Toward Understanding Why Adam Converges Faster Than SGD for Transformers

Yan Pan YPAN2@ANDREW.CMU.EDU
Yuanzhi Li YUANZHIL@ANDREW.CMU.EDU
Carnegie Mellon University

Abstract

While stochastic gradient descent (SGD) is still the most popular optimization algorithm in deep learning, adaptive algorithms such as Adam have established empirical advantages over SGD in some deep learning applications such as training transformers. However, it remains a question why Adam converges significantly faster than SGD in these scenarios. In this paper, we explore one explanation of why Adam converges faster than SGD using a new concept *directional sharpness*. We argue that the performance of optimization algorithms is closely related to the directional sharpness of the update steps, and show SGD has much worse directional sharpness compared to adaptive algorithms. We further observe that only a small fraction of the coordinates causes the bad sharpness and slow convergence of SGD, and propose to use coordinate-wise clipping as a solution to SGD and other optimization algorithms. We demonstrate the effect of coordinate-wise clipping in sharpness reduction and speeding up the convergence of optimization algorithms under various settings, and conclude that the sharpness reduction effect of adaptive coordinate-wise scaling is the reason for Adam's success in practice.

1. Introduction

Stochastic gradient descent (SGD) [3, 26] is one of the most popular optimization algorithms for deep learning. Although SGD is efficient on various large-scale neural networks, in many tasks, such as traning transformers [10, 31], people seek to use the adaptive variants of stochastic gradient methods. Adaptive algorithms, such as Adagrad [11], Adam [18], and AMSGrad [25], can find a "better coordinate-wise scaling" of the gradient, so the size of the update step is adaptive to the local geometry of the function. While adaptive algorithms can converge much faster than SGD in many applications [12, 16, 34], the understanding of the superior performance of Adam-type optimizers in these tasks is limited [7, 34].

In this paper, we explore one explanation of why Adam converges faster than SGD in practice, especially for transformers. We decompose the goal of minimizing an objective function into two parts: gradient correlation and directional sharpness. We argue that the directional sharpness of the update direction is a useful indicator of the performance of optimization algorithms, that high sharpness usually implies low performance, and is more important than the gradient correlation in optimization. We observe through experiments that the update directions of SGD have much higher directional sharpness compared to adaptive algorithms, which explains why SGD converges so slowly. Furthermore, we propose that the imbalanced distribution of gradient across coordinates is the key contributor to SGD's high directional sharpness. We observe that only a small fraction of the parameters contribute to most of SGD's directional sharpness. Hence, we propose to use coordinate-wise clipping as a solution to the problem of slow convergence and bad directional sharpness. We show that clipping can improve the directional sharpness and convergence rate of various non-coordinate-wise-scaling optimization algorithms, and propose that coordinate-wise clipping can be used as a generic component in optimization algorithms. We demonstrate our findings through two experiments under various settings and show that our observations are consistent across different tasks, models, and iterations. We conclude that the adaptive coordinate-wise scaling of Adam can effectively find a balance between optimizing gradient correlation and directional sharpness, and such ability is the key to Adam's fast convergence in deep learning training.

2. Related Work

General Convergence Rates of Adaptive Algorithms. Adaptive algorithms have long been studied and applied in deep learning [1, 11, 18, 23, 25, 30]. Several previous work has proved convex and non-convex convergence rates for Adagrad [9, 11, 20, 32] and Adam or AMSGrad [4, 8, 13, 22, 25, 35, 36]. The best known non-convex convergence rate for Adagrad is $O(\frac{\log T}{\sqrt{T}})$ [9, 20] and $O(\frac{1}{\sqrt{T}})$ for AMSGrad [35]. While the result by Zhou et al. [35] matches the non-convex convergence rate $O(\frac{1}{\sqrt{T}})$ of SGD [15], there is no theoretical proof that Adam can converge asymptotically faster than SGD for general functions [7]. Therefore, there is still a significant gap of work between the theoretical understanding of Adam and its empirical fast performance.

Faster Convergence Rates Under Certain Settings. Another line of work focused on specific settings that Adam might work better than SGD. Adaptive algorithms can work asymptotically better when the stochastic gradients are sparse [11, 35] or when there is a sparse set of noise [2]. Zhang et al. [34] proved that global clipping methods outperforms SGD when the stochastic gradients have heavy-tail noise, argued that Adam can also deal with heavy-tail noise effectively, and designed a new algorithm based on coordinate-wise clipping. The effect of global clipping and normalization methods were also studied in [17, 19]. Our work is inspired by the use of coordinate-wise clipping in algorithm design in [34], but we propose different explanations of the effectiveness of coordinate-wise clipping with new empirical evidence.

3. Directional Sharpness of Adaptive Algorithms

In this section, we introduce a new measurement *directional sharpness* that indicates the performance of optimization algorithms. We show that minimizing the term is extremely important to fast convergence of optimization algorithms and argue that it is closely related to the slow convergence of SGD.

In convex and non-convex optimization, a typical proof strategy is to consider the quadratic Taylor expansion of the objective function

$$f(x_{t+1}) \approx f(x_t) + \underbrace{\nabla f(x_t)^\top (x_t - x_{t+1})}_{\text{gradient correlation}} + \underbrace{(x_t - x_{t+1})^\top \nabla^2 f(x_t)(x_t - x_{t+1})}_{\text{directional sharpness}} \tag{1}$$

where η is the step size. In order for $f(x_{t+1})$ to decrease monotonically, the optimization algorithm should minimize the two terms that depends on the update step. To bound the second-order term, a standard strategy is to use the property of the spectral norm of the Hessian

$$(x_t - x_{t+1})^\top \nabla^2 f(x_t)(x_t - x_{t+1}) \leq \|\nabla^2 f(x_t)\|_2 \|x_t - x_{t+1}\|_2^2. \tag{2}$$

The local Hessian spectral norm is often called the *sharpness* of the function in deep learning [5]. If the function is L-smooth, such that $\|\nabla^2 f(x_t)\|_2 \leq L$ for some constant L, the second-order term is upper bounded by $L\|x_t - x_{t+1}\|_2^2$, so the loss can decrease when $\|x_t - x_{t+1}\|_2$ is sufficiently small. However, there are disadvantages of the smoothness assumption in theoretical proofs. The Hessian can adapt to the geometry of the trajectory and can vary significantly for different algorithms [5, 6]. Furthermore, even if the local geometry and Hessian are fixed, the update direction $x_{t+1} - x_t$ is also extremely important to minimizing the second-order term.

Motivated by the definition of sharpness and the above observations, we define the *directional sharpness* of a function f at x in the direction $v \in \mathbb{R}^d, \|v\|_2 = 1$ as $v^\top \nabla^2 f(x) v$. Although our definition is motivated by the sharpness definition in deep learning, there are important differences between directional sharpness and the general sharpness in deep learning optimization. The general sharpness definition describes the worst-case sharpness and is the supremum of directional sharpness over all directions. However, directional sharpness consider the sharpness in the specific update direction of an optimization algorithm, and can be much lower than the sharpness. Furthermore, the concept of sharpness is typically associated with the landscape and generalization of neural networks as in various works such as Sharpness-Aware Minimization [14] and Edge of Stability [5, 6]. However, we are only interested in optimizing the objective function, or the loss on the training set.

The directional sharpness at x_t in the update direction is extremely important to minimizing $f(x_{t+1})$. When the gradient correlation is similar, the loss $f(x_{t+1})$ directly depends on the directional sharpness at x_t. Furthermore, since directional sharpness is quadratic in the step size η and gradient correlation is linear, if we consider Equation (1) as a quadratic function of η, a lower directional sharpness implies the potential to take a larger step size and possibly lead to a larger local decrement of the objective function. This implies that having a low directional sharpness is a more desirable property for update directions than having a high gradient correlation.

Empirically, we observe that there can be a significant gap between the directional sharpness of different optimization algorithms. In particular, the directional sharpness is much lower for adaptive algorithms than for SGD. We argue that minimizing the directional sharpness is more important for fast convergence of optimization algorithms as compared to minimizing the gradient correlation. We study the update step of different optimization algorithms under the same trajectory and local geometry using pseudo-update steps described in Appendix A, in order to rule out the impact of trajectory. We compute the directional sharpness of different optimization algorithms and visualize the optimization landscape in the update direction of a variety of optimization algorithms in Figure 3. The observation is consistent with our analysis. The directional sharpness of different optimization algorithms varies significantly. For example, the directional sharpness of SGD can be more than 10^7 times the directional sharpness of Adafactor as shown in Figure 2. The update step of SGD has the best correlation with the actual gradient, so the loss decrease faster when the step size is very small, since in this case the linear term dominates the quadratic term in Equation (1). However, because of the large directional sharpness, when the step size increases the quadratic term grows faster than the linear term, so the loss reaches the local minima in the direction after a very small step size. For adaptive algorithms, the directional sharpness is much lower than SGD, so they have the potential to use a much larger step size and the optimal step could give a much lower loss compared to SGD.

In order to explain the sharpness reduction effect of adaptive algorithms, since the strategy for adaptive algorithms is to find a coordinate-wise scaling of the gradient, we investigate the distribution of gradient norm across different coordinates. We visualize a histogram of the absolute value

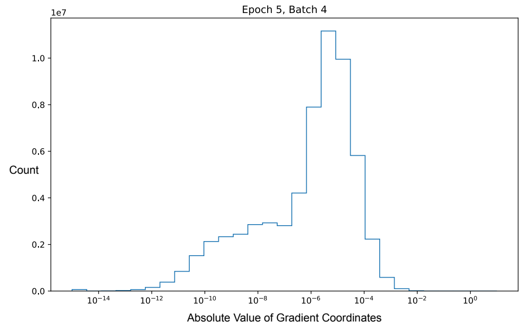


Figure 1: Histogram of momentum distribution for stochastic gradient descent on machine translation.

Algorithm	Sharpness
Adam	0.16190993
SGD	31.04433435
SGD Clipping	1.77876506
Normalized SGD	0.77112307
Normalized SGD Clipping	0.38075357
Adafactor	3.1928×10^{-6}
Adafactor Clipping	2.5258×10^{-6}

Figure 2: The average sharpness of different optimization algorithms when trained on machine translation, in the same experiment and epoch as Figure 3.

