Parameter Efficient Fine-Tuning

Advanced NLP: Summer 2023

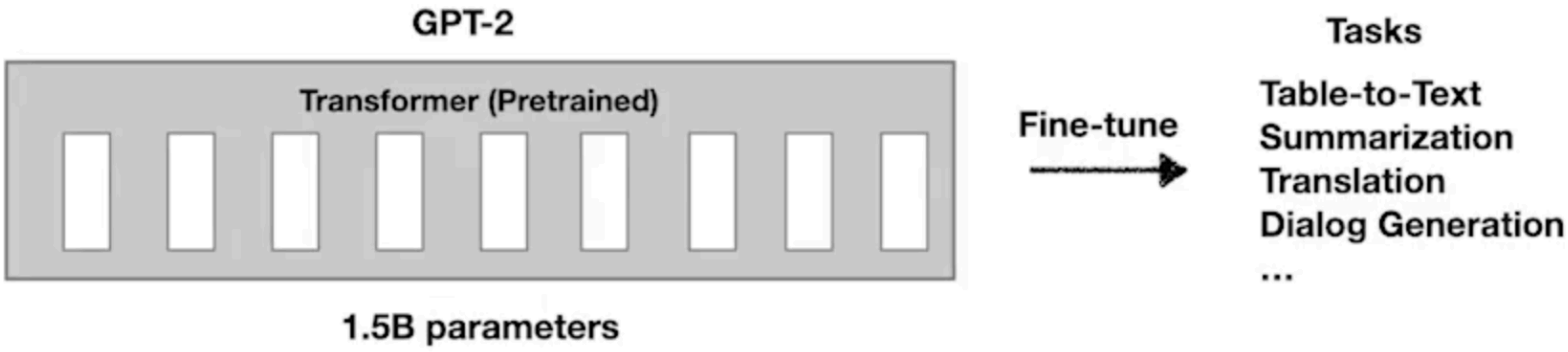
Anoop Sarkar

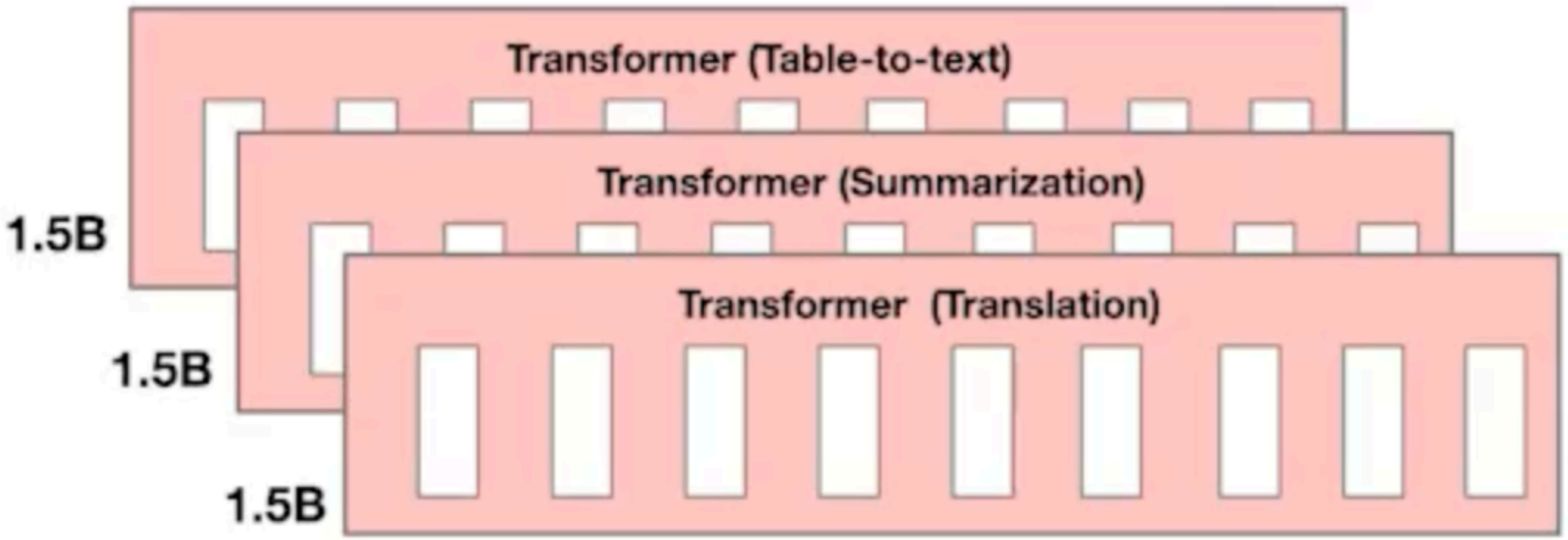
Prefix Tuning

Li and Liang, ACL 2021

https://aclanthology.org/2021.acl-long.353

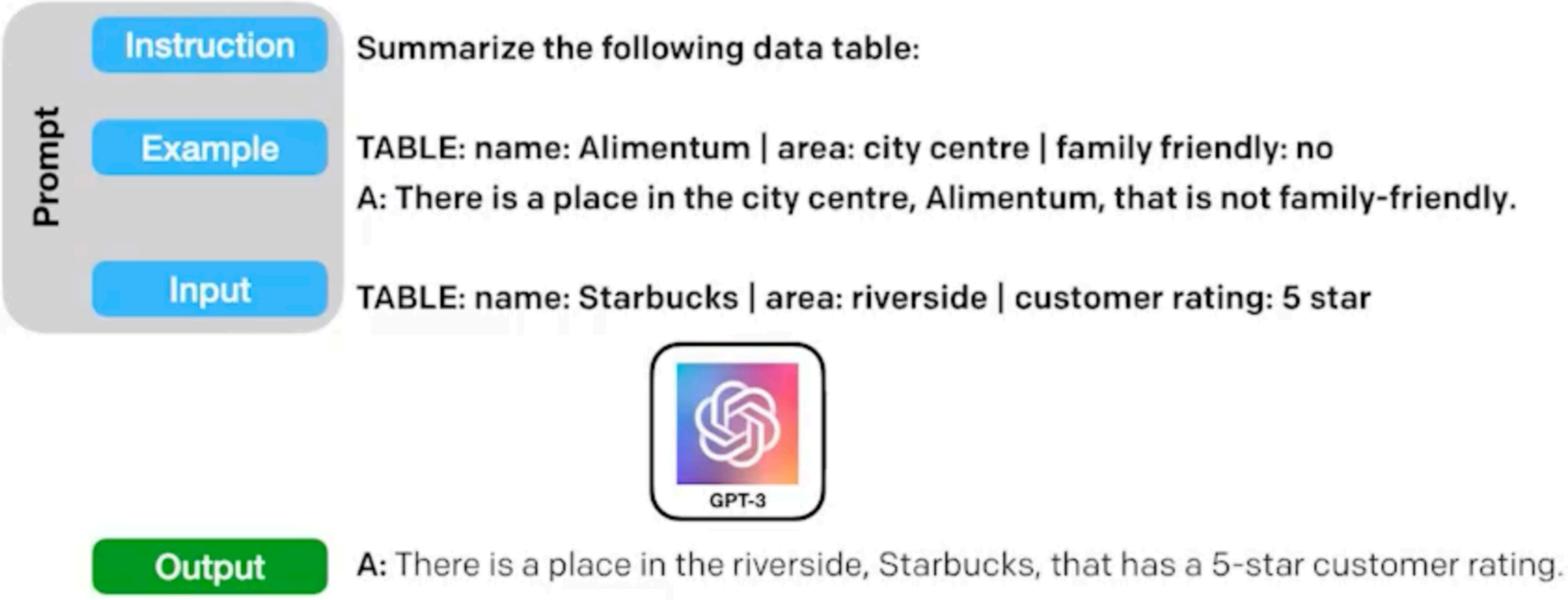
Why not just use fine-tuning





Each task requires a full model copy

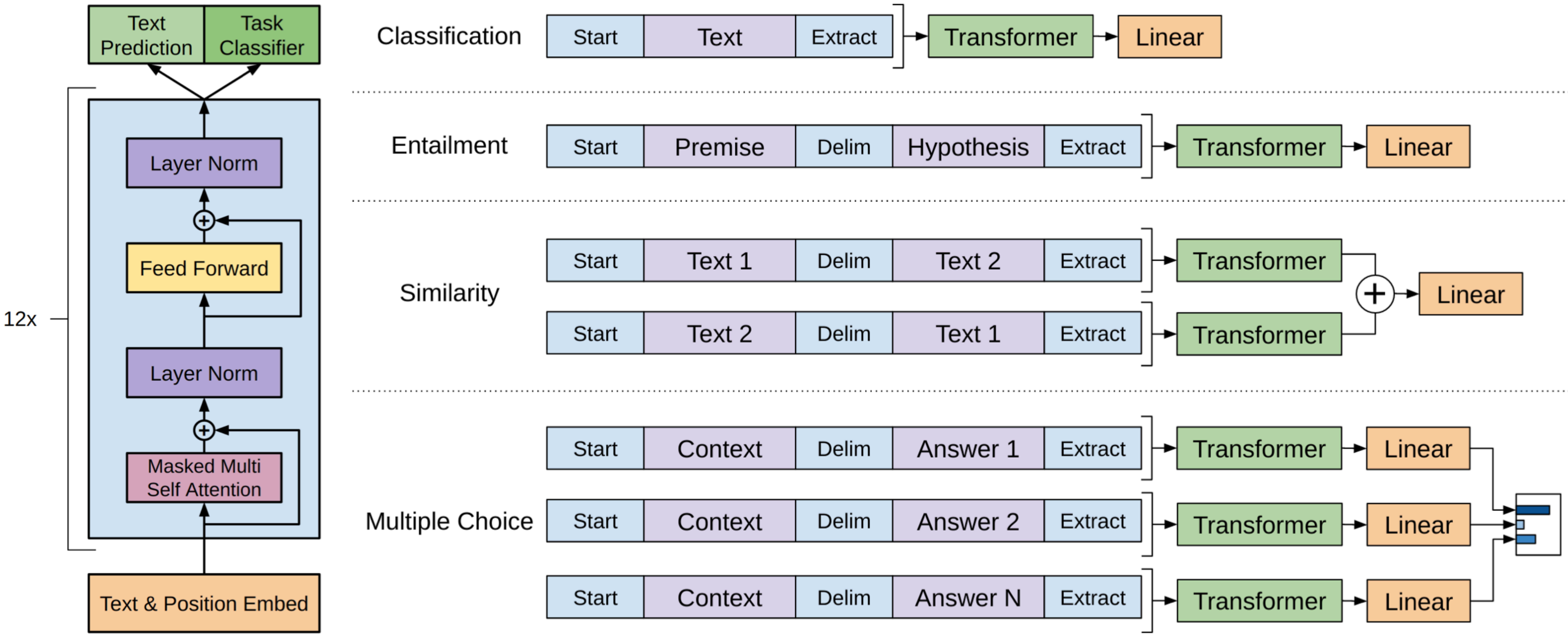
In-context learning using prompts



- No task specific fine-tuning
- Preserves the LM

- Cannot use large training set
- Manual prompts can be suboptimal
- Cannot be used with smaller LMs like GPT-2

