Large Language Models

Parameter Efficient Fine Tuning II

Mohammad Hossein Rohban

Fall 2023

Courtesy: Most of the slides are adopted from the papers by Li Liang 2021 “Prefix-Tuning: Optimizing Continuous Prompts for Generation,” Hu et al 2021, “Intrinsic Dimensionality Explains The Effectiveness of Language Model Fine-tuning” Aghajanyan et al 2020, “LoRA: Low-rank Adaptation Of Large Language Models” and He et al. 2022 “Towards A Unified View of Parameter-efficient Transfer Learning”

Motivation

- Providing proper task-specific context in the input can steer the LM to solve the task more efficiently.
- Encoding of the original input x will change. Why?
  - Guiding the model to extract relevant information from x.
- Does this context exist? How to find it?

Prefix Tuning

Prefix-Tuning: Optimizing Continuous Prompts for Generation

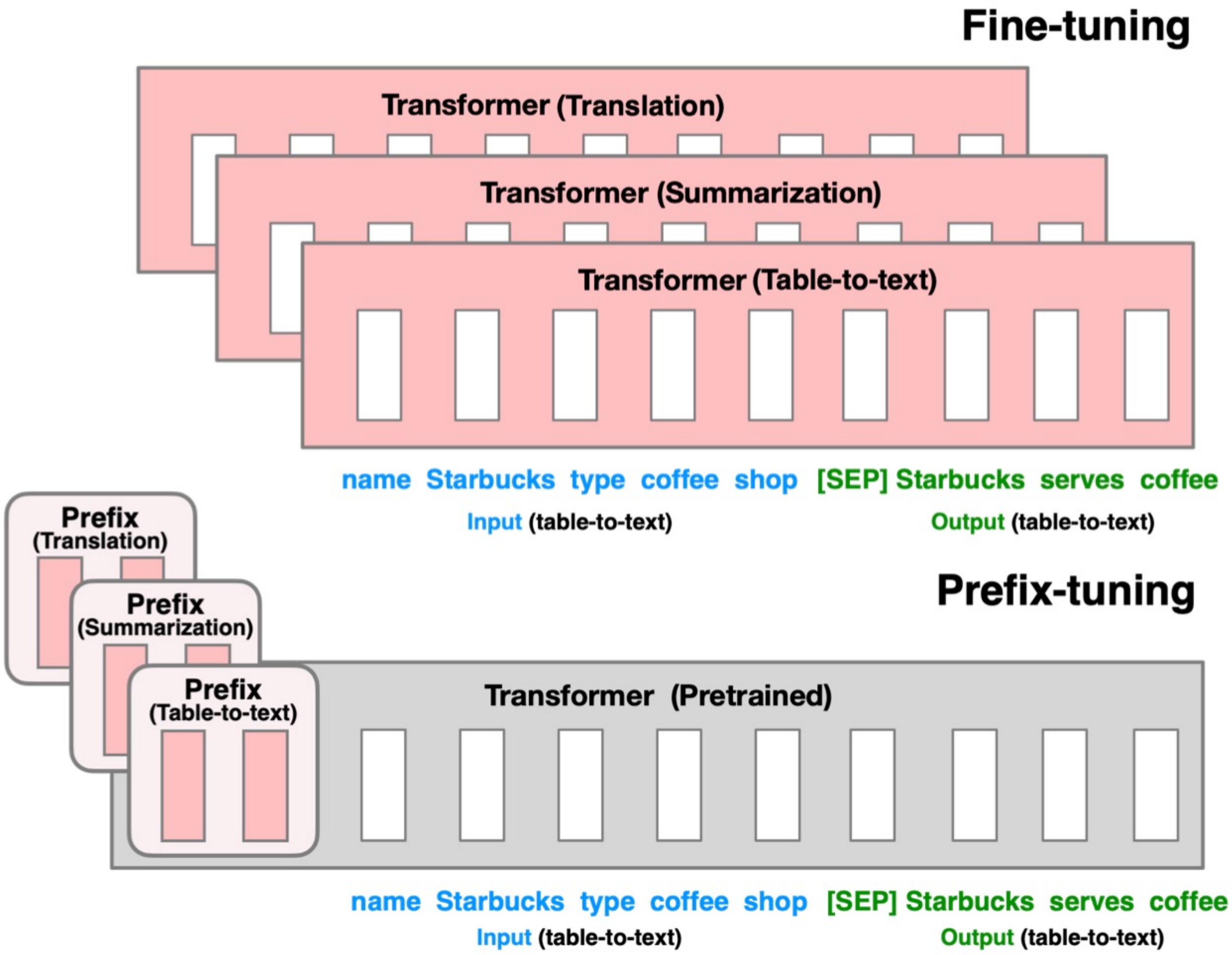
Xiang Lisa Li
Stanford University
xlisali@stanford.edu

Percy Liang
Stanford University
pliang@cs.stanford.edu

- Prepend certain trainable prefix tokens to the input/hidden activations.
- The hidden representation becomes:

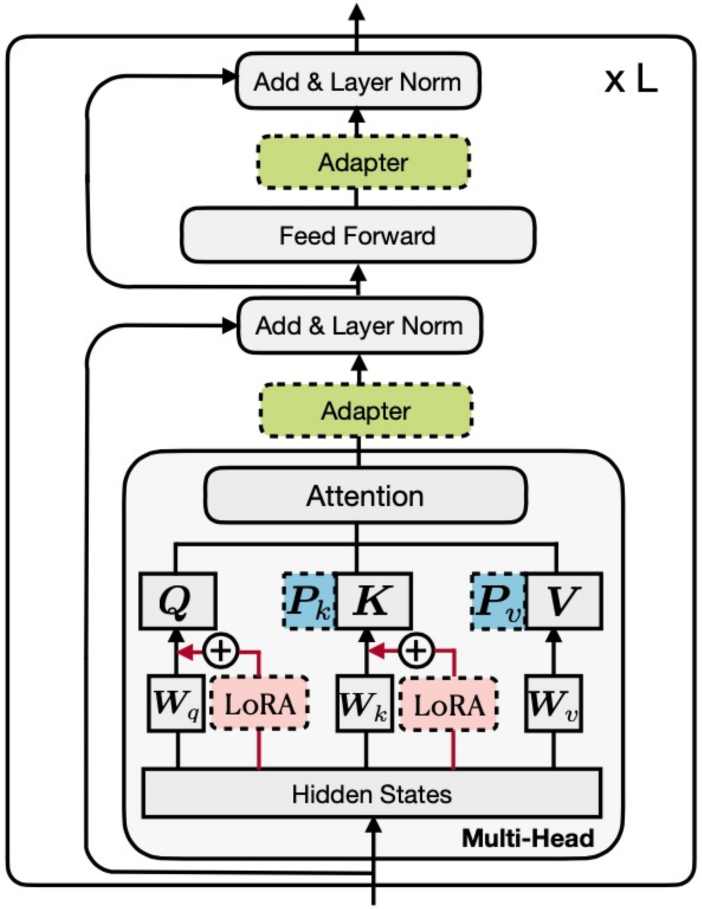
$$h_i = \begin{cases} P_\theta[i,:], & \text{if } i \in \mathsf{P}_{\mathsf{idx}}, \\ \mathrm{LM}_\phi(z_i, h_{<i}), & \text{otherwise.} \end{cases}$$

- All h_i's would indeed be a function of the trainable parameters P_θ. Why?

