Parameter-Efficient Fine-Tuning (PEFT)

李昌群

10-18

背景

355M　11B　175B

预训练模型规模越来越大：RoBERTa、GPT、BART、T5；FLAN、PaLM、ChatGPT…

范式

[Pre-train; Fine-tuning] → **[Pre-train; PEFT]** → **[Pre-train; Instruction-tuning]**

问题挑战

1. **显存开销：** 训练时所微调的参数量
2. **存储开销：** 保存参数占用的存储空间

Parameter-Efficient Fine-Tuning

1. Additive Method

Idea

adding trainable additional parameters to PLMs

2. Selective Method

Idea

selecting and updating the model based on its original parameters

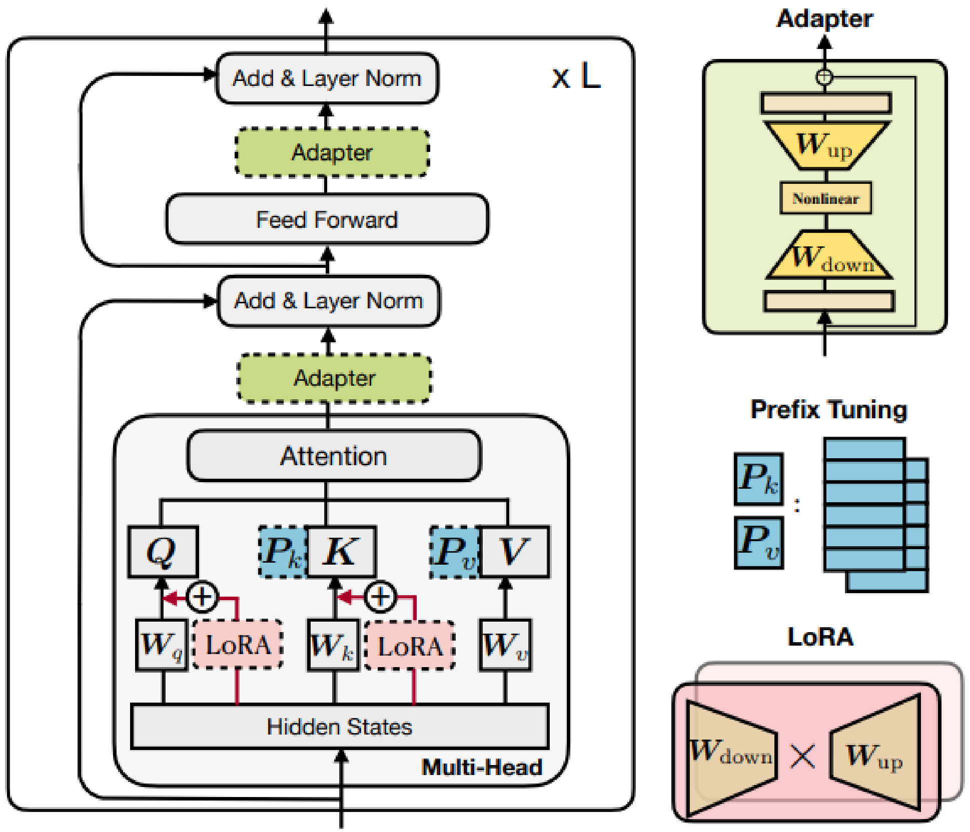
3. Hybrid Method

Idea

mining the effectiveness of different methods and designing unified frameworks to combine them

Existing Methods-- Additive Method

- Idea: adding trainable additional parameters to PLMs



Adapter:

$$\text{Adapter}(\mathbf{x}) = \mathbf{W}_u(\text{ReLU}(\mathbf{W}_d\mathbf{x} + \mathbf{b}_d)) + \mathbf{b}_u$$

Prefix/Prompt-tuning:

$$K_l' = [P_{l,K}; K_l]\ , V_l' = [P_{l,V}; V_l]$$

LoRA:

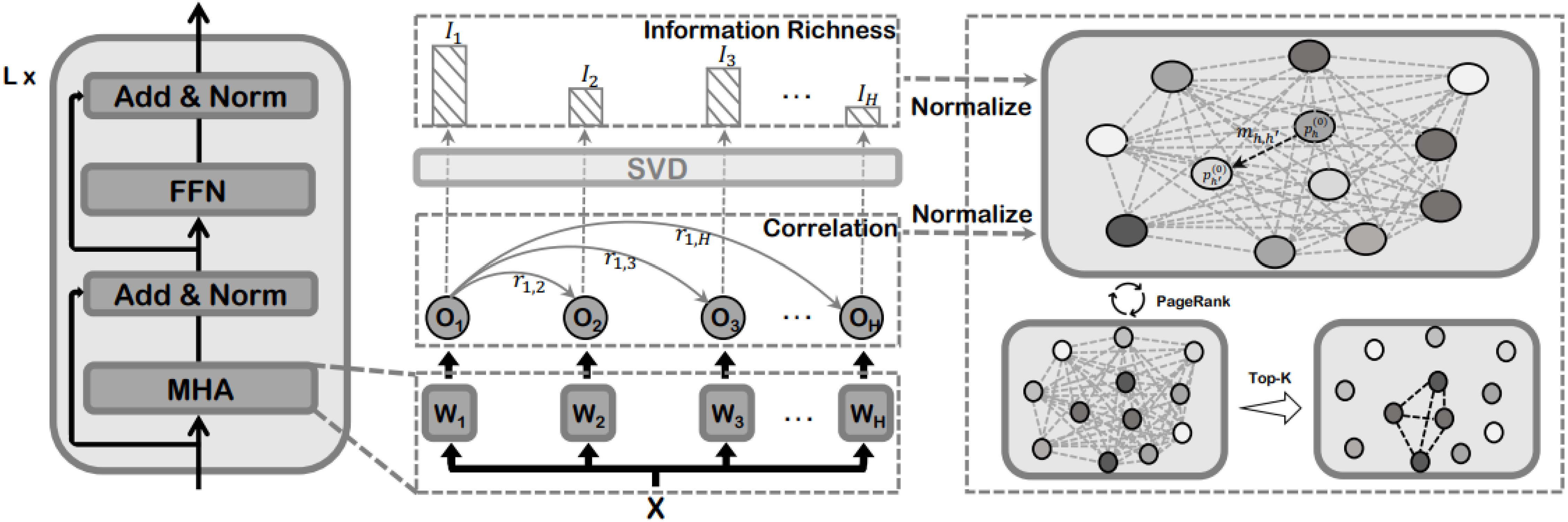
$$W = W^{(0)} + \Delta = W^{(0)} + BA,$$

通过学习低秩矩阵来近似W的参数更新

Existing Methods-- Selective Methods

- Idea: selecting and updating the model based on its original parameters
  - 只对模型的几个top layers进行微调
  - Bitfit: 仅需要对模型的bias项进行微调
  - HiFi: only high-information attention heads are fine-tuned

Selective Methods--HiFi

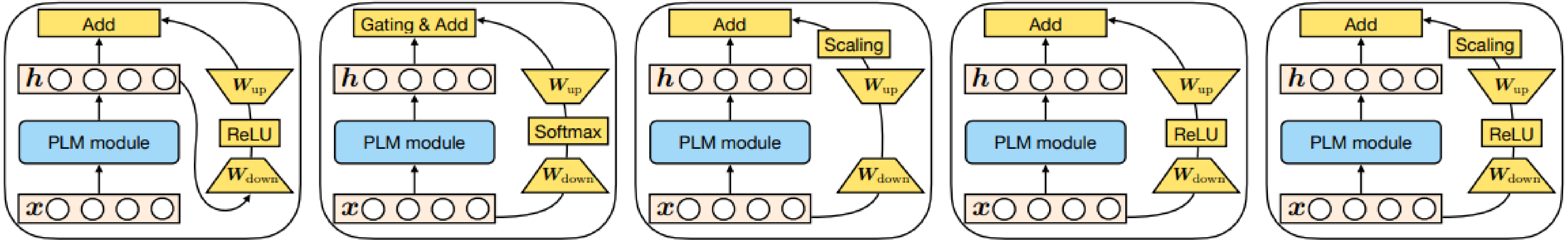


An overview of HiFi

HiFi: High-Information Attention Heads Hold for Parameter-Efficient Model Adaptation ACL2023

Existing Methods-- Hybrid Method

- Idea: mining the effectiveness of different methods and designing unified frameworks to combine them



