New Horizons for Adam Optimization: BGE-Adam for Adaptive and Resilient Performance in Dynamic Environments

Oussama ES SEMYRY, Louise LIGONNIERE, Amina MANSEUR
Ensae Paris
oussama.es-semyry@ensae.fr
louise.ligonniere@ensae.fr
amina.manseur@ensae.fr

Abstract

This project introduces BGE-Adam, an advanced variant of the Adam optimizer designed to improve adaptability, convergence speed, and robustness in dynamic applications. BGE-Adam incorporates algorithmic innovations to enhance model training under diverse, high-dimensional data conditions, making it particularly valuable in fields with high data variability and complexity. In our theoretical experiments, BGE-Adam showed fast convergence and outperformed Adam and SGD in some scenarios. However, experimental results were much less convincing.

1 Introduction

A central problem in Machine Learning is the optimization of some scalar parametrized objective function requiring maximization or minimization with respect to its parameters (for instance, the loss function when estimating a model). Optimization is a fundamental aspect of machine learning, encapsulating the trade-off between efficiency, accuracy, and convergence speed. The evolution of optimization algorithms in this field represents a continuous effort to tackle these challenges.

The basic algorithm on which the following ones rely is gradient descent. Its key idea is to perform few calculations far from the optimal solution, and increase the number of calculations when closer to the optimal value. An improvement is stochastic gradient descent, which introduces a stochastic objective function, for instance by randomly selecting a subset of the data at every step rather than the full dataset ("mini-batch") in order to reduce the computational cost.

Adam (adaptative moment estimation) [1] derives from stochastic gradient descent, combining the advantages of 2 methods:

- AdaGrad [2], which adapts the learning rate based on past gradients[1], allowing it to perform well for sparse gradients.
- RMSProp [3], which uses an exponentially weighted moving average of squared gradients to prevent the learning rate from decreasing too quickly. This allows for a more stable learning rate over time, making it effective for on-line and non-stationary settings.

Adam combines the benefits of momentum-based optimization and adaptive learning rates, and works with sparse gradients and non-stationary objective. An extension of Adam was developed in AdaMax: it replaces the L2 norm (used in Adam) with the infinity norm in its update rule, ensuring stable parameter updates even in the presence of very large gradients.

[1]The update is given by $\theta_{t+1} \leftarrow \theta_t - \alpha \frac{g_t}{\sqrt{\sum_{i=1}^{t} {g_i}^2}}$.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

This paper will focus on BGE-Adam, an improved optimization algorithm from Adam based on entropy weighting and adaptive gradient strategy, introduced by Shao et al. [4]. The goal is to improve the adaptability, convergence, and robustness of Adam. It presents three innovative technologies which we will describe thoroughly in Section 2.2. Shao et al. also presented some experimental results on its performance on lightweight neural network (MobileNetV2), applied to three images datasets (MNIST, CIFAR10, and a specialized medical image dataset). The optimizer showed improved accuracy when compared with the Adam algorithm, and better convergence and robustness.

In this project, we will try and complete those experiments with some theoretical and empirical tests of the optimizer. Our goal is to compare observed performance trends and to further explore BGE-Adam's strengths and limitations. We will evaluate BGE-Adam's adaptability and stability across high-dimensional data conditions by using it to estimate three classification models (logistic regression, a simple neural network and a complex neural network) on the MNIST image dataset. After thorougly presenting the algorithm and its theoretical advantages (Section 2), section 3 will describe the theoretical experiments (on tests functions) and empirical experiments (on image classification tasks) that were conducted.

2 The BGE-Adam optimizer: an upgrade on the Adam algorithm

2.1 The Adam algorithm

Algorithm 1 Adam (Adaptive Moment Estimation)

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize first moment vector)
$v_0 \leftarrow 0$ (Initialize second moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
  $t \leftarrow t + 1$
  $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
  $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
  $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
  $\hat{m}_t \leftarrow \frac{m_t}{1-\beta_1^t}$ (Compute bias-corrected first moment estimate)
  $\hat{v}_t \leftarrow \frac{v_t}{1-\beta_2^t}$ (Compute bias-corrected second raw moment estimate)
  $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}$ (Update parameters)
end while
return θ_t (Resulting parameters)

Algorithm 1 gives the Adam algorithm. The key feature of Adam is the computation of exponential moving averages of the gradient and the squared gradient (m_t and v_t). The weighting for each older datum decreases exponentially with a rate controlled by $\beta_1, \beta_2 \in [0, 1)$.

