SMT: Fine-Tuning Large Language Models with Sparse Matrices

Haoze He*[1], Juncheng Billy Li*[1,2], Xuan Jiang[3], Heather Miller[1,2]

[1]Carnegie Mellon University, [2]Two Sigma Investments, [3]University of California, Berkeley

Paper GitHub

How to select most salient parameters for fine-tuning?

Existing fine-tuning methods of Large Language Models (LLMs) involve trade offs.

- **Full fine-tuning** updates *billions* of parameters → **expensive, prone to overfit**
- **Existing PEFT methods** (LoRA, adapters, etc.) reduce compute cost but still incur a **performance gap** between reparameterization methods and full fine-tuning.

Can we systematically identify *which* parameters matter most?

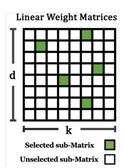

Figure 1. Gradient-Aware Selection

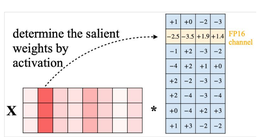

Figure 2. Activation-Aware Selection

We experimented with both Gradient-aware selection and activation-aware selection. **Gradient-aware selection produced much better downstream results** than activation-aware selection.

Gradient-Aware Selection prefers V matrices!

QKV SMT assign all trainable parameters to QKV vectors and select sub-matrices automatically. **95.17% of the trainable parameters are automatically assigned to the V vectors by SMT.**

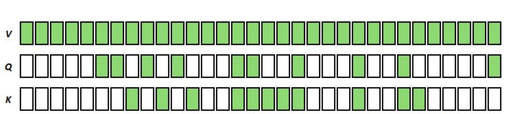

Figure 3. A visualization of trainable Q, K, V layers when fine-tuning 0.86% trainable parameters on LLaMA-7B. All V vectors have trainable parameters, while 22 out of 32 Q vectors and 21 out of 32 K vectors are completely frozen.

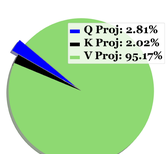

Figure 4. Distribution of trainable parameters among Q, K, V.

Model	Param location	#Params%	BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	AVG
LLaMA-7B	K SMT	0.84	65.5	79.1	76.2	88.3	73.2	80.3	60.8	68.0	73.9
	Q SMT	0.84	65.7	79.3	75.5	88.2	72.5	80.1	59.6	72.5	75.3
	V SMT	0.84	68.7	82.1	78.1	91.6	78.8	83.0	68.7	77.2	78.5
	QKV SMT	0.84	68.7	81.7	78.3	91.6	78.8	84.1	68.7	77.4	78.7

Table 1. K SMT, Q SMT, and V SMT assign all trainable parameters to only K, or only Q, or only V vectors respectively, and fine-tuned 0.86% of the parameters on LLaMA-7B.

Proposed Method - Sparse Matrices

In SMT, we slice the pre-trained weight into N sub-matrices and only fine-tune selected M sub-matrices. The dimension of each sub-matrices is l × l. The number of fine-tuning sub-matrices m ≪ N.

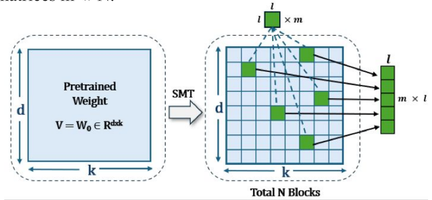

Figure 5. SMT workflow visualization

- SMT selects **salient submatrices** using gradient statistics from a short warm-up phase, enabling targeted parameter updates without architectural changes.
- SMT **reduces memory cost by >5×** compared to sparse masks, and achieves **>95% of full fine-tuning accuracy** while updating **<1% of parameters**.

Cause: V matrices gradient magnitude is 10x larger than Q & K

- This larger gradient leads to more substantial updates, making fine-tuning the V vectors more effective.
- The gradient of the softmax with respect to Q becomes small when $QK^\top$ is large and $\sqrt{d_k}$ is not large enough due to the saturation of the softmax function. (Appendix C)

$$\text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)$$

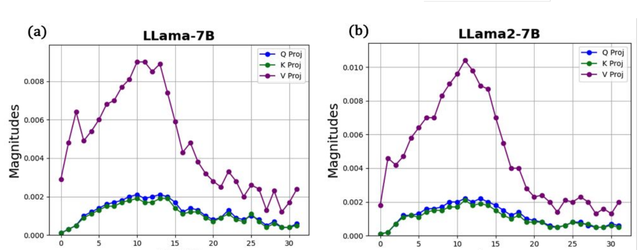

Figure 6. The magnitude of the gradient for the Q, K, and V vectors at each layer in LLMs averaged over 1000 iterations.