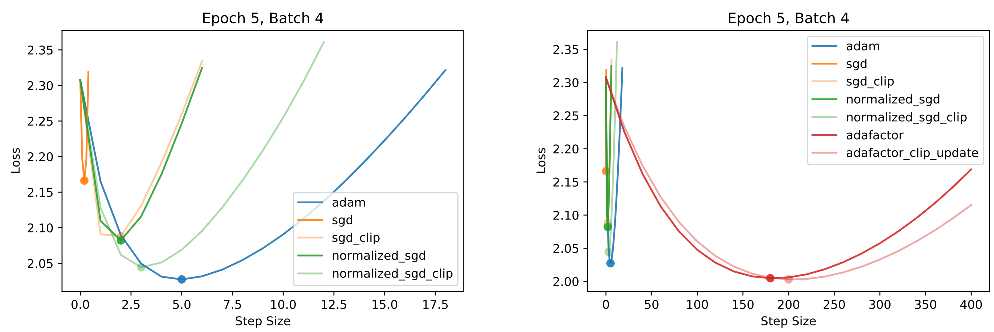


Figure 3: The loss landscape in different update directions on machine translation. The step size is the learning rate normalized by the update step ℓ_2 norm.

of SGD momentum coordinates in Figure 1. We observe that the gradients are distributed unevenly across the coordinates, with half of the coordinates have absolute value ranging from 10^{-12} to 10^{-6}, but also exists an innegligible portion of coordinates that can be as high as 10^{-4} to 10^{-2}, contributing to most of the gradient norm. The histogram suggests that the gradients are concentrated on a small fraction of the coordinates, and this small fraction of coordinates can contribute to a large portion of sharpness, making optimization hard. For adaptive algorithms and normalized optimization algorithms, since they already used some forms of scaling, the imbalanced gradient distribution will not be as severe as SGD, but normalizing the large coordinates might still be beneficial.

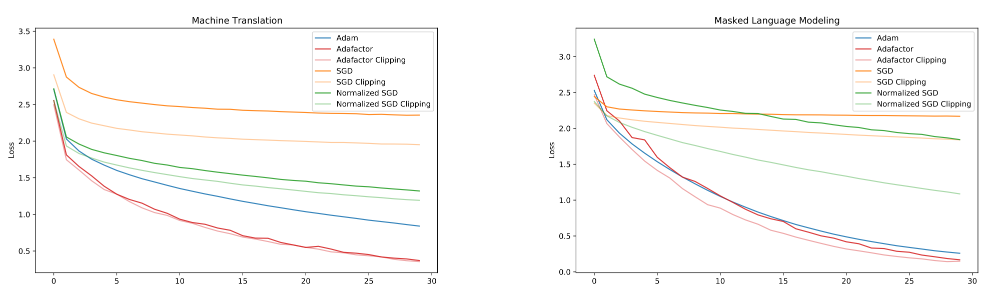
4. Coordinate-wise Clipping

In this section, we propose to use *coordinate-wise clipping* as a solution to the aforementioned imbalanced distribution of gradient based on our experimental findings. We observe that the sharpness is also concentrated in the large coordinates in the gradient, and clipping those coordinates can significantly decrease directional sharpness. Although clipping can decrease the correlation between the update step and the true gradient, since the dependence on the clipped entry is quadratic for the second-order term and linear for the first-order term, it might not be beneficial to use these coordinates. The use of clipping in optimization algorithms is a trade-off between improving gradient correlation and reducing directional sharpness. By clipping the top coordinates in the gradient, although gradient correlation decreases, the directional sharpness can decrease even more to make up the loss.

We consider using clipping on a variety of non-coordinate-wise-scaling algorithms, including SGD, normalized SGD, and Adafactor [28]. We demonstrate that coordinate-wise clipping significantly reduces the sharpness of adaptive algorithms and speeds up the optimization process. Specifically, we compute the threshold τ for the top $k\%$ gradients in terms of the absolute value, and clip the gradient coordinates g_i to $\tilde{g}_i = \operatorname{sgn}(g_i)\min\{|g_i|, \tau\}$ based on their sign. In practice, it is possible to simplify the procedure by setting a fixed threshold. From Figure 2, we can see that by clipping the top 1% coordinates, the directional sharpness decrease significantly. Since we normalize the update step when we compute the directional sharpness, the sharpness reduction effect of coordinate-wise clipping is not due to significant reduction of the norm of the update step, but the improved flatness of the direction. Figure 3 gives a coherent message, that clipped algorithms can find a direction that has larger maximal decrement of the loss in the local geometry.

Finally we demonstrate that clipping algorithms can converge faster than the original counterpart by directly training transformers with the clipping algorithms, with the loss curve shown in Figure 4. According to the result, clipping algorithms can speedup training significantly. Our result suggests that clipping can be used as an generic building block in any non-coordinate-wise-scaling algorithms and speed up training. The new finding can provide insight into designing new optimization algorithms.

Based on our experimental findings, we conjecture that there is a positive correlation between the magnitude of Hessian coordinates and gradient coordinates. The positive correlations is also mentioned in [33], but their proposed correlation is between the norm of Hessian and norm of gradient. We further suggest that there is a positive correlation between the *coordinates* of gradient and Hessian, and the success of Adam is due to the ability to scale down the bad coordinates and reduce the sharpness through coordinate-wise scaling of the gradient. Understanding of this phenomenon could be essential in proving convergence rates for Adam that are faster than SGD.



(*a*) Machine Translation

(*b*) Masked Language Modeling

Figure 4: Clipped optimization algorithms generally converge faster than the original algorithms.

5. Experiments

In Appendices A and B, we demonstrate our findings with two types of experiments. We explore several different tasks and settings and show our results hold in various setting, including training t5 [24] architecture on machine translation datasets and DistilRoBERTa [27] on masked language modeling datasets. We compute the directional sharpness of a variety of optimization algorithms, including SGD, normalized SGD, and Adafactor [28], and visualize the corresponding loss landscape direction, under different local geometry. We show that SGD has bad sharpness under all of the settings, regardless of the task, model, or local geometry. In addition, we demonstrate clipping can always improve the directional sharpness of optimization algorithms, and often result in better local decrement of loss function. We also implement clipping algorithms and use them to train different models, and demonstrate that clipping algorithms converge faster in practice.

6. Conclusion

In summary, our work provides a new insight of why Adam converges faster than SGD in practice. In contrast to assumptions on properties of the gradient, we propose to study directional sharpness as an important indicator for the performance of optimization algorithms in deep learning. We show that adaptive algorithms and clipped optimization algorithms can generally achieve significantly better directional sharpness compared to SGD. We argue that the slow convergence of SGD is related to the high directional sharpness, caused by a positive coordinate-wise gradient-Hessian correlation. We propose to use coordinate-wise clipping as a solution to the problem of high sharpness. We demonstrate the sharpness reduction effect of coordinate-wise clipping and show that it is possible to step into a lower loss in the update direction of clipping algorithms compared to the original algorithms. We further demonstrate the effectiveness of coordinate-wise clipping in a wide range of optimization algorithms without coordinate-wise scaling, including SGD, normalized SGD, and Adafactor. We suggest the use of coordinate-wise clipping as a generic building block in non-convex optimization algorithms. Our work provide useful explanations and conjectures about the superior performance of Adam and further understanding of the results could be useful in theoretical understanding of the empirical advantage of Adam over SGD.

References

[1] Larry Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, 16(1):1–3, 1966.

[2] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signSGD: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pages 560–569. PMLR, 2018.

[3] Léon Bottou et al. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91(8):12, 1991.

[4] Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong. On the convergence of a class of Adam-type algorithms for non-convex optimization. In *International Conference on Learning Representations*, 2019.

[5] Jeremy Cohen, Simran Kaur, Yuanzhi Li, J Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. In *International Conference on Learning Representations*, 2021.

[6] Jeremy M Cohen, Behrooz Ghorbani, Shankar Krishnan, Naman Agarwal, Sourabh Medapati, Michal Badura, Daniel Suo, David Cardoze, Zachary Nado, George E Dahl, et al. Adaptive gradient methods at the edge of stability. *arXiv preprint arXiv:2207.14484*, 2022.

[7] Marina Danilova, Pavel Dvurechensky, Alexander Gasnikov, Eduard Gorbunov, Sergey Guminov, Dmitry Kamzolov, and Innokentiy Shibaev. Recent theoretical advances in non-convex optimization. In *High-Dimensional Optimization and Probability*, pages 79–163. Springer, 2022.

[8] Soham De, Anirbit Mukherjee, and Enayat Ullah. Convergence guarantees for RMSProp and Adam in non-convex optimization and an empirical comparison to nesterov acceleration. *arXiv preprint arXiv:1807.06766*, 2018.

[9] Alexandre Défossez, Léon Bottou, Francis Bach, and Nicolas Usunier. A simple convergence proof of Adam and Adagrad. *arXiv preprint arXiv:2003.02395*, 2020.

[10] 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, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[11] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[12] John Duchi, Michael I Jordan, and Brendan McMahan. Estimation, optimization, and parallelism when data is sparse. *Advances in Neural Information Processing Systems*, 26, 2013.

[13] Biyi Fang and Diego Klabjan. Convergence analyses of online Adam algorithm in convex setting and two-layer ReLU neural network. *arXiv preprint arXiv:1905.09356*, 2019.

[14] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021.

[15] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

[16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[17] Elad Hazan, Kfir Levy, and Shai Shalev-Shwartz. Beyond convexity: Stochastic quasi-convex optimization. *Advances in neural information processing systems*, 28, 2015.

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

[19] Kfir Y Levy. The power of normalization: Faster evasion of saddle points. *arXiv preprint arXiv:1611.04831*, 2016.

[20] Xiaoyu Li and Francesco Orabona. On the convergence of stochastic gradient descent with adaptive stepsizes. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 983–992. PMLR, 2019.

[21] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150, 2011.

[22] Tran Thi Phuong and Le Trieu Phong. On the convergence proof of AMSGrad and a new version. *arXiv preprint arXiv:1904.03590*, 2019.

[23] Boris T Polyak. Introduction to optimization. 1987. *Optimization Software, Inc, New York*.

[24] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67, 2020.

[25] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of Adam and beyond. In *International Conference on Learning Representations*, 2018.

[26] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[27] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[28] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR, 2018.

[29] Jörg Tiedemann. Parallel data, tools and interfaces in OPUS. In *Eight International Conference on Language Resources and Evaluation, MAY 21-27, 2012, Istanbul, Turkey*, pages 2214–2218, 2012.