In-context learning using prompts



https://arxiv.org/pdf/2005.14165.pdf

In-context learning using prompts

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

`Translate English to French:`	*task description*
`cheese =>`	*prompt*

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

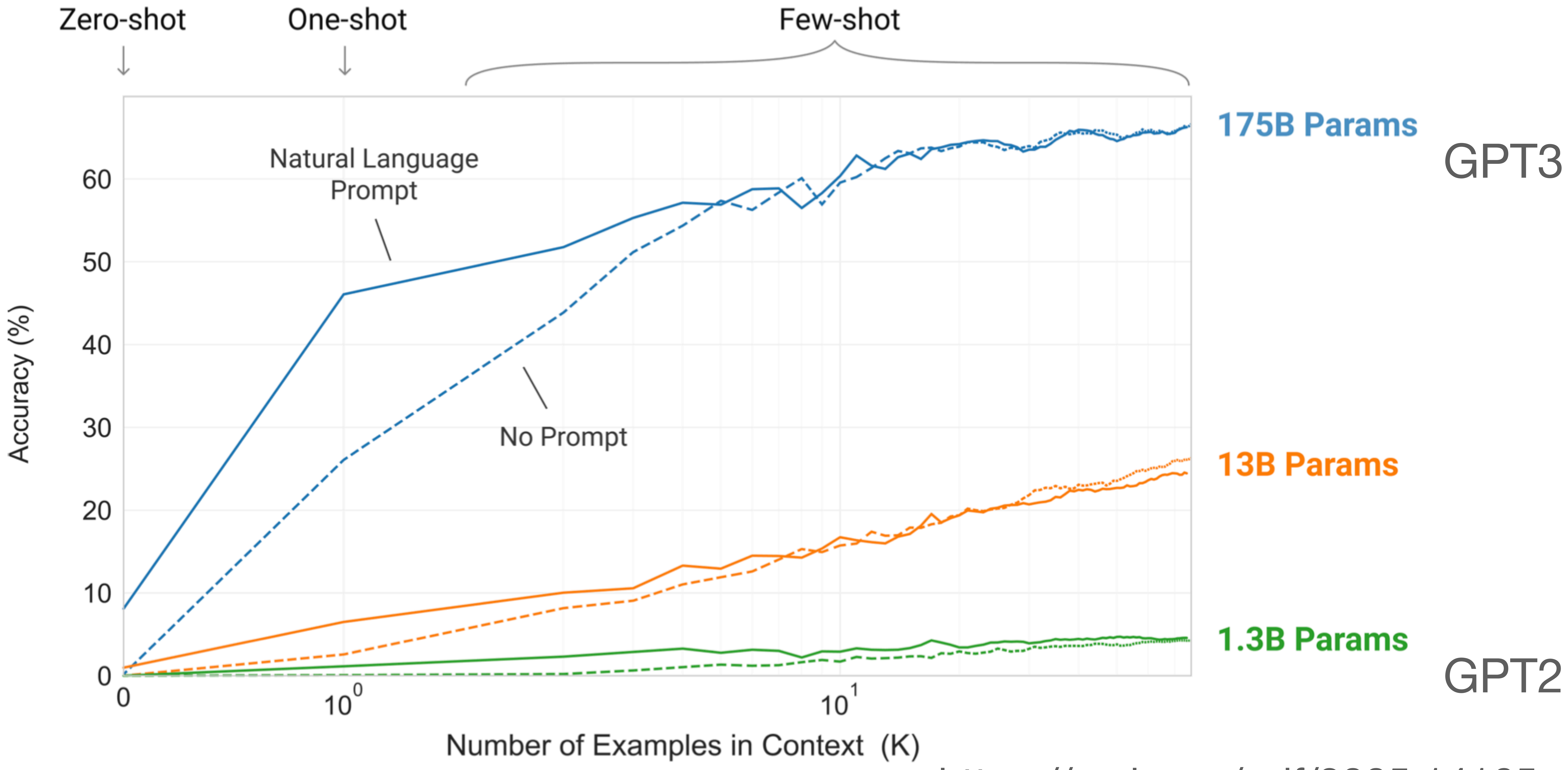
`Translate English to French:`	*task description*
`sea otter => loutre de mer`	*example*
`cheese =>`	*prompt*

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

`Translate English to French:`	*task description*
`sea otter => loutre de mer`	*examples*
`peppermint => menthe poivrée`	
`plush girafe => girafe peluche`	
`cheese =>`	*prompt*

In-context learning using prompts



https://arxiv.org/pdf/2005.14165.pdf

Prompt tuning: enabling smaller LMs

iPet: better prompts for each task improves accuracy for small LMs

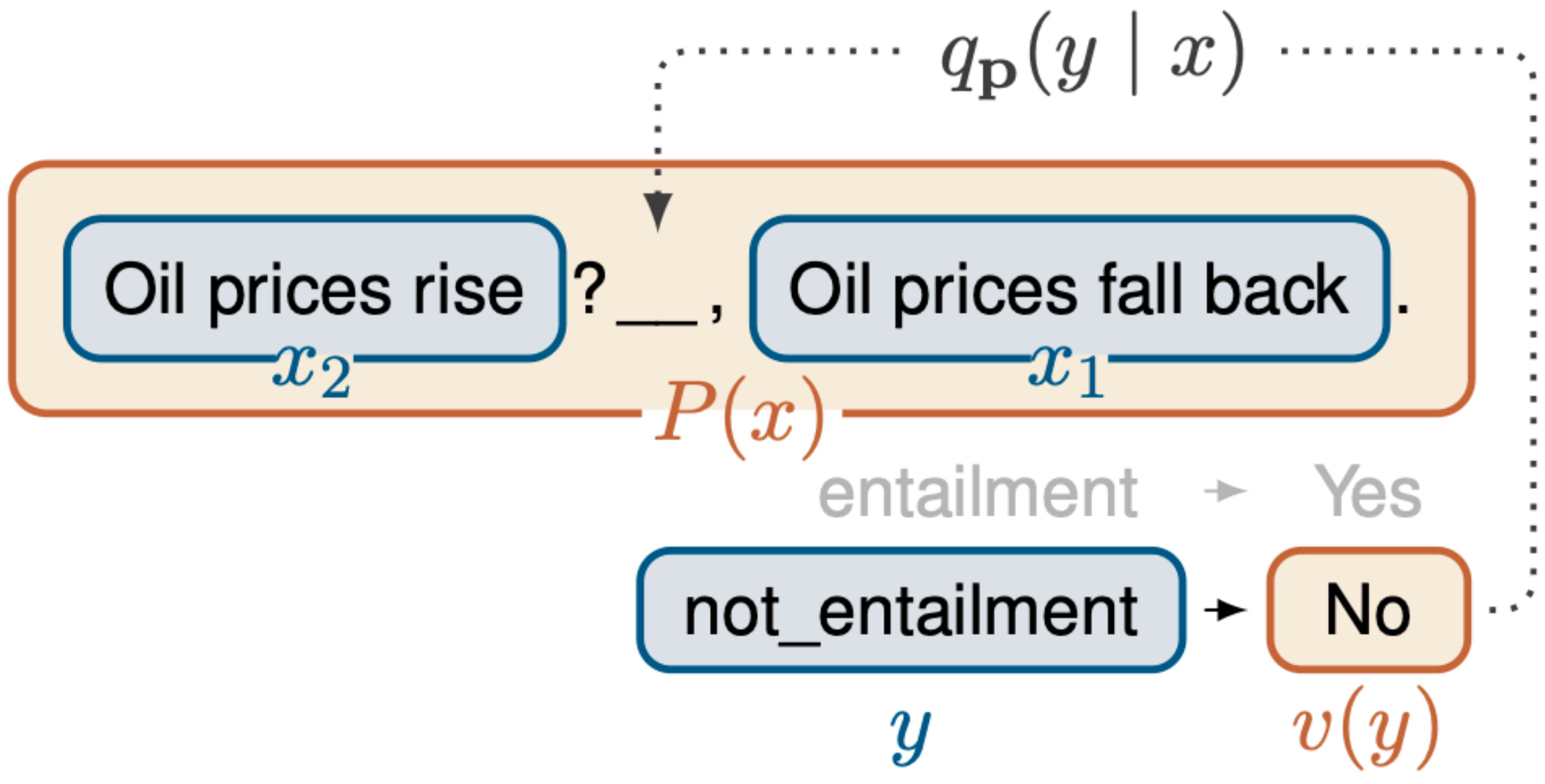


Figure 2: Application of a PVP $\mathbf{p} = (P, v)$ for recognizing textual entailment: An input $x = (x_1, x_2)$ is converted into a cloze question $P(x)$; $q_{\mathbf{p}}(y \mid x)$ for each y is derived from the probability of $v(y)$ being a plausible choice for the masked position.

