Washington University in St. Louis
James McKelvey School of Engineering

CSE 561A: Large Language Models

Spring 2024

Lecture 5: Parameter-Efficient Fine-Tuning

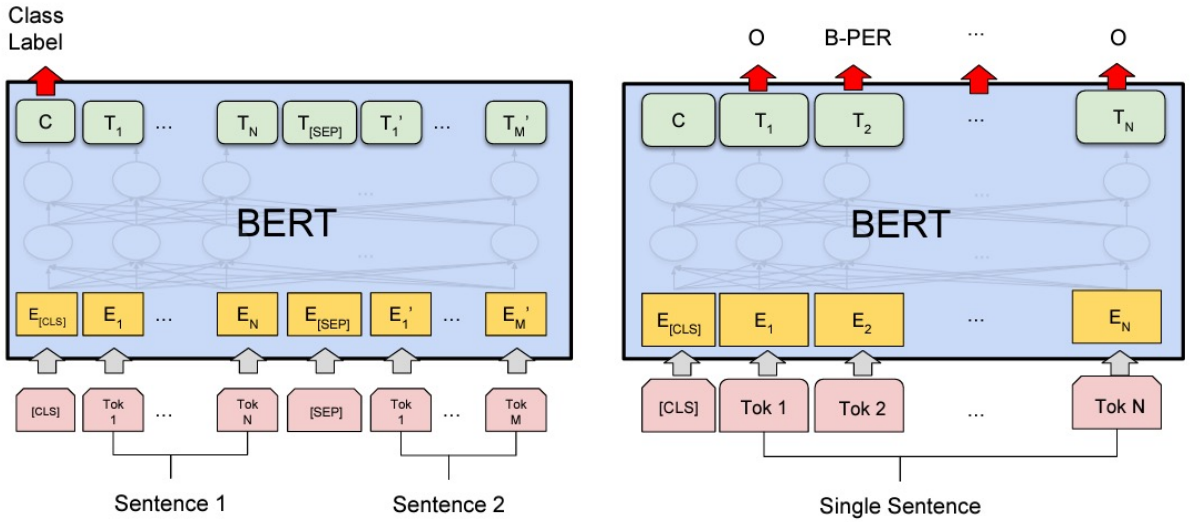
Jiaxin Huang

Course Announcements

- Next lecture will be student-presentation lecture
- Presentation Duration: 30-35 min
- Preview question form for the first student presentation will be sent out today (required to submit 3 times during the semester)
- The second student presentation lecture is on next Tuesday (Feb.6th)
- Presenters (on Feb.6th) please send your slides to me (cc the TAs) before Friday 12:00PM (Feb.2nd)

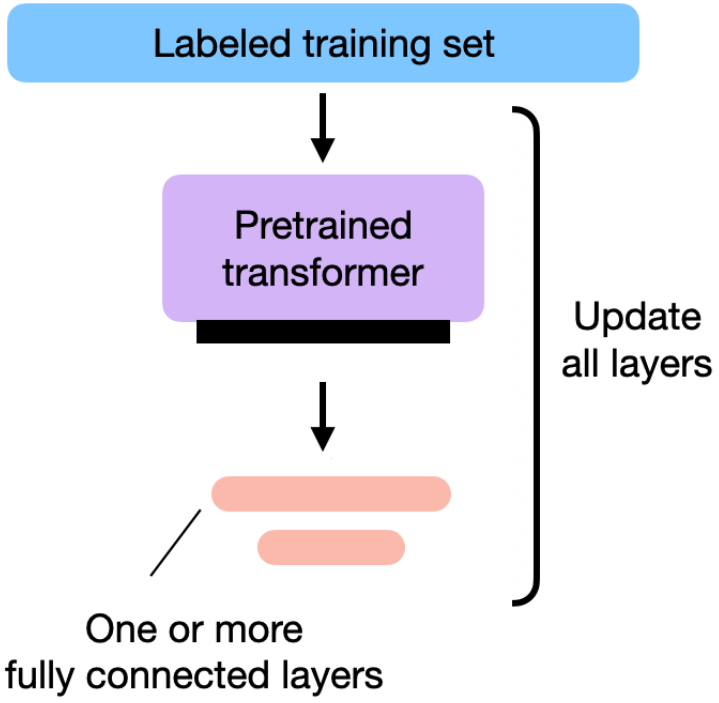
Background: Vanilla Fine-tuning

- Attach a task-specific layer to the last layer of the pre-trained transformer output
- Update the weights of all the parameters by backpropagating gradients on a downstream task



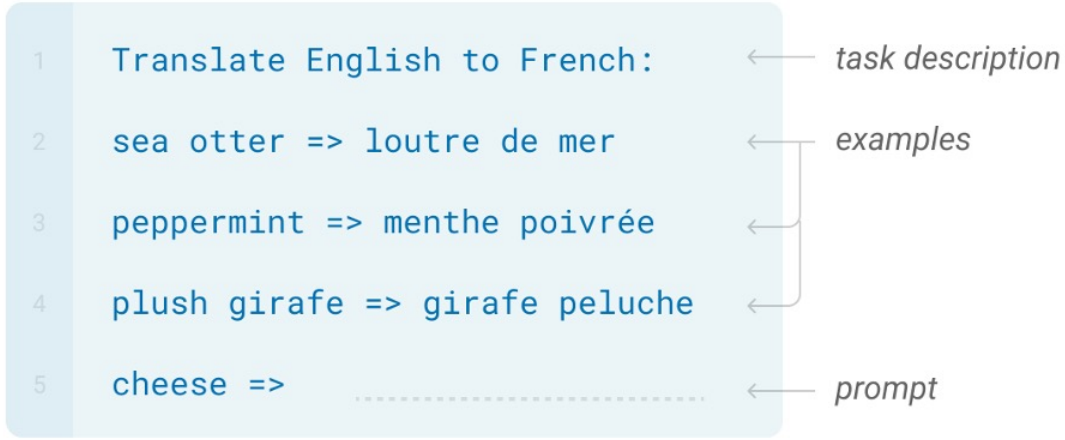
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER



Background: Prompting

- Prompting a language model with a natural description of the task, and possibly several few-shot examples. No gradient updates.



Vanilla Fine-Tuning vs. Prompting

- Vanilla fine-tuning
  - Pros: Can utilize more training data
  - Pros: Lead to stronger performance with more training data
  - Cons: Computationally expensive to train the complete network
  - Cons: Need to store a full set of model weights per task
- Prompting
  - Pros: Training-data efficient
  - Pros: Computational efficient
  - Cons: Performance depends on prompts and examples
  - Cons: Finding a good prompt could be challenging

Parameter-Efficient Fine-Tuning

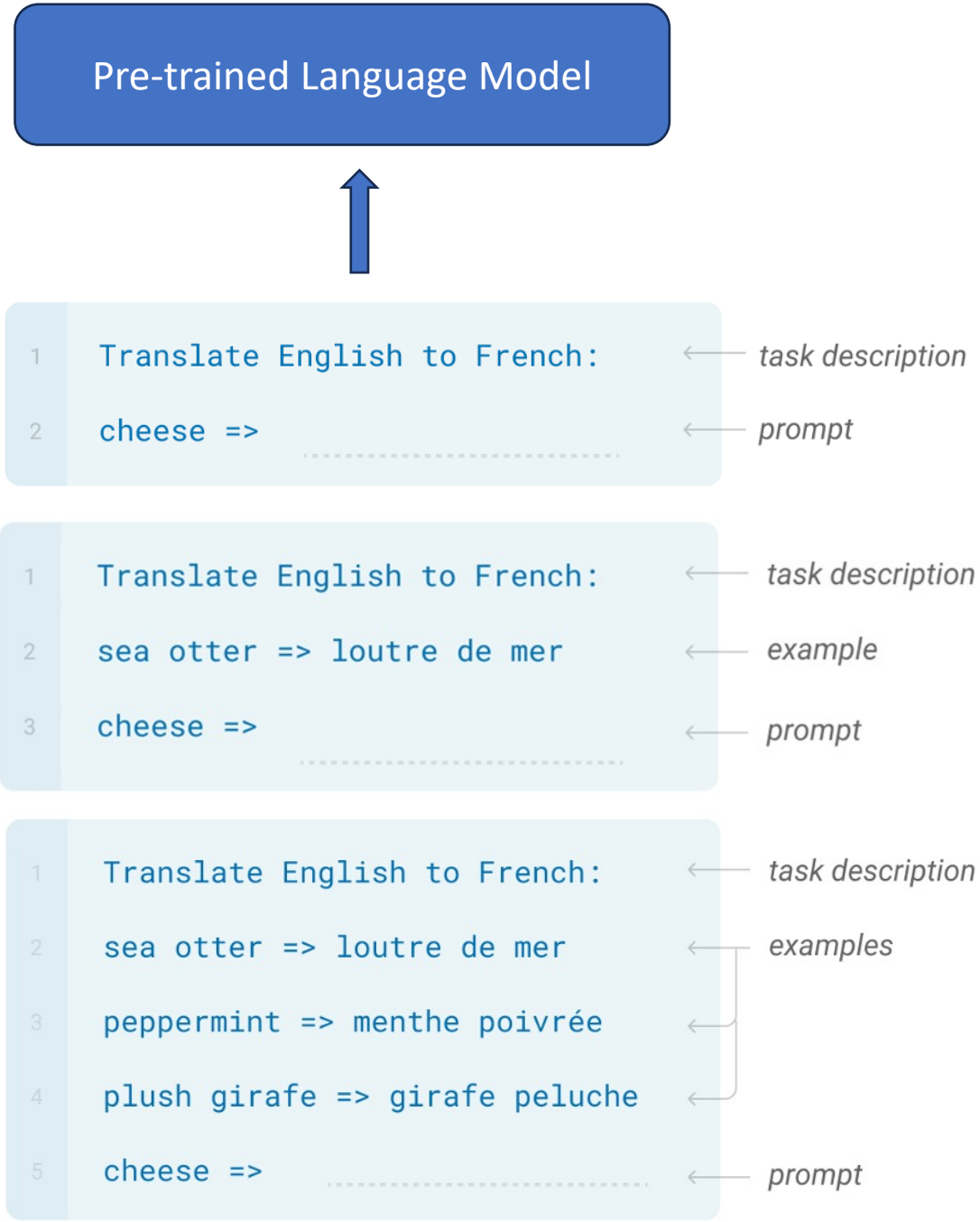
- Rather than fine-tuning the parameters in the entire model, only fine-tune a small set of weights.
  - Addition: add a small external network for each task
    - Prompt-based Methods
    - Adapter-based Methods
  - Reparameterization: reparametrize the model parameter to be more efficient for training
    - LoRA

