Eve: Less Memory, Same Might

Aditya Tomar
adityatomar@berkeley.edu
Department of Electrical Engineering and Computer Sciences, University of California, Berkeley
Berkeley, California, USA

Siddharth Singh (advisor)
ssingh37@umd.edu
Department of Computer Science, University of Maryland
College Park, Maryland, USA

Tom Goldstein (advisor)
tomg@cs.umd.edu
Department of Computer Science, University of Maryland
College Park, Maryland, USA

Abhinav Bhatele (advisor)
bhatele@cs.umd.edu
Department of Computer Science, University of Maryland
College Park, Maryland, USA

Abstract

Adaptive optimizers, which adjust the learning rate for individual parameters, have become the standard for training deep neural networks. One such optimizer is AdamW, a popular adaptive method that maintains two optimizer state values (momentum and variance) per parameter, doubling the model's memory usage during training. Many proposed memory efficient optimizers claim to match AdamW's performance but lack its desirable qualities such as robustness to learning rate changes. This quality is especially desirable when pre-training LLMs, where experimenting with different combinations of hyperparamters to attain the ideal setting is infeasible due to time, cost, and compute constraints. We propose Eve, a Memory **E**fficient Adapti**V**e Moment **E**stimation algorithm that saves memory by reducing the variance term while also preserving AdamW's desirable properties across different training settings. We finetune Llama 2 70B on 64 GPUs and show memory savings of 20% over AdamW. We also compare our method to a recent well-received memory-efficient optimizer called Adam-mini and demonstrate better training stability across various learning rates.

Keywords

adaptive optimization, memory efficiency, approximation, training stability

1 Introduction

The per-parameter adaptivity of optimizers like AdamW allow them to outperform non-adaptive methods like stochastic gradient descent in converging quickly to good minimizers, even in high dimensional spaces [2]. But, the memory requirements for training with such optimizers is extremely high due to the extra states maintained by them. As a result, many memory-efficient approximations to AdamW's algorithm have been proposed. However, these methods either lose valuable qualities of AdamW, such as its robustness to a wider range of learning rates, or require highly tuned and crafted conditions to reach the same minimizers as AdamW. These constraints are unacceptable in a setting where compute and time are limited resources, which is why current memory efficient optimizers have not yet replaced AdamW convincingly.

To this end, we propose Eve, a very simple approximation of the AdamW algorithm that saves a significant amount of memory while preserving the single most important quality of AdamW: robustness to learning rates.

2 Eve: A Memory Efficient Optimizer

In modern neural networks like the transformer, linear layers comprise the majority of parameters. Moreover, for every linear layer's weight matrix of shape $M \times N$, AdamW stores two additional $M \times N$ matrices for the momentum and variance terms. To compress AdamW's optimizer state, we maintain a scalar per row of the variance matrix, reducing it to an $M \times 1$ vector. This prunes the optimizer state from storing $2 \times M \times N$ values to only $M \times (N + 1)$ values for a linear layer. As M and N grow larger for bigger models, the +1 term is amortized, essentially reducing the $2 \times M \times N$ values to $M \times N$ values.

We show our method in Algorithm 1 and illustrate the **row_mean** operation in Figure 1.

Algorithm 1 Eve: Memory Efficient Adaptive Moment Estimation

$m_0 \leftarrow 0$ (Initialize MxN momentum matrix)
$\hat{v_0} \leftarrow 0$ (Initialize Mx1 variance vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
  $t \leftarrow t + 1$
  $g_t \leftarrow \nabla_\theta f(\theta_{t-1})$ (Get gradients w.r.t. objective function at timestep t)
  $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update momentum matrix)
  $\hat{v_t} \leftarrow \beta_2 \cdot \hat{v}_{t-1} + (1 - \beta_2) \cdot$ **row_mean**(g_t^2) (Update variance vector)
  $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot m_t / (\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

