NEURAL INFORMATION PROCESSING SYSTEMS

Prompt Tuning Strikes Back: Customizing Foundation Models with Low-Rank Prompt Adaptation

Abhinav Jain, Swarat Chaudhuri, Thomas Reps, Chris Jermaine

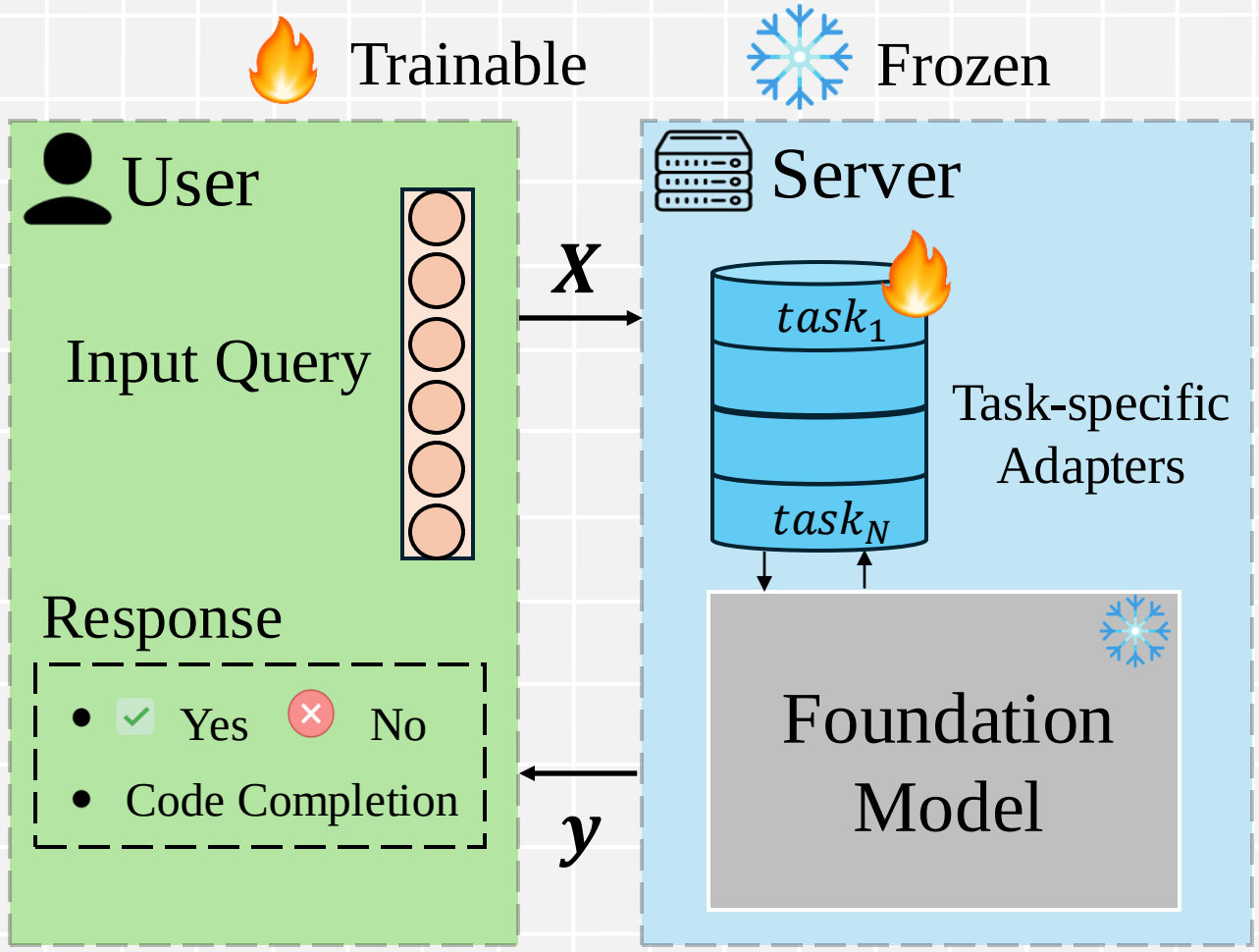
NeurIPS 2024

PEFT – A Brief Overview

Objective: Pick and fine-tune a subset of parameters while being computationally cheaper than SFT but achieving the same performance.