[30] Tijmen Tieleman, Geoffrey Hinton, et al. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[32] Rachel Ward, Xiaoxia Wu, and Leon Bottou. AdaGrad stepsizes: Sharp convergence over nonconvex landscapes. In *International Conference on Machine Learning*, pages 6677–6686. PMLR, 2019.

[33] Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. Why gradient clipping accelerates training: A theoretical justification for adaptivity. In *International Conference on Learning Representations*, 2020.

[34] Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank Reddi, Sanjiv Kumar, and Suvrit Sra. Why are adaptive methods good for attention models? *Advances in Neural Information Processing Systems*, 33, 2020.

[35] Dongruo Zhou, Jinghui Chen, Yuan Cao, Yiqi Tang, Ziyan Yang, and Quanquan Gu. On the convergence of adaptive gradient methods for nonconvex optimization. *arXiv preprint arXiv:1808.05671*, 2018.

[36] Fangyu Zou and Li Shen. On the convergence of weighted adagrad with momentum for training deep neural networks. *arXiv preprint arXiv:1808.03408*, 2018.

Appendix A. Experimental Details

A.1. Tasks, Datasets, and Models

We run our experiments on two tasks, including machine translation and masked language modeling. The details of the dataset, training set size, and model we use are in Table 1. For each dataset, we select the first 10240 data as our training set. Since we're mainly interested in minimizing the training loss, we do not use any test or validation sets, nor any evaluation metrics other than the cross-entropy loss. For machine translation, we use the English to French opus books dataset [29] and t5 model [24]. For masked language modeling, we use the imdb dataset [21] and DistilRoBERTa model [27]. In order to evaluate the function in a offline setting, we generate fixed masks with probability 0.15 at the beginning of the training and does not generate new masks whenever we collate the data.

Task	**Dataset**	**Size**	**Model**
Machine Translation	opus books [29]	10240	t5-small [24]
Masked Language Modeling	imdb [21]	10240	DistilRoBERTa-base [27]

Table 1: Details of the tasks, datasets, training set sizes, and models we use for the two different experiments.

A.2. Optimization Algorithms and Clipping Methods

We use 4 optimization algorithms, including Adam, SGD, normalized SGD, and Adafactor [28]. We use momentum for all of the optimization algorithms to rule out any potential effect of momentum. The clipped optimization algorithms are described in Algorithms 1 to 4. Notice that for Adafactor, we only clip the gradient in the nominator of the final update step, since otherwise the scaling effect could cancel out or even increase the norm. Adafactor is originally used with the relative step sizes α_t, but in certain cases we use a fixed learning rate in place of α_t. In the algorithms, we assume $\mathrm{Clip}(g)$ calculates the clipping threshold τ for the top $k\%$ coordinates and returns $\tilde{g}$ where $\tilde{g}_i = \mathrm{sgn}(g_i)\min\{|g_i|, \tau\}$. We use a large value $k = 1$ in all of our experiments to better demonstrate the effectiveness of clipping. However, significant but weaker effects can also be observed by setting a very small value such as $k = 0.1$. We will include more experiments on the clipping threshold in future versions of the paper.

Algorithm 1: SGD with momentum

Data: initial point x_0, learning rate η, momentum term β
for $t \leftarrow 1, \ldots, T$ **do**
   $g_t \leftarrow \nabla f(x_t)$
   $\hat{g}_t \leftarrow \mathrm{Clip}(g_t)$
   $m_t \leftarrow \beta m_{t-1} + (1-\beta)\hat{g}_t$
   $x_t \leftarrow x_{t-1} - \eta m_t$
end

Algorithm 2: Normalized SGD with momentum for weight matrices and vectors

Data: initial point $x_0 \in \mathbb{R}^{m \times n}$, learning rate η, momentum term β
for $t \leftarrow 1, \ldots, T$ **do**
  $g_t \leftarrow \nabla f(x_t)$
  $\hat{g}_t \leftarrow \mathrm{Clip}(g_t)$
  $m_t \leftarrow \beta m_{t-1} + (1-\beta)\hat{g}_t$
  $v_t \leftarrow \frac{m_t}{\|m_t\|_2} \cdot \sqrt{mn}$
  $x_t \leftarrow x_{t-1} - \eta v_t$
end

Algorithm 3: Adafactor for weight matrices [28]

Data: initial point $x_0 \in \mathbb{R}^{m \times n}$, relative step sizes $\rho_t = \min\{10^{-2}, \frac{1}{\sqrt{t}}\}$, second moment decay $\hat{\beta}_{2t} = 1 - t^{-0.8}$, regularization constants $\epsilon_1 = 10^{-30}$ and $\epsilon_2 = 10^{-3}$, clipping threshold $d = 1$, $\mathrm{RMS}(x) := \frac{\|x\|_F}{\sqrt{mn}}$
for $t \leftarrow 1, \ldots, T$ **do**
  $\alpha_t \leftarrow \max\{\epsilon_2, \mathrm{RMS}(x_{t-1})\}\rho_t$
  $G_t \leftarrow \nabla f(x_{t-1})$
  $\hat{G}_t \leftarrow \mathrm{Clip}(G_t)$
  $R_t \leftarrow \hat{\beta}_{2t} R_{t-1} + (1-\hat{\beta}_{2t})(G_t^2 + \epsilon_1)\mathbf{1}_m$
  $C_t \leftarrow \hat{\beta}_{2t} C_{t-1} + (1-\hat{\beta}_{2t})\mathbf{1}_n^\top (G_t^2 + \epsilon_1)$
  $\hat{V}_t \leftarrow R_t C_t / \mathbf{1}_n^\top R_t$
  $U_t \leftarrow \hat{G}_t / \sqrt{\hat{V}_t}$
  $\hat{U}_t \leftarrow U_t / \max\{1, \mathrm{RMS}(U_t)/d\}$
  $x_t \leftarrow x_{t-1} - \alpha_t \hat{U}_t$
end

Algorithm 4: Adafactor for weight vectors [28]

Data: initial point $x_0 \in \mathbb{R}^{n}$, relative step sizes $\rho_t = \min\{10^{-2}, \frac{1}{\sqrt{t}}\}$, second moment decay $\hat{\beta}_{2t} = 1 - t^{-0.8}$, regularization constants $\epsilon_1 = 10^{-30}$ and $\epsilon_2 = 10^{-3}$, clipping threshold $d = 1$, $\mathrm{RMS}(x) := \frac{\|x\|_2}{\sqrt{n}}$
for $t \leftarrow 1, \ldots, T$ **do**
  $\alpha_t \leftarrow \max\{\epsilon_2, \mathrm{RMS}(x_{t-1})\}\rho_t$
  $G_t \leftarrow \nabla f(x_{t-1})$
  $\hat{G}_t \leftarrow \mathrm{Clip}(G_t)$
  $\hat{V}_t \leftarrow \hat{\beta}_{2t} \hat{V}_{t-1} + (1-\hat{\beta}_{2t})(G_t^2 + \epsilon_1)$
  $U_t = \hat{G}_t / \sqrt{\hat{V}_t}$
  $\hat{U}_t \leftarrow U_t / \max\{1, \mathrm{RMS}(U_t)/d\}$
  $x_t \leftarrow x_{t-1} - \alpha_t \hat{U}_t$
end

A.3. Experiment for Directional Sharpness of Optimization Algorithms

Pseudo-Update Step. Since all algorithms we use has momentum part, we need to compute the momentum term in a different trajectory using "pseudo-update step." Specifically, we compute the momentum term for all the optimization algorithms at time t using the past values of $x_1, \ldots, x_{t-1}$, regardless of the optimization algorithm we use to perform the actual update step. The values we computed for the algorithms were only used to visualize the landscape and compare the sharpness, but not used for training. The momentum parameters are set to the default values [18, 28].

Training Optimizer. We use different training optimizers to compare our results across different local geometry and optimization trajectory. We use SGD momentum with learning rate 10^{-3} and Adam with learning rate 10^{-4} as training optimizers. The momentum parameters are set to the default values [18].

Test Batch. Since computation on the full-batch objective function is very computationally expensive, we sample a fixed random subset of size 1024 as the test dataset at the beginning of the training, and fix it during all epochs and batches, in order to speed up the landscape visualization process. The losses in all the plots are the losses on the test batch.

Landscape Visualization. To visualize the landscape, we update the weight with the desired update step and compute the loss. Afterwards, we reset the weight back to the original value before the update, and repeat the above step with a new step size.

Directional Sharpness. We utilize PyTorch's Hessian-vector product to efficiently compute directional sharpness. We sample 5 batches from the 10 batches in the epoch. To show the effect of clipping and adaptive update steps on the sharpness compared to SGD, we calculate the mean of the ratio of sharpness versus SGD sharpness for the sampled batches in these epochs.

A.4. Experiment for Convergence of Clipped Optimization Algorithms

We demonstrate the convergence of clipped optimization algorithms using a 1% clipping threshold. We manually tune the learning rate to find the best learning rate for the experiments. The learning rate configuration of our experiment is shown in Table 2.

Task	**Algorithm**	**Learning Rate**
Machine Translation	Adam	2×10^{-3}
	Adafactor, 1% Clipping	Relative
	Adafactor	Relative
	Normalized SGD, 1% Clipping	5×10^{-4}
	Normalized SGD	5×10^{-4}
	SGD, 1% Clipping	8×10^{-1}
	SGD	1×10^{-3}
Masked Language Modeling	Adam	2×10^{-3}
	Adafactor, 1% Clipping	1×10^{-2}
	Adafactor	1×10^{-2}
	Normalized SGD, 1% Clipping	5×10^{-5}
	Normalized SGD	5×10^{-5}
	SGD, 1% Clipping	8×10^{-1}
	SGD	1×10^{-2}

Table 2: Learning rate configuration of our experiments. The relative learning rate for Adafactor is defined in Algorithms 3 and 4 and [28].

