

Does the Optimizer Matter? LoRA vs Full Fine-Tuning in NLP

Stanford CS224N Default Project

Andy Dimnaku

Department of Electrical Engineering
Stanford University
adimnaku@stanford.edu

Abstract

The emergence of optimizers claiming improvements over AdamW, including Lion, MARS, and Muon has largely been evaluated in pretraining settings, leaving their behavior under fine-tuning and parameter-efficient adaptation underexplored. We present a systematic comparison of these four optimizers across full fine-tuning and LoRA parameterizations on two tasks: paraphrase detection using the Quora Question Pairs dataset and sonnet generation on Shakespeare’s sonnets. All optimizers and LoRA are implemented from scratch on top of GPT-2, with per-optimizer learning rate sweeps ensuring fair comparison. Our base configuration achieves 0.893 accuracy and 42.086 chrF on the respective test sets, exceeding staff baselines. Across optimizer experiments, we find that no single optimizer dominates: Lion achieves the highest paraphrase accuracy under full fine-tuning (0.900) while offering the lowest memory and latency, MARS leads on sonnet generation under both parameterizations (43.167 full, 42.723 LoRA), and Muon degrades substantially on classification and at larger model scales. Full fine-tuning generally outperforms LoRA, though the margin is optimizer-dependent. Our results demonstrate that optimizer selection should not be decoupled from the parameterization strategy and that AdamW, while a reliable default, is frequently outperformed by Lion and MARS in fine-tuning regimes.

1 Key Information to include

- Mentor: Advit Deepak
- External Collaborators (if you have any): N/A
- Sharing project: N/A
- Late Days (if you are using any):

Team member	Late days remaining
Andy Dimnaku	3
Total pooled	3
Team size	1
Late days applied for the report	1

2 Introduction

AdamW [1] has long served as the default optimizer for training large language models [2, 3]. However, a wave of new optimizers has emerged claiming significant speedups over AdamW: variance reduction (MARS 4), sign-based momentum (Lion 5), and spectral-domain Newton methods (Muon 6). Recent work in [7] systematically benchmarked these optimizers for pretraining, finding that no single optimizer universally dominates and that many previously reported speedups diminish

under fair comparison. However, their study, and the broader optimizer literature, focuses mainly on pretraining from scratch.

In practice, most practitioners interact with large language models through fine-tuning, and increasingly through parameter-efficient methods such as LoRA [8], which constrains weight updates to a low-rank subspace. Because LoRA fundamentally alters the optimization landscape by restricting the space of achievable updates, optimizer behavior that holds under full fine-tuning may not transfer. Despite this, the choice of optimizer for fine-tuning is rarely studied systematically and is typically defaulted to AdamW regardless of the parameterization scheme.

This paper presents a controlled comparison of four optimizers (AdamW, Lion, MARS, and Muon) across full fine-tuning and LoRA on paraphrase detection (a classification task on the Quora Question Pairs dataset and sonnet generation (an autoregressive language modeling task on Shakespeare’s sonnets). All optimizers are implemented from scratch and evaluated under matched conditions with per-optimizer learning rate sweeps. Beyond comparing final performance, we analyze memory footprints, per-step latency, learning rate sensitivity, and conduct ablations over LoRA rank and model scaling. Our key findings are: (1) no single optimizer dominates across all settings, with Lion and MARS frequently outperforming AdamW. (2) Full fine-tuning generally outperforms LoRA, but the margin is optimizer-dependent. (3) Optimizer differences amplify as model size and LoRA rank increase, underscoring that optimizer selection should not be decoupled from the parameterization strategy.

3 Related Work

Optimizers for Deep Learning. Adam [9] and its weight-decoupled variant AdamW [1] remain the most widely used optimizers for training transformers. Recent alternatives have approached the problem from different angles. Lion [5] was discovered via automated program search and uses only the sign of momentum, reducing memory by half relative to Adam. MARS [4] incorporates variance-reduced gradient estimates into the Adam framework to improve signal quality. Muon [6] applies steepest descent under the spectral norm via polar decomposition of the gradient. Other notable directions include Shampoo [10] and its distributed variants for second-order preconditioning. Despite this proliferation, most optimizers are evaluated in isolation under differing experimental setups, making cross-method comparisons unreliable.