Shortcomings of existing methods -

- Maintain multiple adapter-like modules for each task on the server side
- Select and assemble subset of modules every time a batch is processed during inference

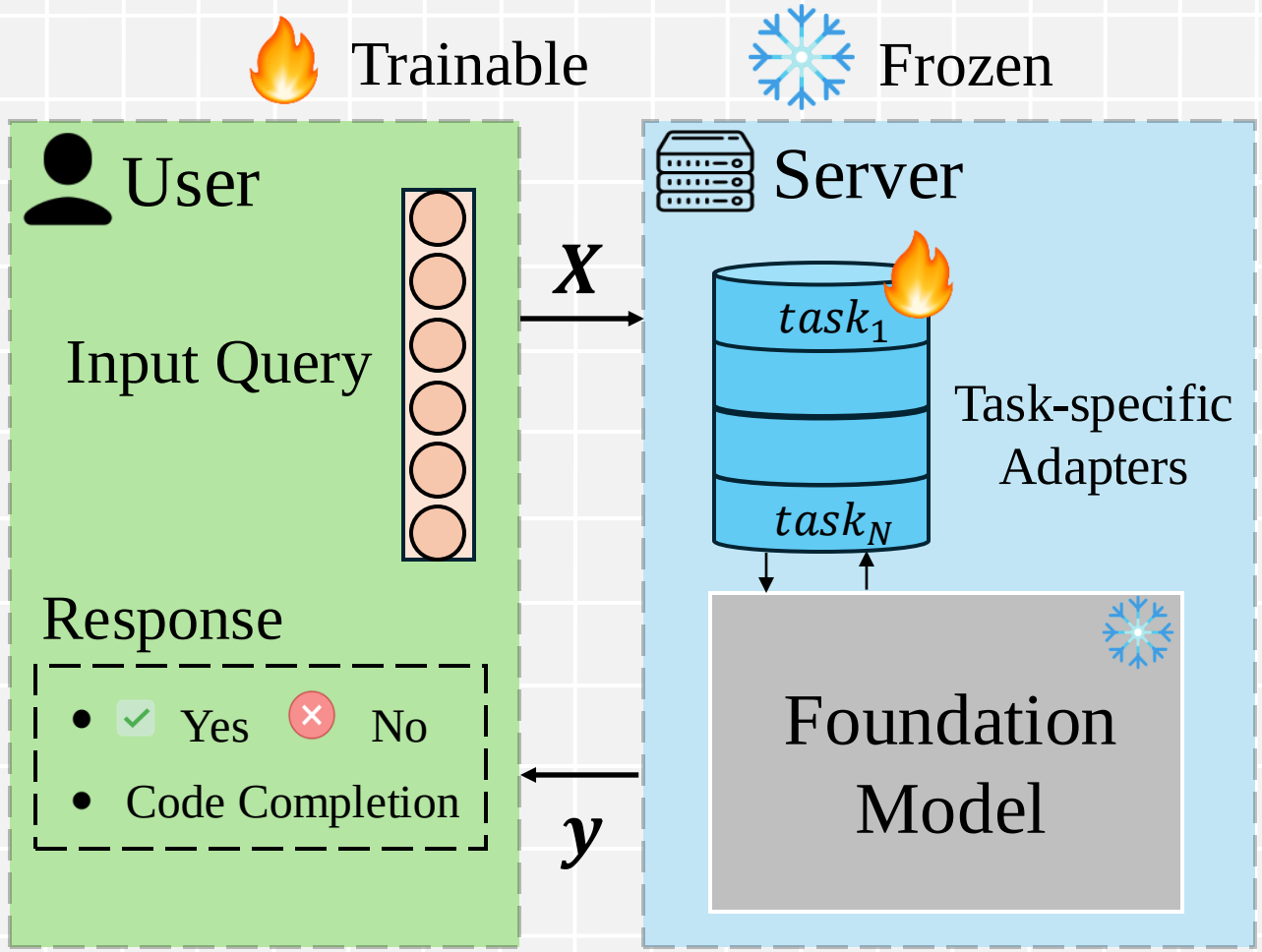


PEFT – A Brief Overview

Prompt-tuning is a promising approach

- Prepends a small set of vectors (soft-prompt) at the input layer for a task
- No server-side task-specific processing

But, it's not as high-performing as other approaches like LoRA!

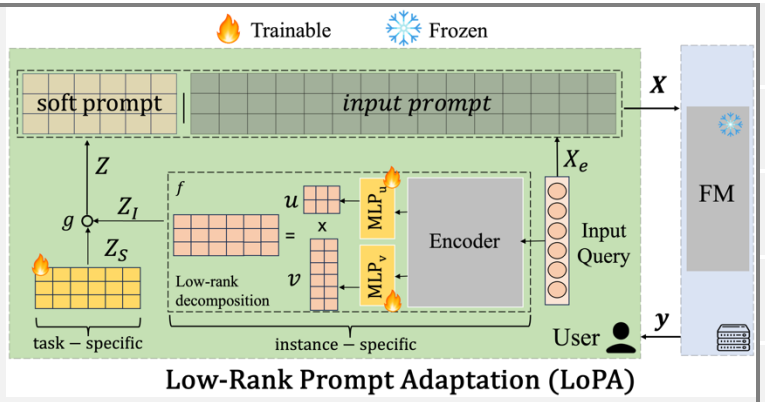


Can we further improve the performance of prompt-tuning while staying parameter-efficient?

LoPA: Low-rank Prompt Adaptation

- Instance-aware prompt tuning-based approach

$$\text{soft-prompt}, \mathrm{Z} = f(x) \;\; \forall x \in D$$



Low-Rank Prompt Adaptation (LoPA)

- Constructs soft prompt using
  - Task-specific component, Z_S : shares task information across samples
  - Instance-specific component, Z_I : incorporates information for each instance
  - Non-linear function combines Z_S and Z_I to get Z
    - Gating function, g activates task-specific information conditioned on instance-specificity

- For parameter efficiency, it approximates the instance-specific component using low-rank decomposition, $Z_I = u \times v$, where u, v are low-rank matrices

Experimental Setup