Content

- **Prompt-based Methods**
  - Prompt Tuning
  - Prefix Tuning
- Adapter-based Methods
  - Adapters
- Reparameterization-based Methods
  - LoRA
- Summary and Comparison of the Methods

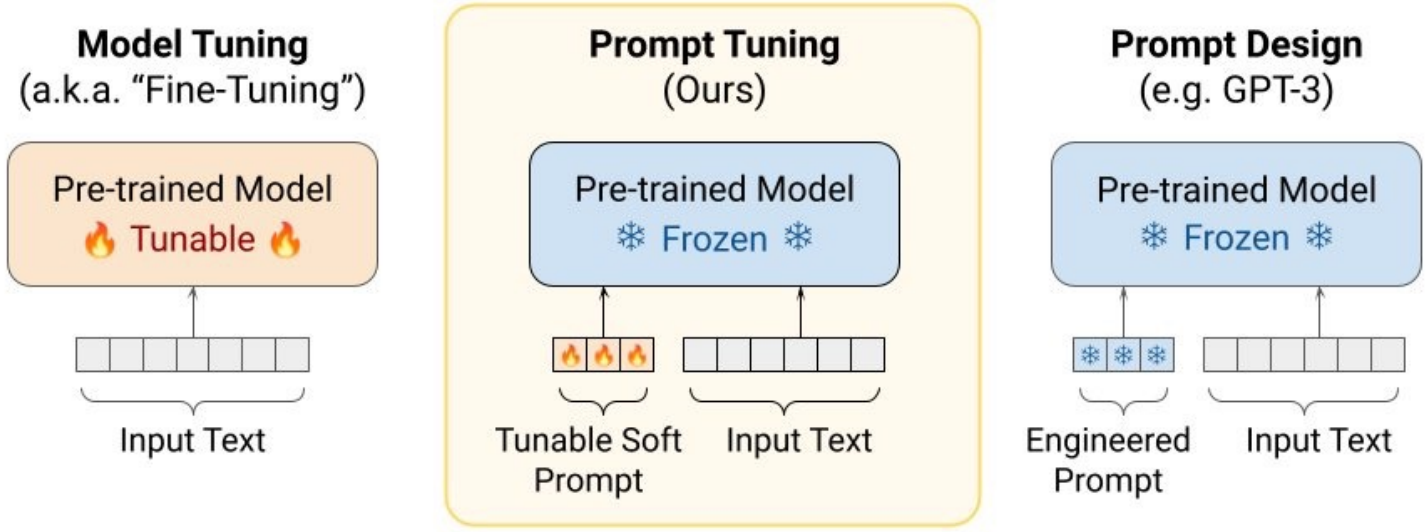
Prompt Engineering

- Paraphrasing the task instruction
- Adding detailed examples
- For each new task, search over the possible sequence space to find the prompt with the best output performance. -> **computationally expensive!**
- Can we use a set of parameters to replace these prompts and train them with labeled sets?



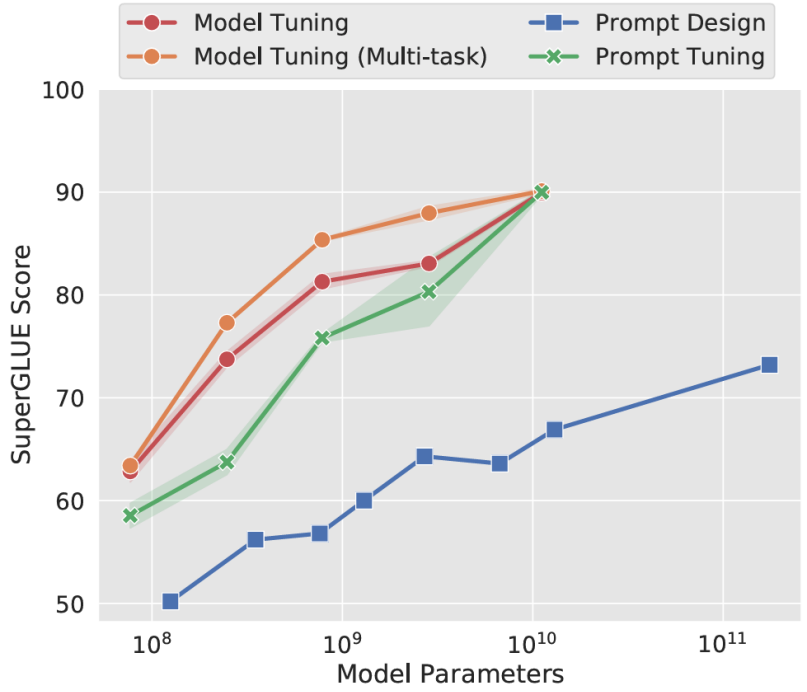
Prompt-Tuning

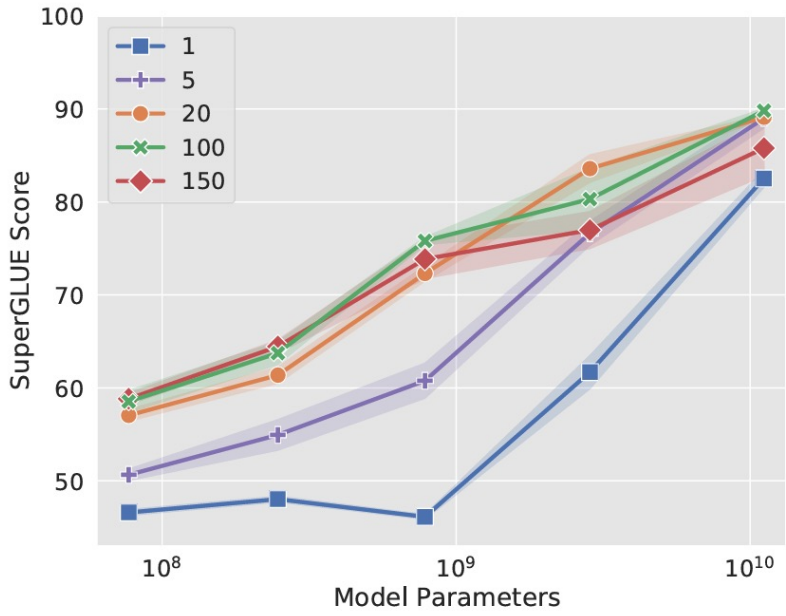
- Prepend a sequence of tokens as tunable embeddings to the input data (as soft prompts)
- freeze the whole Transformer model during training, and only tune the prepended soft prompts
- Only a small set of parameters need to be stored for each task



The Power of Scale for Parameter-Efficient Prompt Tuning (Lester et. al, 2021)

- Prompt-tuning becomes more effective when the pre-trained model becomes larger
- Larger models perform well even with a small number of prompt tokens





