AI VIETNAM
All-in-One Course

# NLP Course

# Parameter-Efficient Fine-Tuning

Nguyen Quoc Thai

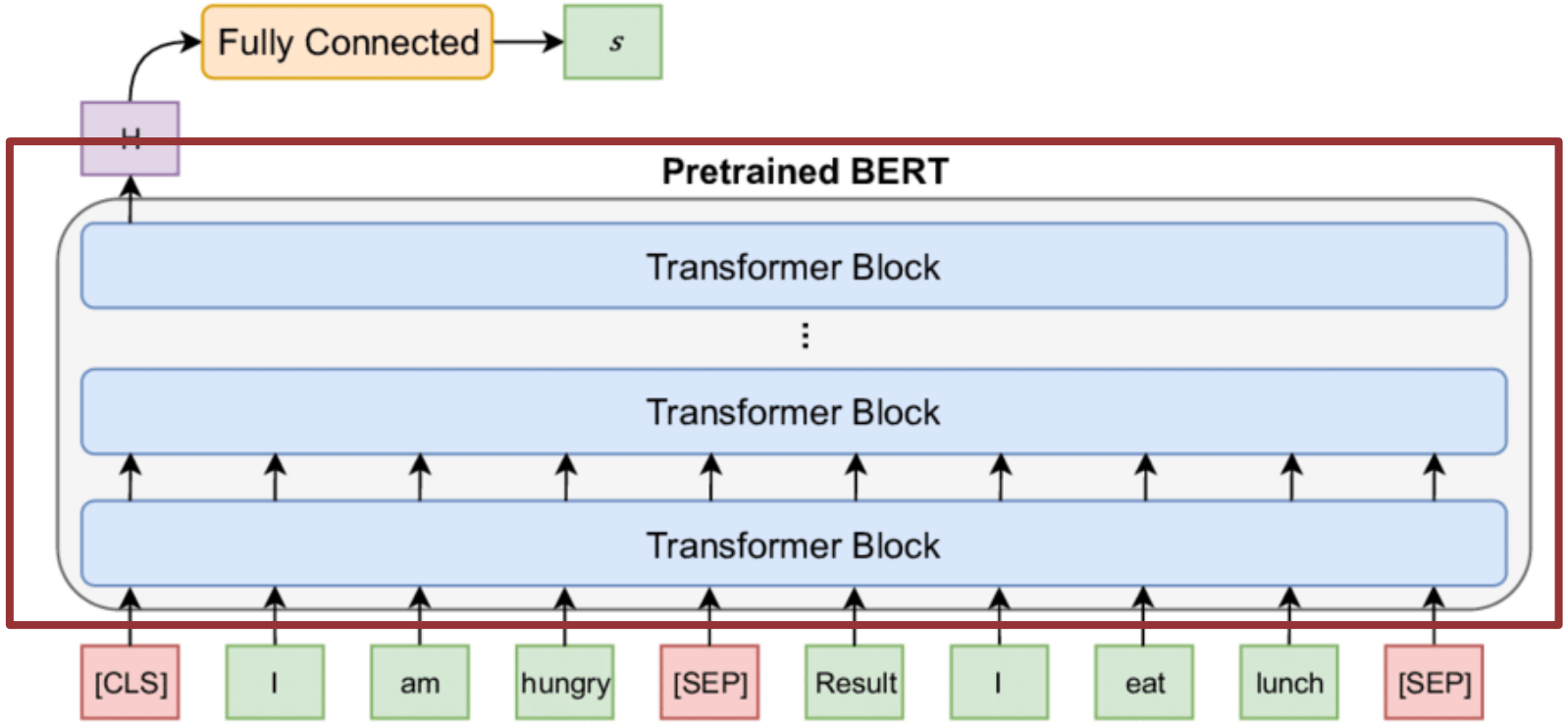