An important point in Adam is that m_t and v_t are initialized with value 0, so they are biased towards 0 especially on the first steps and when β's are close to 1 (since in that case the moving averages are decaying slowly from the first values). Therefore, Adam provides bias-corrected estimates $\hat{m}_t, \hat{v}_t$. This bias correction is especially crucial in the case of sparse gradients: averaging over many gradients by choosing a small value of β_2 is necessary to reach a reliable estimate of the second moment. In that case, a lack of initialization bias correction would create initial steps that are much larger, which could lead to non-convergence.

The stepsize is $\Delta_t = \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t}}$. Notice that

$$\begin{cases} |\Delta_t| \leq \alpha \frac{(1-\beta_1)}{\sqrt{1-\beta_2}} & \text{if} 1 - \beta_1 > \sqrt{1-\beta_2} \\ |\Delta_t| \leq \alpha & \text{otherwise} \end{cases}$$

Therefore, α approximately gives the magnitude of steps in the parameter space. α can be seen as a trust region around the current parameter value, beyond which the current gradient estimate does not provide sufficient information. The ratio $\frac{\hat{m}_t}{\sqrt{\hat{v}_t}} \approx \pm 1$ in most common scenarios. It can be seen as a Signal-to-Noise Ratio (SNR). A smaller SNR means that there is greater uncertainty around $\hat{m}_t$ and whether the direction of $\hat{m}_t$ corresponds to the direction of the true gradient, and therefore Δ_t will be closer to 0. On the contrary, a greater SNR means that g_t does not vary a lot, and therefore the step can be bigger. So Adam has the important advantage to moderate the magnitude of the step depending on the certainty around the gradient estimation.

2.2 The BGE-Adam algorithm

Algorithm 2 BGE-Adam

Inpute: initial point P_0, first moment decay, second moment decay, regularization constant, learning rate lr
Output: p_{new}
Initialize $m_0, v_0, \theta_0, \omega, \alpha, \beta_{1_{\min}}, \beta_{1_{\max}}, \beta_{2_{\min}}, \beta_{2_{\max}}$
for $t = 1$ **to** T **do**
$g_t = \nabla_\theta f_t(\theta_{t-1})$
$m_t = \beta_{1_{t-1}} \cdot m_{t-1} + (1 - \beta_{1_{t-1}}) \cdot g_t$
$v_t = \beta_{2_{t-1}} \cdot v_{t-1} + (1 - \beta_{2_{t-1}}) \cdot g_t^2$
$cr_t = \frac{\|g_t - g_{t-1}\|}{\|g_{t-1}\| + \epsilon}$
$\beta_{1_t} = \beta_{1_{\min}} + (\beta_{1_{\max}} - \beta_{1_{\min}}) \times (1 - cr_t)$; $\beta_{2_t} = \beta_{2_{\min}} + (\beta_{2_{\max}} - \beta_{2_{\min}}) \times (1 - cr_t)$
$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$; $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$
$step\_size = \frac{lr \cdot \sqrt{1-\beta_2^t}}{1-\beta_1^t}$
$\hat{g}_t = \alpha \cdot \hat{g}_{t-1} + (1 - \alpha) \cdot g_t$
$entropy\_adjustment = 1 + \omega \cdot \hat{E}[N_{p.data}(0, 1)]$
$p_{\text{new}} \leftarrow p - step\_size \times \frac{\hat{g}}{\sqrt{\hat{v}} + \epsilon} \times entropy\_adjustment$
end for
Return p_{new}

Algorithm 2 gives the BGE-Adam algorithm. The BGE-Adam algorithm has the same structure as the Adam algorithm and keeps the features that were introduced in the previous section (specifically, the momentum-based optimization and bias-correction term). It introduces three main innovations.

The first innovation is the dynamic adjustement of the decay rates of the moving average estimates of the first and second moments. β_1 and β_2 are adjusted at each iteration according to the change rate of the gradient $cr_t = \frac{\|g_t - g_{t-1}\|}{\|g_{t-1}\| + \epsilon}$. The idea is that the greater the change rate, the less stable is the gradient, which could suggest a local minimum. Therefore, β_1, β_2 are smaller to give less weight to the current gradient in the moment estimation. $\beta_{1_{\min}}, \beta_{1_{\max}}, \beta_{2_{\min}}, \beta_{2_{\max}}$ are bounds imposed on β_1 and β_2 at all timesteps.