(a) Adapter (b) Prefix Tuning (c) LoRA (d) Parallel Adapter (e) Scaled PA

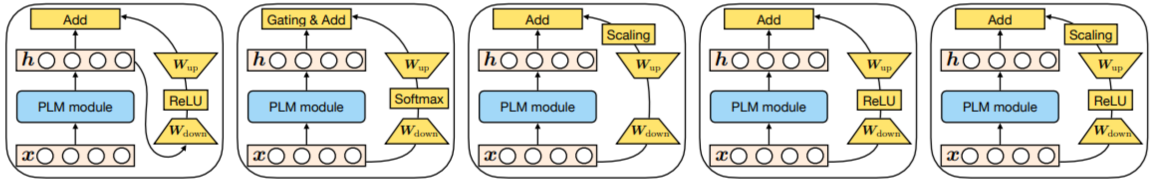
Graphical illustration of existing methods and the proposed variants

从四个维度去对比不同设计

1. Functional Form (adapters, prefix tuning, and LoRA),
2. Modified Representation (attention, FFN),
3. Insertion Form (sequential, parallel),
4. Composition Function (add, gated additive)

Towards a Unified View of Parameter-Efficient Transfer Learning ICLR2022

A unified framework



(a) Adapter (b) Prefix Tuning (c) LoRA (d) Parallel Adapter (e) Scaled PA

- Three findings
  - Scaled parallel adapter is the best variant to modify FFN;
  - FFN can better utilize modification at larger capacities;
  - modifying head attentions like prefix tuning can achieve strong performance with only 0.1% parameters

- Mix-And-Match adapter (MAM Adapter)
  - use prefix tuning with a small bottleneck dimension (l = 30) at the attention sub-layers and allocate more parameter budgets to modify FFN representation using the scaled parallel adapter (r = 512).

Design Spaces

- layer grouping:

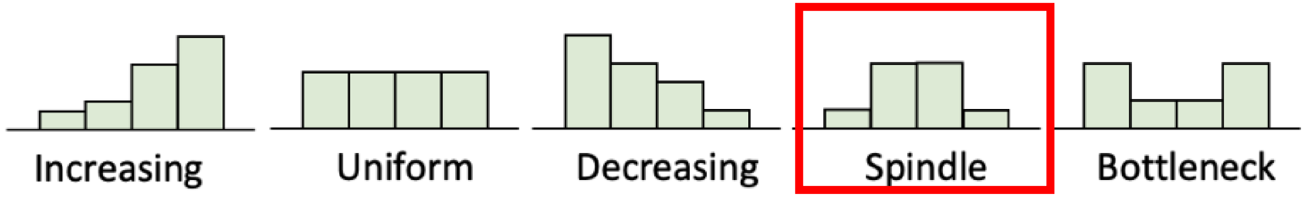
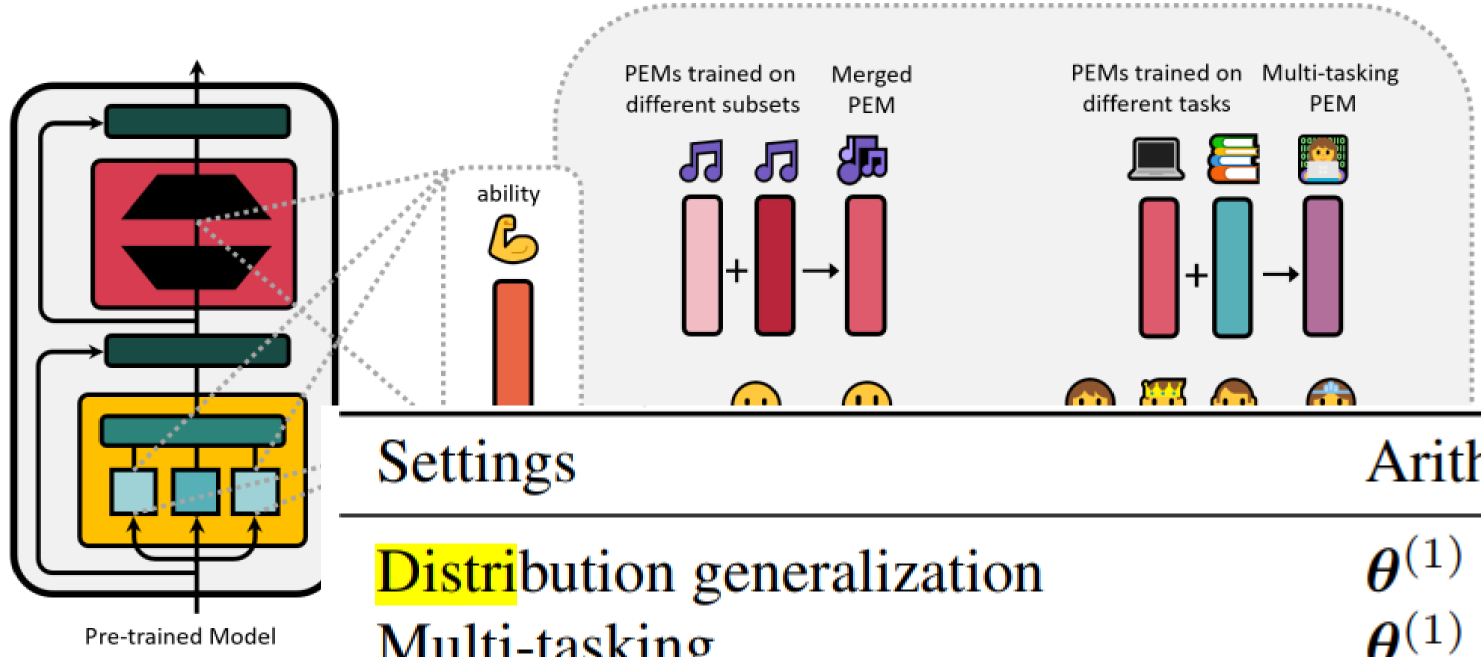


Figure 2: Layer grouping patterns: group ID ($G_1, \ldots G_4$) vs. number of layers per group.

- trainable parameter allocation
  - **Increasing**, **Uniform**, **Decreasing**
- tunable groups
  - part v.s. all
- strategy assignment
  - **Prefix**, **Adapter**, **LoRA**

appropriate

Composing Parameter-Efficient Modules with **Arithmetic Operations**



An ove

Basic Operators:

Addition

$$\boldsymbol{\theta}^{\text{add}} = \boldsymbol{\theta}^{(1)} \oplus \boldsymbol{\theta}^{(2)} = \boldsymbol{\theta}^{(1)} + \boldsymbol{\theta}^{(2)},$$

Negation

ite $\Rightarrow \boldsymbol{h} \leftarrow \boldsymbol{h} + (-\Delta\boldsymbol{h})$

ıan + woman”

Settings	Arithmetic operations
Distribution generalization	$\boldsymbol{\theta}^{(1)} \oplus \boldsymbol{\theta}^{(2)}$
Multi-tasking	$\boldsymbol{\theta}^{(1)} \oplus \boldsymbol{\theta}^{(2)}$
Unlearning	$\ominus\boldsymbol{\theta}$
Domain transfer	$\boldsymbol{\theta}^{(1)} \ominus \boldsymbol{\theta}^{(2)} \oplus \boldsymbol{\theta}^{(3)}$
Detoxifying instruction-tuned LLMs	$\boldsymbol{\theta}^{(1)} \ominus \boldsymbol{\theta}^{(2)}$

Different settings studied in this work and their corresponding arithmetic operations