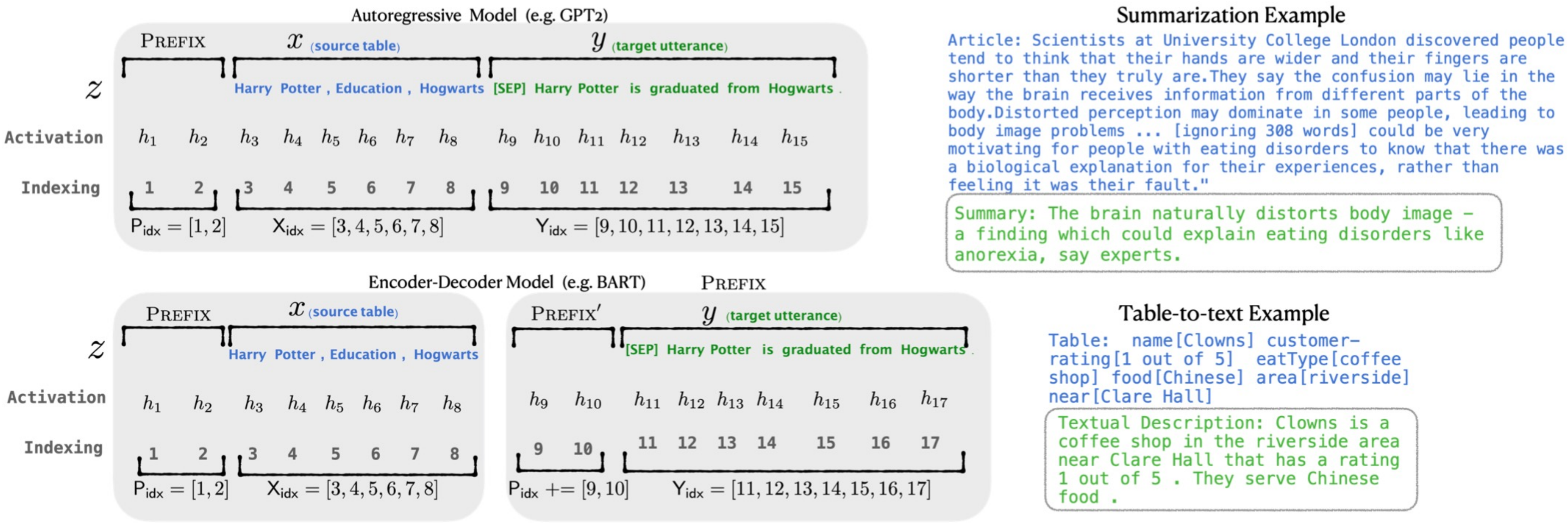

Fine-tuning
Transformer (Translation)
Transformer (Summarization)
Transformer (Table-to-text)
name Starbucks type coffee shop [SEP] Starbucks serves coffee
Input (table-to-text)
Output (table-to-text)
Prefix-tuning
Prefix (Translation)
Prefix (Summarization)
Prefix (Table-to-text)
Transformer (Pretrained)
name Starbucks type coffee shop [SEP] Starbucks serves coffee
Input (table-to-text)
Output (table-to-text)

Prefix Tuning (cont.)

$$\text{head}_i = \text{Attn}(\boldsymbol{x}\boldsymbol{W}_q^{(i)}, \text{concat}(\boldsymbol{P}_k^{(i)}, \boldsymbol{C}\boldsymbol{W}_k^{(i)}), \text{concat}(\boldsymbol{P}_v^{(i)}, \boldsymbol{C}\boldsymbol{W}_v^{(i)}))$$



Prefix Tuning (cont.)



Parametrization of P_θ

- Directly optimizing P_θ leads to unstable optimization.
  - Slight drop in performance.
- Use a smaller P'_θ as input to an MLP with shared trainable weights φ.
- So $P_\theta = MLP_\varphi(P'_\theta)$.
- We can drop P'_θ after training and use the result (P_θ).

Results

	E2E					WebNLG									DART					
	BLEU	NIST	MET	R-L	CIDEr	BLEU			MET			TER ↓			BLEU	MET	TER ↓	Mover	BERT	BLEURT
						S	U	A	S	U	A	S	U	A						
									GPT-2$_{\text{MEDIUM}}$											
FINE-TUNE	68.2	8.62	**46.2**	71.0	2.47	**64.2**	27.7	46.5	**0.45**	0.30	0.38	**0.33**	0.76	0.53	46.2	**0.39**	**0.46**	**0.50**	**0.94**	**0.39**
FT-TOP2	68.1	8.59	46.0	70.8	2.41	53.6	18.9	36.0	0.38	0.23	0.31	0.49	0.99	0.72	41.0	0.34	0.56	0.43	0.93	0.21
ADAPTER(3%)	68.9	8.71	46.1	71.3	2.47	60.4	**48.3**	54.9	0.43	**0.38**	**0.41**	0.35	**0.45**	**0.39**	45.2	0.38	**0.46**	**0.50**	**0.94**	**0.39**
ADAPTER(0.1%)	66.3	8.41	45.0	69.8	2.40	54.5	45.1	50.2	0.39	0.36	0.38	0.40	0.46	0.43	42.4	0.36	0.48	0.47	**0.94**	0.33
PREFIX(0.1%)	**69.7**	**8.81**	46.1	**71.4**	**2.49**	62.9	45.6	**55.1**	0.44	**0.38**	**0.41**	0.35	0.49	0.41	**46.4**	0.38	**0.46**	**0.50**	**0.94**	**0.39**
									GPT-2$_{\text{LARGE}}$											
FINE-TUNE	68.5	8.78	46.0	69.9	2.45	**65.3**	43.1	55.5	**0.46**	0.38	**0.42**	**0.33**	0.53	0.42	**47.0**	**0.39**	0.46	**0.51**	**0.94**	**0.40**
Prefix	**70.3**	**8.85**	**46.2**	**71.7**	**2.47**	63.4	**47.7**	**56.3**	0.45	**0.39**	**0.42**	0.34	**0.48**	**0.40**	46.7	**0.39**	**0.45**	**0.51**	**0.94**	**0.40**
SOTA	68.6	8.70	45.3	70.8	2.37	63.9	52.8	57.1	0.46	0.41	0.44	-	-	-	-	-	-	-	-	-

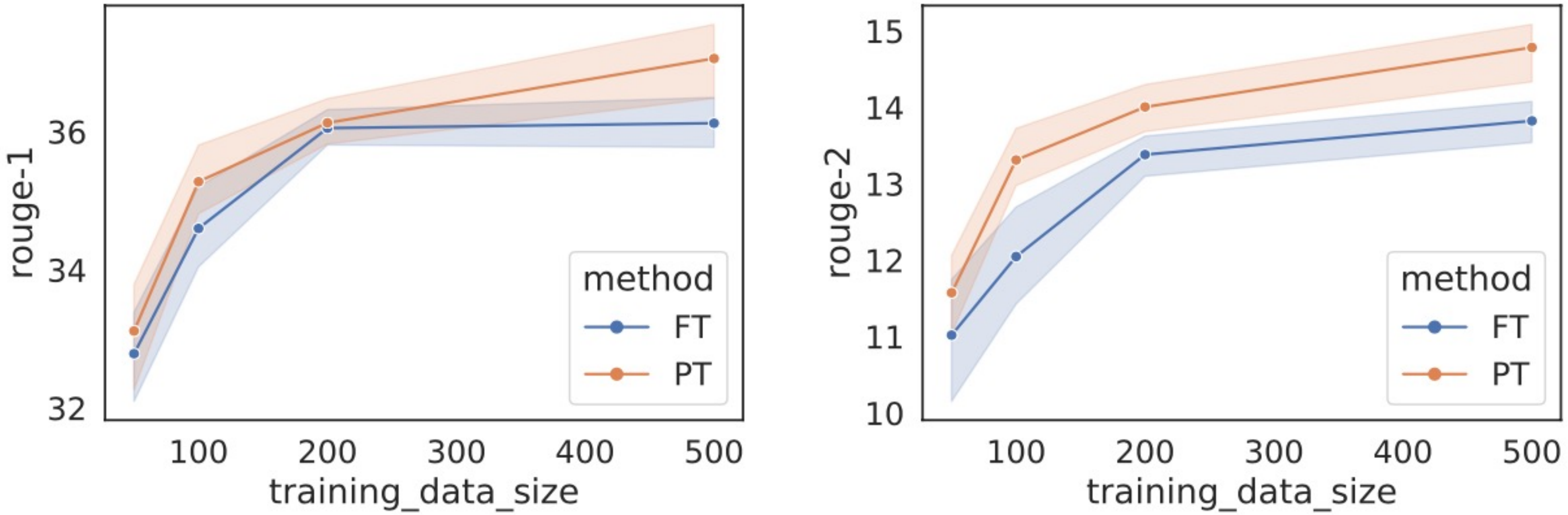
Table 1: Metrics (higher is better, except for TER) for table-to-text generation on E2E (left), WebNLG (middle) and DART (right). With only 0.1% parameters, Prefix-tuning outperforms other lightweight baselines and achieves a comparable performance with fine-tuning. The best score is boldfaced for both GPT-2$_{\text{MEDIUM}}$ and GPT-2$_{\text{LARGE}}$.