The second innovation is the use of a gradient prediction model. The Signal-to-Noise Ratio is now computed using a predicted $\hat{g}_t$, given by $\hat{g}_t = \alpha \cdot \hat{g}_{t-1} + (1 - \alpha) \cdot g_t$. It corresponds to an exponentially weighted moving average, where α corresponds to the weights of past gradients. The closer α is to 1, the smoother the prediction, which enables to filter out some of the noise in the gradient. On the contrary, when α is small, the prediction responds more quickly to changes in the gradient.

The third innovation is the use of entropy weights to introduce randomness into the update steps, given by $e_t = 1 + \omega \cdot N(0, 1)$, where ω corresponds to the weight of noise[2]. This stochastic perturbation term helps prevent the optimization process from easily becoming stuck in local minima and is especially useful when facing complex loss surfaces.

[2]Notice that in **Algorithm 2**, we draw from a gaussian distribution in the same shape as $p.data$, and then take the mean of those draws, rather than just one draw from $N(0, 1)$.

3 Theoretical and Empirical Analysis

In this section, we evaluate the performance of the BGE Adam algorithm in comparison with other optimizers. The study is carried out in two stages: a theoretical evaluation on test functions and an empirical assessment on real-world problems.

3.1 Theoretical Experiment

3.1.1 Experimental Setup

To evaluate the performance of BGE Adam, Adam, and SGD, we designed an experiment targeting optimization problems with challenges like poor conditioning, flat regions, and local minima. The optimization procedure involves a custom implementation of BGE Adam with curvature-sensitive adjustments, while Adam and SGD use their standard PyTorch implementations. All algorithms are tested under identical conditions to ensure a fair comparison in convergence speed and efficiency.

Convergence is determined by the stopping criterion $\max_i |\nabla f(x)_i| < 10^{-6}$, ensuring optimization halts once the gradient components are sufficiently small. Hyperparameters such as learning rates and momentum are manually tuned for each algorithm to minimize iterations and ensure smooth convergence.

BGE Adam is implemented specifically for this study, while Adam and SGD rely on their standard PyTorch implementations, ensuring consistent and unbiased conditions for evaluation.

3.1.2 Conditioning and problem complexity

Optimization problems often involve navigating complex landscapes. Poor conditioning, non-convexity, and flat regions pose significant challenges for descent methods, as highlighted by the No-Free Lunch theorem for optimization [5]. In well-conditioned cases, smooth and strongly convex functions guarantee convergence to a minimizer with a zero gradient.

Another favorable case arises when we relax the assumptions and suppose only that the gradient is Lipschitz continuous. In this case, The convergence of gradient descent is governed by the following inequality:

$$f(x) - f(x - \eta \nabla f(x)) \geq \eta \left(1 - \frac{L}{2}\eta\right) \|\nabla f(x)\|_2^2, \quad \forall x, \forall \eta > 0.$$

Definition 1 *(Lipschitz Continuity)* A function $f : \mathbb{R}^n \to \mathbb{R}$ is **Lipschitz continuous** with constant L if:

$$|f(x) - f(y)| \leq L\|x - y\|, \quad \forall x, y \in \mathbb{R}^n.$$

Remark 1 *(Differentiability and Lipschitz)* For a continuously differentiable function $f(x)$, the Mean Value Theorem gives:

$$\frac{|f(x) - f(y)|}{\|x - y\|} = |f'(x + \theta(y - x))|,$$

for some $\theta \in (0, 1)$.

- If $|f'(x)| \leq L$, then $f(x)$ is Lipschitz continuous.
- Conversely, Lipschitz continuity implies $|f'(x)| \leq L$.

Theorem 1 *(Gradient Lipschitz Continuity)* For $f : \mathbb{R}^n \to \mathbb{R}$, twice continuously differentiable, if:

$$\|\nabla^2 f(x)\| \leq L, \quad \forall x,$$

where $\|\nabla^2 f(x)\|$ is the spectral norm of the Hessian, then $\nabla f(x)$ is Lipschitz continuous with constant L.

3.1.3 Gradient Descent Analysis

In this step, we explore the convergence properties of gradient descent with a fixed step size $\eta > 0$.

$$x_{k+1} = x_k - \eta \nabla f(x_k), \quad \eta > 0. \tag{1}$$

In this section, we assume that $f(x)$ is Lipschitz continuous the gradient $\nabla f(x)$ is Lipschitz continuous, which, however, does not necessarily imply $f(x)$ is Lipschitz continuous.

