中国科学院大学
University of Chinese Academy of Sciences

中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

Scattered or Connected? An Optimized Parameter-efficient Tuning Approach for Information Retrieval

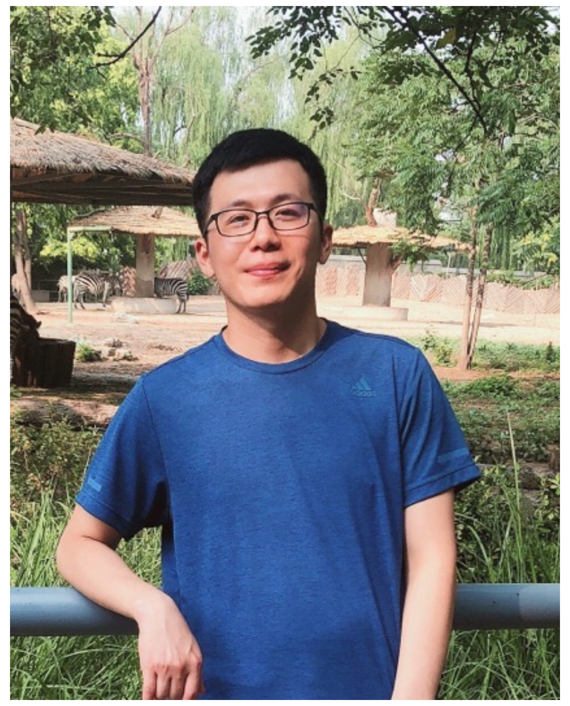
Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, and Xueqi Cheng

