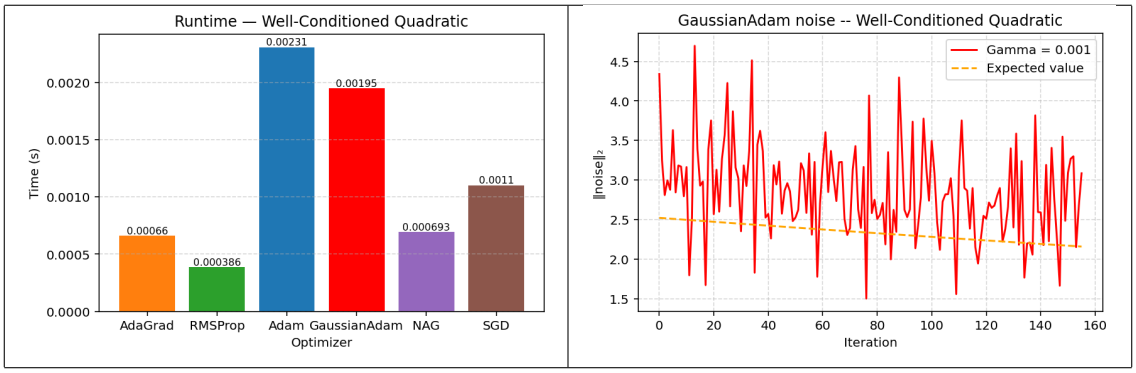

Table 2: Well Conditioned Quadratic Benchmark

From Fig. 1, we see that GaussianAdam converges with fewer iterations than Adam and SGD. It fails to beat the remainder of optimizers considered. As shown in Table 2, Adam is the only optimizer that takes longer than GaussianAdam.

3.1.2 Ill-Conditioned Quadratic Function

If one were to consider a much larger conditioning number κ, the resultant quadratic function may be instead termed an ill-conditioned quadratic. These functions introduce significant curvature disparity across directions, producing long, narrow level sets. Such surfaces are especially difficult for momentum-based gradient descent algorithms. Here we consider a function where $\kappa = 200$.

We see that GaussianAdam comfortably beats both AdaGrad and Adam in an ill-conditioned quadratic function. Adagrad does beat GaussianAdam in terms of runtime, but that result can be attributed to the simplicity of Adagrad's update sequence in comparison to Adam or GaussianAdam. With similar learning rates, the functions considered previously for the well-conditioned quadratic function fail in this benchmark, either stalling (RMSProp) or exploding (NAG).

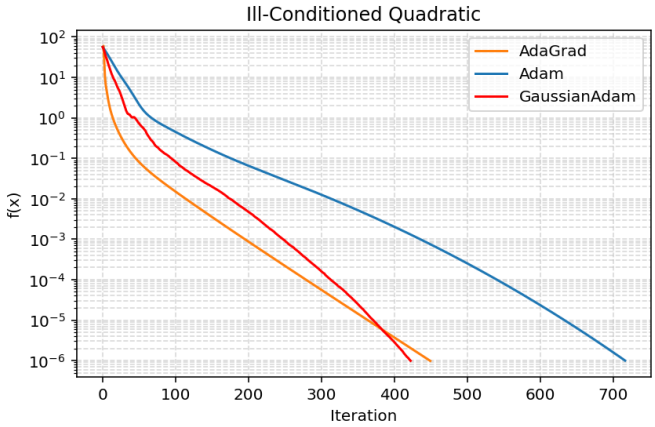

Figure 2: Ill Conditioned Quadratic – Example Number of Iterations for Convergence