- Six Natural Language Understanding tasks from the GLUE benchmark
  - SST-2, MNLI, MRPC, QNLI, QQP and RTE
  - Optimise 355M RoBERTa foundation model
- Three Code Understanding tasks
  - Code Completion from MBPP
  - Input/Output Unit test prediction from CruxEval
  - Optimize 350M CodeGen, 1.3B-7B DeepSeek-Coder, 2.7B Phi-2, 3.8B Phi-3 and 8B Llama3
- Suite of baselines with different customization methods
  - FFT
  - LoRA
  - Standard Prompt Tuning
  - S-IDPG, $Z = Z_S + Z_I$

Results on Natural Language

Tuning	Tunable Parameters	SST-2 (acc)	MNLI (acc)	MRPC (acc & F1)	QNLI (acc)	QQP (acc & F1)	RTE (acc)	Avg
FFT	355M	95.99	**90.40**	90.81	94.60	**90.39**	**85.92**	**91.35**
LoRA	2.36M	**96.22**	90.30	90.77	**94.69**	89.91	85.66	91.26
None	0	59.97	39.60	73.52	50.16	42.34	53.43	53.17
PT	10.2K	84.40	54.67	72.38	58.74	48.20	53.07	61.91
S-IDPG	2.89M	95.30	84.50	78.60	90.48	84.88	77.26	85.17
Ours	1.60M	95.99	89.22	**91.09**	93.74	89.72	83.39	90.53

Table 1: Performance on GLUE tasks. We report the average of accuracy and F1 for both MRPC and QQP. For all the other tasks, we report accuracy. Approaches below the dotted line do not require any modification to the model on the server side. **Bold** denotes the best-performing tuning method for the given model. Underline marks the best result among all prompt tuning methods.