Automatic Configuration Search

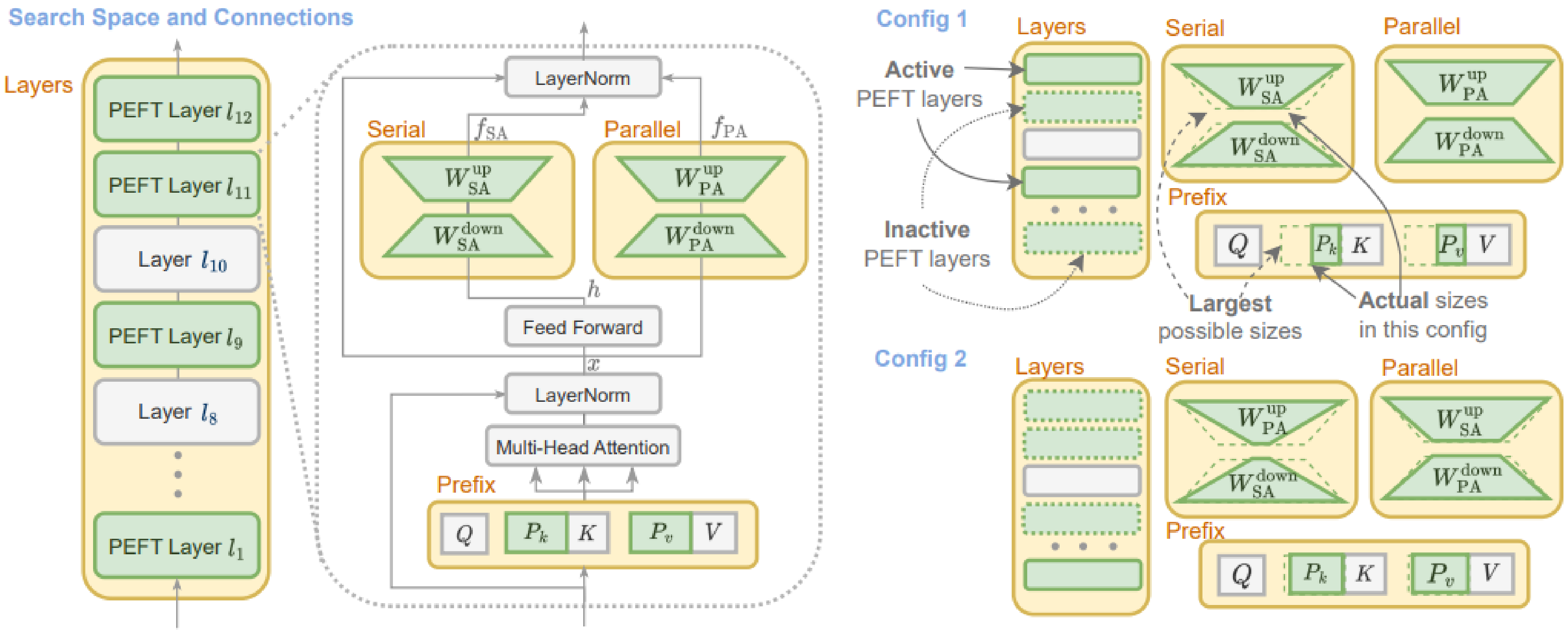


Illustration of the AUTOPEFT **search space**

Three aspects:

PEFT Modules,

Size,

Insertion Layers

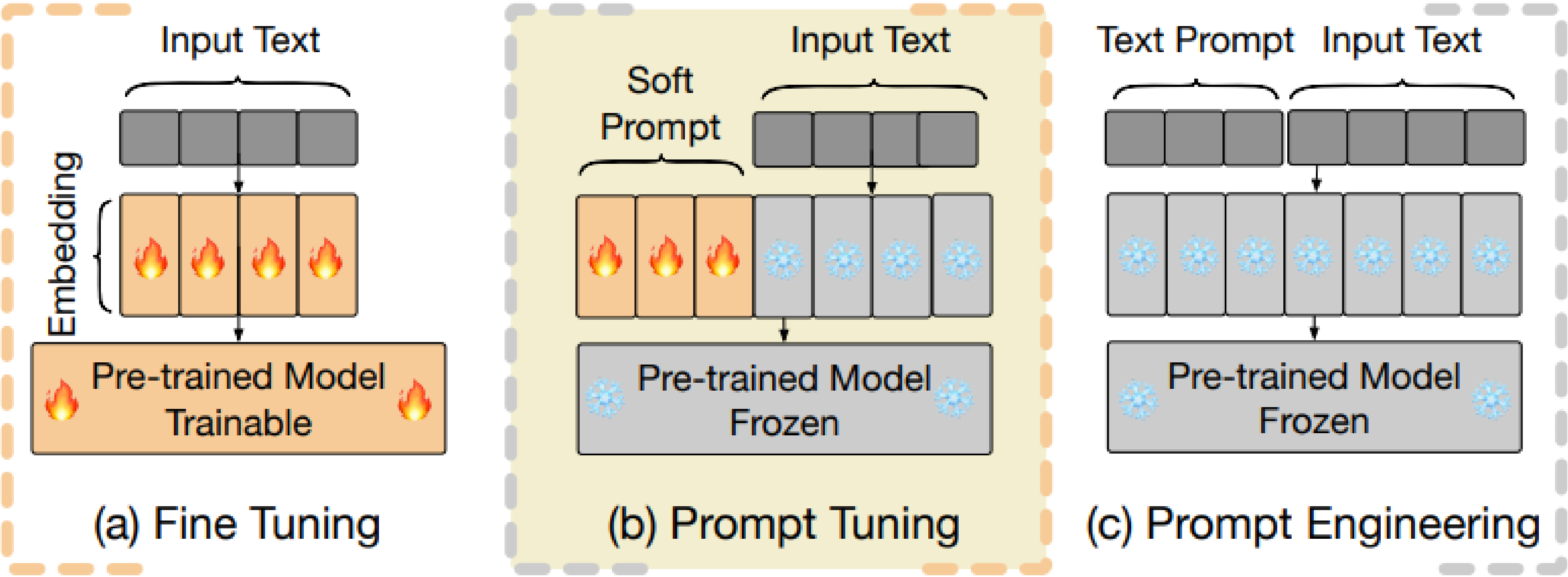
AutoPEFT: Automatic Configuration Search for Parameter-Efficient Fine-Tuning arxiv

SOTA方法的深度探索

- Prompt tuning (PT)
  - Question 1: PT often suffers from slow convergence and is sensitive to the initialization
  - Question 2: PT extends the total length of the input sequence, consequently exacerbating the computation demand

- LoRA
  - Question: ignoring the importance of parameters in different modules

Question 1: Slow convergence and is sensitive to the initialization



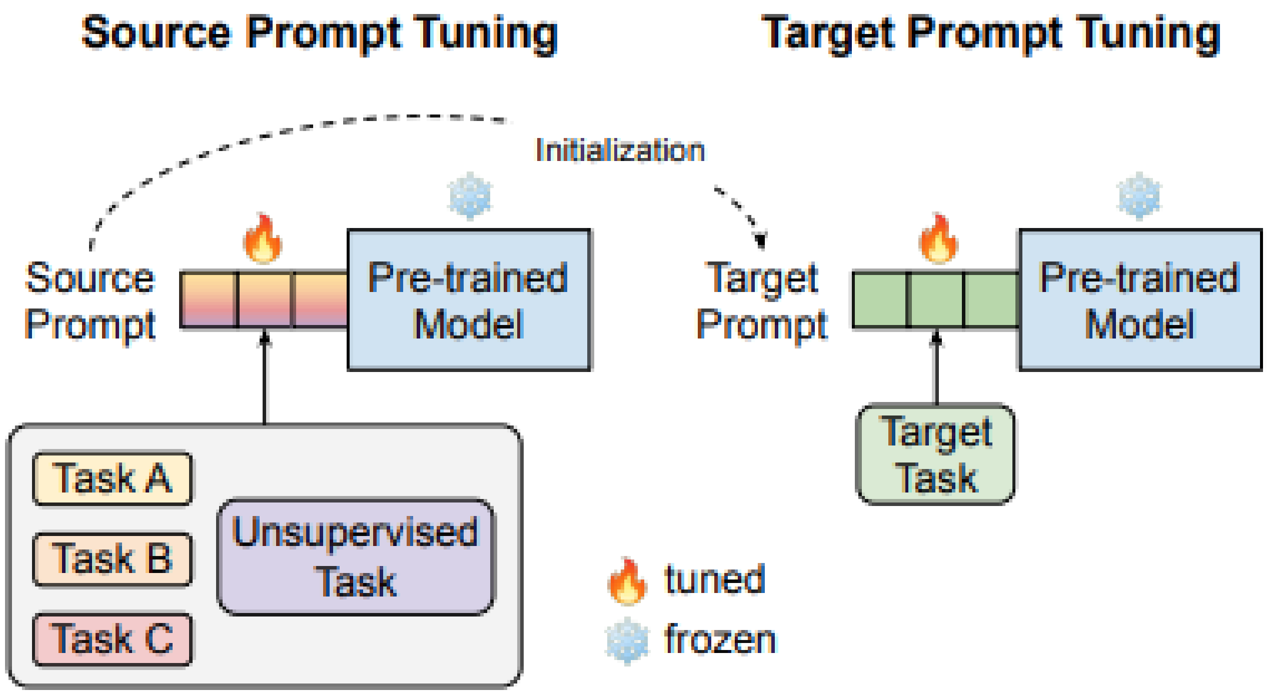
The overview of Fine Tuning (FT), Prompt Tuning (PT), and Prompting Engineering

