Joint Localization and Activation Editing for Low-Resource Fine-Tuning

Wen Lai[1,2] Alexander Fraser[1,2] Ivan Titov[3,4]

[1]Technical University of Munich [2]Munich Center for Machine Learning
[3]University of Edinburgh [4]University of Amsterdam

Munich Center for Machine Learning

ICML International Conference On Machine Learning

Paper Code Blog

Contact: wen.lai@tum.de

Highlight

- We introduce **JoLA**, a **parameter-efficient** fine-tuning method for **low-resource** settings. ⇒ Fewer parameters than LoRA, works with just 200 samples.
- **Main Idea:** **Activation** editing instead of weight updates (like LoRA), **dynamicly selecting** intervention components and strategy.
- **Easy to use:** 3 lines of code, fast training.

Try our code

```
pip install jola
```

Background

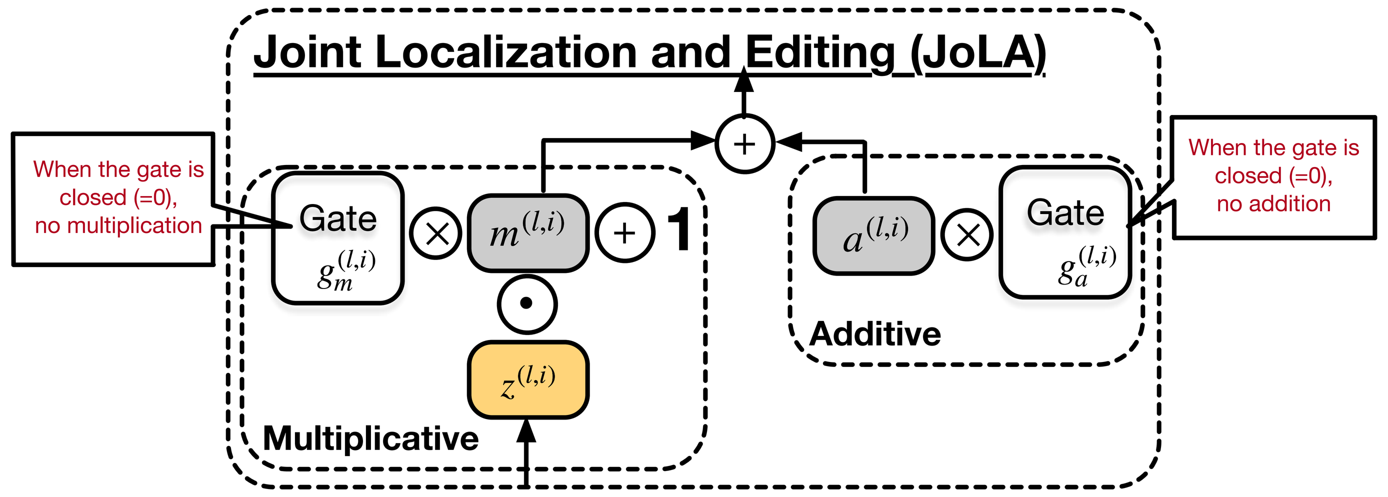
- **PEFT methods** (e.g., LoRA) are efficient but struggle in **low-resource settings**.
- **Activation editing** offers a **lightweight** alternative by modifying intermediate activations—**ideal for small datasets**.
- Key challenges remain:
  - **What to edit?** Bias terms[1], MLP outputs[2], hidden states[3], or attention heads[4]?
  - **How to edit?** Additive, multiplicative, or hybrid?
  - **Task Dependence:** Editing strategies vary by task and dataset.

