ACL 2025 VIENNA

Taming LLMs with Gradient Grouping

Siyuan Li[1,2,*], Juanxi Tian[2,4,*], Zedong Wang[3,*], Xin Jin[2], Zicheng Liu[1,2,†], Wentao Zhang[4], Dan Xu[3]

[1]Zhejiang University, [2]Westlake University, [3]HKUST [4]Peking University

Project page

Introduction and Contributions

Training LLMs poses challenges due to their massive scale and heterogeneous architectures. While adaptive optimizers like AdamW help address gradient variations, they still struggle with efficient and effective parameter-wise learning rate estimation, resulting in training instability, slow convergence, and poor compatibility with PEFT techniques.

- Scaling with Gradient Grouping (SGG) is a flexible LLM optimizer wrapper that scales adaptive learning rates with online grouping constraints rather than replace them in pre-defined groups (like Adam-mini), balancing parameter-wise dynamics and collective optimization behavior.
- Practically, SGG integrates seamlessly with existing optimizers and PEFT techniques, requiring no changes to the training pipeline or model architectures with the CPU and CPU-GPU hybrid implementations.
- SGG's consistent improvement shows the potential of scaling adaptive learning rates with group-wise constraints with LLMs (pre-training, PEFT, RL tasks) and MLLMs. SGG offers an intuitive instantiation of this scheme, while different grouping and scaling strategies are conceivable and might inspire future studies.

Overview of SGG Optimization

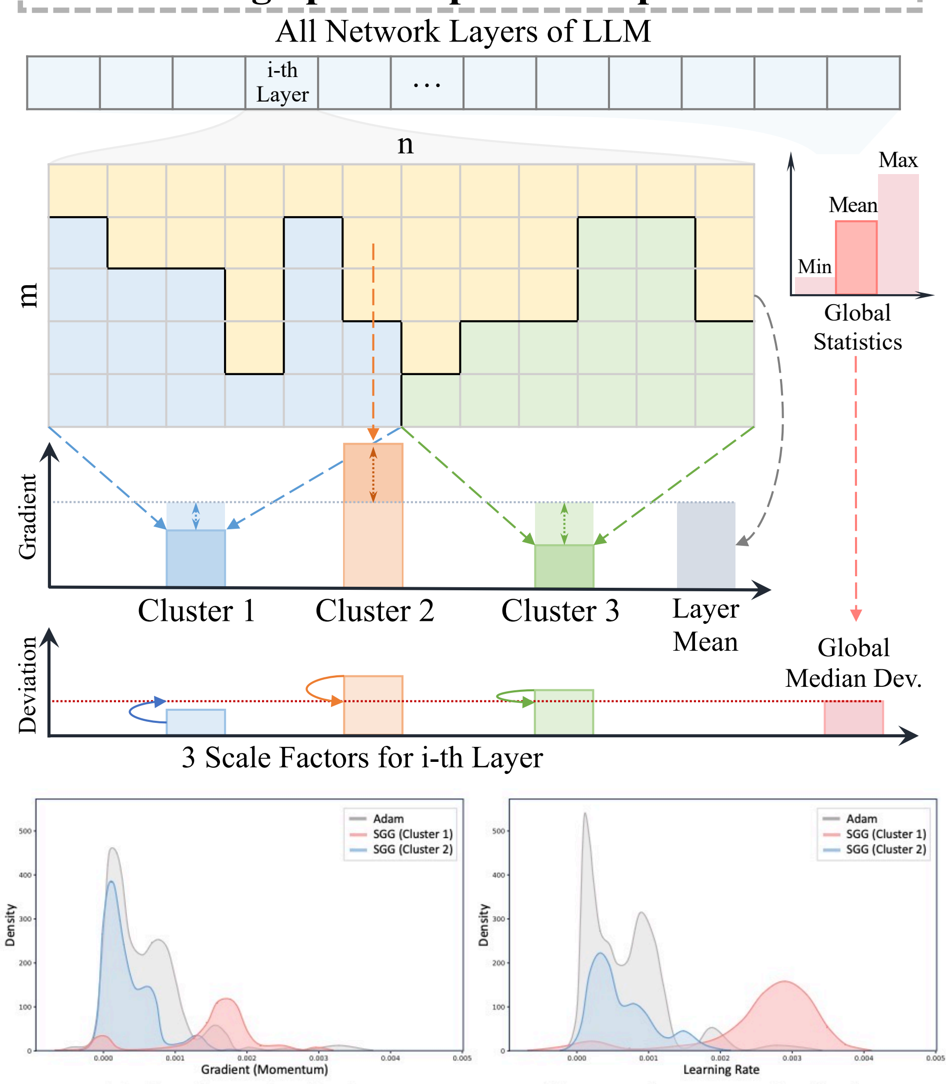
Algorithm 1 Scaling with Gradient Grouping