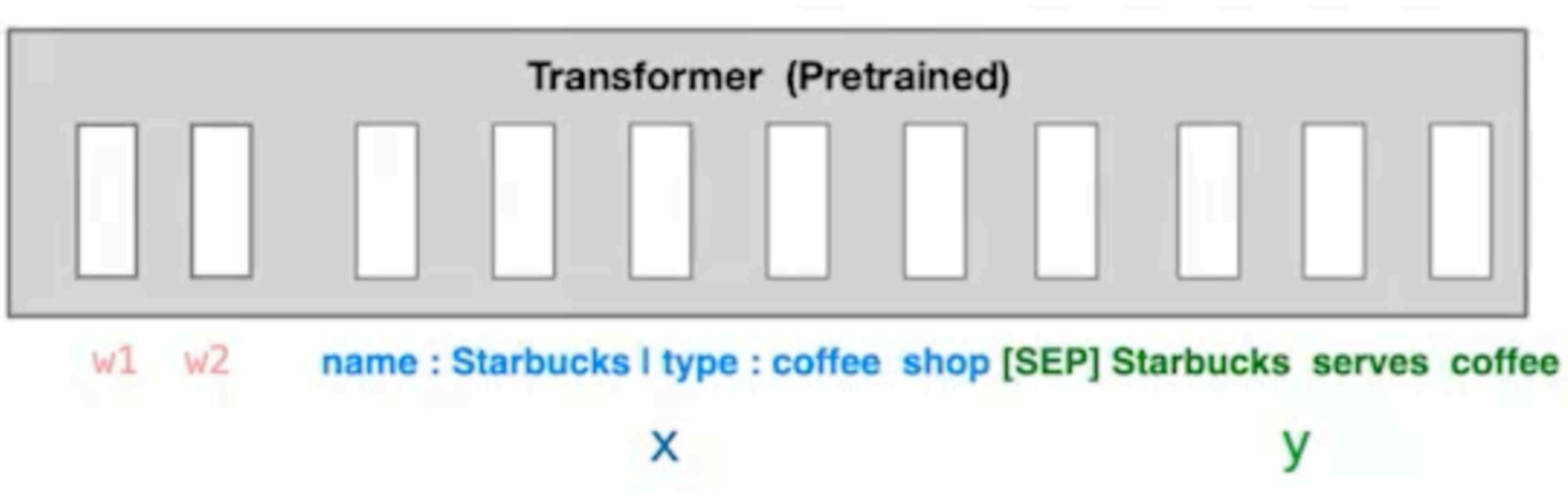
test	GPT-3	175,000	71.8	prompt
	PET	223	74.0	prompt FT
	iPET	223	**75.4**	prompt FT
	SotA	11,000	*89.3*	full FT

https://arxiv.org/pdf/2009.07118.pdf

Prefix Tuning

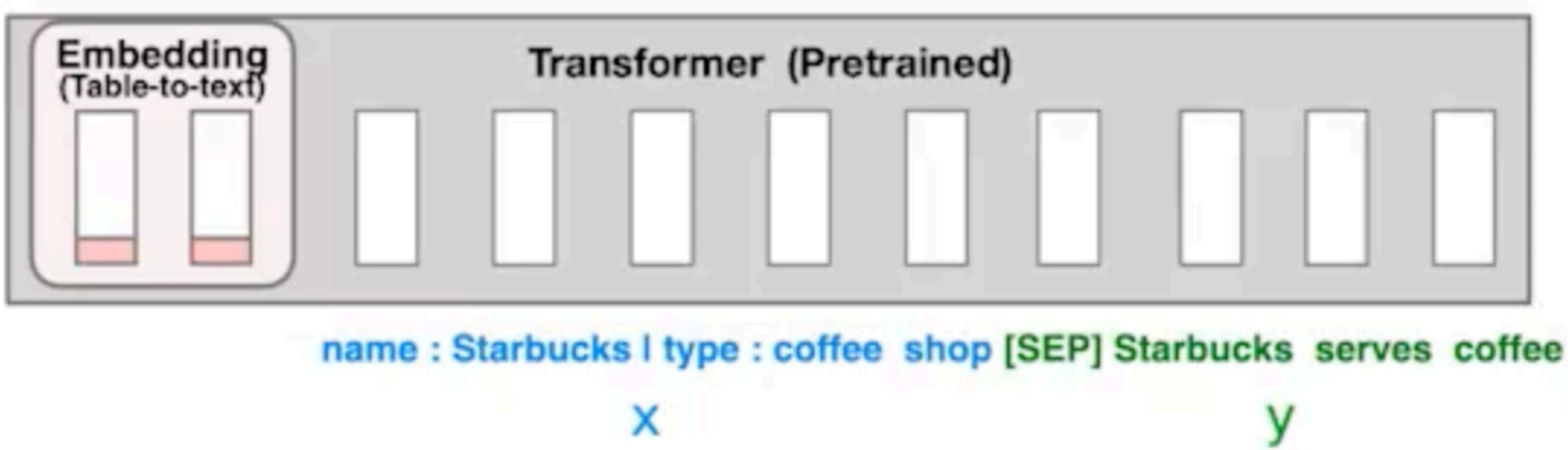
Intuition

- Learn a good instruction that can steer the LM to produce the right output
- Optimize finding actual words
- Involves discrete optimization which is challenging and not expressive



Prefix Tuning
Intuition

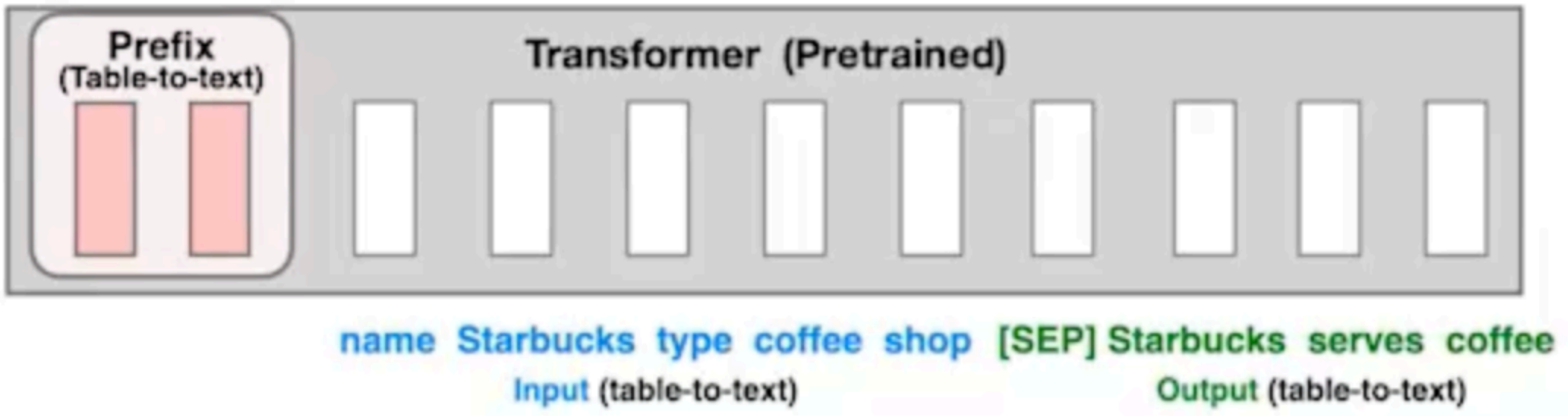
- Optimize the instruction as continuous word embeddings
- More expressive
- Limits the scope of the prompt to a input embeddings



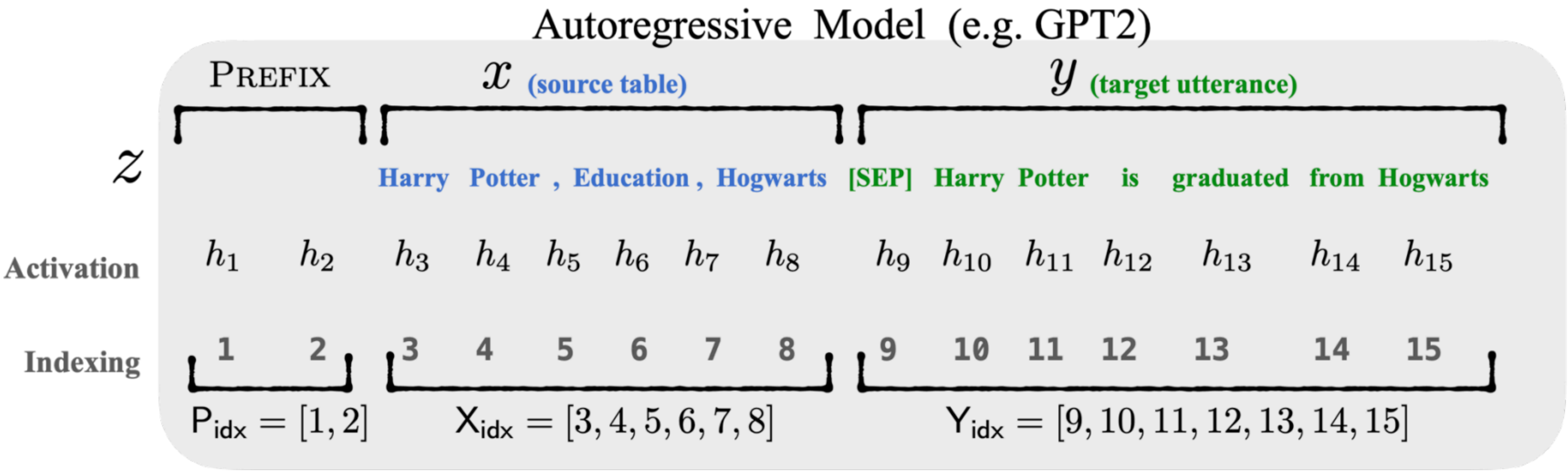
Prefix Tuning

Intuition

- Optimize the instruction as prefix activation for all layers in the instruction
- Very expressive
- All the layers of the prefix can be tuned to create the most expressive prompt