Optimizer Benchmarking. [7] provided the first large-scale systematic comparison of modern optimizers for pretraining, controlling for hyperparameter tuning and evaluating scaling behavior up to 1.2B parameters on the Chinchilla dataset. Their key finding found that optimizer advantages are regime-dependent and many claimed speedups are artifacts of weak baselines. However, their study is limited to pretraining from scratch on a single data mixture and architecture (Llama), leaving fine-tuning regimes unexplored. [11] conducted a broad benchmarking of 15 optimizers across diverse tasks but focused on smaller-scale models and did not consider parameter-efficient fine-tuning.

Parameter-Efficient Fine-Tuning. LoRA [8] has become the dominant approach for parameter-efficient adaptation of large language models, injecting trainable low-rank matrices alongside frozen pretrained weights. Subsequent work has extended this framework through adaptive rank allocation (AdaLoRA [12]), quantized bases (QLoRA [13]), and alternative decompositions. Other parameter-efficient approaches operate in different spaces entirely, such as prefix tuning [14] which prepends learnable tokens to the input. While these methods have been extensively compared in terms of task performance and parameter efficiency, the interaction between the choice of parameter-efficient method and the choice of optimizer has received much less attention. Our work addresses this gap by systematically varying the optimizer under both full and LoRA parameterizations across task types.

4 Approach

This project investigates how the optimizer loss landscape differs between full fine-tuning and LoRA [8], motivated by [7] who showed optimizer performance depends heavily on the training regime with no single optimizer universally dominating when considering pretraining on the chinchilla dataset. Because LoRA restricts weight updates to a low-rank subspace, the effective optimization landscape may differ from full fine-tuning that affects convergence speed, stability, and final performance across

optimizers. Understanding this is important since LoRA is widely used, yet the choice of optimizer is often made without regard to the parameterization scheme.

Model The base model is GPT-2 [15], implemented from scratch following the default project codebase. The implementation includes the standard transformer decoder architecture with multi-head self-attention, layer normalization. For paraphrase detection, the model is extended with a linear classification head that produces a binary prediction from the final hidden state of the last token. For sonnet generation, the model is used as a standard autoregressive language model trained with cross-entropy loss for next token prediction.

Low-Rank Adaptation (LoRA) [8] is a parameter-efficient fine-tuning method that keeps the pretrained weight matrices frozen and injects trainable low-rank decompositions alongside them. For a pretrained weight matrix $W_0 \in \mathbb{R}^{d \times k}$, LoRA parameterizes the update as:

$$h = W_0x + \Delta Wx = W_0x + BAx \quad (1)$$

where $A \in \mathbb{R}^{r \times k}$ and $B \in \mathbb{R}^{d \times r}$ with $\text{rank } r \ll \min(d, k)$. The matrix A is initialized with a random Gaussian ($\mathcal{N}(0, 1)$) and B is initialized to zero, so $\Delta W = BA = 0$ at the start of training and the model begins from the pretrained weights. Only A and B are updated during fine-tuning, reducing the number of trainable parameters from $d \times k$ to $r \times (d + k)$. Following the original paper, we apply LoRA exclusively to the query and value projection matrices (W_Q, W_V) in each attention layer. While subsequent work has explored applying LoRA to additional weight matrices including the MLP layers [8, 16]. The rank r controls the expressivity efficiency tradeoff with lower ranks reduce memory and computation but constrain the space of achievable weight updates.

Optimizers. We compare four optimizers spanning the major categories of modern optimizer design: adaptive moment estimation (AdamW [1]), sign-based momentum (Lion [5]), variance-reduced gradient estimation (MARS [4]), and spectral-domain Newton methods (Muon [6]). All four were implemented from scratch. AdamW serves as the baseline, reflecting its status as the de facto standard for language model training.

AdamW [1] maintains exponential moving averages of the first and second moments of the gradient with decoupled weight decay. Given gradient g_t at step t , the update is:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (2)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (3)$$

$$\alpha_t = \eta \cdot \frac{\sqrt{1 - \beta_2^t}}{1 - \beta_1^t} \quad (4)$$

$$\theta_t = \theta_{t-1} - \alpha_t \frac{m_t}{\sqrt{v_t} + \epsilon} \quad (5)$$

$$\theta_t = \theta_t \cdot (1 - \alpha_t \lambda) \quad (6)$$

where η is the learning rate, λ is the weight decay coefficient, and β_1, β_2 are the moment decay rates. We note that our implementation applies weight decay using the bias-corrected learning rate rather than the base rate η . AdamW stores two state tensors (m_t, v_t) per parameter, resulting in a $2 \times$ memory overhead relative to the model parameters.