(a) Prompt length

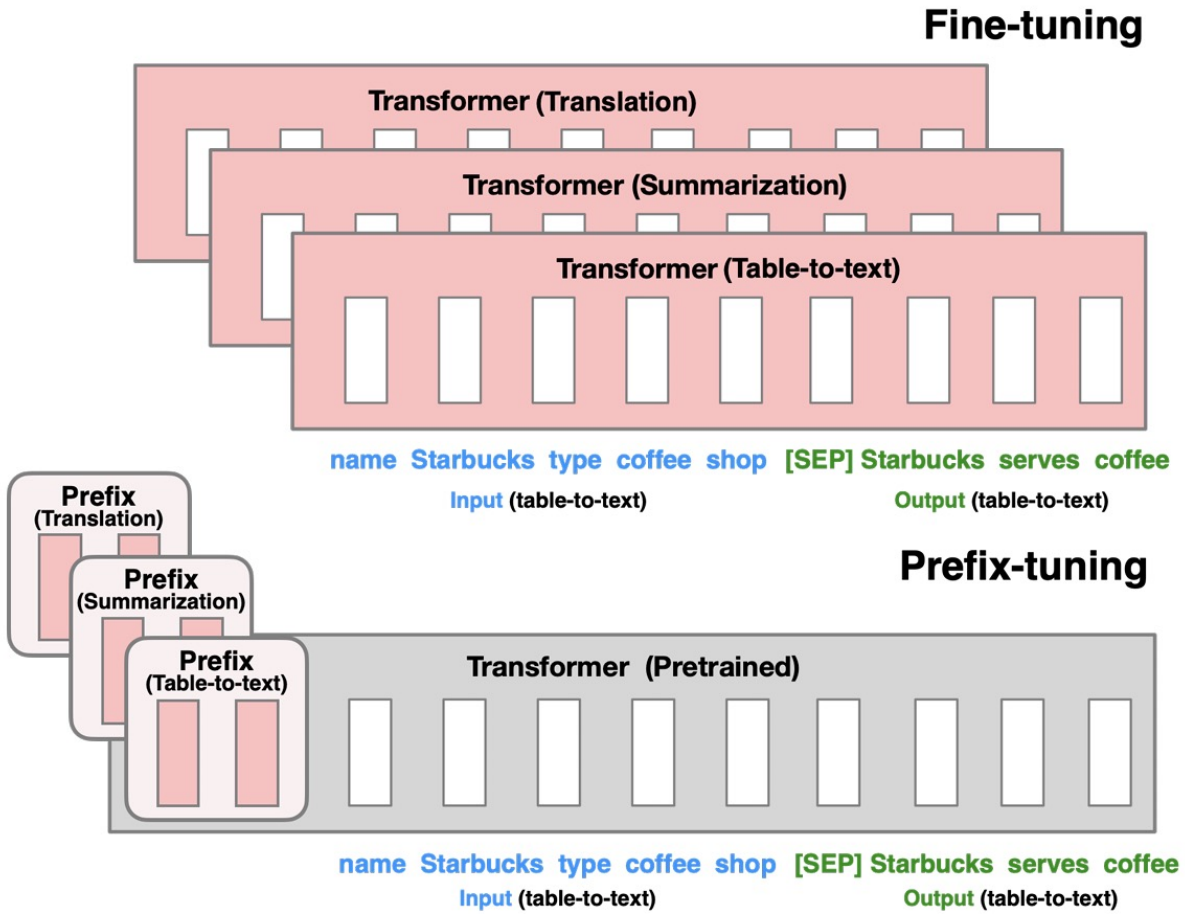
The Power of Scale for Parameter-Efficient Prompt Tuning (Lester et. al, 2021)

- Initializing prompt tokens with real tokens in vocabulary is helpful

(b) Prompt initialization

Prefix-Tuning: Optimizing Continuous Prompts for Generation (Li et. al, 2021)

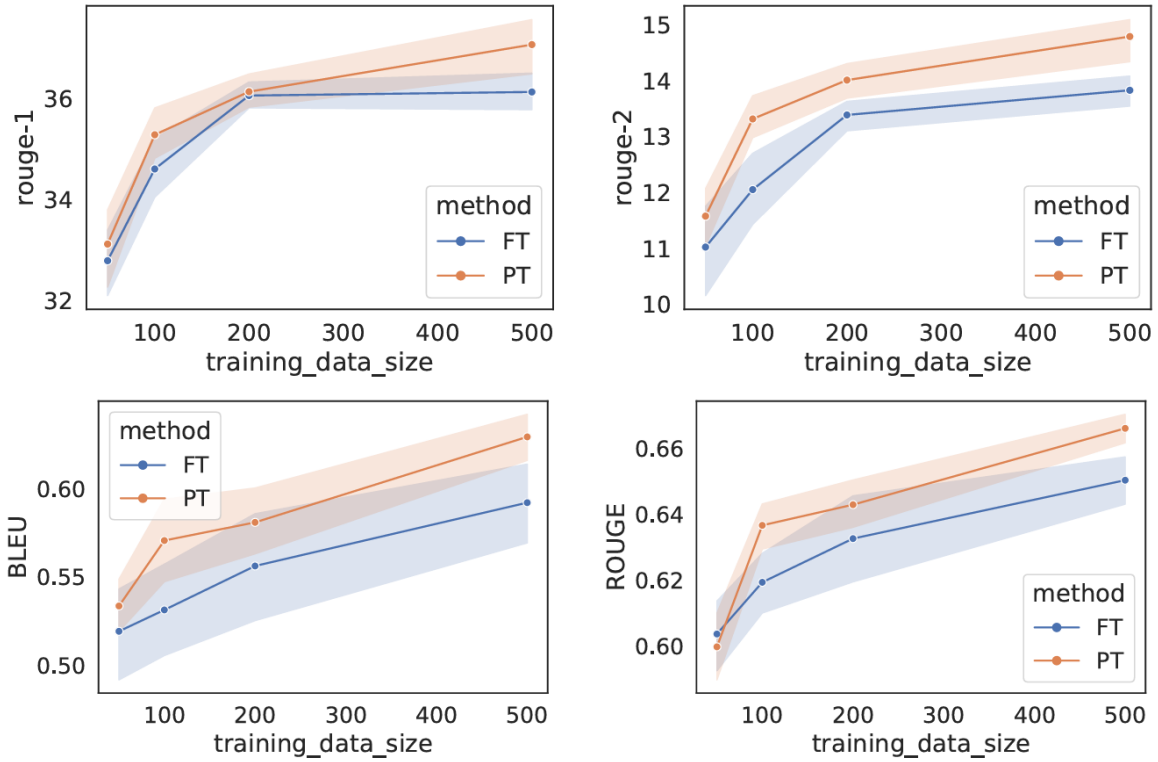
- Similar with prompt-tuning, except that the soft prompt tokens are prepended to each layer in the Transformer instead of just the input layer.