Qualitative Results on Table-to-Text (low data setting)

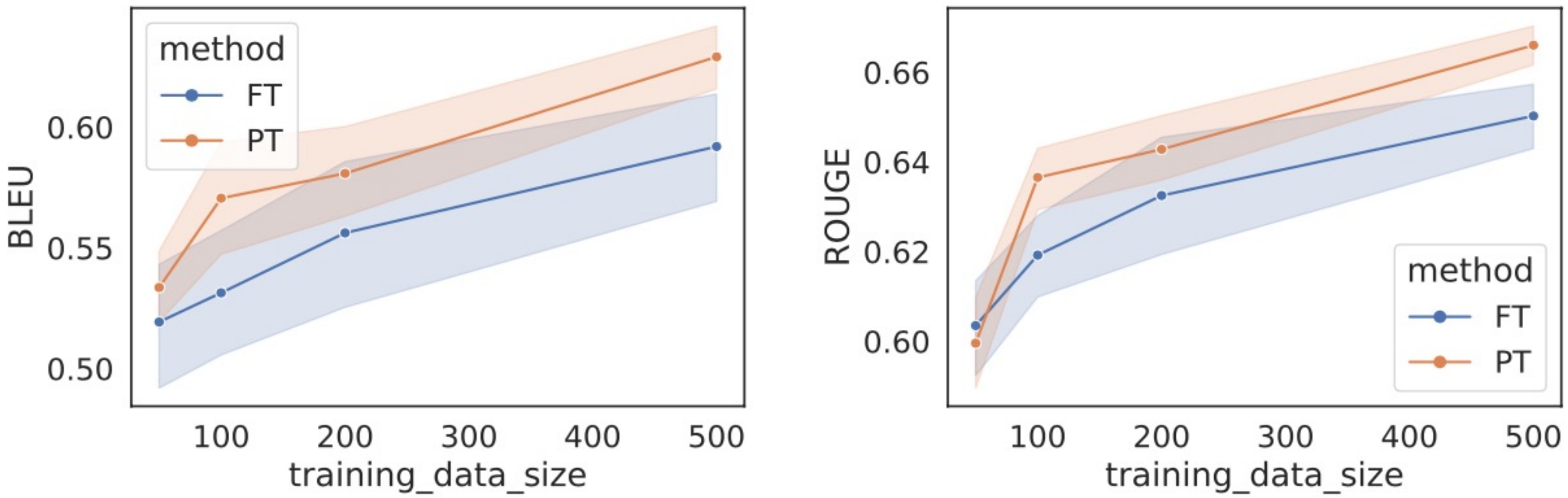
Source	name : The Eagle | type : coffee shop | food : Chinese | price : cheap | customer rating : average | area : riverside | family friendly : no | near : Burger King
Prefix (50)	The Eagle is a cheap Chinese coffee shop located near Burger King.
Prefix (100)	The Eagle is a cheap coffee shop located in the riverside near Burger King. It has average customer ratings.
Prefix (200)	The Eagle is a cheap Chinese coffee shop located in the riverside area near Burger King. It has average customer ratings.
Prefix (500)	The Eagle is a coffee shop that serves Chinese food. It is located in the riverside area near Burger King. It has an average customer rating and is not family friendly.
FT (50)	The Eagle coffee shop is located in the riverside area near Burger King.
FT (100)	The Eagle is a cheap coffee shop near Burger King in the riverside area. It has a low customer rating and is not family friendly.
FT (200)	The Eagle is a cheap Chinese coffee shop with a low customer rating. It is located near Burger King in the riverside area.
FT (500)	The Eagle is a cheap Chinese coffee shop with average customer ratings. It is located in the riverside area near Burger King.