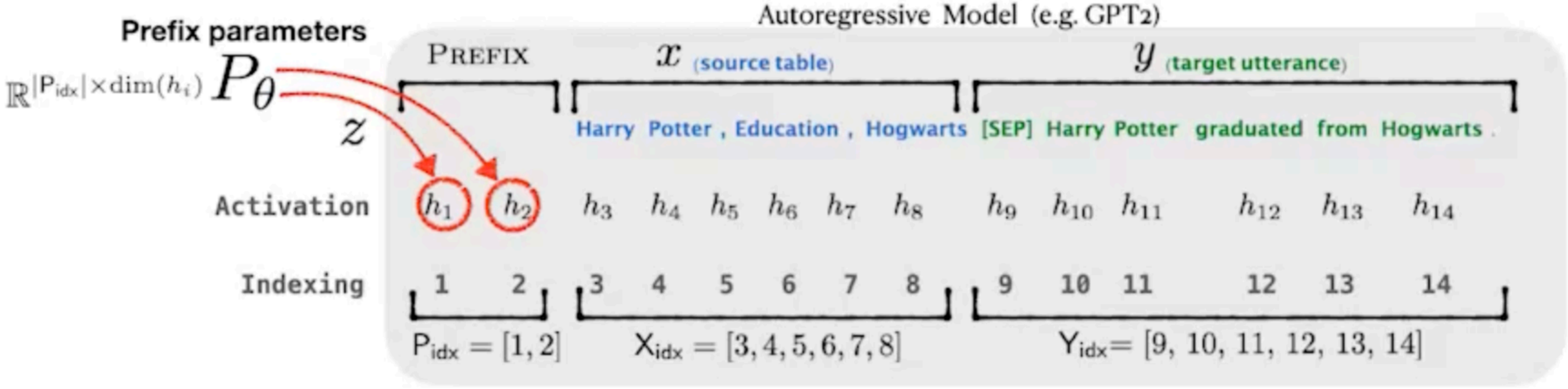


Prefix Tuning
Autoregressive Modelling



Prefix Tuning

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

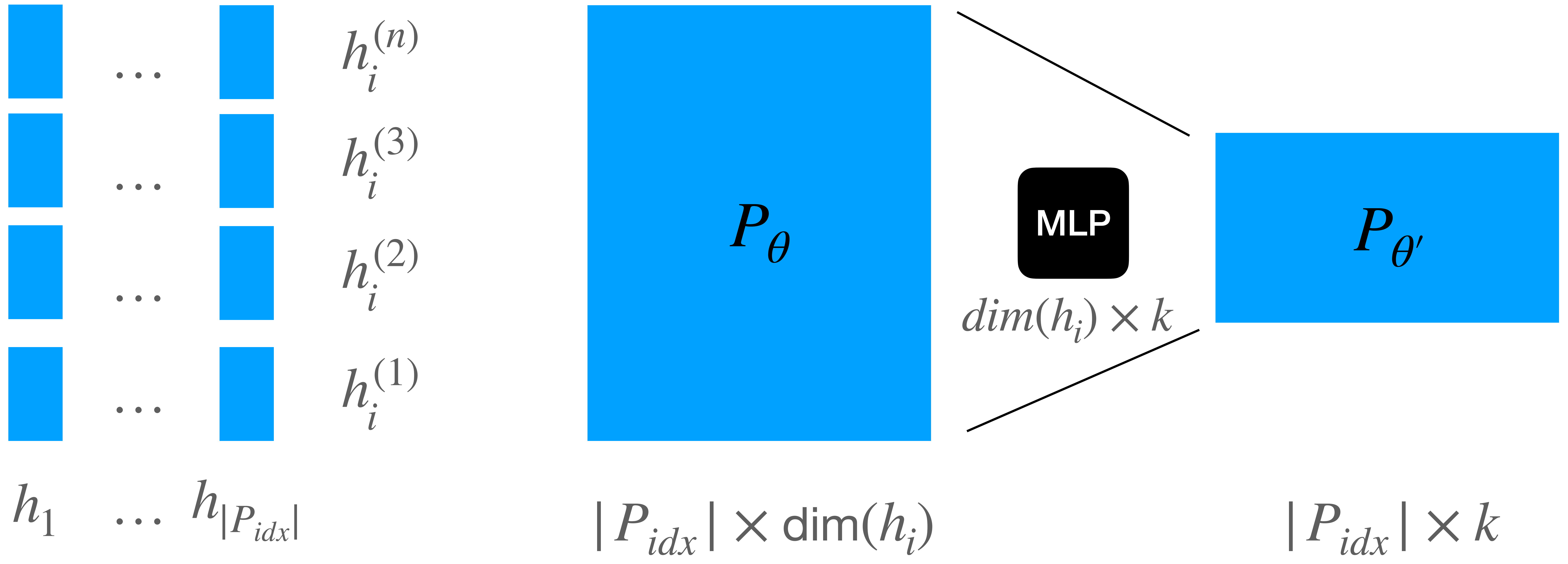


$$\max_\theta \log p_{\phi,\theta}(y \mid x) = \sum_{i \in \mathsf{Y}_{\text{idx}}} \log p_{\phi,\theta}(z_i \mid h_{<i})$$

freeze LM parameters ϕ
update prefix parameters θ

Prefix Re-parametrization

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



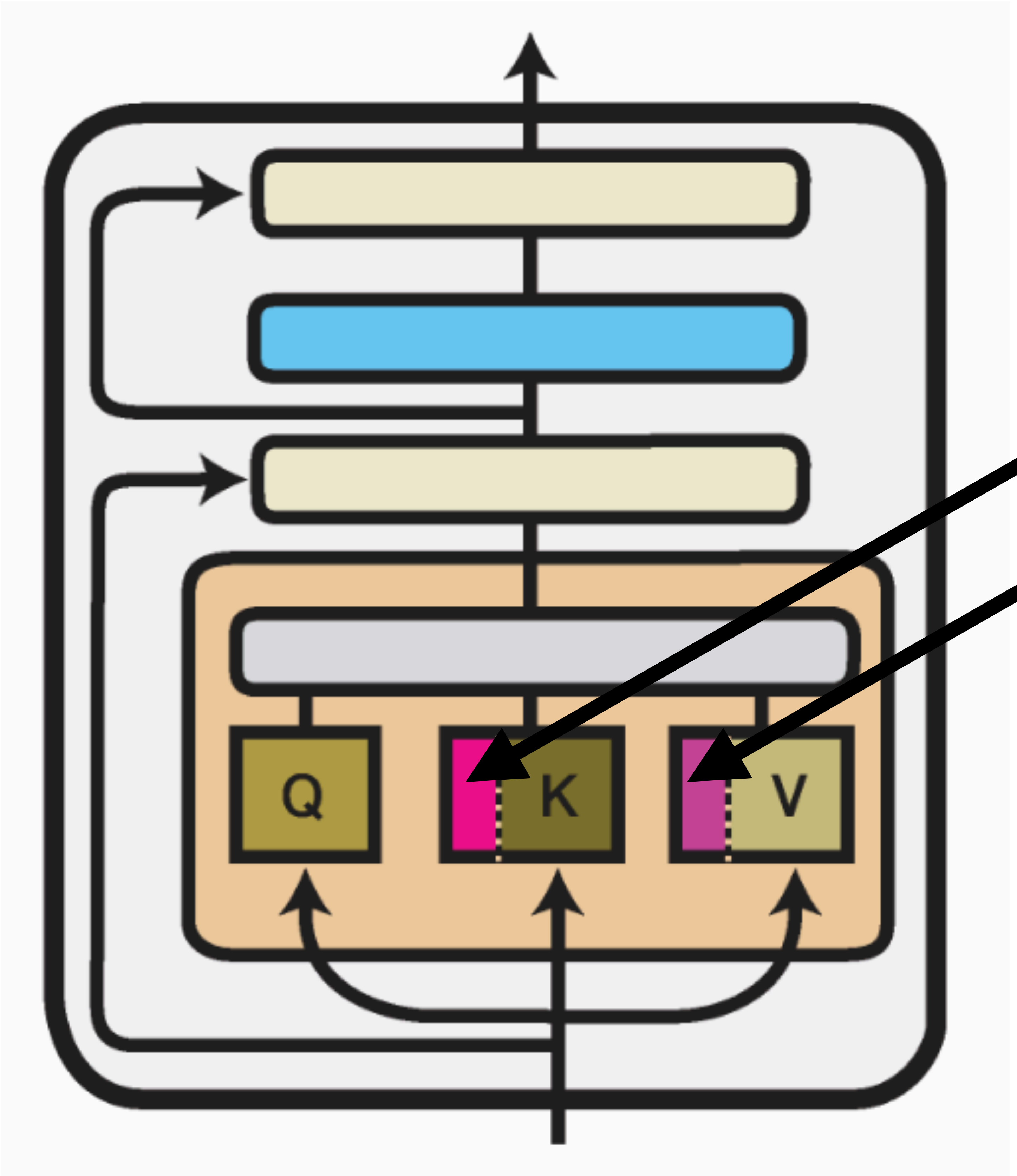
$P_\theta[i, :]$

$|P_{idx}| \times \text{dim}(h_i)$

Once training is complete we store only P_θ (throw away the MLP)

k is 512 for table-to-text and 800 for summarization

Effect of Prefix Tuning



Self-Attention over the added virtual prefix tokens

https://docs.adapterhub.ml/methods.html#prefix-tuning

Prefix Tuning

Vs. Finetuning

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.

* The number in the parenthesis refers to the training size.

Prefix Tuning

Extrapolation to unseen categories

Trained on 9 categories

Astronaut, University, Monument, Building, ComicsCharacter, Food, Airport, SportsTeam, City, and WrittenWork

extrapolates

Test on 5 unseen categories

Athlete, Artist, MeanOfTransportation, CelestialBody, Politician

x: [103_Colmore_Row | architect | John_Madin]
[John_Madin | birthPlace | Birmingham]
[Birmingham | leaderName | Andrew_Mitchell]

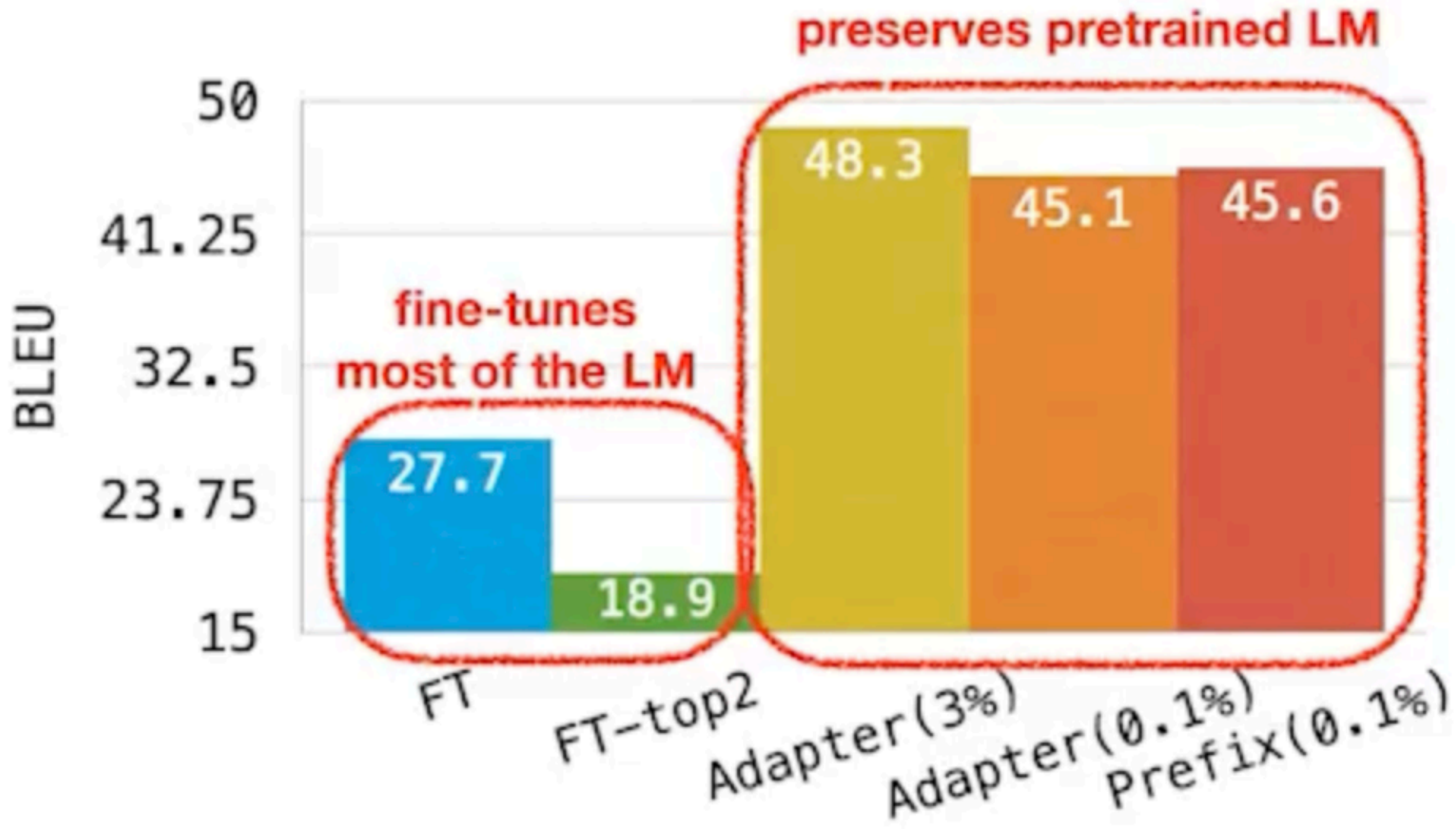
y: John Madin was born in Birmingham (with Andrew Mitchell as a key leader) and became an architect, designing 103 Colmore Row.

