ICML International Conference On Machine Learning

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

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

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

23rd July, 2025

JoLA is ...

- a **new approach** for low-resource fine-tuning. $\rightarrow$ Comparable to LoRA on large datasets, and superior on small ones (e.g., 200 samples).

JoLA is ...

- a **new approach** for low-resource fine-tuning. → Comparable to LoRA on large datasets, and superior on small ones (e.g., 200 samples).
- a **parameter-efficient** approach. → Fewer trainable and active parameters than LoRA.

JoLA is ...

- a **new approach** for low-resource fine-tuning. → Comparable to LoRA on large datasets, and superior on small ones (e.g., 200 samples).
- a **parameter-efficient** approach. → Fewer trainable and active parameters than LoRA.
- build on **activation editing** from interpretability. → Modify the activations of selected components while keeping the rest intact.

JoLA is ...

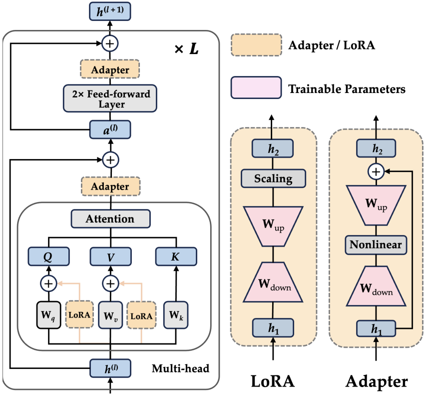
- a **new approach** for low-resource fine-tuning. → Comparable to LoRA on large datasets, and superior on small ones (e.g., 200 samples).
- a **parameter-efficient** approach. → Fewer trainable and active parameters than LoRA.
- build on **activation editing** from interpretability. → Modify the activations of selected components while keeping the rest intact.
- **user friendly**. → 3 lines of code, fast training.

Try Our Code (pip install jola)

```
# Load models
jola_model = JoLAModel.jola_from_pretrained(**jola_config["model_config"])
# Unfreeze relevant parameters
jola_model.unfreeze_jola_params()
# Train
jola_trainer.train()
```

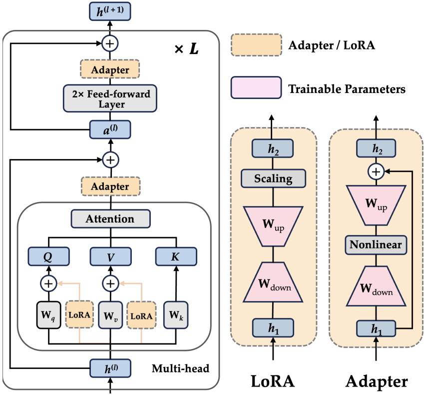
1. Background
2. Introducing JoLA
3. Experimental Setups
4. Results
5. Analysis

Parameter-Efficient Fine-Tuning (PEFT)



- **Adapters** (Houlsby et al. 2019): Learnable modules inserted after sub-layers.
- **LoRA** (Hu et al. 2021): Adds low-rank matrices in parallel to W_q and W_v in attention layers.

Parameter-Efficient Fine-Tuning (PEFT)



- **Adapters** (Houlsby et al. 2019): Learnable modules inserted after sub-layers.
- **LoRA** (Hu et al. 2021): Adds low-rank matrices in parallel to W_q and W_v in attention layers.

- PEFT avoids modifying the original weights, which makes it deployment-friendly. However, methods like LoRA still introduce additional parameters (e.g., 0.826% for LLaMA-3-8B).
- The effectiveness of standard PEFT is limited in low-resource scenarios with only a few hundred examples.

Activation Editing: A New Paradigm

Activation editing enables parameter-efficient fine-tuning by learning small interventions on internal activations, achieving strong results with minimal updates (e.g., 0.0035% for LoFIT), even in low-data regimes.

Activation Editing: A New Paradigm

Activation editing enables parameter-efficient fine-tuning by learning small interventions on internal activations, achieving strong results with minimal updates (e.g., 0.0035% for LoFIT), even in low-data regimes.