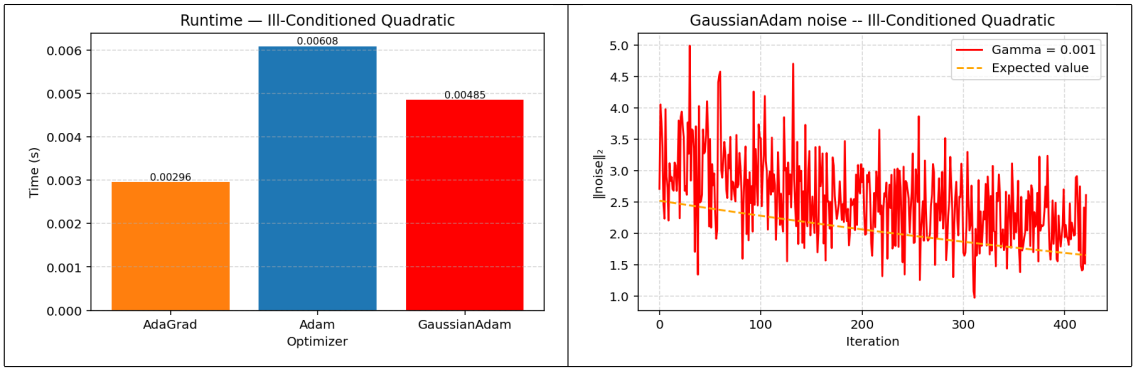

Table 3: Ill Conditioned Quadratic Benchmark

Given that GaussianAdam is a stochastic optimizer, it is important to note that there will be inherent differences in each run – where some runs perform exceedingly well. In order to better represent the performance of GaussianAdam, a 100 runs of GaussianAdam were carried out on the functions mentioned above, and the runtime and number of iterations from each run was recorded. The means and variances of the number of iterations until convergence, and the total runtime for each run have been presented in Table 4 and Table 5.

Conditioning Number	**Mean Iters**	**Std Iters**	**Mean Time (s)**	**Std Time (s)**
$\kappa = 10$	157.71	2.53	0.00182	0.00009
$\kappa = 200$	423.11	10.54	0.00484	0.00017

Table 4: Average performance of GaussianAdam across 100 runs on well-conditioned and ill-conditioned quadratic benchmarks.

3.2 Rosenbrock Function

The Rosenbrock function, also sometimes referred to as the "Banana Shaped Valley" function, and a popular benchmark to test optimizers due to its ill-shaped curvature. The 2D Rosenbrock function[4] is

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2. \tag{16}$$

Among all the standard gradient-descent based algorithms discussed for far, Adam is the only function that is capable of traversing the parameter space without exploding. With a well tuned standard deviation decay parameter γ, GaussianAdam should be able to find the valley early (earlier than Adam), and the noise must decay enough to ensure that GaussianAdam does not "jump" out of the valley.

