Winner-Take-All Column Row Sampling for Memory Efficient Adaptation of Language Model

Zirui Liu[1*], Guanchu Wang[1*], Shaochen Zhong[1], Zhaozhuo Xu[2], Daochen Zha[1], Ruixiang Tang[1], Zhimeng Jiang[3], Kaixiong Zhou[1], Vipin Chaudhary[4], Shuai Xu[4], Xia Hu[1]

[1]Rice University, [2]Stevens Institute of Technology, [3]Texas A&M University, [4]Case Western Reserve University

{zl105, gw22, hz88, daochen.zha, ruixiang.tang, kaixiong.zhou, xia.hu}@rice.edu; zxu79@stevens.edu; zhimengj@tamu.edu; {vipin, sxx214}@case.edu

Memory Bottleneck of Finetuning LLMs

- **Forward phase and Backward phase of LLM Finetuning:**

$$\texttt{Forward Pass} \quad \boldsymbol{Z} = \texttt{MatMul}(\boldsymbol{H}, \boldsymbol{W}),$$
$$\texttt{Backward Pass} \quad \nabla \boldsymbol{H} = \texttt{MatMul}(\nabla \boldsymbol{Z}, \boldsymbol{W}^\top),$$
$$\nabla \boldsymbol{W} = \texttt{MatMul}(\boldsymbol{H}^\top, \nabla \boldsymbol{Z}),$$

where $\texttt{MatMul}(\cdot,\cdot)$ is the General Matrix Multiplication operation, $\boldsymbol{H}$ and $\boldsymbol{Z}$ are the activation and output feature maps, respectively. $\boldsymbol{W}$ is the weight. $\nabla\boldsymbol{H}$, $\nabla\boldsymbol{W}$, and $\nabla\boldsymbol{Z}$ are the gradient of $\boldsymbol{H}$, $\boldsymbol{W}$, and $\boldsymbol{Z}$, respectively. The activations $\boldsymbol{H}$ are stored $\boldsymbol{H}$ in GPU memory during the forward pass, for calculating the weight gradient $\nabla\boldsymbol{W}$ in the backward pass.

- **Memory Bottleneck of LLM Finetuning:** Although the model parameters contribute to the memory footprint, activations (e.g., storing $\boldsymbol{H}$) are the main memory bottleneck during training. As shown in the right-side figure, for T5 models, activations may take roughly $73 \sim 88\%$ of the total memory, depending on the batch size B and sequential length S.

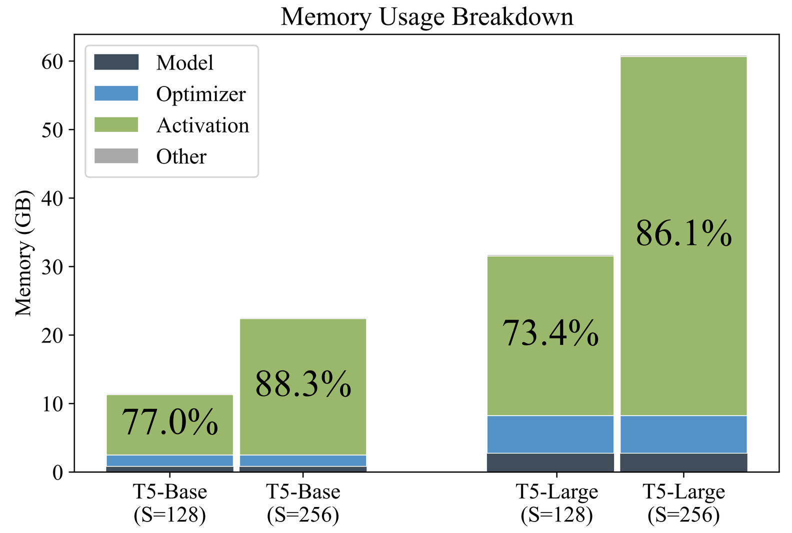


Figure. The GPU memory usage for fine-tuning T5. The batch size is 64 and sequential length is 128 or 256.