https://arxiv.org/pdf/2208.09847.pdf

1. CAS Key Lab of Network Data Science and Technology, Institute of Computing Technology, Chinese Academy of Sciences

2. University of Chinese Academy of Sciences

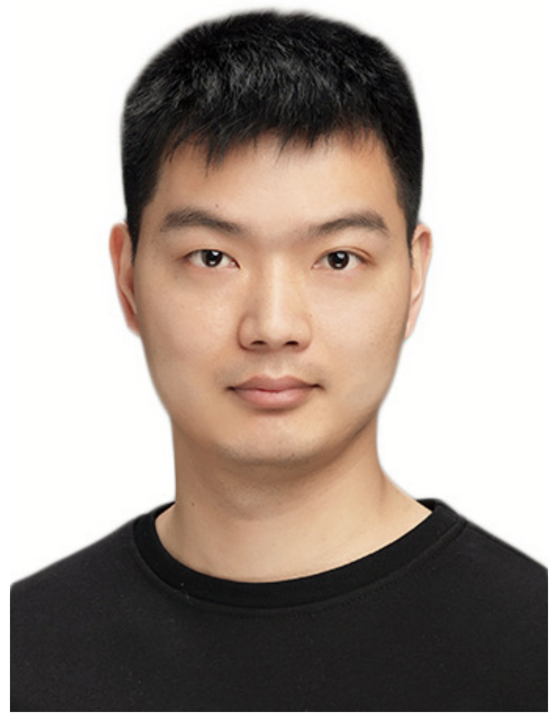
Authors



Xinyu Ma
Ph.D. student
University of the Chinese Academy of Sciences

Jiafeng Guo
Professor
Chinese Academy of Sciences

Ruqing Zhang
Assistant Professor
Chinese Academy of Sciences



Yixing Fan
Associate Professor
Chinese Academy of Sciences

Xueqi Cheng
Professor
Chinese Academy of Sciences

Full Fine-tuning

- Fine-tune all the parameters of pre-trained models(PTM) on downstream tasks

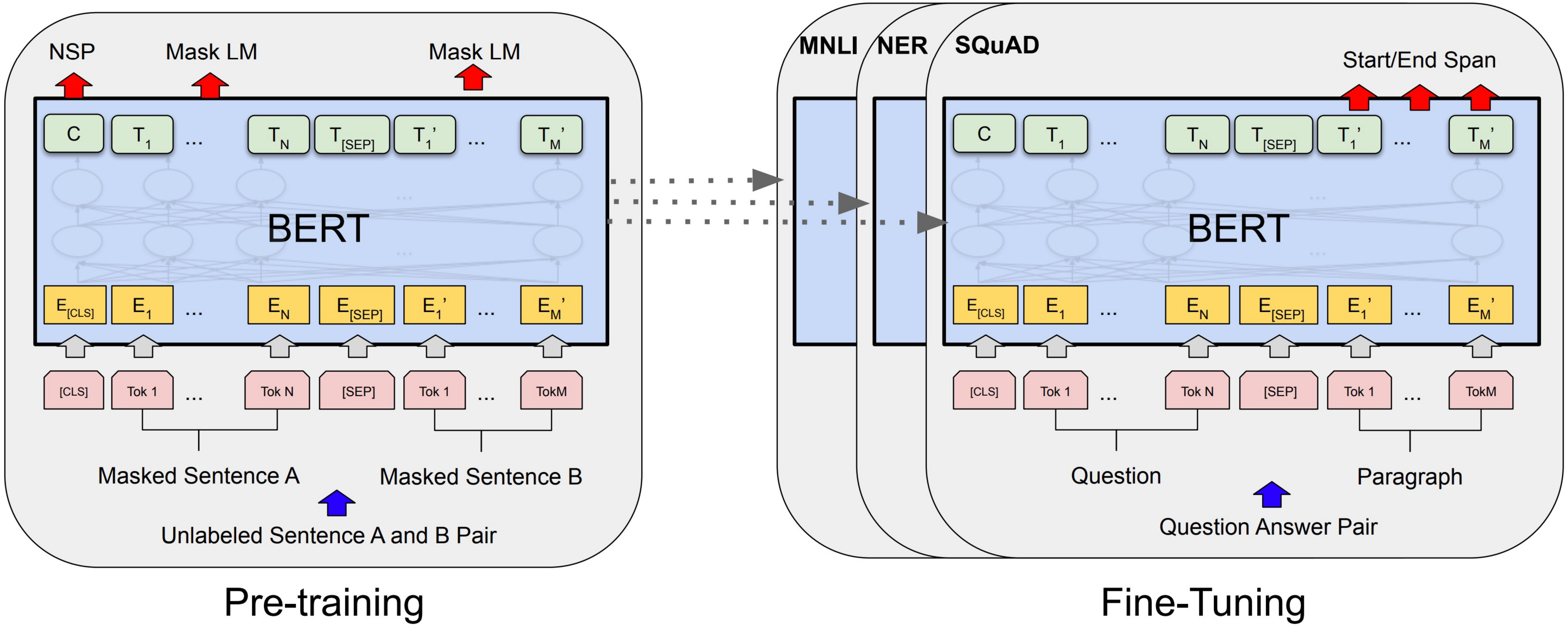
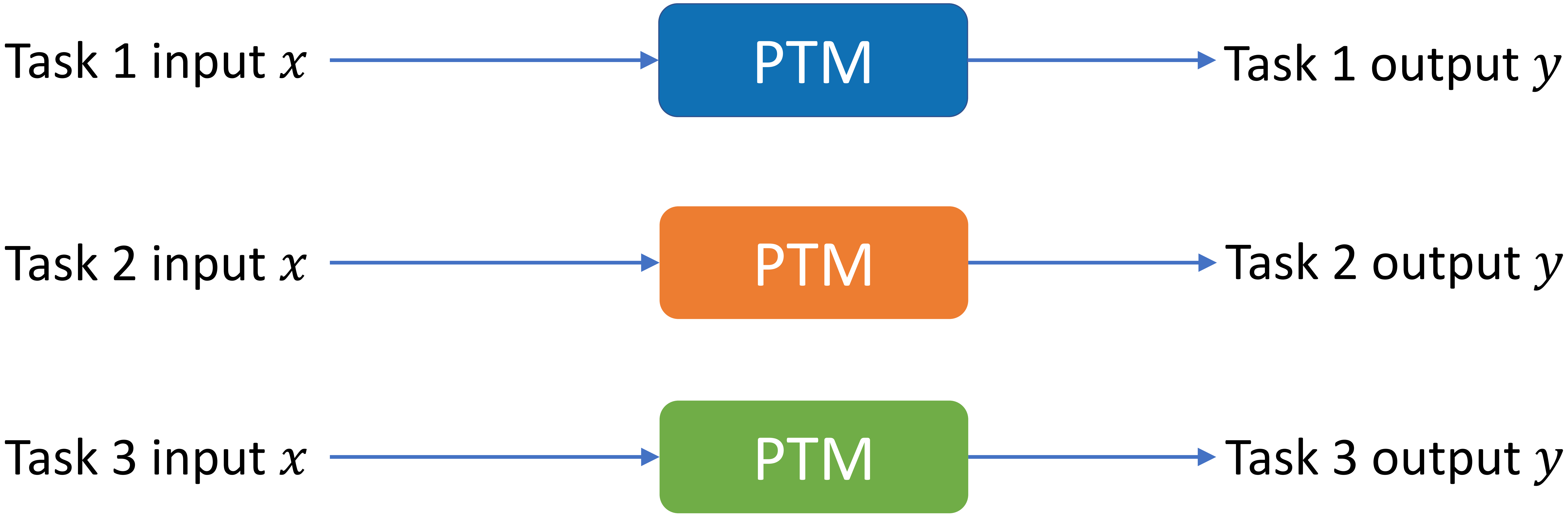


