A Minimalist Optimizer Design for LLM Pretraining

Athanasios Glentis[*1] **Jiaxiang Li**[*1] **Andi Han**[2] **Mingyi Hong**[1]

Abstract

Training large language models (LLMs) typically relies on adaptive optimizers such as Adam, which require significant memory to maintain first- and second-moment matrices, which are known as *optimizer states*. While recent works such as GaLore, Fira and APOLLO have proposed state-compressed variants to reduce memory consumption, a fundamental question remains: What is the *minimal* amount of optimizer state that is truly necessary to retain state-of-the-art performance in LLM pretraining? In this work, we systematically investigate this question using a bottom-up approach. We find that two (memory- and compute-efficient) optimization techniques are particularly effective: (1) column-wise gradient normalization significantly boosts the performance of plain SGD without requiring momentum; and (2) adding first-order momentum only to the output layer – where gradient variance is highest – yields performance competitive with fully adaptive methods such as Muon. Based on these insights, we propose `SCALE` (Stochastic Column-normAlized Last-layer momEntum), a new optimizer that combines column-normalized SGD with last-layer momentum, where column normalization refers to normalizing the gradient along the output dimension. Across multiple LLaMA models (60M–1B), `SCALE` matches or exceeds the performance of Adam while using only 35–45% of the total memory. It also consistently outperforms memory-efficient optimizers such as GaLore, Fira and APOLLO, making it a strong candidate for large-scale pretraining under memory constraints. Code is available at this link.

[*]Equal contribution [1]Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, USA [2]School of Mathematics and Statistics, University of Sydney. RIKEN AIP. Correspondence to: Athanasios Glentis <glent007@umn.edu>, Jiaxiang Li <li003755@umn.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

1. Introduction

Adaptive optimizers such as RMSProp (Hinton et al., 2012), Adam (Kingma & Ba, 2015) are the default optimizers for large-scale attention-based deep neural networks such as large language models (LLMs). While effective, these optimizers incur significant memory overhead due to their reliance on maintaining both first- and second-moment estimates of the gradient, which are also known as *optimizer states*. For example, Adam requires storing two additional tensors per parameter tensor, tripling the memory usage compared to vanilla stochastic gradient descent (SGD).

On the other hand, despite its superior memory efficiency, vanilla SGD performs poorly when applied directly in LLM training, due to the absence of adaptive scaling in the update step (See Figure 3 for experiment results, also see Zhao et al. (2025); Zhang et al. (2020)). This has motivated a wave of recent research (Zhao et al., 2024; Zhang et al., 2024; Liu et al., 2024; Xu et al., 2024; Pethick et al., 2025; Ma et al., 2024) focused on developing memory-efficient alternatives to Adam that aim to retain its performance while reducing memory consumption. Despite the growing literature, there has been no systematic study to identify which specific algorithmic components are most essential for designing highly-performed yet minimal-memory optimizers. This motivates our central research question:

> What is the minimum subset of optimizer states beyond the SGD that is required to achieve state-of-the-art pretraining performance?

In this work, we pursue this question through a bottom-up, minimalist approach. Concretely, we: **(1)** Identify two key components that enhance the performance of vanilla SGD: gradient normalization and momentum; **(2)** Analyze how these components can be adapted in the most memory-efficient manners;**(3)** Justify our design choices through a combination of theoretical insights and empirical evidence.

Our study suggests that two techniques, when used together, are particularly effective: *(i)* **column-wise gradient normalization**, which improves SGD performance significantly without requiring momentum; and *(ii)* adding **first-order momentum exclusively to the output layer**, where gradient variance is highest. These insights lead us to pro-

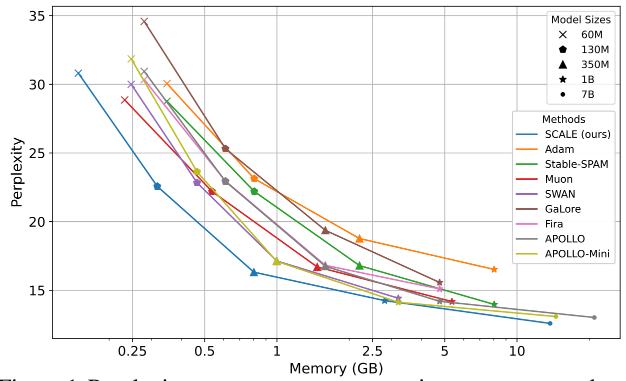


Figure 1. Perplexity v.s. memory consumption among a number of SOTA algorithms. Solutions achieved towards the left-bottom side of the plot represent better performance/memory trade-off (see Appendix A.3 for the details of the memory estimation).

pose SCALE (Stochastic Column-normAlized Last-layer momEntum), a minimalist optimizer that combines these two techniques, which requires roughly the same amount of memory as compared to vanilla SGD. For example, for 1B (resp. 7B) model, SCALE only requires 10% (resp. 2%) more memory as compared to vanilla SGD. Meanwhile, SCALE achieves competitive performance as compared with SOTA optimizers Adam and Muon, with only 35% and 52% of the memory cost for training 1B models, respectively. Please see Fig. 1 for an illustration of performance and memory trade-off among different SOTA algorithms.

2. Methodology

Denote the optimization problem of LLM as

$$\min_{\theta=[\theta_1,\ldots,\theta_L]} \ell(\theta) := \frac{1}{n}\sum_{i=1}^{n} \ell(\theta;\xi_i) \tag{2.1}$$

where ℓ is the loss function, $\theta = [\theta_1, ..., \theta_L]$ is the model trainable parameters, with θ_l the l-th layer, $l = 1, 2, ..., L$. With attention-based network, we can simply assume that each $\theta_l \in \mathbb{R}^{d_{l,\text{in}}\times d_{l,\text{out}}}$ is a weight matrix, with the input dimension $d_{l,\text{in}}$ and output dimension $d_{l,\text{out}}$. Here ξ_i with $i = 1, ..., n$ represents training data samples.

To solve (2.1), vanilla SGD draws a small batch of i.i.d samples $\{\xi_{t,b}\}_{b=1,..,B}$ at iteration t and performs update toward the negative stochastic gradient direction $g^t := \frac{1}{B}\sum_{b=1}^{B} \nabla\ell(\theta^t;\xi_{t,b})$, where B is batch size and η_t is the learning rate. Although memory-efficient, SGD performs poorly in LLM training due to the lack of adaptive scaling (Zhao et al., 2025), and we verify this in Fig. 3 where we run SGD and Adam on LLaMA 60M pretraining task.

In contrast, adaptive algorithms such as Adam (Kingma & Ba, 2015) update parameters using a more sophisticated scheme (where $\odot$ represents element-wise product):

$$\theta^{t+1} = \theta^t - \eta_t m^t/(\sqrt{v^t}+\epsilon), \text{ where}$$
$$m^t = \beta_1 m^{t-1} + (1-\beta_1)g^t, v^t = \beta_2 v^{t-1} + (1-\beta_2)g^t \odot g^t. \tag{2.2}$$

We examine Adam via the two essential components: The first is **Gradient Normalization**. The key difference of Adam from SGD is the normalization factor v^t in the denominator. One could first normalize each element of SGD, resulting in the sign-SGD update as follows:

$$\theta^{t+1} = \theta^t - \eta_t \frac{g^t}{\sqrt{g^t \odot g^t}} = \theta^t - \eta_t\, \text{sign}(g^t) \tag{2.3}$$

where sign stands for taking the sign for each element; The second is **Exponential Moving Average (EMA)**. The stochasticity of the mini-batch sample $\{\xi_{t,b}\}_{b=1,..,B}$ of g^t could be smoothed by taking the exponential moving average (EMA) for both the numerator and denominator for the update (2.3), resulting in the second line of (2.2) for updating the numerator and denominator. In the next sections, we examine each of these components separately and discuss their effectiveness in isolation. We start from the gradient normalization, which is memory-free. Then we discuss how to use EMA in a more memory-friendly way to boost the performance of the optimizer.

2.1. Gradient normalization

In view of how sign-SGD plays as the key connecting SGD and Adam, in this section we inspect the effect of different gradient normalizations. Denote G^t the stochastic gradient of current iteration (we use upper letters since we assume the weight blocks θ_l in (2.1) are matrices). Different normalization techniques arise from using different matrix norms in the following steepest descent formulation:

$$\Theta^{t+1} = \Theta^t + \eta_t\Delta^t, \quad \Delta^t = \operatorname*{argmin}_{\|\Delta\|\le\rho} \langle G^t, \Delta\rangle \tag{2.4}$$

where $G^t \in \mathbb{R}^{m\times n}$ denotes the (stochastic) gradient at iteration t. In particular, we let $\|A\|_{p\to q}$ to represent the operator norm of matrix A induced by vector norm ℓ_p and ℓ_q.We can derive the following results based on (Bernstein & Newhouse, 2024):

$$\begin{aligned}
&\operatorname*{argmin}_{\|\Delta\|_{2\to2}\le1} \langle G, \Delta\rangle = -UV^\top, \quad G = U\Sigma V^\top \text{ is the SVD},\\
&\operatorname*{argmin}_{\|\Delta\|_{1\to2}\le1} \langle G, \Delta\rangle = -\left[\frac{\text{col}_1(G)}{\|\text{col}_1(G)\|}, ..., \frac{\text{col}_n(G)}{\|\text{col}_n(G)\|}\right],\\
&\operatorname*{argmin}_{\|\Delta\|_{2\to\infty}\le1} \langle G, \Delta\rangle = -\begin{bmatrix} \frac{\text{row}_1(G)}{\|\text{row}_1(G)\|} \\ \cdots \\ \frac{\text{row}_m(G)}{\|\text{row}_m(G)\|}\end{bmatrix},\\
&\operatorname*{argmin}_{\|\Delta\|_{1\to\infty}\le1} \langle G, \Delta\rangle = -\,\text{sign}(G),
\end{aligned} \tag{2.5}$$

which we refer to as *singular-value*, *column-wise*, *row-wise* and *sign normalization*, respectively.

It is worth mentioning that multiple existing works can be summarized in this gradient normalization framework. For example, Sign-SGD/Adam utilize $\|\cdot\|_{1\to\infty}$ (*sign normalization*), Muon (Jordan et al., 2024) utilizes $\|\cdot\|_{2\to 2}$ norm (*singular-value normalization*), whereas SCION (Pethick et al., 2025) and SlimAdam (Kalra et al., 2025) apply different norms for different layers. In Appendix A.5, we compare the computation of different normalizations in (2.5), where singular-value normalization is most time consuming.

