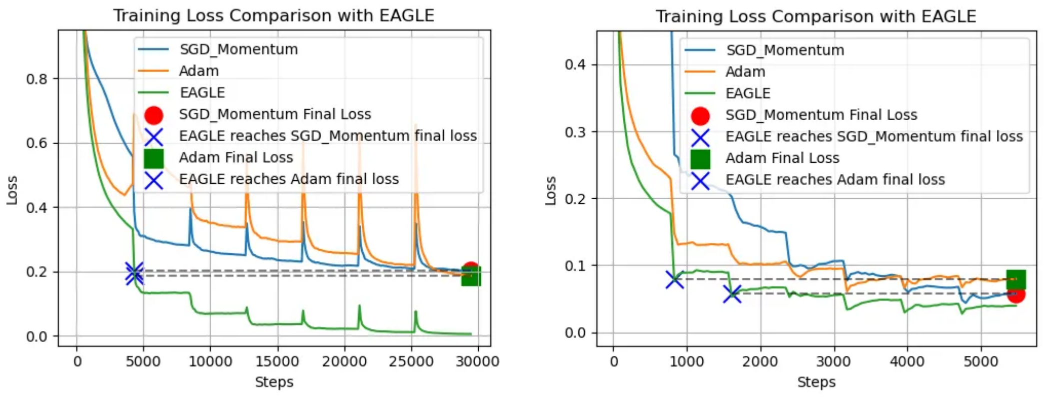
EAGLE: Early Approximated Gradient-based Learning-rate Estimator

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Improving the learning efficiency of deep learning models remains a significant research focus. In this paper, we propose **EAGLE (Early Approximated Gradient-based Learning-rate Estimator)**, a novel optimization method that accelerates parameter optimization. Firstly, to achieve faster loss convergence, EAGLE possesses the unique parameter update rule that leverages the local curvature of the loss landscape, derived from gradient variations between consecutive training steps. Secondly, to enhance training stability, it introduces a branching mechanism that adaptively switches to the existing Adam update rule under specific conditions where the EAGLE update rule might become unstable (e.g., extremely small gradient differences or locally upward convex shapes). In experiments on the GLUE SST-2 text classification task using a pre-trained GPT-2 model, EAGLE reached respectively the SGD with momentum's final loss value **6.83× faster** and the Adam's final loss value **6.77× faster**. Similarly, on the CIFAR-10 image classification task using a pre-trained ViT-B/16 model, EAGLE reached respectively the SGD with momentum's final loss value **3.41× faster** and the Adam's final loss value **6.60× faster**. To ensure reproducibility and promote further improvements, our code is publicly available on GitHub: https://github.com/keiotakmin/EAGLE



(a) Text (SST-2) Classification

(b) Image (CIFAR-10) Classification

Figure 1: Training Loss vs. Steps: EAGLE vs. Baseline Optimizers.

1 Intrduction

Background In the field of optimization, second-order methods such as Newton's method and quasi-Newton methods have been extensively studied and their application to deep learning has been attempted. These methods approximate the objective function as a quadratic function by utilizing the Hessian matrix or its approximations, enabling parameter updates that incorporate curvature information. For example, on a simple quadratic function, they can reach the optimum in a single step[1]. Even in more complex landscapes, they can approach minima via more direct paths[2]. Consequently, they are theoretically expected to converge faster than first-order methods [3][4].

Challenges Despite these theoretical advantages, Newton's method remains impractical for training large-scale models in modern deep learning. Computing and storing the full Hessian matrix incurs $\mathcal{O}(N^2)$ memory and computation costs for N parameters[5], which is prohibitive when modern networks contain millions to billions of parameters. While quasi-Newton methods like L-BFGS[6] reduce these costs by maintaining only a small window of past gradient–parameter pairs, their applicability in deep learning is limited: most empirical studies focus on fully connected layers, and extensions to other architectures like CNNs[7] and RNNs[8] remain underexplored. For these reasons, the promise of "curvature-aware" fast convergence has not been fully realized in practical deep learning.

Practicality of First-Order Methods As a result, first-order methods with high scalability and practicality have become the mainstream in modern deep learning. Stochastic Gradient Descent: SGD[9] remains a fundamental approach, achieving success in large-scale learning due to its computational and memory efficiency, and compatibility with mini-batch learning and momentum[10][11]. Adaptive methods such as Adam[12] have been emphasized for their suitability in large-scale problems involving massive datasets or model parameters. Extensions of Adam, including learning rate warmup and weight decay, have been shown to accelerate convergence in Transformer training[13] and enable stable large-scale pretraining of models like BERT[14]. However, these first-order optimization methods do not explicitly utilize curvature information, typically requiring many iterations to reach satisfactory convergence.

Contributions Based on these background and challenges, we propose a new optimization method: EAGLE (early approximated gradient-based learning-rate estimator). EAGLE aims to merge the practicality of first-order methods with the fast convergence potential of second-order methods, resulting in accelerating parameter optimization.

Our main contributions of this research are as follows:

- Clarifying the gap between the theoretical appeal of fast convergence from second-order methods and the practical limitations
- Proposing a novel optimization algorithm designed to benefit from curvature information in a practical form
- Demonstrating that EAGLE exhibits superior characteristics in both convergence speed and final performance compared to conventional methods, successfully balancing optimization speed and practical usability

2 The EAGLE Algorithm

In this section, we detail the EAGLE algorithm. EAGLE introduces a novel update rule: **EAGLE Update Rule** that enhances parameter update efficiency, and an **Adaptive Switching Mechanism** that improves training stability.