Lion [5] was discovered through symbolic program search over a space of optimizer update rules. It uses only the sign of an interpolated momentum term, eliminating the need for second-moment estimation:

$$u_t = \text{sign}(\beta_1 m_{t-1} + (1 - \beta_1) g_t) \quad (7)$$

$$m_t = \beta_2 m_{t-1} + (1 - \beta_2) g_t \quad (8)$$

$$\theta_t = \theta_{t-1} - \eta (u_t + \lambda \theta_{t-1}) \quad (9)$$

where the update direction u_t is computed from an interpolation using β_1 before the momentum m_t is updated using β_2 . Because Lion stores only a single momentum buffer and applies a uniform magnitude update via the sign operation, it requires half the optimizer memory of AdamW and produces faster per-step computation. The sign operation means all parameters receive updates of equal magnitude.

MARS [4] combines Adam-style adaptive updates with variance reduction to improve gradient signal quality. At each step, MARS computes a variance-reduced gradient estimate by correcting the current

gradient using the scaled difference between successive gradients:

$$c_t = g_t + \gamma \cdot \frac{\beta_1}{1 - \beta_1} (g_t - g_{t-1}) \quad (10)$$

$$\tilde{c}_t = \begin{cases} c_t / \|c_t\|_2 & \text{if } \|c_t\|_2 > 1 \\ c_t & \text{otherwise} \end{cases} \quad (11)$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \tilde{c}_t \quad (12)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (13)$$

$$\theta_{t+1} = \theta_t - \eta \frac{m_t}{\sqrt{v_t} + \epsilon} \quad (14)$$

where c_t is the variance-reduced gradient controlled by γ , with the correction term scaled by $\beta_1/(1 - \beta_1)$ to account for the momentum coefficient. The corrected gradient is clipped to unit norm before being fed into the first moment estimate, while the second moment v_t tracks the raw gradient g_t rather than the corrected version. MARS requires storing the previous gradient in addition to the two moment buffers, resulting in higher memory overhead than standard AdamW. There other variants of MARS which is adapted on top of other optimizers, however, for this paper we consider the Adam style above.

Muon [6] applies Newton’s method in the spectral domain by orthogonalizing the momentum update with respect to the matrix spectral norm. The update uses heavy-ball momentum with a Nesterov lookahead before orthogonalization:

$$M_t = \beta M_{t-1} + g_t \quad (15)$$

$$\tilde{g}_t = g_t + \beta M_t \quad (16)$$

$$U_t = \text{NS5}(\tilde{g}_t) \cdot \sqrt{|\tilde{g}_t|} \quad (17)$$

$$\theta_t = \theta_{t-1} - \eta U_t \quad (18)$$

where NS5 denotes five iterations of the Newton–Schulz recurrence with fixed coefficients $(a, b, c) = (3.4445, -4.7750, 2.0315)$, which approximates the orthogonal factor of the polar decomposition. The Nesterov lookahead in $\tilde{g}_t$ extrapolates the momentum forward before orthogonalization, improving convergence. The $\sqrt{|\tilde{g}_t|}$ scaling factor normalizes the update to be approximately RMS-consistent across weight matrices of different sizes. Like Lion, Muon stores only a single momentum buffer, yielding low memory allocation. However, the matrix operations involved in the Newton–Schulz iterations and Nesterov step make it the slowest optimizer per step.

5 Experiments

5.1 Data

We considered a *paraphrase detection* and a *sonnet generation* for the experiments which originated from the default project specifications. For *paraphrase detection*, I use the Quora Question Pairs dataset, which contains 141,506 training examples, 20,215 development examples, and 40,431 test examples. Each example consists of a pair of questions with a binary label indicating whether they are paraphrases of one another. For *sonnet generation*, I use the Shakespeare Sonnets dataset, which contains 154 sonnets split into 143 training examples and 12 test examples. The task is to generate the remaining lines of a sonnet given its first three lines. Due to these data sets differing it gives a diversity in dataset size and objective.

5.2 Evaluation method

We will be considering 2 metrics to quantifying the performance for the prior datasets. For the *paraphrase detection* dataset, we will use accuracy given the nature of the binary labels of this dataset. For *sonnet generation*, the goal is given the first 3 lines of a 12 line sonnet to closely generate language as close to the original sonnet as possible. So we are using the chrF metric [17], which is a character level n-gram comparison metric similar to BLEU [18].