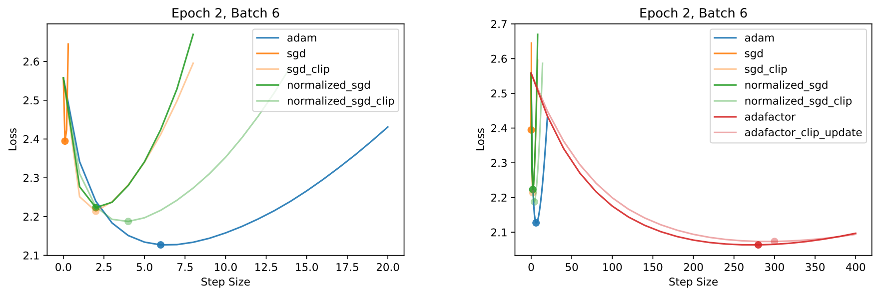
Appendix B. Directional Sharpness Results

In this section we show our experimental result for the directional sharpness of optimization algorithms. For each of the landscape visualization, we show two plots, where one of them has Adafactor and the other does not. The rest of the plots are the same with different scales. We repeat each experiment with 3 different random seeds.

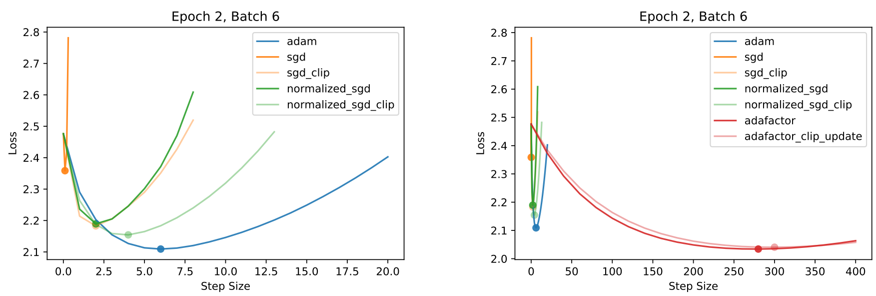
B.1. SGD Trajectory

Task	Epoch	Algorithm	Sharpness Ratio
Machine Translation	2	Adam	0.0033224481
		SGD	1.0000000000
		SGD, 1% Clipping	0.0290167637
		Normalized SGD	0.0105240090
		Normalized SGD, 1% Clipping	0.0059459802
		Adafactor	0.0000022583
		Adafactor, 1% Clipping	0.0000016335
	5	Adam	0.0040285159
		SGD	1.0000000000
		SGD, 1% Clipping	0.0408742142
		Normalized SGD	0.0171114082
		Normalized SGD, 1% Clipping	0.0073663930
		Adafactor	0.0000003899
		Adafactor, 1% Clipping	0.0000002193
	10	Adam	0.0058940997
		SGD	1.0000000000
		SGD, 1% Clipping	0.0739500316
		Normalized SGD	0.0404778099
		Normalized SGD, 1% Clipping	0.0170601111
		Adafactor	0.0000016636
		Adafactor, 1% Clipping	0.0000012989
	20	Adam	0.0116556393
		SGD	1.0000000000
		SGD, 1% Clipping	0.1343988454
		Normalized SGD	0.0765858411
		Normalized SGD, 1% Clipping	0.0320773869
		Adafactor	0.0000067047
		Adafactor, 1% Clipping	0.0000053747

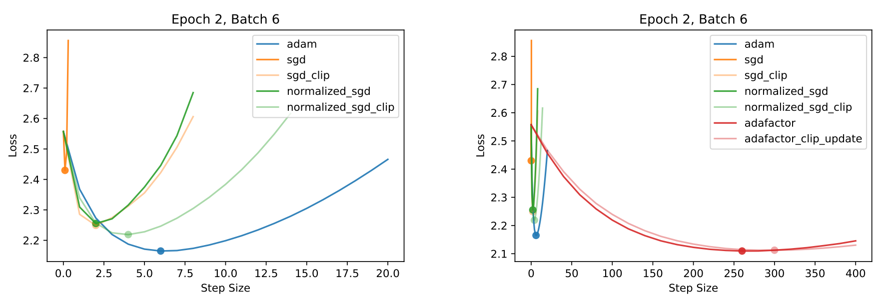
Table 3: Average ratio of directional sharpness of optimization algorithms with respect to SGD on the machine translation task in SGD trajectory.



(a) Experiment 1

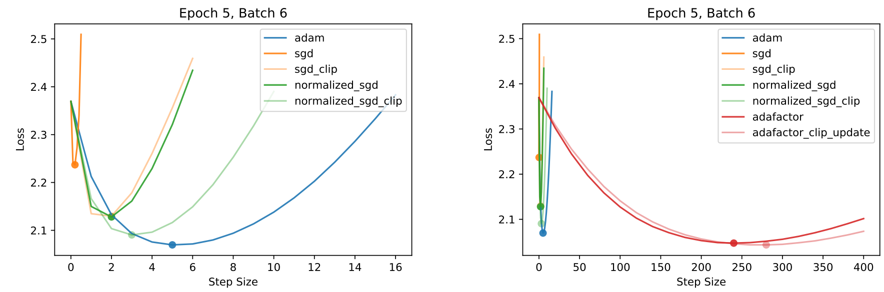


(b) Experiment 2

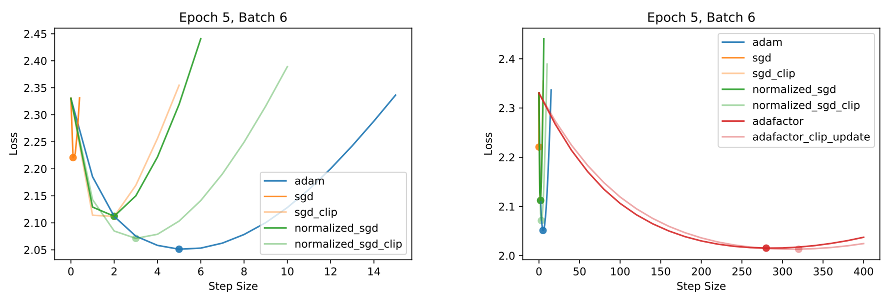


(c) Experiment 3

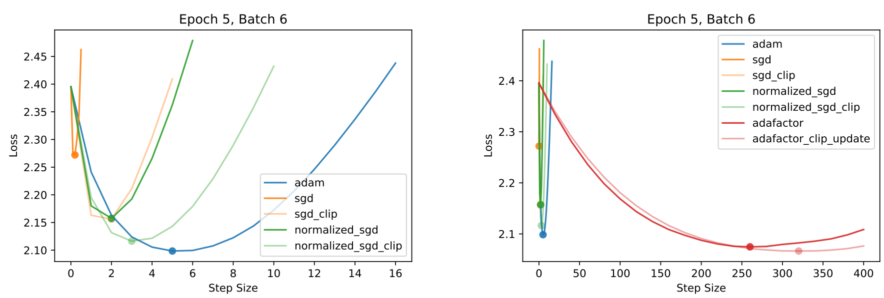
Figure 5: Landscape visualization of machine translation in SGD trajectory at Epoch 2.



(a) Experiment 1

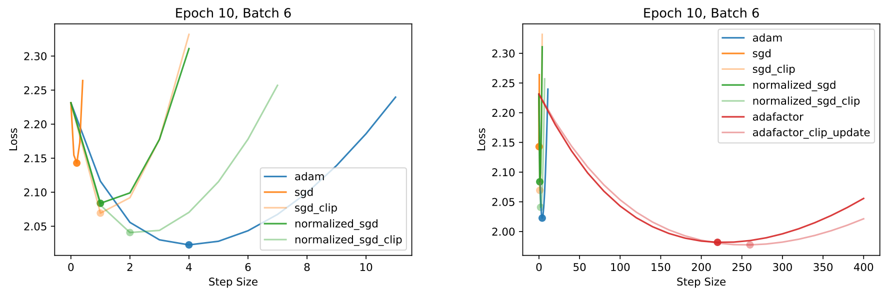


(b) Experiment 2

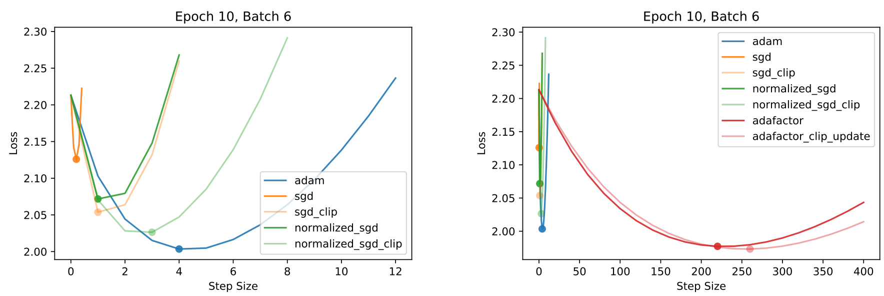


(c) Experiment 3

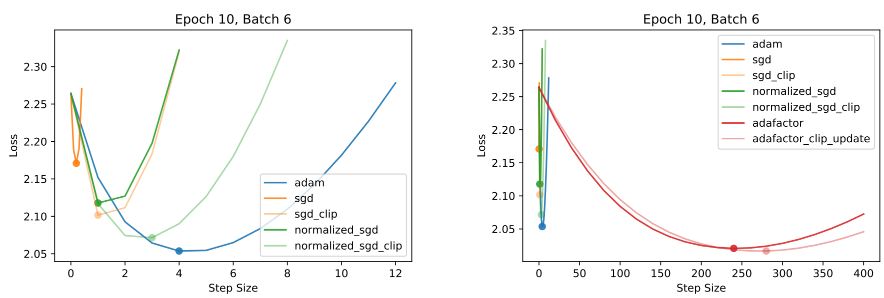
Figure 6: Landscape visualization of machine translation in SGD trajectory at Epoch 5.