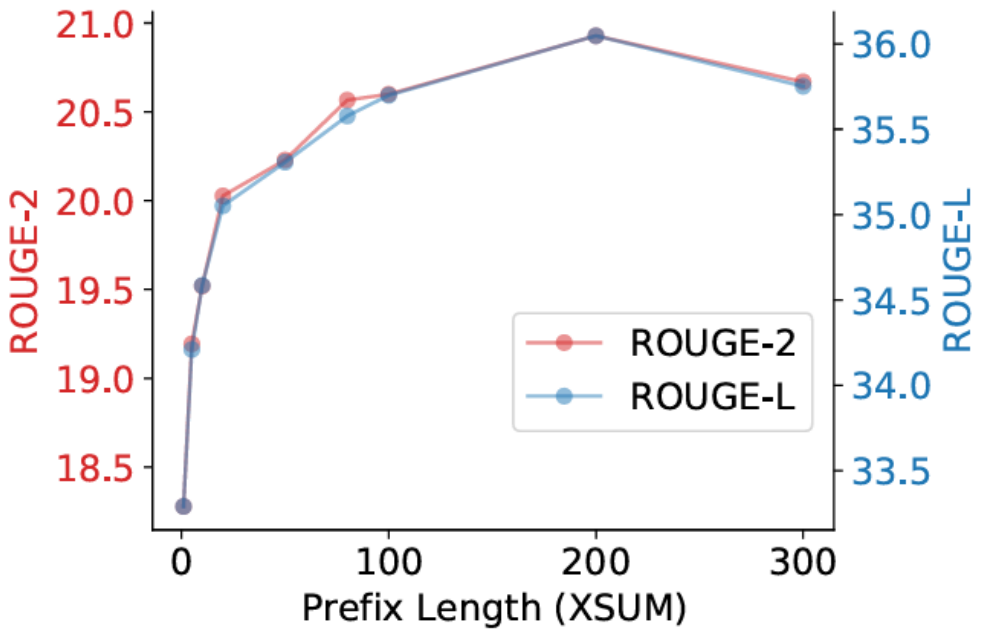
Experiments on Text Generation

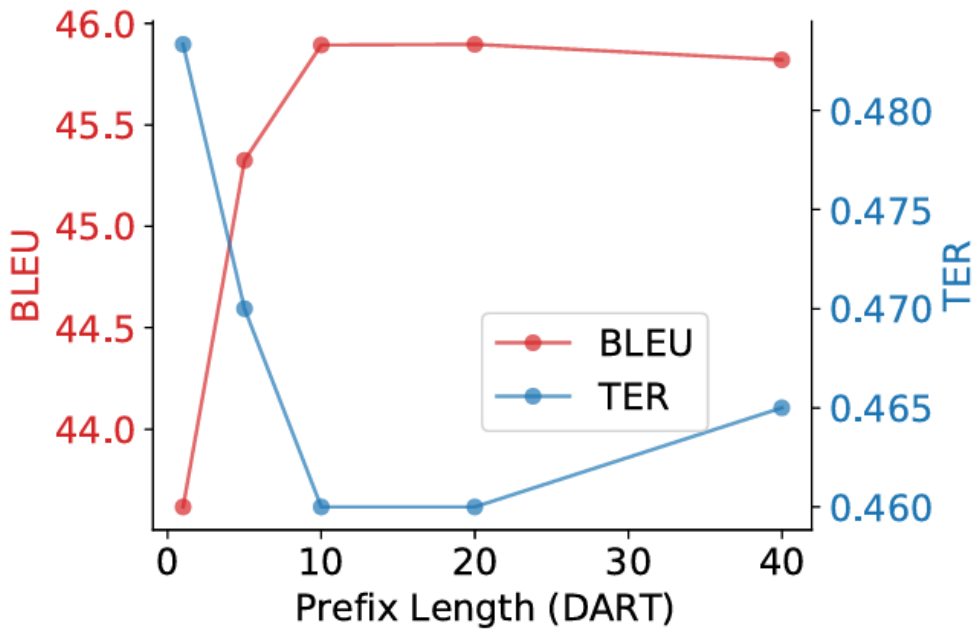
- Prefix-tuning works better than fine-tuning under low-data settings.

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 Length





- Performance increases as the prefix length increases up to athreshold (200 for summarization and 10 for table-to-text) and then a slight performance drop occurs.

Prefix Initialization

- Random initialization leads to low performance with high variance.
- Initializing the prefix with real words significantly improves generation, as shown in Figure5.
- Initializing with task relevant words such as “summarization” and “table-to-text” obtains slightly better performance than task irrelevant words such as “elephant” and “divide”.

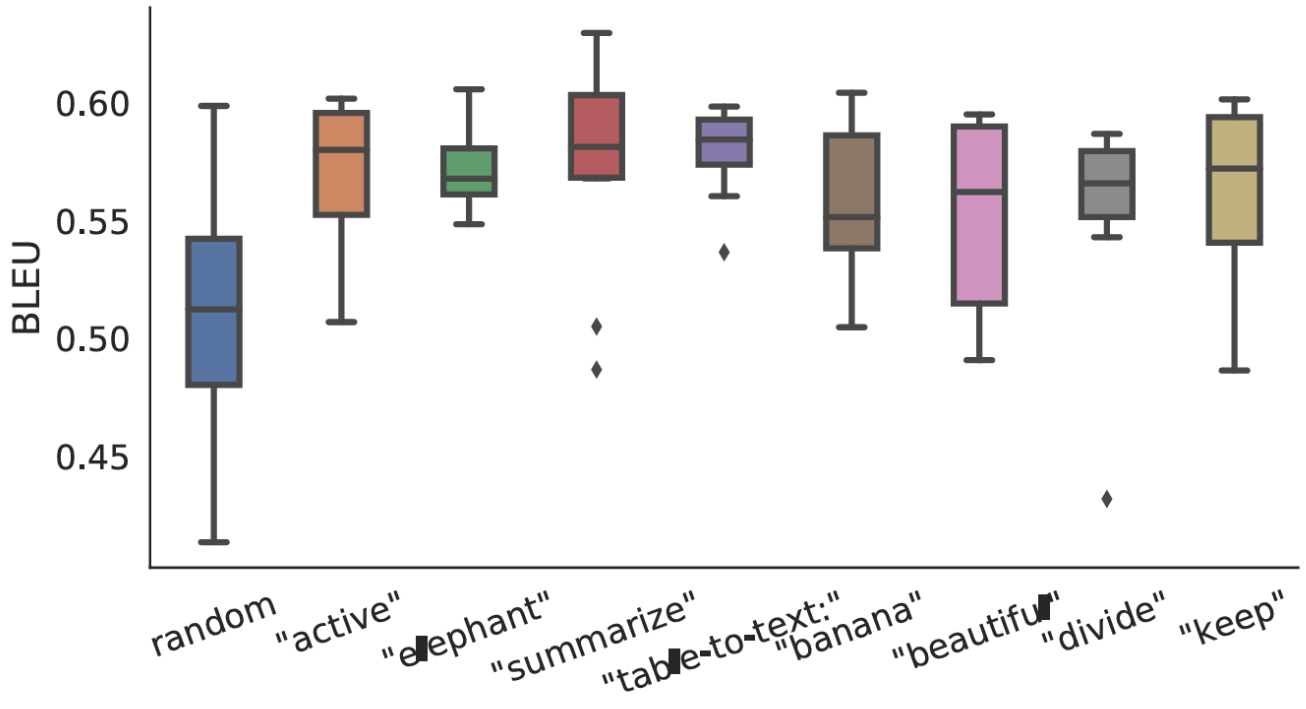


Figure 5: Initializing the prefix with activations of real words significantly outperforms random initialization, in low-data settings.

Issues with Prompt/Prefix-Tuning

- Optimal prefix length may be different for tasks
- The prefix occupies the length of your input context to the Transformer

Content

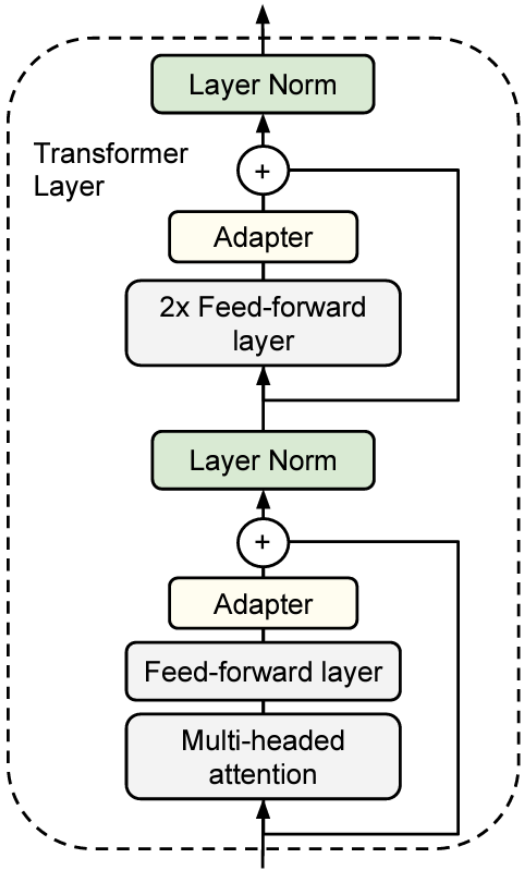
- Addition-based Methods
  - Prompt Tuning
  - Prefix Tuning
- **Adapter-based Methods**
  - Adapters
- Reparameterization-based Methods
  - LoRA
- Summary and Comparison of the Methods

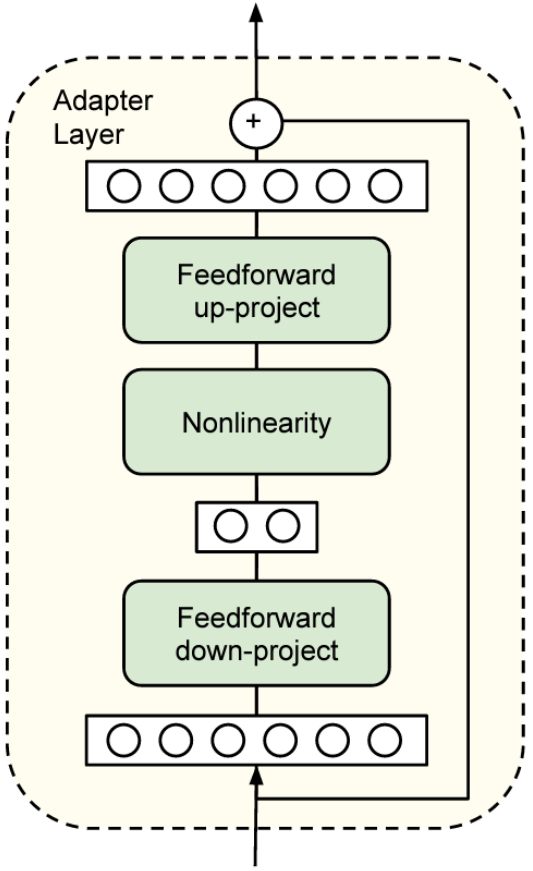
What are Adapters?