System Efficiency - competitive cost and better performance

- SMT updates only a subset of parameters, reducing backward compute to **0.5%** of full fine-tuning.
  → *Offers up to **14× speedup** vs. full fine-tuning with CPU offloading.*
- SMT computes **partial gradients only**, cutting forward-pass memory cost by **50%**.
  → ***Saves 50GB of activation memory** at 256 batch size.*
- SMT reduces the memory costs of the optimizer gradients to 0.5%.
  → ***Saves 67% GPU memory, 105GB** for 13B models*

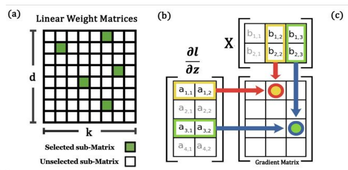

Figure 7. SMT Partial Backward Propagation: Weight updates and compute graph

LLaMA-7B			
PEFT method	**#Params%**	**Time/s**	**Speedup**
Full Fine-tuning	100	243.84	1×
SMT	1.26	16.68	14.6×
LoRA	1.26	17.82	13.6×
DoRA	1.27	18.04	13.5×
SpIEL	1.26	25.45	9.6×

Table 2. Speedup Compared to SFT and other SOTAs

Performance - CommonSense Reasoning NLP Benchmark

Model	PEFT method	#Params%	BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	AVG
ChatGPT(175B)	-	-	73.1	85.4	68.5	78.5	66.1	89.8	79.9	74.8	77.0
LLaMA-7B	LoRA(Best)	0.83	67.5	80.8	78.2	83.4	80.4	78.0	62.6	79.1	76.3
	DoRA(Best)	0.84	69.7	83.4	78.6	87.2	81.0	81.9	66.2	79.2	78.4
	SpIEL(Best)	0.84	67.7	81.2	78.6	84.0	80.2	78.3	62.8	78.8	76.5
	SMT	0.84	68.7	81.7	78.3	91.6	78.8	84.1	68.7	77.4	**78.7**
	SMT(Best)	4.91	72.0	82.9	80.7	93.3	82.4	86.1	70.6	83.0	**81.4**
	Full Fine-tuning	100	69.9	84.2	78.9	92.3	83.3	86.6	72.8	83.4	81.4
LLaMA-13B	LoRA(Best)	0.67	72.1	83.5	80.5	90.5	83.7	82.8	68.3	82.4	80.5
	DoRA(Best)	0.68	72.4	84.9	81.5	92.4	84.2	84.2	69.6	82.8	81.5
	SpIEL(Best)	0.68	73.2	84.3	81.4	91.2	84.1	83.1	68.8	82.8	81.1
	SMT	0.68	71.1	84.4	81.7	93.7	83.2	86.7	73.7	85.2	**82.4**
	SMT(Best)	4.91	72.6	86.1	81.9	95.0	86.1	88.2	77.1	87.4	**84.3**
LLaMA2-7B	LoRA(Best)	0.83	69.8	79.9	79.5	83.6	82.6	79.8	64.7	81.0	77.6
	DoRA(Best)	0.42	72.0	83.1	79.9	89.1	83.0	84.5	71.0	81.2	80.5
	SpIEL(Best)	0.83	70.5	80.6	80.8	85.8	83.4	81.2	65.8	81.8	78.3
	SMT	0.84	72.0	83.8	80.8	93.3	82.8	86.7	74.0	81.0	**81.8**
	SMT(Best)	4.91	72.6	85.2	82.0	94.4	85.7	87.8	74.5	85.0	**83.4**
	Full Fine-tuning	100	72.8	83.4	78.7	92.7	85.5	86.2	74.7	83.4	82.2
LLaMA3-8B	LoRA(Best)	0.70	70.8	85.2	79.9	91.7	84.3	84.2	71.2	79.0	80.8
	DoRA(Best)	0.71	74.6	89.3	79.9	95.5	85.6	90.5	80.4	85.8	85.2
	SpIEL(Best)	0.70	72.1	83.6	80.0	91.8	85.4	91.2	76.8	80.8	82.7
	SMT	0.71	75.7	88.4	81.4	96.2	88.2	92.7	83.2	88.6	**86.8**
	SMT(Best))	3.01	75.1	89.9	82.4	96.3	88.8	92.6	82.8	89.6	**87.2**

Bold texts dedicate the performance of SMT under the same numbers of parameters where LoRA, DoRA, and SpIEL achieve the best performance.

Blue texts dedicate the best performance of SMT.

Table 3. Accuracy comparison of LLaMA 7B, LLaMA 13B, LLaMA-2-7B, and LLaMA-3-8B

SMT for Deepseek-Distill Model (Recent Results)

Table 4. Accuracy comparison of DeepSeek-R1-Distill models

Model	PEFT method	#Params%
Deepseek-R1-Distill-LLaMA-8B	SMT	
	Full FT	