Prefix-tuning Outperforms FT in low-data regimes

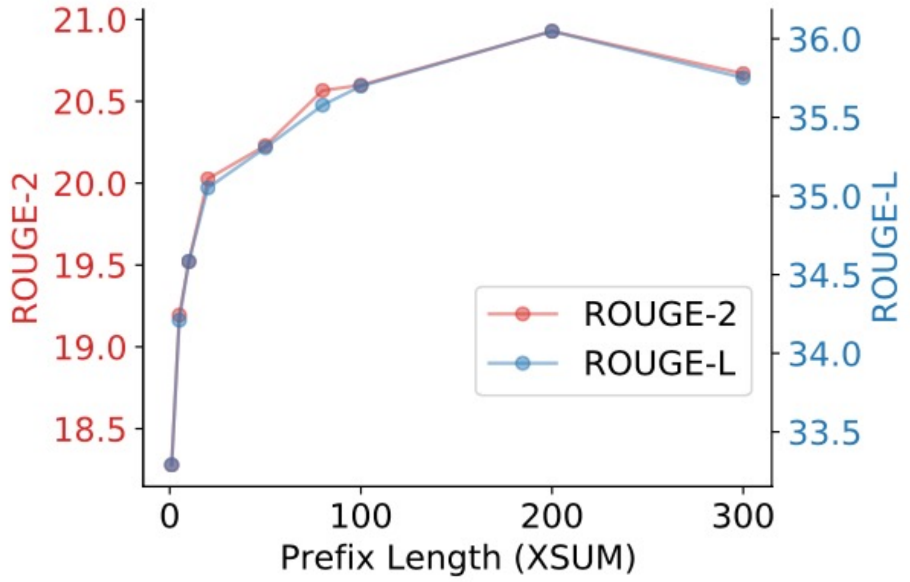
Summarization

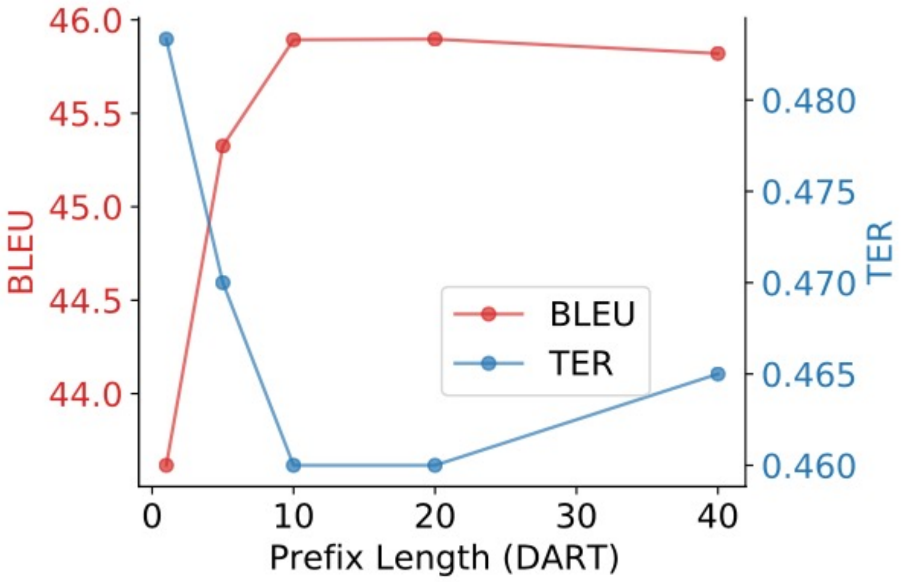


Text-to-Table

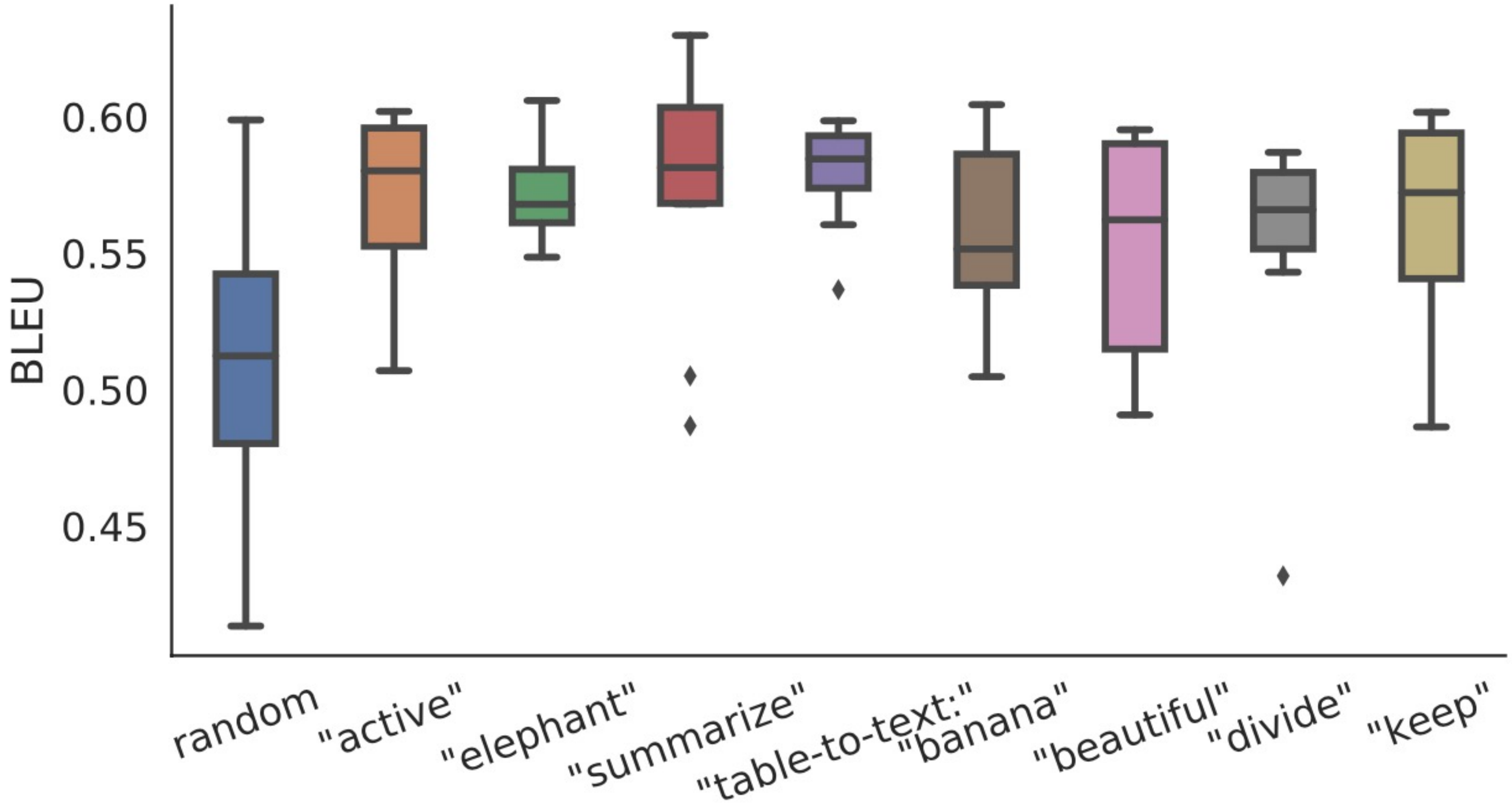


Ablation (Prefix length)





Ablation (Initialization of Prefixes)



INTRINSIC DIMENSIONALITY EXPLAINS THE EFFECTIVENESS OF LANGUAGE MODEL FINE-TUNING

Armen Aghajanyan, Luke Zettlemoyer, Sonal Gupta
Facebook
{armenag,lsz,sonalgupta}@fb.com