x: [Albennie_Jones | genre | Rhythm_and_blues]
[Albennie_Jones | birthPlace | Errata,_Mississippi]
[Rhythm_and_blues | derivative | Disco]

y: Albennie Jones, born in Errata, Mississippi, is a performer of rhythm and blues, of which disco is a derivative.

Prefix Tuning

Extrapolation to unseen categories



aka PaSTA

Parameter-Efficient Tuning with Special Token Adaptation

Xiaocong Yang[†,*], **James Y. Huang**[‡], **Wenxuan Zhou**[‡] and **Muhao Chen**[‡]
[†]Tsinghua University; [‡]University of Southern California
yangxc.18@sem.tsinghua.edu.cn;
{huangjam,zhouwenx,muhaoche}@usc.edu

https://aclanthology.org/2023.eacl-main.60.pdf

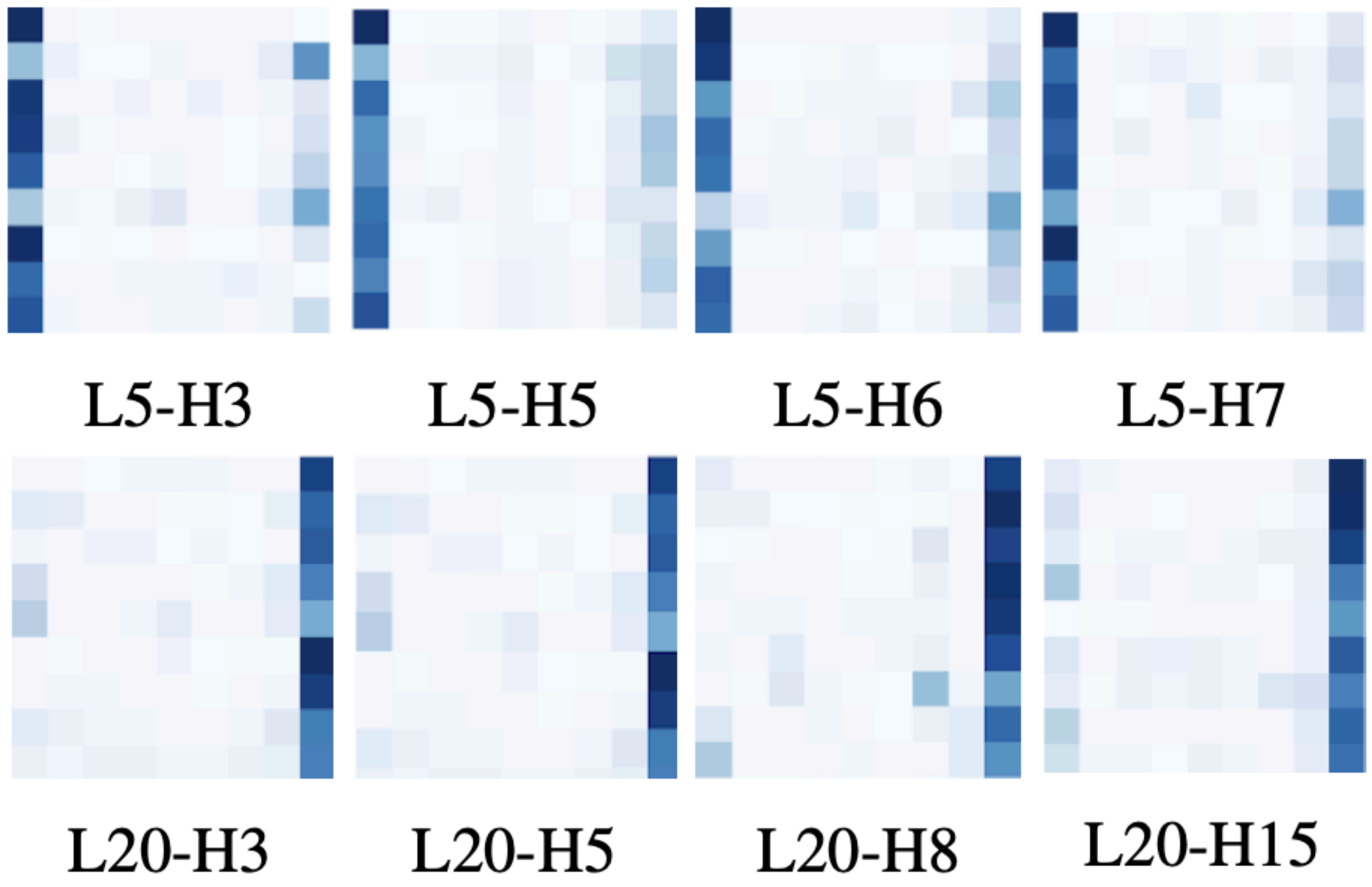


Figure 1: Examples of vertical attention heads in the 5-th and 20-th layer of BERT-large with a random sample from CoLA (Warstadt et al., 2019) as input. Heads in the first row and second row assign most of maximal attention weights to `[CLS]` and `[SEP]` respectively.

https://aclanthology.org/2023.eacl-main.60.pdf

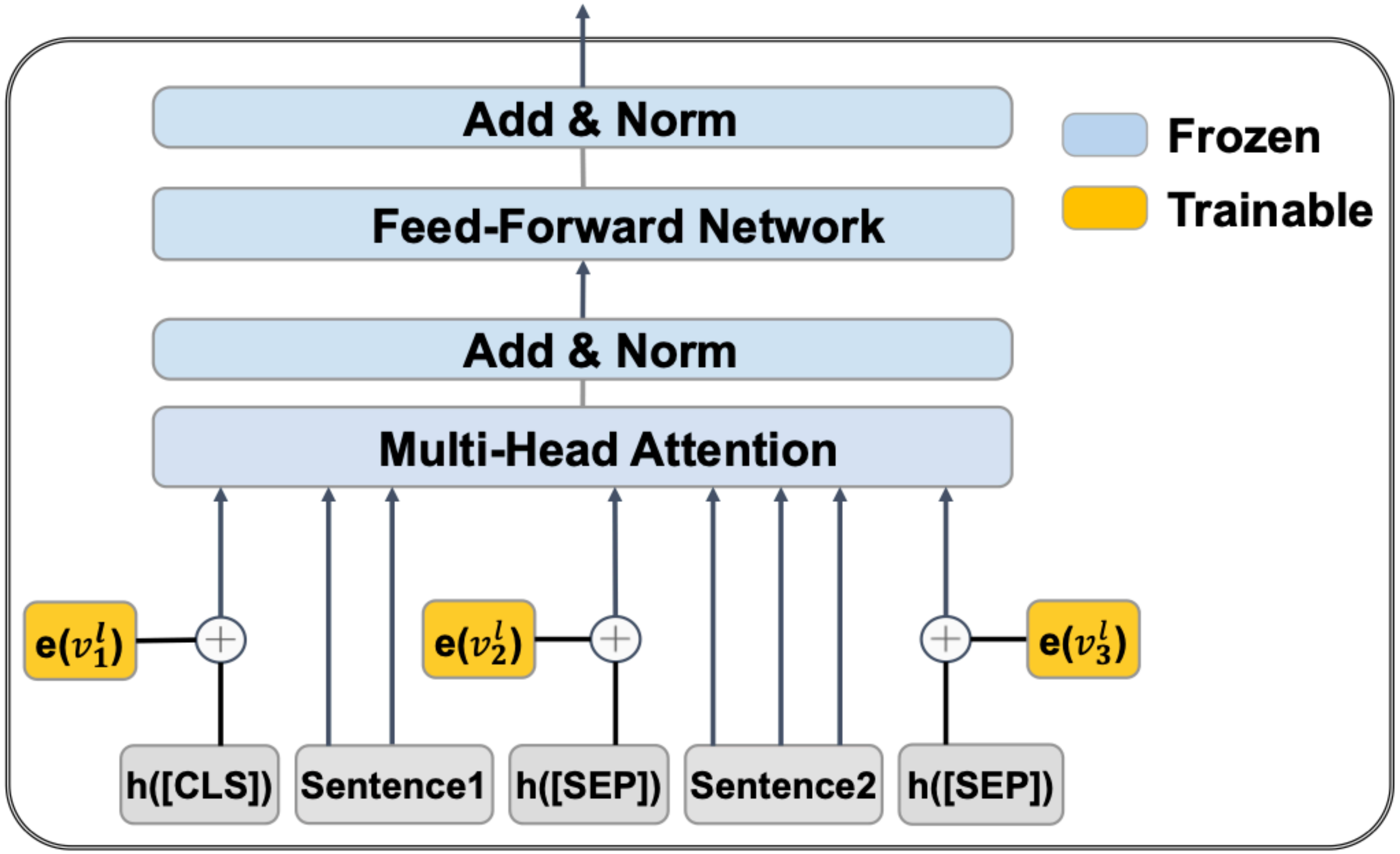


Figure 2: Architecture of PASTA layer in Transformer. Skip-connections in Transformers are not shown for brevity. At layer l we add a trainable vector $\mathbf{e}(\mathbf{v}_p^l) \in \mathbb{R}^d$ to the hidden representation of the p-th special token in the input sequence, and freeze the weights of the PLM.