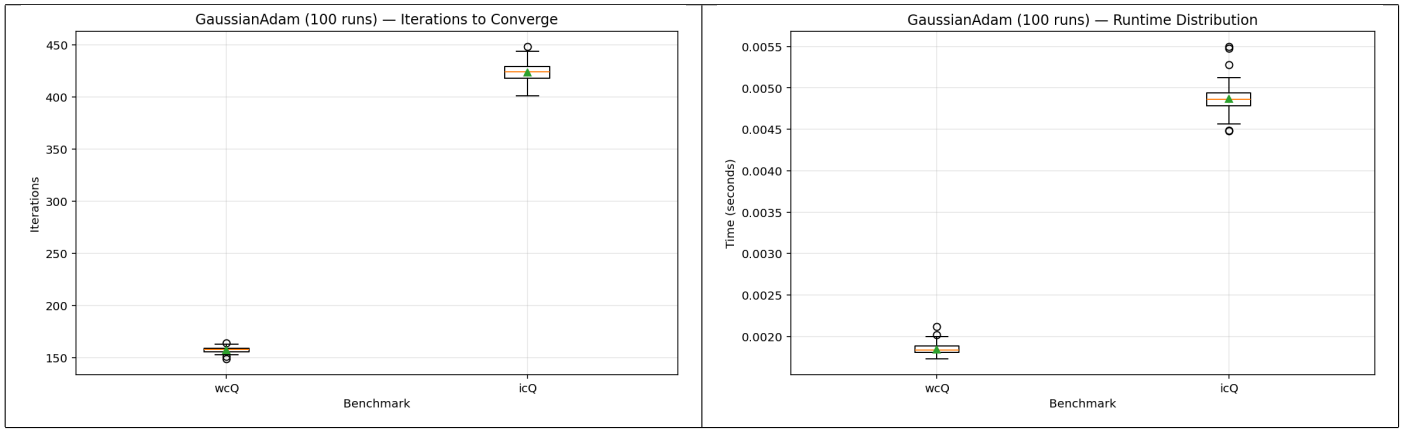

Table 5: Average number of iterations (left) and average runtime of GaussianAdam in well-conditioned and ill-conditioned quadratic functions.

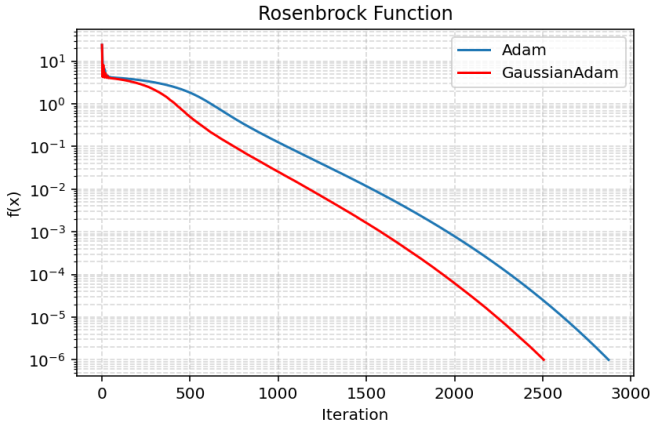

Figure 3: Rosenbrock Function – Example Number of Iterations for Convergence

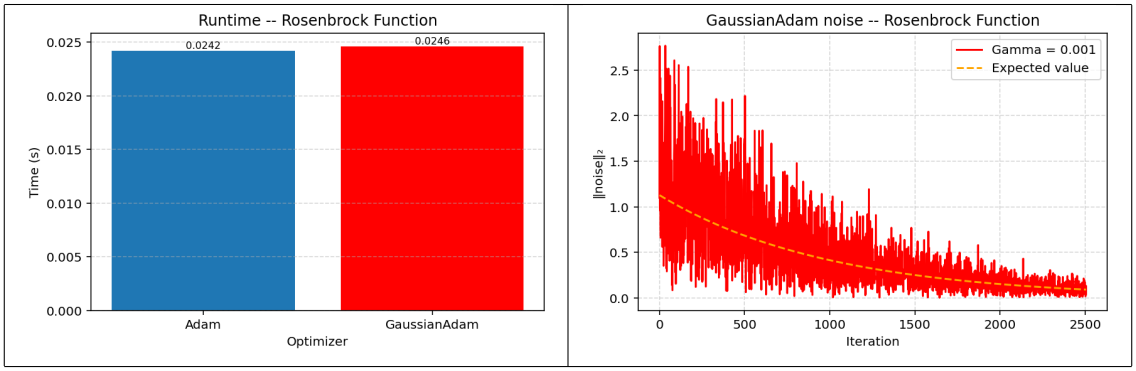

Table 6: Rosenbrock Function Benchmark

It is observed that GaussianAdam is able to converge to a minimum with significantly fewer iterations than Adam, although maintaining a comparable runtime. A statistical analysis of GaussianAdam's performance is outlined in Table 7 and Table 5, arrived at by carrying out 100 runs of GaussianAdam on the Rosenbrock function surface.

Adam takes 2875 iterations to converge (under these fixed parameters). From Table 7 we see that GaussianAdam has a mean iteration count of 2547.46, and a standard deviation of 95.84 iterations. It is evident that Adam's convergence is placed over 3 standard deviations from the mean number of iterations GaussianAdam requires to converge.