2.1 EAGLE Update Rule

The foundation of EAGLE is the "EAGLE update rule," a simple approach designed to efficiently approximate and utilize the curvature information of the loss function. It is formulated as follows:

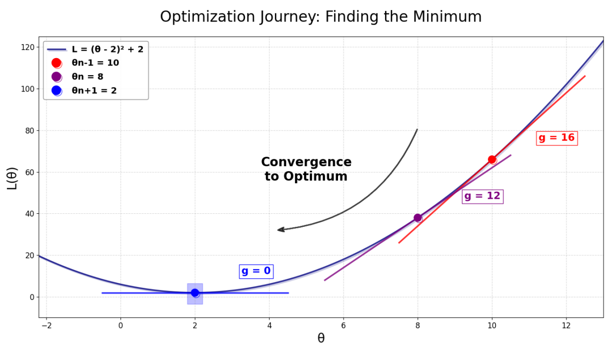


Figure 2: Visualization of the parameter update process in the example $L = (\theta - 2)^2 + 2$. Three key points are shown: $\theta_{n-1} = 10$ with $g = 16$, $\theta_n = 8$ with $g = 12$, and the optimal parameter $\theta_{n+1} = 2$ with $g = 0$.

$$\theta_{n+1} = \theta_n - \eta \cdot \frac{\theta_n - \theta_{n-1}}{\dfrac{\partial L}{\partial \theta_n} - \dfrac{\partial L}{\partial \theta_{n-1}}} \cdot \frac{\partial L}{\partial \theta_n} \tag{1}$$

The EAGLE update rule utilizes not only the current parameters θ_n and gradient $\partial L/\partial\theta_n$, but also the previous step's parameters θ_{n-1} and gradient $\partial L/\partial\theta_{n-1}$. By calculating the differences between these values and taking their ratio, we can estimate the curvature of the loss function between the current and previous steps.

The EAGLE update rule can be theoretically justified as an approximation of Newton's method. In Newton's method, the update is $\theta_{t+1} = \theta_t - H_t^{-1} g_t$, where H_t is the Hessian matrix. While computing the full Hessian matrix is prohibitively expensive, it is possible to approximate H_t^{-1} using the ratio of differences, providing a directional estimate of the inverse Hessian-gradient product along the optimization trajectory. Through this formulation, EAGLE implicitly captures curvature information without explicitly computing second derivatives, achieving a balance between the fast convergence and the computational efficiency.

To verify the effectiveness of the EAGLE update rule, we consider a simple quadratic function $L = (\theta - 2)^2 + 2 = \theta^2 - 4\theta + 6 \quad (\partial L/\partial\theta = 2\theta - 4)$ as an example in Fig. 2. For theoretical clarity, we omit the explicit learning rate η from equation (1) in this example—while in practice, a learning rate improves stability in complex neural-network loss landscapes (Appendix A). Setting $\partial L/\partial\theta_n = g_n$, we apply the EAGLE update rule to estimate the optimal parameter θ_{n+1}.

$$\theta_{n+1} = \theta_n - \frac{\theta_n - \theta_{n-1}}{g_n - g_{n-1}} \cdot g_n \quad \Rightarrow \quad \theta_{n+1} = 8 - \frac{8 - 10}{12 - 16} \cdot 12 = 2$$

At $\theta_{n+1} = 2$ in the original function $L = (\theta - 2)^2 + 2$, the gradient becomes $g_{n+1} = 2 \cdot 2 - 4 = 0$, coinciding exactly with the minimum point. This result demonstrates that, for a purely quadratic loss, one EAGLE update suffices to locate the minimizer. By generalizing this behavior, we expect that in neural-network training—where the loss is locally well-approximated by a downward-opening quadratic—EAGLE's update rule can significantly accelerate convergence.

2.2 Adaptive Switching Mechanism

The EAGLE update rule presents two primary challenges in practical applications. To address these issues, we introduce an "Adaptive Switching Mechanism" for the update rule, which aims to improve the overall stability of training. Specifically, EAGLE operates in conjunction with Adam: when conditions indicate that the EAGLE update rule is likely to be unstable or ineffective, the algorithm automatically switches to an Adam update rule. We adopt Adam as the alternative method due to its widely recognized robustness and generalization performance. However, the switching framework itself is agnostic to the choice of backup optimizer and can, in principle, accommodate any suitable

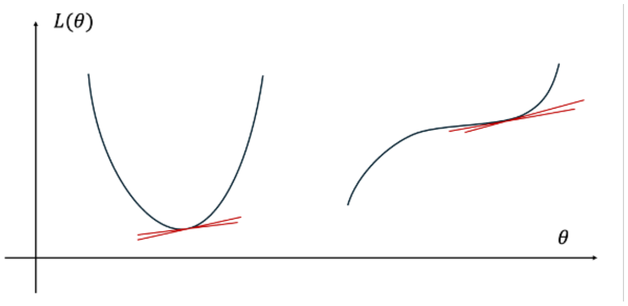


Figure 3: Gradient Change Patterns Causing Divergent Updates: Two Example Scenarios

alternative. In the following two subsections, we first describe the practical limitations of the EAGLE update rule, and then present the specific criteria under which the update rule transitions between EAGLE and Adam.