https://aclanthology.org/2023.eacl-main.60.pdf

Results on GLUE with BERT-large

	%Param	RTE acc.	CoLA mcc.	STS-B Spearman	MRPC F1	SST-2 acc.	QNLI acc.	MNLI(m/mm) acc.	QQP F1	Avg.
Full Finetuning*	100%	70.1	60.5	86.5	89.3	**94.9**	92.7	**86.7/85.9**	**72.1**	**81.6**
Adapter**	3.6%	71.5	59.5	**86.9**	89.5	94.0	90.7	84.9/85.1	71.8	81.1
Diff-Prune†	0.5%	70.6	61.1	86.0	**89.7**	94.1	**93.3**	86.4/86.0	71.1	81.5
P-tuning v2	0.29%	70.1	60.1	86.8	88.0	94.6	92.3	85.3/84.9	70.6	81.0
BitFit‡	0.08%	**72.0**	59.7	85.5	88.9	94.2	92.0	84.5/84.8	70.5	80.9
PASTA	**0.015%-0.022%**	70.8	**62.3**	86.6	87.9	94.4	92.8	83.4/83.4	68.6	80.9

https://aclanthology.org/2023.eacl-main.60.pdf

Ablation study on GLUE and CoNLL-2003

	CoLA	RTE	MRPC	STS-B	CoNLL2003
PaSTA	**65.4**	**76.2**	89.7	**90.8**	**94.0**
- w/o [CLS]	58.8	72.6	91.4	90.2	93.7
- w/o [SEP]	64.5	71.1	91.9	90.3	93.7
- shared vector	64.7	74.7	**92.1**	90.0	93.9
- classifier only	36.5	54.2	81.5	64.9	77.4

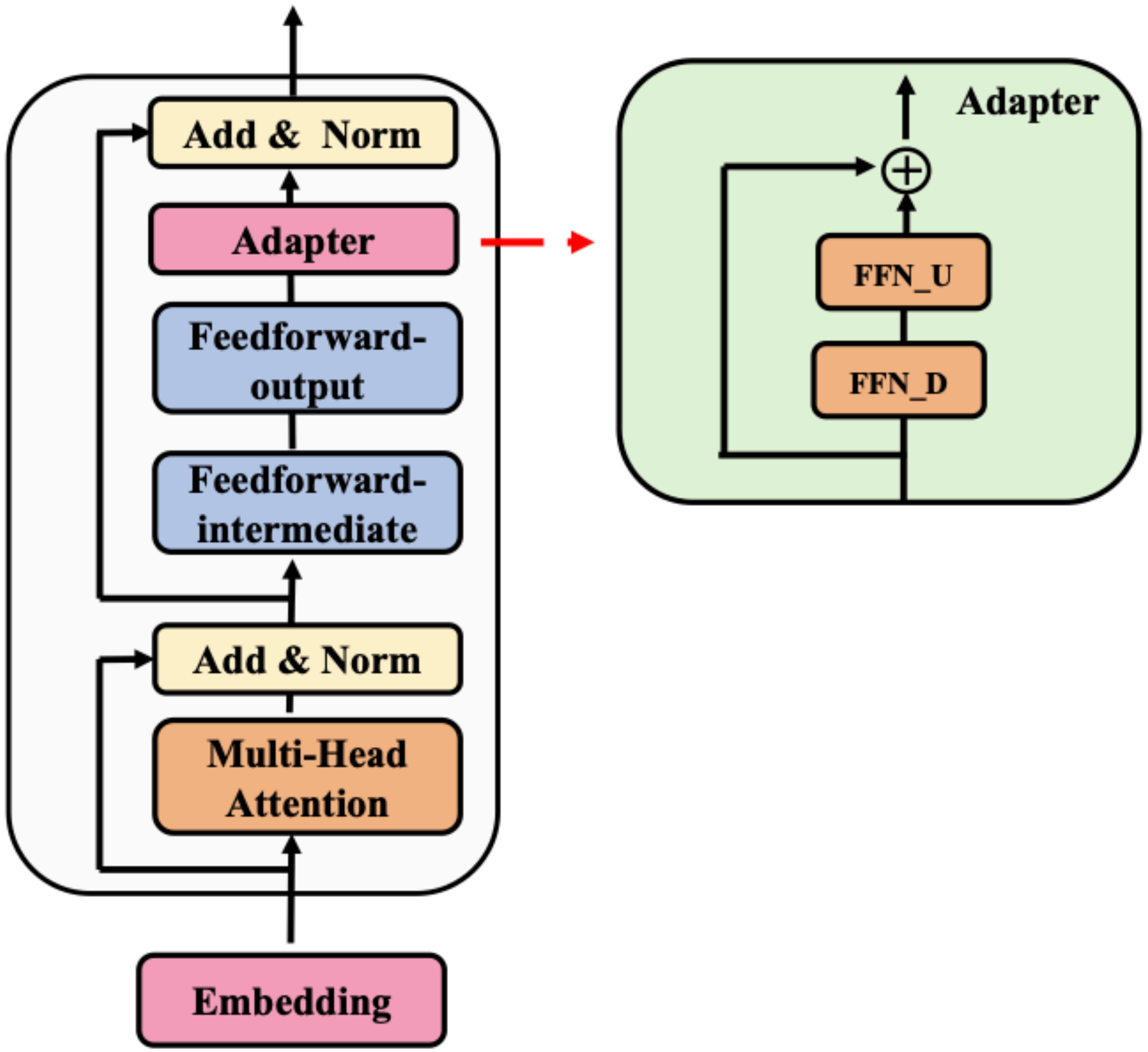
Table 4: Performance of ablation study with BERT-large on GLUE and CoNLL2003 development sets.

https://aclanthology.org/2023.eacl-main.60.pdf

Adapters

Bottleneck Adapters

- Given a hidden layer h^ℓ for layer ℓ in a Transformer layer (before Add & Norm)
- $h^\ell \leftarrow h^\ell + f(h^\ell \cdot W_{\text{down}}) \cdot W_{\text{up}}$
- W_{down} lowers the dimensionality from $\dim(h^\ell)$ down to k where $k << \dim(h^\ell)$
- W_{up} raises the dimensionality from k back up to $\dim(h^\ell)$
- f is a non-linear function (GeLU)
- $h^{\ell+1} = \text{Add+LN}(h^\ell)$



https://arxiv.org/abs/2205.12410

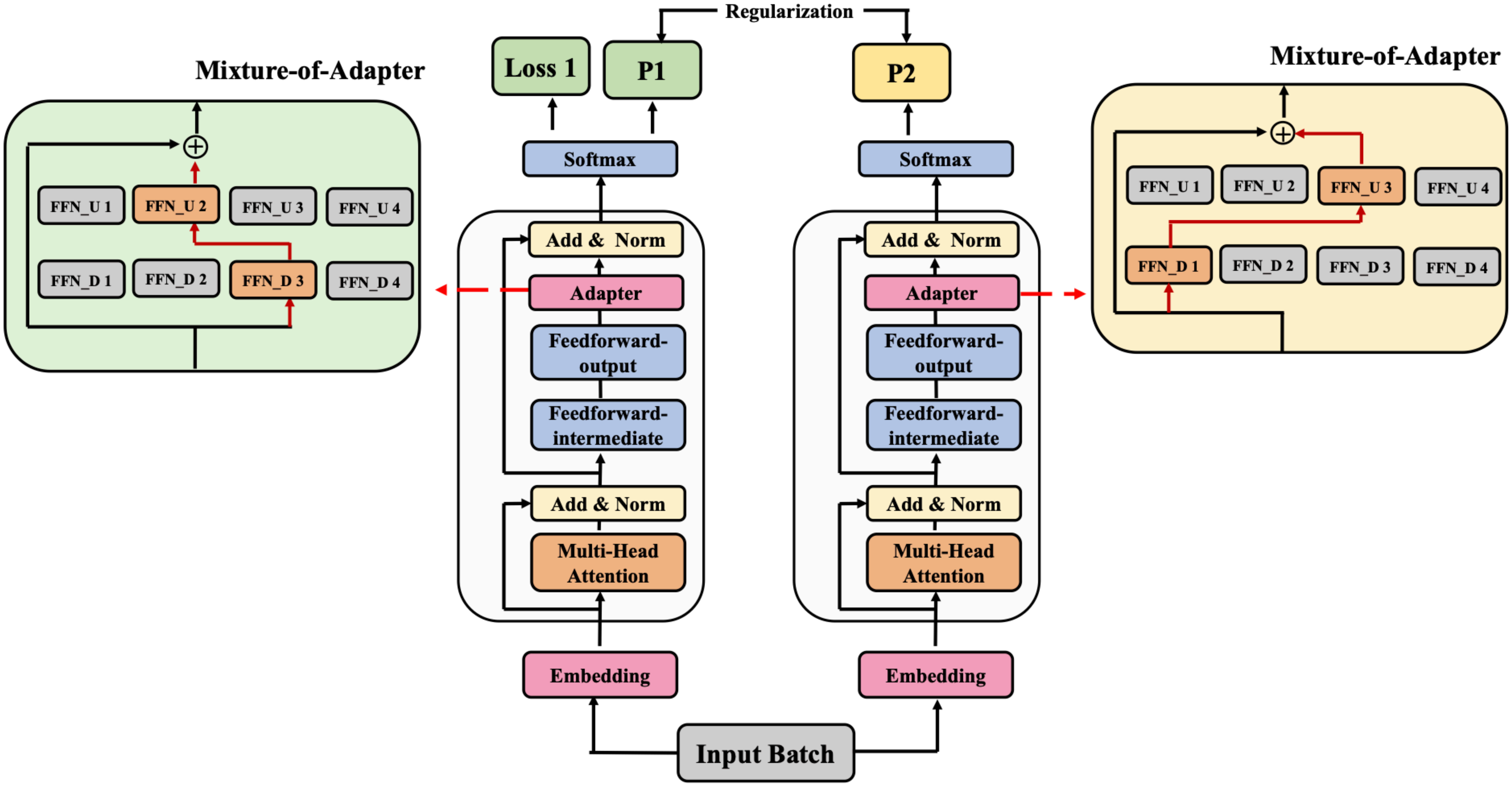
Also see: https://www.cs.huji.ac.il/labs/learning/Papers/allerton.pdf

Bottleneck Adapters

https://docs.adapterhub.ml/

Mixture of Adapters

https://arxiv.org/abs/2205.12410



Mixture of Adapters

Regularization loss

- For each layer ℓ use M different feed-forward networks for projecting down to k and for projecting up to $\dim(h^\ell)$
- $A_\ell = \{W^{\ell,j}_{\text{down}}, W^{\ell,k}_{\text{down}}\}$ and $B_\ell = \{W^{\ell,j}_{\text{up}}, W^{\ell,k}_{\text{up}}\}$
- where $j, k \in [0, M-1]$
- $h^\ell \leftarrow {\color{blue}h^\ell} + {\color{red}f(h^\ell \cdot W^{\ell,i}_{\text{down}}) \cdot W^{\ell,j}_{\text{up}}}$
- Pick i, j at random
- Pick i, j twice for each input batch.