- **Intervention component**
  - Bias term - BitFIT (Ben Zaken et al. 2022)
  - MLP layers output - RED (Wu et al. 2024a)
  - Hidden outputs (representation) within MLP layers - ReFT (Wu et al. 2024b)
  - Attention head outputs - LoFIT (Yin et al. 2024)

Activation Editing: A New Paradigm

Activation editing modifies model activations rather than weights, drastically reducing trainable parameters (e.g., LoFIT uses just 0.0035%) and performing well even on small datasets.

Activation Editing: A New Paradigm

Activation editing modifies model activations rather than weights, drastically reducing trainable parameters (e.g., LoFIT uses just 0.0035%) and performing well even on small datasets.

- **Intervention Strategy**
  - Given an activation output $z_t^{(l,i)} \in \mathbb{R}^{d_l}$ for i-th component at layer l, we apply the transformation: $z_t^{(l,i)'} = f(z_t^{(l,i)})$
    - **Additive:** $z_t^{(l,i)'} = z_t^{(l,i)} + a^{(l,i)}$
    - **Multiplicative:** $z_t^{(l,i)'} = m^{(l,i)} \odot z_t^{(l,i)}$
    - **Hybrid:** $z_t^{(l,i)'} = m^{(l,i)} \odot z_t^{(l,i)} + a^{(l,i)}$

What Remains Unclear for Activation Editing?

1. Which internal component yields the best intervention outcome?
   - Attention head outputs are the most effective intervention targets.

What Remains Unclear for Activation Editing?

1. Which internal component yields the best intervention outcome?
   - Attention head outputs are the most effective intervention targets.
2. Should we use additive, multiplicative, or hybrid operations for optimal results?
   - Additive bias offsets consistently lead to greater performance improvements than multiplicative scaling.

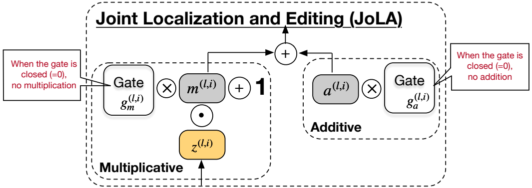
What Remains Unclear for Activation Editing?

1. Which internal component yields the best intervention outcome?
   - Attention head outputs are the most effective intervention targets.
2. Should we use additive, multiplicative, or hybrid operations for optimal results?
   - Additive bias offsets consistently lead to greater performance improvements than multiplicative scaling.
3. Can activation editing perform well in **low-resource settings** (e.g., 200 samples)?
   - Performance is highly sensitive to hyperparameter choices, requiring careful manual tuning for each task.

Please refer to the detailed analysis of the results in the paper (Section 3.1, Appendix C and Appendix F.3).

JoLA Framework: An overview

Our goal: Design a simple and general approach to **dynamically learn where and how** to edit activations in low-resource settings.

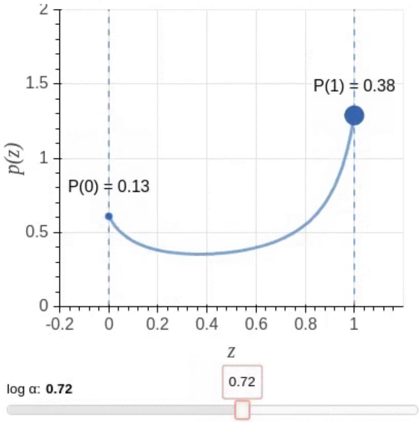


- $z^{(l,i)}$ – original (head / MLP) activation
- $a^{(l,i)}$ – additive modification
- $m^{(l,i)}$ – component-wise multiplicative modification

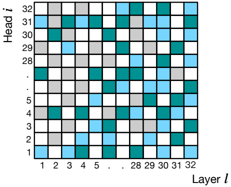
Each gate is just a **random variable during training** (no input!) and becomes a scalar expectation at inference time.

Gating Mechanism: Learn Sparse Edits

- Gates follow a **mixed discrete-continuous distribution**, implemented via the **Hard Concrete** distribution (Louizos et al. 2017).
- The probability that a gate is non-zero acts as a L_0 **regularizer**, encouraging sparsity by controlling the expected number of active (open) gates.