2.2.1 Based on Gradient Difference

The first practical challenge arises when the gradient difference $\Delta(\partial L/\partial\theta)$ becomes exceedingly small, causing the update magnitude to diverge. This phenomenon is characterized by the following relationship:

$$\Delta\frac{\partial L}{\partial\theta} \to 0 \quad \Rightarrow \quad \frac{\Delta\theta}{\Delta\frac{\partial L}{\partial\theta}} \to \pm\infty \quad \Rightarrow \quad \theta_{n+1} = \theta_n - \eta\cdot\frac{\Delta\theta}{\Delta\frac{\partial L}{\partial\theta}}\cdot\frac{\partial L}{\partial\theta_n} = \theta_n \mp \infty \to \mp\infty$$

This instability typically manifests in two common phases in Fig 3. First, during the late stages of training, gradients tend to become small as parameters approach minima. Consequently, the gradient differences between consecutive steps can become extremely small, potentially causing the divergent update (left of Fig 3). Second, in locally flat regions, gradients remain nearly constant at small values, similarly leading to extremely small gradient differences (right of Fig 3). To address this challenge, we introduce the following branching mechanism:

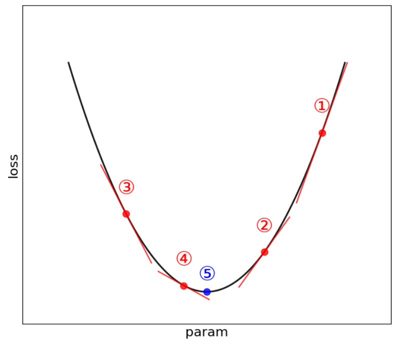
$$\textit{condition1}: \left|\Delta\left(\frac{\partial L}{\partial\theta}\right)\right| < \textit{threshold}, \quad \textit{update\_rule} = \begin{cases} \textit{Adam} & \textit{(if condition1)} \\ \textit{EAGLE} & \textit{(otherwise)} \end{cases} \tag{2}$$

This mechanism enables a flexible update strategy: Adam is applied in regions where gradient changes are minimal (to ensure numerical stability), while EAGLE is applied in regions where gradient changes are sufficiently large, taking advantage of its fast convergence properties.

The *threshold* is a critical hyperparameter in this design. Since the optimal threshold varies depending on the training stage and network architecture, a single fixed threshold cannot adequately guarantee versatility. If the threshold is set too large, the optimizer defaults too frequently to Adam, negating the benefits of EAGLE. If it is too small, the intended effect—mitigating update divergence—is lost. To overcome this, we introduce an adaptive thresholding mechanism based on the statistical properties of recent gradient norms. The threshold τ_n is dynamically computed as follows:

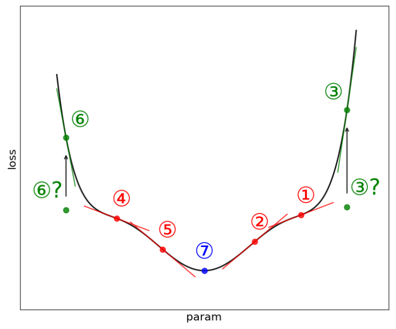
$$\tau_n = \min\left(\max\left(\alpha\cdot\frac{\sigma_g}{\mu_g+\epsilon}, \tau_{\min}\right), \tau_{\max}\right) \tag{3}$$

where μ_g and σ_g denote the mean and standard deviation of the gradient norms over the last 10 steps. The ratio σ_g/μ_g represents the coefficient of variation (CV), which quantifies the relative variability of gradients. The hyperparameter α is a scaling factor (we use 5×10^{-3} in our experiments), while $\tau_{\min}$ and $\tau_{\max}$ (set to 10^{-5} and 10^{-2}, respectively) define lower and upper bounds for the threshold. $\epsilon = 10^{-8}$ is a small constant added for numerical stability. This mechanism is activated from the fifth training step; before that, a fixed initial threshold τ_0 is used. When gradient variability is high



(a) Consistently Positive Second Derivative

Point with 1: large positive gradient, 2:small positive gradient, 3:large negative gradient, 4:small negative gradient, 5:zero gradient (global minimum)



(b) Sign-Changing Second Derivative

Point with 1: small positive gradient, 2:large positive gradient, 4:small negative gradient, 5:large negative gradient, 7:zero gradient (global minimum). 3,6:Incorrect estimation point

Figure 4: Illustrative Loss Landscapes

(i.e., large CV), the loss surface is assumed to be complex, and the threshold increases, favoring Adam updates. Conversely, when the CV is small, indicating smoother loss landscapes, the threshold decreases, allowing EAGLE updates to take precedence. Through this design, the optimizer can automatically switch between update rules depending on the local gradient behavior.

2.2.2 Based on Loss Landscape

The second practical challenge arises when the loss surface is locally non-convex. The EAGLE update rule is expected to function effectively when the loss function has no local solutions and can be approximated as downward convex. However, even if the global landscape is approximately convex, there may be local regions where the curvature is upward-opening, in which case EAGLE update can fail to achieve optimal parameter estimation. To address this challenge, we categorize the loss function shapes and its variation pattern based on the sign of the second derivative, and analyze EAGLE's effectiveness in each case.