Lemma 1 *(Descent Lemma)* Assume $\nabla f(x)$ is Lipschitz continuous with Lipschitz constant L, then:

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|x - y\|^2.$$

Remark 2 *Notice that there is no assumption on convexity. But if we assume strong convexity of $f(x)$, then by Theorem 1:*

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2}\|x - y\|^2.$$

The proof (2) also implies:

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle - \frac{L}{2}\|x - y\|^2.$$

Lemma 2 *(Sufficient Decrease Lemma)* Assume $\nabla f(x)$ is Lipschitz continuous with Lipschitz constant L, then the gradient descent method satisfies:

$$f(x) - f(x - \eta \nabla f(x)) \geq \eta \left(1 - \frac{L}{2}\eta\right) \|\nabla f(x)\|^2, \quad \forall x, \forall \eta > 0.$$

Lemma 2 gives:

$$f(x - \eta \nabla f(x)) \leq f(x) + \langle \nabla f(x), -\eta \nabla f(x) \rangle + \frac{L}{2}\|\eta \nabla f(x)\|^2.$$

Lemma (2) implies that the gradient descent method (2.1) decreases the cost function, i.e., $f(x_{k+1}) < f(x_k)$ for any $\eta \in (0, \frac{2}{L})$. In practice, it is difficult to obtain the exact value of L. However, any sufficiently small positive step size η can ensure stability of the iteration (2.1) in the sense of $f(x_{k+1}) < f(x_k)$.

3.1.4 Test Functions

The following functions are tested, each presenting unique optimization challenges:

- **Quadratic Function:** This function highlights sensitivity to conditioning, where poor conditioning can cause numerical instability.Defined as $f(x) = \|\varepsilon^\top x\|^2$, with $\varepsilon = (1, \varepsilon, \varepsilon^2, \ldots, \varepsilon^n)$. Tests: $n = 2$, $\varepsilon = 1$ (well-conditioned) and $\varepsilon = 0.02$ (poorly-conditioned).
- **Non-Convex Gaussian Kernel:** $f(x) = 1 - \exp(-\|x\|^2)$, a non-convex "bell" shape with a large, flat plateau around its peak. The unique localized minimum tests the ability of algorithms to converge in non-convex landscapes where gradients vanish.
- **Rosenbrock Function:** Characterized by a curved valley ("banana" shape) with flat regions near the global minimum, slowing convergence.

These functions encompass challenges such as poor conditioning, non-convexity, flat regions, and multiple local minima. They provide a rigorous framework to assess the adaptability and robustness of optimization algorithms, including the BGE-Adam optimizer, under varied conditions.

3.1.5 Results

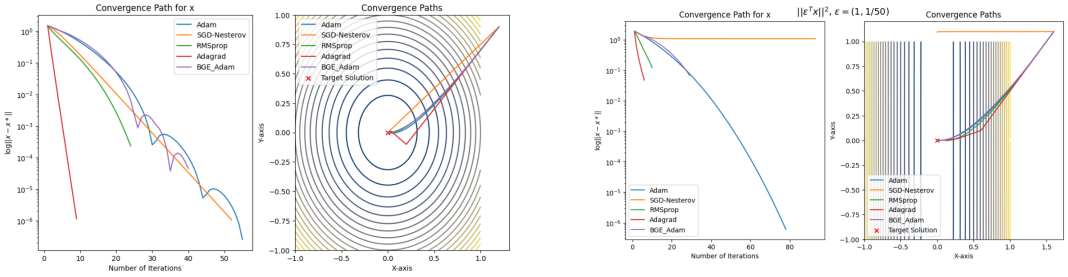
In this section, we present the results of our synthetic experiments, comparing the BGE-Adam with Adam, SGD with Nesterov, RMSprop, and Adagrad. The optimal hyperparameters vary for each algorithm, specifically tuned to achieve the closest proximity to the solution in the fewest iterations. Our empirical analysis of hyperparameter tuning revealed the following key insights:

1. **Learning Rate**: The learning rate is the most critical hyperparameter, affecting step size and performance across various surfaces. Smaller rates are needed for irregular surfaces to avoid oscillations. BGE Adam adapts the learning rate dynamically, maintaining stability even with suboptimal values on complex surfaces.