Figure from Devlin et.al. 2019

Full Fine-tuning

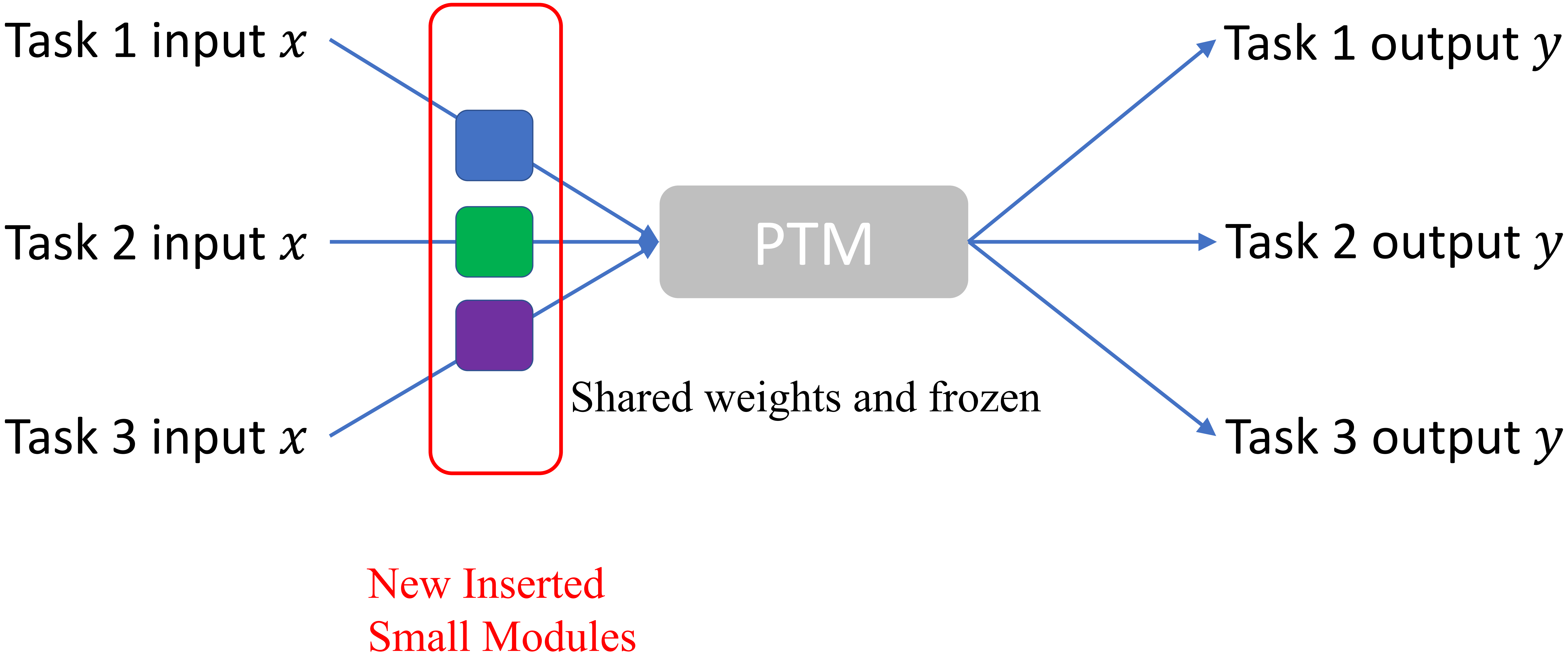
- Each task needs a separate copy of fine-tuned model parameters



- Less feasible and prohibitively expensive as the model size and the number of tasks increase greatly

Parameter-efficient Tuning

- Only fine-tune a small number of parameters



Parameter-efficient Tuning

- Representative methods: Adapter (Hously et.al), Prefix-tuning (Liang et.al.), Lora (Hu et.al), MAM-Adapter (He et.al)