Mixture of Adapters

Regularization loss

- Fine tuning loss: $\mathscr{L} = -\sum_{c=1}^{C} \delta(x, \hat{x}) \log \text{softmax}((z^{\mathscr{A}}(x))$
- where δ is 1 if the two arguments are equal
- $\hat{x}$ is the right answer for input x
- $z^{\mathscr{A}}(x)$ are the logits for the fine-tuning output softmax activation (using adapter $\mathscr{A}$

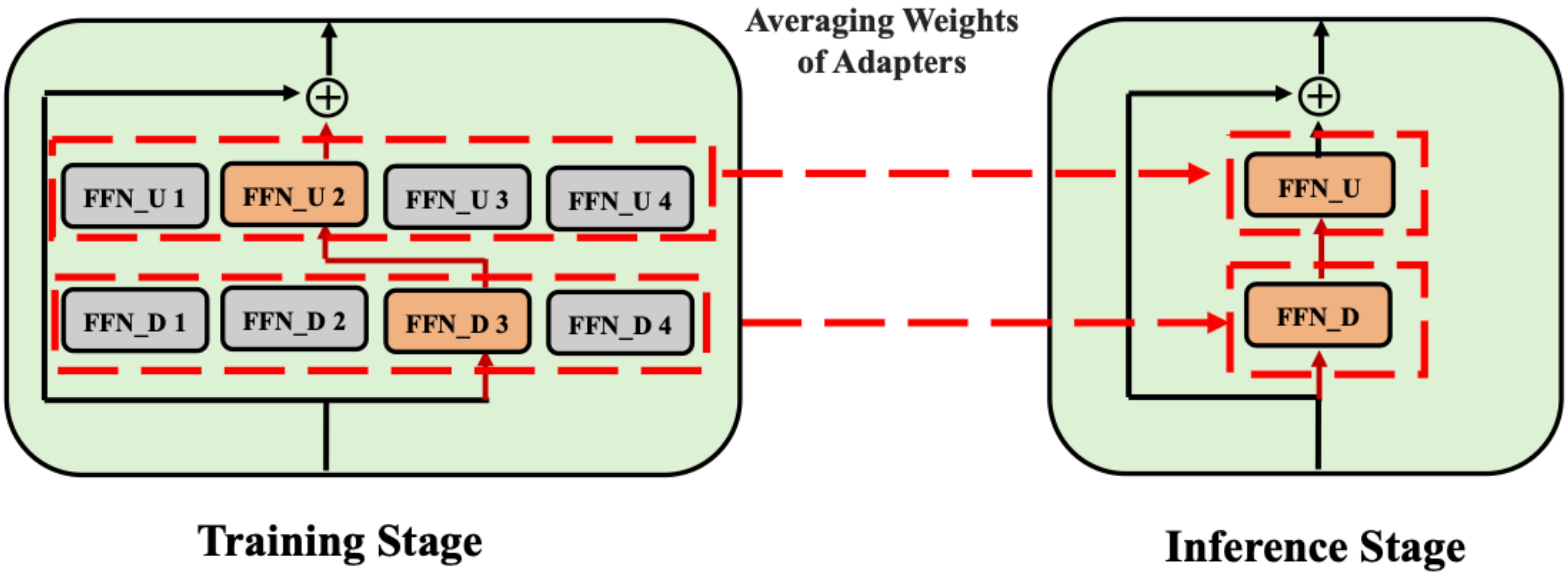
Mixture of Adapters

Regularization loss

- Let $\mathscr{A} = \{A^L_{\ell=1}\}$ and $\mathscr{B} = \{B^L_{\ell=1}\}$ be the adapter modules.
- Pick i, j twice for each input batch.
- Let $D(\mathscr{X}, \mathscr{Y}) = \mathrm{KL}(z^{\mathscr{X}}(x) \| z^{\mathscr{Y}}(x))$ where x is the input to the LLM with frozen parameters; only $\mathscr{X}, \mathscr{Y}$ are trained against fine-tuning prediction loss.
- Add following consistency loss to fine-tuning a LLM
  - $\mathscr{L} \leftarrow \mathscr{L} + \frac{1}{2}(D(\mathscr{A}, \mathscr{B}) + D(\mathscr{B}, \mathscr{A}))$

Mixture of Adapters

https://arxiv.org/abs/2205.12410



$$W^{\ell}_{\text{down}} = \frac{1}{M}\sum_{j=1}^{M} W_{\text{down}^{\ell,j}} \qquad W^{\ell}_{\text{up}} = \frac{1}{M}\sum_{j=1}^{M} W_{\text{up}^{\ell,j}}$$

Results on GLUE with ROBERTa-large

Model	#Param.	MNLI Acc	QNLI Acc	SST2 Acc	QQP Acc	MRPC Acc	CoLA Mcc	RTE Acc	STS-B Pearson	**Avg.**
Full Fine-tuning†	355.0M	90.2	94.7	96.4	92.2	90.9	68.0	86.6	**92.4**	88.9
Pfeiffer Adapter†	3.0M	90.2	94.8	96.1	91.9	90.2	68.3	83.8	92.1	88.4
Pfeiffer Adapter†	0.8M	90.5	94.8	96.6	91.7	89.7	67.8	80.1	91.9	87.9
Houlsby Adapter†	6.0M	89.9	94.7	96.2	92.1	88.7	66.5	83.4	91.0	87.8
Houlsby Adapter†	0.8M	90.3	94.7	96.3	91.5	87.7	66.3	72.9	91.5	86.4
LoRA†	0.8M	90.6	94.8	96.2	91.6	90.2	68.2	85.2	92.3	88.6
AdaMix Adapter	0.8M	**90.9**	**95.4**	**97.1**	**92.3**	**91.9**	**70.2**	**89.2**	**92.4**	**89.9**

https://arxiv.org/abs/2205.12410

Results on GLUE with BERT-base

Model	#Param.	**Avg.**
Full Fine-tuning†	110M	82.7
Houlsby Adapter†	0.9M	83.0
BitFit$^{\diamond}$	0.1M	82.3
Prefix-tuning†	0.2M	82.1
LoRA†	0.3M	82.2
UNIPELT (AP)†	1.1M	83.1
UNIPELT (APL)†	1.4M	83.5
AdaMix Adapter	0.9M	**84.5**

https://arxiv.org/abs/2205.12410

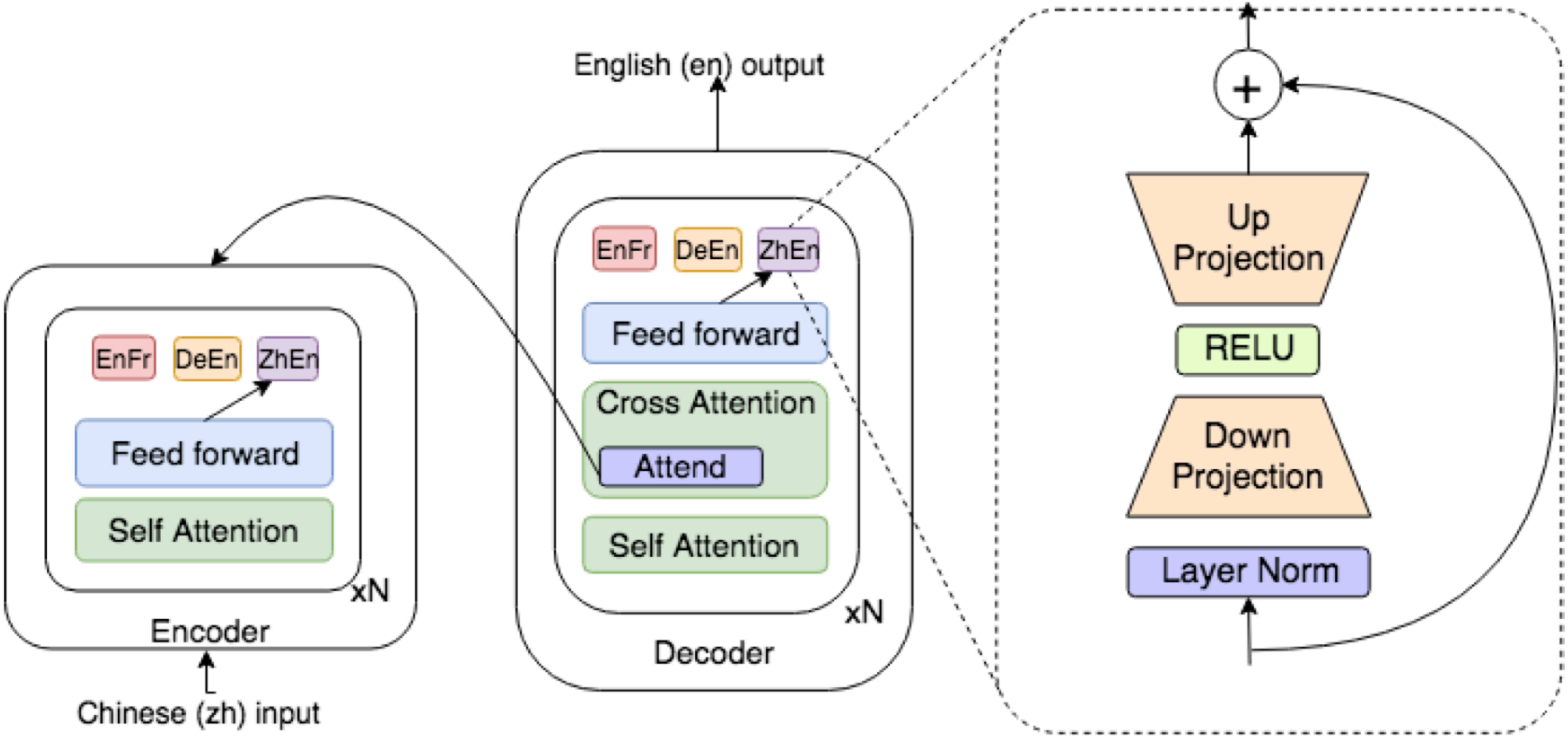
Results on E2E with GPT2-medium

Model	#Param.	BLEU	NIST	MET	ROUGE-L	CIDEr
Full Fine-tuning†	354.92M	68.2	8.62	46.2	71.0	2.47
Lin AdapterL†	0.37M	66.3	8.41	45.0	69.8	2.40
Lin Adapter†	11.09M	68.9	8.71	46.1	71.3	2.47
Houlsby Adapter†	11.09M	67.3	8.50	46.0	70.7	2.44
$\mathrm{FT}^{Top2\dagger}$	25.19M	68.1	8.59	46.0	70.8	2.41
PreLayer†	0.35M	69.7	8.81	46.1	71.4	2.49
LoRA†	0.35M	70.4	8.85	**46.8**	71.8	2.53
LoRA (repr.)	0.35M	69.8	8.77	46.6	71.8	2.52
AdaMix Adapter	0.42M	69.8	8.75	**46.8**	71.9	2.52
AdaMix LoRA	0.35M	**71.0**	**8.89**	**46.8**	**72.2**	**2.54**

https://arxiv.org/abs/2205.12410

Simple, Scalable Adaptation for Neural Machine Translation

Ankur Bapna **Naveen Arivazhagan** **Orhan Firat**
Google AI
{ankurbpn,navari,orhanf}@google.com