Consistently Positive Second Derivative When the loss function is globally convex—that is, its second derivative is strictly positive—the gradient changes monotonically with the parameter in the left side of Fig 4. Transitions between four key points on this curve fall into three basic patterns:

- Transition 1–1: Monotonic decrease in positive gradient (Point 1$\rightarrow$Point 2)
- Transition 1–2: Monotonic increase in negative gradient (Point 3$\rightarrow$Point 4)
- Transition 1–3: Oscillation near the optimum (Point 2 $\leftrightarrow$ Point 4)

For convergence analysis, we validate using the simple quadratic function shown in Fig 2. Since the effectiveness for Transition 1–1 has already been confirmed in section 2.1, we focus on the Transitions 1–2(equation (4)) and 1–3(equation (5)).

$$(\theta_{n-1}, g_{n-1}) = (-8, -20),\ (\theta_n, g_n) = (-3, -10) \quad \Rightarrow \quad \theta_{n+1} = -3 - \frac{5}{10}\cdot(-10) = 2 \tag{4}$$

$$(\theta_{n-1}, g_{n-1}) = (-1, -6),\ (\theta_n, g_n) = (5, 6) \quad \Rightarrow \quad \theta_{n+1} = 5 - \frac{6}{12}\cdot 6 = 2 \tag{5}$$

The obtained $\theta_{n+1} = 2$ coincides with the parameter that minimizes the function. Therefore, when the loss function is globally convex and its gradient changes monotonically, the EAGLE update

Table 1: Gradient Sign Transitions and EAGLE Effectiveness

Transition	∇L_{n-1}	∇L_n	$\Delta \nabla L$	EAGLE Effective?
1–1	Large +	Small +	$-$	✓
1–2	Large $-$	Small $-$	$+$	✓
1–3	$+/-$	$-/+$	$-/+$	✓
2–1	Small +	Large +	$+$	✗
2–2	Small $-$	Large $-$	$-$	✗

rule exhibits convergence toward the global minimum across all fundamental transition patterns, confirming its effectiveness.

Sign-Changing Second Derivative When the loss surface exhibits local concave–convex structure (i.e., the second derivative changes sign), the gradient response to parameter changes becomes more complex in the right side of Fig 4. In this scenario, two new transition patterns not observed in the "consistently positive second derivative" case exist:

- Transition 2–1: Increasing positive gradient (Point 1$\rightarrow$Point 2)
- Transition 2–2: Decreasing negative gradient (Point 4$\rightarrow$Point 5)

Since the EAGLE update rule is designed to drive the gradient toward zero under the assumption of downward curvature, applying it here can lead to incorrect updates that increase the loss. Specifically in Fig 4, in Transition 2-1, Point 3?, and in Transition 2-2, Point 6? are estimated as parameters where the gradient converges to zero, and the actual losses and gradients at these parameters are at Points 3 and 6, indicating inappropriate updates. We summarize these findings by examining the signs of the previous gradient ∇L_{n-1}, current gradient ∇L_n, and gradient change $\Delta \nabla L$ in Table 1. From this, we identify two simultaneous conditions under which the EAGLE update rule fails: "Consistency of gradient signs between consecutive steps" and "Consistency of signs between current gradient and gradient change". To handle these inappropriate updates, we introduce the following branching mechanism:

$$\textit{condition2} : (\nabla L_{n-1} \cdot \nabla L_n \geq 0) \wedge (L_n \cdot \Delta \nabla L \geq 0), \quad \textit{update\_rule} = \begin{cases} \textit{Adam} & \textit{(if condition2)} \\ \textit{EAGLE} & \textit{(otherwise)} \end{cases} \tag{6}$$

2.2.3 Integration

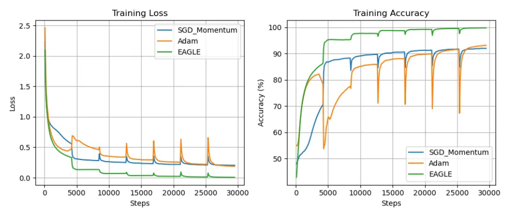
Based on these analyses, we integrate the two branching criteria into a unified switching mechanism. Both the gradient-difference threshold check from section 2.2.1 (condition 1) and the loss-landscape check from section 2.2.2 (condition 2) serve to trigger a switch to the Adam update. Therefore, we combine them with a logical OR and formulate a single selection rule:

$$\textit{condition} : \textit{condition1} \vee \textit{condition2}, \quad \textit{update\_rule} = \begin{cases} \textit{Adam} & \textit{(if condition)} \\ \textit{EAGLE} & \textit{(otherwise)} \end{cases} \tag{7}$$

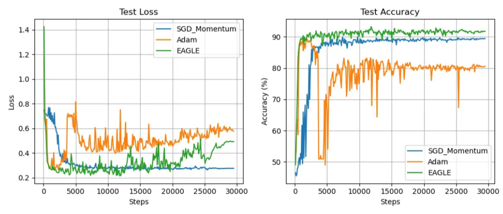
This unified branching mechanism enables a consistent handling of both practical challenges—namely, divergence caused by vanishing gradient differences and incorrect updates in locally concave regions—within a single framework.

3 Related Work

Adaptive Moment Estimation: Adam is an optimization method that combines the advantages of AdaDelta [15] , Momentum [16], and RMSprop [17]. The first-order moment (mean) incorporates the Momentum concept similar to AdaDelta, calculating the moving average of gradients to consider the directional trends of past gradients and appropriately determine the update direction. This



(a) Training Loss/Accuracy



(b) Test Loss/Accuracy

Figure 5: Results on SST-2 (GPT-2) Text Classification

Table 2: Optimizer Training/Test Loss and Accuracy on GPT-2 (SST-2) Task

Optimizer	Train Loss	Train Acc (%)	Test Loss	Test Acc (%)
SGD_Momentum	0.2035	91.98	**0.2760**	89.45
Adam	0.1863	93.15	0.5773	80.62
EAGLE	**0.0057**	**99.81**	0.4929	**91.74**

Table 3: EAGLE Update Usage Analysis on GPT-2 (SST-2) Task

Optimizer	Avg Usage (%)	Min Usage (%)	Max Usage (%)
EAGLE	20.50	12.55	32.11

mechanism enables stable updates even with significant gradient changes. The second-order moment (variance) incorporates the RMSprop concept similar to AdaDelta, calculating the moving average of squared gradient values to adaptively adjust learning rates individually for each parameter. This mechanism enables efficient learning based on parameter importance and update necessity. While Adam has relatively many hyperparameters, they exhibit high robustness, and their default values ($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1e - 8$) function well across a wide range of tasks.