2. **Momentum Strength** (Adam's β_1): Momentum accelerates convergence by improving the algorithm's efficiency. Algorithms with momentum, like BGE Adam and Adam, converge faster than those without it, such as Adagrad, which converges more slowly and requires more iterations in complex problems.

3. **Smoothing Constant** (Adam's β_2 or RMSprop's α): The smoothing constant reduces oscillations, enhancing stability. BGE Adam is the smoothest, with an additional correction factor that minimizes oscillations, as seen in phase portraits.

Figures 1 and 2 compare the convergence, stability, smoothness, and speed of various optimization algorithms (*BGE-Adam*, *Adam*, *SGD-Nesterov*, *RMSProp*, and *Adagrad*). On each panel, the left figure depicts the logarithmic norm of the error $\log \|x_k - x^*\|$ versus the number of iterations, while the right figure illustrates the convergence paths in the parameter space.



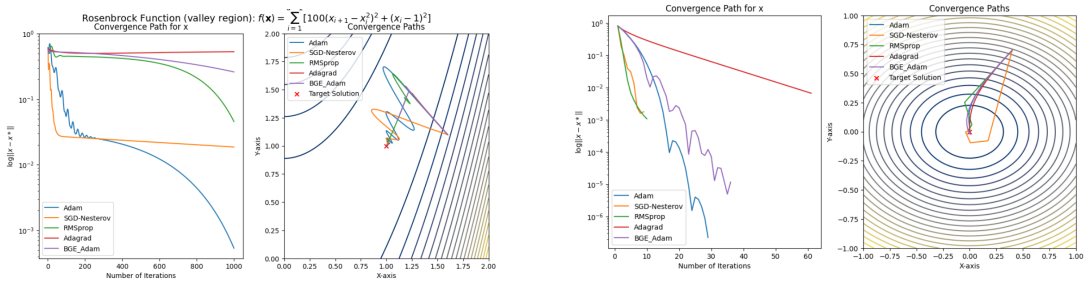
(a) Quadratic Function with good Conditioning

(b) Quadratic Function with bad Conditioning

Figure 1: Comparison of convergence paths for different optimizers.

Good Conditioning: BGE-Adam converges rapidly below 10^{-5} with smooth trajectories, outperforming Adam, which is slower but stable. Other algorithms (SGD-Nesterov, RMSprop, Adagrad) suffer from oscillations or stagnation.

Bad Conditioning: Adam excels, achieving convergence below 10^{-5} with a smooth trajectory. BGE-Adam (purple) converges slowly and stalls near 10^{-3}. SGD-Nesterov (orange) oscillates significantly, delaying convergence, while RMSprop (green) and Adagrad (red) stagnate above 10^{-2}, highlighting their inefficiency in poorly conditioned settings.



(a) Rosenbrock Function

(b) Non-Convex Gaussian Kernel

Figure 2: Comparison of convergence paths for different optimizers.

Rosenbrock Function : Adam shows excellent smoothness, converging directly to the target with minimal oscillations, reaching a stable radius below 10^{-3} in under 200 iterations. BGE_Adam remains stable with fewer oscillations than SGD-Nesterov, stabilizing around a radius of 10^{-2} after 300 iterations but failing to converge further. SGD-Nesterov, despite reaching a similar radius, exhibits significant instability in its path.

Non-Convex Gaussian Kernel: Adam achieves 10^{-5} with a stable trajectory, outperforming BGE-Adam, which stagnates near 10^{-3}. Other methods either stagnate or oscillate, confirming their inefficiency for ill-conditioned problems.

3.2 Empirical experiments

We conducted empirical experiments comparing the performance of BGE-Adam, Adam, and SGD on the MNIST dataset using three models: logistic regression, a simple neural network, and a complex neural network. The primary objective is to assess the generalization, convergence, and stability of different optimizers. The methodology involves two main steps:

- **Hyperparameter search** : Cross-validation determines optimal hyperparameters for Adam and SGD across the three models, while BGE-Adam autonomously adjusts its betas through its inherent mechanism.
- **Performance comparison**: With optimal hyperparameters, we analyze training loss, validation accuracy, gradient behavior, and test set metrics, including accuracy, precision, recall, and F1-score.

3.2.1 Data description

The MNIST dataset, widely used in the field of Machine Learning, consists of grayscale images of handwritten digits (0-9). Each image is 28×28 pixels, flattened into a feature vector of size 784. The dataset is divided into two subsets: a training set of 60,000 samples and a test set of 10,000 samples. Labels represent the digit class and are encoded as integers (0 to 9) or as one-hot vectors for certain tasks. For our experiments, the training set was further split into a training subset (48,000 samples) and a validation subset (12,000 samples) using an 80/20 ratio.