5.3 Experimental details

For both datasets I initially built strong initial hypermeter pipeline setups models which were used to beat the STAFF-Baselines before comparing optimizers. All experiments are built on GPT-2 Medium [15], a decoder-only transformer with 355M parameters, 24 layers, a hidden dimension of 1,024, and 16 attention heads. Models are trained on a single NVIDIA L40S GPU 48 GB GPU using BF16 mixed-precision via torch.autocast, which reduces memory footprint while maintaining training stability. The AdamW optimizer is used as the baseline optimizer across both tasks, with checkpoints selected based on best validation performance rather than final epoch. All experiments are run built off the default project with task-specific training and evaluation scripts.

Paraphrase Detection For paraphrase detection, the full model is fine-tuned using a classification objective. Each input is formed by concatenating the two sentences, and the hidden state of the final token is projected to a 2-class logit via a linear head. The model is trained with cross-entropy loss, a batch size of 32, a learning rate of $6e-5$, and for 2 epoch. The 2 epochs were sufficient due to the large dataset size. No LoRA adapters are used and all parameters are updated.

Sonnet Generation For sonnet generation, the model is fine-tuned using a causal language modeling objective through a cross-entropy loss computed over all next-token prediction positions. Given the extremely small dataset size, full fine-tuning risks rapid overfitting, so we use LoRA adapters [8] applied to the query and value projection matrices of all 24 attention layers with rank $r=16$ and $\alpha=16$ (scale=1) with roughly 1.57M trainable parameters (0.44% of total). The model is trained with AdamW at a learning rate of $3e-4$, batch size 16, for 10 epochs, with the checkpoint achieving the best development ChrF score saved. Text generation at inference uses beam search with 5 beams, top-p nucleus sampling ($p=0.9$), temperature 0.7, a repetition penalty of 1.3, a no-repeat n-gram constraint of order 3, and a minimum generation length of 100 tokens. Development quality is measured using ChrF (character 6-gram F-score with $\beta=2$) computed against held-out reference sonnets on the development set.

Optimizer Experiments For optimizer experiments we conducted a systematic sweep across four optimizers, AdamW, Lion, MARS, and Muon, applied under two parameterization regimes (full fine-tuning and LoRA with $r=16$, $\alpha=16$, scale=1) for both tasks. All other hyperparameters were held fixed at the values established in the base setup above. Each optimizer was swept over a range of 5 learning rates spanning roughly two orders of magnitude, with ranges chosen to reflect each optimizer’s characteristics. Specifically, for full fine-tuning the learning rate grids were: AdamW $\in \{1e-5, 3e-5, 1e-4, 3e-4, 1e-3\}$, Lion $\in \{1e-6, 3e-6, 1e-5, 3e-5, 1e-4\}$, MARS $\in \{1e-5, 3e-5, 1e-4, 3e-4, 1e-3\}$, and Muon $\in \{1e-4, 3e-4, 1e-3, 3e-3, 1e-2\}$. For the LoRA parameterization higher learning rates were used: AdamW $\in \{1e-4, 3e-4, 1e-3, 3e-3, 1e-2\}$, Lion $\in \{1e-5, 3e-5, 1e-4, 3e-4, 1e-3\}$, MARS $\in \{1e-4, 3e-4, 1e-3, 3e-3, 1e-2\}$, and Muon $\in \{1e-3, 3e-3, 1e-2, 3e-2, 1e-1\}$. For all other none specified parameters the prior mentioned were used. AdamW uses momentum coefficients $\beta_1 = 0.9$ and $\beta_2 = 0.999$ with $\epsilon = 10^{-6}$ and weight decay $\lambda = 0.1$. Lion uses $\beta_1 = 0.9$ and $\beta_2 = 0.99$ with weight decay $\lambda = 0.1$. MARS uses $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, variance reduction coefficient $\gamma = 0.025$, and weight decay $\lambda = 0.1$. Muon uses momentum $\beta = 0.95$ with Nesterov lookahead enabled and 5 Newton–Schulz iterations.

5.4 Results

Using the initial setup described above, we evaluated our base model on the development and test sets for both Paraphrase Detection and Sonnet Generation, with results shown in Table 1. Our base configuration outperforms the staff baselines across all four evaluation splits, establishing a strong foundation for the subsequent optimizer experiments. Building on this baseline, we conducted optimizer experiments across both full fine-tuning and LoRA parameterizations, with the best development set performance for each configuration reported in Table 2.