Question 1: Slow convergence and is sensitive to the initialization

Method: Prompt Transfer

SPoT用一个或多个源任务训练的prompts

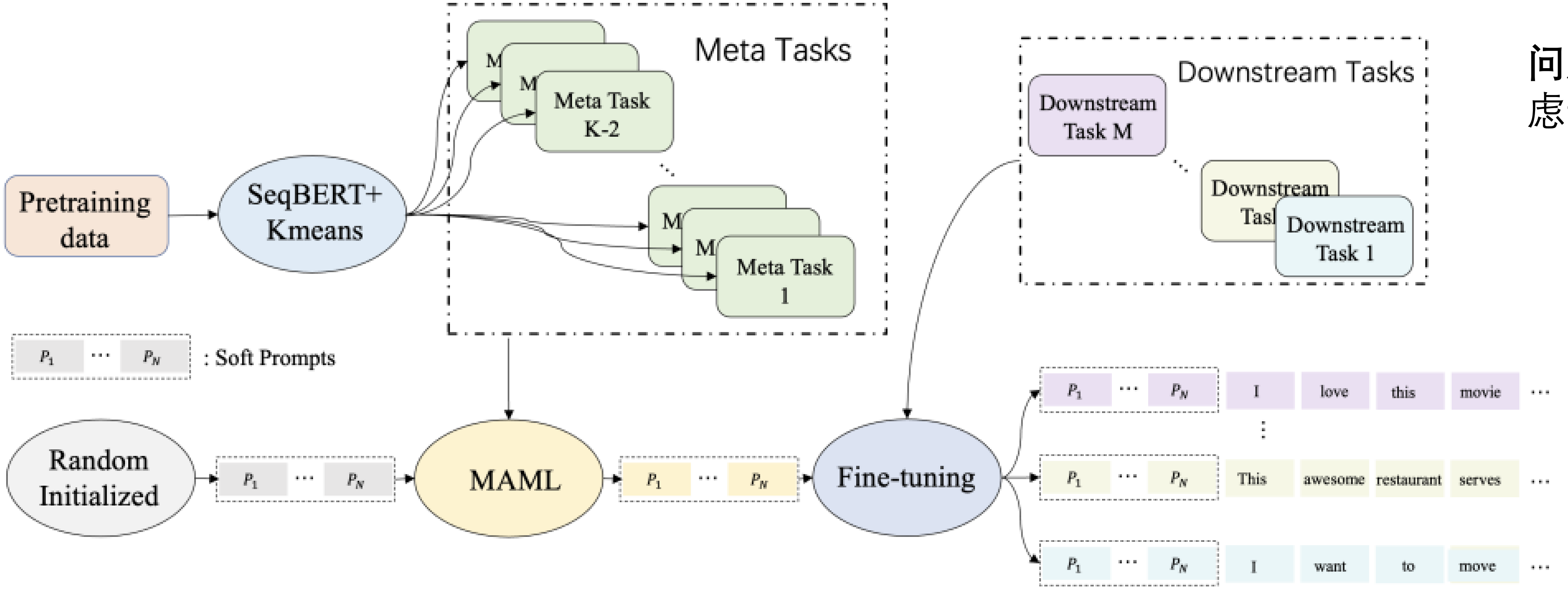
任务之间的相似度是一个重要的影响因素



SPoT: Better Frozen Model Adaptation through Soft Prompt Transfer ACL2022

Question 1: Slow convergence and is sensitive to the initialization

Method: Meta-learning: Learning to learn, aims to improve the learning algorithm itself



问题：之前的工作都没有考虑预训练数据的潜在结构

动机：如何学习任务的一般特征，而不是具体特征

Meta-Learning: Learning to learn, aims to improve the learning algorithm itself

Learning a Better Initialization for Soft Prompts via Meta-Learning COLING2022

Question 2: larger computation demand

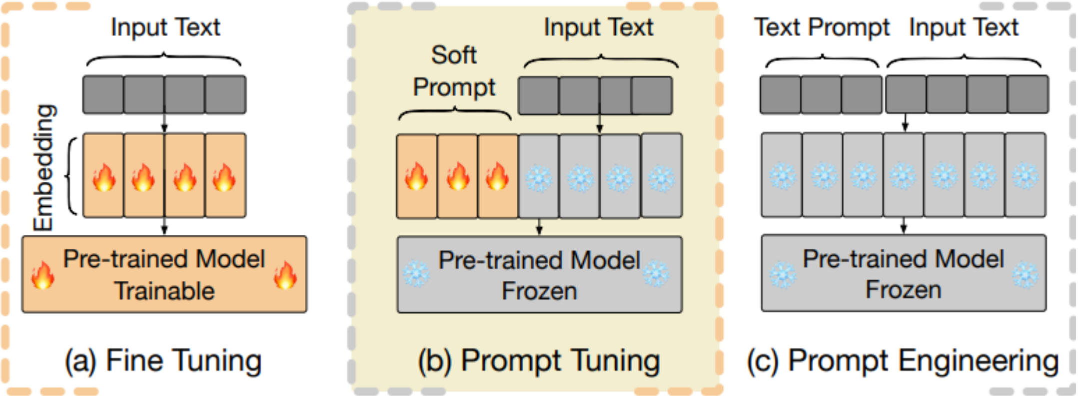


Figure 1: The overview of Fine Tuning (FT), Prompt Tuning (PT), and Prompting Engineering

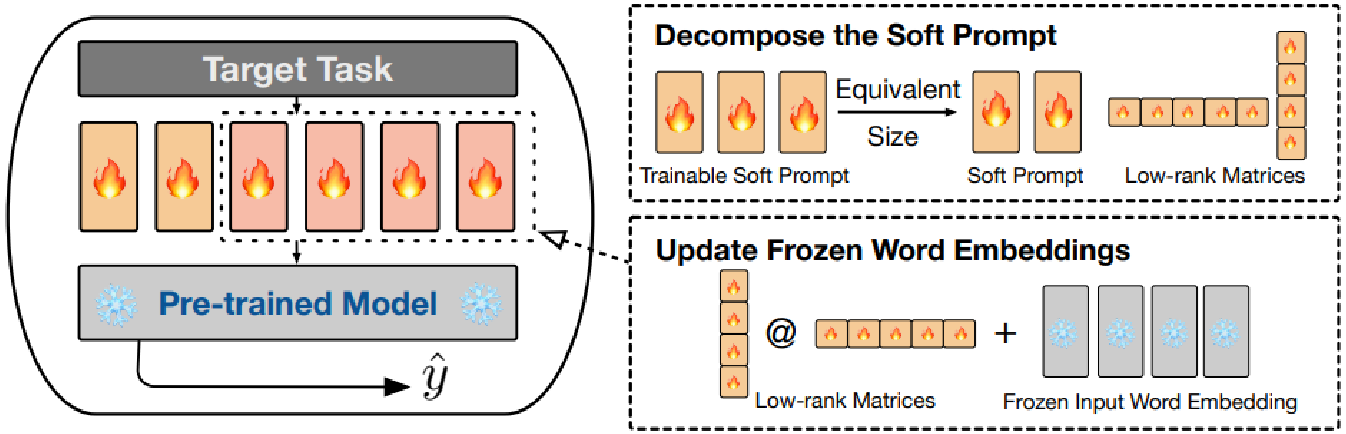
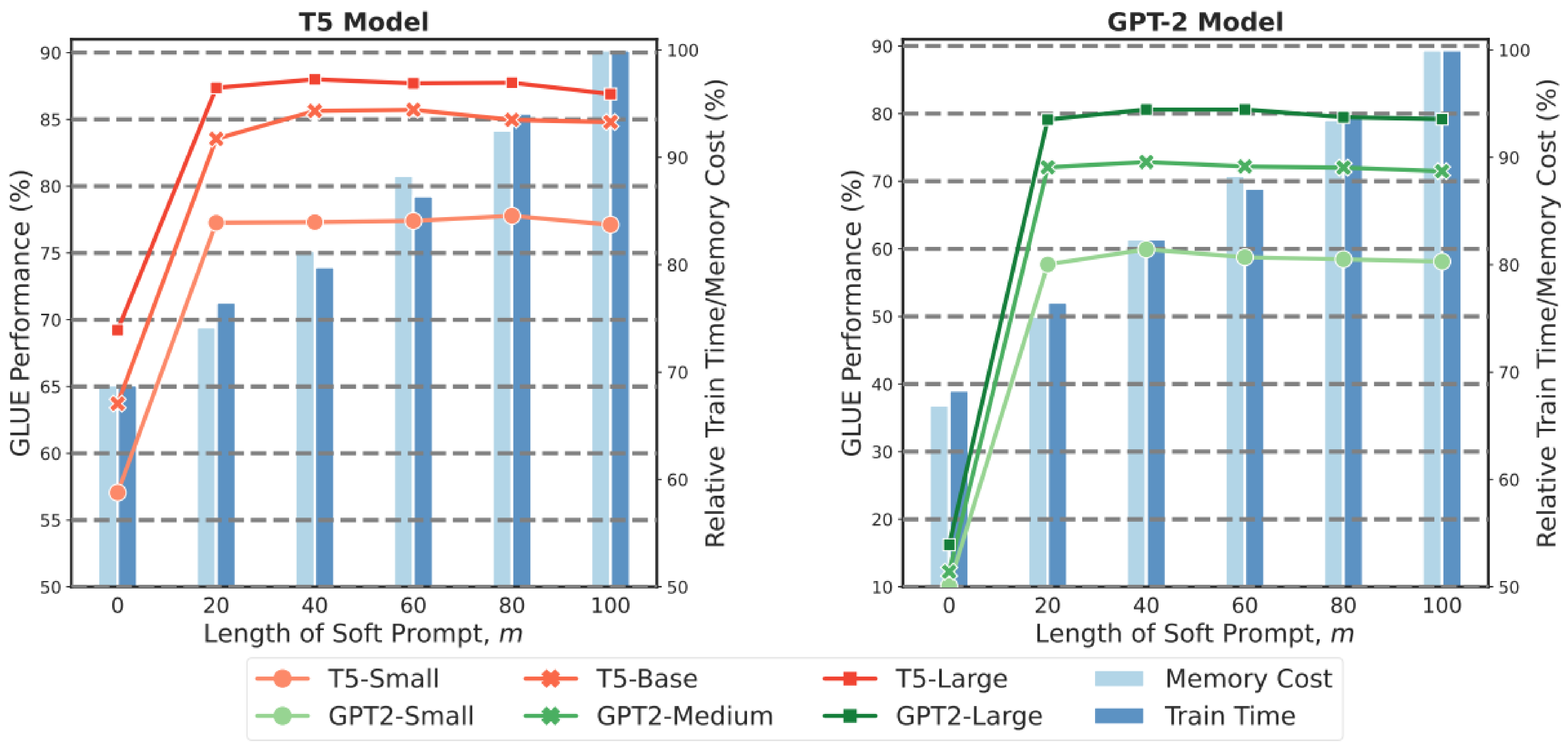


