

Lather, Rinse, Repeat: The Shampoo Optimizer

Stanford CS224N { Default } Project

Ricky Rios

Department of Computer Science
Stanford University
rjrios@stanford.edu

Abstract

Fine-tuning large language models is typically performed using first-order optimization methods such as AdamW. While these optimizers are computationally efficient, they rely only on local gradient information and fail capture the broader curvature of the loss landscape. In this project, I investigate whether the second-order optimizer Shampoo can improve optimization dynamics when fine-tuning GPT-2 models.

I evaluate Shampoo on two main NLP tasks: paraphrase detection and Sonnet text generation. I compare Shampoo against the widely used AdamW optimizer while keeping all other aspects of the training pipeline fixed. My results show that for certain tasks, namely paraphrase detection, Shampoo improves validation accuracy demonstrates faster convergence during training. However, performance on the text generation task measured using chrF decreases relative to AdamW and appears less stable. These results suggest that second-order preconditioning can improve optimization efficiency for classification tasks but may require further tuning for generative tasks.

1 Key Information to include

- Mentor: Ahmed Ahmed
- External Collaborators (if you have any): n/a
- Sharing project: n/a
- Late Days (if you are using any): 0

Team member	Late days remaining
<Ricky Rios>	<3>
Total pooled	<3>
Team size	<1>
Late days applied for the report	<1>

2 Introduction

Training large language models depends heavily on the choice of optimizer. Most large models are trained using first-order optimizers such as Adam or AdamW. These methods are simple and computationally efficient, but they rely only on local gradient information and do not explicitly incorporate the curvature of the loss landscape. This can lead to unstable training and often oscillations at higher learning rates.

Second-order optimization methods attempt to address this limitation by using curvature information to guide parameter updates. In practice, however, second-order methods are often too expensive for large neural networks because they require computing or approximating Hessian matrices which can be large.

Shampoo aims to solve this by incorporating second-order information but avoiding Hessian calculations. [1]. Shampoo is a second-order optimizer that approximates curvature information using structured statistics of gradient tensors. Instead of computing the full Hessian, Shampoo maintains matrix statistics for each tensor dimension and uses them to precondition gradients. This provides more information about the geometry of the optimization landscape while remaining computationally tractable.

In this project I explore whether replacing the AdamW optimizer with Shampoo improves optimization dynamics when fine-tuning GPT-2. The goal is to determine whether second-order preconditioning leads to improved convergence behavior and downstream performance when fine-tuning a language model.

3 Related Work

Adam and AdamW are widely used optimizers for training and fine-tuning transformer models due to their adaptive learning rate mechanisms and computational efficiency [2, 3]. These methods scale gradients element-wise using running estimates of the first and second moments of gradients. Their design makes them extremely compatible with the backpropagation techniques used to train neural networks, but can be unstable and result in oscillations without enough information about the loss landscape.

Second-order optimization methods, however, aim to improve training by incorporating curvature information from the loss landscape. Classical approaches such as Newton’s method can provide faster convergence but are typically infeasible for modern neural networks due to the computational cost of computing and storing Hessian matrices. Several works have explored scalable approximations of second-order methods, including natural gradient approaches [4] and Kronecker-factored approximations such as K-FAC [5]. The find that incorporating second order information does improve overall training but needs to be balanced with computational efficiency.

Thus, Shampoo was born. Shampoo is a scalable second-order optimizer that approximates full-matrix preconditioning using tensor-structured statistics of gradients [1]. By maintaining matrix statistics along each tensor dimension, Shampoo can incorporate richer curvature information than first-order optimizers while remaining practical for deep learning models.

Other works have also explored improvements to Shampoo that combine second-order preconditioning with adaptive optimization techniques to improve stability during training [6] similar to the AdamW optimizer. These methods aim to retain the benefits of curvature-aware updates while addressing the sensitivity of second-order optimizers.

4 Approach

My approach builds on the default GPT-2 fine-tuning framework provided for the CS224N project.

4.1 Baseline

I built the GPT-2 base model by completing the course code provided to us and implementing the missing portions needed for fine-tuning. In particular, I implemented the rest of the forward pass, the multi-head self-attention calculations, and the AdamW optimizer used for the baseline system.