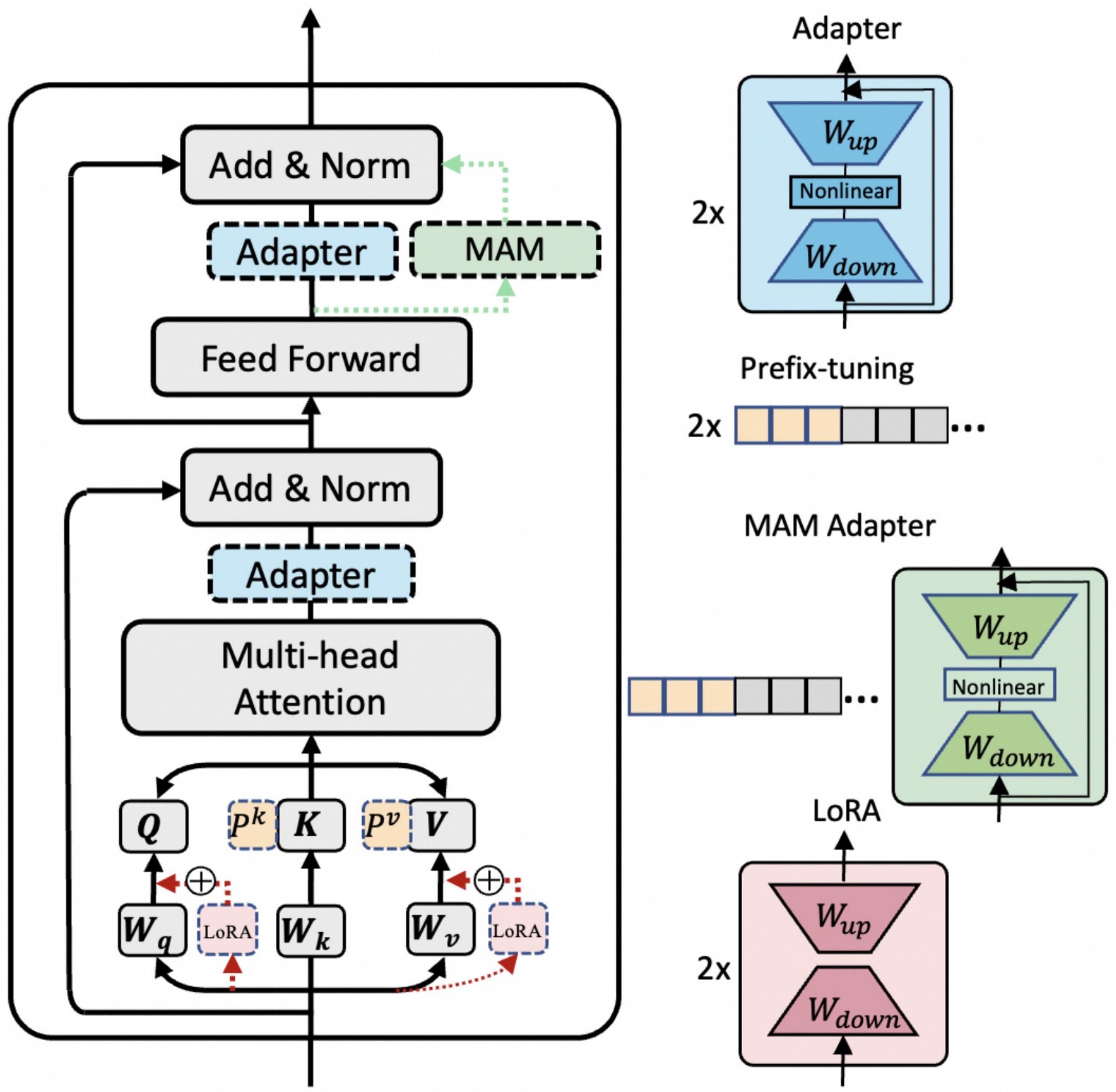


Figure 1: Illustration of a Transformer layer and several representative parameter-efficient tuning methods. Note that MAM Adapter uses a parallel adapter on FFN sub-layer and prefix-tuning on self-attention sub-layer.

(1) Contribution: A Comprehensive Study on IR Tasks

- Can existing methods perform as well in IR as in NLP?

Method	#Params	MARCO Passage		TREC2019 Passage		MARCO Doc		TREC2019 Doc	
		MRR@10	R@1000	nDCG@10	R@100	MRR@100	R@100	nDCG@10	R@100
Full fine-tuning	100%	**0.316**	**0.949**	**0.600**	**0.715**	**0.312**	**0.801**	**0.462**	**0.409**
Bitfit	0.09%	0.262	0.921	0.562	0.677	0.264	0.785	0.437	0.345
Prefix-tuning	0.5% (l=32)	0.294	0.939	0.596	0.692	0.266	0.782	0.423	0.326
Adapter	0.5% (r=16)	0.304	0.941	**0.606**	0.696	0.255	0.770	0.418	0.370
MAM Adapter	0.5% (r=16,l=16)	0.304	0.944	**0.609**	0.712	0.280	0.799	0.458	0.381
LoRA	0.5% (r=16)	0.302	0.943	**0.608**	0.707	0.271	0.794	0.417	0.376
Prefix-tuning	3.6% (l=200)	0.304	0.943	0.580	0.702	0.265	0.775	0.395	0.376
Adapter	6.7% (r=200)	0.316	0.946	0.587	0.687	0.270	0.785	0.433	0.400
MAM Adapter	6.7% (r=200,l=200)	0.314	0.947	**0.616**	**0.720**	0.283	0.792	0.438	0.402
LoRA	6.7% (r=200)	0.316	0.946	0.597	0.715	0.279	0.794	0.417	0.379

Table 1: Dense Retrieval

- Unlike the promising results in NLP, all representative methods cannot achieve a comparable performance over full fine-tuning with less than 1% of the model parameters on all datasets

(1) Contribution: A Comprehensive Study on IR Tasks

- Can existing methods perform as well in IR as in NLP?