Experimental insights. We conduct the preliminary experiment on pretraining LLaMA models (See Section 4 for the experiment setting) to test SGD (2.4) with different normalizations (applying to all the layers) as specified in (2.5). We report the pretraining perplexities in Table 1, and we notice that all the normalizations improve over SGD, however none of them alone could match the performance of Stable-SPAM (which is a stabilized version of AdamW) or Muon, two of the state-of-the-art algorithms. In particular, singular-value normalization and column-wise normalization demonstrate better performance than row-wise and sign normalization. Given its lower cost, we adopt column-wise normalization as our base algorithm moving forward.

	60M	**130M**	**350M**
Adam	30.05	23.13	18.77
Stable-SPAM	28.77	22.20	16.80
Muon	28.86	22.20	16.70
singular-value	34.15	25.25	18.73
column-wise	39.89	28.85	20.38
row-wise	79.27	37.67	21.63
sign	54.36	40.42	27.95

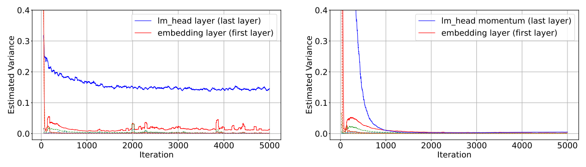
Table 1. Preliminary experiments results (perplexity) of pretraining LLaMA models on C4 dataset, using different norms for steepest gradient descent (2.4) methods as specified in (2.5). For the singular-value normalization, we use the inexact Newton-Schulz (NS) iteration (again see Jordan et al. (2024) for details) for fast approximation.

2.2. Momentum in the last layer

So far, gradient normalization has been applied uniformly across all layers. We now investigate layer-specific momentum, motivated by the hypothesis that not all layers benefit equally from exponential moving average (EMA).

Layer-wise gradient variance. To identify the most important layer which provides the largest performance gain when incorporating momentums, we conjecture that the stochastic gradients of different layers have different variances, and layers with higher gradient variances require momentums more than layers with lower gradient variances.

We first conduct a simple experiment on LLaMA 130M to check the gradient variance of different layers. To estimate the gradient variance, one need the full gradient (by input the entire training dataset) which is not practicable. Instead, we take a much larger training batch as input[1] to estimate the



(a) SGD-col-norm (b) SGD-col-norm-mmt-last

Figure 2. Estimated variance of the stochastic gradients (and momentum when applicable) for different layers in two methods (smoothed by 50 iterations window). We observe that when running SGD with column-wise normalization (SGD-col-norm, left plot), the variance of the last layer (lm_head) is largest for most of the time, following by the variance of the first layer (embedding) and other layers. After applying momentum to the last layer (SGD-col-norm-mmt-last, right plot), the variance of the momentum of last layer (lm_head momentum) decreases to a very low level. Interestingly, the variance of the first layer in plot (b) is also smaller than the one in plot (a).

true gradient. The experiment results are shown in Figure 2. We observe that the variance of the last layer is largest for most of the time, followed by the variance of the first layer then other layers.

Next, we show theoretically that momentum helps the most for the layers with larger gradient variances. We inspect the theoretical property of applying SGD with momentum (SGD-M) to the LLM optimization problem (2.1). Consider the following SGD-M algorithm to solve (2.1):

$$\begin{aligned} m_l^t &= \beta_l m_l^{t-1} + (1-\beta_l) g^t,\ g^t = \nabla_{\theta_l}\ell(\theta^t;\xi_t) \\ \theta_l^{t+1} &= \theta_l^t - \eta_l m_l^t \end{aligned} \tag{2.6}$$

where $l = 1, \ldots, L$ represents different layers, i.e. we assume different layers contain different momentum with different hyperparameters. We have the following theoretical result (see Appendix A.9 for the proof).

Theorem 2.1. *Suppose $\ell(\theta)$ in (2.1) is lower bounded by ℓ^*, γ-smooth (i.e. $\nabla\ell(\theta)$ is Lipschitz continuous with constant γ), also the stochastic gradient is unbiased $\mathbb{E}_{\xi_t}\nabla_{\theta_l}\ell(\theta^t;\xi_t) = \nabla_{\theta_l}\ell(\theta^t)$ and with bounded variance:*

$$\mathbb{E}_{\xi_t}\|\nabla_{\theta_l}\ell(\theta^t;\xi_t) - \nabla_{\theta_l}\ell(\theta^t)\|^2 \le \sigma_l^2 \tag{2.7}$$

for all $l = 1, \ldots, L$ and $t = 0, \ldots, T-1$. With appropriate choice of hyperparameters $\eta_l \ge \eta$ (see (A.20) and (A.23)) and $\beta_l \le 1-\delta$ (δ is an absolute constant), we have the following convergence result for update (2.6):

$$\begin{aligned} \frac{1}{T}\sum_{t=1}^{T}\sum_{l=1}^{L}\mathbb{E}\|\nabla_l^t\|^2 &\le \frac{2L\gamma^{3/2}\mathbb{E}\Delta_1}{\delta^2\sqrt{T}} \\ &+ \sum_{l=1}^{L}\left(\frac{1-\beta_l}{1+\beta_l}\frac{L\sqrt{\gamma}}{4\sqrt{T}} + \frac{L\gamma^{3/2}}{2\sqrt{T}} + \frac{1-\beta_l}{\beta_l^3}\frac{\gamma^2}{4LT}\right)\frac{\sigma_l^2}{\delta^2} \end{aligned} \tag{2.8}$$

where $\Delta_1 = \ell(\theta^1) - \ell^*$.

[1]We take the common training batch size as 32 and the large batch size as 512.

Algorithm 1 `SCALE`: Stochastic Column-normalized Last-layer Momentum

```
Input: Initialized trainable parameters $\theta^0$, hyperparameters $\beta_t$ and $\eta_{t,l}$.
for $t = 0, 1, ..., T-1$ do
    Sample mini-batch data $\{\xi_{t,b}\}_{b=1,...,B}$;
    for Layers $l = 1, ..., L$ do
        Compute the stochastic gradient $g_l^t := \frac{1}{B}\sum_{b=1}^{B} \nabla_{\theta_l}\ell(\theta^t; \xi_{t,b})$;
        if $l = L$ (last layer) then
            $m_l^t = \beta_t m_l^{t-1} + (1-\beta_t)\nabla_{\theta_l}\ell(\theta^t; \xi_t)$;
        else
            $m_l^t = \nabla_{\theta_l}\ell(\theta^t; \xi_t)$ (record the gradient directly);
        end if
        $\theta_l^{t+1} = \theta_l^t - \eta_l \mathcal{C}(m_l^t)$ where $\mathcal{C}$ is the column-wise normalization;
    end for
end for
```

Remark 2.1. *Theorem 2.1 suggests that by taking different β_l for different layers $l = 1, ..., L$, the convergence could be improved. The second term of the right-hand side of (2.8) takes the form*

$$f(x) = \frac{1-x}{1+x} + c\frac{1-x}{x^3},\ c := \frac{\gamma}{L^2 T^{3/2}}$$

where x represents the layer-wise momentum hyperparameter β_l. It is straightforward to verify that $f(x)$ is decreasing in $(0, 1-\delta]$, therefore an optimal strategy is to pick $\beta_l = 1-\delta$. However from a memory-efficient point of view, only $\beta_l = 0$ saves the memory of momentums.

Now, if σ_l (variance of the gradient of the l-th layer) is significantly higher than other layers, then taking β_l higher than other layers will result in better convergence. On the other hand, if σ_l is close to) zero, taking $\beta_l = 0$ will provide memory-efficiency without harming the convergence too much. In particular, if the variances $\sigma_l \approx 0$ for $l = 1, ..., L-1$, one could take $\beta_l = 0$ for $l = 1, ..., L-1$ and only keep the momentum for the last layer without significantly damaging the convergence rate.

We also conduct experiments in Appendix A.6 and found that column-wise normalization could outperform singular-value normalization when combining with last-layer momentums (see Table 5).

3. Proposed Algorithm

We now present our new algorithm `SCALE` in Algorithm 1. The proposed algorithm is a simple combination of column-wise normalization and last-layer momentum. In Appendix A.2, we discuss in detail how our algorithm relates to existing works, as well as possible theoretical insights.

Methods	Sign	Col-wise	Row-wise	Singular-val	1st order EMA	2nd order EMA	Memory (7B)
SGD							13.48
Adafactor						✓[†]	13.48
Adam	✓				✓	✓	40.43
Muon				✓	✓		26.95
SWAN			✓	✓		⋆	14.52
`SCALE`		✓			Last Layer		13.74

[†]: Adafactor only records the 2nd order EMA of the column and row norms, effectively reducing the memory from $d_{\text{in}} \times d_{\text{out}}$ to $d_{\text{in}} + d_{\text{out}}$

[⋆]: SWAN applies Adam for the first and last layer, following the practice of GaLore (see Ma et al. (2024, Appendix J)).

Table 2. Summarization of related methods. "Sign", "Col-wise", "Row-wise" and "Singular-val" correspond to four normalizations in (2.5), respectively. "1st order EMA" and "2nd order EMA" stand for first and second order EMA. The last column records the memory (GB) of weights and optimizer states for LLaMA 7B training (see Appendix A.3 for the details of the estimation).

4. Experiments

In this section, we test the proposed Algorithm 1 for LLM pretraining. We test on pretraining LLaMA (60M, 130M, 350M, 1B) models on the C4 (Colossal Clean Crawled Corpus) dataset (Raffel et al., 2020). The detailed experiment settings are provided in Appendix A.7.

Results. We report the results of the evaluation perplexity in Table 3. We can see that the proposed algorithm outperforms existing memory efficient optimizers, also (nearly) matches the performance of the state of the art (Stable-SPAM and Muon), especially for larger models, with only 35–65% of the memory. We also contrast the performance with the memory consumed in Figure 1, where we can see that the proposed method is indeed at the Pareto frontier for optimal memory use while maintaining the (near) state-of-the-art performance. This makes it a strong candidate for large-scale pretraining under memory constraints.