Full Fine-Tuning vs. LoRA Across nearly all optimizer and task combinations, full fine-tuning outperforms LoRA, with the exception of Muon on Sonnet Generation, where LoRA achieves 42.491 chrF compared to 38.098 under full fine-tuning. This is a surprising result due to the small size of the sonnet training set of 143 examples expecting full fine-tuning to overfit. Although, full fine-tuning does mostly outperform LoRA requires significantly less memory getting close to similar performance.

Table 1: Results on Paraphrase Detection and Sonnet Generation tasks for both the development (Dev) and test sets with their comparisons to the teaching staff baselines. Best results per split are shown in **bold**.

Task	Split	Ours	STAFF-BASELINE
Paraphrase Detection	Dev	0.895	0.882
	Test	0.893	0.856
Sonnet Generation	Dev	42.951	41.414
	Test	42.086	41.208

Table 2: Best performance of each optimizer across fine-tuning strategies. We report accuracy on the development set for Paraphrase Detection and chrF for Sonnet Generation. **Bold** denotes the best result per column.

Task	Method	Optimizer			
		AdamW	Lion	MARS	Muon
Paraphrase Detection (Acc.↑)	Full Fine-Tuning	0.895	0.900	0.899	0.635
	LoRA	0.878	0.873	0.865	0.425
Sonnet Generation (ChrF↑)	Full Fine-Tuning	42.585	42.398	43.167	38.098
	LoRA	42.473	42.323	42.723	42.491

Optimizer Comparison No single optimizer dominates across all settings. Lion achieves the highest accuracy on Paraphrase Detection under full fine-tuning (0.900), while AdamW leads under LoRA (0.878). For Sonnet Generation, MARS obtains the best chrF scores under both parameterizations (43.167 full, 42.723 LoRA), outperforming the AdamW baseline in both cases. This is notable given that AdamW remains the default optimizer for LLM training.

Muon presents the most distinctive behavior. On Paraphrase Detection it performs substantially worse than the other optimizers, achieving only 0.635 accuracy under full fine-tuning and 0.425 under LoRA. This degradation most likely comes from its use of the spectral norm for steepest descent updates, which produces more aggressive parameter changes that may prevent convergence on the classification objective with the optimizer originally being made for pretraining. On Sonnet Generation, Muon’s LoRA performance (42.491) recovers to be comparable with the other optimizers, suggesting that it performs much better with predicting next token regime of the sonnet generation task.

6 Analysis

Due to their different update behaviors, different optimizers have vastly different learning behaviors, memory requirements, and latencies. In order to understand them deeper we measured the comparisons for each of the with the results being in 3.

Learning Rate Sensitivity. Interestingly, the optimal learning rates in Table 3 span up to two orders of magnitude across optimizers within the same setting. Lion consistently requires the lowest rates (e.g., 3×10^{-6} for Paraphrase full fine-tuning) due to its sign-based update rule applies uniform magnitude steps regardless of gradient scale. Muon favors the largest rates, peaking at 1×10^{-2} for Sonnet full fine-tuning, consistent with its spectral-norm steepest descent. Two systematic trends can be seen with LoRA configurations favor higher rates than their full fine-tuning counterparts across all optimizers on paraphrase. Also, sonnet Generation selects higher rates than Paraphrase Detection, likely due to the significantly smaller training set and fewer steps per epoch demanding more aggressive updates within the limited budget.

Memory and Latency. Under full fine-tuning (Table 3), MARS has the highest memory allocation (~ 8 GiB), approximately $1.5\times$ that of AdamW (~ 5.3 GiB), due to maintaining variance-reduced gradient estimates on top of standard moment buffers, while Lion and Muon achieve the lowest allocation (~ 4 GiB) since neither maintains second-moment state. For latency, Lion is the fastest

Table 3: Per-step latency, memory footprint, and selected learning rate across optimizers for Paraphrase Detection and Sonnet Generation. Full denotes full fine-tuning; LoRA denotes low-rank adaptation. All runs use a single GPU.