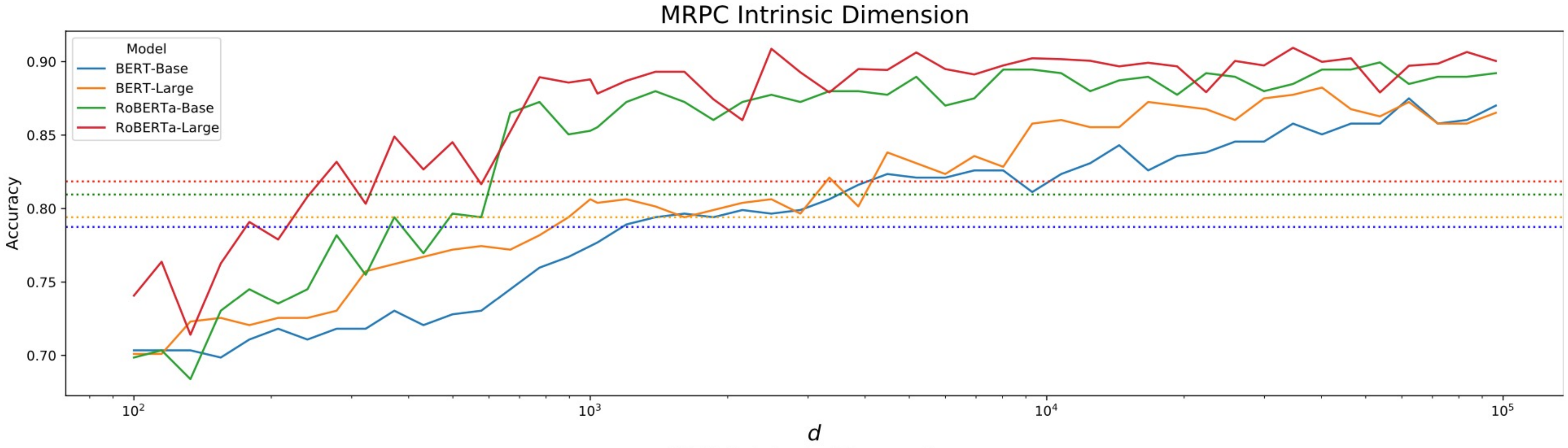
Intrinsic Dimensionality of a Model

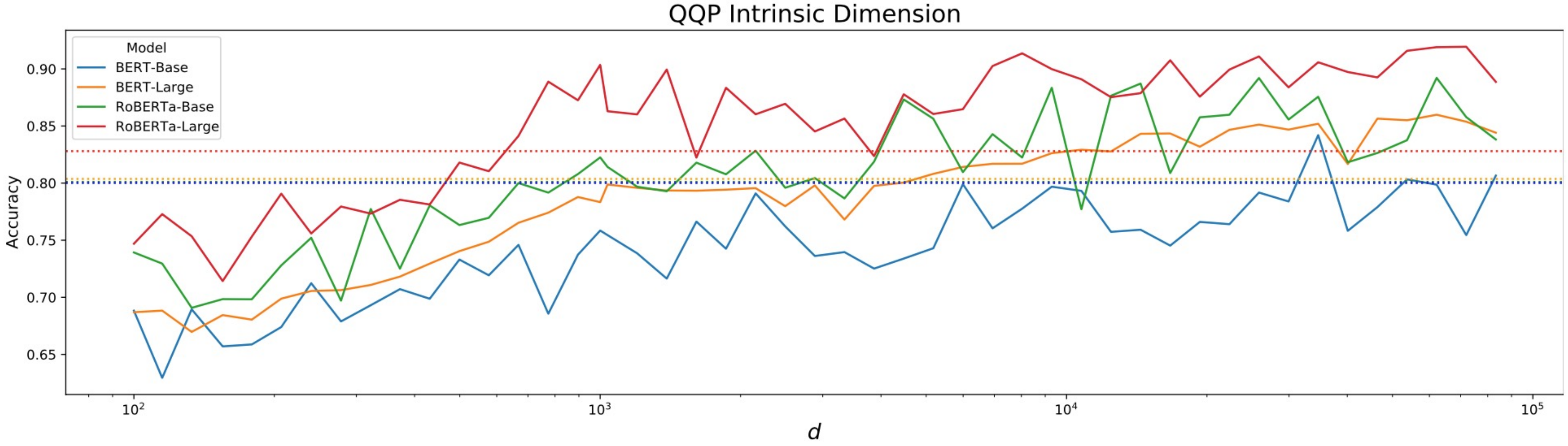
- Model with trainable parameters $\theta^D \in \mathbb{R}^D$.
- Map θ to a lower dimensional space $\theta^d \in \mathbb{R}^d$.
- Solve the optimization (training) in that space:

$$\theta^D = \theta_0^D + P(\theta^d)$$

with $\theta^D = P(\theta^d)$ (FastFood Transform)

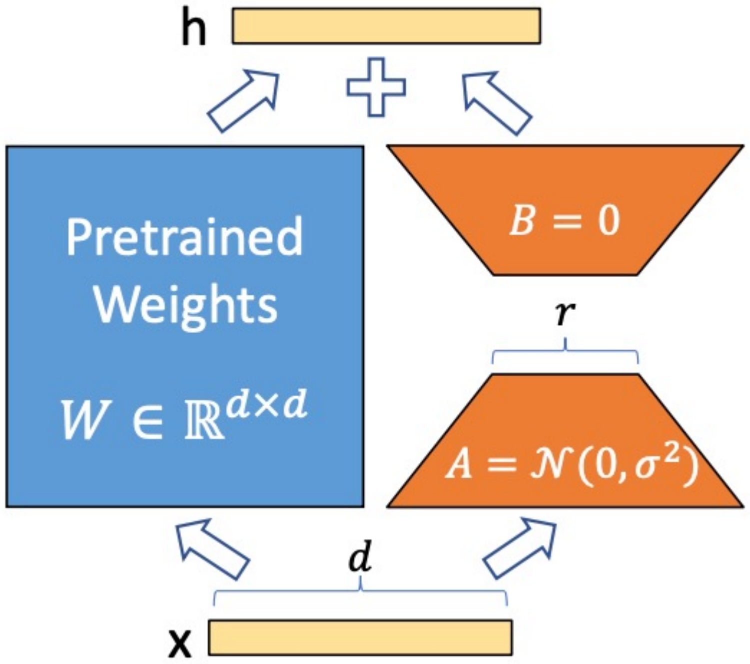
- Let d_{90} be the dimensionality that results to 90% of the performance of full fine tuning.
- Structure aware intrinsic dimension $\theta_i^D = \theta_{0,i}^D + \lambda_i P(\theta^{d-m})_i$


MRPC Intrinsic Dimension
Model
BERT-Base
BERT-Large
RoBERTa-Base
RoBERTa-Large
Accuracy
0.70
0.75
0.80
0.85
0.90
10^2
10^3
10^4
10^5
d


QQP Intrinsic Dimension
Model
BERT-Base
BERT-Large
RoBERTa-Base
RoBERTa-Large
Accuracy
0.65
0.70
0.75
0.80
0.85
0.90
10^2
10^3
10^4
10^5
d

LoRA (Low Rank Adaptation)

- Learned overparameterized models facilitate learning on a low dimensional space.
- So   … weight updates could possibly be low rank.



LoRA: Low-Rank Adaptation of Large Language Models

Edward Hu* **Yelong Shen*** **Phillip Wallis** **Zeyuan Allen-Zhu**
Yuanzhi Li **Shean Wang** **Lu Wang** **Weizhu Chen**
Microsoft Corporation
{edwardhu, yeshe, phwallis, zeyuana,
yuanzhil, swang, luw, wzchen}@microsoft.com
yuanzhil@andrew.cmu.edu
(Version 2)

Problem Statement

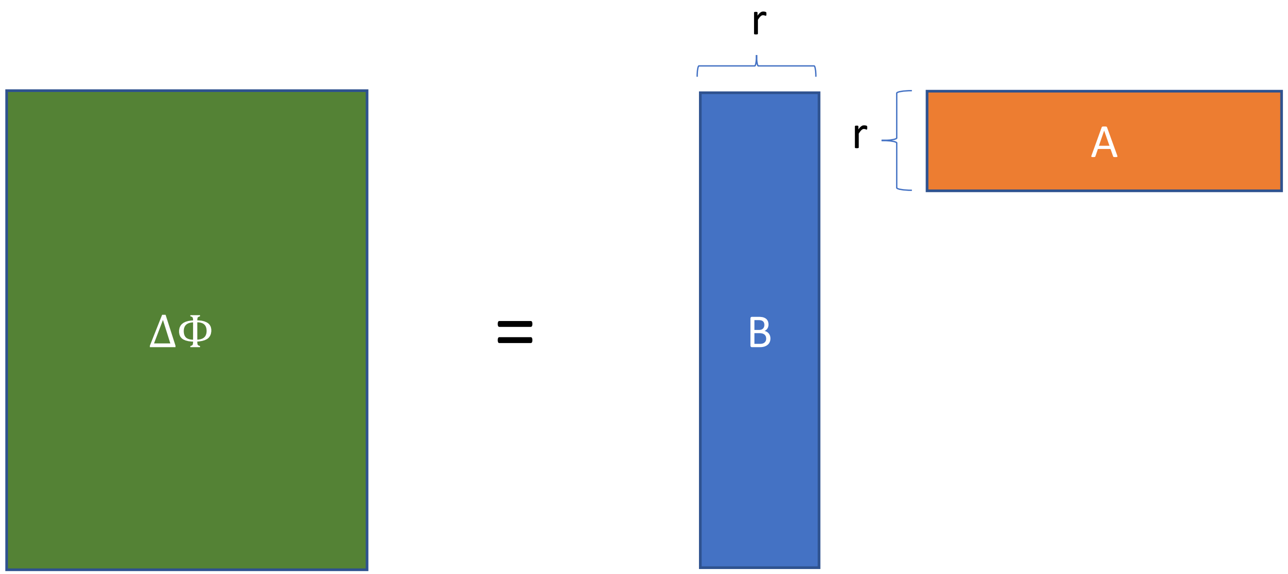
- Given a pretrained autoregressive language model $P_{\Phi_0}(y|x)$.
- Also given a downstream conditional text generation task $\mathcal{Z} = \{(x_i, y_i)\}_{i=1..N}$.
    - e.g. NL2SQL x_i = seq. of natural lang. query; y_i = SQL command
- Update the weights to $\Phi_0 + \Delta\Phi$ to optimize:

$$\max_{\Phi} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log\left(P_{\Phi}(y_t|x, y_{<t})\right)$$

- Now let $\Delta\Phi(\Theta)$ be a function of Θ, which lives in a lower dimensional space.

Solution

- We let $\Delta\Phi(\Theta) = BA$, so $\Theta = (A, B)$.



- Random Gaussian initialization of A, and B = 0. Why?
- Only weights in the self-attention module are trainable; MLPs are frozen.