Data Preprocessing : The pixel values of the images were normalized to the $[0, 1]$ range to improve numerical stability and training efficiency. Labels were converted to the appropriate format for loss computation.

3.2.2 Models and estimation strategy

Models and optimizers We used three models of increasing complexity for the experiments. The first is a *Logistic Regression*, a linear classifier mapping the space of input features of dimension 784 to the space of output classes of dimension 10 through a single linear transformation followed by a log-softmax activation. The second is a *Simple Neural Network* with one hidden layer of 64 ReLU-activated neurons and a fully connected output layer. The third is a *Complex Neural Network* with three hidden layers (128, 64, and 32 neurons), ReLU activations, dropout regularization to reduce overfitting, and a fully connected output layer.

For all models, the loss function used is *cross-entropy*, which measures the divergence between actual truths and model predictions. The expression of this function for a single data point i is $\ell_i = -\sum_{m=1}^{M} y_{i,m} \log(p_{i,m})$ with $y_{i,m}$ is a binary indicator (0 or 1) indicating whether the class label m is correct for the observation i, $p_{i,m}$ the predicted probability that the observation belongs to the class m and M the total number of classes (here $M = 10$).

Hyperparameters and cross-validation To select the best hyperparameters, we used 5-fold cross-validation, dividing the dataset into five equal subsets. Each subset served once as the validation set while the others were used for training, ensuring every observation was validated once.

We evaluated configurations such as learning rate, Adam's betas, and SGD's momentum, using accuracy as the main metric. The best hyperparameters, selected based on the configuration minimizing average validation loss across folds, were used for subsequent experiments.

3.2.3 Results

Figure 3 shows that Adam converges quickly with a lower training loss than SGD and BGE-Adam, indicating more efficient optimization. SGD closely follows Adam, while BGE-Adam stabilizes at a higher, slightly increasing loss, reflecting optimization challenges. For validation accuracy, Adam slightly outperforms SGD with reduced oscillations, whereas BGE-Adam exhibits lower accuracy and significant fluctuations, likely due to its higher, unstable gradient norms.

Figure 4 shows that for the single neural network, Adam achieves faster convergence and a lower final training loss. SGD performs slightly worse but remains very close, while BGE-Adam converges more slowly yet reaches a similar loss eventually. For validation accuracy, Adam and SGD perform

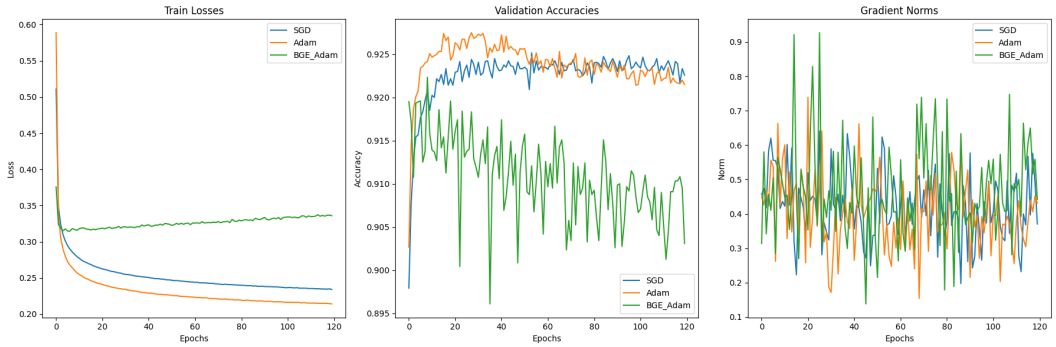


Figure 3: Training Loss, Validation Accuracy, and Gradient Norm for Logistic Regression over Epochs