Winner Takes All Column-Row Sampling

- **WTA-CRS Estimator.** WTA-CRS estimator defined in Equation (5) splits the budget k into two parts. Namely, the first part explicitly sums the expectation terms for the largest probability group $\mathcal{C}$ ($|\mathcal{C}| < k$), while stochastically average $k - |\mathcal{C}|$ samples drawn from $\mathcal{D}\backslash\mathcal{C}$ to estimate the remaining terms, up to scale:

$$(\texttt{WTA-CRS}) \quad \hat{g}(\boldsymbol{X},\boldsymbol{Y}) = \sum_{c\in\mathcal{C}} f(c)p(c) + \frac{1-\sum_{c\in\mathcal{C}} p_c}{k-|\mathcal{C}|} \sum_{j=1}^{k-|\mathcal{C}|} f(j), \quad i_1,\cdots,i_{k-|\mathcal{C}|} \overset{\text{i.i.d}}{\sim} \mathcal{P}^{\mathcal{D}\backslash\mathcal{C}}. \tag{5}$$

- **System Implementation.** As shown in the following right-side figure, a transformer block consists of linear layers, `TensorMul`, and other operations (e.g., `GeLU`, `Dropout`, `LayerNorm`). `TensorMul` refers to the multiplication between two four-dimensional tensors. Our WTA-CRS can be applied to the backward pass of Linear-Q, -K, -V, -O, -U, -D, TensorMul-1, and TensorMul-2 (in green), while leaving the forward pass unchanged, as shown in the following left-side figure. The activations of Dropout and GELU operations (in blue) can be losslessly compressed. The Softmax and LayerNorm operators (in gray) remain unchanged.

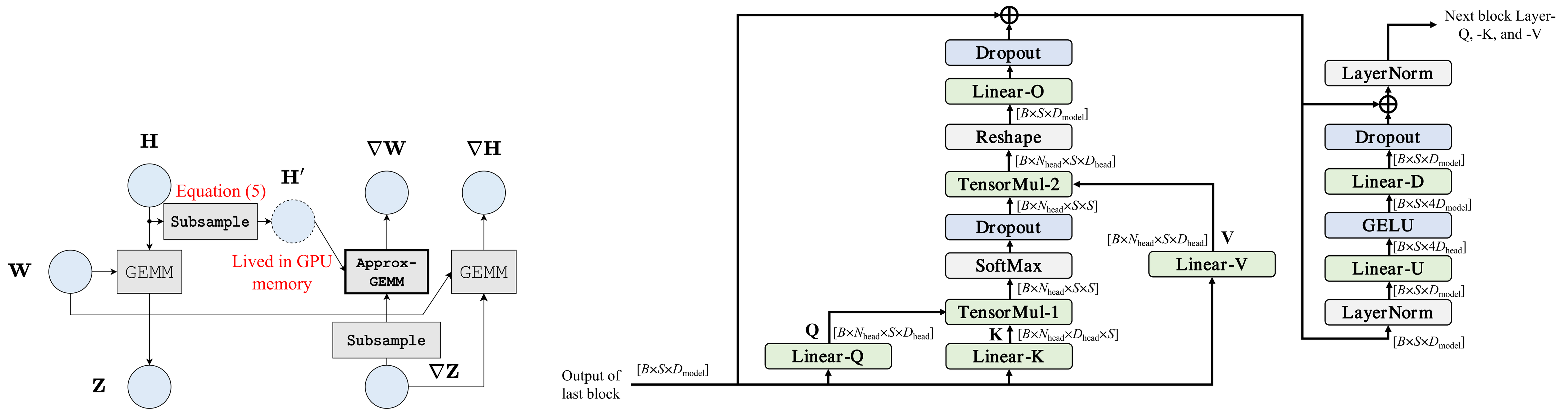


Figure. **Left:** The illustration of how to deploy WTA-CRS to linear layers. **Right:** Application of WTA-CRS to a Transformer block. B, S, D_{model}, N_{head}, and D_{head} are the batch size, sequence length, hidden size, number of attention heads, and head dimension, respectively. WTA-CRS is applied to the operators in green; the activation maps of operators in blue can be losslessly compressed; and those in gray are not compressed.