Model Size	**60M**	**130M**	**350M**	**1B**
Tokens	1.4B	2.6B	7.8B	13.1B
Adam	30.05 (0.35G)	23.13 (0.81G)	18.77 (2.21G)	16.52 (8.04G)
Stable-SPAM	**28.77** (0.35G)	**22.20** (0.81G)	16.80 (2.21G)	**13.97** (8.04G)
Muon	**28.86** (0.23G)	**22.20** (0.54G)	16.70 (1.47G)	14.18 (5.36G)
GaLore	34.58 (0.28G)	25.31 (0.61G)	19.37 (1.59G)	15.57 (4.76G)
Fira	30.34 (0.28G)	22.96 (0.61G)	16.82 (1.59G)	15.10 (4.76G)
SWAN (from (Ma et al., 2024))	30.00 (0.25G)	22.83 (0.46G)	17.14 (1.00G)	14.42 (3.20G)
APOLLO	30.94 (0.28G)	22.93 (0.61G)	16.75 (1.59G)	14.20 (4.76G)
APOLLO-Mini	31.85 (0.25G)	23.63 (0.46G)	17.11 (1.00G)	14.13 (3.20G)
`SCALE` (ours)	30.81 (0.15G)	22.57 (0.32G)	**16.32** (0.80G)	14.25 (2.81G)

Table 3. Experiment results for pretraining on LLaMA models on C4 dataset.

5. Conclusion and Limitations

In this paper, we design a memory efficient optimizer through a minimalist approach. The proposed algorithm utilizes the building blocks that lead to the success of Adam but further refine them to make it more memory efficient. We motivate each of our construction step by theoretical or experimental tests. The resulting algorithm, `SCALE`, achieves superior pretraining performance while only requires 35-45% of Adam memory. Limitations of current work include that it does not involve other models or pretraining datasets. Future works include scaling up the optimizer to even larger models and testing it on more datasets.

References

Bernstein, J. and Newhouse, L. Old optimizer, new norm: An anthology. *arXiv preprint arXiv:2409.20325*, 2024.

Chen, X., Feng, K., Li, C., Lai, X., Yue, X., Yuan, Y., and Wang, G. Fira: Can we achieve full-rank training of llms under low-rank constraint? *arXiv preprint arXiv:2410.01623*, 2024.

Han, A., Li, J., Huang, W., Hong, M., Takeda, A., Jawanpuria, P., and Mishra, B. SLTrain: a sparse plus low rank approach for parameter and memory efficient pretraining. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL `https://openreview.net/forum?id=MXze4H7opg`.

Hinton, G., Srivastava, N., and Swersky, K. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8):2, 2012.

Huang, T., Hu, H., Zhang, Z., Jin, G., Li, X., Shen, L., Chen, T., Liu, L., Wen, Q., Wang, Z., et al. Stable-spam: How to train in 4-bit more stably than 16-bit adam. *arXiv preprint arXiv:2502.17055*, 2025a.

Huang, T., Zhu, Z., Jin, G., Liu, L., Wang, Z., and Liu, S. SPAM: Spike-aware adam with momentum reset for stable LLM training. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL `https://openreview.net/forum?id=L9eBxTCpQG`.

Jordan, K., Jin, Y., Boza, V., You, J., Cesista, F., Newhouse, L., and Bernstein, J. Muon: An optimizer for hidden layers in neural networks, 2024. URL `https://kellerjordan.github.io/posts/muon/`.

Kalra, D. S., Kirchenbauer, J., Barkeshli, M., and Goldstein, T. When can you get away with low memory adam? *arXiv preprint arXiv:2503.01843*, 2025.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations (ICLR)*, 2015.

Liao, X., Li, S., Xu, Y., Li, Z., Liu, Y., and He, Y. Galore +: Boosting low-rank adaptation for llms with cross-head projection. *arXiv preprint arXiv:2412.19820*, 2024.

Liu, H., Li, Z., Hall, D. L. W., Liang, P., and Ma, T. Sophia: A scalable stochastic second-order optimizer for language model pre-training. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=3xHDeA8Noi`.

Liu, J., Su, J., Yao, X., Jiang, Z., Lai, G., Du, Y., Qin, Y., Xu, W., Lu, E., Yan, J., et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025.

Liu, Y., Gao, Y., and Yin, W. An improved analysis of stochastic gradient descent with momentum. *Advances in Neural Information Processing Systems*, 33:18261–18271, 2020.

Luo, Q., Yu, H., and Li, X. Badam: A memory efficient full parameter training method for large language models. *arXiv preprint arXiv:2404.02827*, 2024.

Ma, C., Gong, W., Scetbon, M., and Meeds, E. SWAN: Preprocessing sgd enables adam-level performance on llm training with significant memory reduction. *arXiv preprint arXiv:2412.13148*, 2024.

Muhamed, A., Li, O., Woodruff, D., Diab, M., and Smith, V. GRASS: Compute efficient low-memory llm training with structured sparse gradients. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 14978–15003, 2024.

Pan, R., Liu, X., Diao, S., Pi, R., Zhang, J., Han, C., and Zhang, T. Lisa: layerwise importance sampling for memory-efficient large language model fine-tuning. *Advances in Neural Information Processing Systems*, 37: 57018–57049, 2024.

Pethick, T., Xie, W., Antonakopoulos, K., Zhu, Z., Silveti-Falls, A., and Cevher, V. Training deep learning models with norm-constrained lmos. *arXiv preprint arXiv:2502.07529*, 2025.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21 (140):1–67, 2020.

Ramesh, A. V., Ganapathiraman, V., Laradji, I. H., and Schmidt, M. Blockllm: Memory-efficient adaptation of llms by selecting and optimizing the right coordinate blocks. *arXiv preprint arXiv:2406.17296*, 2024.

Reddi, S. J., Kale, S., and Kumar, S. On the convergence of adam and beyond. In *International Conference on Learning Representations (ICLR)*, 2019.

Shazeer, N. and Stern, M. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pp. 4596–4604. PMLR, 2018.

Xu, M., Xiang, L., Cai, X., and Wen, H. No more adam: Learning rate scaling at initialization is all you need. *arXiv preprint arXiv:2412.11768*, 2024.

Zhang, J., Karimireddy, S. P., Veit, A., Kim, S., Reddi, S., Kumar, S., and Sra, S. Why are adaptive methods good for attention models? *Advances in Neural Information Processing Systems*, 33:15383–15393, 2020.

Zhang, Y., Chen, C., Shi, N., Sun, R., and Luo, Z.-Q. Adam can converge without any modification on update rules. *Advances in neural information processing systems*, 35: 28386–28399, 2022.

Zhang, Y., Chen, C., Li, Z., Ding, T., Wu, C., Ye, Y., Luo, Z.-Q., and Sun, R. Adam-mini: Use fewer learning rates to gain more. *arXiv preprint arXiv:2406.16793*, 2024.

Zhao, J., Zhang, Z., Chen, B., Wang, Z., Anandkumar, A., and Tian, Y. Galore: Memory-efficient LLM training by gradient low-rank projection. In *Forty-first International Conference on Machine Learning*, 2024.

Zhao, R., Morwani, D., Brandfonbrener, D., Vyas, N., and Kakade, S. M. Deconstructing what makes a good optimizer for autoregressive language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Zhu, H., Zhang, Z., Cong, W., Liu, X., Park, S., Chandra, V., Long, B., Pan, D. Z., Wang, Z., and Lee, J. Apollo: Sgd-like memory, adamw-level performance, 2024. URL https://arxiv.org/abs/2412.05270.

Zhu, H., Zhang, Z., Cong, W., Liu, X., Park, S., Chandra, V., Long, B., Pan, D. Z., Wang, Z., and Lee, J. APOLLO: SGD-like memory, adamw-level performance. In *Eighth Conference on Machine Learning and Systems*, 2025. URL https://openreview.net/forum?id=mJrPkdcZDj.

A. Appendix

A.1. Related Works

Memory-efficient variations of Adam. A recent line of works aims to improve memory efficiency of Adam, compressing the historical states stored, namely the first- and second-order statistics by the use of gradient projections. GaLore (Zhao et al., 2024), being one of the pioneering works, stores the states in a low-rank subspaces that capture most gradient information. Fira (Chen et al., 2024) achieves superior performance than GaLore by re-introducing full-rank information to the low-rank gradients. APOLLO (Zhu et al., 2024) constructs the update based on gradient scaling factors that are estimated from the ratio between the low-dimensional gradient and the low-dimensional Adam update; APOLLO-Mini is a memory efficient version of APOLLO by estimating in a rank-1 subspace. GRASS (Muhamed et al., 2024) improves GaLore by designing sparse projection matrices guided by the norm of the rows of the gradients. SlimAdam (Kalra et al., 2025) compresses second-order moments based on signal-to-noise analysis. Some other methods group parameters into blocks and apply block-wise updates (Luo et al., 2024; Ramesh et al., 2024; Pan et al., 2024), block-wise scaling (Zhang et al., 2024), channel-wise scaling (Zhu et al., 2025), to further reduce the memory costs.

Towards stateless optimizers. More recent works start to question the necessity of optimizer states as required by Adam. Adafactor (Shazeer & Stern, 2018) is perhaps the first work to record the momentum of the column and row norms of the gradient matrix, reducing the memory of optimizer states from $\mathcal{O}(d^2)$ to $\mathcal{O}(d)$ (assuming that the gradient matrix is d by d). SWAN (Ma et al., 2024) adopts gradient whitening and normalization, which matches the performance of Adam when it is applied only to the intermediate layers (and the first and last layers still use Adam, see Section 4 for details). (Zhao et al., 2025) demonstrates SGD with signed momentum is able to recover the performance of Adam. SGD-SaI (Xu et al., 2024) verifies that proper learning-rate scaling at initialization is sufficient to achieve good performance.

Novel optimizers with better performance than Adam. Another line of work focuses on proposing new optimizers to achieve improved convergence properties compared to Adam. SPAM and Stable-SPAM (Huang et al., 2025b;a) reset momentum and clip spike gradients, stabilizing mixed-precision LLM training and yield better performance than Adam. Sophia (Liu et al., 2024) accelerates training by leveraging an estimated diagonal Hessian. Muon (Jordan et al., 2024) proposes gradient orthogonalization to further stabilize and enhance LLM training. Other approaches (Bernstein & Newhouse, 2024; Pethick et al., 2025; Kalra et al., 2025) explore various (layer-wise) normalization schemes to design better adaptive optimizers.

A.2. Connection of proposed algorithm to existing works, also convergence

We now give a discussion on the connection of the proposed method to existing works. The proposed algorithm utilizes column-wise normalization, which is also discussed in (Pethick et al., 2025). In particular, (Pethick et al., 2025) uses momentum for all the layers, also singular-value normalization for most of layers except for the last layer, where they also propose an option of column-wise normalization (since (Pethick et al., 2025) denotes the weight matrices as $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, their row-wise normalization is the same operation as our column-wise normalization); Row-wise normalization is utilized in SWAN (Ma et al., 2024), and our method differs from SWAN in the following aspects: first, we use column-wise normalization and SWAN uses row-wise normalization; second, SWAN essentially uses both the row-wise and singular-value normalization while we only utilize column-wise normalization; third, SWAN applies AdamW for the embedding and LM head layers (the first and last layers) which significantly increases the memory overhead, whereas our approach only introduces first-order momentum for the last layer. We summarize the techniques used in different papers in Table 2.