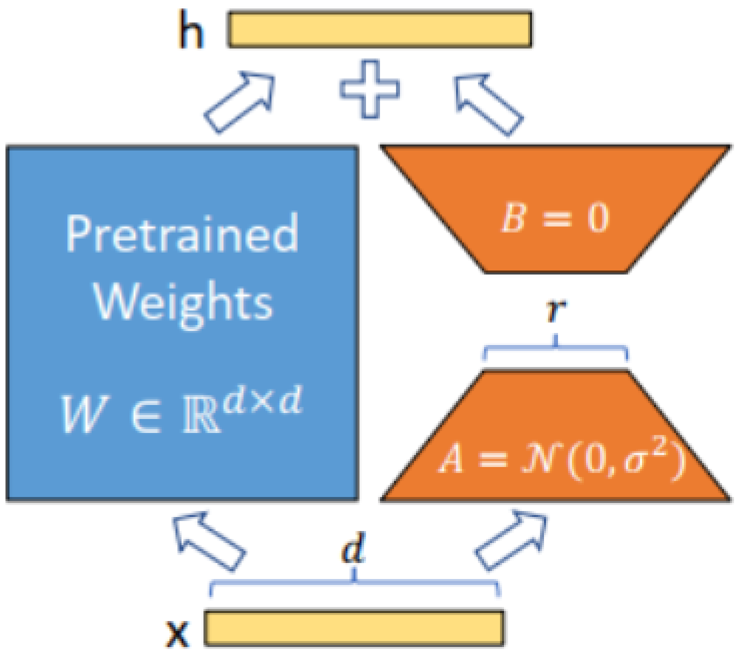
Figure 2: Decomposed Prompt Tuning

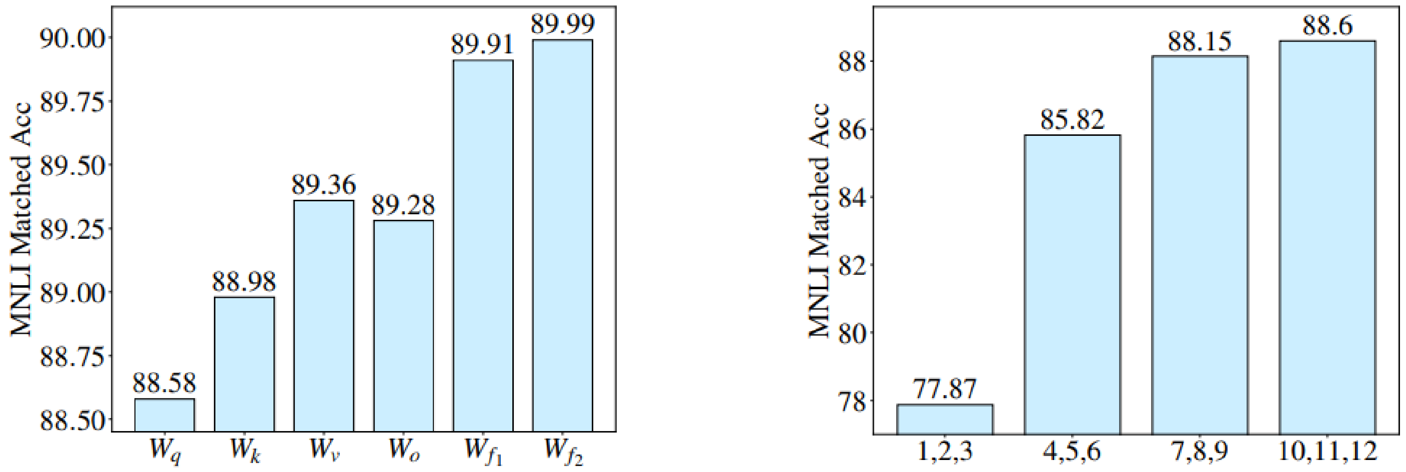
DePT: Decomposed Prompt Tuning for Parameter-Efficient Fine-tuning arxiv 2023/9

Question 2: larger computation demand



Question of LoRA: ignoring the importance of parameters in different modules





(a) Selected weight matrix

(b) Selected layers

The performance of LoRA when fine-tuning specific modules or layers

$$W = W^{(0)} + \boxed{\Delta} = W^{(0)} + \boxed{BA},$$

LoRA

$A \in \mathbb{R}^{r \times d_2}$ and $B \in \mathbb{R}^{d_1 \times r}$

AdaLoRA (Adaptive Low-Rank Adaptation)

How can we allocate the parameter budget adaptively according to importance of modules to improve the performance of parameter-efficient fine-tuning?

Method

1. Reconstructing Low-Rank Matrices

$$W = W^{(0)} + \Delta = W^{(0)} + \boxed{P\Lambda Q,} \qquad W = W^{(0)} + \Delta = W^{(0)} + \boxed{BA},$$

2. Orthogonal constraints for P, Q

$$R(P, Q) = \|P^\top P - I\|_\mathsf{F}^2 + \|QQ^\top - I\|_\mathsf{F}^2.$$

3. Compute the Sensitivity Scores

$$S_{k,i} = s(\lambda_{k,i}) + \frac{1}{d_1}\sum_{j=1}^{d_1} s(P_{k,ji}) + \frac{1}{d_2}\sum_{j=1}^{d_2} s(Q_{k,ij}),$$

Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning ICLR2023

Main results

Method	**# Params**	**MNLI** m/mm	**SST-2** Acc	**CoLA** Mcc	**QQP** Acc/F1	**QNLI** Acc	**RTE** Acc	**MRPC** Acc	**STS-B** Corr	**All** Ave.
Full FT	184M	89.90/90.12	95.63	69.19	**92.40/89.80**	94.03	83.75	89.46	91.60	88.09
BitFit	0.1M	89.37/89.91	94.84	66.96	88.41/84.95	92.24	78.70	87.75	91.35	86.02
HAdapter	1.22M	90.13/90.17	95.53	68.64	91.91/89.27	94.11	84.48	89.95	91.48	88.12
PAdapter	1.18M	90.33/90.39	95.61	68.77	92.04/89.40	94.29	85.20	89.46	91.54	88.24
LoRA$_{r=8}$	1.33M	90.65/90.69	94.95	69.82	91.99/89.38	93.87	85.20	89.95	91.60	88.34
AdaLoRA	1.27M	**90.76/90.79**	**96.10**	**71.45**	**92.23/89.74**	**94.55**	**88.09**	**90.69**	**91.84**	**89.31**
HAdapter	0.61M	90.12/90.23	95.30	67.87	91.65/88.95	93.76	85.56	89.22	91.30	87.93
PAdapter	0.60M	90.15/90.28	95.53	69.48	91.62/88.86	93.98	84.12	89.22	91.52	88.04
HAdapter	0.31M	90.10/90.02	95.41	67.65	91.54/88.81	93.52	83.39	89.25	91.31	87.60
PAdapter	0.30M	89.89/90.06	94.72	69.06	91.40/88.62	93.87	84.48	89.71	91.38	87.90
LoRA$_{r=2}$	0.33M	90.30/90.38	94.95	68.71	91.61/88.91	94.03	85.56	89.71	**91.68**	88.15
AdaLoRA	0.32M	**90.66/90.70**	**95.80**	**70.04**	**91.78/89.16**	**94.49**	**87.36**	**90.44**	91.63	**88.86**