SC '24, November 17-22, 2024, Atlanta, GA

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/10.1145/nnnnnnn.nnnnnnn

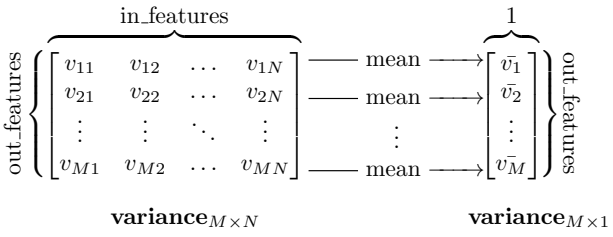

Figure 1: Eve takes the per-row average of the variance matrix. Note that Algorithm 1 Line 8 is mathematically equivalent to this

3 Experiments

We perform a model size and learning rate ablation study. We train two models, GPT2-124M and GPT2-350M [3], using Eve, AdamW, Adafactor [4], SM3 [1], and Adam-mini, which is a recent memory efficient optimizer introduced by Zhang et al. [8]. For each model, we train for 100B tokens with different learning rates. We record the final validation perplexities in Figure 2.

Across all learning rates, we observe that Eve matches AdamW for both models, whereas the other memory-efficient optimizers are either unstable or simply less performant, especially for the 350M model. For smaller learning rates, Adam-mini matches AdamW and Eve's performance. However, note that AdamW and Eve together achieve the best PPL at higher learning rates, like 1e-3, where the other optimizers struggle to do the same. Moreover, Yao et al. [7] empirically show that a PPL difference greater than 0.5 is the difference between an X and a 2X parameter model, so the PPL differences in Figure 2 can be considered non-trivial.

We include validation loss curves corresponding to the best learning rate for each optimizer in Figure 3. We also run downstream tasks on the final trained models, as shown in Figure 4.

4 Memory Savings

We utilize AxoNN [5, 6] to distribute and train models ranging from 1.1B up to 70B parameters on 64 NVIDIA A100 (40 GB each) GPUs using tensor parallelism. As seen in Figure 5, for Llama 2 70B, we save 267 GBs, or 20.27% more memory than AdamW, the equivalent of ~ 7 more A100 (40 GB) GPUs! This not only helps reduce monetary costs but also saves energy and makes training large models more accessible.

5 Conclusion

In this paper, we introduced Eve, a memory efficient optimizer that matches AdamW's performance across a range of learning rates while using a much smaller memory footprint. We compared Eve with other established memory-efficient optimizers for different model sizes and learning rates and showed better training stability. We trained Llama 2 70B on 64 A100 GPUs, where we used 20% less GPU memory compared to AdamW. As future work, we plan to pre-train larger models using our optimizer and investigate how Eve is able to approximate AdamW so well.