Convergence of Algorithm 1. We remark here that the convergence provided in Pethick et al. (2025, Section 5) could be adopted to provide a convergence analysis for Algorithm 1, since Algorithm 1 differs from the general framework of Pethick et al. (2025, Algorithm 2) only by the fact that Algorithm 1 applies different momentum for different layers. However, we believe that such convergence analysis will not provide insights on the superior performance of the proposed algorithm, since existing analysis also does not provide explanations of the superior performance of Adam over SGD in terms of convergence rate (Reddi et al., 2019; Zhang et al., 2022).

A.3. Details of memory estimation for 1B and 7B models

Here we compute the memory estimate for both 1B and 7B models. We only compute the major parameters, including embedding layers, attention and MLP layers. We follow prior works (Zhao et al., 2024; Han et al., 2024) in estimating the memory using `bfloat16` format, where each floating point number occupies 2 bytes.

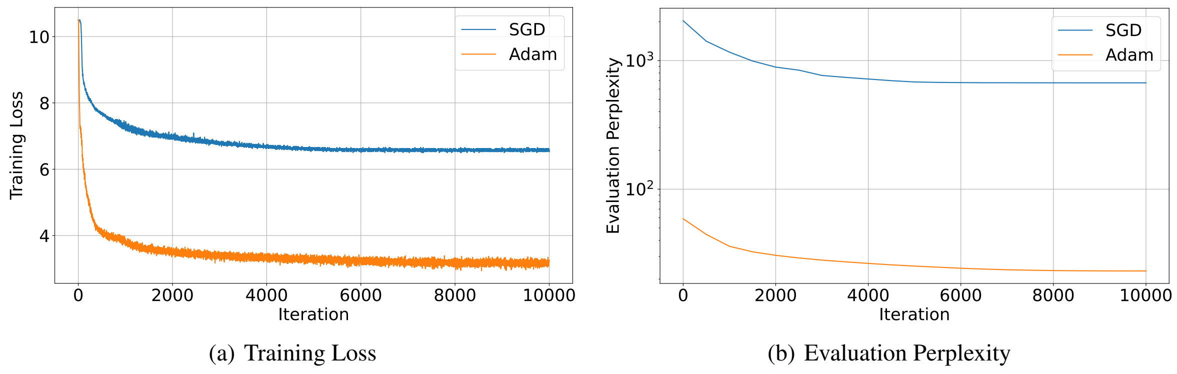
7B model: Pre-last layers include 6.607B parameters and last layer includes 0.131B parameters, which in total leads to 6.738B parameters.

- **SGD**: Only the parameter states are stored, which amount to 13.476G memory.
- **Adafactor**: Apart from the parameter states, Adafactor stores a row-wise and column-wise momentum, which is 0.005G memory. In total, Adafactor requires 13.481G memory.
- **Adam**: Apart from the parameter states, Adam/AdamW store first and second order momentum, which costs 26.952G. In total, Adam/AdamW requires 40.428G memory.
- **Muon**: Apart from the parameter states, Muon stores first-order momentum, which costs 13.476G. In total, Muon requires 26.952G memory.
- **SWAN**: Apart from the parameter states, SWAN additionally stores first-order and second-order momentum of the first and last layer, which costs 1.048G. In total, SWAN requires 14.524G.
- **`SCALE`** (Our method): Apart from parameter states, `SCALE` additionally stores first-order momentum of last-layer weight, which costs 0.262G. In total, `SCALE` requires 13.738G memory.

1B model: Pre-last layers include 1.273B parameters and last layer includes 0.066B parameters, which in total leads to 1.339B parameters.

- **SGD**: Only the parameter states are stored, which amount to 2.678G memory.
- **Adafactor**: Apart from the parameter states, Adafactor stores a row-wise and column-wise momentum, which is 0.002G memory. In total, Adafactor requires 2.68G memory.
- **Adam**: Apart from the parameter states, Adam/AdamW store first and second order momentum, which costs 5.356G. In total, Adam/AdamW requires 8.034G memory.
- **Muon**: Apart from the parameter states, Muon stores first-order momentum, which costs 2.678G. In total, Muon requires 5.356G memory.
- **SWAN**: Apart from the parameter states, SWAN additionally stores first-order and second-order momentum of the first and last layer, which costs 0.524G. In total, SWAN requires 3.202G.
- **`SCALE`** (Our method): Apart from parameter states, `SCALE` additionally stores first-order momentum of last-layer weight, which costs 0.131G. In total, `SCALE` requires 2.809G memory.

A.4. SGD fails to deliver reasonable performance



(a) Training Loss (b) Evaluation Perplexity

Figure 3. Comparison of SGD and Adam training loss and evaluation perplexity on LLaMA 130M model. With the best tuned learning rates, SGD fails to converge to any reasonable level of perplexity.

A.5. Computation cost of different normalizations

In terms of the computational cost, the normalization techniques discussed in (2.5) can be quite different. In particular, singular-value normalization requires computing full SVD, which is the most time consuming comparing to the rest. Even though efficient approximation methods of SVD has been studied in (Jordan et al., 2024; Ma et al., 2024) (e.g., the Newton-Schulz (NS) procedure), they are still much more time consuming as compare with the other three normalization techniques, see Table 4 for a test on the time required for different normalizations.

dimension d	1024	2048	4096
singular-value	79.77	354.27	1958.66
singular-value (NS)	6.03	7.00	14.41
column-wise	0.10	0.12	0.17
row-wise	0.09	0.11	0.13
sign	0.03	0.03	0.03

Table 4. Time (ms) consumed by each of the normalization methods on a torch matrix tensor with dimension $d_{\text{in}} = d_{\text{out}} = d$, testing on a single NVIDIA A40 GPU. Here the singular-value normalization are computed both exactly (first row, using `torch.linalg.svd` directly) and inexactly using Newton-Schulz (NS) iteration (second row, see (Jordan et al., 2024) for details)[3].

A.6. Col-wise normalization is better when combining with last-layer momentum

It is observed in Table 1 that singular-value normalization obtains better performance than the column-wise normalization for pretraining. We conduct another simple experiment to check the performance of the two normalizations with last-layer-momentum. The results are summarized in Table 5, from which we can see that the performance of two normalizations + last-layer-momentum is similar, and we choose column-wise due to the computational time concern in view of Table 4.

Model Size	**60M**	**130M**	**350M**
Tokens	1.4B	2.6B	7.8B
Muon	**28.86**	**22.20**	16.70
Singular-val + last-layer-momentum	31.20	22.33	16.67
Column-wise + last-layer-momentum (ours)	30.81	22.57	**16.32**

Table 5. Evaluation perplexity of two normalizations (singular-value and column-wise) when combined with last-layer-momentum.

A.7. Details of the experiments

We performed wandb sweeps for all methods we tested up to models of size 350M, searching learning rates within $\{0.00005, 0.0001, 0.0003, 0.0005, 0.001, 0.003, 0.005, 0.01\}$. For the 1B model, due to resource constraints we manually tune the learning rates using as starting point the optmial learning rate from the 350M sweep. For SCALE, the reported results are using learning rates 1e-3 for models sizes 60M, 130M and 350M and 2e-4 for 1B. In addition, our reported result for the 1B model uses the same learning rate scaling technique used by Muon (Liu et al., 2025), as we find to result to a minor gain; without it, the achieved perplexity for the 1B model is 14.34. Also, we set the last layer's momentum parameter $\beta = 0.9$, being a common choice for first order momentum. In addition, for all vector parameters we employ the Adam optimizer, following (Jordan et al., 2024; Liao et al., 2024). This does not influence the memory usage because the vector parameters are orders of magnitude smaller in size compared to the matrix parameters.

Baselines. We compare the proposed Algorithm 1 with Adam, Stable-SPAM, Muon, GaLore, Fira and SWAN (see Section A.1 for a more detailed introduction of these works). Stable-SPAM and Muon provide the state-of-the-art pretraining perplexities over Adam. GaLore and Fira are two memory-efficient optimizers that project the gradients to low rank. SWAN is an optimizer that uses column normalized SGD except for the first and last layers, and it could achieve state-of-the-art[4].

It is worth noticing that GaLore, Fira and SWAN run Adam for the first and last layers for stable training. For the 60M model the first and last layers contain over 50% of the total network parameters, and around 40% for 130M model. For the 350M this goes down to less than 20% and for the 1B to about 10%. Therefore, for smaller models these methods have

[3]The difference of column- and row-wise normalization may originate from the way PyTorch stores tensors with strides, see this link.
[4]We copy SWAN's result directly from (Ma et al., 2024) since we cannot replicate the result exactly due to code unavailability.

limited memory savings as compared to Adam, which trains a significant percentage of the network parameters.

For all LLaMA experiments, we follow (Zhao et al., 2024) and set the sequence length to 256 and the batch size to 512, train using BF16 format and report evaluation perplexity as our metric. We also use a cosine learning rate with linear warm-up for the first 10% of the iterations. For low-rank optimizers (GaLore and Fira) we follow their suggested hyperparameters (including the rank) but tune the learning rates. For muon we follow the implementation from (Liu et al., 2025). For all the implemented methods, we conduct a hyperparameter search via wandb sweeps (see Appendix A.7 for details).

A.8. Learning Rate Sensitivity analysis

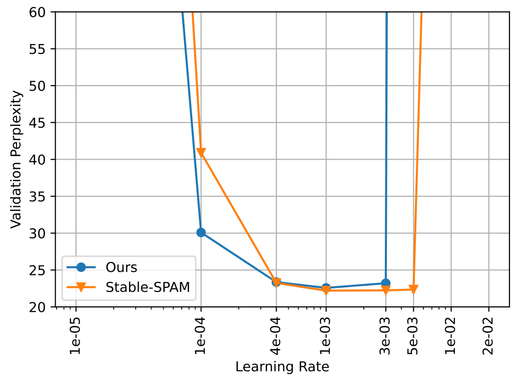


Figure 4. Learning rate sensitivity analysis, comparing Stable-SPAM (a stabilized version of Adam) and our method. Results from the 130M LLaMA model.

In Figure 4 we test the performance of our algorithm, `SCALE`, with different learning rates and compare it with that of Stable-SPAM (Huang et al., 2025a). We observe both algorithms behave similarly withing a reasonable range of learning rates.

A.9. Proofs for Section 2.2

In this section, we conduct the proof for Theorem 2.1. The proof follows (Liu et al., 2020). First, we have the following lemmas, which are variations of Liu et al. (2020, Lemma 1 and 2).