Year 2023

# CONTENT

# 1 – Background

## Fine-Tuning

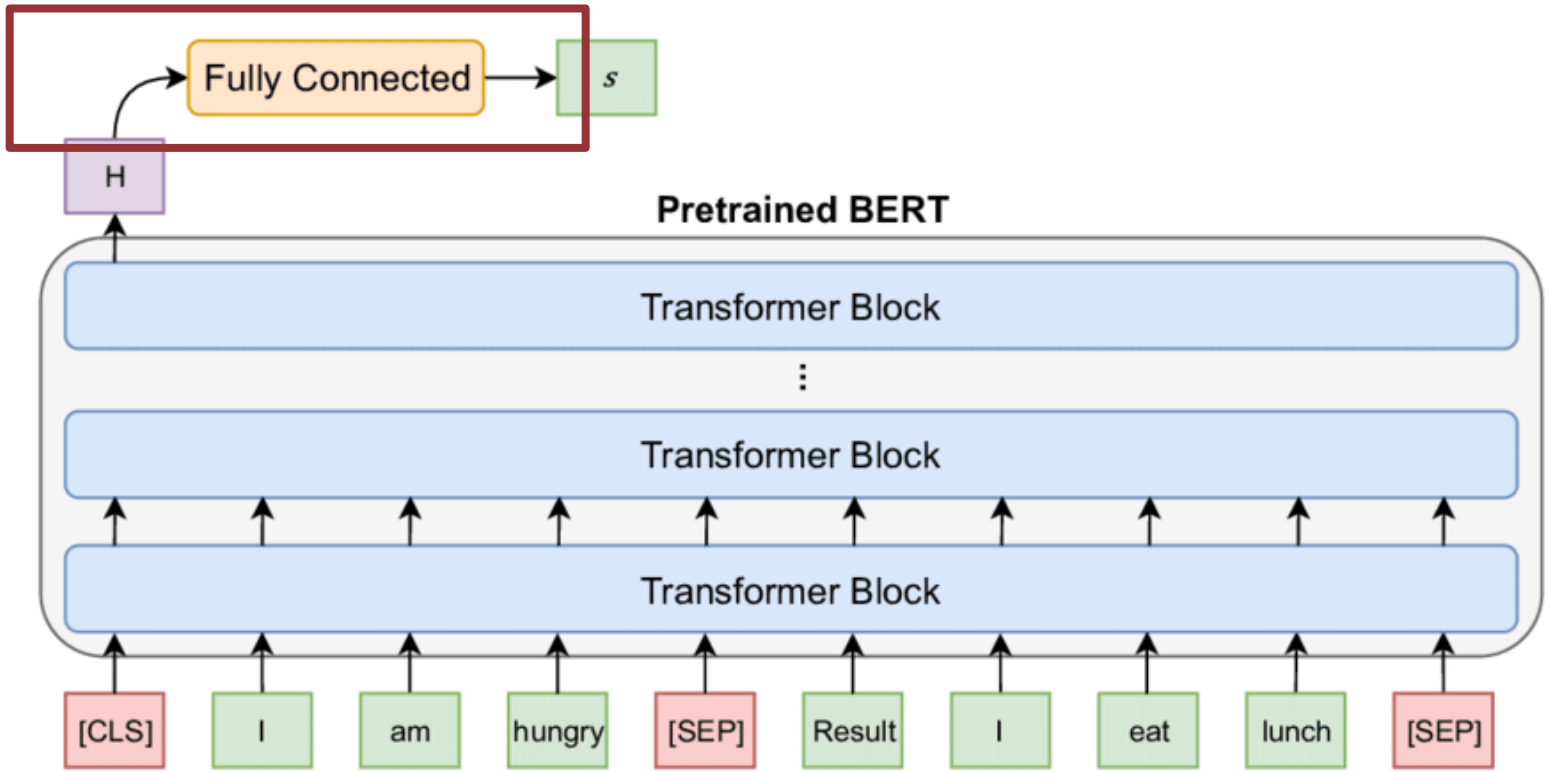
- Pretrain a language model on task