- Vanilla fine-tuning can be seen as adding an extra layer to the top of a Transformer
- Adapter modules perform more general architectural modifications: injecting new layers/modules into the original network.
- During training, the original network weights are untouched, only the adapter weights are updated.

Parameter-Efficient Transfer Learning for NLP (Houlsby et. al, 2019)

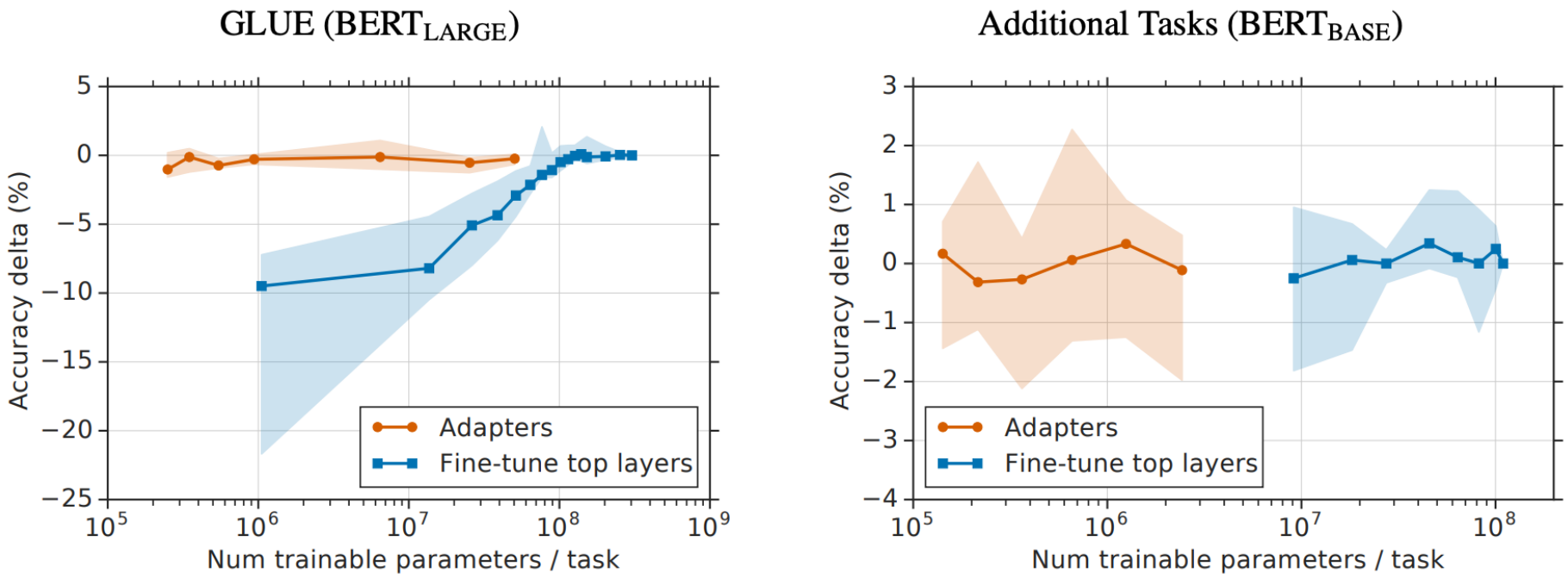
- Adding adapter layers to each transformer layer: after the self-attention layer and the feed-forward layer
- Adapter modules have two main features
  - a small number of parameters
  - a near-identity initialization
- Only adapter layers and the final classification layer is updated during training





Comparison with Fine-Tuning

- Adapter-based tuning achieves a similar performance to full fine-tuning with several orders of magnitude fewer trained parameters.



Pros and Cons of Adapter-based Methods

- Pros:
  - Empirically very effective in multi-task settings
  - Computationally efficient compared to full fine-tuning
- Cons:
  - Adding in new layers makes the model slower during inference time
  - Make the model size larger

Content

- Addition-based Methods
  - Prompt Tuning
  - Prefix Tuning
- Specification-based Methods
  - Adapters
- **Reparameterization-based Methods**
  - LoRA
- Summary and Comparison of the Methods

Intrinsic Dimension

- An objective function's intrinsic dimension measures the minimum number of parameters needed to reach a satisfactory solution to the objective.
- Alternatively, the intrinsic dimension represents the lowest dimensional subspace in which one can optimize the original objective function to within a certain level of approximation error.

Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. Aghajanyan et. al, 2020.

Intrinsic Dimension

- Let θ^D be the parameters of a model
- Instead of optimizing θ^D, the subspace method optimizes θ^d in a lower dimensional space

$$\theta^D = \theta_0^D + P(\theta^d) \qquad P : \mathbb{R}^d \rightarrow \mathbb{R}^D$$

- P is often a linear projection:

$$\theta^D = \theta_0^D + \theta^d M$$

- Fine-tuning tasks have a low intrinsic dimension: the number of parameters to be modified are several orders of magnitude less than the the full parameterization of the pre-trained model.

Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. Aghajanyan et. al, 2020.

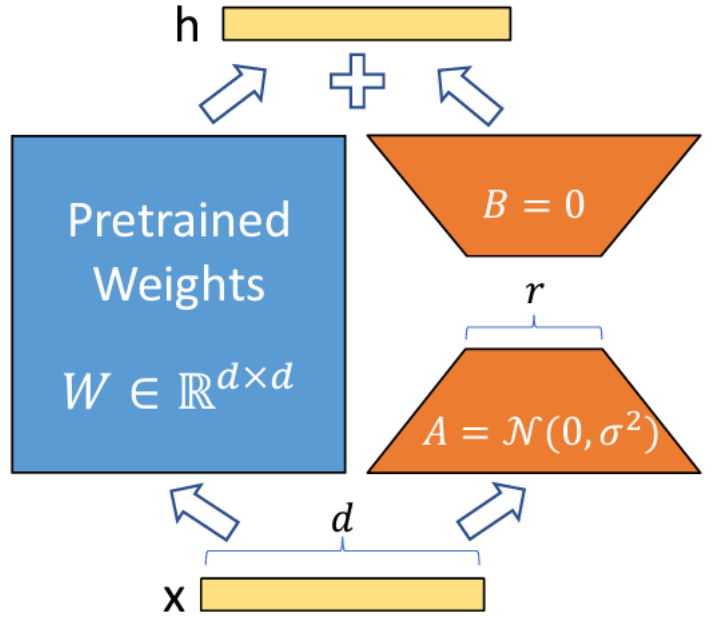
LoRA: Low-Rank Adaptation of Large Language Models (Hu et. al, 2021)

- A neural network contains many dense layers which perform matrix multiplication.
- Inspired by the low intrinsic dimension assumption, hypothesize that the update weights can also have a low intrinsic rank
- Pre-trained matrix to be update:

$$W_0 \in \mathbb{R}^{d \times k}$$

- Updated matrix

$$W_0 + \Delta W$$



LoRA: Low-Rank Adaptation of Large Language Models (Hu et. al, 2021)

- Reparametrize the updated weight with low-rank decomposition

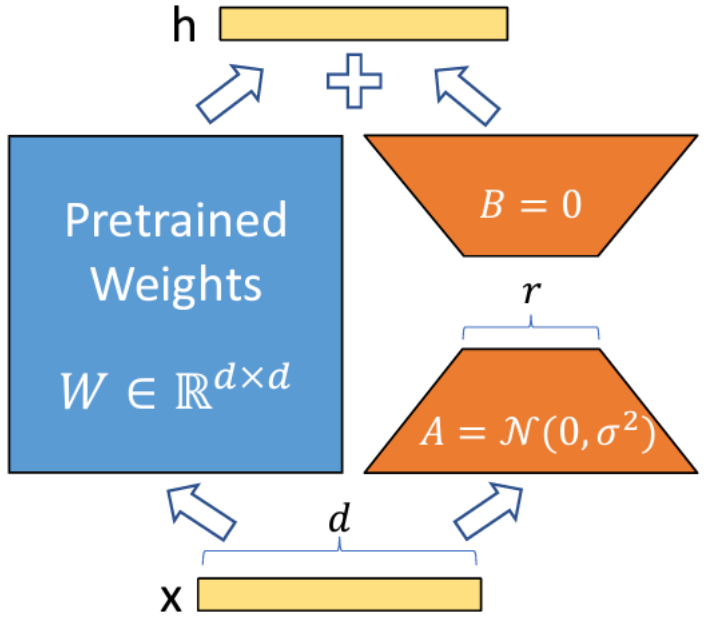
$$W_0 + \Delta W = W_0 + BA$$

- where A and B are low rank matrices

$$A \in \mathbb{R}^{r \times k} \quad B \in \mathbb{R}^{d \times r}$$

- For the hidden state h of an input x , $h = W_0 x$
- The updated hidden state is now

$$h = W_0 x + \Delta W x = W_0 x + BAx$$



Applying LoRA to Transformers

- In principle, LoRA can be applied to any weight matrices in deep learning
- In this study, they focus on applying LoRA to attention matrices in Transformers
- r ranges from 2 to 64
- For GPT3-175B
- VRAM: 1.2TB -> 350GB
- Checkpoint storage: 350GB -> 35MB (10000x smaller)

Comparison with Other Fine-Tuning Methods

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

- LoRA outperforms several baselines with comparable or fewer trainable parameters.