4 Experiments

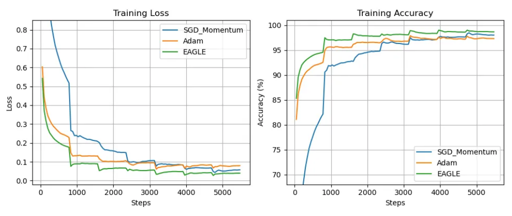
In this study, we evaluate SGD with Momentum, Adam, and our proposed EAGLE on two tasks. The first is text classification—fine-tuning GPT-2 Small (117 M parameters) on SST-2 (GLUE). The second is image classification—fine-tuning ViT-B/16 (86 M parameters) on CIFAR-10. We record convergence speed, training/testing loss and accuracy, and the usage ratio change in EAGLE update rule as evaluation metrics. Experimental setup is detailed in Appendix B.

4.1 Text Classification

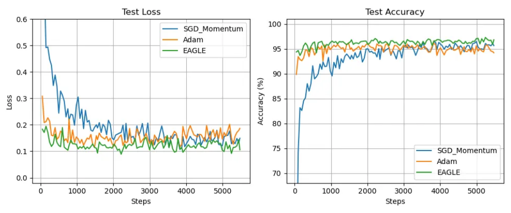
Figure 5 and Table 2, 3 summarize the fine-tuning results of GPT-2 Small on SST-2.

Convergence Speed We measure how many steps EAGLE requires to reach the final training loss achieved by each baseline optimizer. Out of a total of 29,469 steps, EAGLE reaches SGD with momentum's final loss (0.2035) in 4,313 steps—an approximately 6.83× speedup. Similarly, it reaches Adam's final loss (0.1863) in 4,355 steps—an approximately 6.77× speedup. These results demonstrate that EAGLE substantially accelerates loss convergence on the SST-2 text classification task, fulfilling the primary objective of this study.

Final Performance For final training loss, EAGLE achieved values 98.06% and 96.94% lower than SGD with momentum and Adam, respectively. For final training accuracy, EAGLE outperformed them by 7.83% and 6.66%. For final test accuracy, EAGLE outperformed them by 2.29% and 11.12%. Furthermore, EAGLE consistently outperformed the other optimizers in test accuracy throughout most of the training process. These results indicate that EAGLE not only achieves stable learning, fulfilling the primary objective of this study, but also contributes to improving final model performance.



(a) Training Loss/Accuracy



(b) Test Loss/Accuracy

Figure 6: Results on CIFAR-10 (ViT-B/16) Image Classification

Table 4: Optimizer Training/Test Loss and Accuracy on ViT-B/16 (CIFAR-10) Task

Optimizer	Train Loss	Train Acc (%)	Test Loss	Test Acc (%)
SGD_Momentum	0.0574	98.03	0.1493	95.67
Adam	0.0797	97.33	0.1865	94.22
EAGLE	**0.0396**	**98.68**	**0.1059**	**96.84**

Table 5: EAGLE Update Usage Analysis on ViT-B/16 (CIFAR-10) Task

Optimizer	Avg Usage (%)	Min Usage (%)	Max Usage (%)
EAGLE	9.54	7.62	14.78

EAGLE Update Usage Ratio The EAGLE update usage ratio shows a roughly monotonic decrease as training progresses (Appendix C). This indicates that the EAGLE update rule is actively applied in the high-loss, high-gradient regime of early training, then transitions to the more stable Adam update rule as convergence approaches. Therefore, we can conclude that the adaptive branching mechanism functions effectively, successfully balancing efficient exploration and stable convergence.

4.2 Image Classification

Figure 6 and Table 4, 5summarize the fine-tuning results of ViT-B/16 on the CIFAR-10.

Convergence Speed Out of a total of 5,473 training steps, EAGLE reaches SGD with momentum's final loss (0.0574) in 1,607 steps—an approximately 3.41× speedup. Likewise, it reaches Adam's final loss (0.0797) in 829 steps—an approximately 6.60× speedup. These results demonstrate that EAGLE successfully accelerates loss convergence on the image classification task as well, fulfilling our primary objective.

Final Performance For final training loss, EAGLE achieved values 31.01% and 50.31% lower than SGD with momentum and Adam, respectively. For final training accuracy, EAGLE outperformed them by 0.65% and 1.35%. For final test accuracy, EAGLE exceeded them by 1.17% and 2.62%. Notably, EAGLE already reaches the baselines' final test accuracy levels in the early phase and then consistently outperforms them thereafter. These results indicate that EAGLE not only achieves stable learning but also contributes to improving final model performance.

EAGLE Update Usage Ratio While the EAGLE update rule usage ratio is generally lower compared to the text classification task (Appendix C), considering its strong convergence speed and final performance, it remains effective in the image recognition domain.

Based on these results, the proposed EAGLE optimizer outperforms the representative baselines—SGD with Momentum and Adam—on both GPT-based text classification and ViT-based image classification in terms of convergence speed and final performance, thereby achieving the objectives of this study.