Memory Cost and Accuracy on the GLUE Datasets

Table. Peak memory usage (GB) and compression rate of fine-tuning T5-Base and -Large.

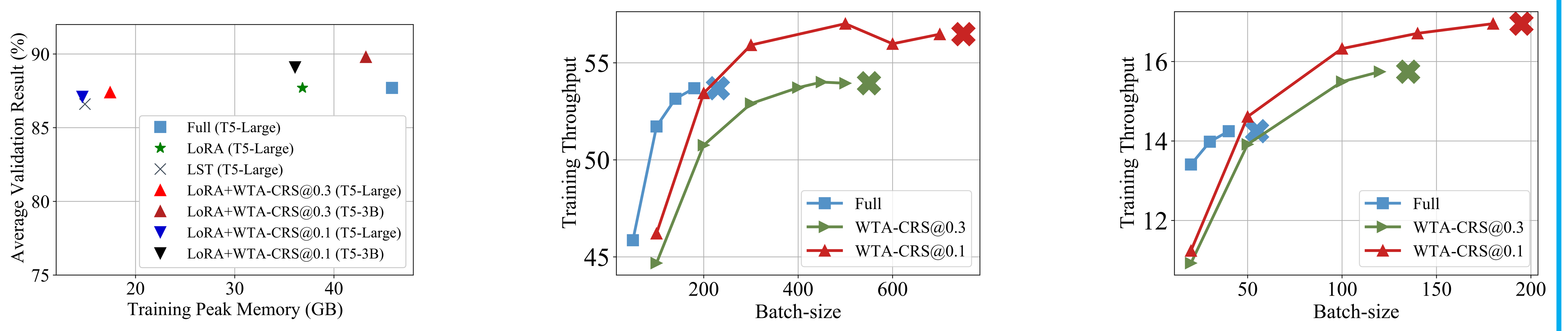
	FP	LoRA	LST	WTA-CRS@0.3	WTA-CRS@0.1	LoRA+WTA-CRS@0.3	LoRA+WTA-CRS@0.1
T5-Base	17.66 (1×)	13.84 (1.3×)	5.50 (3.2×)	8.44 (2.1×)	7.30 (2.4×)	6.50 (2.7×)	5.44 (3.2×)
T5-Large	45.85 (1×)	36.83 (1.2×)	14.85 (3.1×)	21.58 (2.1×)	18.46 (2.5×)	17.44 (2.6×)	14.64 (3.13×)

Table. The GLUE benchmark results with T5 and Bert at different scales.