Lemma A.1. *Suppose that the assumptions in Theorem 2.1 hold. For SGD-M (2.6), we have*

$$\mathbb{E}\left[\left\|m_l^t - (1-\beta_l)\sum_{i=1}^{t}\beta_l^{t-i}\nabla_{\theta_l}\ell(\theta^i)\right\|^2\right] \le \frac{1-\beta_l}{1+\beta_l}\left(1-\beta_l^{2t}\right)\sigma_l^2. \tag{A.1}$$

Proof. It is straightforward to see $m_l^t = (1-\beta_l)\sum_{i=1}^{t}\beta_l^{t-i}g_l^i$, where $g_l^i := \nabla_{\theta_l} f(\theta^i;\xi_i)$.

We have

$$\mathbb{E}\left[\left\|m_l^t - (1-\beta_l)\sum_{i=1}^{t}\beta_l^{t-i}\nabla_{\theta_l}\ell(\theta^i)\right\|^2\right] = (1-\beta_l)^2\mathbb{E}\left\|\sum_{i=1}^{t}\beta_l^{t-i}(g_l^i - \nabla_{\theta_l}\ell(\theta^i))\right\|^2.$$

Therefore

$$\begin{aligned}
&\mathbb{E}\left[\left\|m_l^t - (1-\beta_l)\sum_{i=1}^t \beta_l^{t-i}\nabla_{\theta_l}\ell(\theta^i)\right\|^2\right]\\
=&(1-\beta_l)^2\mathbb{E}_{\xi_1}\mathbb{E}_{\xi_2}\cdots\mathbb{E}_{\xi_t}\left\|\sum_{i=1}^t \beta_l^{t-i}(g_l^i - \nabla_{\theta_l}\ell(\theta^i))\right\|^2\\
=&(1-\beta_l)^2\mathbb{E}_{\xi_1}\mathbb{E}_{\xi_2}\cdots\mathbb{E}_{\xi_t}\left[\sum_{i=1}^t\sum_{j=1}^t \langle \beta_l^{t-i}(g_l^i - \nabla_{\theta_l}\ell(\theta^i)), \beta_l^{t-j}(g_l^j - \nabla_{\theta_l}\ell(\theta^j))\rangle\right].
\end{aligned}$$

Due to unbiasedness of stochastic gradients, the cross terms cancel, therefore we have

$$\begin{aligned}
&\mathbb{E}\left[\left\|m_l^t - (1-\beta_l)\sum_{i=1}^t \beta_l^{t-i}\nabla_{\theta_l}\ell(\theta^i)\right\|^2\right]\\
=&(1-\beta_l)^2\sum_{i=1}^t \beta_l^{2(t-i)}\mathbb{E}_{\xi_1}\mathbb{E}_{\xi_2}\cdots\mathbb{E}_{\xi_t}\left\|g_l^i - \nabla_{\theta_l}\ell(\theta^i)\right\|^2 \le \frac{1-\beta_l}{1+\beta_l}(1-\beta_l^{2t})\sigma_l^2,
\end{aligned}$$

for all layers $l = 1, ..., L$. □

Lemma A.2. *Suppose that the assumptions in Theorem 2.1 hold. For SGD-M (2.6), we have*

$$\mathbb{E}\left[\left\|\frac{1-\beta_l}{1-\beta_l^t}\sum_{i=1}^t \beta_l^{t-i}\nabla_{\theta_l}\ell(\theta^i) - \nabla_{\theta_l}\ell(\theta^t)\right\|^2\right] \le \sum_{i=1}^{t-1} a_{l,t,i}\mathbb{E}\left[\left\|\theta^{i+1}-\theta^i\right\|^2\right], \tag{A.2}$$

for all layers $l = 1, ..., L$, where

$$a_{l,t,i} = \frac{\gamma^2\beta_l^{t-i}}{1-\beta_l^t}\left(t-i+\frac{\beta_l}{1-\beta_l}\right). \tag{A.3}$$

Proof. Since

$$
\begin{aligned}
&\mathbb{E}\left[\left\|\frac{1-\beta_l}{1-\beta_l^t}\sum_{i=1}^t \beta_l^{t-i}\nabla_{\theta_l}\ell(\theta^i)-\nabla_{\theta_l}\ell(\theta^t)\right\|^2\right] \\
&=\left(\frac{1-\beta_l}{1-\beta_l^t}\right)^2\sum_{i,j=1}^t \mathbb{E}\left[\left\langle \beta_l^{t-i}\left(\nabla_{\theta_l}\ell(\theta^t)-\nabla_{\theta_l}\ell(\theta^i)\right), \beta_l^{t-j}\left(\nabla_{\theta_l}\ell(\theta^t)-\nabla_{\theta_l}\ell(\theta^j)\right)\right\rangle\right] \\
&\le \left(\frac{1-\beta_l}{1-\beta_l^t}\right)^2\sum_{i,j=1}^t \beta_l^{2t-i-j}\left(\frac{1}{2}\mathbb{E}\left[\left\|\nabla_{\theta_l}\ell(\theta^t)-\nabla_{\theta_l}\ell(\theta^i)\right\|^2\right]+\frac{1}{2}\mathbb{E}\left[\left\|\nabla_{\theta_l}\ell(\theta^t)-\nabla_{\theta_l}\ell(\theta^j)\right\|^2\right]\right) \\
&=\left(\frac{1-\beta_l}{1-\beta_l^t}\right)^2\sum_{i=1}^t\left(\sum_{j=1}^t \beta_l^{2t-i-j}\right)\frac{1}{2}\mathbb{E}\left[\left\|\nabla_{\theta_l}\ell(\theta^t)-\nabla_{\theta_l}\ell(\theta^j)\right\|^2\right] \\
&\quad+\left(\frac{1-\beta_l}{1-\beta_l^t}\right)^2\sum_{j=1}^t\left(\sum_{i=1}^t \beta_l^{2t-i-j}\right)\frac{1}{2}\mathbb{E}\left[\left\|\nabla_{\theta_l}\ell(\theta^t)-\nabla_{\theta_l}\ell(\theta^i)\right\|^2\right] \\
&=\left(\frac{1-\beta_l}{1-\beta_l^t}\right)^2\sum_{i=1}^t \frac{\beta_l^{t-i}\left(1-\beta_l^t\right)}{1-\beta_l}\mathbb{E}\left[\left\|\nabla_{\theta_l}\ell(\theta^t)-\nabla_{\theta_l}\ell(\theta^i)\right\|^2\right] \\
&=\frac{1-\beta_l}{1-\beta_l^t}\sum_{i=1}^t \beta_l^{t-i}\mathbb{E}\left[\left\|\nabla_{\theta_l}\ell(\theta^t)-\nabla_{\theta_l}\ell(\theta^i)\right\|^2\right] \\
&\le \frac{1-\beta_l}{1-\beta_l^t}\sum_{i=1}^t \beta_l^{t-i}(t-i)\sum_{j=i}^t \mathbb{E}\left[\left\|\nabla_{\theta_l}\ell(\theta^{j+1})-\nabla_{\theta_l}\ell(\theta^j)\right\|^2\right]
\end{aligned}
$$

where we use Cauchy-Schwarz inequality for the first inequality and the last is by AM-GM inequality. Now applying the Lipschitz smoothness assumption, we have

$$
\begin{aligned}
&\mathbb{E}\left[\left\|\frac{1-\beta_l}{1-\beta_l^t}\sum_{i=1}^t \beta_l^{t-i}\nabla_{\theta_l}\ell(\theta^i)-\nabla_{\theta_l}\ell(\theta^t)\right\|^2\right] \\
&\le \frac{1-\beta_l}{1-\beta_l^t}\sum_{i=1}^t \beta_l^{t-i}(t-i)\sum_{j=i}^t \gamma^2\mathbb{E}\left[\left\|\theta^{j+1}-\theta^j\right\|^2\right] \\
&\le \frac{1-\beta_l}{1-\beta_l^t}\sum_{j=1}^{t-1}\left(\sum_{i=1}^j \beta_l^{t-i}(t-i)\right)\gamma^2\mathbb{E}\left[\left\|\theta^{j+1}-\theta^j\right\|^2\right].
\end{aligned}
$$

Now define

$$
a'_{l,t,i}=\frac{1-\beta_l}{1-\beta_l^t}\gamma^2\sum_{i=1}^j \beta_l^{t-i}(t-i)=\frac{\gamma^2\beta_l^t}{1-\beta_l^t}\left(-(t-1)-\frac{1}{1-\beta_l}\right)+\frac{\gamma^2\beta_l^{t-j}}{1-\beta_l^t}\left(t-j+\frac{\beta_l}{1-\beta_l}\right)\le a_{l,t,i},
$$

then we get (A.2). □

Lemma A.3. *For SGD-M (2.6), define the auxiliary sequence by*

$$
z_l^t:=\begin{cases}\theta_l^t & t=1\\ \frac{1}{1-\beta_l}\theta_l^t-\frac{\beta_l}{1-\beta_l}\theta_l^{t-1} & t\ge 2\end{cases} \tag{A.4}
$$

and denote $z^t=[z_1^t,...,z_L^t]$ the entire auxiliary variable at iteration t. Then we have

$$
z_l^{t+1}-z_l^t=-\eta_l g_l^t
$$

and

$$z_l^t - \theta_l^t = -\frac{\beta_l}{1-\beta_l}\eta_l m_l^{t-1}.$$

Proof. For $t = 1$, we have (since $m^0 = 0$)

$$z_l^2 - z_1^1 = \frac{1}{1-\beta_l}\theta_l^2 - \frac{\beta_l}{1-\beta_l}\theta_l^1 - \theta_l^1 = \frac{1}{1-\beta_l}(\theta_l^2 - \theta_l^1) = -\eta_l g_l^1.$$

For $t \geq 2$, we have

$$\begin{aligned} z_l^{t+1} - z_1^t =& \frac{1}{1-\beta_l}(\theta_l^{t+1} - \theta_l^t) - \frac{\beta_l}{1-\beta_l}(\theta_l^t - \theta_l^{t-1}) \\ =& \frac{1}{1-\beta_l}(-\eta_l m_l^t) - \frac{\beta_l}{1-\beta_l}(-\eta_l m_l^{t-1}) \\ =& \frac{1}{1-\beta_l}(-\eta_l m_l^t + -\eta_l\beta_l m_l^{t-1}) \\ =& -\eta_l g_l^t. \end{aligned}$$

For $z_l^t - \theta_l^t$, it can be computed similarly. □

We need the following proposition to show the final convergence of Theorem 2.1.