(*a*) Experiment 1

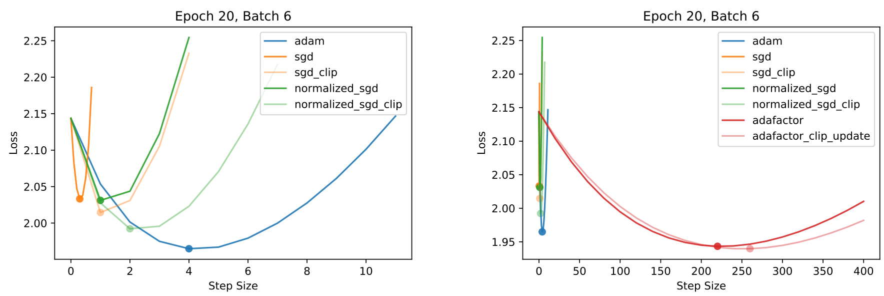


(*b*) Experiment 2

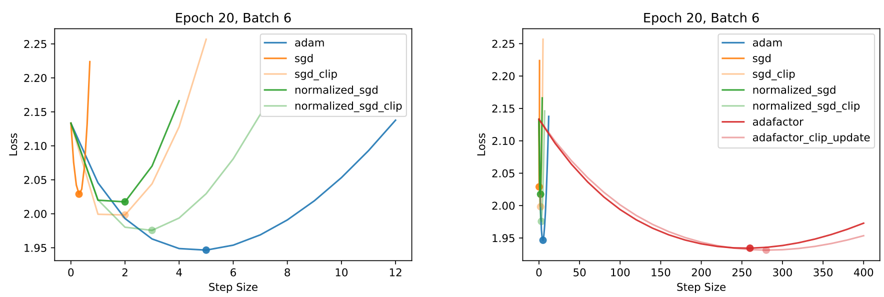


(*c*) Experiment 3

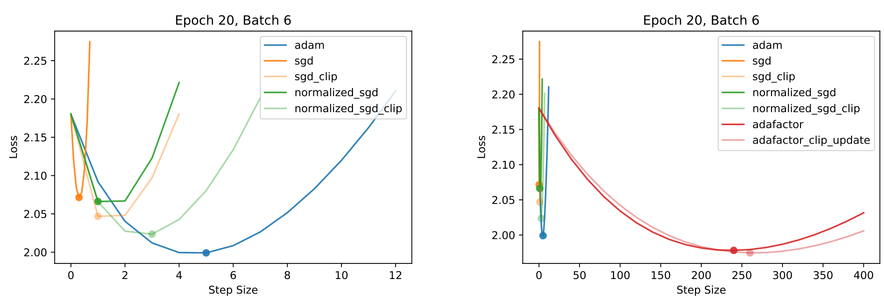
Figure 7: Landscape visualization of machine translation in SGD trajectory at Epoch 10.



(a) Experiment 1



(b) Experiment 2

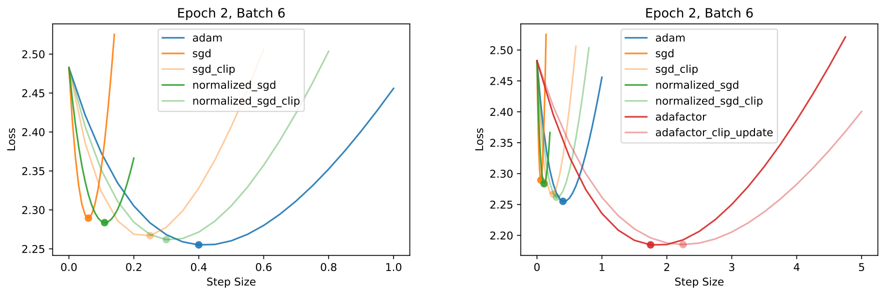


(c) Experiment 3

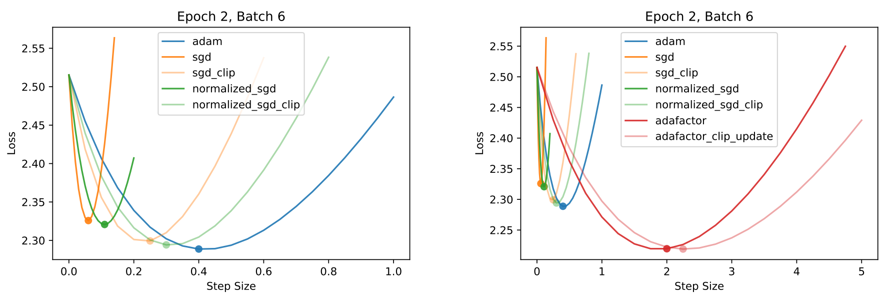
Figure 8: Landscape visualization of machine translation in SGD trajectory at Epoch 20.

Task	Epoch	Algorithm	Sharpness Ratio
Masked Language Modeling	2	Adam	0.0326442945
		SGD	1.0000000000
		SGD, 1% Clipping	0.1026522666
		Normalized SGD	0.2829808263
		Normalized SGD, 1% Clipping	0.0494325193
		Adafactor	0.0026639386
		Adafactor, 1% Clipping	0.0018618562
	5	Adam	0.0210801368
		SGD	1.0000000000
		SGD, 1% Clipping	0.1070835966
		Normalized SGD	0.2701119184
		Normalized SGD, 1% Clipping	0.0500896616
		Adafactor	0.0018551471
		Adafactor, 1% Clipping	0.0011664041
	10	Adam	0.0118272613
		SGD	1.0000000000
		SGD, 1% Clipping	0.0998883003
		Normalized SGD	0.2986719498
		Normalized SGD, 1% Clipping	0.0473530160
		Adafactor	0.0032185243
		Adafactor, 1% Clipping	0.0020526722
	20	Adam	0.0085812985
		SGD	1.0000000000
		SGD, 1% Clipping	0.0695261436
		Normalized SGD	0.2720821873
		Normalized SGD, 1% Clipping	0.0340152932
		Adafactor	0.0062594142
		Adafactor, 1% Clipping	0.0037904315

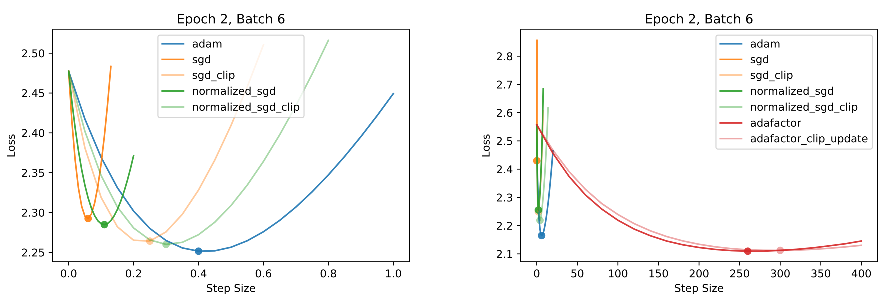
Table 4: Average ratio of directional sharpness of optimization algorithms with respect to SGD on the masked language modeling task in SGD trajectory.



(a) Experiment 1

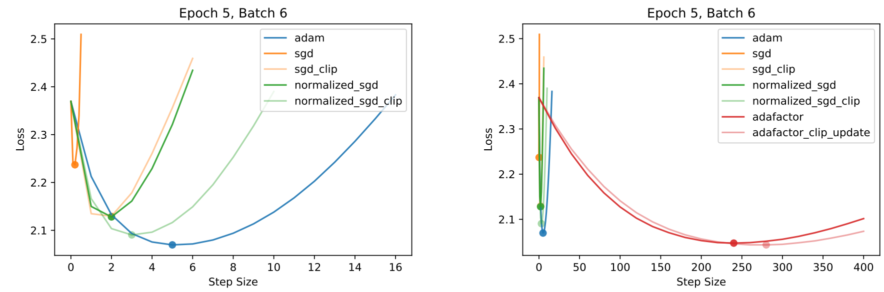


(b) Experiment 2

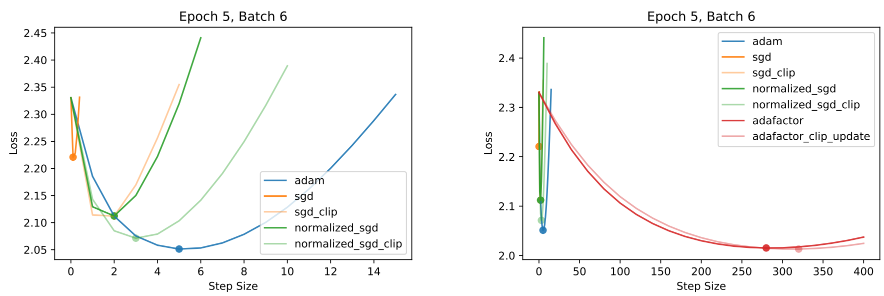


(c) Experiment 3

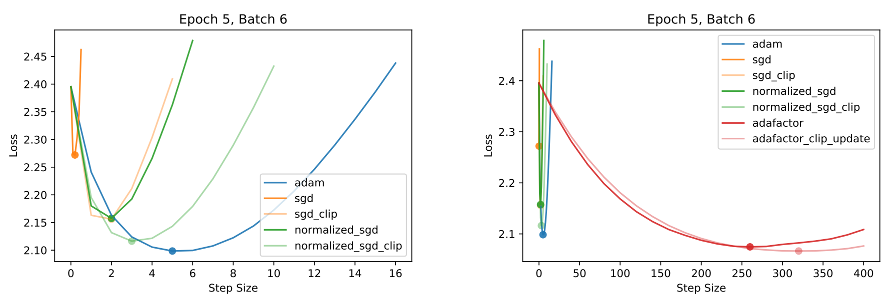
Figure 9: Landscape visualization of machine translation in SGD trajectory at Epoch 2.



(*a*) Experiment 1

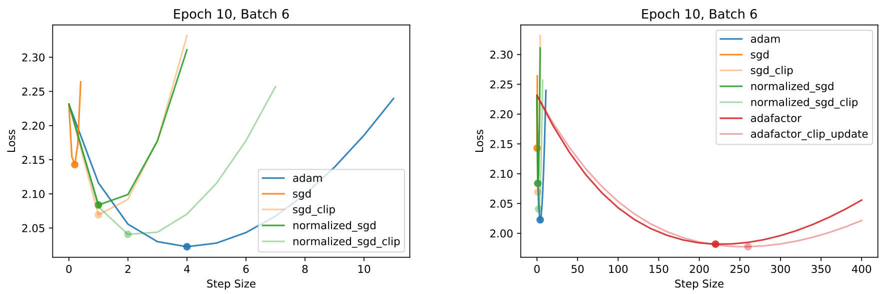


(*b*) Experiment 2

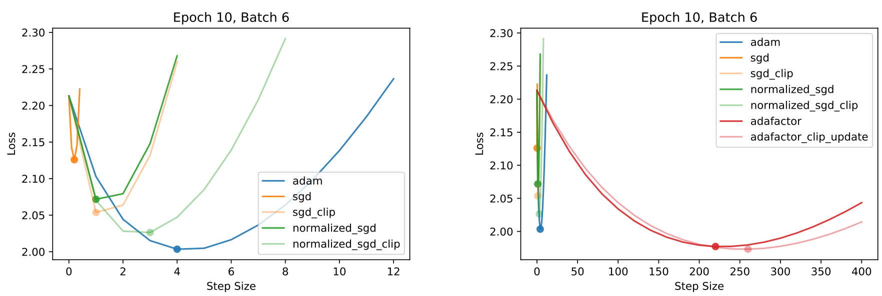


(*c*) Experiment 3

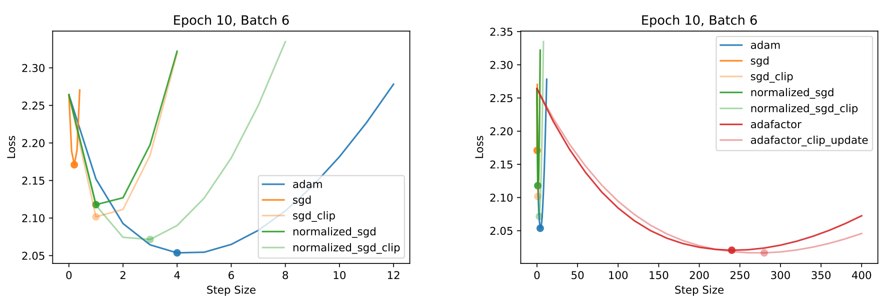
Figure 10: Landscape visualization of machine translation in SGD trajectory at Epoch 5.