The forward pass combines token embeddings and positional embeddings to produce predictions for downstream tasks. Within each transformer block, attention is computed using scaled dot-product attention. Given queries Q , keys K , and values V , attention is computed as

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \quad (1)$$

where the dimensionality of the key vectors is represented by d_k . Multi-head attention allows the model to attend to multiple features simultaneously. It is computed as

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \quad (2)$$

where each head is defined as

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V). \quad (3)$$

I also implemented the AdamW optimizer as the baseline training method. AdamW performs parameter updates using first and second moment estimates of gradients while applying decoupled weight decay. The update rule can be written as

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (4)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (5)$$

$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} - \alpha \lambda \theta_t \quad (6)$$

where g_t is the gradient, m_t is the first moment, v_t is the second moment, and λ is the weight decay coefficient. After finishing these components, I had a working GPT-2 baseline that I later used as the reference for evaluating my implementation of Shampoo.

4.2 Shampoo Optimizer

After implementing the baseline GPT-2 model with AdamW, I replaced the optimizer with Shampoo, the second-order optimization method [1]. Unlike first-order methods such as AdamW, which scale gradients element-wise, Shampoo rescales gradients using matrix statistics that approximate curvature information of the loss landscape.

For a parameter tensor W with gradient G_t at step t , Shampoo maintains preconditioning matrices for each tensor dimension. For a matrix parameter $W \in \mathbb{R}^{m \times n}$, the gradient statistics are accumulated using an exponential moving average

$$L_t = \beta_2 L_{t-1} + (1 - \beta_2) G_t G_t^T \quad (7)$$

$$R_t = \beta_2 R_{t-1} + (1 - \beta_2) G_t^T G_t \quad (8)$$

where β_2 controls the smoothing of the second-moment statistics and L_t and R_t capture curvature information along the row and column dimensions of the parameter tensor. The exponential moving average follows the same form used in adaptive optimizers such as Adam [2].

The gradient is then preconditioned using matrix inverse roots of these statistics:

$$\tilde{G}_t = L_t^{-1/4} G_t R_t^{-1/4} \quad (9)$$

This transformation rescales the gradient using curvature information from both tensor dimensions. The parameter update is then performed as

$$W_{t+1} = W_t - \eta \tilde{G}_t \quad (10)$$

where η is the learning rate.

In my Shampoo implementation, I used a learning rate of 1×10^{-3} , $\epsilon = 1 \times 10^{-4}$ for numerical stability, and $\beta_2 = 0.9$ as the exponential moving average coefficient for the second-moment statistics. I recomputed the inverse-root preconditioners every 20 steps rather than at every update in order to reduce computational cost. Aside from these stability and efficiency parameters, Shampoo introduces very few hyperparameters to tune, which is one of the advantages highlighted in the original paper [1]. Weight decay was supported through decoupled weight decay, although in my experiments I left the rest of the optimizer settings fixed so that the main comparison focused on the effect of Shampoo itself.

I implemented Shampoo using the PyTorch optimizer interface and replaced the AdamW update rule in the training pipeline while keeping the rest of the GPT2 implementation unchanged. This includes the model architecture, datasets, and training procedure. This allowed me to isolate the effect of second-order preconditioning on convergence speed and downstream performance.

5 Experiments

5.1 Data

The paraphrase detection task uses the Quora Question Pairs dataset, which contains approximately 400,000 pairs of questions labeled according to whether they are semantically equivalent. The model receives two questions as input and predicts whether they are paraphrases. For the generation task I use a dataset consisting of Shakespeare’s sonnets. The model is fine-tuned to generate text in the style of Shakespeare.

5.2 Evaluation method

For the paraphrase detection task I evaluate performance using development set accuracy, which measures the fraction of correctly predicted examples. For the sonnet generation task I use chrF, a character-level F-score metric that measures similarity between generated text and reference text using character n-grams. I also compare the losses during training.

5.3 Experimental Details

All experiments were conducted using the GPT-2 base model provided in the default project framework. I kept the model architecture and training pipeline fixed and changed only the optimizer in order to isolate the effect of Shampoo.

For the baseline optimizer, I used AdamW. I then replaced AdamW with Shampoo. For both tasks, I performed a learning-rate sweep to evaluate training stability across different learning rates. For the sonnet generation I tested 0.01, 0.05, 0.001, 0.0001, 5×10^{-5} , and 1×10^{-5} . Note that 0.01 and 0.05 were omitted because they resulted in losses much higher than the other rates. The learning rates tested for paraphrase detection were 0.01, 0.001, 0.0001, and 1×10^{-5} .