Benchmark	Mean Iters	Std Iters	Mean Time (s)	Std Time (s)
Rosenbrock	2547.46	95.84	0.02799	0.00196

Table 7: Average performance of GaussianAdam across 100 runs on the Rosenbrock benchmark function.

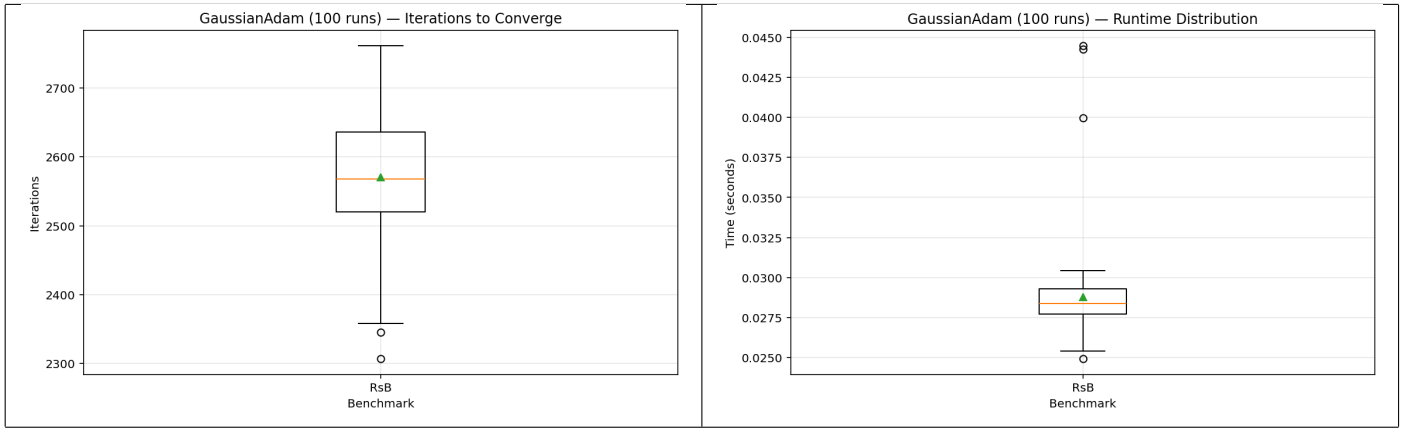

Table 8: Average number of iterations (left) and average runtime (right) of GaussianAdam in the Rosenbrock function.

3.3 MNIST

The MNIST Neural Network Benchmark is used as a benchmark for optimizers – introducing high-dimensional noisy gradient estimates, along with a non-convex loss surface. In such higher dimensional surfaces, local minima are rare, however saddle points do come up as problems where standard optimizers such as Adam can stall[5]. Hypothetically, one may use a stochastic sampler such as GaussianAdam to escape these saddle points.

However, it is observed that in the MNIST benchmark, GaussianAdam is unable to surpass Adam in terms of accuracy or runtime.

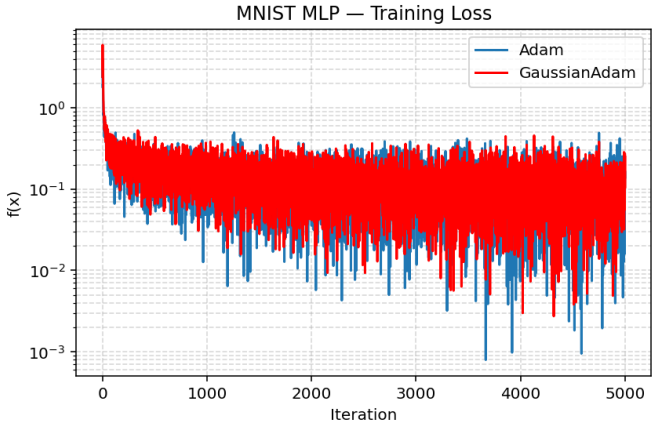

Figure 4: MNIST Neural Network – Example Number of Iterations for Convergence

From GaussianAdam's learning rate $\alpha_t = \alpha_0 + \xi_t$, it is always guaranteed that it will have a learning rate greater than that of Adam $\alpha = \alpha_0$. As a result, GaussianAdam is unable to enter sharp crevasses that Adam can (this is visible in Fig. 4.