Require: Parameters $\{\theta_l\}_{l=1}^{L}$, global learning rate schedule η, optimizer hyperparameters (β_1, β_2), objective $\mathcal{L}$, dataset $\mathcal{D}$. SGG hyperparameters: cluster number K, recluster interval T, scaling EMA decay β_3.

Ensure: Optimized model parameters θ.

Initialize:
RandomInit($\{\theta_l^0\}_{l=1}^{L}$) ▷ Model parameters
$\{\alpha_l^0\}_{l=1}^{L} \leftarrow \eta$ ▷ Adaptive learning rates
$\{\mathcal{C}_l\}_{l=1}^{L} \leftarrow \mathbf{0}$ ▷ Cluster assignment
$\{\mathcal{S}_l\}_{l=1}^{L} \leftarrow \mathbf{1}$ ▷ Cluster scaling factor
for each iteration $t = 1, 2, \dots$ **do**
  $\eta^t \leftarrow$ LRScheduler(η, t)
  for each layer $l = 1, 2, \dots, L$ **do**
    // — Standard Gradient-based Update Steps —
    Gradient Computation
    $g_l^t \leftarrow \nabla_{\theta_l^{t-1}} \mathcal{L}(\theta^{t-1}, \mathcal{D})$
    Momentum Estimation
    $m_l^t \leftarrow \text{MomentumEstimate}(g_l^t, m_l^{t-1}, \beta_1)$
    Adaptive Learning Rate Estimation
    $\alpha_l^t \leftarrow \text{LREstimate}(\alpha_l^{t-1}, m_l^t, \beta_2, \eta^t)$
    // — SGG Specific Steps —
    if $t \mod T == 0$ **then** ▷ Re-clutering
      Assign Gradient Clusters
      $\mathcal{C}_l^t \leftarrow \text{GradCluster}(m_l^t, K)$
      Update Cluster Scaling Factors
      $\mathcal{S}_l^t \leftarrow \text{ScaleUpdate}(\mathcal{C}_l^t, m_l^t, \beta_3)$
    end if
    Apply Learning Rate Scaling
    $\alpha_l^t \leftarrow \alpha_l^t \cdot \mathcal{S}_l^t[\mathcal{C}_l^t]$ ▷ Cluster-specific scaling
    Parameter Update
    $\theta_l^t \leftarrow \theta_l^{t-1} - \alpha_l^t \cdot m_l^t$
  end for
end for

SGG with online grouping and group-specific LR scaling upon adaptive LR optimizer.