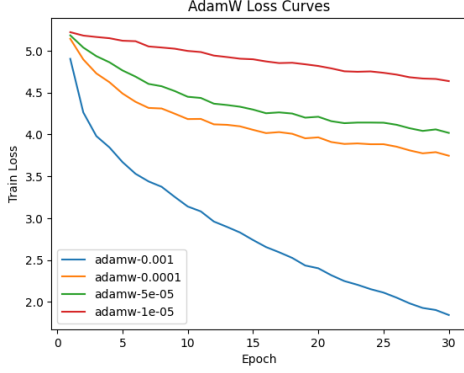
All models were trained for 8 epochs for paraphrase detection and 10 epochs sonnet generation. Each epoch for paraphrase detection took roughly 45 minutes while it took 5 seconds for sonnet generation. The GPT2 dimension used was 768 and the batch size was 32 for the paraphrase task and 12 for the sonnet task. During training I tracked both training loss and development accuracy in order to compare convergence behavior across different random seeds. I also evaluated the final model performance using development accuracy for the paraphrase detection task and chrF score for the sonnet generation task.

The final experiment I performed was taking the optimal learning rate for Shampoo for the paraphrase task and running the training pipeline under various random seeds to determine a confidence interval.

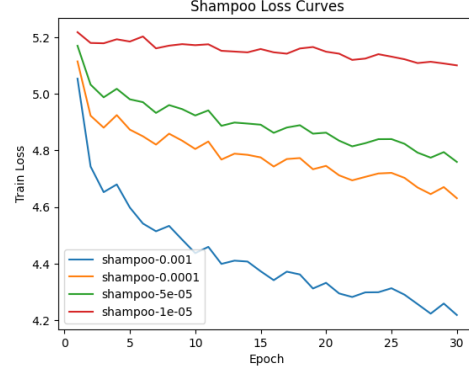
5.4 Results

Optimizer	Learning Rate	Final Dev Accuracy	Final Train Loss
AdamW	10^{-5}	41.63	4.6361
AdamW	5×10^{-5}	30.09	4.0180
AdamW	10^{-4}	32.36	3.7454
AdamW	10^{-3}	39.91	1.8445
Shampoo	10^{-5}	40.07	5.1007
Shampoo	5×10^{-5}	30.50	4.7597
Shampoo	10^{-4}	32.69	4.6315
Shampoo	10^{-3}	36.44	4.2191

Table 1: Learning rate sweep comparing AdamW and Shampoo optimizers on the sonnet generation task.



(a) Adam Loss vs Epoch

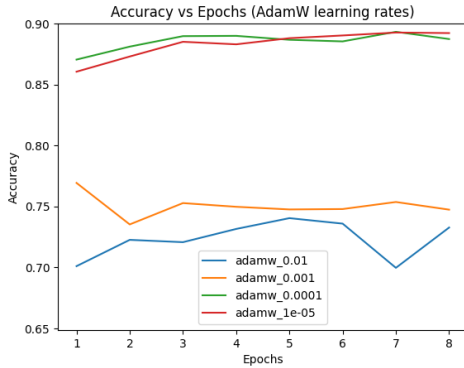


(b) Shampoo Loss vs Epoch

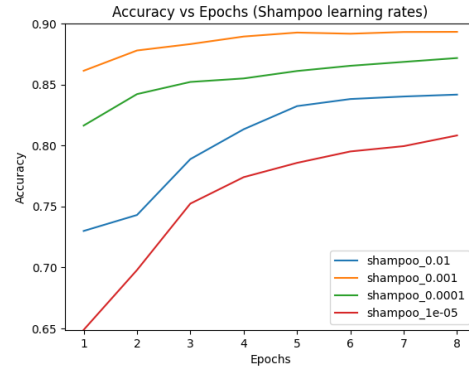
For the Shakespeare sonnet generation task, I evaluated both optimizers using the chrF metric. Table 1 summarizes the final training losses and accuracies across different learning rates. The AdamW baseline achieved a chrF score of 41.519, while Shampoo achieved a lower score of 40.07. These results indicate that Shampoo did not perform as well as AdamW on this generation task under the tested configuration as also seen by the unstable loss curves. Because of this weaker performance, the remainder of the analysis focuses primarily on the paraphrase detection task, where the behavior of the optimizer can be examined more clearly.