- LoPA outperforms prompt tuning by a significant margin of **28.62** points

Src: https://arxiv.org/abs/2405.15282

Results on Natural Language

Tuning	Tunable Parameters	SST-2 (acc)	MNLI (acc)	MRPC (acc & F1)	QNLI (acc)	QQP (acc & F1)	RTE (acc)	Avg
FFT	355M	95.99	**90.40**	90.81	94.60	**90.39**	**85.92**	**91.35**
LoRA	2.36M	**96.22**	90.30	90.77	**94.69**	89.91	85.66	91.26
None	0	59.97	39.60	73.52	50.16	42.34	53.43	53.17
PT	10.2K	84.40	54.67	72.38	58.74	48.20	53.07	61.91
S-IDPG	2.89M	95.30	84.50	78.60	90.48	84.88	77.26	85.17
Ours	1.60M	<u>95.99</u>	<u>89.22</u>	**<u>91.09</u>**	<u>93.74</u>	<u>89.72</u>	<u>83.39</u>	<u>90.53</u>

Table 1: Performance on GLUE tasks. We report the average of accuracy and F1 for both MRPC and QQP. For all the other tasks, we report accuracy. Approaches below the dotted line do not require any modification to the model on the server side. **Bold** denotes the best-performing tuning method for the given model. <u>Underline</u> marks the best result among all prompt tuning methods.

- LoPA outperforms prompt tuning by a significant margin of **28.62** points
- LoPA achieves performance close to FFT and LoRA within **1** point while using fewer parameters

Src: https://arxiv.org/abs/2405.15282

Results on Code

- LoPA consistently improves pass@1 over prompt tuning – with modest improvements on smaller FMs to larger improvements on larger FMs

Model	Tuning	#Params	Code Understanding		Code Generation
			CruxEval-I	CruxEval-O	MBPP
DeepseekCoder-1.3B	FFT	1.3B	**45.0**	34.8	**44.76**
	LoRA	4.7M	35.5	**36.0**	44.14
	None	0	26.8	29.8	34.08
	PT	20.5K	41.2	31.2	34.49
	S-IDPG	16.3M	26.0	28.5	42.50
	LoPA	4.2M	43.0	34.5	44.66
DeepseekCoder-7B	FFT	7B	*OOM*	*OOM*	*OOM*
	LoRA	11.8M	47.5	**49.8**	53.38
	None	0	39.3	44.0	50.51
	PT	41.0K	45.8	44.8	37.47
	S-IDPG	32.1M	40.5	41.5	**53.59**
	LoPA	6.35M	**50.0**	48.0	52.46

Table 2: Performance comparison on CruxEval and MBPP tasks. We report average *pass*@1 scores. Approaches below the dotted line are prompt-tuning methods, which do not require any modification to the model on the server side. **Bold** denotes the best-performing tuning method for the given model. Underline marks the best result among all prompt-tuning methods. *OOM* indicates that the corresponding tuning approach exceeded the available GPU memory and ran out of memory.