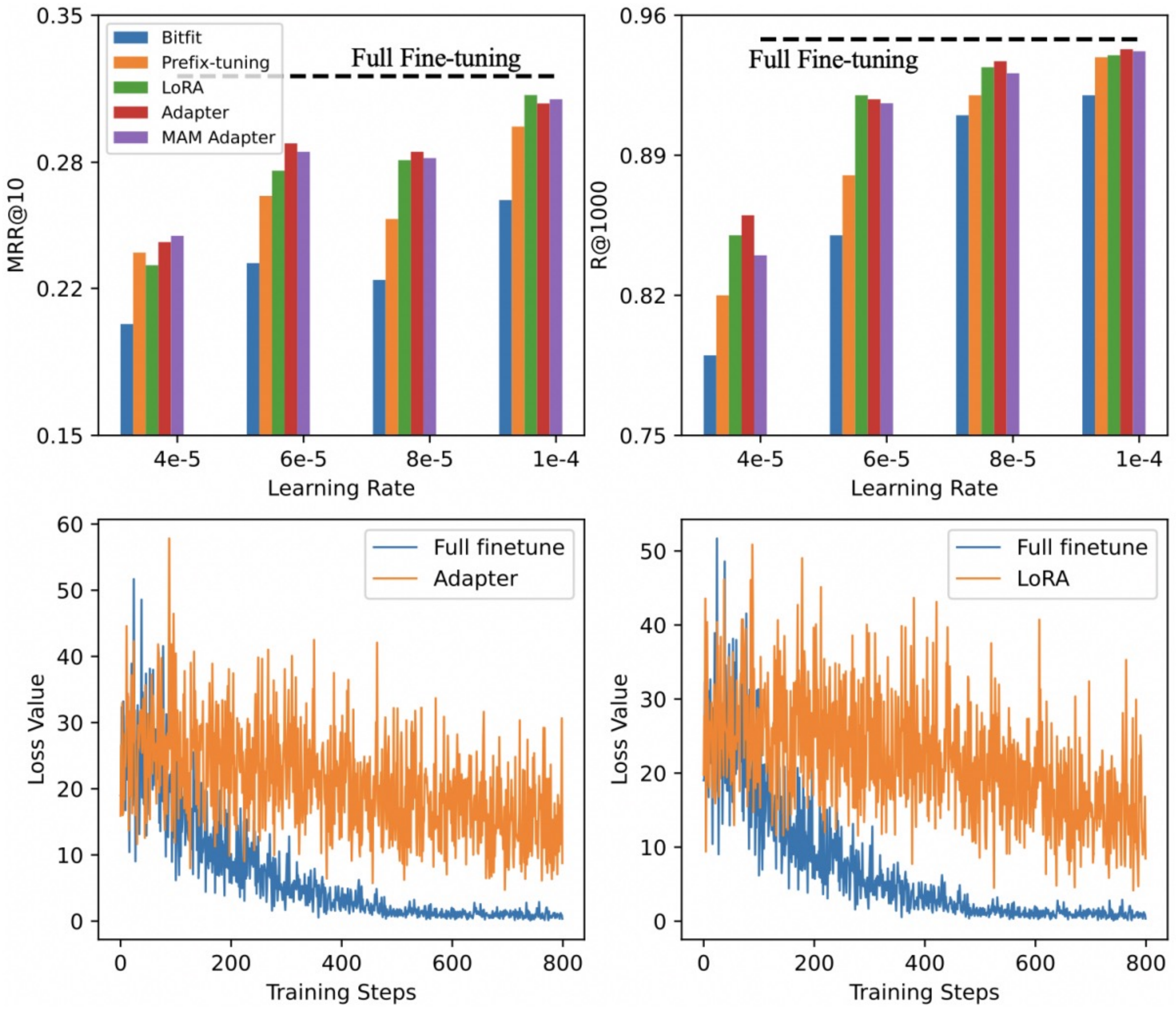
Method	#Params	MARCO Passage		TREC2019 Passage		MARCO Doc		TREC2019 Doc	
		MRR@10	MRR@100	nDCG@10	nDCG100	MRR@10	MRR@100	nDCG@10	nDCG@100
Full fine-tuning	100%	**0.376**	**0.383**	**0.738**	**0.637**	**0.404**	**0.408**	**0.657**	**0.536**
Bitfit	0.09%	0.325	0.334	0.562	0.483	0.364	0.357	0.630	0.531
Prefix-tuning	0.5% (l=32)	0.355	0.363	0.705	0.626	0.387	0.381	0.640	0.530
Adapter	0.5% (r=16)	0.366	0.371	0.714	0.626	0.397	0.392	0.653	0.534
MAM Adapter	0.5% (r=16,l=16)	0.365	0.373	0.717	0.629	0.390	0.395	0.632	0.531
LoRA	0.5% (r=16)	0.363	0.372	0.720	0.635	0.386	0.392	0.637	0.529
Prefix-tuning	3.6% (l=200)	0.363	0.371	0.722	0.632	0.384	0.389	0.640	0.532
Adapter	6.7% (r=200)	0.373	0.381	0.735	0.637	0.402	0.407	0.631	0.528
MAM Adapter	6.7% (r=200,l=200)	0.369	0.380	0.731	0.633	0.397	0.402	0.630	0.528
LoRA	6.7% (r=200)	0.370	0.378	0.730	0.631	0.401	0.396	0.647	0.530

Table 2: Re-ranking

- Unlike the promising results in NLP, all representative methods cannot achieve a comparable performance over full fine-tuning with less than 1% of the model parameters on all datasets

Observation

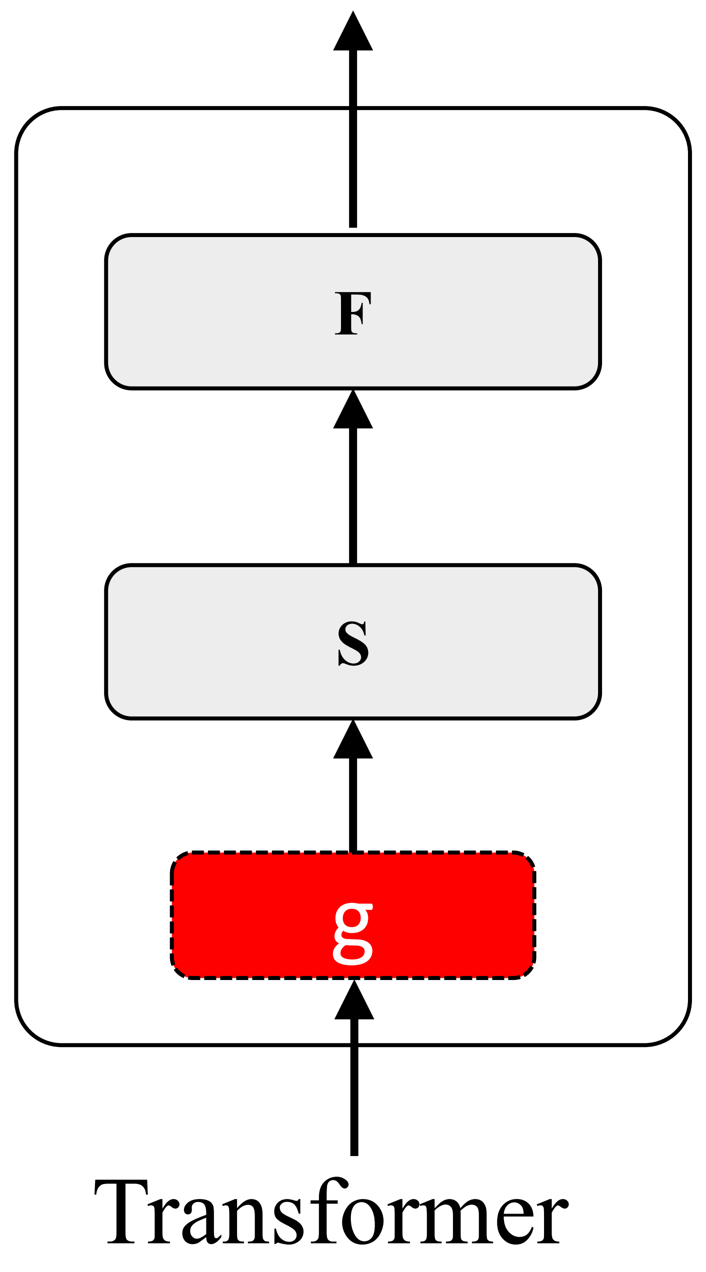
- Parameter-efficient but not learning-efficient