Optimizer	Learning Rate	Final Dev Accuracy	Final Train Loss
AdamW	10^{-5}	0.8922	0.1310
AdamW	10^{-4}	0.8873	0.0701
AdamW	10^{-3}	0.7473	0.4813
AdamW	10^{-2}	0.7327	0.5259
Shampoo	10^{-5}	0.8082	0.4186
Shampoo	10^{-4}	0.8718	0.2764
Shampoo	10^{-3}	0.8932	0.1027
Shampoo	10^{-2}	0.8417	0.2591

Table 2: Learning rate sweep comparing AdamW and Shampoo on the paraphrase detection task.



(a) Adam Accuracy vs Epoch



(b) Shampoo Accuracy vs Epoch

For the paraphrase detection, table 2 summarizes the learning-rate sweep for AdamW and Shampoo. The results show that model performance varies significantly across learning rates for both optimizers.

For AdamW, the best-performing configuration used a learning rate of 10^{-5} , achieving a final development accuracy of 0.8922. Other learning rates resulted in lower performance, with the 10^{-3} and 10^{-2} runs finishing at 0.7473 and 0.7327 development accuracy.

For Shampoo, the best-performing configuration used a learning rate of 10^{-3} , reaching a final development accuracy of 0.8932. Lower learning rates produced lower final accuracy, with the 10^{-5} configuration reaching 0.8082 and the 10^{-4} configuration reaching 0.8718. Shampoo with learning rate 10^{-2} reached 0.8417 development accuracy.

Seed	Final Dev Acc	Final Train Loss
14	0.8919	0.1065
297	0.8958	0.0968
1653	0.8932	0.1014
5042	0.8941	0.0993
7788	0.8940	0.0995
11711	0.8936	0.0978
Mean	0.8938	0.0997
Std	0.0015	0.0034

Table 3: Results across multiple random seeds for Shampoo with learning rate 10^{-3} .

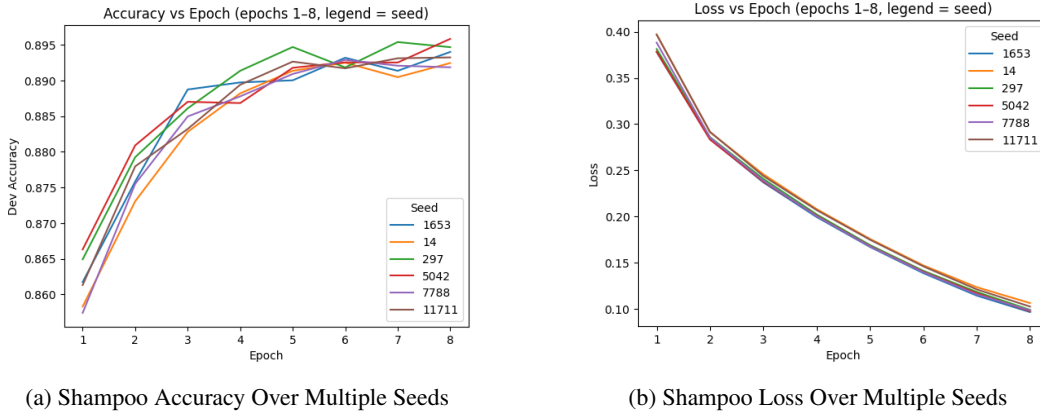


Figure 3: Example detection outputs from the three evaluated models.

To evaluate that success of the optimizer versus the effects of random sampling, I ran the best Shampoo configuration (learning rate 10^{-3}) across multiple random seeds. As shown in Table 3, the final development accuracy across six runs ranged from 0.8919 to 0.8958, with a mean of 0.8938 and a standard deviation of 0.0015. Training loss values showed similarly low variance. The small difference between seeds suggests that the optimization process is stable and that the observed performance is reproducible rather than the result of a favorable random initialization.

6 Analysis

The experiments reveal several differences between AdamW and Shampoo in terms of optimization behavior. The most noticeable difference is the learning-rate order of magnitude in which each optimizer performed the best. For the paraphrase detection task, AdamW achieved the highest development accuracy with a learning rate of 10^{-5} , reaching a final accuracy of 0.8922. In contrast, Shampoo performed better with a higher learning rate of 10^{-3} , achieving a slightly higher development accuracy of 0.8932. This behavior is consistent with the design of the Shampoo optimizer, which rescales gradients using preconditioning and therefore can remain more stable even under larger learning rates.