Activation Status



- $g_m^{(l,i)} = 0 \ \& \ g_a^{(l,i)} = 0$
- $g_m^{(l,i)} = 0 \ \& \ g_a^{(l,i)} \neq 0$
- $g_m^{(l,i)} \neq 0 \ \& \ g_a^{(l,i)} = 0$
- $g_m^{(l,i)} \neq 0 \ \& \ g_a^{(l,i)} \neq 0$

- $z^{(l,i)\prime} = z^{(l,i)}$
- $z^{(l,i)\prime} = z^{(l,i)} + g_a^{(l,i)} \cdot a^{(l,i)}$
- $z^{(l,i)\prime} = \left(\mathbf{1} + g_m^{(l,i)} \cdot m^{(l,i)}\right) \odot z^{(l,i)}$
- $z^{(l,i)\prime} = \left(\mathbf{1} + g_m^{(l,i)} \cdot m^{(l,i)}\right) \odot z^{(l,i)} + g_a^{(l,i)} \cdot a^{(l,i)}$

- Given an activation output $z_t^{(l,i)} \in \mathbb{R}^{d_l}$ for i-th head at layer l, the activation can be optimized to four status during training.
  1. original activation (no modification)
  2. add a bias vector (additive modification)
  3. add a scale vector (multiplicative modification)
  4. both scale and bias vector (hybrid modification)

Training Objectives

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

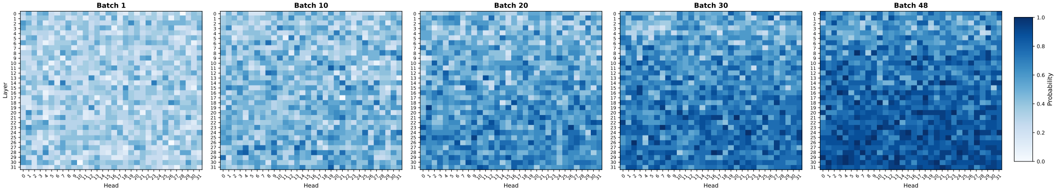
where,

- $L_{xent}(\cdot)$: Standard cross-entropy loss
- $L_C(\phi)$: 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.

Gate Status During Training



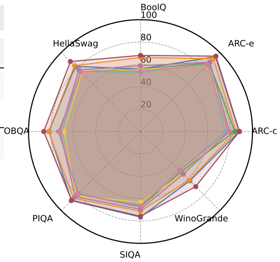
- All gates are initially open. Then, JoLA learns **which components** (e.g., attention heads) to modify and **how** (additively or multiplicatively)
- Interestingly, multiplicative gate (g_m) tends to **close more frequently**

- Baselines
  1. **Zero-shot**:LLaMA-3 and Qwen-2.5
  2. **PEFT method**: LoRA
  3. **Activation editing during training**: BitFit (Ben Zaken et al. 2022), RED (Wu et al. 2024a), ReFT (Wu et al. 2024b), and LoFIT (Yin et al. 2024)
  4. **Activation editing during inference**: RePE (Zou et al. 2023)

- Evaluation Setups
  1. **Low-resource scenario:** 200 training samples
  2. Commonsense Reasoning (8 tasks), Natural Language Understanding (14 tasks) and Natural Language Generation (4 tasks)
  3. Accuracy for reasoning and understanding tasks, BLEU, ROUGE-L and BERTScore for generation tasks

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**



- Activation editing baselines show varying levels of success across tasks, but their performance is often limited by sensitivity to hyperparameters and layer selection.
- **JoLA consistently outperforms all baselines** across all three task types, achieving robust improvements with minimal tuning.