Task	Mode	Optimizer	lr	Step Time (s) ↓	Allocated (GiB) ↓	Peak GPU (GiB) ↓
Paraphrase	Full	AdamW	3e-5	0.516	5.32	25.47
		Lion	3e-6	0.288	3.99	24.14
		MARS	3e-5	0.515	7.96	28.10
		Muon	3e-4	1.014	4.00	24.14
	LoRA	AdamW	1e-4	0.192	1.37	19.10
		Lion	1e-5	0.104	1.36	19.09
		MARS	3e-4	0.178	1.38	19.11
		Muon	1e-3	0.171	1.36	19.09
Sonnet	Full	AdamW	1e-3	0.275	5.41	10.70
		Lion	1e-4	0.479	4.09	9.38
		MARS	1e-3	0.455	8.05	13.34
		Muon	1e-2	0.933	4.09	9.38
	LoRA	AdamW	1e-3	0.173	1.46	7.10
		Lion	3e-4	0.152	1.46	7.10
		MARS	1e-3	0.156	1.47	7.11
		Muon	1e-3	0.196	1.46	7.10

Table 4: LoRA rank ablation on Sonnet Generation (chrF↑, Dev) using GPT-2 Medium. We sweep rank $r \in \{1, 8, 16, 32, 64, 128\}$ across all four optimizers. **Bold** denotes the best optimizer per rank.

Rank	Trainable	% Total	Optimizer			
			AdamW	Lion	MARS	Muon
$r = 1$	0.098M	0.03%	40.690	40.579	40.394	40.162
$r = 8$	0.786M	0.22%	42.377	42.486	42.445	42.042
$r = 16$	1.573M	0.44%	42.473	42.323	42.723	42.491
$r = 32$	3.146M	0.89%	41.838	43.055	42.504	42.894
$r = 64$	6.291M	1.77%	42.832	42.784	43.006	41.516
$r = 128$	12.583M	3.55%	40.536	42.569	29.043	41.969

optimizer at 0.288s per step under Paraphrase full fine-tuning, $1.8\times$ faster than AdamW and $3.5\times$ faster than Muon, which is consistently the slowest (1.014s) due to the cost of computing polar decompositions at each step. Switching to LoRA reduces allocated memory by $3\text{--}4\times$ across all optimizers and compresses the latency gap substantially (0.104–0.196s on Paraphrase), as the dominant cost shifts to the shared frozen forward pass. However, peak GPU memory remains high (~ 19 GiB on Paraphrase) since the full base model is still held in memory with a large batch size of 16.

Scaling LoRA Rank To further investigate how optimizer behavior interacts with LoRA parameterization, we conducted a rank ablation study sweeping $r \in \{1, 8, 16, 32, 64, 128\}$ on Sonnet Generation. All hyperparameters and pipeline configurations remain identical to those in Table 2, with only the rank varied, isolating its effect on cross-optimizer performance. The results are reported in Table 4.

All four optimizers exhibit a consistent inverted-U trend across rank, performing worst at the extremes ($r = 1$ and $r = 128$) and peaking in the $r = 32\text{--}64$ range. At $r = 1$, the trainable subspace (0.098M parameters, 0.03% of the model) is too restrictive for any optimizer to adequately adapt the model, with all scores clustering around 40.2–40.7. Performance improves sharply at $r = 8$ and plateaus through the mid-ranks, where Lion peaks at $r = 32$ (43.055) and MARS at $r = 64$ (43.006). At $r = 128$ (12.6M parameters, 3.55%), performance degrades, most dramatically for MARS, which collapses. Comparing across optimizers, Lion and MARS outperform the AdamW baseline at most ranks, with Lion proving the most robust across the full sweep, maintaining scores above 42.3 for all $r \geq 8$. Muon is competitive in the mid-ranks but shows greater variance.

Table 5: Model size ablation on Sonnet Generation (chrF $\uparrow$, Dev) using LoRA with $r = 16$. **Bold** denotes the best optimizer per model size.

Model	Total Params	Trainable	Optimizer			
			AdamW	Lion	MARS	Muon
GPT-2 Base	124.4M	1.18M (0.95%)	42.212	42.610	42.406	42.398
GPT-2 Medium	354.8M	3.15M (0.89%)	42.473	42.323	42.723	42.491
GPT-2 Large	774.0M	5.90M (0.76%)	40.779	42.303	40.165	32.061
GPT-2 XL	1,557.6M	9.83M (0.63%)	42.331	42.701	43.208	28.457

Scaling Model Size To further investigate how optimizer behavior interacts with model scale, we conducted an ablation study varying the backbone architecture across GPT-2 Base (124.4M), Medium (354.8M), Large (774.0M), and XL (1,557.6M) on Sonnet Generation with LoRA $r = 16$. All hyperparameters and pipeline configurations remain identical to those in Table 2, with only the model size varied, isolating its effect on cross-optimizer performance. The results are reported in Table 5.