(a) Experiment 1

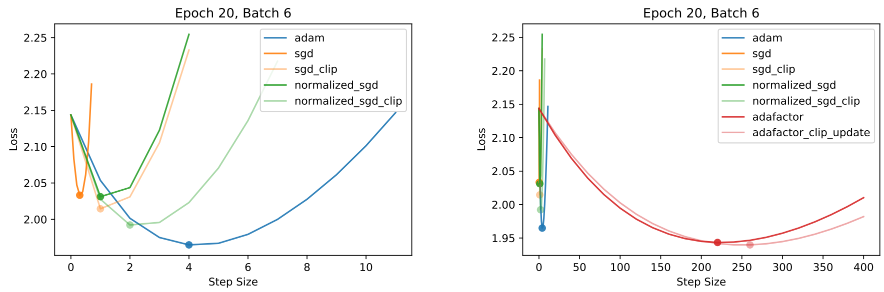


(b) Experiment 2

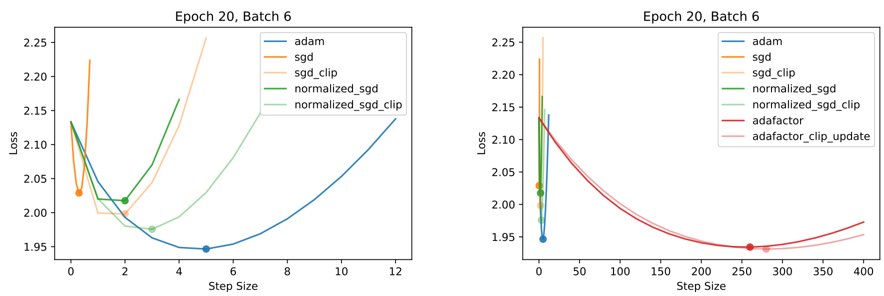


(c) Experiment 3

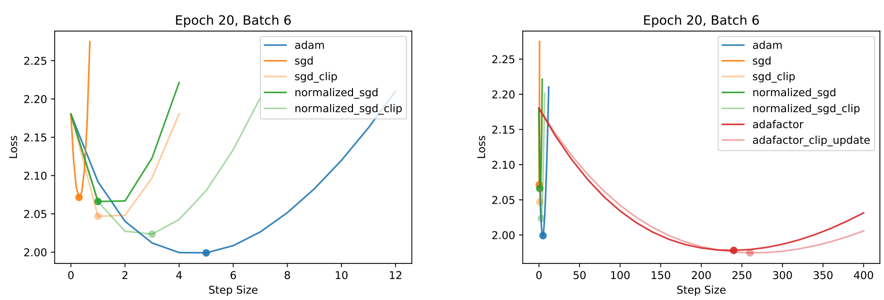
Figure 11: Landscape visualization of machine translation in SGD trajectory at Epoch 10.



(a) Experiment 1



(b) Experiment 2



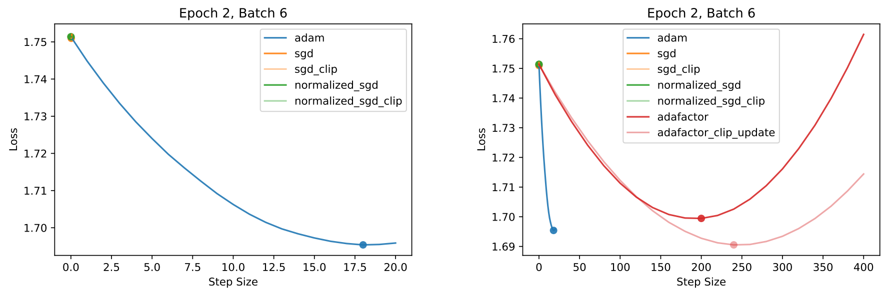
(c) Experiment 3

Figure 12: Landscape visualization of machine translation in SGD trajectory at Epoch 20.

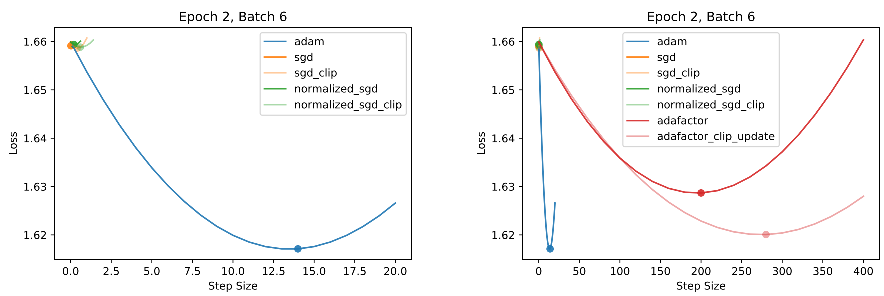
B.2. Adam Trajectory

Task	Epoch	Algorithm	Sharpness Ratio
Machine Translation	2	Adam	-0.0020924184
		SGD	1.0000000000
		SGD, 1% Clipping	0.0082061978
		Normalized SGD	-0.0007742774
		Normalized SGD, 1% Clipping	0.0009812440
		Adafactor	0.0000011169
		Adafactor, 1% Clipping	0.0000002866
	5	Adam	0.0000473345
		SGD	1.0000000000
		SGD, 1% Clipping	0.0003763665
		Normalized SGD	0.0017194651
		Normalized SGD, 1% Clipping	0.0003852057
		Adafactor	0.0000001931
		Adafactor, 1% Clipping	0.0000001583
	10	Adam	0.0001294576
		SGD	1.0000000000
		SGD, 1% Clipping	0.0022812745
		Normalized SGD	0.0023490605
		Normalized SGD, 1% Clipping	0.0007795924
		Adafactor	0.0000003584
		Adafactor, 1% Clipping	0.0000002897
	20	Adam	0.0001363408
		SGD	1.0000000000
		SGD, 1% Clipping	0.0018917628
		Normalized SGD	0.0020970826
		Normalized SGD, 1% Clipping	0.0006429824
		Adafactor	0.0000005021
		Adafactor, 1% Clipping	0.0000004034

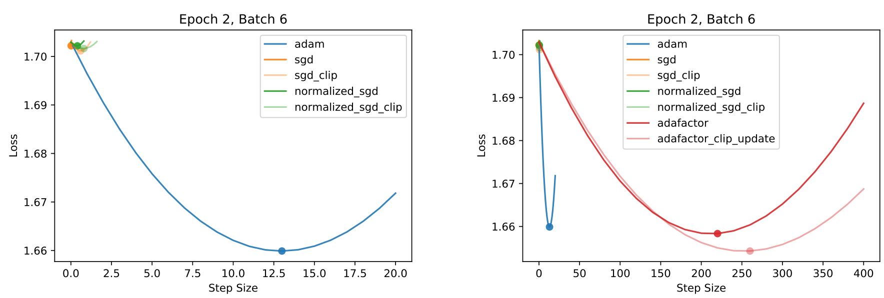
Table 5: Average ratio of directional sharpness of optimization algorithms with respect to SGD on the machine translation task in Adam trajectory.



(*a*) Experiment 1

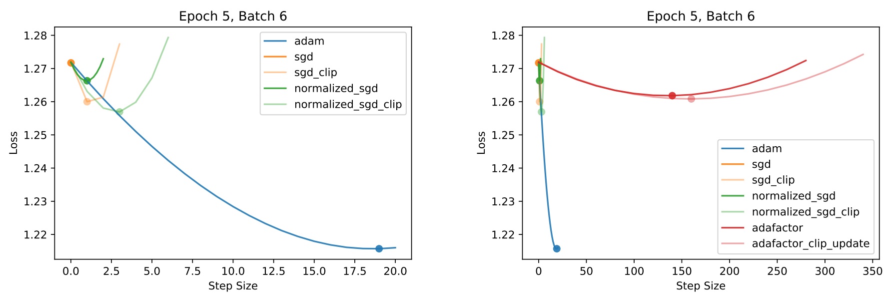


(*b*) Experiment 2

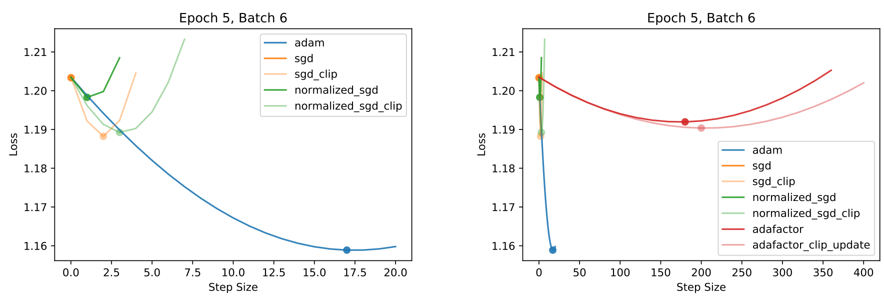


(*c*) Experiment 3

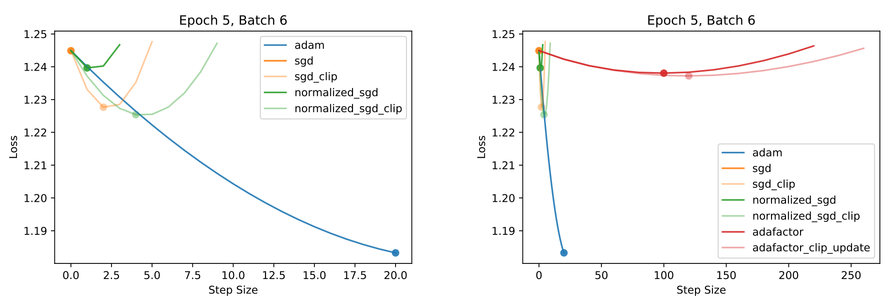
Figure 13: Landscape visualization of machine translation in Adam trajectory at Epoch 2.