(a) Gradiant Distribution (b) Learning Rate Distribution

Comparison Experiments

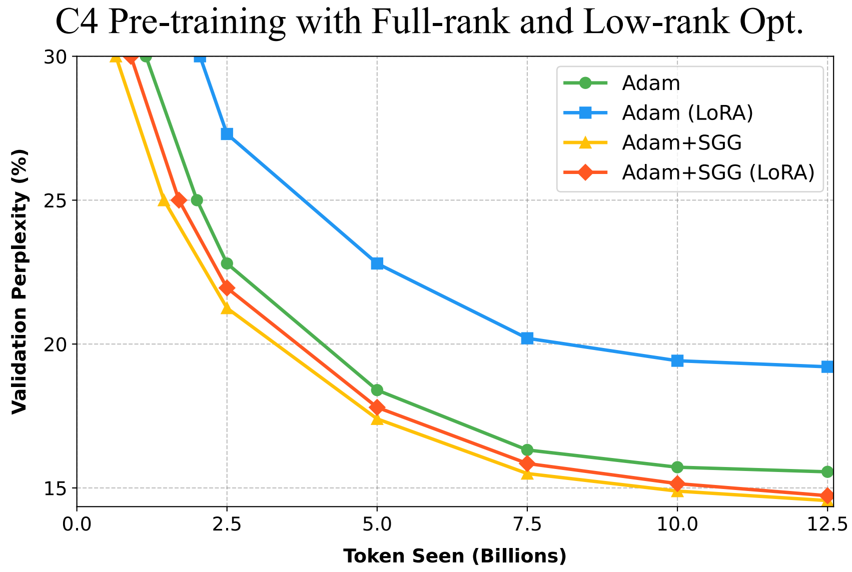
LLM Comparison: C4 Pre-training with diverse LLaMA sizes and popular LLM optimizers.

Method	Venue	60M	130M	350M	1B
Pre-training with Full-Rank Optimizers					
Adam†	ICLR'15	34.06	25.08	18.80	15.56
NAdam	ICLR'18	35.86	28.88	19.24	15.78
RAdam	ICLR'20	30.43	25.17	19.13	15.65
LAMB	ICLR'20	33.04	24.37	18.26	15.84
Adan	TPAMI'23	32.01	23.14	17.32	14.70
Adam+**SGG**	**Ours**	**30.31**	**22.18**	**17.28**	**14.30**
△ *Gains*		**-3.75**	**-2.90**	**-1.52**	**-1.26**
Pre-training with Memory-efficient Optimizers					
Adam-mini†	ICLR'25	34.10	24.85	19.05	16.07
Adafactor†	ICML'18	32.57	23.98	17.74	15.19
Low-Rank†	arXiv'22	78.18	45.51	37.41	34.53
CAME	ACL'23	31.37	23.38	17.45	14.68
CAME+**SGG**	**Ours**	**30.15**	22.91	17.09	14.35
△ *Gains*		**-1.22**	**-0.46**	**-0.36**	**-0.33**
APOLLO†	MLSys'25	31.55	22.94	16.85	14.20
APOLLO+**SGG**	**Ours**	30.18	**22.52**	**16.54**	**13.95**
△ *Gains*		**-1.37**	**-0.42**	**-0.31**	**-0.25**
Low-Rank Pre-training					
LoRA†	ICLR'22	34.99	33.92	25.58	19.21
ReLoRA†	ICLR'23	37.04	29.37	29.08	18.33
GaLore†	ICML'24	34.88	25.36	18.95	15.64
LoRA+**SGG**	**Ours**	**30.62**	**23.62**	**17.86**	**14.73**
△ *Gains*		**-4.37**	**-10.30**	**-7.72**	**-4.48**
Training Tokens		1.1B	2.2B	6.4B	13.1B

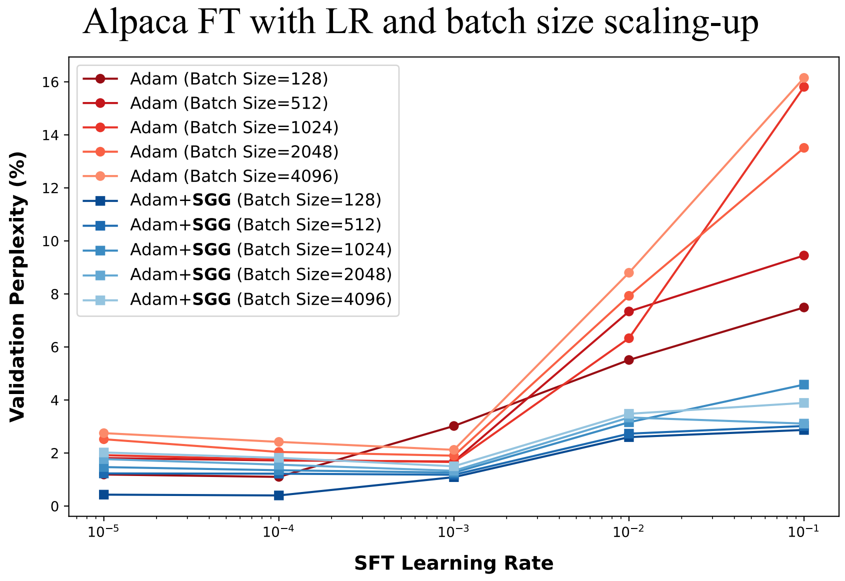
Considering a neural net layer $W \in \mathbb{R}^{m \times n}$ $(m \leq n)$ with LoRA's rank $r \leq n$ and SGG's clusters $K \ll m$.