- Sensitive to learning rate and unstable training leading to slow convergence

(2) Contribution: A Theoretical Analysis

- Why the standard setup of parameter-efficient tuning methods falls short in IR?

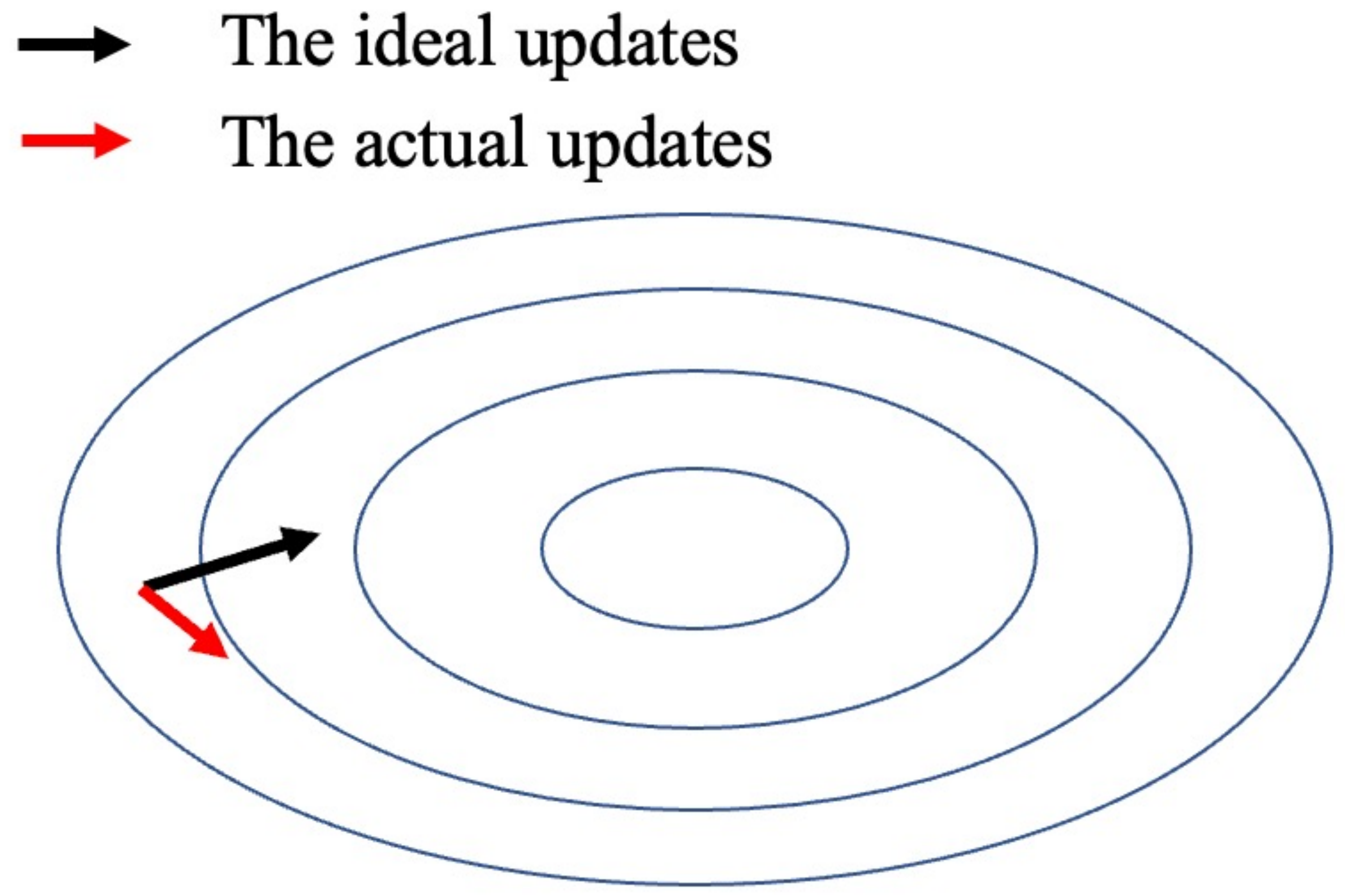


Different components contribute different to the final optimization direction

$$\triangle\,(ideal) = \underline{\triangle\,F, \triangle\,S, \triangle, g}$$

VS.

