Eve: Less Memory, Same Might

Aditya Tomar[1], Siddharth Singh[2] (advisor), Tom Goldstein[2] (advisor), Abhinav Bhatele[2] (advisor)

[1]Department of Electrical Engineering and Computer Sciences, University of California, Berkeley [2]Department of Computer Science, University of Maryland

Abstract

Adaptive optimizers, which adjust the learning rate for individual parameters, have become the standard for training deep neural networks. AdamW [1] is a popular adaptive method that maintains two optimizer state values (momentum and variance) per parameter, doubling the model's memory usage during training. Several 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 hyperparameters is infeasible. 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 hyperparameter settings.

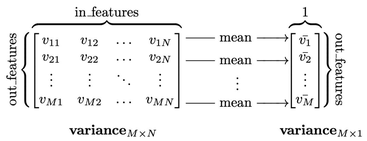
Eve: A Memory Efficient Optimizer

- Linear layers comprise the majority of parameters in modern neural networks
- For every linear layer's weight matrix (MxN), AdamW stores two MxN matrices for the momentum and variance terms
- We compress AdamW's optimizer state by performing a per-row average of the variance matrix, reducing it to an Mx1 vector for each linear layer
- This reduces the optimizer state from storing 2xMxN values to only Mx(N+1) values for a linear layer

Algorithm 1 Eve: Memory Efficient Adaptive Moment Estimation

1: $m_0 \leftarrow 0$ (Initialize MxN momentum matrix)
2: $\hat{v}_0 \leftarrow 0$ (Initialize Mx1 variance vector)
3: $t \leftarrow 0$ (Initialize timestep)
4: **while** θ_t not converged **do**
5: $\quad t \leftarrow t + 1$
6: $\quad g_t \leftarrow \nabla_\theta f(\theta_{t-1})$ (Get gradients w.r.t. objective function at timestep t)
7: $\quad m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update momentum matrix)
8: $\quad \hat{v}_t \leftarrow \beta_2 \cdot \hat{v}_{t-1} + (1 - \beta_2) \cdot \mathbf{row\_mean}(g_t^2)$ (Update variance vector)
9: $\quad \theta_t \leftarrow \theta_{t-1} - \alpha \cdot m_t / (\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)
10: **end while**
11: **return** θ_t (Resulting parameters)

Note: Line 8 is mathematically equivalent to taking the row_mean of the variance matrix. Also, we perform bias correction on the momentum and variance terms in the same fashion as AdamW (not shown here).



Perplexity: Learning Rate Ablation Results

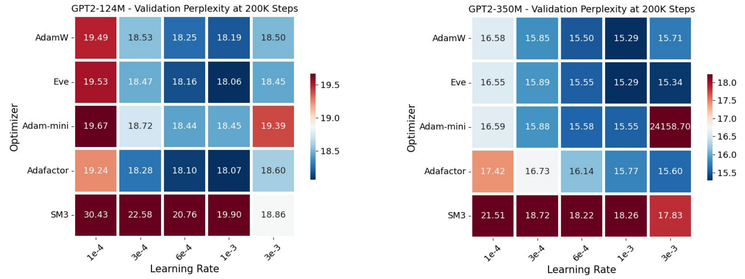


Fig 1: Final validation PPLs across different learning rates and optimizers for two models after 200K steps

- Eve matches AdamW across all learning rates, whereas other optimizers are less performant, especially for GPT2-350M
- Yao et al. [3] empirically showed that a PPL difference > 0.5 is the difference between an X and a 2X parameter model, so the PPL differences above may be non-trivial
- Adam-mini is close to AdamW for smaller learning rates, but AdamW and Eve perform better at higher learning rates

Validation Loss

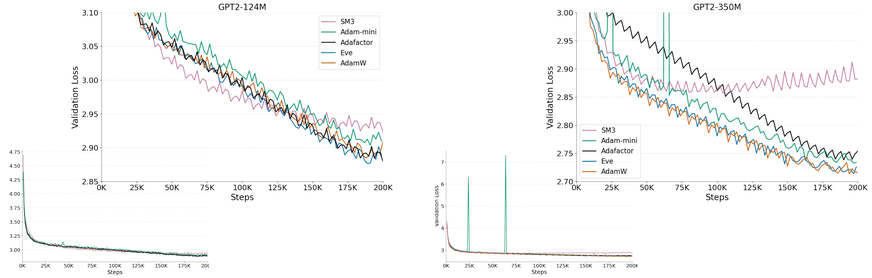


Fig 2: Validation loss curves (zoomed-in and out) using best learning rates for GPT2-124M and GPT2-350M for 200K steps. We find that Eve very closely follows AdamW throughout pre-training with no spikes or instabilities

Memory Savings and Downstream tasks

- We utilize AxoNN [4] to parallelize model training for models ranging from 1.1B to 70B parameters on 64 NVIDIA A100 (40 GB) GPUs using tensor parallelism
- 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
- Moreover, as the model size increases, the amount of memory saved also increases

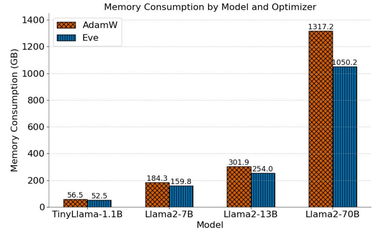


Fig 3: Comparison of total memory utilization across all 64 GPUs between AdamW and Eve for LLMs

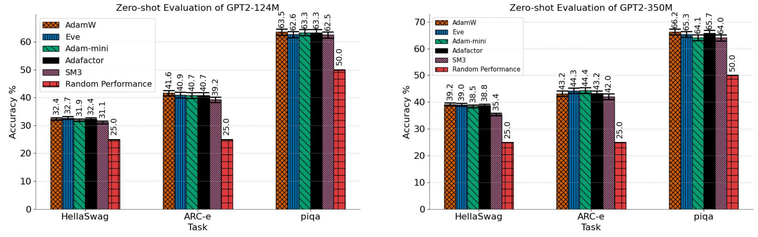


Fig 4: Zero-shot evaluations for final GPT2-124M and GPT2-350M checkpoints with best learning rates

Conclusion

- We introduce Eve, a memory efficient optimizer that matches AdamW's performance across a range of learning rates while using a much smaller memory footprint
- We save memory by maintaining a scalar per row of the variance matrix for all linear layers
- We present an ablation study, where we compare Eve with other established memory-efficient optimizers for different model sizes and learning rates and show better training stability
- We train Llama 2 70B on 64 A100 GPUs, where we use 20% less GPU memory compared to AdamW
- As future work, we plan to pre-train larger models using our optimizer and investigate how and why Eve is able to approximate AdamW so well

Acknowledgements

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

References

[1] Loshchilov et al., Decoupled Weight Decay Regularization, ICLR 2019
[2] Zhang et al., Adam-mini: Use Fewer Learning Rates To Gain More, arXiv:2406.16793
[3] Yao et al., ZeroQuant-V2: Exploring Post-training Quantization in LLMs from Comprehensive Study to Low Rank Compensation, arXiv:2303.08302
[4] Singh et al., AxoNN: An asynchronous, message-driven parallel framework for extreme-scale deep learning, IPDPS 2022