Which Matrices Should We Apply LoRA to?

	# of Trainable Parameters = 18M						
Weight Type	W_q	W_k	W_v	W_o	W_q, W_k	W_q, W_v	W_q, W_k, W_v, W_o
Rank r	8	8	8	8	4	4	2
WikiSQL ($\pm 0.5\%$)	70.4	70.0	73.0	73.2	71.4	**73.7**	**73.7**
MultiNLI ($\pm 0.1\%$)	91.0	90.8	91.0	91.3	91.3	91.3	**91.7**

- Putting all the parameters in ΔW_q or ΔW_k results in significantly lower performance, while adapting both W_q and W_v yields a good result.

Optimal Rank for LoRA

- r = 4 and r = 8 already give a good result, and increasing r does not cover more meaningful subspaces

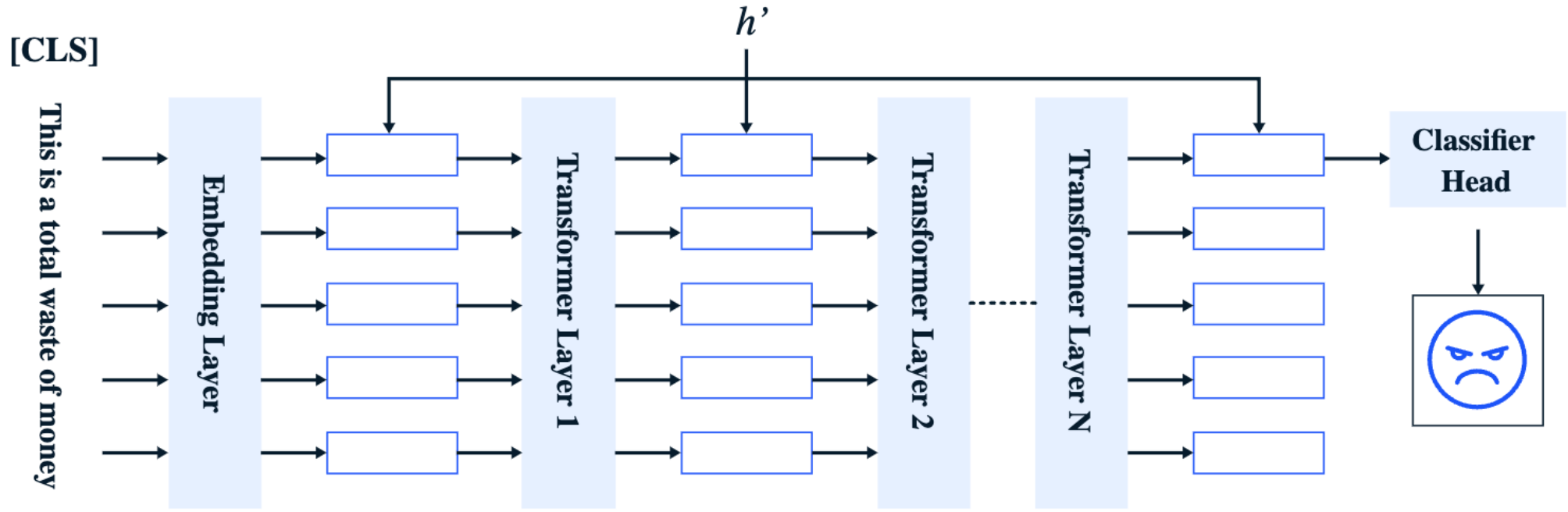
	Weight Type	$r = 1$	$r = 2$	$r = 4$	$r = 8$	$r = 64$
WikiSQL($\pm 0.5\%$)	W_q	68.8	69.6	70.5	70.4	70.0
	W_q, W_v	73.4	73.3	73.7	73.8	73.5
	W_q, W_k, W_v, W_o	74.1	73.7	74.0	74.0	73.9
MultiNLI ($\pm 0.1\%$)	W_q	90.7	90.9	91.1	90.7	90.7
	W_q, W_v	91.3	91.4	91.3	91.6	91.4
	W_q, W_k, W_v, W_o	91.2	91.7	91.7	91.5	91.4

Content

- Addition-based Methods
  - Prompt Tuning
  - Prefix Tuning
- Specification-based Methods
  - Adapters
- Reparameterization-based Methods
  - LoRA
- **Summary and Comparison of the Methods**