Results

Model & Method	# Trainable Parameters	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
RoB$_{\text{base}}$ (FT)*	125.0M	**87.6**	94.8	90.2	**63.6**	92.8	**91.9**	78.7	91.2	86.4
RoB$_{\text{base}}$ (BitFit)*	0.1M	84.7	93.7	**92.7**	62.0	91.8	84.0	81.5	90.8	85.2
RoB$_{\text{base}}$ (Adpt$^{\text{D}}$)*	0.3M	87.1$_{\pm.0}$	94.2$_{\pm.1}$	88.5$_{\pm1.1}$	60.8$_{\pm.4}$	93.1$_{\pm.1}$	90.2$_{\pm.0}$	71.5$_{\pm2.7}$	89.7$_{\pm.3}$	84.4
RoB$_{\text{base}}$ (Adpt$^{\text{D}}$)*	0.9M	87.3$_{\pm.1}$	94.7$_{\pm.3}$	88.4$_{\pm.1}$	62.6$_{\pm.9}$	93.0$_{\pm.2}$	90.6$_{\pm.0}$	75.9$_{\pm2.2}$	90.3$_{\pm.1}$	85.4
RoB$_{\text{base}}$ (LoRA)	0.3M	87.5$_{\pm.3}$	**95.1**$_{\pm.2}$	89.7$_{\pm.7}$	63.4$_{\pm1.2}$	**93.3**$_{\pm.3}$	90.8$_{\pm.1}$	**86.6**$_{\pm.7}$	**91.5**$_{\pm.2}$	**87.2**
RoB$_{\text{large}}$ (FT)*	355.0M	90.2	**96.4**	**90.9**	68.0	94.7	**92.2**	86.6	92.4	88.9
RoB$_{\text{large}}$ (LoRA)	0.8M	**90.6**$_{\pm.2}$	96.2$_{\pm.5}$	**90.9**$_{\pm1.2}$	**68.2**$_{\pm1.9}$	**94.9**$_{\pm.3}$	91.6$_{\pm.1}$	**87.4**$_{\pm2.5}$	**92.6**$_{\pm.2}$	**89.0**
RoB$_{\text{large}}$ (Adpt$^{\text{P}}$)†	3.0M	90.2$_{\pm.3}$	96.1$_{\pm.3}$	90.2$_{\pm.7}$	**68.3**$_{\pm1.0}$	**94.8**$_{\pm.2}$	**91.9**$_{\pm.1}$	83.8$_{\pm2.9}$	92.1$_{\pm.7}$	88.4
RoB$_{\text{large}}$ (Adpt$^{\text{P}}$)†	0.8M	**90.5**$_{\pm.3}$	**96.6**$_{\pm.2}$	89.7$_{\pm1.2}$	67.8$_{\pm2.5}$	**94.8**$_{\pm.3}$	91.7$_{\pm.2}$	80.1$_{\pm2.9}$	91.9$_{\pm.4}$	87.9
RoB$_{\text{large}}$ (Adpt$^{\text{H}}$)†	6.0M	89.9$_{\pm.5}$	96.2$_{\pm.3}$	88.7$_{\pm2.9}$	66.5$_{\pm4.4}$	94.7$_{\pm.2}$	92.1$_{\pm.1}$	83.4$_{\pm1.1}$	91.0$_{\pm1.7}$	87.8
RoB$_{\text{large}}$ (Adpt$^{\text{H}}$)†	0.8M	90.3$_{\pm.3}$	96.3$_{\pm.5}$	87.7$_{\pm1.7}$	66.3$_{\pm2.0}$	94.7$_{\pm.2}$	91.5$_{\pm.1}$	72.9$_{\pm2.9}$	91.5$_{\pm.5}$	86.4
RoB$_{\text{large}}$ (LoRA)†	0.8M	**90.6**$_{\pm.2}$	96.2$_{\pm.5}$	**90.2**$_{\pm1.0}$	68.2$_{\pm1.9}$	**94.8**$_{\pm.3}$	91.6$_{\pm.2}$	**85.2**$_{\pm1.1}$	**92.3**$_{\pm.5}$	**88.6**
DeB$_{\text{XXL}}$ (FT)*	1500.0M	91.8	**97.2**	92.0	72.0	**96.0**	92.7	93.9	92.9	91.1
DeB$_{\text{XXL}}$ (LoRA)	4.7M	**91.9**$_{\pm.2}$	96.9$_{\pm.2}$	**92.6**$_{\pm.6}$	**72.4**$_{\pm1.1}$	**96.0**$_{\pm.1}$	**92.9**$_{\pm.1}$	**94.9**$_{\pm.4}$	**93.0**$_{\pm.2}$	**91.3**

Table 2: RoBERTa$_{\text{base}}$, RoBERTa$_{\text{large}}$, and DeBERTa$_{\text{XXL}}$ with different adaptation methods on the GLUE benchmark. We report the overall (matched and mismatched) accuracy for MNLI, Matthew's correlation for CoLA, Pearson correlation for STS-B, and accuracy for other tasks. Higher is better for all metrics. * indicates numbers published in prior works. † indicates runs configured in a setup similar to Houlsby et al. (2019) for a fair comparison.

Results (cont.)

Model & Method	# Trainable Parameters	E2E NLG Challenge BLEU	NIST	MET	ROUGE-L	CIDEr
GPT-2 M (FT)*	354.92M	68.2	8.62	46.2	71.0	2.47
GPT-2 M ($\text{Adapter}^{\text{L}}$)*	0.37M	66.3	8.41	45.0	69.8	2.40
GPT-2 M ($\text{Adapter}^{\text{L}}$)*	11.09M	68.9	8.71	46.1	71.3	2.47
GPT-2 M ($\text{Adapter}^{\text{H}}$)	11.09M	$67.3_{\pm.6}$	$8.50_{\pm.07}$	$46.0_{\pm.2}$	$70.7_{\pm.2}$	$2.44_{\pm.01}$
GPT-2 M (FT^{Top2})*	25.19M	68.1	8.59	46.0	70.8	2.41
GPT-2 M (PreLayer)*	0.35M	69.7	8.81	46.1	71.4	2.49
GPT-2 M (LoRA)	0.35M	$\mathbf{70.4}_{\pm.1}$	$\mathbf{8.85}_{\pm.02}$	$\mathbf{46.8}_{\pm.2}$	$\mathbf{71.8}_{\pm.1}$	$\mathbf{2.53}_{\pm.02}$
GPT-2 L (FT)*	774.03M	68.5	8.78	46.0	69.9	2.45
GPT-2 L ($\text{Adapter}^{\text{L}}$)	0.88M	$69.1_{\pm.1}$	$8.68_{\pm.03}$	$46.3_{\pm.0}$	$71.4_{\pm.2}$	$\mathbf{2.49}_{\pm.0}$
GPT-2 L ($\text{Adapter}^{\text{L}}$)	23.00M	$68.9_{\pm.3}$	$8.70_{\pm.04}$	$46.1_{\pm.1}$	$71.3_{\pm.2}$	$2.45_{\pm.02}$
GPT-2 L (PreLayer)*	0.77M	70.3	8.85	46.2	71.7	2.47
GPT-2 L (LoRA)	0.77M	$\mathbf{70.4}_{\pm.1}$	$\mathbf{8.89}_{\pm.02}$	$\mathbf{46.8}_{\pm.2}$	$\mathbf{72.0}_{\pm.2}$	$2.47_{\pm.02}$

Table 3: GPT-2 medium (M) and large (L) with different adaptation methods on the E2E NLG Challenge. For all metrics, higher is better. LoRA outperforms several baselines with comparable or fewer trainable parameters. Confidence intervals are shown for experiments we ran. * indicates numbers published in prior works.

Results (cont.)