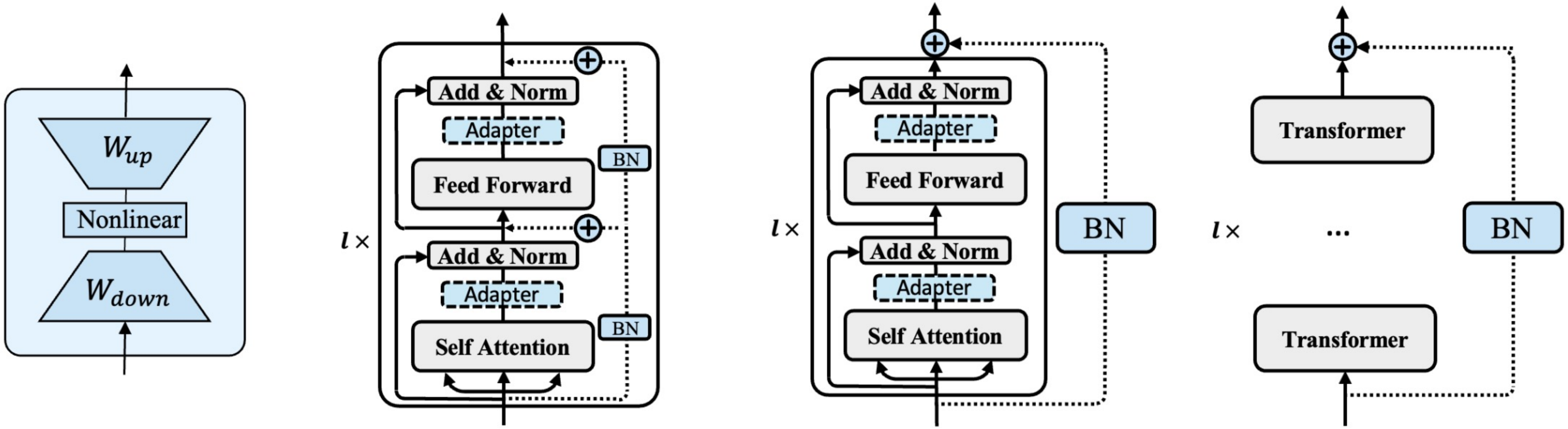
$$\triangle\,(actual) = \triangle\,g$$



- A discrepancy between the ideal optimization direction and the actual update direction

(3) Contribution: Propose a New Method

- Can we design a parameter-efficient tuning approach to stabilize the training process for IR?



(a) The aside module: BN (b) IAA-S: outside the sub-layer (c) IAA-L: outside the layer (d) IAA-M: outside the model

- Insert extra modules in an aside manner beyond inside manner using the idea of ResNet

Main Results

Table 4: Comparisons between IAA and the baselines at the retrieval stage. Two-tailed t-tests demonstrate the improvements of IAA over baselines are statistically significant ($p \leq 0.05$). $*$ indicate significant improvements over full fine-tuning. † indicate significant improvements over best parameter-efficient tuning methods (PET) at the same setting.

Method	#Params	MARCO Passage		TREC2019 Passage		MARCO Doc		TREC2019 Doc	
		MRR@10	R@1000	nDCG@10	R@100	MRR@100	R@100	nDCG@10	R@100
Full fine-tuning	100%	0.316	0.949	0.600	0.715	**0.312**	**0.801**	**0.462**	**0.409**
Best PET	0.5%	0.304	0.944	0.609	0.712	0.280	0.799	0.458	0.381
IAA-S Adapter	0.5% (r=8,ar=8)	0.312†	0.941	0.605	0.719	0.285	0.785	0.454	0.384
IAA-L Adapter	0.5% (r=12,ar=12)	0.314†	0.943	0.615†	0.735*	0.292	0.792	0.446	0.391
IAA-M Adapter	0.5% (r=15,ar=24)	0.309	0.941	0.602	0.721	0.287	0.782	0.449	0.385
Best PET	6.7%	0.316	0.946	0.616	0.720	0.283	0.792	0.438	0.402
IAA-S Adapter	6.7% (r=100,ar=100)	0.324	0.947	0.581	0.719	0.290	0.798	0.441	0.398
IAA-L Adapter	6.7% (r=50,ar=300)	**0.327**†*	**0.951**	**0.617***	**0.735**†	0.295†	0.795	0.439	0.395
IAA-M Adapter	6.7% (r=185,ar=960)	0.321	0.948	0.592	0.710	0.285	0.793	0.437	0.402

- Our best IAA model with tuning less than 1% of the model parameters achieve a comparable performance over full fine-tuning, and is significantly better than the best PET at the retrieval stage.

Main Results

Table 5: Comparisons between IAA and the baselines on the re-ranking stage. Two-tailed t-tests demonstrate the improvements of IAA over baselines are statistically significant ($p \leq 0.05$). $*$ indicate significant improvements over full fine-tuning. † indicate significant improvements over best parameter-efficient tuning methods (PET) at the same setting.