Summary of Parameter-Efficient Fine-Tuning

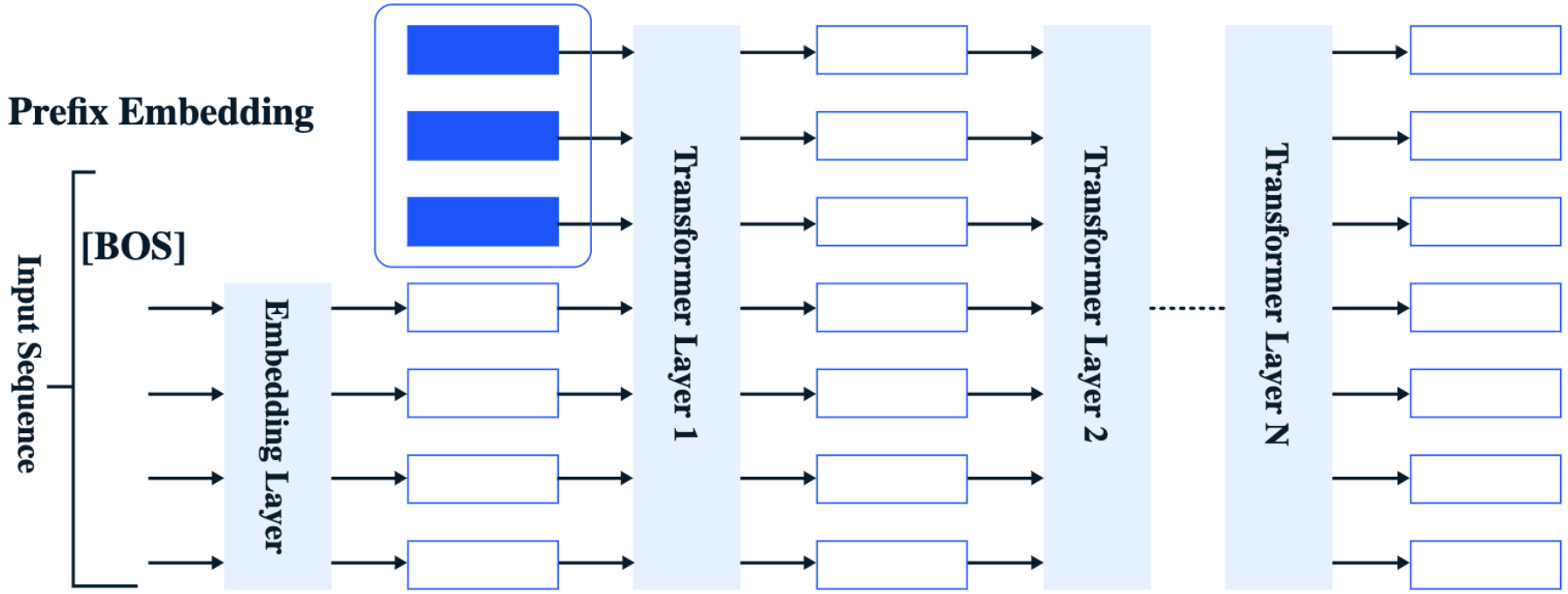
- Vanilla Fine-Tuning



Source: https://www.leewayhertz.com/parameter-efficient-fine-tuning/

Summary of Parameter-Efficient Fine-Tuning

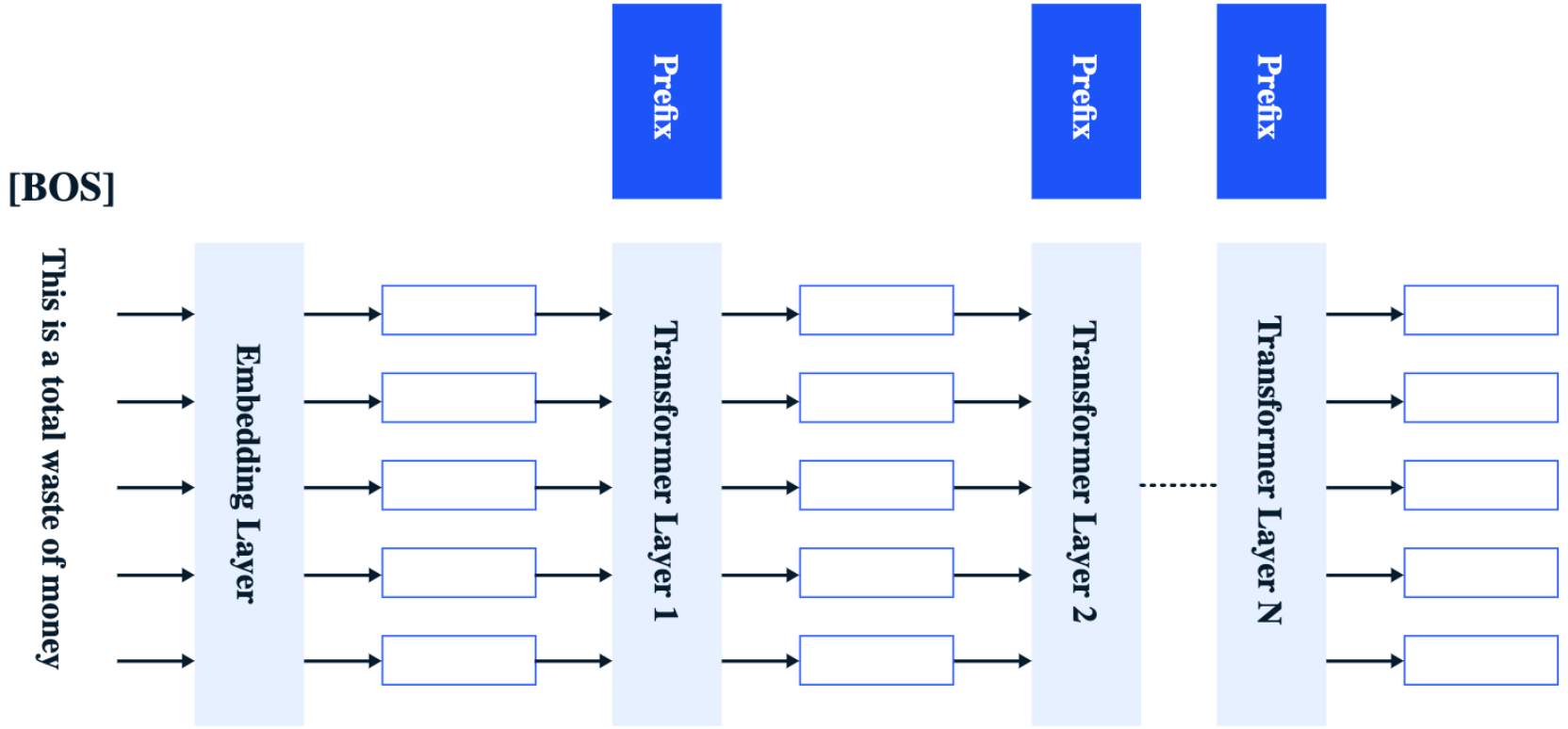
- Prompt-Tuning



Source: https://www.leewayhertz.com/parameter-efficient-fine-tuning/

Summary of Parameter-Efficient Fine-Tuning

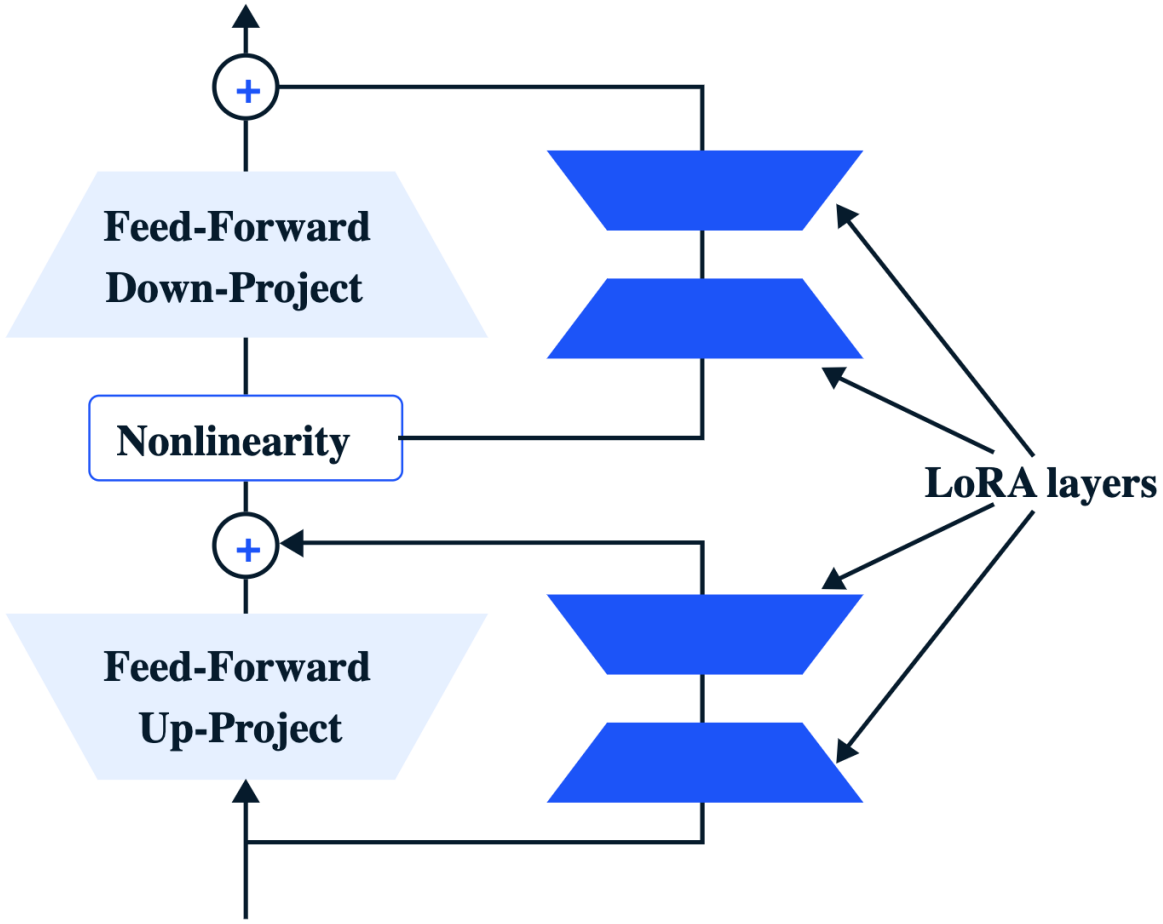
- Prefix-Tuning



Source: https://www.leewayhertz.com/parameter-efficient-fine-tuning/

Summary of Parameter-Efficient Fine-Tuning

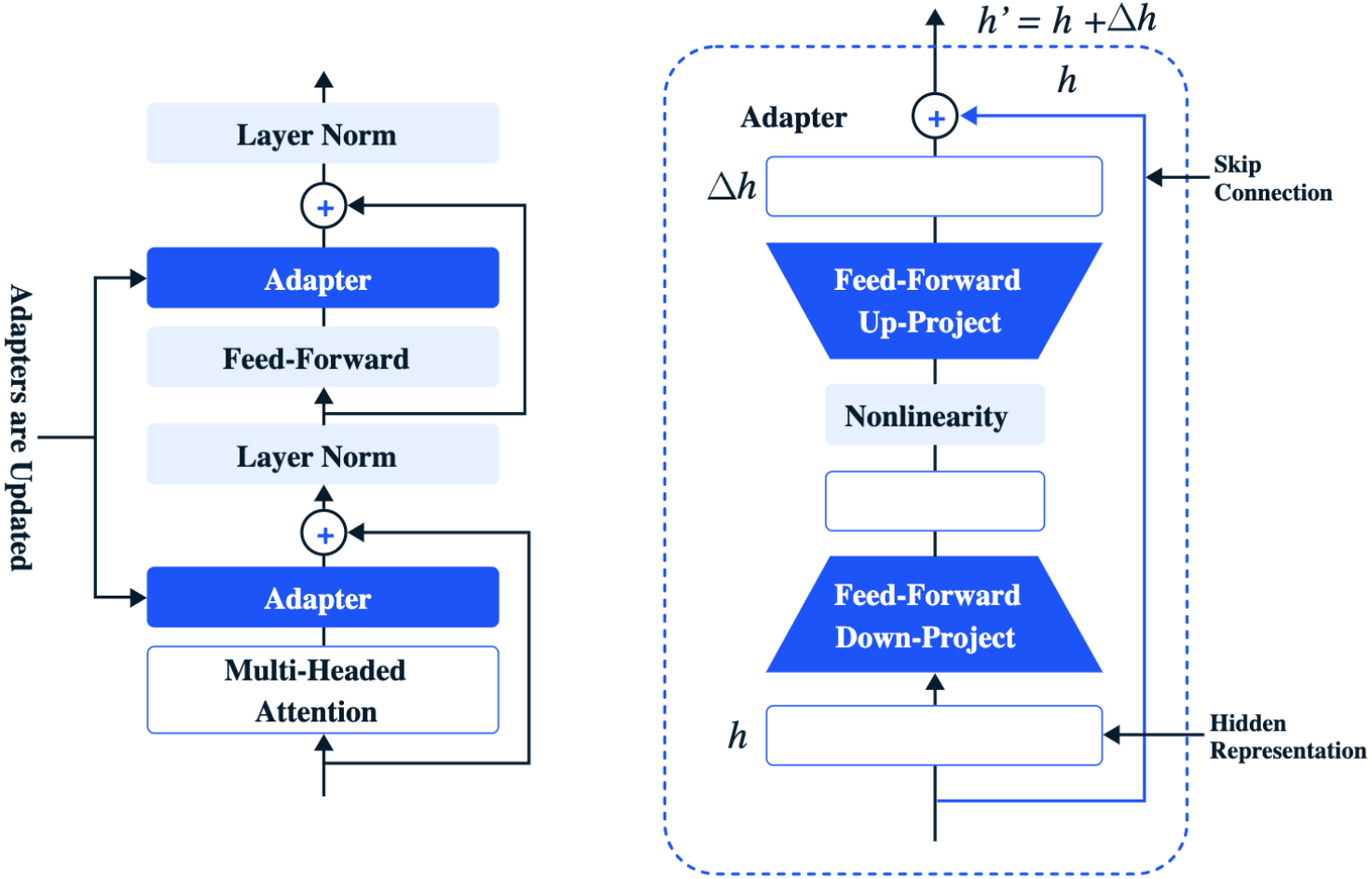
- LoRA



Source: https://www.leewayhertz.com/parameter-efficient-fine-tuning/

Summary of Parameter-Efficient Fine-Tuning

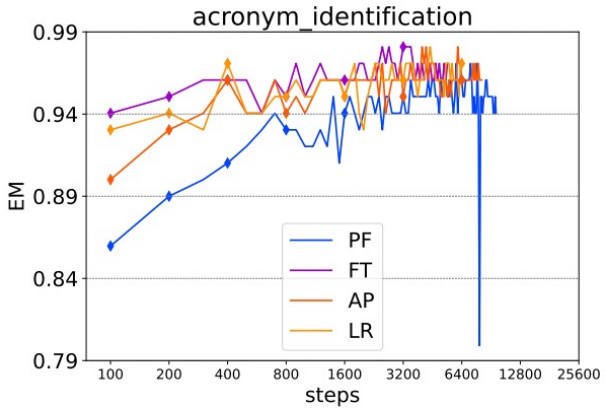
- Adapters

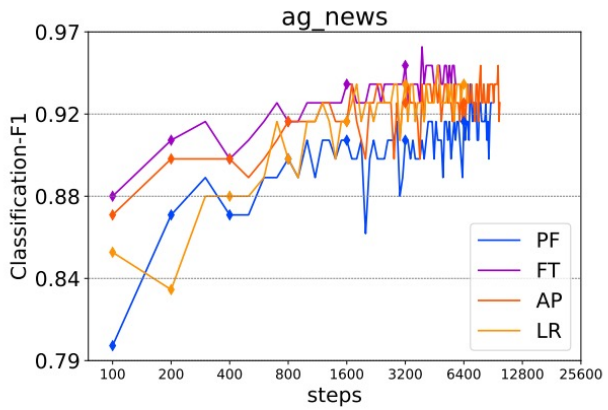


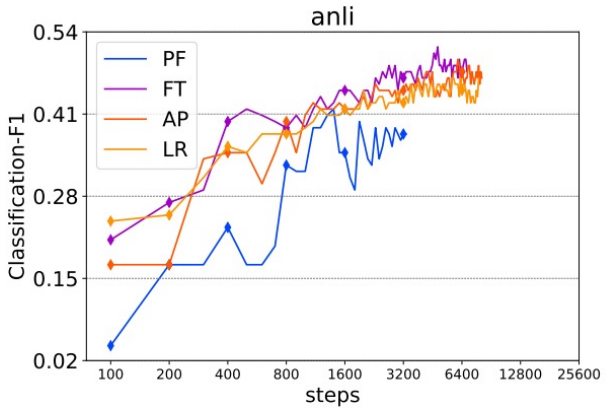