At Base scale, all four optimizers perform comparably (42.2–42.6 chrF), with Lion narrowly beating at 42.610. As models scale up, optimizer differences become more pronounced. Lion is the most stable across all sizes, never dropping below 42.3, and achieves the best score at both Base and Large. MARS shows the highest ceiling overall, reaching 43.208 at XL, the single best result in the ablation, but is inconsistent dropping to 40.165 at Large. AdamW follows a similar non-monotonic pattern, dipping to 40.779 at Large before recovering at XL (42.331). The most interesting finding is Muon’s sharp degradation at larger scales: from competitive performance at Base (42.398) and Medium (42.491) to 32.061 at Large and 28.457 at XL, suggesting that Muon become increasingly unstable as the frozen parameter space grows while the trainable proportion shrinks. Across optimizers, no single model size is the best with AdamW, Muon, MARS, and Lion receiving their peaks at Medium, Medium, XL, and XL respectively. Further ablation experiment about the qualitative differences of the learning curves can be seen in the Appendix A.2 and A.3 on sonnet generation for LoRA and full finetuning respectively.

7 Conclusion

We presented a systematic comparison of four optimizers (AdamW, Lion, MARS, and Muon) across full fine-tuning and LoRA parameterizations on Paraphrase Detection and Sonnet Generation. Our findings show that no single optimizer dominates across all settings: Lion achieves the best accuracy on Paraphrase Detection under full fine-tuning while offering the lowest latency and memory footprint, MARS consistently leads on Sonnet Generation under both parameterizations, and AdamW remains a competitive default. Muon, despite its strong prior performances on pretraining, exhibits instability on classification tasks and degrades sharply at larger model scales. Our LoRA rank ablation reveals a consistent inverted-U pattern across all optimizers, with optimal performance in the $r = 32$ –64 range, while the model scaling ablation shows that optimizer differences amplify with model size, with Lion proving the most robust across scales. This brings into question that AdamW should not necessary be the standard optimizer for post-training with Lion and MARS also being strong competitors with Lion also introducing memory benefits.

Several limitations of this work suggest directions for future investigation. Our hyperparameter sweeps were constrained by compute budget. A more thorough grid or Bayesian search could shift the relative rankings. All experiments used the GPT-2 architecture family, and it remains open whether these findings transfer to other architectures or to tasks requiring longer reasoning trajectories such as chain-of-thought reasoning. Finally, we considered only LoRA as a parameter-efficient method, which operates in weight space and an exciting direction is exploring alternative parameterizations such as prefix tuning, which operates in token space, could reveal interesting optimizer-parameterization.

Team contributions

Since I worked alone, all parts of the projects have been implemented by myself from the base default project, LoRA, additional optimizers, experiments, and report/poster writing.

References

- [1] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.
- [2] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yearly, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kam-badur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collet, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Conguet, Virginie Do, Vish Vogeti, Vitor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuwei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkan Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank

Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabisa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024.

- [3] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiayi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025.
- [4] Huizhuo Yuan, Yifeng Liu, Shuang Wu, Xun Zhou, and Quanquan Gu. Mars: Unleashing the power of variance reduction for training large models, 2025.
- [5] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms, 2023.
- [6] Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024.

- [7] Kaiyue Wen, David Hall, Tengyu Ma, and Percy Liang. Fantastic pretraining optimizers and where to find them, 2025.
- [8] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.
- [9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.
- [10] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization, 2018.
- [11] Robin M. Schmidt, Frank Schneider, and Philipp Hennig. Descending through a crowded valley - benchmarking deep learning optimizers, 2021.
- [12] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning, 2023.
- [13] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023.
- [14] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation, 2021.
- [15] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.
- [16] John Schulman and Thinking Machines Lab. Lora without regret. *Thinking Machines Lab: Connectionism*, 2025. <https://thinkingmachines.ai/blog/lora/>.
- [17] Maja Popović. chrF: character n-gram f-score for automatic mt evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal, 2015. Association for Computational Linguistics.
- [18] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin, editors, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

A Appendix

A.1 Sentiment Classification Experiments

We evaluate GPT-2 fine-tuning under two regimes on the SST (5-class) and CFIMDB (binary) sentiment classification datasets. In the **last-linear-layer** setting, all GPT-2 parameters are frozen and only the task-specific classification head is trained. In the **full-model** setting, all parameters including GPT-2 are updated end-to-end. Both pipelines use cross-entropy loss and are optimized with AdamW. At each epoch, the model is evaluated on the development set, and only the checkpoint achieving the highest dev accuracy is saved, ensuring the reported results reflect the best generalization rather than the final epoch. The classification head applies dropout before a linear projection from the hidden state of the last token to the label space.