Method	#Params	MARCO Passage		TREC2019 Passage		MARCO Doc		TREC2019 Doc	
		MRR@10	MRR@100	nDCG@10	nDCG100	MRR@10	MRR@100	nDCG@10	nDCG@100
Full fine-tuning	100%	0.376	0.383	0.738	0.637	0.404	0.408	0.657	0.536
Best PET	0.5%	0.366	0.371	0.720	0.635	0.397	0.392	0.653	0.534
IAA-S Adapter	0.5% (r=8,ar=8)	0.371	0.377	0.731†	0.632	0.395	0.393	0.655	0.533
IAA-L Adapter	0.5% (r=12,ar=12)	0.373†	0.379†	0.732†	0.633	0.399	0.403†	0.656	0.537
IAA-M Adapter	0.5% (r=15,ar=24)	0.369	0.373	0.725	0.630	0.393	0.391	0.652	0.531
Best PET	6.7%	0.373	0.381	0.735	0.637	0.402	0.407	0.647	0.530
IAA-S Adapter	6.7% (r=100,ar=100)	0.382†	0.385	**0.742**	0.635	0.408	0.412	0.651	0.535
IAA-L Adapter	6.7% (r=50,ar=300)	**0.385***†	**0.392***†	0.740	**0.639**	**0.412**†	**0.414**	**0.657**†	**0.538**
IAA-M Adapter	6.7% (r=185,ar=960)	0.379	0.384	0.739	0.636	0.404	0.410	0.649	0.529

- Our best IAA model with tuning less than 1% of the model parameters achieve a comparable performance over full fine-tuning, and is significantly better than the best PET at the Re-ranking stage.

Convergence Analysis

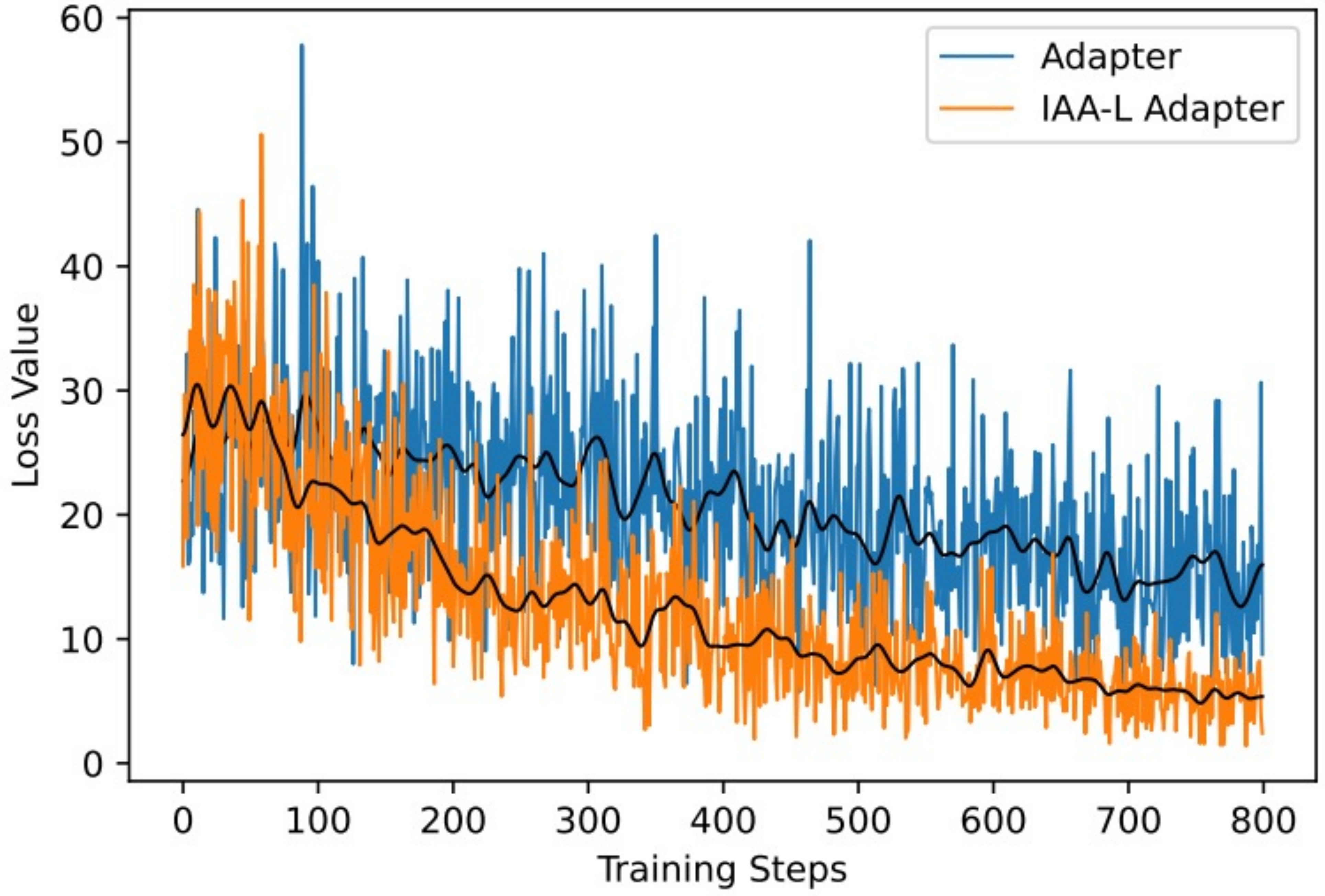


Figure 5: The loss value over training steps.

- IAA-L Adapter has a lower loss value than Adapter and also converges faster than Adapter

Conclusions

(1) **A comprehensive empirical studies** of parameter-efficient tuning methods in IR scenarios, at both the retrieval stage and the re-ranking stage.

(2) We find that these methods are **not learning-efficient** and give a mathematical analysis.

(3) Based on the above, we thus introduce **the aside module** to help to stabilize the optimization process.

Thanks！

Xinyu Ma

maxinyu17g@ict.ac.cn