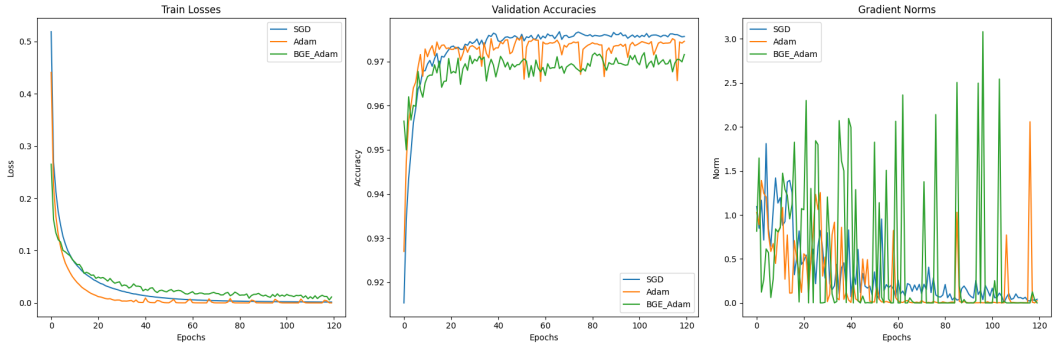


Figure 4: Training Loss, Validation Accuracy, and Gradient Norm for Simple Neural Network over Epochs

comparably, though Adam exhibits occasional negative fluctuations. BGE-Adam lags slightly in accuracy but shows fewer fluctuations. Its higher and more variable gradient norms indicate greater difficulty stabilizing compared to the other optimizers.

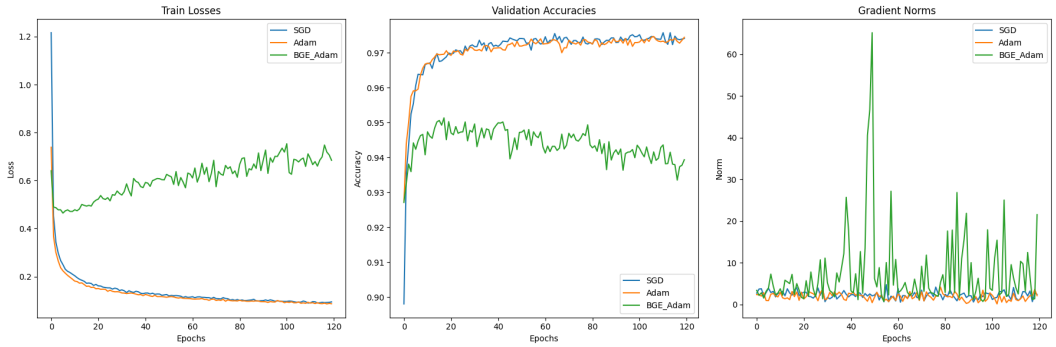


Figure 5: Training Loss, Validation Accuracy, and Gradient Norm for Complex Neural Network over Epochs

Figure 5 shows that for the complex neural network, Adam and SGD achieve rapid convergence to a low training loss, outperforming BGE-Adam, whose loss shows a slight upward trend. In validation accuracy, Adam and SGD perform similarly and maintain an advantage. BGE-Adam, however, exhibits instability with significant fluctuations and lower accuracy. Its gradient norms display sharper peaks than in previous models, suggesting overfitting and greater difficulty in stabilizing learning.

Overall training conclusion: Adam demonstrates global superiority across all three models due to its fast convergence, high accuracy, and stability, with SGD closely following. SGD offers greater

Table 1: Model evaluation results for different optimizers and metrics

Model	Metric	BGE	Adam	SGD
Logistic Regression	Accuracy	91.43%	92.47%	92.51%
	Precision	91.46%	92.45%	92.48%
	Recall	91.43%	92.47%	92.51%
	F1-score	91.43%	92.45%	92.48%
Simple NN	Accuracy	97.03%	97.38%	97.60%
	Precision	97.04%	97.38%	97.60%
	Recall	97.03%	97.38%	97.60%
	F1-score	97.03%	97.38%	97.60%
Complex NN	Accuracy	93.02%	97.32%	97.45%
	Precision	93.35%	97.32%	97.46%
	Recall	93.02%	97.32%	97.45%
	F1-score	93.07%	97.32%	97.45%

stability than BGE-Adam but occasionally converges more slowly than Adam. BGE-Adam exhibits instability, particularly in gradient norms, which limits its effectiveness for these models.

For logistic regression, Adam (92.47%) and SGD (92.51%) achieve nearly identical performance, both slightly outperforming BGE-Adam (91.43%). Although the difference is small, it remains consistent across all metrics, indicating BGE-Adam's slight sub-optimality for this simpler model, likely due to convergence challenges.

In the simple neural network, SGD achieves the highest accuracy (97.60%), followed closely by Adam (97.38%) and BGE-Adam (97.03%). These minor differences highlight SGD's effectiveness in stabilizing gradients for simpler architectures, with Adam maintaining strong performance and BGE-Adam trailing slightly.