Table 6: Time Efficiency (Seconds per Epoch)

Optimizer	GPT Model		ViT Model	
	Python	PyTorch	Python	PyTorch
SGD_Momentum	160.39s	110.70s	191.04s	19.80s
Adam	361.06s	110.19s	348.53s	22.05s
EAGLE	897.06s	-	447.33s	-

5 Limitations

While EAGLE demonstrates impressive convergence acceleration and performance improvements in our experiments, we acknowledge several limitations of our current approach.

Is Adam Really The Best Choice? Our EAGLE algorithm was implemented with an adaptive switching mechanism that by default uses Adam as the backup optimizer (due to its known robustness). We also implemented a variant using SGD with momentum as the fallback; however, the Adam-based configuration consistently yielded better training outcomes (Appendix D).

Tradeoffs In Steps vs Wall-Clock Time In our pure-Python implementation, EAGLE requires approximately 1.29–5.59× more wall-clock time per epoch compared to SGD with momentum or Adam (Tab 6). Furthermore, because the PyTorch/C++ backend for EAGLE has not yet been developed, the gap widens when comparing against PyTorch-optimized SGD with momentum and Adam. Consequently, the per-step runtime overhead negates EAGLE's advantage in step-based convergence speed.

6 Conclusion and Future works

In this paper, we proposed EAGLE, an optimizer that combines lightweight curvature-aware update rule with an adaptive switching mechanism to Adam. In experiments on both an NLP task (fine-tuning GPT-2 on SST-2) and a vision task (fine-tuning ViT-B/16 on CIFAR-10), EAGLE converged faster than standard optimizers – roughly 6.83× faster than SGD with momentum and 6.77× faster than Adam on SST-2, and 3.41× faster than SGD and 6.60× faster than Adam on CIFAR-10– while also achieving better final accuracy in both cases. These results indicate that EAGLE effectively accelerates loss convergence and often leads to higher generalization performance. In summary, EAGLE leverages simple curvature estimates from gradient differences to take large early steps, then smoothly falls back to Adam when needed for stability, successfully combining fast exploration with stable convergence.

Future work will pursue several directions to make EAGLE more practical and broadly applicable:

1. **Optimized implementation:** Develop a PyTorch/C++ (GPU) version of EAGLE to eliminate the current Python bottleneck and achieve per-step speed comparable to other optimizers.
2. **Broader applications:** Apply EAGLE to other domains such as generative modeling (e.g. GANs or diffusion models) and reinforcement learning, to test its benefits beyond supervised tasks.
3. **Theoretical analysis:** Investigate the convergence properties and switching behavior of EAGLE formally, to better understand when and why the adaptive strategy yields gains.

These directions aim to address the current limitations and unlock the full potential of curvature-aware adaptive optimization for future deep learning applications.

References

[1] Rohan V Kashyap. A survey of deep learning optimizers - first and second order methods. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 37(4):1–24, 2023.

[2] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

[3] James Martens. Deep learning via Hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 735–742. ICML, 2010.

[4] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, NY, 2 edition, 2006.

[5] Rohan Anil, Vineet Gupta, Tomer Koren, Kevin Regan, and Yoram Singer. Second order optimization made practical. *arXiv preprint arXiv:2002.09018*, 2020.

[6] Donald Goldfarb, Yi Ren, and Achraf Bahamou. Practical quasi-newton methods for training deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25:1097–1105, 2012.

[8] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, 27, 2014.

[9] Herbert Robbins and Sutton Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[10] Ali Ramezani-Kebrya, Kimon Antonakopoulos, Volkan Cevher, Ashish Khisti, and Ben Liang. On the generalization of stochastic gradient descent with momentum. *Journal of Machine Learning Research*, 25:1–56, 2024.

[11] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

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

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, volume 1, pages 4171–4186, 2019.

[15] Matthew D. Zeiler. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[16] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, volume 28 of *Proceedings of Machine Learning Research*, pages 1139–1147. PMLR, 2013.

[17] Geoffrey Hinton. Neural networks for machine learning, lecture 6a: Overview of mini-batch gradient descent. Coursera Lecture Slides, 2012. Slide 29 of Lecture 6.