Proposition A.1. *Suppose that the assumptions in Theorem 2.1 hold. For SGD-M (2.6), we have*

$$\begin{aligned} &\mathbb{E}\left[\ell\left(z^{t+1}\right)\right] \\ \leq& \mathbb{E}\left[\ell\left(z^t\right)\right] + \rho_0\left(\sum_{l=1}^L \eta_l\right)\sum_{l=1}^L\left(\eta_l^2(\frac{\beta_l}{1-\beta_l})^2\gamma^2\frac{1-\beta_l}{1+\beta_l}\right)\sigma_l^2 + \sum_{l=1}^L\frac{\gamma\eta_l^2}{2}\sigma_l^2 \\ &+ \sum_{l=1}^L(-\eta_l + \frac{\eta_l}{2\rho_0} + \frac{\gamma\eta_l^2}{2})\mathbb{E}\|\nabla_l\ell(\theta^t)\|^2 + 2\rho_0\left(\sum_{l=1}^L\eta_l\right)\sum_{l=1}^L\left(\eta_l^2(\frac{\beta_l}{1-\beta_l})^2\gamma^2(1-\beta_l^{t-1})^2\mathbb{E}\left[\left\|\nabla_l\ell(\theta^t)\right\|^2\right]\right) \\ &+ 2\rho_0\left(\sum_{l=1}^L\eta_l\right)\sum_{l=1}^L\left(\eta_l^2\gamma^2(\frac{1-\beta_l^t}{1-\beta_l})^2\mathbb{E}\left[\left\|\frac{1-\beta_l}{1-\beta_l^t}\sum_{i=1}^t\beta_l^{t-i}\nabla_l\ell(\theta^i) - \nabla_l\ell(\theta^t)\right\|^2\right]\right) \end{aligned} \tag{A.5}$$

where the auxiliary sequence z_l^t is defined by (A.4).

Proof. By Lemma A.3 we have that

$$z_l^{t+1} - z_l^t = -\eta_l g_l^t$$

and

$$z_l^t - \theta_l^t = -\frac{\beta_l}{1-\beta_l}\eta_l m_l^{t-1},$$

for all $l = 1, 2, ..., L$.

Now using the Lipschitz smooth of ℓ, we get

$$\begin{aligned} \mathbb{E}_{\xi_t}\left[\ell\left(z^{t+1}\right)\right] &\leq \ell\left(z^t\right) + \mathbb{E}_{\xi_t}\left[\left\langle\nabla\ell\left(z^t\right), z^{t+1} - z^t\right\rangle\right] + \frac{L}{2}\mathbb{E}_{\xi_t}\left[\left\|z^{t+1} - z^t\right\|^2\right] \\ &= \ell\left(z^t\right) + \sum_{l=1}^L\mathbb{E}_{\xi_t}\left[\left\langle\nabla_{\theta_l}\ell\left(z^t\right), -\eta_l g_l^t\right\rangle\right] + \sum_{l=1}^L\frac{\gamma\eta_l^2}{2}\mathbb{E}_{\xi_t}\left[\left\|g_l^t\right\|^2\right] \\ &= \ell\left(z^t\right) + \sum_{l=1}^L\left[\left\langle\nabla_{\theta_l}\ell\left(z^t\right), -\eta_l\nabla_l\ell(\theta^t)\right\rangle\right] + \sum_{l=1}^L\frac{\gamma\eta_l^2}{2}\mathbb{E}_{\xi_t}\left[\left\|g_l^t\right\|^2\right] \end{aligned}$$

where we use the unbiasedness of the gradient estimator in the last line. Now we bound the second term as follows:

$$
\begin{aligned}
&\mathbb{E}\left[\left\langle\nabla_{\theta_l}\ell\left(z^t\right), -\eta_l\nabla_l\ell(\theta^t)\right\rangle\right] \\
=&\mathbb{E}\left[\left\langle\nabla_{\theta_l}\ell\left(z^t\right) - \nabla_l\ell(\theta^t), -\eta_l\nabla_l\ell(\theta^t)\right\rangle\right] - \eta_l\mathbb{E}\|\nabla_l\ell(\theta^t)\|^2 \\
\leq&\eta_l\frac{\rho_0}{2}\gamma^2\mathbb{E}\|z^t-\theta^t\|^2 + (\frac{\eta_l}{2\rho_0} - \eta_l)\mathbb{E}\|\nabla_l\ell(\theta^t)\|^2 \\
=&\eta_l\frac{\rho_0}{2}\sum_l\left(\eta_l^2\mathbb{E}\left[\left(\frac{\beta_l}{1-\beta_l}\right)^2\gamma^2\|m_l^{t-1}\|^2\right]\right) + (\frac{\eta_l}{2\rho_0} - \eta_l)\mathbb{E}\|\nabla_l\ell(\theta^t)\|^2
\end{aligned}
$$

where we use Cauchy-Schwarz inequality and ρ_0 is a positive constant to be determined.

Therefore, we get

$$
\begin{aligned}
\mathbb{E}_{\xi_t}\left[\ell\left(z^{t+1}\right)\right] \leq&\ell\left(z^t\right) + \frac{\rho_0}{2}\left(\sum_{l=1}^L\eta_l\right)\sum_{l=1}^L\left(\eta_l^2(\frac{\beta_l}{1-\beta_l})^2\gamma^2\mathbb{E}\|m_l^{t-1}\|^2\right) \\
&+\sum_{l=1}^L(\frac{\eta_l}{2\rho_0}-\eta_l)\mathbb{E}\|\nabla_l\ell(\theta^t)\|^2 + \sum_{l=1}^L\frac{\gamma\eta_l^2}{2}\mathbb{E}_{\xi_t}\left[\left\|g_l^t\right\|^2\right]
\end{aligned}
\tag{A.6}
$$

Now from Lemma A.1 we have

$$
\begin{aligned}
\mathbb{E}\left[\left\|m_l^{t-1}\right\|^2\right] &\leq 2\mathbb{E}\left[\left\|m_l^{t-1} - (1-\beta_l)\sum_{i=1}^{t-1}\beta_l^{t-1-i}\nabla_l\ell(\theta^i)\right\|^2\right] + 2\mathbb{E}\left[\left\|(1-\beta_l)\sum_{i=1}^{t-1}\beta_l^{t-1-i}\nabla_l\ell(\theta^i)\right\|^2\right] \\
&\leq 2\frac{1-\beta_l}{1+\beta_l}\sigma_l^2 + 2\mathbb{E}\left[\left\|(1-\beta_l)\sum_{i=1}^{t-1}\beta_l^{t-1-i}\nabla_l\ell(\theta^i)\right\|^2\right]
\end{aligned}
$$

and

$$
\begin{gathered}
\mathbb{E}\left[\left\|\frac{1-\beta_l}{1-\beta_l^{t-1}}\sum_{i=1}^{t-1}\beta_l^{t-1-i}\nabla_l\ell(\theta^i)\right\|^2\right] \leq 2\mathbb{E}\left[\left\|\nabla_l\ell(\theta^t)\right\|^2\right] + 2\mathbb{E}\left[\left\|\frac{1-\beta_l}{1-\beta_l^{t-1}}\sum_{i=1}^{t-1}\beta_l^{t-1-i}\nabla_l\ell(\theta^i) - \nabla_l\ell(\theta^t)\right\|^2\right], \\
\mathbb{E}\left[\left\|g_l^t\right\|^2\right] \leq \sigma_l^2 + \mathbb{E}\left[\left\|\nabla_l\ell(\theta^t)\right\|^2\right].
\end{gathered}
$$

plug these into (A.6) we get:

$$\begin{aligned}
&\mathbb{E}_{\xi_t}\left[\ell\left(z^{t+1}\right)\right] \\
\leq &\ell\left(z^t\right)+\rho_0\left(\sum_{l=1}^L \eta_l\right)\sum_{l=1}^L\left(\eta_l^2(\frac{\beta_l}{1-\beta_l})^2\gamma^2\frac{1-\beta_l}{1+\beta_l}\sigma_l^2\right) \\
&+\rho_0\left(\sum_{l=1}^L \eta_l\right)\sum_{l=1}^L\left(\eta_l^2(\frac{\beta_l}{1-\beta_l})^2\gamma^2\mathbb{E}\left[\left\|(1-\beta_l)\sum_{i=1}^{t-1}\beta_l^{t-1-i}\nabla_l\ell(\theta^i)\right\|^2\right]\right) \\
&+\sum_{l=1}^L(\frac{\eta_l}{2\rho_0}-\eta_l)\mathbb{E}\|\nabla_l\ell(\theta^t)\|^2+\sum_{l=1}^L\frac{\gamma\eta_l^2}{2}\left(\sigma_l^2+\mathbb{E}\left[\left\|\nabla_l\ell(\theta^t)\right\|^2\right]\right) \\
=&\ell\left(z^t\right)+\rho_0\left(\sum_{l=1}^L \eta_l\right)\sum_{l=1}^L\left(\eta_l^2(\frac{\beta_l}{1-\beta_l})^2\gamma^2\frac{1-\beta_l}{1+\beta_l}\right)\sigma_l^2+\sum_{l=1}^L\frac{\gamma\eta_l^2}{2}\sigma_l^2 \\
&+\sum_{l=1}^L(\frac{\eta_l}{2\rho_0}-\eta_l+\frac{\gamma\eta_l^2}{2})\mathbb{E}\|\nabla_l\ell(\theta^t)\|^2 \\
&+\rho_0\left(\sum_{l=1}^L \eta_l\right)\sum_{l=1}^L\left(\eta_l^2(\frac{\beta_l}{1-\beta_l})^2\gamma^2(1-\beta_l^{t-1})^2\mathbb{E}\left[\left\|\frac{1-\beta_l}{1-\beta_l^{t-1}}\sum_{i=1}^{t-1}\beta_l^{t-1-i}\nabla_l\ell(\theta^i)\right\|^2\right]\right) \\
\leq&\ell\left(z^t\right)+\rho_0\left(\sum_{l=1}^L \eta_l\right)\sum_{l=1}^L\left(\eta_l^2(\frac{\beta_l}{1-\beta_l})^2\gamma^2\frac{1-\beta_l}{1+\beta_l}\right)\sigma_l^2+\sum_{l=1}^L\frac{\gamma\eta_l^2}{2}\sigma_l^2 \\
&+\sum_{l=1}^L(-\eta_l+\frac{\eta_l}{2\rho_0}+\frac{\gamma\eta_l^2}{2})\mathbb{E}\|\nabla_l\ell(\theta^t)\|^2+2\rho_0\left(\sum_{l=1}^L \eta_l\right)\sum_{l=1}^L\left(\eta_l^2(\frac{\beta_l}{1-\beta_l})^2\gamma^2(1-\beta_l^{t-1})^2\mathbb{E}\left[\left\|\nabla_l\ell(\theta^t)\right\|^2\right]\right) \\
&+2\rho_0\left(\sum_{l=1}^L \eta_l\right)\sum_{l=1}^L\left(\eta_l^2(\frac{\beta_l}{1-\beta_l})^2\gamma^2(1-\beta_l^{t-1})^2\mathbb{E}\left[\left\|\frac{1-\beta_l}{1-\beta_l^{t-1}}\sum_{i=1}^{t-1}\beta_l^{t-1-i}\nabla_l\ell(\theta^i)-\nabla_l\ell(\theta^t)\right\|^2\right]\right)
\end{aligned}$$