Despite the stochastic nature of the proposed algorithm, GaussianAdam still demonstrates stability. From Fig. 4, we see that GaussianAdam follows a trajectory comparable to that of Adam, and results in an average test accuracy (**95.52%**) only marginally lower than that of Adam of **96.38%** ($\Delta < 1\%$). This evidences that GaussianAdam

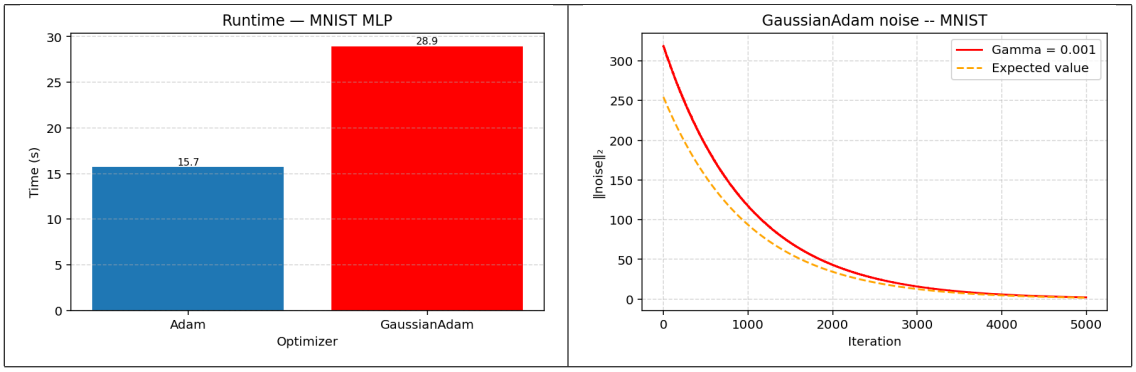

Table 9: MNIST Neural Network Benchmark

is capable of tolerating significant noise injection.

The observed stability of GaussianAdam is fundamentally driven by the proposed standard deviation decay mechanism mediated by the hyperparameter γ. As training progresses, the noise term ξ_t vanishes, and the learning rate α_t converges back to α_0, effectively allowing the algorithm to behave purely like Adam.

Run #	**Final Training Loss**	**Test Accuracy (%)**
1	0.1397	95.34
2	0.0954	95.82
3	0.1078	95.47
4	0.1214	95.40
5	0.0480	95.62
Mean	**0.1025**	**95.53**
Std Dev	**0.0309**	**0.17**

Table 10: Detailed experimental results of GaussianAdam on MNIST over 5 independent runs.

3.4 Rastrigin Function

The Rastrigin function[6] is a classical and widely used benchmark for global optimization. Since GaussianAdam is a stochastic sampler, it is imperative that its global exploration ability is evaluated. Typical deterministic gradient-descent based optimizers are known to perform poorly in such environments[7]. The function is highly non-convex, with multiple local minima, and unique global minimum at $f(x = 0) = 0$. It is defined for a vector $x \in \mathbb{R}^d$ as

$$f(x) = Ad + \sum_{i=1}^{d} x_i^2 - A\cos(2\pi x_i), \qquad A = 10. \tag{17}$$

For the default hyperparameters, GaussianAdam performs poorly in this benchmark – similar to Adam or Adagrad, settling in early in a local minimum. The hyperparameters of GaussianAdam – γ and σ_0 must be tuned appropriately for better results.

On increasing the initial standard deviation $\sigma_0 = 10$, and decreasing the standard deviation decay $\gamma = 0.0001$, we see significantly better results, as shown in Fig.7. In fact, one can visualize the various local minima that GaussianAdam explores in the same figure.