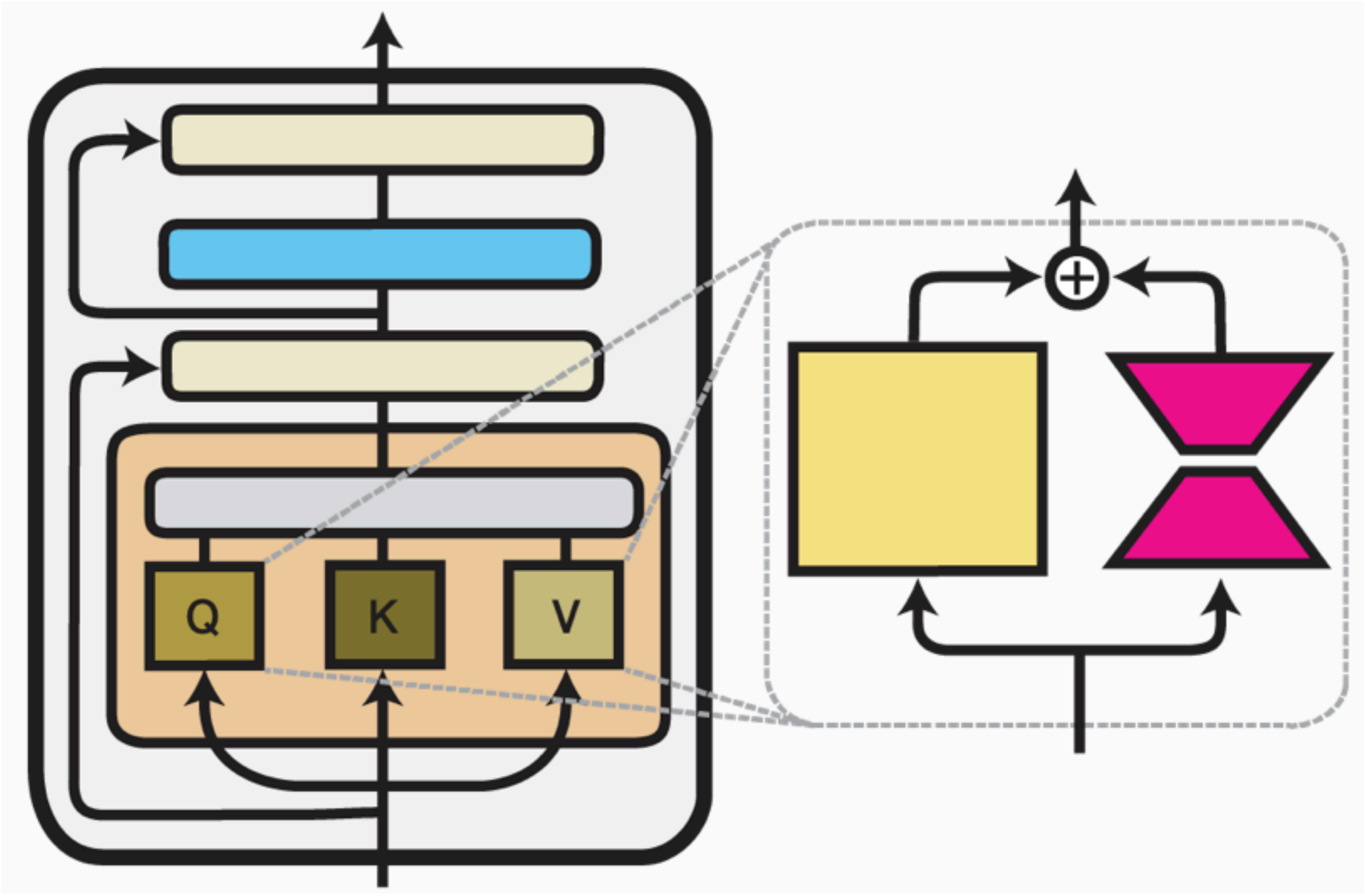

English (en) output
EnFr
DeEn
ZhEn
Feed forward
Self Attention
xN
Encoder
Chinese (zh) input
EnFr
DeEn
ZhEn
Feed forward
Cross Attention
Attend
Self Attention
xN
Decoder
+
Up Projection
RELU
Down Projection
Layer Norm

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)

LoRA

- Can be applied to any Transformer-based Large Language Model
- But specifically designed for autoregressive and causal LMs like GPTx
- Just like other Transformer adapters, LoRA adds a small set of parameters for fine-tuning and keeps the original parameters frozen
- This can help a lot when LLM parameter sizes are as large as 175 billion.

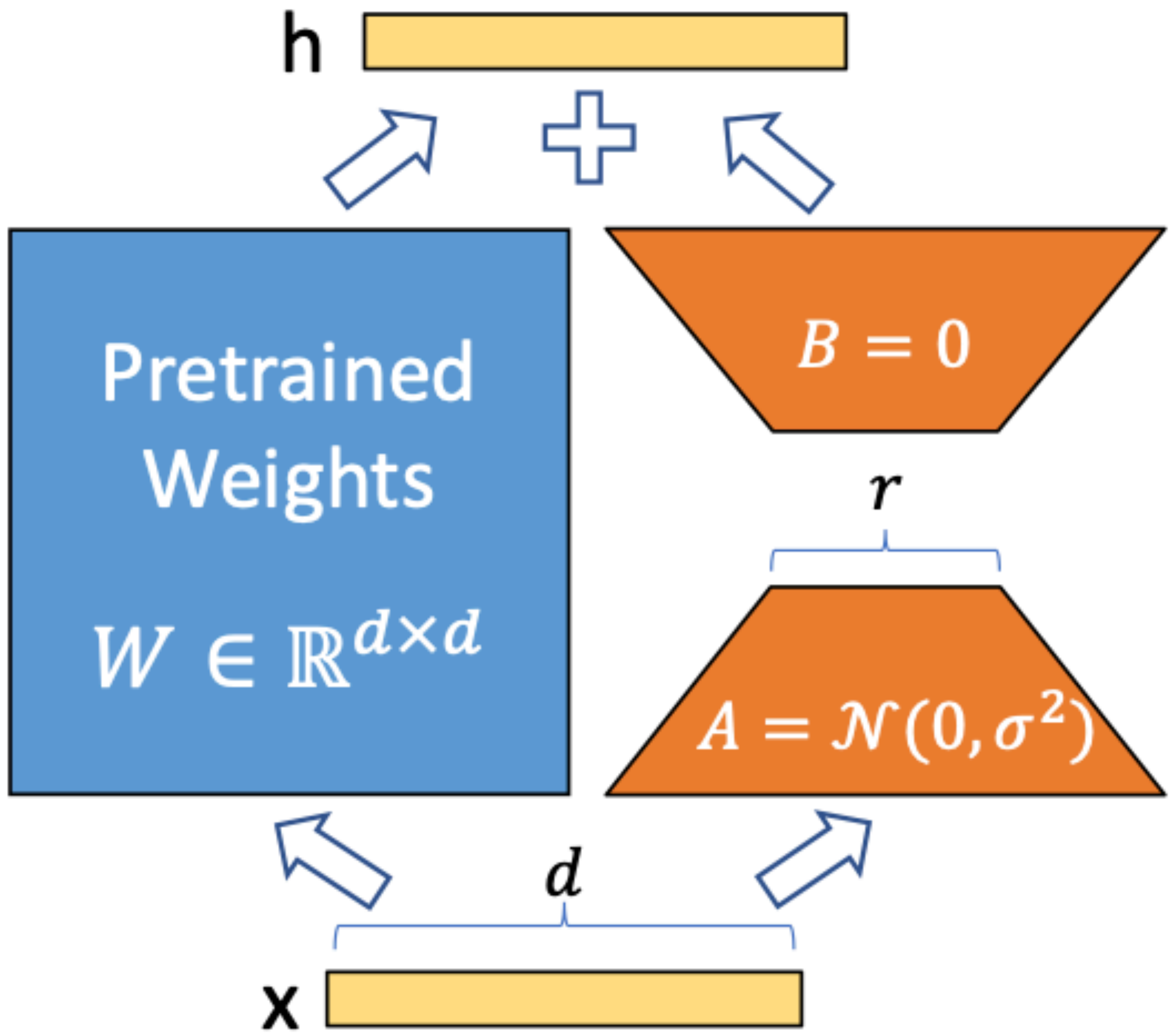


LoRA

- Only use adapters in the attention matrices: Q, K, V
- Each matrix is called W_p here, p for pre-trained
- Adapter methods modify W_p to be $W_p + BA$ where $B \in \mathbb{R}^{d \times r}, A \in \mathbb{R}^{r \times k}$
- Rank r << min(*d*, *k*)
- Let BA be zero at start of training
- Scale the parameters after backpropagation by $\frac{\alpha}{r}$ where α is a hyperparameter set to a constant value depending on r (set to the first r in training)

LoRA

- Initialize B to zeroes
- Initialize A using random Gaussian initialization



Initialize to zeroes

Initialize to values from random Gaussian

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	$\mathbf{70.4}_{\pm.1}$	$\mathbf{8.89}_{\pm.02}$	$\mathbf{46.8}_{\pm.2}$	$\mathbf{72.0}_{\pm.2}$	$2.47_{\pm.02}$

GPT-3 175B validation accuracy vs. number of trainable parameters