Category	Method	Adaptive LR	Basic State	Extra State	Low-Rank	Plugin	Extra Branch	C4↓	GPU Memory
Classical Opt.	SGD	✗	Weight & Grad.	✗	✗	✗	✗	–	$2mn$
Adaptive LR Opt.	Adam	Param-wise mn	Weight & Grad.	2nd-Moment mn	✗	✗	✗	23.36	$3mn$
Efficient Opt.	CAME	Param-wise mn	Weight & Grad.	NMF $2(m+n)$	NMF	✗	✗	-1.64	$2mn+2(m+n)$
PEFT	LoRA	✗	Full-rank Grad.	✗	LoRA	✓	$r(m+n)$	+5.06	$+3r(m+n)$
Opt. Wrapper	**SGG**	Group-wise K	Base Opt.	Indices $(mn+K)$	Clustering	✓	✗	**-1.99**	+0

C4 Pre-training with Full-rank and Low-rank Opt.



Alpaca FT with LR and batch size scaling-up



MLLM Comparison: Top-1 accuracy (%) with LLaVA variants is reported, where MMB and MMB^{CN} denote MMbench and MMbench (Chinese).

Optimizer	Image Question Answering				Benchmarks			Avg.
	GQA	VizWiz	SciVQA^I	VQA^T	MMB	MMB^{CN}	POPE	
LLaVA-v1.5 Full-Rank SFT								
AdamW	62.0	50.0	66.8	**58.2**	64.3	58.3	85.9	63.6
Adafactor	62.7	48.2	70.7	57.1	66.1	60.4	86.0	64.5
LAMB	43.8	53.3	61.5	43.4	43.2	41.8	81.2	52.6
AdamW+**SGG**	62.4	50.2	69.8	57.4	65.9	60.1	**86.3**	64.6
△ *Gains*	**+0.4**	**+0.2**	**+3.0**	**-0.8**	**+1.6**	**+1.8**	**+0.4**	**+1.0**
Adafactor+**SGG**	**62.8**	50.6	**71.6**	57.3	**66.3**	**60.8**	86.0	**65.1**
△ *Gains*	**+0.1**	**+2.4**	**+0.9**	**+0.2**	**+0.2**	**+0.4**	**+0.0**	**+0.6**
LAMB+**SGG**	44.0	**53.3**	61.8	43.5	43.3	41.9	81.3	52.7
△ *Gains*	**+0.2**	**+0.0**	**+0.3**	**+0.1**	**+0.1**	**+0.1**	**+0.1**	**+0.1**
LLaVA-v1.5 Low-Rank SFT (AdamW)								
LoRA	63.0	47.8	68.4	58.2	66.1	58.9	86.4	64.1
LoRA+**SGG**	**63.4**	**51.0**	**70.1**	**58.6**	**66.7**	**59.4**	**86.6**	**65.1**
△ *Gains*	**+0.4**	**+2.2**	**+1.5**	**+0.4**	**+0.6**	**+0.5**	**+0.2**	**+1.0**
LLaVA-v1.5 8-bit Low-Rank SFT (AdamW)								
Q-LoRA	54.3	50.7	66.4	52.5	56.0	49.8	82.9	58.9
Q-LoRA+**SGG**	**55.1**	**51.3**	**66.7**	**53.0**	**56.1**	**51.0**	**83.4**	**59.5**
△ *Gains*	**+0.8**	**+0.6**	**+0.3**	**+0.5**	**+0.1**	**+0.2**	**+0.5**	**+0.6**