# 1 – Background

## Fine-Tuning

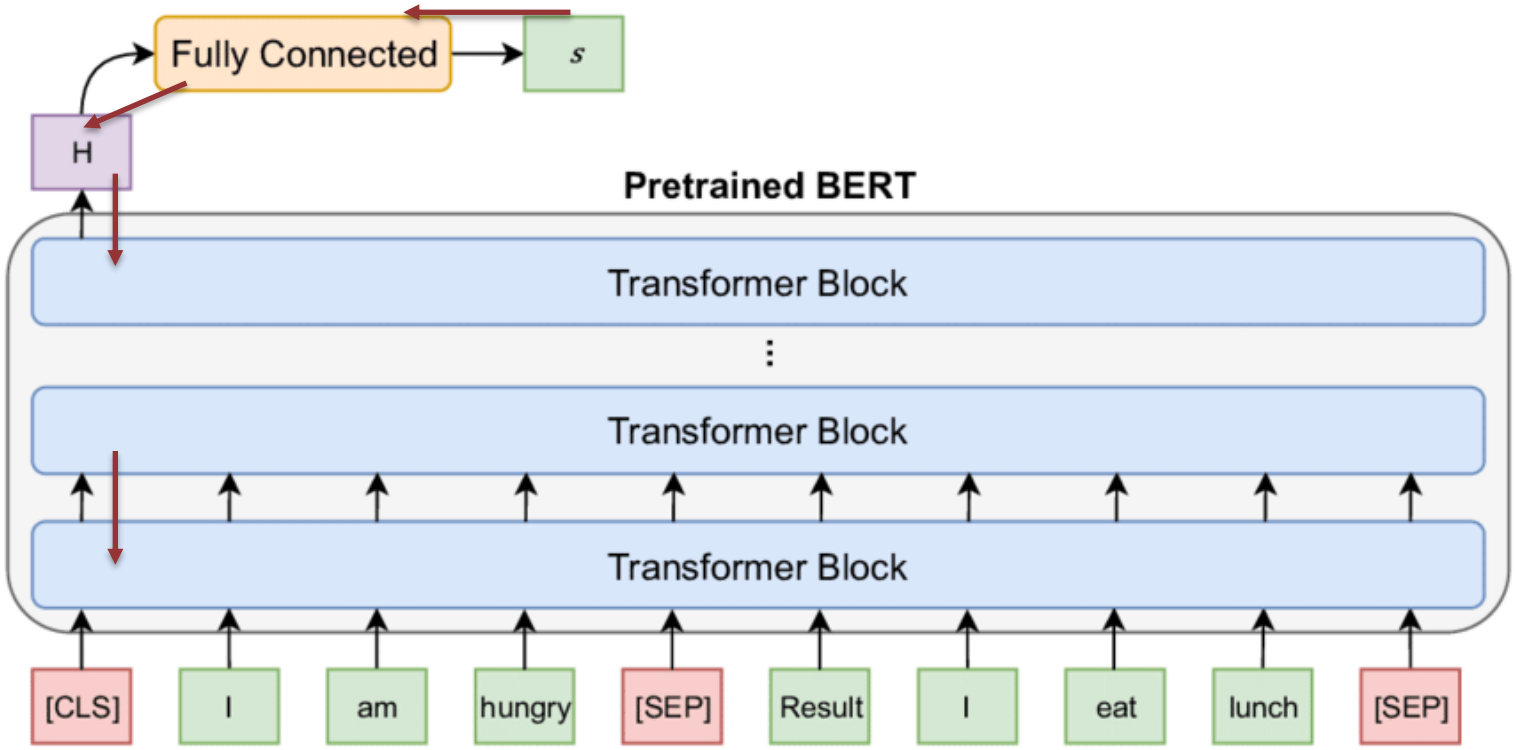
- Pretrain a language model on task
- Attach a small task specific layer