For the complex neural network, SGD (97.45%) and Adam (97.32%) deliver comparable results, with a slight edge for SGD. In contrast, BGE-Adam performs significantly worse, with an accuracy of 93.02%. This decline could stem from gradient oscillations or challenges in hyperparameter tuning for complex architectures. SGD and Adam emerge as the most robust options for advanced models.

Finally, precision, recall, and F1-score metrics show uniformity across all optimizers and models, reflecting balanced performance on the test set. However, BGE-Adam consistently falls slightly behind, with its limitations becoming more evident in complex architectures.

4 Conclusion

In this project, we compared the efficiency of the BGE-Adam algorithm, an improved version of Adam, with other classical optimizers (Adam, SGD, RMSProp, Adagrad) through synthetic experiments and empirical tests on different models applied to the MNIST database.

The theoretical experiments highlight the strengths of BGE-Adam, including faster convergence and more stable trajectories on complex functions, often outperforming several optimizers. However, BGE-Adam remains less effective than Adam in certain cases, occasionally matching its performance in specific scenarios. On models applied to MNIST, its performance proved less robust, with Adam and SGD dominating in terms of accuracy, stability, and convergence speed, particularly for complex networks. While BGE-Adam performed competitively on simple models, it showed limitations such as increased instability and sub-optimal convergence in more complex architectures.

BGE-Adam is therefore an interesting candidate for complex tasks, although further research is needed to confirm its robustness on more varied datasets or in real application contexts.

Code

You can find the GitHub repository for this project at the following link: `https://github.com/AminaManseur29/Advanced_ML_project.git`.

References

[1] Diederik P. Kingma and Jimmy Lei Ba. Adam: a method for stochastic optimization. *Published as a conference paper at ICLR 2015*, 2015.

[2] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011. URL `http://jmlr.org/papers/v12/duchi11a.html`.

[3] T. Tieleman and G. Hinton. Lecture 6.5- rmsprop, coursera: Neural networks for machine learning. Technical report, Technical report, 2012. Online course material.

[4] Yichuan Shao, Jiantao Wang, Haijing Sun, Hao Yu, Lei Xing, Qian Zhao, and Le Zhang. An improved bge-adam optimization algorithm based on entropy weighting and adaptive gradient strategy. *Symmetry 2024, 16, 623*, 2024. URL `https://doi.org/10.3390/sym16050623`.

[5] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997. doi: 10.1109/4235.585893.

Appendix

4.1 Appendix 1 : Proofs of the main results

***Proof* 1** *(Theorem 1)* Using the Fundamental Theorem of Calculus for $g(t) = \nabla f(x + th)$:

$$\nabla f(x + h) - \nabla f(x) = \int_0^1 \nabla^2 f(x + th)h \, dt.$$

Applying the spectral norm $\|Ax\| \leq \|A\|\|x\|$:

$$\|\nabla f(x + h) - \nabla f(x)\| \leq \int_0^1 \|\nabla^2 f(x + th)\|\|h\| \, dt.$$

Since $\|\nabla^2 f(x)\| \leq L$, we have:

$$\|\nabla f(x + h) - \nabla f(x)\| \leq \int_0^1 L \, dt\|h\| = L\|h\|.$$

Letting $h = y - x$, we conclude:

$$\boxed{\|\nabla f(y) - \nabla f(x)\| \leq L\|y - x\|.}$$

Thus, $\nabla f(x)$ is Lipschitz continuous with constant L.

***Proof* 2** *(Lemma 1)* Let $g(t) = f(x + t(y - x))$. The fundamental theorem of calculus gives:

$$g(1) - g(0) = \int_0^1 g'(t)dt,$$

thus:

$$f(y) - f(x) = \int_0^1 \langle \nabla f(x + t(y - x)), y - x\rangle dt.$$

Let $z(t) = x + t(y - x)$. By subtracting $\langle \nabla f(x), y - x\rangle$ from both sides, we get:

$$|f(y) - f(x) - \langle \nabla f(x), y - x\rangle| = \int_0^1 \langle \nabla f(z(t)) - \nabla f(x), y - x\rangle dt.$$

Using the Cauchy-Schwarz inequality:

$$\leq \int_0^1 \|\nabla f(z(t)) - \nabla f(x)\|\|y - x\|dt,$$

$$\leq \int_0^1 Lt\|y - x\|dt\|y - x\| = \frac{L}{2}\|y - x\|^2.$$