Src: https://arxiv.org/abs/2405.15282

Results on Code

- LoPA consistently improves pass@1 over prompt tuning – with modest improvements on smaller FMs to larger improvements on larger FMs
- IDPG performs worse than PT in CruxEval tasks – simply encoding an instance-specific prompt does not guarantee improvements

Model	Tuning	#Params	Code Understanding		Code Generation
			CruxEval-I	CruxEval-O	MBPP
DeepseekCoder-1.3B	FFT	1.3B	**45.0**	34.8	**44.76**
	LoRA	4.7M	35.5	**36.0**	44.14
	None	0	26.8	29.8	34.08
	PT	20.5K	41.2	31.2	34.49
	S-IDPG	16.3M	26.0	28.5	42.50
	LoPA	4.2M	43.0	34.5	44.66
DeepseekCoder-7B	FFT	7B	*OOM*	*OOM*	*OOM*
	LoRA	11.8M	47.5	**49.8**	53.38
	None	0	39.3	44.0	50.51
	PT	41.0K	45.8	44.8	37.47
	S-IDPG	32.1M	40.5	41.5	**53.59**
	LoPA	6.35M	**50.0**	48.0	52.46

Table 2: Performance comparison on CruxEval and MBPP tasks. We report average *pass*@1 scores. Approaches below the dotted line are prompt-tuning methods, which do not require any modification to the model on the server side. **Bold** denotes the best-performing tuning method for the given model. Underline marks the best result among all prompt-tuning methods. *OOM* indicates that the corresponding tuning approach exceeded the available GPU memory and ran out of memory.

Src: https://arxiv.org/abs/2405.15282

Results on Code

- LoPA consistently improves pass@1 over prompt tuning – with modest improvements on smaller FMs to larger improvements on larger FMs
- IDPG performs worse than PT in CruxEval tasks – simply encoding an instance-specific prompt does not guarantee improvements
- LoPA performs on par with LoRA while using fewer trainable params

Model	Tuning	#Params	Code Understanding		Code Generation
			CruxEval-I	CruxEval-O	MBPP
DeepseekCoder-1.3B	FFT	1.3B	**45.0**	34.8	**44.76**
	LoRA	4.7M	35.5	**36.0**	44.14
	None	0	26.8	29.8	34.08
	PT	20.5K	41.2	31.2	34.49
	S-IDPG	16.3M	26.0	28.5	42.50
	LoPA	4.2M	43.0	34.5	44.66
DeepseekCoder-7B	FFT	7B	*OOM*	*OOM*	*OOM*
	LoRA	11.8M	47.5	**49.8**	53.38
	None	0	39.3	44.0	50.51
	PT	41.0K	45.8	44.8	37.47
	S-IDPG	32.1M	40.5	41.5	**53.59**
	LoPA	6.35M	**50.0**	48.0	52.46

Table 2: Performance comparison on CruxEval and MBPP tasks. We report average *pass*@1 scores. Approaches below the dotted line are prompt-tuning methods, which do not require any modification to the model on the server side. **Bold** denotes the best-performing tuning method for the given model. Underline marks the best result among all prompt-tuning methods. *OOM* indicates that the corresponding tuning approach exceeded the available GPU memory and ran out of memory.

Src: https://arxiv.org/abs/2405.15282

Results on Code

Conclusions

- Averaged across all tasks and models, LoPA showed relative improvements of **28.52%** and **20.16%** over PT and IDPG.
- LoPA outperformed LoRA in **11/24** cases while being in the **0.5%** range in remainder of tasks

Tuning	Tunable Parameters	SST-2 (acc)	MNLI (acc)	MRPC (acc & F1)	QNLI (acc)	QQP (acc & F1)	RTE (acc)	Avg
FFT	355M	95.99	**90.40**	90.81	94.60	**90.39**	**85.92**	**91.35**
LoRA	2.36M	**96.22**	90.30	90.77	**94.69**	89.91	85.66	91.26
None	0	59.97	39.60	73.52	50.16	42.34	53.43	53.17
PT	10.2K	84.40	54.67	72.38	58.74	48.20	53.07	61.91
S-IDPG	2.89M	95.30	84.50	78.60	90.48	84.88	77.26	85.17
Ours	1.60M	95.99	89.22	**91.09**	93.74	89.72	83.39	90.53

Table 1: Performance on GLUE tasks. We report the average of accuracy and F1 for both MRPC and QQP. For all the other tasks, we report accuracy. Approaches below the dotted line do not require any modification to the model on the server side. **Bold** denotes the best-performing tuning method for the given model. Underline marks the best result among all prompt tuning methods.