On running a brute-force hyperparameter search on GaussianAdam in the Rastrigin loss surface, the following results were found.
Although the best results are seen when initial standard deviation are high, leaving a high standard deviation unchecked by a low decay parameter γ can cause the function to exhibit erratic convergence where the algorithm

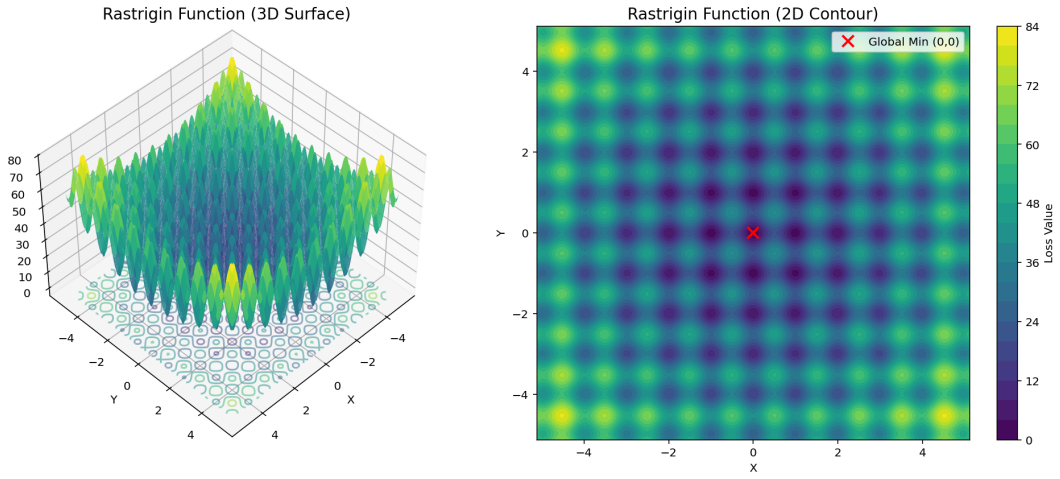

Figure 5: Rastrigin Function ($A = 10$)

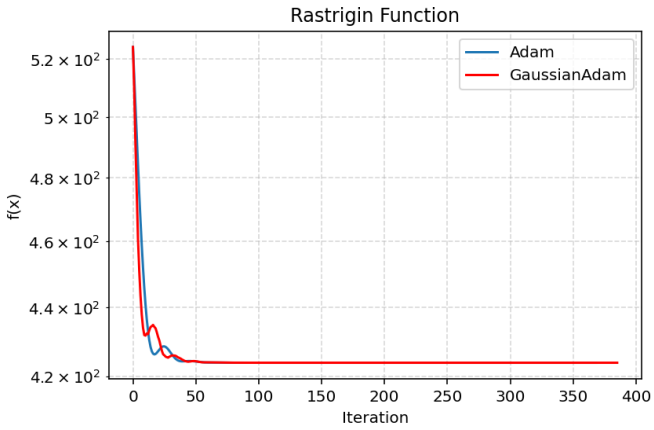

Figure 6: Adam and GaussianAdam on the Rastrigin Function Benchmark with default hyperparameters.

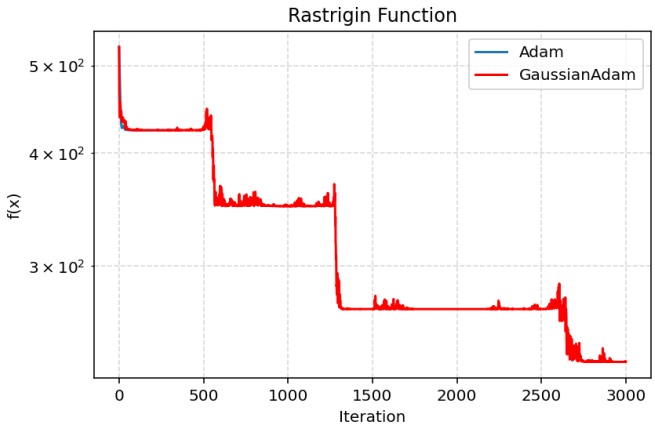

Figure 7: GaussianAdam tuned to $\gamma = 10^{-4}$ and $\sigma_0 = 10$ on the Rastrigin Function Benchmark

converges in worse local optima even after exploring better minima. This is demonstrated in Fig. 9 with $\gamma = 0.0001$ and $\sigma_0 = 15$.