(a) Experiment 1

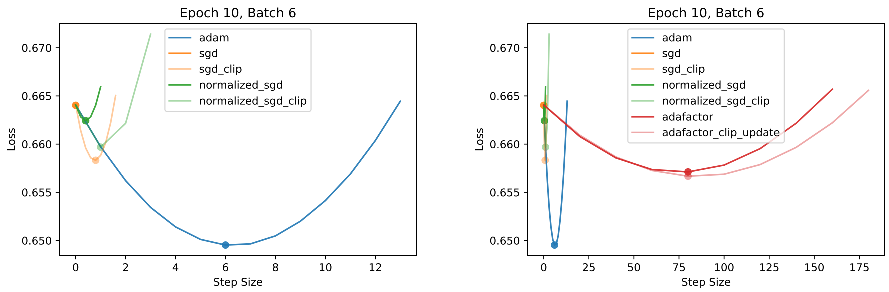


(b) Experiment 2

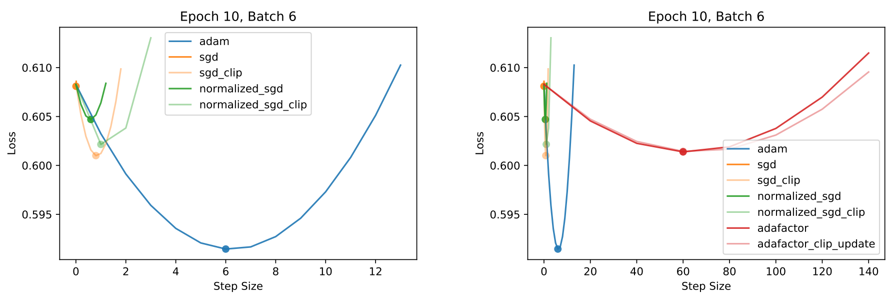


(c) Experiment 3

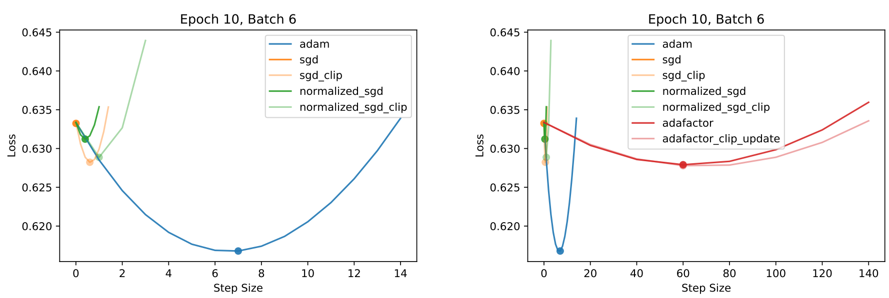
Figure 14: Landscape visualization of machine translation in Adam trajectory at Epoch 5.



(*a*) Experiment 1

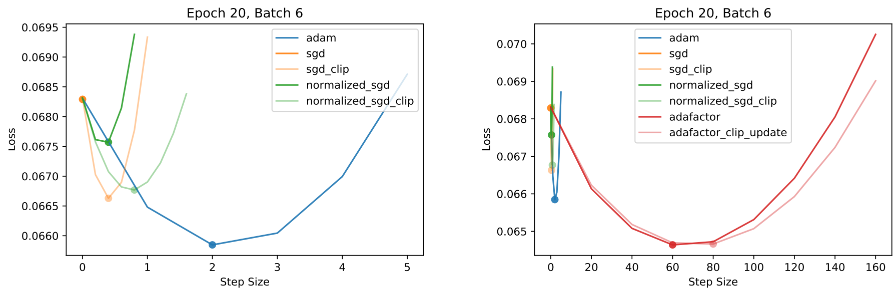


(*b*) Experiment 2

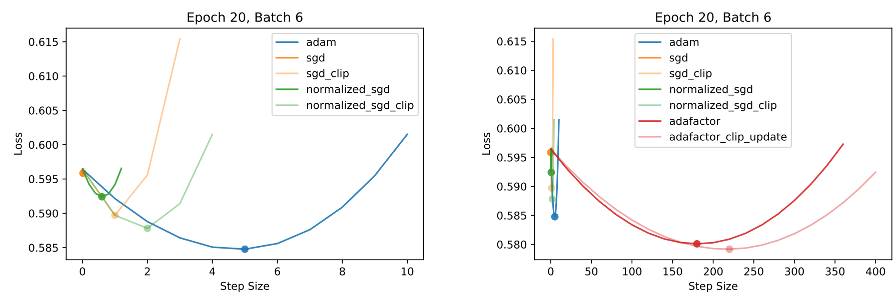


(*c*) Experiment 3

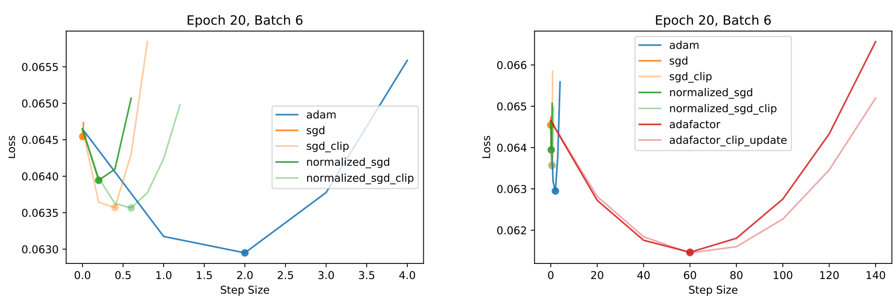
Figure 15: Landscape visualization of machine translation in Adam trajectory at Epoch 10.



(a) Experiment 1



(b) Experiment 2

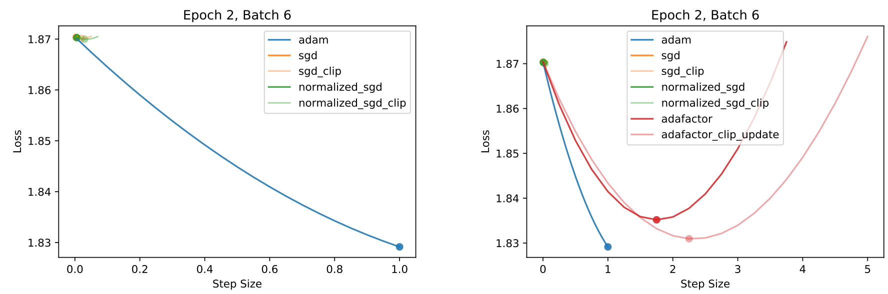


(c) Experiment 3

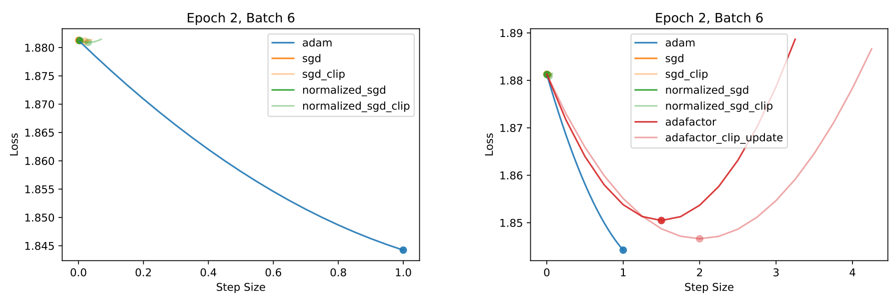
Figure 16: Landscape visualization of machine translation in Adam trajectory at Epoch 20.

Task	Epoch	Algorithm	Sharpness Ratio
Masked Language Modeling	2	Adam	0.0026182668
		SGD	1.0000000000
		SGD, 1% Clipping	0.0620023869
		Normalized SGD	0.2729090348
		Normalized SGD, 1% Clipping	0.0298001689
		Adafactor	0.0025936360
		Adafactor, 1% Clipping	0.0015477958
	5	Adam	0.0014779775
		SGD	1.0000000000
		SGD, 1% Clipping	0.0263968925
		Normalized SGD	0.2155425640
		Normalized SGD, 1% Clipping	0.0122737985
		Adafactor	0.0027598900
		Adafactor, 1% Clipping	0.0015772839
	10	Adam	0.0034328710
		SGD	1.0000000000
		SGD, 1% Clipping	0.0297756218
		Normalized SGD	0.1949167394
		Normalized SGD, 1% Clipping	0.0139795561
		Adafactor	0.0092244251
		Adafactor, 1% Clipping	0.0051509819
	20	Adam	0.0040885673
		SGD	1.0000000000
		SGD, 1% Clipping	0.0339636744
		Normalized SGD	0.2222520762
		Normalized SGD, 1% Clipping	0.0170976330
		Adafactor	0.0093870774
		Adafactor, 1% Clipping	0.0055506315

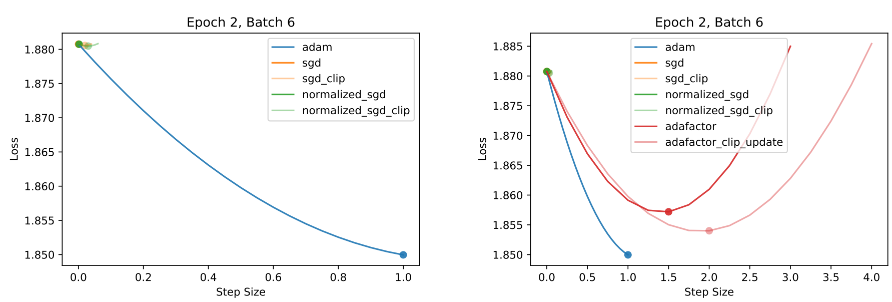
Table 6: Average ratio of directional sharpness of optimization algorithms with respect to SGD on the masked language modeling task in Adam trajectory.