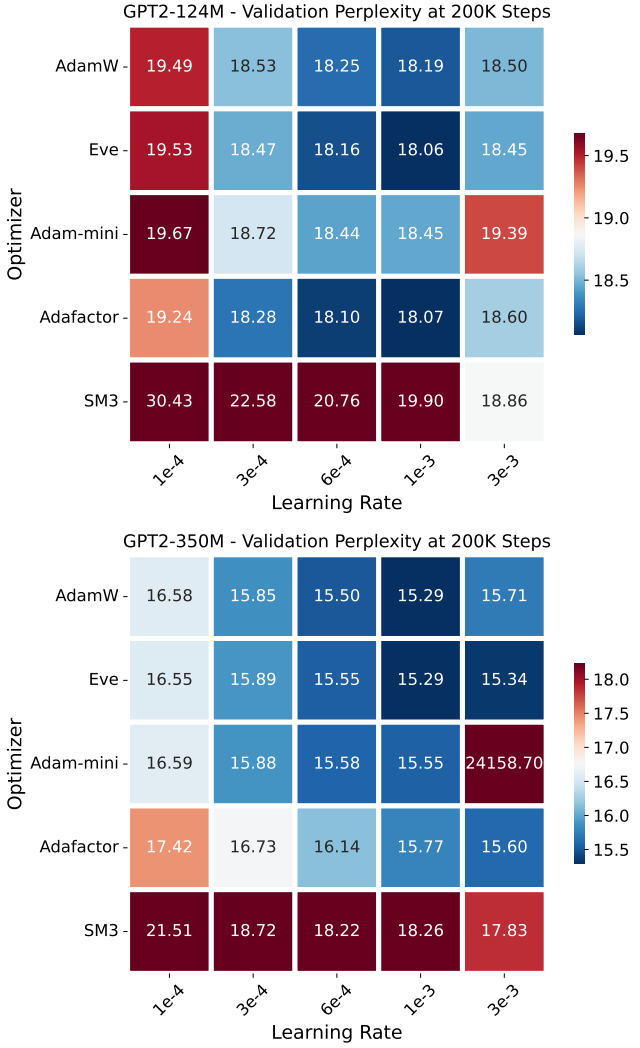

Figure 2: Final validation PPLs across different learning rates for all optimizers on GPT2-124M and GPT2-350M after 200K steps (100B tokens)

References

[1] Rohan Anil, Vineet Gupta, Tomer Koren, and Yoram Singer. 2019. Memory Efficient Adaptive Optimization. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/file/8f1fa0193ca2b5d2fa0695827d8270e9-Paper.pdf

[2] Ilya Loshchilov and Frank Hutter. 2017. Fixing Weight Decay Regularization in Adam. *CoRR* abs/1711.05101 (2017). arXiv:1711.05101 http://arxiv.org/abs/1711.05101

[3] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. *Language Models are Unsupervised Multitask Learners.* Technical Report.

[4] Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive Learning Rates with Sublinear Memory Cost. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 4596–4604. https://proceedings.mlr.press/v80/shazeer18a.html

[5] Siddharth Singh and Abhinav Bhatele. 2022. AxoNN: An asynchronous, message-driven parallel framework for extreme-scale deep learning. In *Proceedings of the IEEE International Parallel & Distributed Processing Symposium (IPDPS '22)*. IEEE Computer Society.

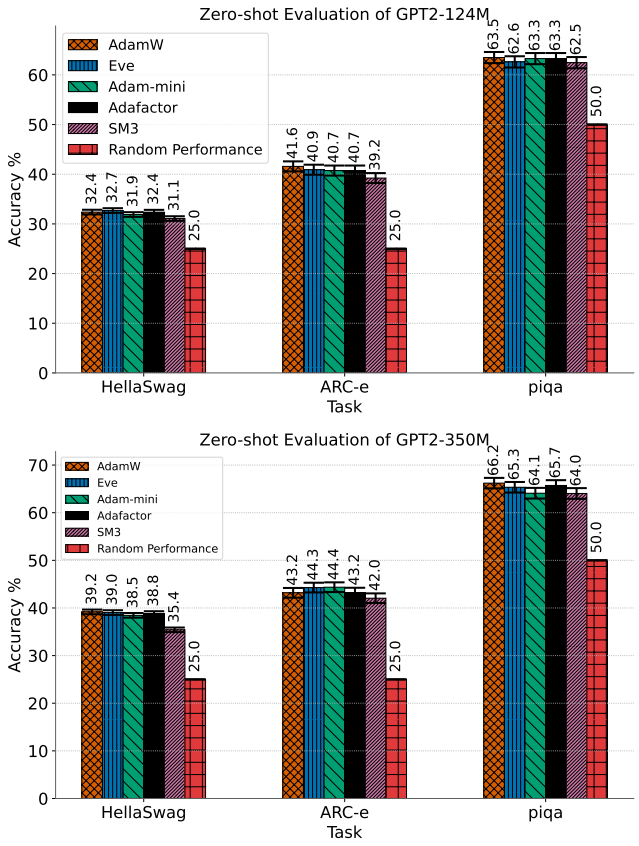

Figure 4: Zero-shot evaluations for final GPT2-124M and GPT2-350M checkpoints with best learning rates