Table 6: Hyperparameters for sentiment classification experiments.

Hyperparameter	Last-Linear-Layer	Full-Model
Learning Rate	1×10^{-3}	1×10^{-5}
Batch Size	64	8
Dropout	0.3	0.3
Epochs	10	10
Optimizer	AdamW	AdamW
Checkpoint Selection	Best Dev Acc	Best Dev Acc

Table 7: Dev set accuracy for sentiment classification on SST and CFIMDB.

Dataset	Fine-Tune Mode	Dev Accuracy
SST	Last-Linear-Layer	0.466
SST	Full-Model	0.520
CFIMDB	Last-Linear-Layer	0.849
CFIMDB	Full-Model	0.980

A.2 Sonnet Generation LoRA Curves



Figure 1: Training curves for Sonnet Generation with LoRA across all four optimizers.

From the best runs achieved for each optimizer as in Table 2, the corresponding training curves on Sonnet Generation with LoRA are shown in Figure 1. AdamW, Lion, and MARS exhibit similar training loss trajectories, decreasing steadily over the course of training (Figure 1b). Muon, by contrast, makes minimal progress on training loss, plateauing around a loss of ~ 6 . Despite this, Muon achieves comparable development chrF scores (Figure 1a), suggesting it still learns effective generation patterns even without substantially reducing the training objective. For the other three optimizers, development chrF and validation loss (Figures 1a, 1c) remain relatively stable after the initial epochs.

A.3 Sonnet Generation Full Finetuning Curves

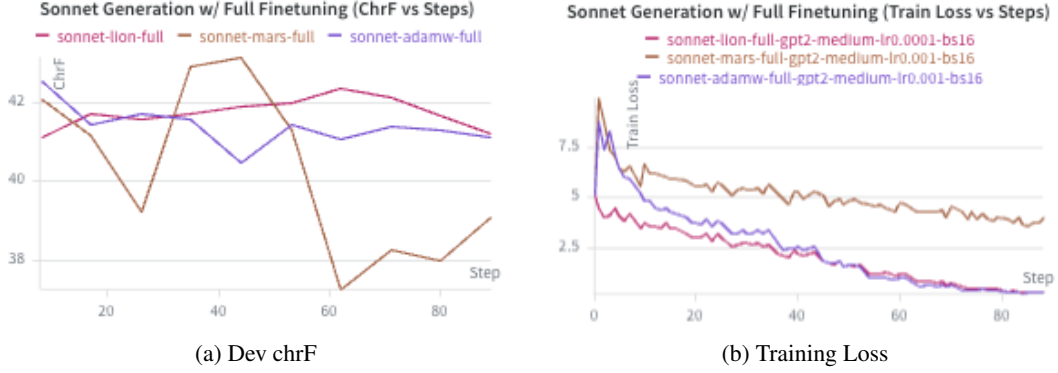


Figure 2: Training curves for Sonnet Generation with full fine-tuning across all four optimizers.

The corresponding full fine-tuning curves on Sonnet Generation are shown in Figure 2, with Muon omitted due to its substantially higher loss and lower chrF making it difficult to visualize trends among the remaining optimizers. Unlike the LoRA setting, clearer differences emerge between optimizers in the training dynamics. Lion and AdamW decrease their training loss at similar rates (Figure 2b), while MARS reduces its training loss more slowly, not reaching the same depth as the other two. Despite this slower training loss reduction, MARS achieves a notably higher peak development chrF around step 50 (Figure 2a), substantially outperforming both Lion and AdamW at that point. This contrast, lower training loss not translating to better generation quality, highlights an interesting difference in generalization behavior across optimizers, where MARS appears to extract more transferable representations per unit of training loss reduction. Lion and AdamW maintain more stable development chrF trajectories throughout training, consistent with their steady and similar convergence profiles.