The learning-rate sweep also highlights clear modes of failure for both optimizers. AdamW performance degraded a lot as the learning rate increased beyond 10^{-4} . In particular, the runs with learning rates of 10^{-3} and 10^{-2} converged poorly, finishing with dev-set accuracies of 0.7473 and 0.7327. Shampoo exhibited a different behavior. Very small learning rates resulted in much slower training, with the 10^{-5} configuration reaching only 0.8082 development accuracy. These results suggest that Shampoo can support larger learning rates making progress per epoch. It is also show that Shampoo can train effectively at smaller learning rates as well, but may require more epochs to achieve the

same level of accuracy as AdamW. Another important observation is the difference in training loss. Under their best-performing configurations, Shampoo achieved a lower final training loss (0.1027) than AdamW (0.1310). This suggests that the curvature-aware updates used by Shampoo may allow the optimizer to move more efficiently through the loss landscape during fine-tuning.

The multi-seed experiments further show that Shampoo training is stable across random initialization. Across six runs with learning rate 10^{-3} , the final development accuracy had a mean of 0.8938 with a standard deviation of 0.0015. The small variance indicates that the optimizer produces consistent results and that the observed performance difference is unlikely to be caused by random fluctuations.

For the Shakespeare sonnet generation task, however, the results showed a different pattern. The AdamW baseline achieved a chrF score of 41.519, while Shampoo achieved a lower score of 40.07. This suggests that the advantages of Shampoo observed in the paraphrase detection task did not transfer as cleanly to the generation task under the tested configuration. For all the learning rates tested, the Shampoo optimizer exhibited oscillations and instability achieving worse loss and chrF scores compared the AdamW. This may be attributed to the smaller dataset and more complex nature of the task.

Overall, these experiments suggest that Shampoo can achieve comparable or slightly better performance than AdamW for the paraphrase detection task when the learning rate is chosen appropriately. However, the results also indicate that optimizer performance is highly sensitive to hyperparameter selection and may vary across tasks.

7 Conclusion

In this project, I investigated whether replacing the AdamW optimizer with the Shampoo optimizer could improve optimization when fine-tuning GPT-2 for specific downstream tasks. I implemented Shampoo within the default project code and evaluated its behavior across a large range of learning rates while keeping the model architecture and training pipeline the same.

The experimental results show that Shampoo can achieve comparable or even better validation results than AdamW on the paraphrase detection task if the learning rate is adequately chosen. The best AdamW configuration achieved a development accuracy of 0.8922, while the best Shampoo configuration achieved 0.8932. In addition, Shampoo achieved a lower final training loss and produced consistent results across multiple random seeds, indicating stable optimization behavior.

However, the results were much less precise for the Shakespeare sonnet generation task. On this task, the AdamW baseline achieved a higher chrF score than Shampoo for all learning rates tested, suggesting that the advantages of second-order preconditioning maybe specific to task prediction type, however, more work is needed to determine the exact cause.

Overall, these experiments suggest that second-order preconditioning methods such as Shampoo can be competitive with standard first-order optimizers when fine-tuning transformer models. At the same time, the results highlight that optimizer performance is strongly influenced by hyperparameter selection and may vary depending on the task.

Future work could explore several directions to better understand the behavior of second-order optimizers. First, it would be useful to evaluate Shampoo on a wider range of tasks beyond paraphrase detection and sonnet generation, including other classification and generation problems, to determine whether the observed behavior generalizes across task types. Second, additional work could focus on improving Shampoo’s performance on generation tasks such as sonnet generation, potentially through better hyperparameter tuning or other model alterations. Third, it would be valuable to compare Shampoo against other second-order optimization methods to better understand how different curvature-aware approaches affect training dynamics. Finally, future experiments could investigate the impact of learning-rate schedulers, which may interact differently with second-order preconditioning and could improve convergence for both optimizers.

References

- [1] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. *ICML*, 2018.

- [2] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.
- [3] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *ICLR*, 2019.
- [4] Shun-ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 1998.
- [5] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. *ICML*, 2015.
- [6] Nikhil Vyas, Depen Morwani, Rosie Zhao, Mujin Kwun, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. Soap: Improving and stabilizing shampoo using adam. *arXiv preprint arXiv:2501.XXXXX*, 2025.