Results with DeBERTaV3-base on GLUE development set

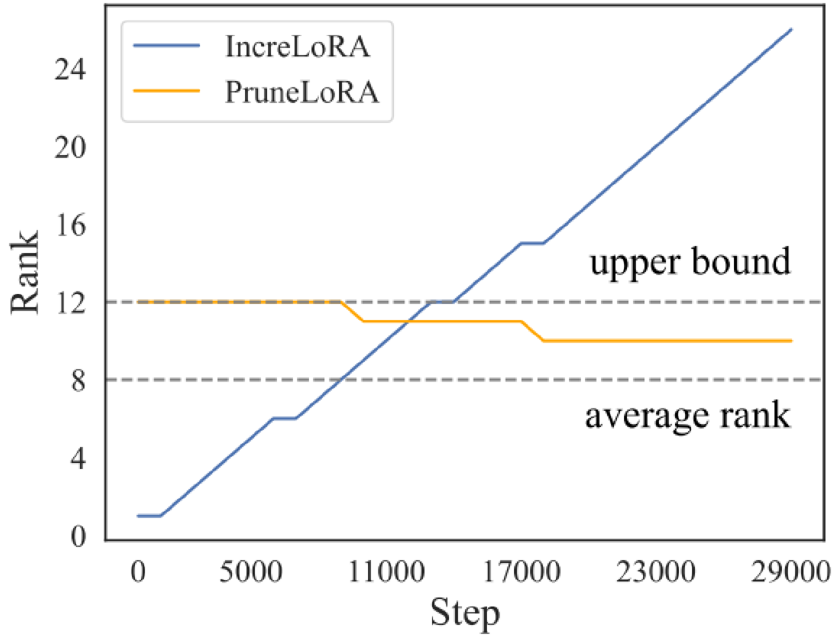
Main results

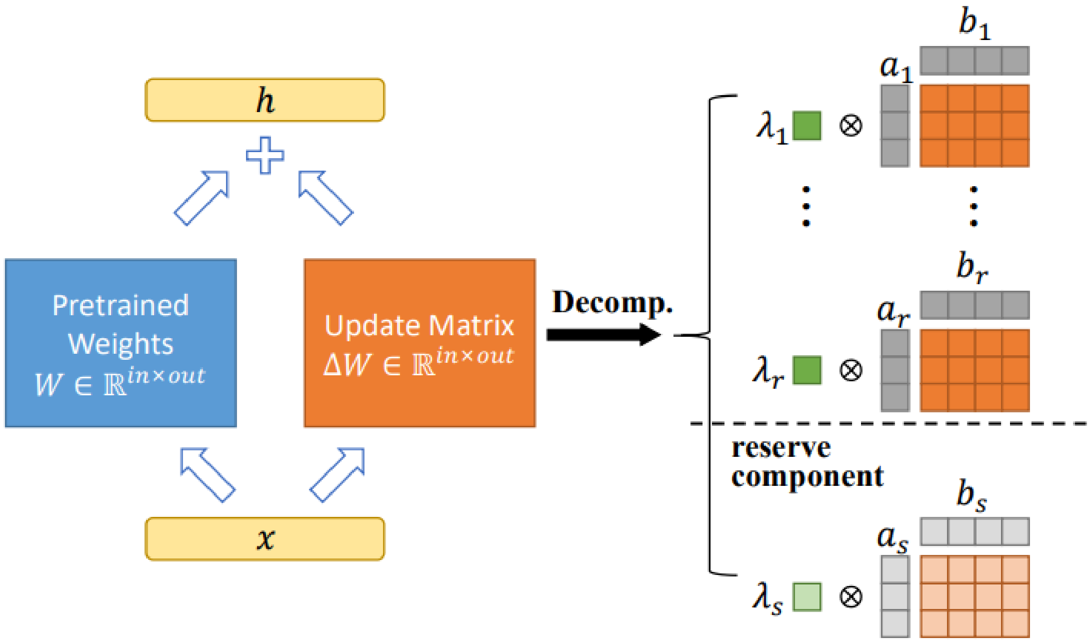
# Params	Method	XSum	CNN/DailyMail
100%	Full FT	**45.49 / 22.33 / 37.26**	44.16 / 21.28 / 40.90
2.20%	LoRA	43.95 / 20.72 / 35.68	**45.03** / 21.84 / 42.15
	AdaLoRA	**44.72 / 21.46 / 36.46**	45.00 / **21.89** / **42.16**
1.10%	LoRA	43.40 / 20.20 / 35.20	44.72 / 21.58 / 41.84
	AdaLoRA	**44.35 / 21.13 / 36.13**	**44.96 / 21.77 / 42.09**
0.26%	LoRA	43.18 / 19.89 / 34.92	43.95 / 20.91 / 40.98
	AdaLoRA	**43.55 / 20.17 / 35.20**	**44.39 / 21.28 / 41.50**
0.13%	LoRA	42.81 / 19.68 / 34.73	43.68 / 20.63 / 40.71
	AdaLoRA	**43.29 / 19.95 / 35.04**	**43.94 / 20.83 / 40.96**

Results with BART-large on XSum and CNN/DailyMail

Incremental Parameter Allocation Method

- Limitations of AdaLoRA
  - under limited training conditions, the upper bound of the rank of the pruned parameter matrix is still affected by the preset values





The variations of rank in *layer.10.attention.self.value proj*

IncreLoRA: Incremental Parameter Allocation Method for Parameter-Efficient Fine-tuning arxiv 2023/8

Main results

Method	#Params	MNLI Acc.	SST-2 Acc.	CoLA Mcc.	QQP Acc.	QNLI Acc.	RTE Acc.	MRPC Acc.	STS-B Corr.	ALL Avg.
Full FT	184M	89.90	95.63	69.19	92.40	94.03	83.75	89.46	91.60	88.25
BitFit	0.1M	89.37	94.84	66.96	88.41	92.24	78.70	87.75	91.35	86.20
HAdapter	1.22M	90.13	95.53	68.64	91.91	94.11	84.48	89.95	91.48	88.28
PAdapter	1.18M	90.33	95.61	68.77	92.04	94.29	85.20	89.46	91.54	88.41
$\text{LoRA}_{r=8}$	1.33M	90.65	94.95	69.82	91.99	93.87	85.20	89.95	91.60	88.50
AdaLoRA	1.27M	90.76	96.10	71.45	92.23	**94.55**	88.09	90.69	91.84	89.46
IncreLoRA	1.33M	**90.93±0.04**	**96.21±0.20**	**71.82±0.76**	**92.25±0.03**	94.45±0.12	**88.21±0.21**	**91.01±0.62**	**91.93±0.26**	**89.61±0.16**
HAdapter	0.61M	90.12	95.30	67.87	91.65	93.76	85.56	89.22	91.30	88.10
PAdapter	0.60M	90.15	95.53	69.48	91.62	93.98	84.12	89.22	91.52	88.20
HAdapter	0.31M	90.10	95.41	67.65	91.54	93.52	83.39	89.25	91.31	87.77
PAdapter	0.30M	89.89	94.72	69.06	91.40	93.87	84.48	89.71	91.38	88.06
$\text{LoRA}_{r=2}$	0.33M	90.30	94.95	68.71	91.61	94.03	85.56	89.71	91.68	88.32
AdaLoRA	0.32M	90.66	95.80	70.04	91.78	94.49	87.36	90.44	91.63	89.03
IncreLoRA	0.34M	**90.71±0.05**	**96.26±0.13**	**71.13±0.77**	**91.78±0.09**	**94.65±0.16**	**88.52±0.30**	**91.13±0.98**	**91.89±0.21**	**89.51±0.22**

Experiment results based on the GLUE development set

Others

- Single--> Multiple
  - Attentional Mixtures of soft prompt, Curriculum prompt,
  - Mixture-of-Adaptations
- Reducing the memory requirements
  - 既微调又量化：QLoRA
- Reducing the storage requirements
  - One Network, Many Masks