Model	Tuning	#Params	Code Understanding		Code Generation
			CruxEval-I	CruxEval-O	MBPP
DeepseekCoder-1.3B	FFT	1.3B	**45.0**	34.8	**44.76**
	LoRA	4.7M	35.5	**36.0**	44.14
	None	0	26.8	29.8	34.08
	PT	20.5K	41.2	31.2	34.49
	S-IDPG	16.3M	26.0	28.5	42.50
	LoPA	4.2M	43.0	34.5	44.66
DeepseekCoder-7B	FFT	7B	*OOM*	*OOM*	*OOM*
	LoRA	11.8M	47.5	**49.8**	53.38
	None	0	39.3	44.0	50.51
	PT	41.0K	45.8	44.8	37.47
	S-IDPG	32.1M	40.5	41.5	**53.59**
	LoPA	6.35M	**50.0**	48.0	52.46

Table 2: Performance comparison on CruxEval and MBPP tasks. We report average *pass*@1 scores. Approaches below the dotted line are prompt-tuning methods, which do not require any modification to the model on the server side. **Bold** denotes the best-performing tuning method for the given model. Underline marks the best result among all prompt-tuning methods. *OOM* indicates that the corresponding tuning approach exceeded the available GPU memory and ran out of memory.

Src: https://arxiv.org/abs/2405.15282

Performance as a function of soft-prompt length

- PT and IDPG see performance improvements initially but then it either plateaus or drops.
- LoPA does not exhibit performance fluctuations. Non-linear composition of Z prevents over-fitting.
- LoPA with smaller m outperforms PT and IDPG with larger m. Dimension of the offset subspace is much smaller which LoPA can learn effectively.

Src: https://arxiv.org/abs/2405.15282

Performance as a function of function encoding Z

$Z =$	SST-2 (acc)	MNLI (acc)	MRPC (acc, F1)	QNLI (acc)	QQP (acc, F1)	RTE (acc)	Avg
$\text{concat}(Z_S, Z_I)$	94.50	78.45	76.00	91.43	76.11	84.12	83.44
$\max(Z_S, Z_I)$	**95.99**	**89.37**	88.62	**93.74**	78.44	**85.92**	88.68
$Z_S \circ g(Z_I)$	**95.99**	89.22	**91.09**	**93.74**	**89.72**	83.39	**90.53**

Table 3: Performance on GLUE tasks. We report the average of accuracy and F1 for both MRPC and QQP. For all the other tasks, we report accuracy. **Bold** denotes the best-performing tuning method for the given model.

- Simply concatenating $\boldsymbol{Z_I}$ and $\boldsymbol{Z_S}$ is not sufficient. It can even perform worse than IDPG which combines them using a linear function.
- Non-linear functions capture complex relationships between $\boldsymbol{Z_I}$ and $\boldsymbol{Z_S}$ and exhibit better performance overall.

Src: https://arxiv.org/abs/2405.15282

Conclusions and Future Work

- Designed LoPA to optimize foundation models which is an instance-specific soft-prompting PEFT method
- Used a low-rank approximation of instance-specific soft prompt to enable parameter-efficiency
- Outperforms existing soft-prompting baselines and performs on par with LoRA and FFT on many tasks

- Explore LoPA's effectiveness on real-life obscure tasks where newer attention patterns must be learned
- Consider LoPA as a Conditional Auto-Encoder compressing knowledge from different instances and providing it as additional information to the Foundation Model

Src: https://arxiv.org/abs/2405.15282