Appendix

A Effect of EAGLE's Learning-Rate Adaptation

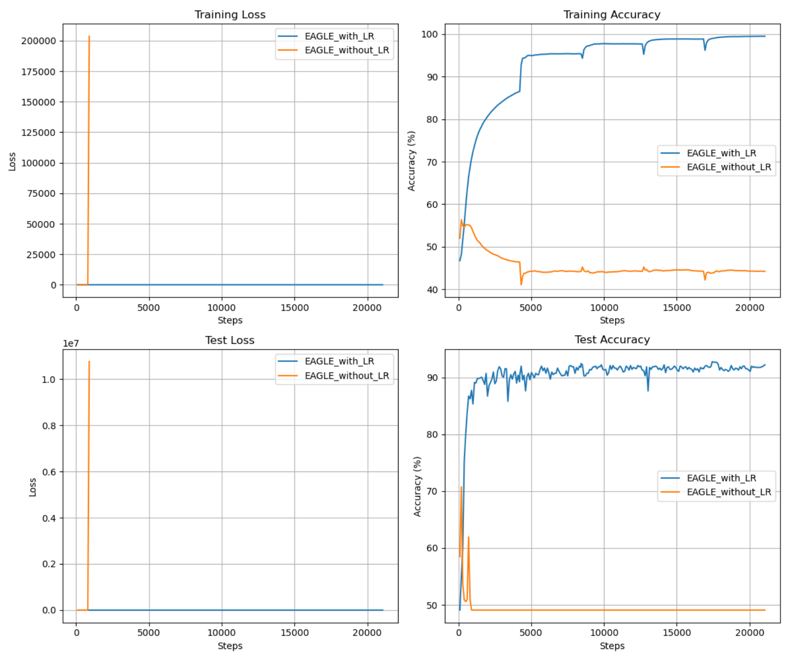


Figure 7: GPT-2 on SST-2

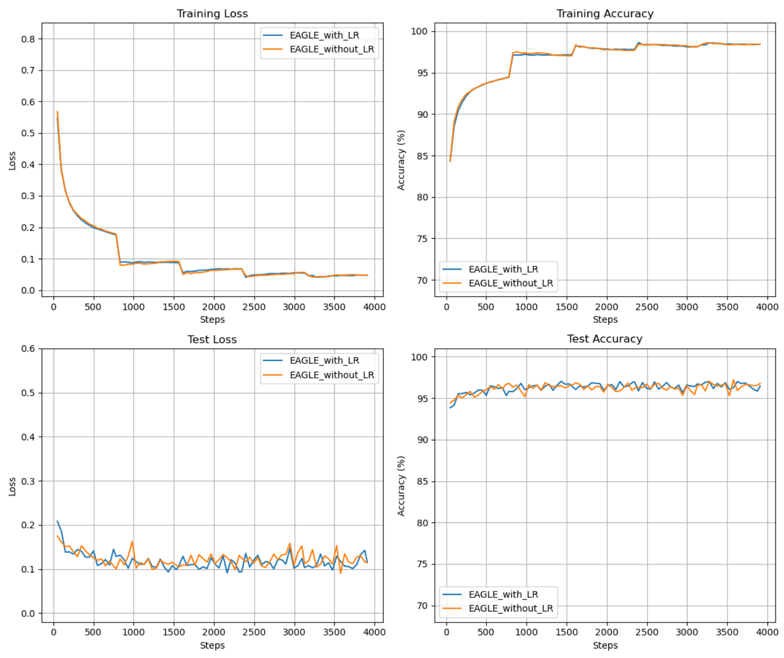


Figure 8: ViT-B/16 on CIFAR-10

B Experimental Setup

All experiments were conducted on a single NVIDIA A100 80 GB PCIe GPU.

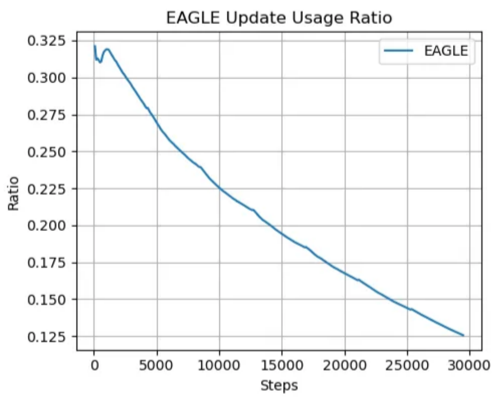
Table 7: Experimental settings for evaluating optimization algorithms across different tasks.

Task	**Hyperparameters**		
	SGD w/ Momentum	**Adam**	**EAGLE (Ours)**

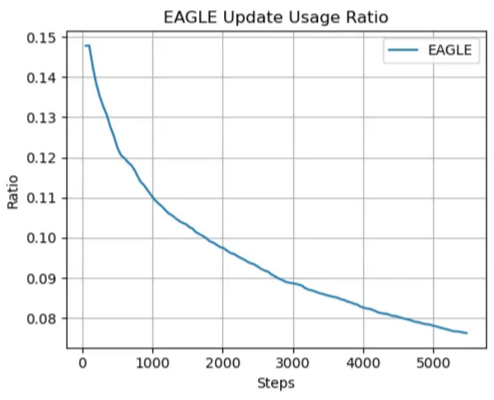
Text Classification	lr = 5e-5
	momentum = 0.9
	weight decay = 1e-2
	batch size = 16
		lr = 5e-5
		$\beta_1 = 0.9$, $\beta_2 = 0.999$
		weight decay = 1e-2
		batch size = 16
			lr = 5e-5
			threshold = 1e-4
			weight decay = 1e-2
			batch size = 16

Image Classification	lr = 1e-2
	momentum = 0.9
	weight decay = 1e-4
	batch size = 64
		lr = 1e-4
		$\beta_1 = 0.9$, $\beta_2 = 0.999$
		weight decay = 1e-4
		batch size = 64
			lr = 1e-4
			threshold = 5e-4
			weight decay = 1e-4
			batch size = 64

C Detailed Changes of EAGLE Update Rule Usage Ratio



(a) GPT-2 on SST-2



(b) ViT-B/16 on CIFAR-10

Figure 9: EAGLE Update Rule Usage Ratio

D Adam-based EAGLE (EAGLE, EAGLE-A) vs SGD with Momentum-based EAGLE (EAGLE-S)