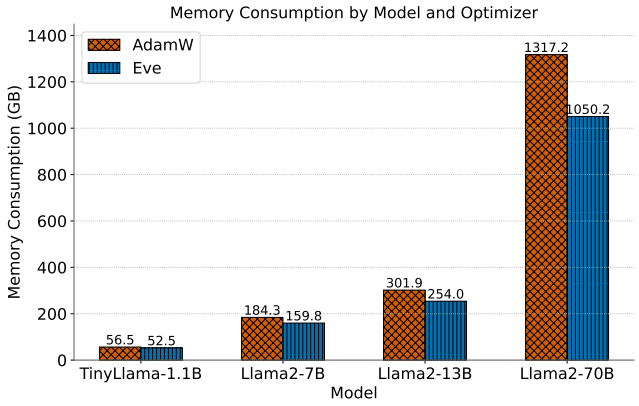

Figure 5: Comparison of total memory utilization across all 64 GPUs between AdamW and Eve for LLMs

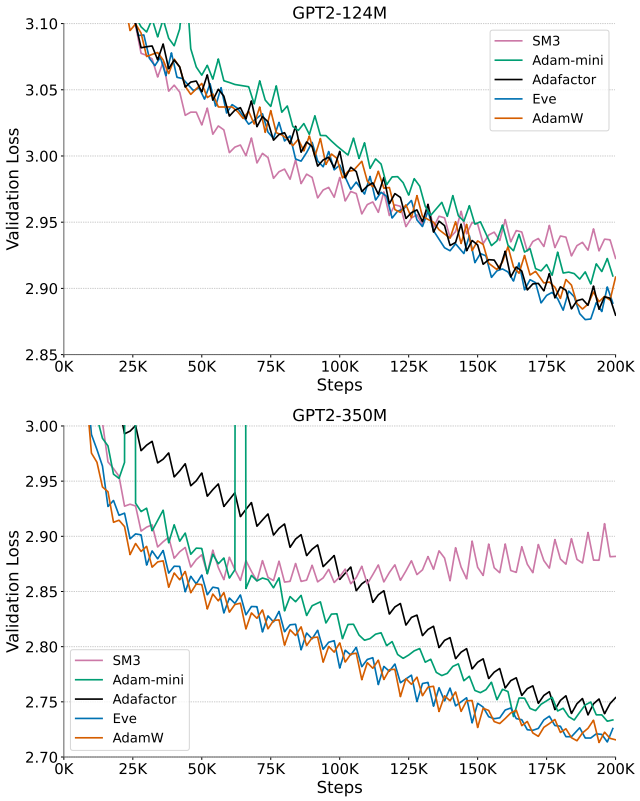

Figure 3: Validation loss curves using best learning rates for GPT2-124M and GPT2-350M for 200K steps (100B tokens)

[6] Siddharth Singh, Prajwal Singhania, Aditya K. Ranjan, Zack Sating, and Abhinav Bhatele. 2024. A 4D Hybrid Algorithm to Scale Parallel Training to Thousands of GPUs. arXiv:2305.13525 [cs.LG]

[7] Zhewei Yao, Xiaoxia Wu, Cheng Li, Stephen Youn, and Yuxiong He. 2023. ZeroQuant-V2: Exploring Post-training Quantization in LLMs from Comprehensive Study to Low Rank Compensation. arXiv:2303.08302 [cs.LG] https://arxiv.org/abs/2303.08302

[8] Yushun Zhang, Congliang Chen, Ziniu Li, Tian Ding, Chenwei Wu, Yinyu Ye, Zhi-Quan Luo, and Ruoyu Sun. 2024. Adam-mini: Use Fewer Learning Rates To Gain More. arXiv:2406.16793 [cs.LG] https://arxiv.org/abs/2406.16793