Source: https://www.leewayhertz.com/parameter-efficient-fine-tuning/

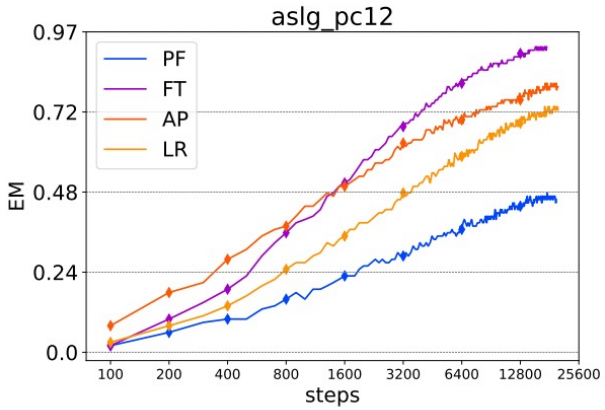
Performance Comparison on Various NLP Tasks

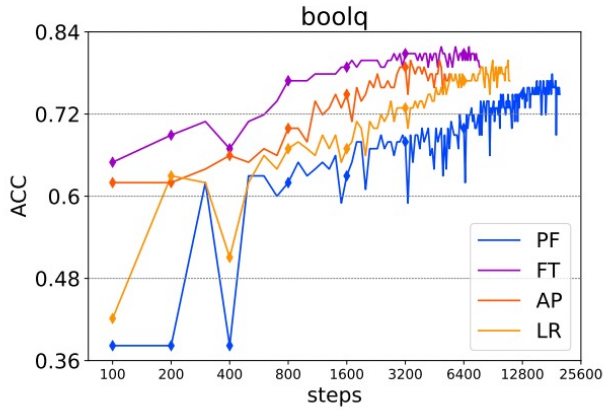
- If you have enough data and computing resources:
- Overall performance (on T5-base): Full fine-tuning > LoRA > Adapters > Prefix Tuning > Prompt Tuning

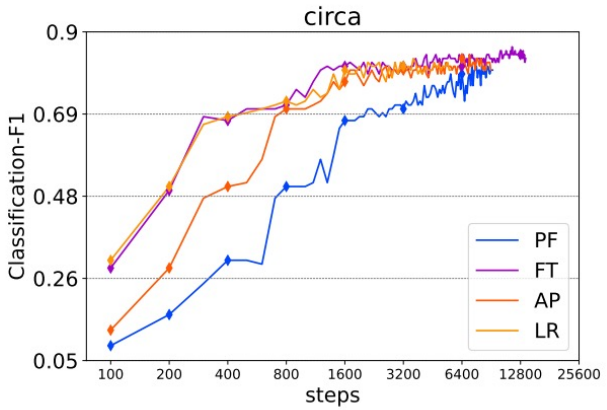












Delta Tuning: A Comprehensive Study of Parameter Efficient Methods for Pre-trained Language Models (Ding et. al, 2023)

Discussion Question

- Suppose you have two tasks, and you want to use multi-task prompt tuning to train a soft prompt for each of them.
- (a) Prepend task 1(or 2) prompt when training on task 1(or 2) data
- (b) Prepend task 1 and task 2 prompt together when training on all task data
- Which one works better? (Hint: different conditions lead to different answers)