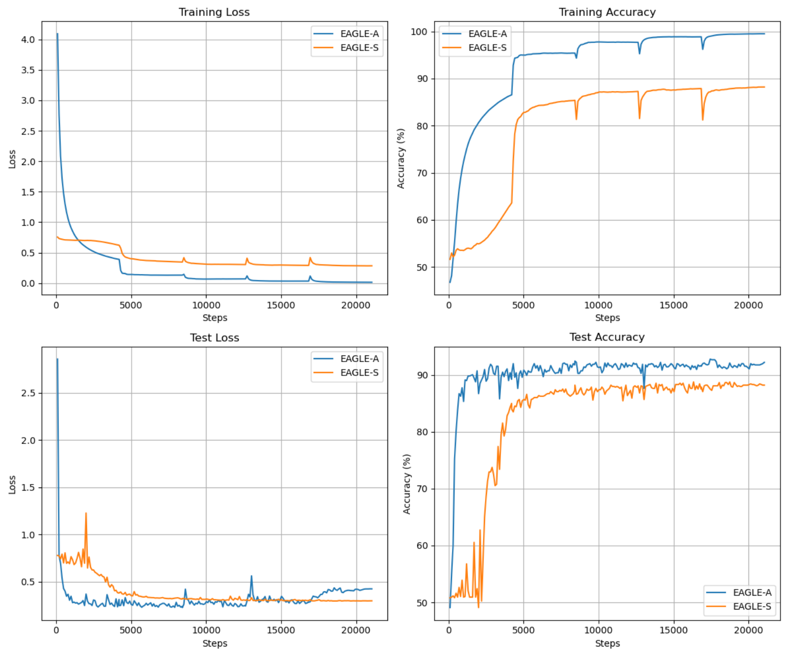


Figure 10: GPT-2 on SST-2

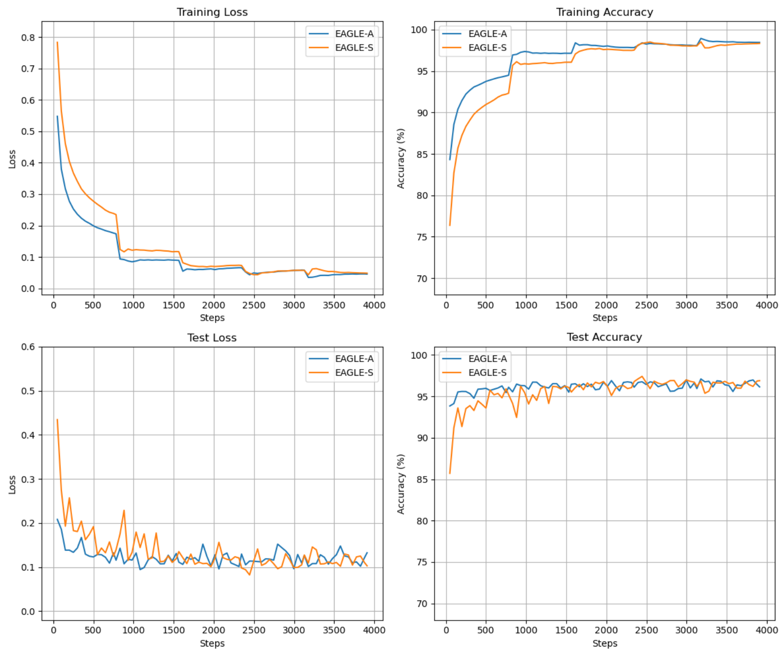


Figure 11: ViT-B/16 on CIFAR-10

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The paper explicitly claims that EAGLE accelerates convergence and improves performance, and the experiments confirm this. For example, the paper reports that "EAGLE converged faster than standard optimizers – roughly 6.83× faster than SGD… and 3.41× faster than SGD… on CIFAR-10 – while achieving better final accuracy in both cases". This matches the contributions stated in the abstract and introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: The paper includes a dedicated Limitations section (Section 5). It explicitly discusses the choice of backup optimizer and the runtime overhead of the current Python implementation.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The paper is purely empirical and not contains substantial formal theorems or proofs. All findings are demonstrated via experiments rather than theoretical analysis.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper provides detailed experimental setup in Appendix B and in-text. It specifies the models and datasets used ("fine-tuning GPT-2 Small... on SST-2" and "ViT-B/16 on CIFAR-10") and includes hyperparameters.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We publicly release our code on GitHub. We also use public benchmark datasets (SST-2 and CIFAR-10) for all experiments, and provide instructions in the supplemental material and repository.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.

- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All training and testing details are specified. Section 4 and Appendix B list model sizes, dataset splits, and hyperparameters. The supplemental table also includes full experimental settings for each task.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [NA]

   Justification: The paper reports single-run results and does not present error bars, confidence intervals, or results over multiple trials. For example, Tables 2 and 3 list only point estimates (train/test losses and accuracies) with no variance measures

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: In Appendix B.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: The research follows standard practices (using public data and models, no human subjects). There is no indication of ethical issues; all aspects conform to NeurIPS ethics guidelines.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: The paper does not discuss broader societal impacts, and the work (improving an optimizer) has no direct controversial impact.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The work uses existing pretrained models (GPT-2 and ViT) and does not release new high-risk generative models or datasets that would require additional safeguards.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No]

    Justification: The paper uses public datasets (implicitly covered by known licenses) and openly uses pretrained GPT-2 and ViT models. However, it does not explicitly cite the creators of GPT-2 (OpenAI) alongside mention of using it (whereas ViT is cited). Thus the original authors of GPT-2 are not credited in the text.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

Justification: We introduce their new optimization code and provide it via a public repository. The supplementary materials (Appendix B/Table 7) document how to use these resources (e.g. hyperparameters, tasks) alongside the assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: No crowdsourced data collection or human-subject experiments are involved.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: There are no study participants, so this question is not applicable.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The mention of GPT-2 is as a pre-trained model for an NLP task, not as a subject of study or deployment requiring special discussion. The paper does not involve novel use of LLMs beyond standard fine-tuning.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.