Attentional Mixtures of Soft Prompts

针对目标任务，如何利用保存在源prompt的知识？

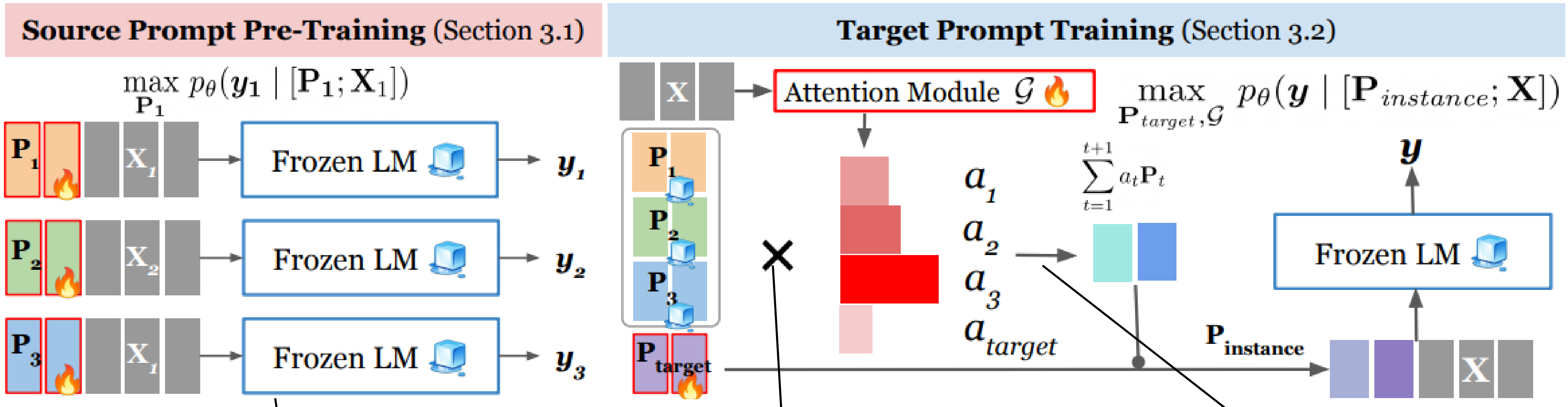


Figure 2: Overview of ATTEMPT. The parts framed in red are updated during training while other parts are intact.

1. 在多个source task上训练多个prompt

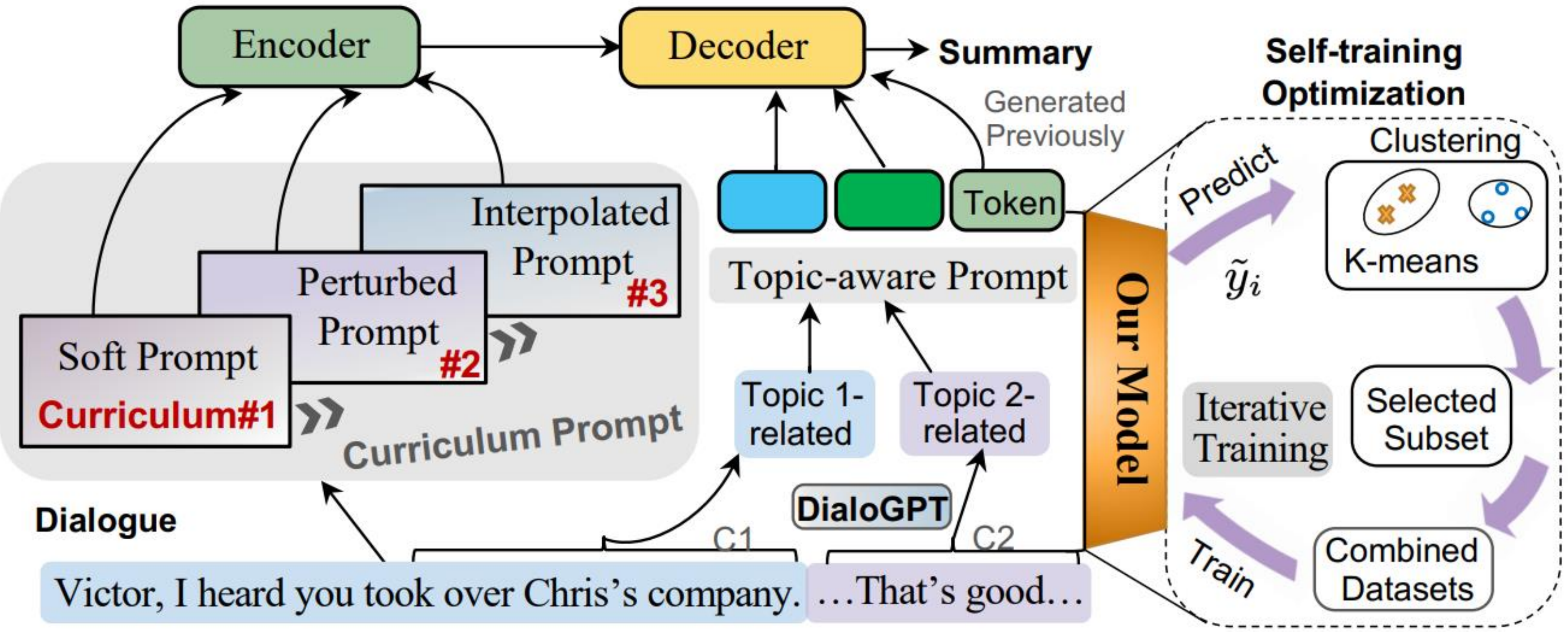
2. Input-prompt Attentions

3. Prompt Interpolation

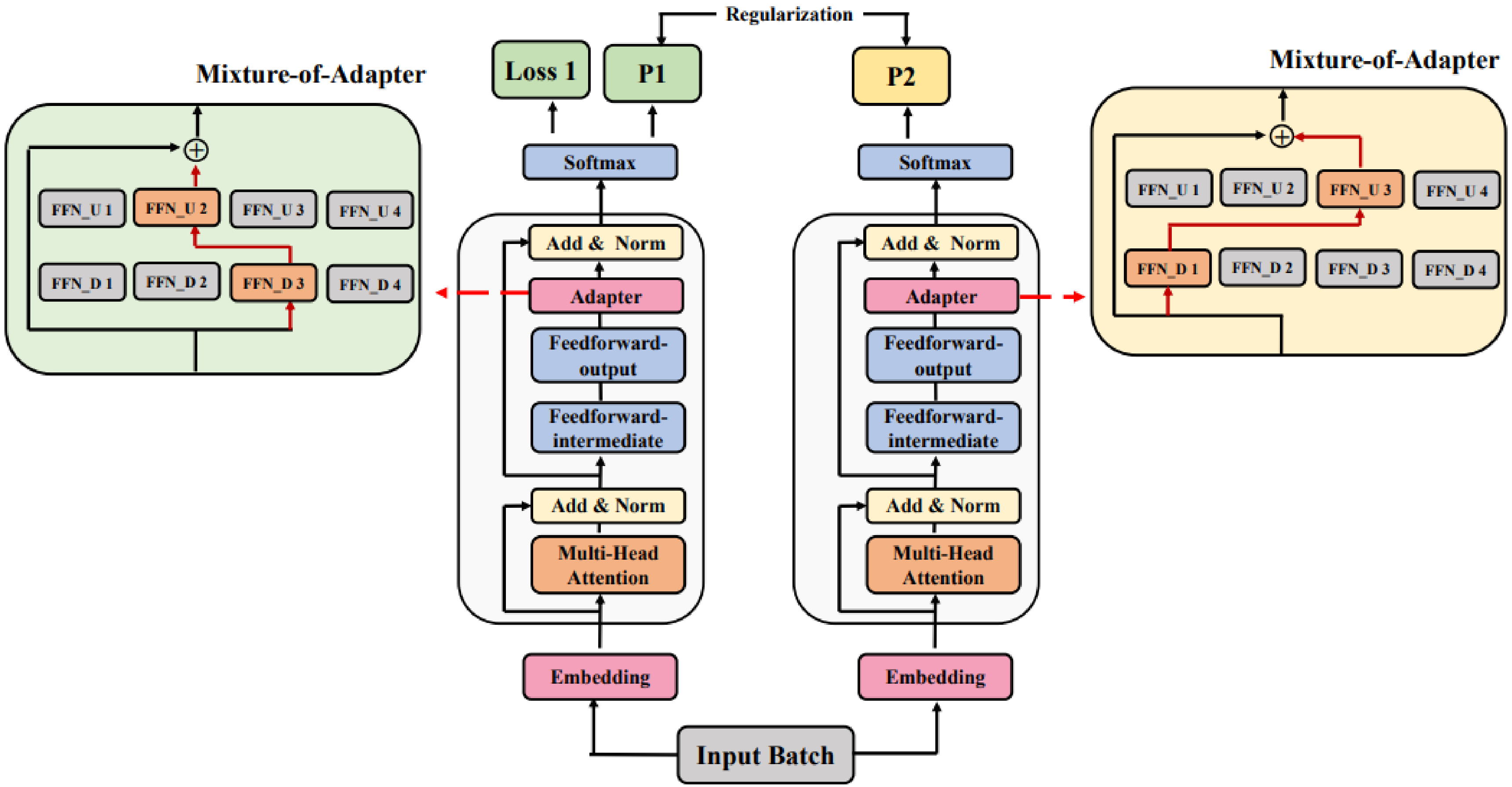
Curriculum Prompt

Transformer-based encoder–decoder architecture with heterogeneous prompts

- **Heterogeneous prompts**
  - Curriculum Prompt
  - Topic-aware Prompt
- **Self-training optimization**