Model&Method	# Trainable Parameters	WikiSQL Acc. (%)	MNLI-m Acc. (%)	SAMSum R1/R2/RL
GPT-3 (FT)	175,255.8M	**73.8**	89.5	52.0/28.0/44.5
GPT-3 (BitFit)	14.2M	71.3	91.0	51.3/27.4/43.5
GPT-3 (PreEmbed)	3.2M	63.1	88.6	48.3/24.2/40.5
GPT-3 (PreLayer)	20.2M	70.1	89.5	50.8/27.3/43.5
GPT-3 ($\text{Adapter}^{\text{H}}$)	7.1M	71.9	89.8	53.0/28.9/44.8
GPT-3 ($\text{Adapter}^{\text{H}}$)	40.1M	73.2	**91.5**	53.2/29.0/45.1
GPT-3 (LoRA)	4.7M	73.4	**91.7**	**53.8/29.8/45.9**
GPT-3 (LoRA)	37.7M	**74.0**	**91.6**	53.4/29.2/45.1

Table 4: Performance of different adaptation methods on GPT-3 175B. We report the logical form validation accuracy on WikiSQL, validation accuracy on MultiNLI-matched, and Rouge-1/2/L on SAMSum. LoRA performs better than prior approaches, including full fine-tuning. The results on WikiSQL have a fluctuation around $\pm 0.5\%$, MNLI-m around $\pm 0.1\%$, and SAMSum around $\pm 0.2/\pm 0.2/\pm 0.1$ for the three metrics.

Comparison to other PEFTs

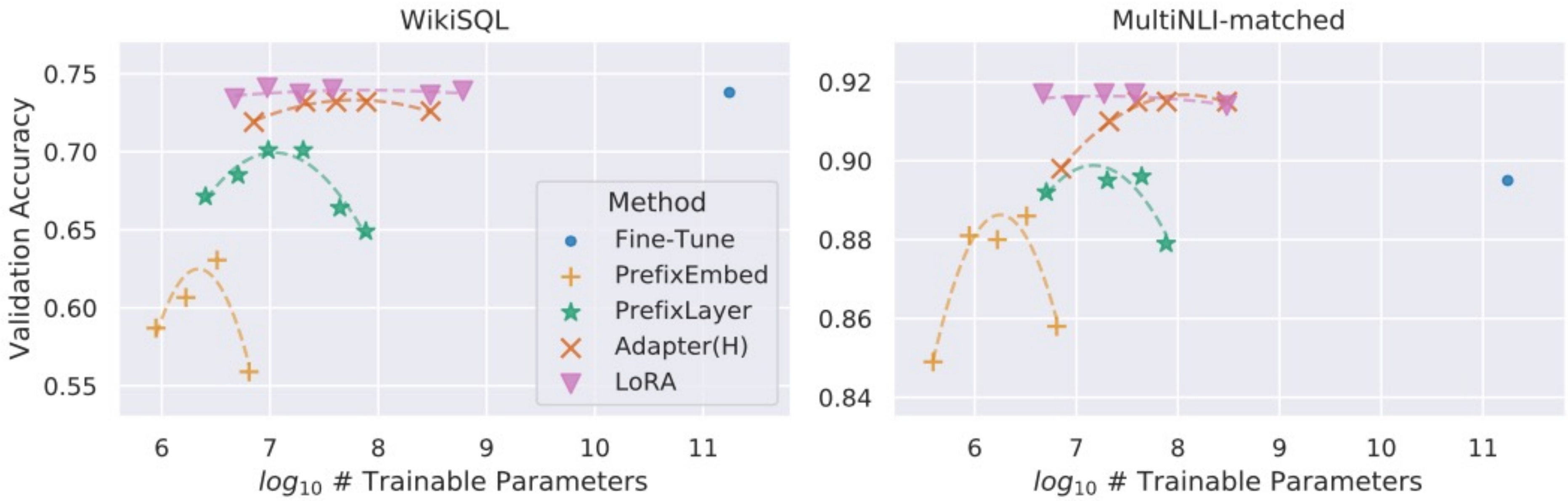


Figure 2: GPT-3 175B validation accuracy vs. number of trainable parameters of several adaptation methods on WikiSQL and MNLI-matched. LoRA exhibits better scalability and task performance. See Section F.2 for more details on the plotted data points.

Why r=1 works well in practice?

- Let $A_{r=8}$ and $A_{r=64}$ be the learned matrices for r = 8, and 64.
- Do they extract similar features from the token embeddings?
- How to measure this?
- Each A can be considered as a subspace.
- Find how similar these two subspaces are?

Why r=1 works well in practice? (cont.)

$$A = V\Sigma U$$

$$\Longrightarrow A = \sum_{i=1}^{r} \sigma_i v_i\, u_i^T$$

$$\Longrightarrow\ Ax = \sum_{i=1}^{r} \sigma_i v_i\, u_i^T x$$

$$\Longrightarrow Ax = \sum_{i=1}^{r} \sigma_i \langle u_i, x\rangle v_i$$

Pick highest σ_i , compare the corresponding u_i's in two A's

Why r=1 works well in practice? (cont.)

- Grassmann distance:

$$\phi(A_{r=8}, A_{r=64}, i, j) = \frac{||U_{A_{r=8}}^{i\top} U_{A_{r=64}}^{j}||_F^2}{\min(i, j)} \in [0, 1]$$

Why r=1 works well in practice? (cont.)

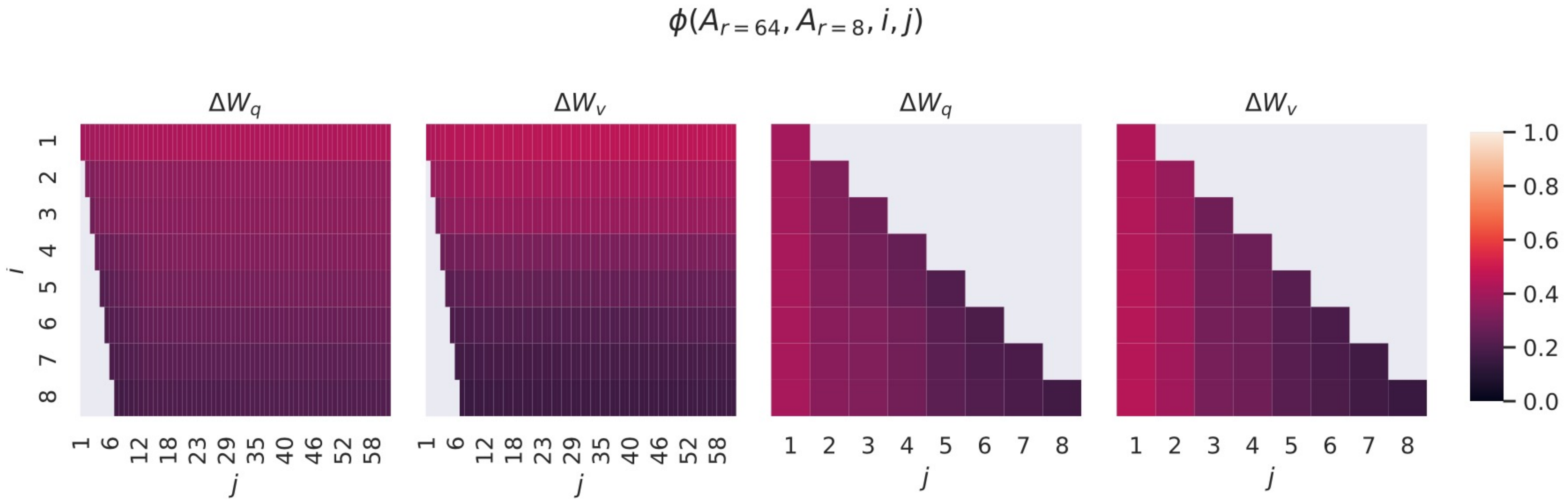


Figure 3: Subspace similarity between column vectors of $A_{r=8}$ and $A_{r=64}$ for both ΔW_q and ΔW_v. The third and the fourth figures zoom in on the lower-left triangle in the first two figures. The top directions in $r = 8$ are included in $r = 64$, and vice versa.