Hyperparameter	Search Range	Optimal Value
Learning Rate (α_0)	$[0.01, 5.0]$	0.312
Standard Deviation Decay (γ)	$[0.001, 0.01]$	0.0067
Initial Noise Std (σ_0)	$[1, 10]$	10.0
Final Loss (f_{min})	–	**4.975**

Table 11: Hyperparameter search space and optimal values for GaussianAdam on the Rastrigin function.

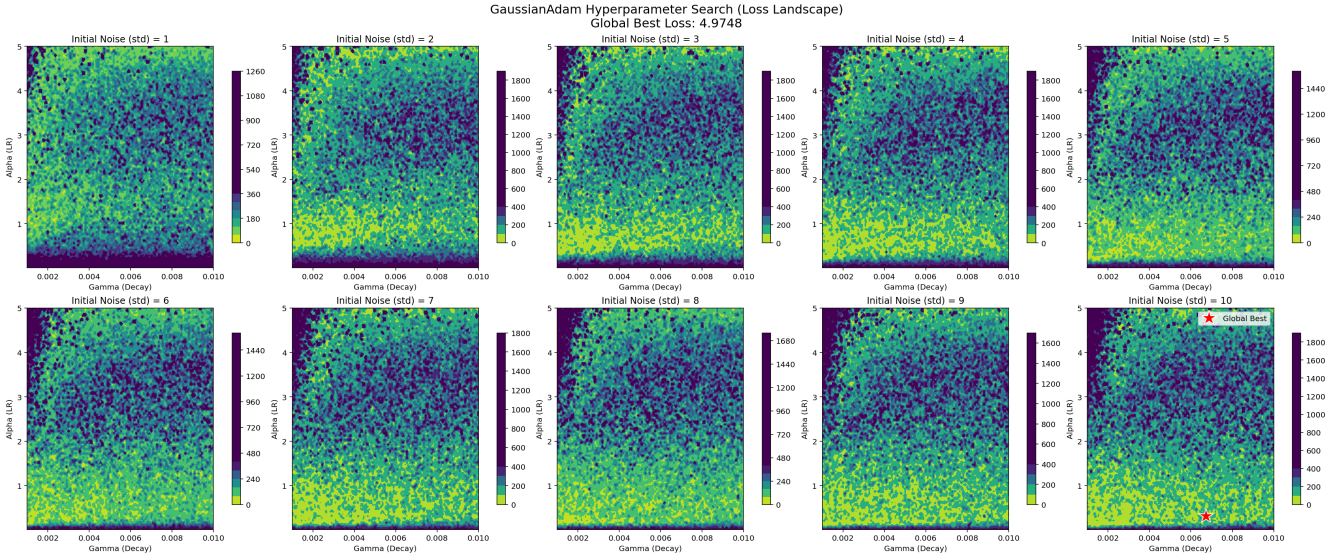

Figure 8: GaussianAdam hyperparameter search on the Rastrigin Function Loss Surface

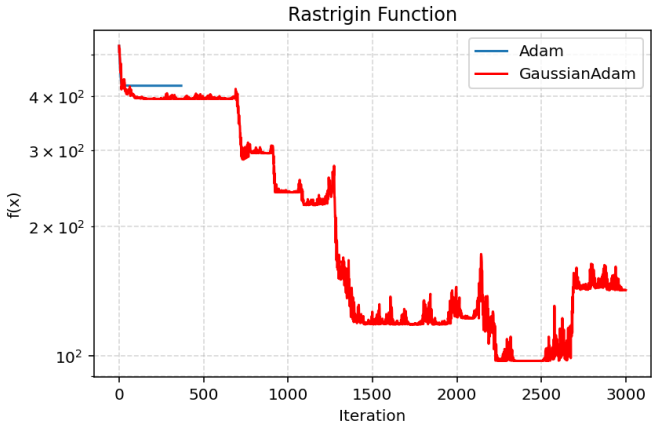

Figure 9: GaussianAdam with $\gamma = 0.0001$ and $\sigma_o = 15$ on the Rastrigin Loss Surface