Mixture-of-Adaptations



Mixture-of-Adaptations (AdaMix) with adapters

AdaMix: Mixture-of-Adaptations for Parameter-efficient Model Tuning EMNLP2022

Reducing the memory requirements

- **参数高效的微调（PEFT）**：这种方法在微调模型的时，只调整其中一小部分参数，而大部分预训练的参数则保持不变。其中，低秩适应(LoRA)是最受欢迎的方法，它的主要思想是将适应器权重分解为两个低秩矩阵的乘积。尽管这样可以得到不错的性能，但模型的内存占用依然很大。

- **参数量化**：量化旨在减少LLMs的参数或激活的位宽，从而提高其效率和可伸缩性。简而言之，就是将模型的权重参数从浮点数转化为整数，从而使模型更小更快。但当我们压缩得过猛，比如使用非常低的位数来表示时，模型的准确率会大打折扣。此外，还有一个主要的挑战是如何处理参数分布中的异常值，因为它们在量化时可能导致重大错误。

参数量化--QLoRA

Main innovations:

(a) 4-bit NormalFloat (NF4), a new data type that is information theoretically optimal for normally distributed weights,

(b) Double Quantization to reduce the average memory footprint by quantizing the quantization constants,

(c) Paged Optimizers to manage memory spikes

Table 3: Experiments comparing 16-bit BrainFloat (BF16), 8-bit Integer (Int8), 4-bit Float (FP4), and 4-bit NormalFloat (NF4) on GLUE and Super-NaturalInstructions. QLoRA replicates 16-bit LoRA and full-finetuning.

Dataset	GLUE (Acc.)	Super-NaturalInstructions (RougeL)				
Model	RoBERTa-large	T5-80M	T5-250M	T5-780M	T5-3B	T5-11B
BF16	88.6	40.1	42.1	48.0	54.3	62.0
BF16 replication	88.6	40.0	42.2	47.3	54.9	-
LoRA BF16	88.8	40.5	42.6	47.1	55.4	60.7
QLoRA Int8	88.8	40.4	42.9	45.4	56.5	60.7
QLoRA FP4	88.6	40.3	42.4	47.5	55.6	60.9
QLoRA NF4 + DQ	-	40.4	42.7	47.7	55.3	60.9

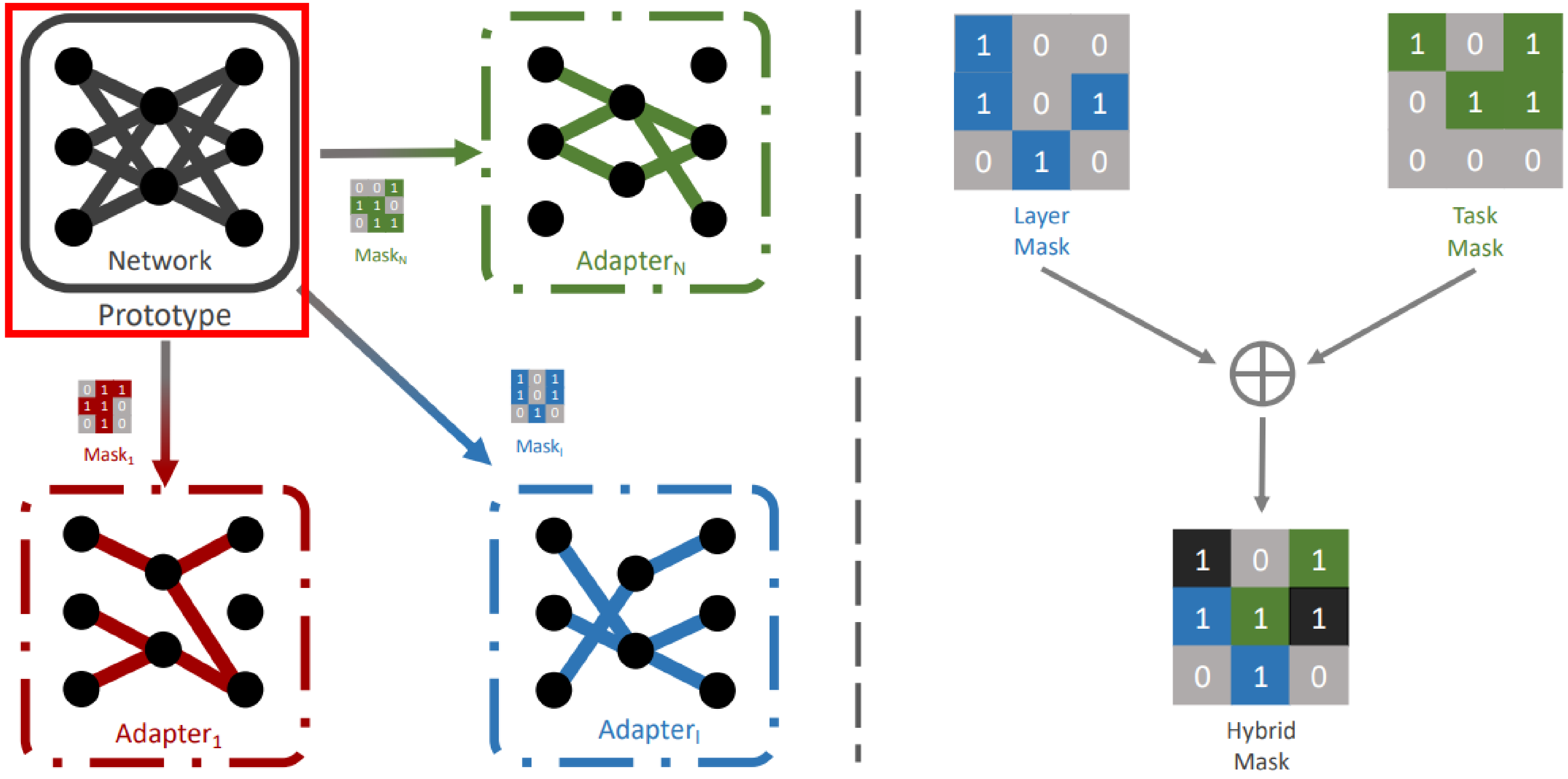
QLoRA: Efficient Finetuning of Quantized LLMs NeurIPS2023

One Network, Many Masks

概念：

$$\text{Bit-Level Storage} = \sum_{i=1}^{N} \rho_i b_i$$

参数量的变化:	存储的变化:
nP,	nP * 32
nP+n,	nP * 1 + n * 32



One Network, Many Masks:Towards More Parameter-Efficient Transfer Learning ACL2023

One Network, Many Masks

Model	%FT Params per task	%BLS per task	CoLA	SST-2	MRPC	QQP	STS-B	MNLI	QNLI	RTE	Avg
Single-Task Learning											
$T5_{BASE}$ †	100.0%	100.000%	54.85	92.19	88.18/91.61	91.46/88.61	89.55/89.41	86.49	91.60	67.39	84.67
Adapter (bn=64)	1.070%	1.070%	62.64	94.07	87.36/91.06	90.25/87.28	89.88/89.55	85.76	92.85	71.01	85.61
Multi-Task Hypernetworks											
Hyperformer++†	0.290%	0.290%	63.73	94.03	89.66/92.63	90.28/87.20	90.00/89.66	85.74	93.02	75.36	86.48
Multi-Task Training											
$T5_{BASE}$ †	12.500%	12.500%	54.88	92.54	**90.15/93.01**	**91.13/88.07**	88.84/88.53	**85.66**	92.04	75.36	85.47
Adapter (bn=64)	0.130%	0.130%	**62.08**	93.57	89.49/92.64	90.25/87.13	87.54/87.41	85.14	92.80	72.22	85.48
Adapter (bn=6)	0.013%	0.013%	58.34	93.61	86.20/90.44	90.10/86.98	86.96/86.66	84.02	92.38	67.63	83.94
$\text{PROPETL}_{\text{Adapter}}$ (bn=64)	0.156%	0.011%	61.43	**94.22**	87.36/90.97	90.13/87.14	90.32/90.12	85.34	**93.01**	75.60	**85.97**
$\text{PROPETL}_{\text{Adapter}}$ (bn=6)	0.016%	**0.001%**	54.59	93.53	87.36/91.02	90.15/87.04	**90.70/90.50**	85.08	92.79	**75.86**	85.32

GLUE Results on T5

Thanks