ΔW only amplifies directions that are not emphasized in W

	$r=4$			$r=64$		
	ΔW_q	W_q	Random	ΔW_q	W_q	Random
$\|\|U^\top W_q V^\top\|\|_F =$	0.32	21.67	0.02	1.90	37.71	0.33
$\|\|W_q\|\|_F = 61.95$	$\|\|\Delta W_q\|\|_F = 6.91$			$\|\|\Delta W_q\|\|_F = 3.57$		

Table 7: The Frobenius norm of $U^\top W_q V^\top$ where U and V are the left/right top r singular vector directions of either (1) ΔW_q, (2) W_q, or (3) a random matrix. The weight matrices are taken from the 48th layer of GPT-3.

Unified View of PEFT methods (cont.)

- Adapter

$$\boldsymbol{h} \leftarrow \boldsymbol{h} + f(\boldsymbol{h}\boldsymbol{W}_{\text{down}})\boldsymbol{W}_{\text{up}}$$

Unified View of PEFT methods (cont.)

- Prefix tuning

$$
\begin{aligned}
\text{head} &= \text{Attn}(\boldsymbol{x}\boldsymbol{W}_q, \text{concat}(\boldsymbol{P}_k, \boldsymbol{C}\boldsymbol{W}_k), \text{concat}(\boldsymbol{P}_v, \boldsymbol{C}\boldsymbol{W}_v)) \\
&= \text{softmax}\big(\boldsymbol{x}\boldsymbol{W}_q \text{concat}(\boldsymbol{P}_k, \boldsymbol{C}\boldsymbol{W}_k)^\top\big) \begin{bmatrix} \boldsymbol{P}_v \\ \boldsymbol{C}\boldsymbol{W}_v \end{bmatrix} \\
&= (1-\lambda(\boldsymbol{x}))\text{softmax}(\boldsymbol{x}\boldsymbol{W}_q\boldsymbol{W}_k^\top\boldsymbol{C}^\top)\boldsymbol{C}\boldsymbol{W}_v + \lambda(\boldsymbol{x})\text{softmax}(x\boldsymbol{W}_q\boldsymbol{P}_k^\top)\boldsymbol{P}_v \\
&= (1-\lambda(\boldsymbol{x}))\underbrace{\text{Attn}(\boldsymbol{x}\boldsymbol{W}_q, \boldsymbol{C}\boldsymbol{W}_k, \boldsymbol{C}\boldsymbol{W}_v)}_{\text{standard attention}} + \lambda(\boldsymbol{x})\underbrace{\text{Attn}(\boldsymbol{x}\boldsymbol{W}_q, \boldsymbol{P}_k, \boldsymbol{P}_v)}_{\text{independent of } \boldsymbol{C}}
\end{aligned}
$$

$$
\lambda(\boldsymbol{x}) = \frac{\sum_i \exp(\boldsymbol{x}\boldsymbol{W}_q\boldsymbol{P}_k^\top)_i}{\sum_i \exp(\boldsymbol{x}\boldsymbol{W}_q\boldsymbol{P}_k^\top)_i + \sum_j \exp(\boldsymbol{x}\boldsymbol{W}_q\boldsymbol{W}_k^\top\boldsymbol{C}^\top)_j}
$$

$$
\boldsymbol{h} \leftarrow (1-\lambda(\boldsymbol{x}))\boldsymbol{h} + \lambda(\boldsymbol{x})\Delta\boldsymbol{h}, \quad \Delta\boldsymbol{h} := \text{softmax}(\boldsymbol{x}\boldsymbol{W}_q\boldsymbol{P}_k^\top)\boldsymbol{P}_v
$$

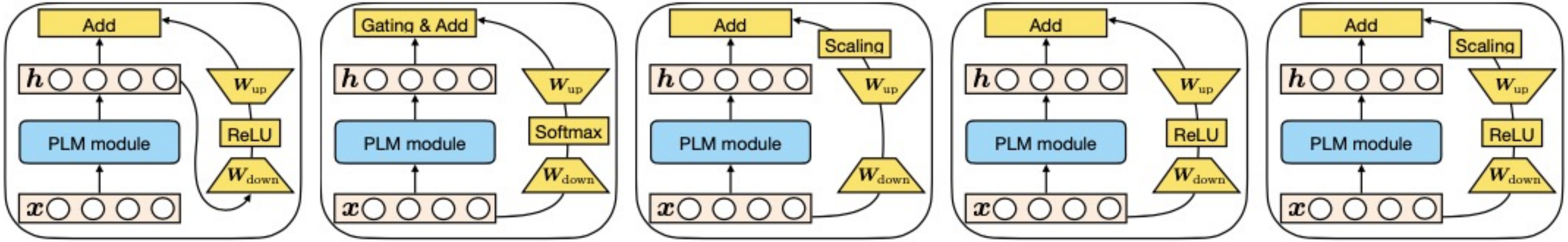
$$
\boldsymbol{h} \leftarrow (1-\lambda(\boldsymbol{x}))\boldsymbol{h} + \lambda(\boldsymbol{x})f(\boldsymbol{x}\boldsymbol{W}_1)\boldsymbol{W}_2
$$

Unified View of PEFT methods (cont.)

Table 1: Parameter-efficient tuning methods decomposed along the defined design dimensions. Here, for clarity, we directly write the adapter nonlinear function as ReLU which is commonly used. The bottom part of the table exemplifies new variants by transferring design choices of existing approaches.

Method	$\Delta\boldsymbol{h}$ functional form	insertion form	modified representation	composition function
		Existing Methods		
Prefix Tuning	$\text{softmax}(\boldsymbol{x}\boldsymbol{W}_q\boldsymbol{P}_k^\top)\boldsymbol{P}_v$	parallel	head attn	$\boldsymbol{h} \leftarrow (1-\lambda)\boldsymbol{h} + \lambda\Delta\boldsymbol{h}$
Adapter	$\text{ReLU}(\boldsymbol{h}\boldsymbol{W}_{\text{down}})\boldsymbol{W}_{\text{up}}$	sequential	ffn/attn	$\boldsymbol{h} \leftarrow \boldsymbol{h} + \Delta\boldsymbol{h}$
LoRA	$\boldsymbol{x}\boldsymbol{W}_{\text{down}}\boldsymbol{W}_{\text{up}}$	parallel	attn key/val	$\boldsymbol{h} \leftarrow \boldsymbol{h} + s \cdot \Delta\boldsymbol{h}$
		Proposed Variants		
Parallel adapter	$\text{ReLU}(\boldsymbol{h}\boldsymbol{W}_{\text{down}})\boldsymbol{W}_{\text{up}}$	parallel	ffn/attn	$\boldsymbol{h} \leftarrow \boldsymbol{h} + \Delta\boldsymbol{h}$
Muti-head parallel adapter	$\text{ReLU}(\boldsymbol{h}\boldsymbol{W}_{\text{down}})\boldsymbol{W}_{\text{up}}$	parallel	head attn	$\boldsymbol{h} \leftarrow \boldsymbol{h} + \Delta\boldsymbol{h}$
Scaled parallel adapter	$\text{ReLU}(\boldsymbol{h}\boldsymbol{W}_{\text{down}})\boldsymbol{W}_{\text{up}}$	parallel	ffn/attn	$\boldsymbol{h} \leftarrow \boldsymbol{h} + s \cdot \Delta\boldsymbol{h}$

Unified View of PEFT methods (cont.)



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

Remarks

- Prefix tuning can be thought of as a "parallel" computation to the PLM layer, whereas the typical adapter is "sequential" computation.
- Adapters are more flexible w.r.t. where they are inserted than prefix tuning
  - Adapters typically modify attention or FFN outputs, while prefix tuning only modifies the attention output of each head.
- Prefix tuning applies to each attention head, while adapters are always single-headed, which makes prefix tuning more expressive.