Ablation Study 1: Gating Mechanism

	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**

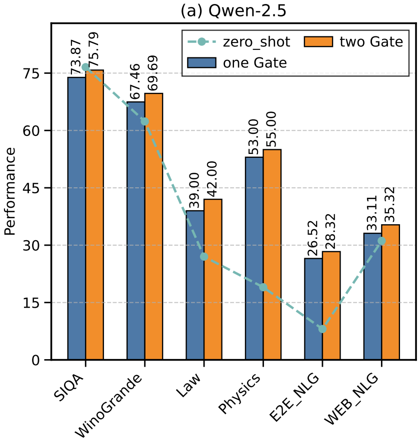
The gating mechanism significantly enhances performance, demonstrating its effectiveness for both attention head and MLP layer interventions.

Ablation Study 1: Gating Mechanism

	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**

The combination of components (attention head and MLP) with a gate mechanism shows improvement but still underperforms compared to a single intervention. The conclusion is in line with previous investigations.

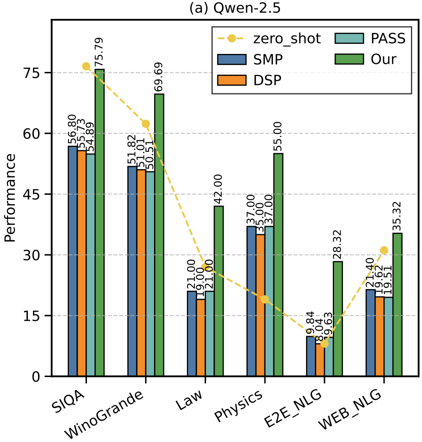
Ablation Study 2: Number of Gates



- **one gate:** $g_m^{(l,i)}$ and $g_a^{(l,i)}$ share the same gate.
- **two gate:** $g_m^{(l,i)}$ and $g_a^{(l,i)}$ are different gates.

Although the shared gating configuration outperforms the zero-shot baseline, it lags behind the configuration with separate gates, highlighting the benefit of fine-grained control.

Ablation Study 3: Different Head Selection Strategies

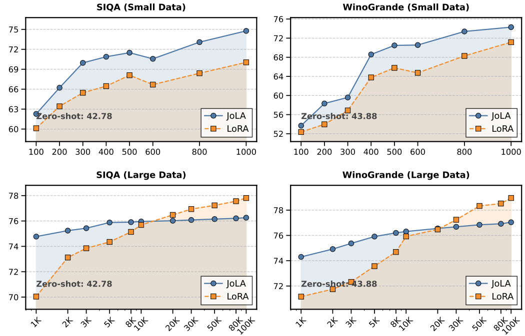


Compared Strategies:

- **SMP** (Zhang et al. 2021): Trains a pruner to rank and drop less important heads.
- **DSP** (Li et al. 2021): Uses Gumbel-Softmax to select top-K heads.
- **PASS** (Ding et al. 2024): Applies robust optimization for deterministic sparsity.

JoLA consistently outperforms other attention head pruning strategies, demonstrating the effectiveness of its learned, task-adaptive selection mechanism.

Different Data Size

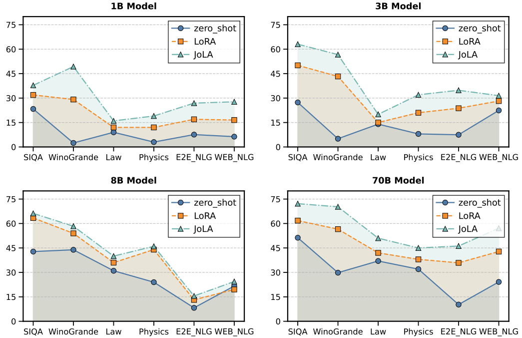


Data Size:

- **small data size:** 100 - 1,000 samples;
- **large data size:** 1,000 - 100,0000 samples;

- JoLA significantly outperforms LoRA on small datasets (even with just 100 samples).
- JoLA remains competitive or slightly better with 5,000–10,000 samples.
- LoRA gains a modest edge as data scales to 20k–100k.

Different Model Size



Model Size:

- LLaMA (1B, 3B, 8B, 70B)

- JOLA consistently delivers significant performance improvements across all model sizes.
- larger models benefit more substantially from JOLA's dynamic selection mechanism

Thank you!
Questions & Comments?

Paper **Code** **Blog**

Contact: wen.lai@tum.de