Now the last term above can be replaced since

$$\begin{aligned}
\mathbb{E}\left[\left\|\frac{1-\beta_l}{1-\beta_l^t}\sum_{i=1}^t\beta_l^{t-i}\nabla_l\ell(\theta^i)-\nabla_l\ell(\theta^t)\right\|^2\right]=&\mathbb{E}\left[\left\|\frac{\beta_l(1-\beta_l)}{1-\beta_l^t}\sum_{i=1}^{t-1}\beta_l^{t-1-i}\nabla_l\ell(\theta^i)-\frac{1-\beta_l^{t-1}}{1-\beta_l^t}\beta_l\nabla_l\ell(\theta^t)\right\|^2\right] \\
=&\beta_l^2(\frac{1-\beta_l^{t-1}}{1-\beta_l^t})^2\mathbb{E}\left[\left\|\frac{1-\beta_l}{1-\beta_l^{t-1}}\sum_{i=1}^{t-1}\beta_l^{t-1-i}\nabla_l\ell(\theta^i)-\nabla_l\ell(\theta^t)\right\|^2\right]
\end{aligned}$$

□

Now we return to the proof of Theorem 2.1.

Proof. [Proof of Theorem 2.1] By Proposition A.1 and Lemma A.2, we have

$$\begin{aligned}
&\mathbb{E}\left[\ell\left(z^{t+1}\right)\right] \\
\leq&\mathbb{E}\left[\ell\left(z^{t}\right)\right]+\rho_0\left(\sum_{l=1}^{L}\eta_l\right)\sum_{l=1}^{L}\left(\eta_l^2(\frac{\beta_l}{1-\beta_l})^2\gamma^2\frac{1-\beta_l}{1+\beta_l}\right)\sigma_l^2+\sum_{l=1}^{L}\frac{\gamma\eta_l^2}{2}\sigma_l^2 \\
&+\sum_{l=1}^{L}(-\eta_l+\frac{\eta_l}{2\rho_0}+\frac{\gamma\eta_l^2}{2})\mathbb{E}\|\nabla_l\ell(\theta^t)\|^2+2\rho_0\left(\sum_{l=1}^{L}\eta_l\right)\sum_{l=1}^{L}\left(\eta_l^2(\frac{\beta_l}{1-\beta_l})^2\gamma^2(1-\beta_l^{t-1})^2\mathbb{E}\left[\left\|\nabla_l\ell(\theta^t)\right\|^2\right]\right) \\
&+2\rho_0\left(\sum_{l=1}^{L}\eta_l\right)\sum_{l=1}^{L}\left(\eta_l^2\gamma^2(\frac{1-\beta_l^t}{1-\beta_l})^2\sum_{i=1}^{t-1}a_{l,t,i}\mathbb{E}\left[\left\|\theta^{i+1}-\theta^i\right\|^2\right]\right)
\end{aligned} \tag{A.7}$$

Now define the potential function:

$$\phi^t=\ell\left(z^t\right)-\ell^*+\sum_{i=1}^{t-1}\sum_{l=1}^{L}c_{l,i}\left\|\theta_l^{t+1-i}-\theta_l^{t-i}\right\|^2 \tag{A.8}$$

where $c_{l,i}$ are constants to be determined.

From (A.7) we get

$$\begin{aligned}
&\mathbb{E}\left[\phi^{t+1}\right]-\mathbb{E}\left[\phi^{t}\right] \\
\leq&\rho_0\left(\sum_{l=1}^{L}\eta_l\right)\sum_{l=1}^{L}\left(\eta_l^2(\frac{\beta_l}{1-\beta_l})^2\gamma^2\frac{1-\beta_l}{1+\beta_l}\right)\sigma_l^2+\sum_{l=1}^{L}\frac{\gamma\eta_l^2}{2}\sigma_l^2 \\
&+\sum_{l=1}^{L}(-\eta_l+\frac{\eta_l}{2\rho_0}+\frac{\gamma\eta_l^2}{2})\mathbb{E}\|\nabla_l\ell(\theta^t)\|^2+2\rho_0\left(\sum_{l=1}^{L}\eta_l\right)\sum_{l=1}^{L}\left(\eta_l^2(\frac{\beta_l}{1-\beta_l})^2\gamma^2(1-\beta_l^{t-1})^2\mathbb{E}\left[\left\|\nabla_l\ell(\theta^t)\right\|^2\right]\right) \\
&+2\rho_0\left(\sum_{l=1}^{L}\eta_l\right)\sum_{l=1}^{L}\left(\eta_l^2\gamma^2(\frac{1-\beta_l^t}{1-\beta_l})^2\sum_{i=1}^{t-1}a_{l,t,i}\mathbb{E}\left[\left\|\theta^{i+1}-\theta^i\right\|^2\right]\right) \\
&+\sum_{l=1}^{L}c_{l,1}\mathbb{E}\left\|\theta_l^{t+1}-\theta_l^t\right\|^2+\sum_{i=1}^{t-1}\sum_{l=1}^{L}(c_{l,i+1}-c_{l,i})\mathbb{E}\left[\left\|\theta_l^{t+1-i}-\theta_l^{t-i}\right\|^2\right]
\end{aligned} \tag{A.9}$$

For the term $\sum_{l=1}^{L}c_{l,1}\mathbb{E}\left\|\theta_l^{t+1}-\theta_l^t\right\|^2$, we can bound it by (denote $\nabla_l^t=\nabla_{\theta_l}\ell(\theta^t)$)

$$\begin{aligned}
&\sum_{l=1}^{L}c_{l,1}\mathbb{E}\left\|\theta_l^{t+1}-\theta_l^t\right\|^2=\sum_{l=1}^{L}c_{l,1}\eta_l^2\mathbb{E}\left[\left\|m_l^t\right\|^2\right] \\
\leq&2\sum_{l=1}^{L}c_{l,1}\mathbb{E}\left[\left\|m_l^t-(1-\beta_l)\sum_{i=1}^{t}c_{l,1}\beta_l^{t-i}\nabla_l^i\right\|^2\right]+2\sum_{l=1}^{L}\mathbb{E}\left[\left\|(1-\beta_l)\sum_{i=1}^{t}\beta_l^{t-i}\nabla_l^i\right\|^2\right] \\
\leq&2\sum_{l=1}^{L}c_{l,1}\frac{1-\beta_l}{1+\beta_l}\sigma_l^2+2\sum_{l=1}^{L}c_{l,1}\mathbb{E}\left[\left\|(1-\beta_l)\sum_{i=1}^{t}\beta_l^{t-i}\nabla_l^i\right\|^2\right] \\
\leq&\sum_{l=1}^{L}c_{l,1}\eta_l^2\left(2\frac{1-\beta_l}{1+\beta_l}\sigma_l^2+4\mathbb{E}\left[\left\|\nabla_l^t\right\|^2\right]\left(1-\beta_l^t\right)^2\right)+4\sum_{l=1}^{L}c_{l,1}(1-\beta_l^t)^2\eta_l^2\mathbb{E}\left[\left\|\frac{1-\beta_l}{1-\beta_l^t}\sum_{i=1}^{t}\beta^{t-i}\nabla_l^i-\nabla_l^t\right\|^2\right] \\
\leq&\sum_{l=1}^{L}c_{l,1}\eta_l^2\left(2\frac{1-\beta_l}{1+\beta_l}\sigma_l^2+4\mathbb{E}\left[\left\|\nabla_l^t\right\|^2\right]\right)+4\sum_{l=1}^{L}c_{l,1}(1-\beta_l^t)^2\eta_l^2\sum_{i=1}^{t-1}a_{l,t,i}\mathbb{E}\left[\left\|\theta^{i+1}-\theta^i\right\|^2\right]
\end{aligned} \tag{A.10}$$

where for the last three inequalities we use (A.1) and

$$\mathbb{E}\left[\left\|\frac{1-\beta_l}{1-\beta_l^t}\sum_{i=1}^{t}\beta_l^{t-i}\nabla_l^i\right\|^2\right] \leq 2\mathbb{E}\left[\left\|\nabla_l^t\right\|^2\right] + 2\mathbb{E}\left[\left\|\frac{1-\beta_l}{1-\beta_l^t}\sum_{i=1}^{t}\beta_l^{t-i}\nabla_l^i - \nabla_l^t\right\|^2\right],$$

and (A.2), respectively.

Now plugging this back to (A.9) we get:

$$\begin{aligned}
&\mathbb{E}\left[\phi^{t+1}\right] - \mathbb{E}\left[\phi^t\right] \\
\leq&\rho_0 s\sum_{l=1}^{L}\left(\eta_l^2(\frac{\beta_l}{1-\beta_l})^2\gamma^2\frac{1-\beta_l}{1+\beta_l}\right)\sigma_l^2 + \sum_{l=1}^{L}\frac{\gamma\eta_l^2}{2}\sigma_l^2 + \sum_{l=1}^{L}2c_{l,1}\eta_l^2\frac{1-\beta_l}{1+\beta_l}\sigma_l^2 \\
&+\sum_{l=1}^{L}(-\eta_l + \frac{\eta_l}{2\rho_0} + \frac{\gamma\eta_l^2}{2} + 4c_{l,1}\eta_l^2)\mathbb{E}\|\nabla_l^t\|^2 + 2\rho_0\left(\sum_{l=1}^{L}\eta_l\right)\sum_{l=1}^{L}\left(\eta_l^2(\frac{\beta_l}{1-\beta_l})^2\gamma^2(1-\beta_l^{t-1})^2\mathbb{E}\left[\left\|\nabla_l^t\right\|^2\right]\right) \\
&+4\sum_{i=1}^{t-1}\sum_{l=1}^{L}c_{l,1}(1-\beta_l^t)^2\eta_l^2 a_{l,t,i}\mathbb{E}\left[\left\|\theta^{i+1}-\theta^i\right\|^2\right] \\
&+2\rho_0 s\sum_{i=1}^{t-1}\sum_{l=1}^{L}\left(\eta_l^2\gamma^2(\frac{1-\beta_l^t}{1-\beta_l})^2 a_{l,t,i}\mathbb{E}\left[\left\|\theta^{i+1}-\theta^i\right\|^2\right]\right) \\
&+\sum_{i=1}^{t-1}\sum_{l=1}^{L}(c_{l,i+1}-c_{l,i})\mathbb{E}\left[\left\|\theta_l^{t+1-i}-\theta_l^{t-i}\right\|^2\right]
\end{aligned} \tag{A.11}$$

where we denote $s := \sum_{l=1}^{L}\eta_l$.