4 Conclusions

GaussianAdam, a stochastic variant of the Adam optimizer that injects decayed, positive normal noise into the learning rate is introduced in this report. It has been demonstrated that GaussianAdam on average performs (either slightly or significantly) better than or comparable to Adam on local minimization benchmarks, all the while retaining Adam-like stability and capable of tolerating significant noise injection without stalling or exploding. The algorithm is annealed by the standard deviation decay parameter γ, allowing it to behave highly stochastically during its initial steps – encouraging exploration, and then gradually morphs into a standard deterministic Adam in its later stages. This design ensures that GaussianAdam retains the convergence guarantees of Adam – preventing divergence, while still benefiting from initial parameter agitation.

On the highly multimodal Rastrigin function, the algorithm outperformed standard baselines when tuned to a high initial variance, allowing the algorithm to exhibit a "jump-and-settle" dynamic, where high noise is required to traverse high energy barriers on the loss surface, while a low base learning rate (that the function eventually converges to) is required to settle into a stable convergence phase.

GaussianAdam proves to be a robust optimizer that trades training loss precision for broader landscape exploration. On average it is capable of surpassing Adam in simple convex scenarios, and its resistance to early trapping makes it a potential candidate for complex non-convex loss surfaces.

5 Future Work

Continuations of this work in the future require an Ablation Study to determine whether the noise introduced in GaussianAdam is truly responsible for its performance in convex loss surfaces, or if the success can be attributed to a simple addition to the learning rate. This can be done by adding the decaying expected value curve to the learning rate directly.

In this report GaussianAdam is never benchmarked against other standard global optimizers. It is imperative that GaussianAdam is tested against well-established global optimizers such as Stochastic Gradient Langevin Dynamics and AdamW (weight decay).

GaussianAdam must be tested on datasets harder than MNIST to assess its accuracy further.

The same algorithm can be tested out on different distributions with much heavier "tails" than the Gaussian distribution, such the Levy Alpha-Stable Distribution or Cauchy Distribution. This would allow the algorithm to make huge jumps (Levy Flights) occasionally[8].

References

[1] Hall Papadopoulos Greg B Fotopoulos Paul Popovich Nicholas. "REVIEW NON-CONVEX OPTIMIZATION METHOD FOR MACHINE LEARNING". In: (2024). arXiv: 2410.02017v1.

[2] Arvind Neelakantan et al. "Adding Gradient Noise Improves Learning for Very Deep Networks". In: (2015). arXiv: 1511.06807 [cs.LG].

[3] Diederik P. Kingma and Jimmy Lei Ba. "Adam: A Method for Stochastic Optimization". In: (2014). arXiv: 1412.6980 [cs.LG].

[4] H. H. Rosenbrock. "An Automatic Method for Finding the Greatest or Least Value of a Function". In: *The Computer Journal* (1960).

[5] Yann N. Dauphin et al. "Identifying and Attacking the Saddle Point Problem in High-Dimensional Non-Convex Optimization". In: (2014). arXiv: 1406.2572.

[6] LM Rastrigin. "Systems of extremal control". In: *Nauka* (1974).

[7] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. "On the Convergence of Adam and Beyond". In: *Published as a conference paper at ICLR 2018* (2018). arXiv: 1904.09237 [cs.LG].

[8] Umut Simsekli, Levent Sagun, and Mert Gurbuzbalaban. "A Tail-Index Analysis of Stochastic Gradient Noise in Deep Neural Networks". In: (2019). arXiv: 1901.06053 [stat.ML].