Model	Method	CoLA	SST-2	MRPC	QQP	MNLI	QNLI	RTE	STS-B	AVG
BERT-Large	Full	$66.8_{\pm0.31}$	$93.5_{\pm0.29}$	$89.5_{\pm0.26}$	$88.5_{\pm0.03}$	$86.4_{\pm0.19}$	$92.1_{\pm0.24}$	$72.6_{\pm0.36}$	$90.2_{\pm0.76}$	85.0
	LoRA	$65.9_{\pm0.27}$	$93.8_{\pm0.17}$	$90.8_{\pm0.37}$	$87.6_{\pm0.08}$	$85.9_{\pm0.05}$	$92.0_{\pm0.2}$	$71.3_{\pm0.18}$	$90.3_{\pm0.09}$	84.7
	WTA-CRS@0.3	$64.7_{\pm0.44}$	$93.5_{\pm0.0}$	$89.3_{\pm0.39}$	$88.2_{\pm0.04}$	$85.2_{\pm0.03}$	$91.9_{\pm0.12}$	$73.8_{\pm0.54}$	$90.4_{\pm0.02}$	84.6
	LoRA+WTA-CRS@0.3	$66.0_{\pm0.33}$	$93.3_{\pm0.29}$	$89.7_{\pm1.32}$	$87.6_{\pm0.02}$	$86.0_{\pm0.07}$	$91.9_{\pm0.14}$	$72.4_{\pm0.17}$	$89.7_{\pm0.04}$	84.6
T5-Large	Full	$61.3_{\pm1.01}$	$96.3_{\pm0.0}$	$93.4_{\pm0.13}$	$89.7_{\pm0.01}$	$89.8_{\pm0.07}$	$94.2_{\pm0.05}$	$85.3_{\pm0.17}$	$91.8_{\pm0.08}$	87.7
	LoRA	$63.3_{\pm0.26}$	$96.4_{\pm0.14}$	$93.5_{\pm0.16}$	$88.5_{\pm0.03}$	$89.5_{\pm0.05}$	$94.3_{\pm0.07}$	$84.2_{\pm0.68}$	$91.7_{\pm0.13}$	87.7
	LST	$59.9_{\pm0.77}$	$95.8_{\pm0.06}$	$91.8_{\pm0.08}$	$88.4_{\pm0.01}$	$88.7_{\pm0.05}$	$94.2_{\pm0.02}$	$82.5_{\pm0.18}$	$91.4_{\pm0.07}$	86.6
	WTA-CRS@0.3	$60.9_{\pm1.18}$	$96.3_{\pm0.25}$	$93.6_{\pm0.57}$	$89.3_{\pm0.04}$	$89.5_{\pm0.12}$	$94.1_{\pm0.03}$	$84.4_{\pm0.34}$	$91.3_{\pm0.05}$	87.4
	LoRA+WTA-CRS@0.3	$62.9_{\pm1.19}$	$96.2_{\pm0.05}$	$93.6_{\pm0.47}$	$88.3_{\pm0.02}$	$89.2_{\pm0.08}$	$94.0_{\pm0.07}$	$83.9_{\pm0.95}$	$91.3_{\pm0.03}$	87.4
T5-3B	LoRA	$70.1_{\pm0.37}$	$96.8_{\pm0.29}$	$94.0_{\pm0.27}$	$89.9_{\pm0.0}$	$91.0_{\pm0.14}$	$95.6_{\pm0.05}$	$85.9_{\pm0.36}$	$92.9_{\pm0.08}$	89.5
	LoRA+WTA-CRS@0.3	$71.4_{\pm0.35}$	$96.4_{\pm0.06}$	$94.6_{\pm0.39}$	$90.0_{\pm0.05}$	$91.0_{\pm0.06}$	$95.6_{\pm0.12}$	$86.3_{\pm0.36}$	$92.9_{\pm0.09}$	89.8

- Under similar memory budget, WTA-CRS achieves higher accuracy than other methods, improving down-streaming task performance.
- WTA-CRS achieves a superior trade-off between accuracy and memory usage compared to baselines. Specifically, WTA-CRS has negligible accuracy drop, while the peak memory usage is reduced by $2.1\times \sim 2.7\times$ (when combined with LoRA).

Experimental Results



- WTA-CRS achieves better accuracy-memory trade-off than state-of-the-art memory-efficient tuning methods, e.g., LST and LoRA.
- WTA-CRS enables faster training speed under the same hardware. On the T5-Large model, WTA-CRS@0.1 shows $1.08\times$ higher training throughput; on the T5-3B model, WTA-CRS@0.3 and WTA-CRS@0.1 achieve $1.14\times$ and $1.21\times$ higher training throughput, respectively.

Acknowledgments

The authors thank the anonymous reviewers for their helpful comments. The work is in part supported by NSF grants NSF IIS-1849085, IIS-2224843, and NSF Awards 2117439 and 2112606. This work made use of the High Performance Computing Resource in the Core Facility for Advanced Research Computing at Case Western Reserve University.