Motivation

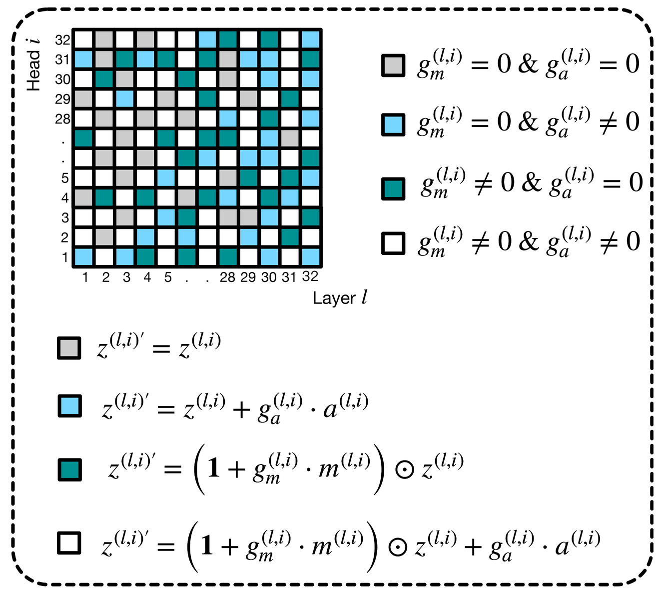
- **Component Selection:** Editing multiple components often leads to overfitting, while attention heads are more effective targets. (See section 3.1)
- **Intervention Strategy:** Bias offsets (additive) consistently contribute more to performance improvements than scaling (multiplicative). (See section 3.1)
- **Performance on low-resource settings:** Relies on fixed heuristics or manual selection, with unstable performance in low-resource settings. (See Appendix C and Apendix F.3)

Method

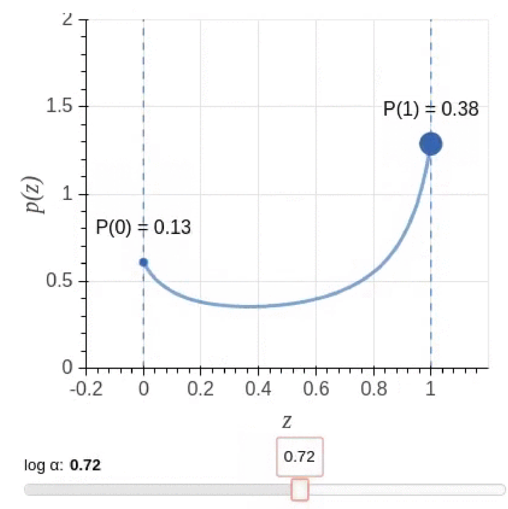
① JoLA Framework: For each head, we learn two scalar gates ($g_m^{(l,i)}$, $g_a^{(l,i)}$) and two vectors ($m^{(l,i)}$, $a^{(l,i)}$).



② Four Activation Statuses:



③ Gate: Standard concrete distributions (**No Input!**).



④ Training Objectives:

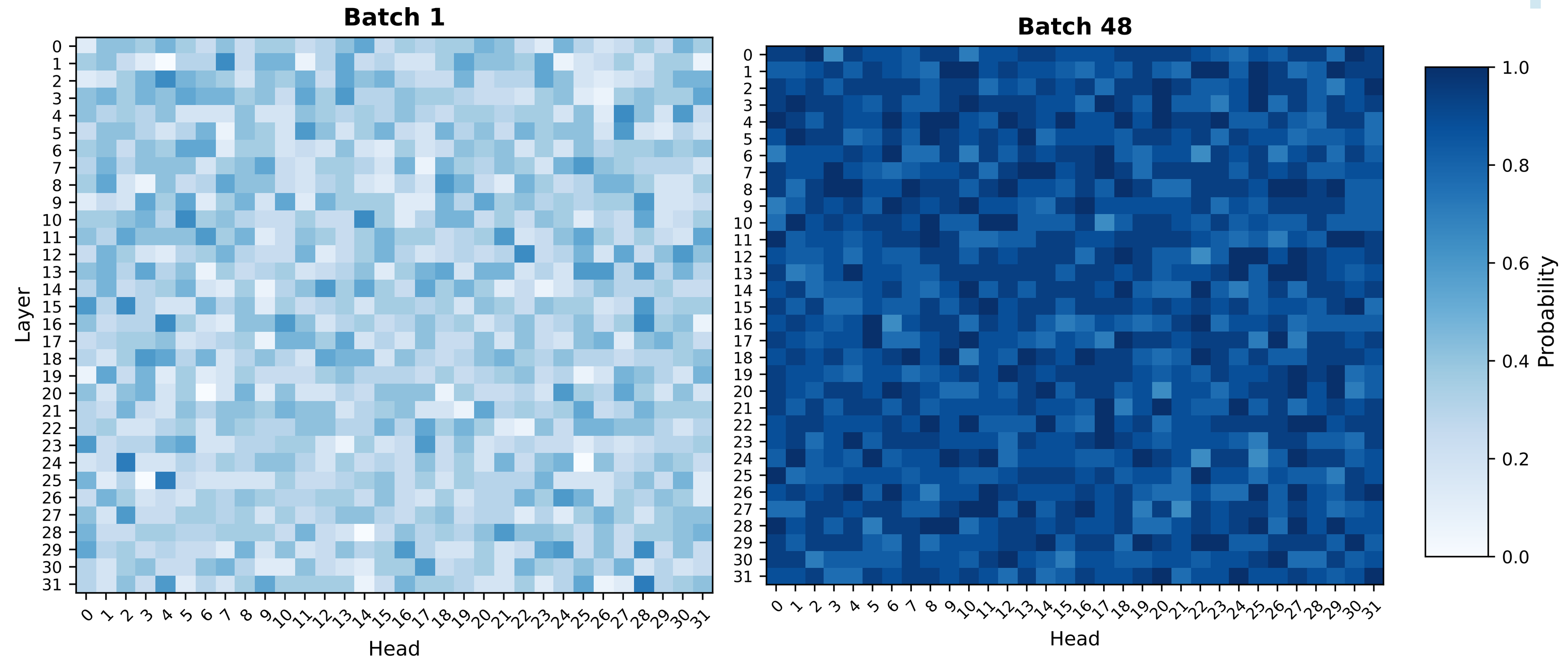
$$L(\mathbf{m}, \mathbf{a}, \phi) = L_{xent}(\mathbf{m}, \mathbf{a}) + \lambda L_C(\phi)$$

- $L_{xent}(\cdot)$ is the standard cross-entropy loss, $L_C(\phi)$ is the L_0 regularizer defined as:

$$L_C(\phi) = \sum_{l,i} \Big(1 - P(g_a^{(l,i)} = 0 \mid \phi_a^{(l,i)})$$
$$+ 1 - P(g_m^{(l,i)} = 0 \mid \phi_m^{(l,i)})\Big)$$

- $L_C(\phi)$ regularizes the number of open gates, encouraging the model to close gates as training progresses.
- Most gates are closed at convergence, i.e., only a few interventions are applied.

⑤ Gate Status during Training:



Experiments & Results

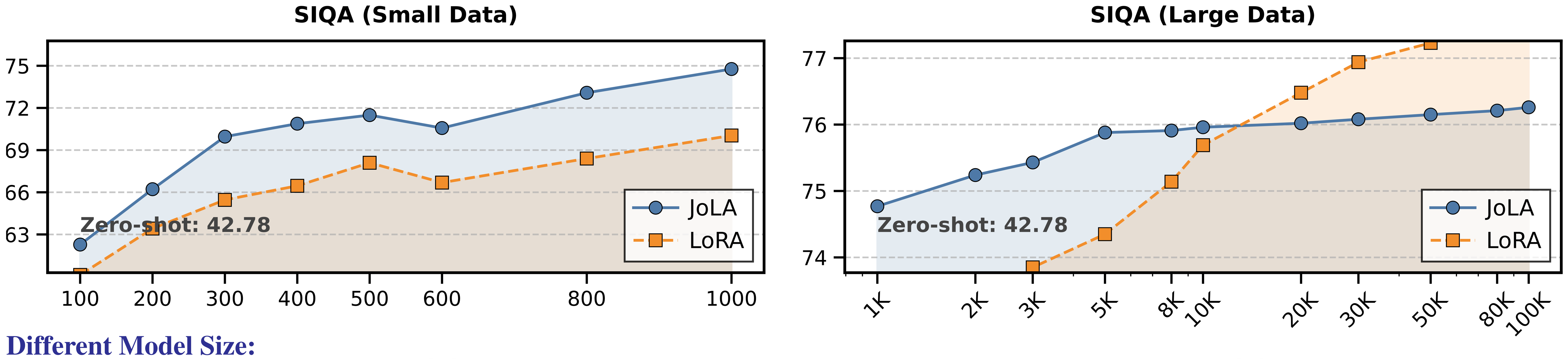
Main Results:

Llama-3.1-8B-Instruct					
	Reasoning	Understanding	Generation		
	ACC ↑	ACC ↑	BLEU ↑	ROUGE-L ↑	BERTScore ↑
zero_shot	53.70	40.00	12.56	36.70	77.23
LoRA	66.58	42.07	13.27	36.97	77.74
BitFit	63.05	35.02	9.25	28.81	74.83
RED	46.19	37.33	11.24	32.40	76.24
RePE	63.61	35.54	8.49	27.61	74.30
ReFT	65.95	40.89	12.60	36.89	77.21
LoFIT	56.19	27.76	11.88	32.09	76.71
JoLA	**70.55**	**47.00**	**17.07**	**40.65**	**80.54**

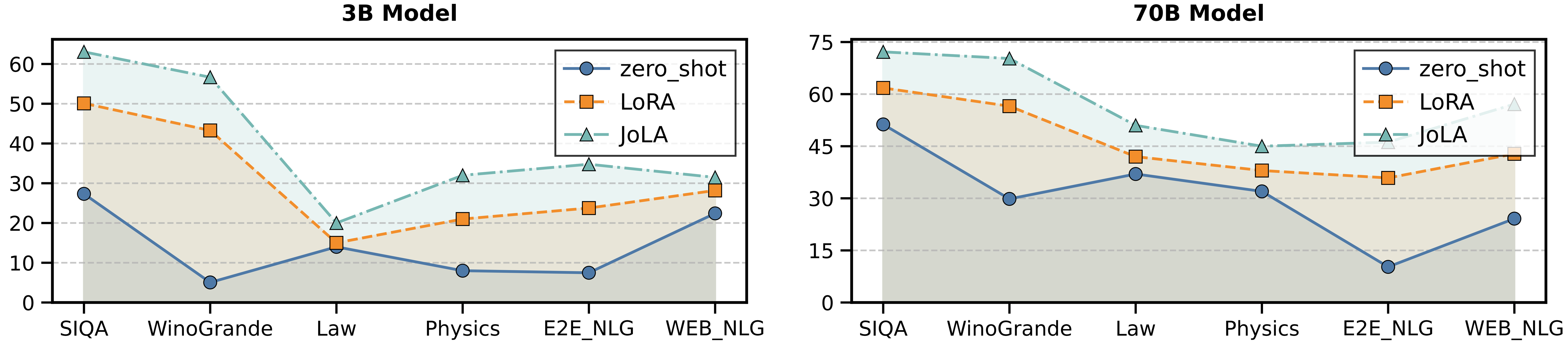
Ablation: Gate Mechnism

	Reasoning		Understanding		Generation	
	SIQA	WinoGrande	Law	Physics	E2E_NLG	WEB_NLG
MLP w/o gate	50.10	51.62	34.00	20.00	10.31	14.45
MLP with gate	**52.46**	**52.43**	**36.00**	**23.00**	**11.23**	**16.25**
Attention w/o gate	55.94	55.33	36.00	7.00	14.77	18.12
Attention with gate	**66.22**	**58.33**	**40.00**	**46.00**	**15.54**	**24.39**
Attention + MLP w/o gate	52.17	48.74	23.00	13.00	8.23	12.36
Attention + MLP with gate	**53.28**	**52.07**	**27.00**	**16.00**	**10.42**	**14.83**

Different Data Size:



Different Model Size:



References: [1] BitFIT (Ben Zaken et al., 2022) [2] RED (Wu et al., 2024a) [3] ReFT (Wu et al., 2024b) [4] LoFIT (Yin et al., 2024)