(a) Experiment 1

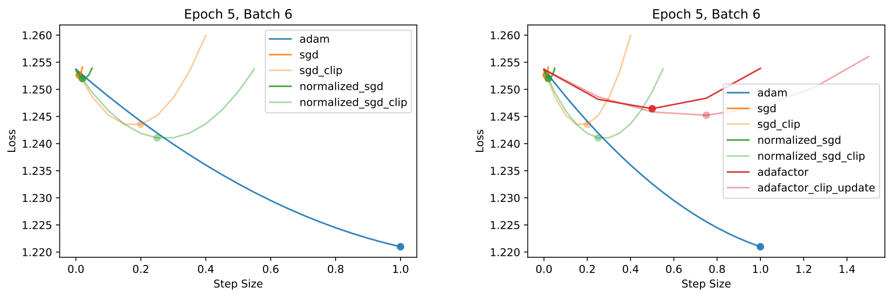


(b) Experiment 2

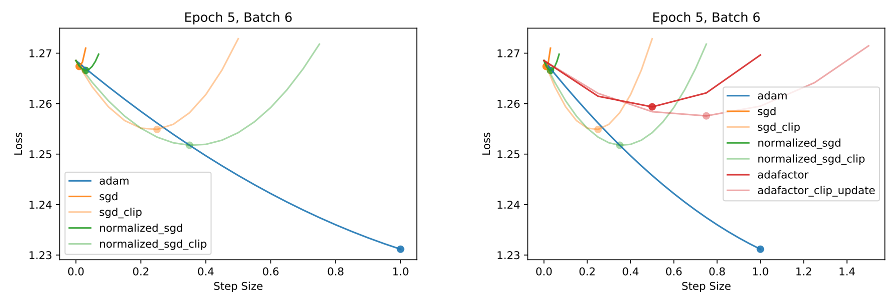


(c) Experiment 3

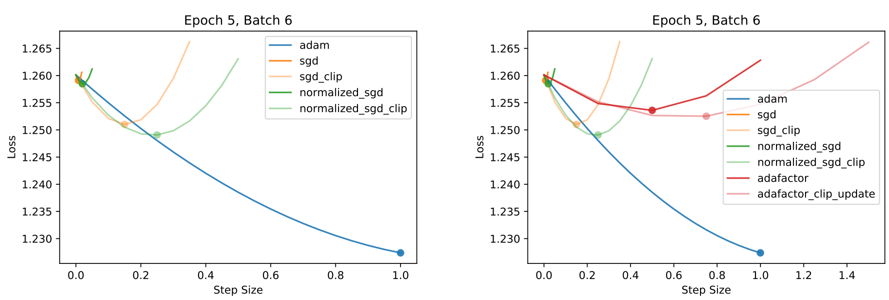
Figure 17: Landscape visualization of masked language modeling in Adam trajectory at Epoch 2.



(*a*) Experiment 1

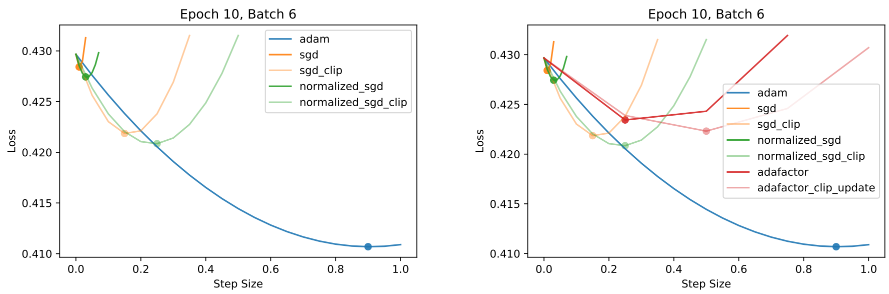


(*b*) Experiment 2

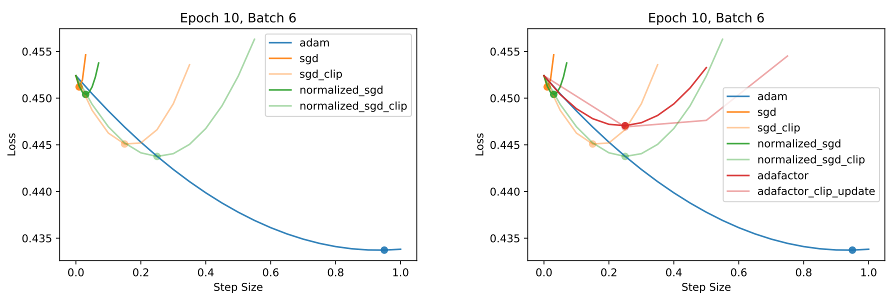


(*c*) Experiment 3

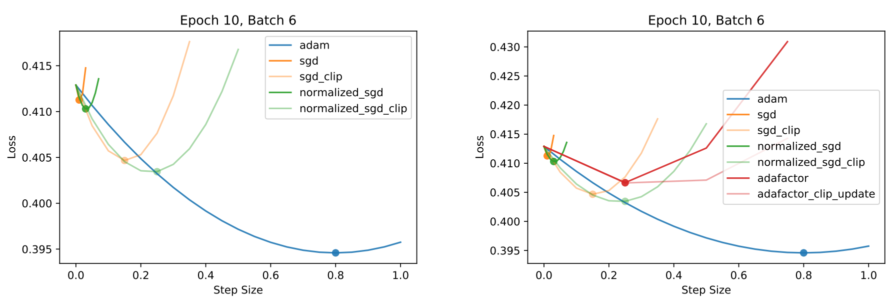
Figure 18: Landscape visualization of masked language modeling in Adam trajectory at Epoch 5.



(*a*) Experiment 1

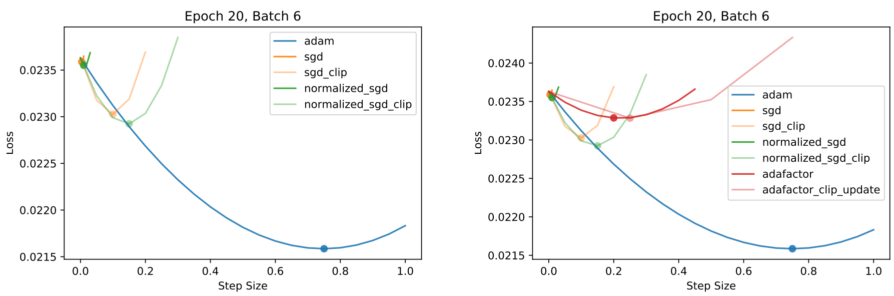


(*b*) Experiment 2

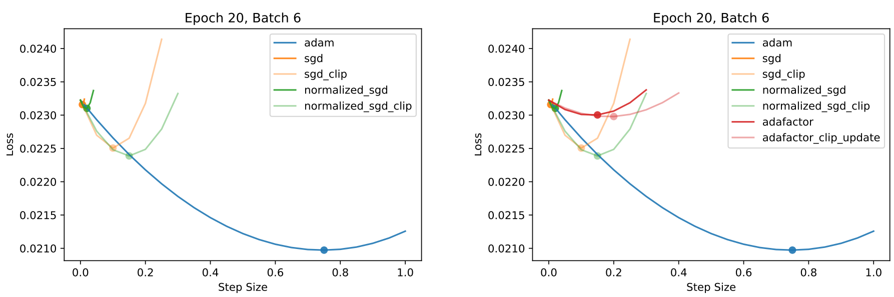


(*c*) Experiment 3

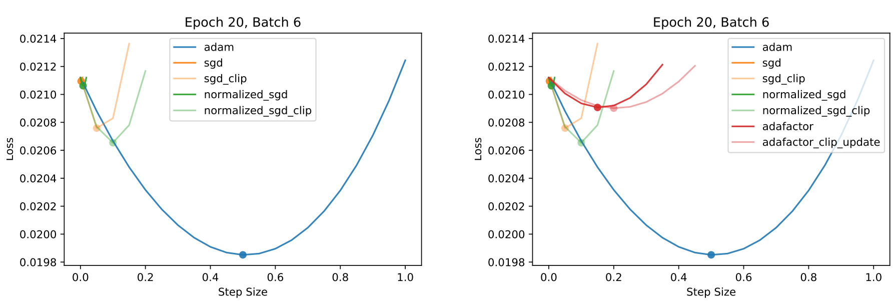
Figure 19: Landscape visualization of masked language modeling in Adam trajectory at Epoch 10.



(a) Experiment 1



(b) Experiment 2



(c) Experiment 3

Figure 20: Landscape visualization of masked language modeling in Adam trajectory at Epoch 20.

B.3. Discussion

As we can observe, our observation is very coherent across different tasks, model architectures, iterations, and local geometry. The directional sharpness is relatively stable for the same task across iterations, and coordinate-wise clipping always improve the sharpness of the direction and find a better direction to optimize.

Trade-off Between Directional Sharpness and Gradient Correlation. While we want the directional sharpness of our optimization algorithm to be small in order to decrease loss faster, having as small sharpness as possible does not necessarily lead to fast loss decrement. Adafactor almost always has the lowest directional sharpness across all tasks, iterations, and local geometry, but Adafactor does not always find a good direction to optimize. In many cases, the loss does not decrease significantly even for the optimal step size, and the direction can be even worse than SGD. This shows that merely minimizing the directional sharpness is not enough for an optimization algorithm to work well. As discussed in Section 4, gradient correlation is also important in the convergence of optimization algorithms. However, we can conclude that high sharpness will lead to bad performance, as demonstrated by the performance of SGD, even if SGD has good gradient correlation.

Effect of Trajectory. It is well known that different optimization algorithms can follow different trajectory and converge to different in deep learning. Landscape visualizations show that Adafactor performs well in SGD trajectory but not Adam trajectory on the machine translation task. This shows that different optimization algorithms has local geometry with different properties. The effect of trajectory is therefore an interesting problem to study. However, we point out that almost in all cases, Adam has good performance and significantly outperforms SGD, so trajectory is not necessarily related to the explanation for Adam's excellent performance in practice.