To make the last three lines of above non-positive, we could take

$$c_{l,i+1} \leq c_{l,i} - \left(4c_{l,1}(1-\beta_l^t)^2\eta_l^2 + 2\rho_0 s\eta_l^2\gamma^2(\frac{1-\beta_l^t}{1-\beta_l})^2\right)a_{l,t,t-i}$$

for all $l = 1, ..., L$.

We can take (since $1-\beta_l^t < 1$)

$$c_{l,i+1} = c_{l,i} - \left(4c_{l,1}\eta_l^2 + 2\rho_0 s\eta_l^2\gamma^2(\frac{1}{1-\beta_l})^2\right)\gamma^2\beta_l^i\left(i + \frac{\beta_l}{1-\beta_l}\right)$$

also we can take $c_{l,1}$ such that

$$\begin{aligned}
c_{l,1} =&\sum_{i=1}^{\infty}\left(4c_{l,1}\eta_l^2 + 2\rho_0 s\eta_l^2\gamma^2(\frac{1}{1-\beta_l})^2\right)\gamma^2\beta_l^i\left(i + \frac{\beta_l}{1-\beta_l}\right) \\
=&\left(4c_{l,1}\eta_l^2 + 2\rho_0 s\eta_l^2\gamma^2(\frac{1}{1-\beta_l})^2\right)\gamma^2\left(\sum_{i=1}^{\infty}i\beta_l^i + \frac{\beta_l}{1-\beta_l}\sum_{i=1}^{\infty}\beta_l^i\right) \\
=&\left(4c_{l,1}\eta_l^2 + 2\rho_0 s\eta_l^2\gamma^2(\frac{1}{1-\beta_l})^2\right)\frac{\gamma^2\beta_l(1+\beta_l)}{(1-\beta_l)^2}
\end{aligned}$$

i.e.

$$c_{l,1} = \frac{2\rho_0 s\eta_l^2\gamma^4\beta_l\frac{1+\beta_l}{(1-\beta_l)^4}}{1-4\eta_l^2\frac{\gamma^2\beta_l(1+\beta_l)}{(1-\beta_l)^2}}. \tag{A.12}$$

Note that we can give an upper bound for $c_{l,1}$ by requiring the denominator $\geq 1/2$, i.e.

$$\eta_l \leq \frac{1-\beta_l}{\gamma\sqrt{8\beta_l(1+\beta_l)}}, \tag{A.13}$$

consequently

$$c_{l,1} \leq 4\rho_0 s\eta_l^2\gamma^4\beta_l\frac{1+\beta_l}{(1-\beta_l)^4}. \tag{A.14}$$

Now we have (since the last three terms of (A.11) sum to negative)

$$\begin{aligned}&\mathbb{E}\left[\phi^{t+1}\right]-\mathbb{E}\left[\phi^{t}\right]\\ \leq&\rho_0 s\sum_{l=1}^{L}\left(\eta_l^2(\frac{\beta_l}{1-\beta_l})^2\gamma^2\frac{1-\beta_l}{1+\beta_l}\right)\sigma_l^2+\sum_{l=1}^{L}\frac{\gamma\eta_l^2}{2}\sigma_l^2+\sum_{l=1}^{L}2c_{l,1}\eta_l^2\frac{1-\beta_l}{1+\beta_l}\sigma_l^2\\ &+\sum_{l=1}^{L}\left(-\eta_l+\frac{\eta_l}{2\rho_0}+\frac{\gamma\eta_l^2}{2}+4c_{l,1}\eta_l^2+2\rho_0 s\eta_l^2(\frac{\beta_l}{1-\beta_l})^2\gamma^2(1-\beta_l^{t-1})^2\right)\mathbb{E}\|\nabla_l^t\|^2\end{aligned} \tag{A.15}$$

Now take $\rho_0 = 2$, and take η_l such that

$$-\frac{3}{4}\eta_l+\frac{\gamma\eta_l^2}{2}+4c_{l,1}\eta_l^2+4s\eta_l^2(\frac{\beta_l}{1-\beta_l})^2\gamma^2(1-\beta_l^{t-1})^2\leq-\frac{\eta_l}{2}, \tag{A.16}$$

the coefficient of $\mathbb{E}\|\nabla_l^t\|$ can be greatly simplified. Note that we can guarantee (A.16) if

$$\eta_l \leq \min\left\{\frac{1}{8\gamma},\frac{1}{8\gamma L}(\frac{1-\beta_l}{\beta_l})^2,\frac{1-\beta_l}{4\gamma}\sqrt[3]{\frac{1-\beta_l}{L\beta_l(1+\beta_l)}}\right\} \tag{A.17}$$

for all $l = 1, 2, ..., L$. Here each term in the right hand side of (A.17) is bounding each term on the left hand side of (A.16).

We get

$$\begin{aligned}&\sum_{l=1}^{L}\frac{\eta_l}{2}\mathbb{E}\|\nabla_l^t\|^2\leq\mathbb{E}\left[\phi^{t}\right]-\mathbb{E}\left[\phi^{t+1}\right]\\ &+2s\sum_{l=1}^{L}\left(\eta_l^2(\frac{\beta_l}{1-\beta_l})^2\gamma^2\frac{1-\beta_l}{1+\beta_l}\right)\sigma_l^2+\sum_{l=1}^{L}\frac{\gamma\eta_l^2}{2}\sigma_l^2+\sum_{l=1}^{L}2c_{l,1}\eta_l^2\frac{1-\beta_l}{1+\beta_l}\sigma_l^2.\end{aligned} \tag{A.18}$$

Note that we have defined $s := \sum_{l=1}^{L}\eta_l$. We can upper bound it by $s \leq L/\sqrt{\gamma T}$ if we assume $\eta_l \leq 1/\sqrt{\gamma T}$. Combining (A.13) and (A.17) we get:

$$\eta_l \leq \min\left\{\frac{1}{8\gamma},\frac{1}{8\gamma L}(\frac{1-\beta_l}{\beta_l})^2,\frac{1-\beta_l}{\gamma\sqrt{8\beta_l(1+\beta_l)}},\frac{1-\beta_l}{4\gamma}\sqrt[3]{\frac{1-\beta_l}{L\beta_l(1+\beta_l)}},\frac{1}{\sqrt{\gamma T}}\right\}, \tag{A.19}$$

which is satisfied if (since $\beta_l(1+\beta_l) \leq 2$)

$$\eta_l = \min\left\{\frac{1}{8\gamma},\frac{1}{8\gamma L}(\frac{1-\beta_l}{\beta_l})^2,\frac{1-\beta_l}{4\gamma},\frac{1-\beta_l}{4\gamma}\sqrt[3]{\frac{1-\beta_l}{2L}},\frac{1}{\sqrt{\gamma T}}\right\}. \tag{A.20}$$

By taking the product of two of the terms in the RHS of the above relation, we have the following upper bound on η_l^2:

$$\eta_l^2 \leq \frac{1}{8\gamma L}(\frac{1-\beta_l}{\beta_l})^2\frac{1}{\sqrt{\gamma T}}. \tag{A.21}$$

Plugging (A.21) and (A.14) into (A.18), we obtain:

$$\begin{aligned}\sum_{l=1}^{L}\eta_l\mathbb{E}\|\nabla_l^t\|^2\leq&2(\mathbb{E}\left[\phi^{t}\right]-\mathbb{E}\left[\phi^{t+1}\right])\\ &+\sum_{l=1}^{L}\left(\frac{1-\beta_l}{1+\beta_l}\frac{1}{4\gamma T}+\frac{1}{2T}+\frac{1-\beta_l}{\beta_l^3}\frac{\sqrt{\gamma}}{4L^2T^{3/2}}\right)\sigma_l^2.\end{aligned} \tag{A.22}$$

Now since $\beta_l \leq 1-\delta$, we have that η_l is lower bounded

$$\begin{aligned}\eta_l &= \min\left\{\frac{1}{8\gamma}, \frac{1}{8\gamma L}(\frac{1-\beta_l}{\beta_l})^2, \frac{1-\beta_l}{4\gamma}, \frac{1-\beta_l}{4\gamma}\sqrt[3]{\frac{1-\beta_l}{2L}}\right\} \\ &\geq \min\left\{\frac{1}{8\gamma}, \frac{\delta^2}{8\gamma L}, \frac{\delta}{4\gamma}, \frac{\delta}{4\gamma}\sqrt[3]{\frac{\delta}{2L}}, \frac{1}{\sqrt{\gamma T}}\right\} =: \eta \end{aligned} \tag{A.23}$$

we obtain:

$$\begin{aligned}\sum_{l=1}^{L}\mathbb{E}\|\nabla_l^t\|^2 \leq & \frac{2(\mathbb{E}\left[\phi^t\right] - \mathbb{E}\left[\phi^{t+1}\right])}{\eta} \\ & + \frac{1}{\eta}\sum_{l=1}^{L}\left(\frac{1-\beta_l}{1+\beta_l}\frac{1}{4\gamma T} + \frac{1}{2T} + \frac{1-\beta_l}{\beta_l^3}\frac{\sqrt{\gamma}}{4L^2T^{3/2}}\right)\sigma_l^2. \end{aligned} \tag{A.24}$$

Now by telescoping sum of (A.22) for $t = 1, ..., T$, also notice that $1/\eta = \Theta(\sqrt{\gamma T}L\gamma/\delta^2)$ as T grows, we get our final result of

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{l=1}^{L}\mathbb{E}\|\nabla_l^t\|^2 \leq \frac{2L\gamma^{3/2}\mathbb{E}\Delta_1}{\delta^2\sqrt{T}} + \sum_{l=1}^{L}\left(\frac{1-\beta_l}{1+\beta_l}\frac{L\sqrt{\gamma}}{4\sqrt{T}} + \frac{L\gamma^{3/2}}{2\sqrt{T}} + \frac{1-\beta_l}{\beta_l^3}\frac{\gamma^2}{4LT}\right)\frac{\sigma_l^2}{\delta^2}. \tag{A.25}$$

This completes the proof. □