# 1 – Background

## Fine-Tuning

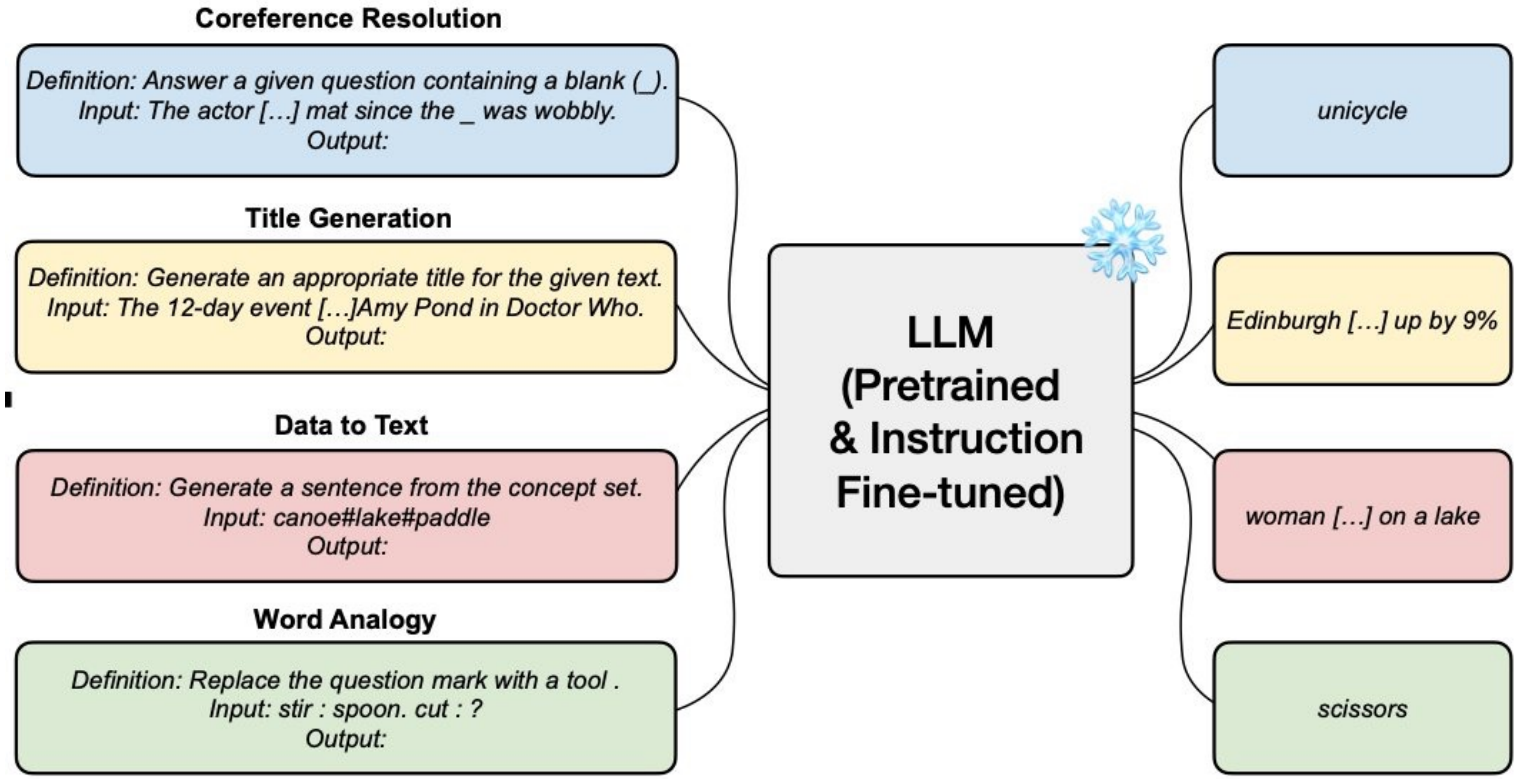
- Pretrain a language model on task
- Attach a small task specific layer
- Fine-tune the weights of full NN by propagating gradients on a downstream task



# 1 – Background

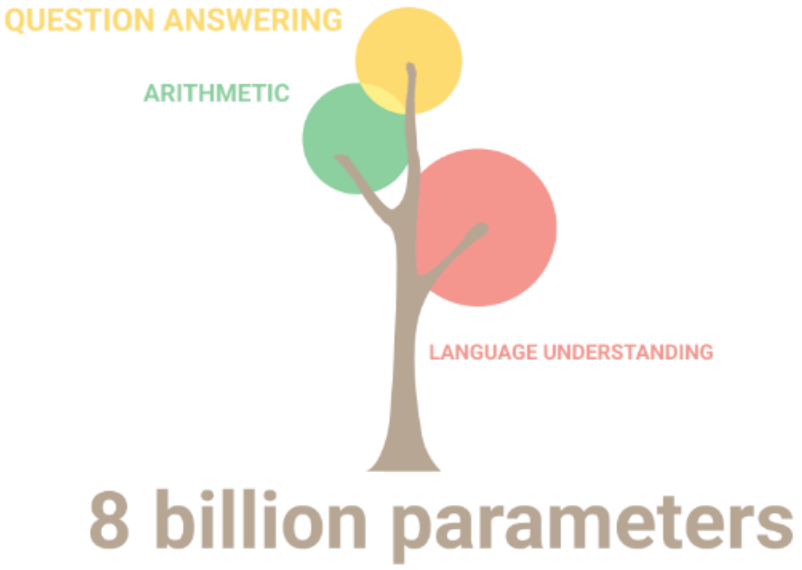
## In-context Learning

- Pretrain a language model based on "prompt" – demonstrates NLP tasks
- No need to update the model weights at all



# 1 – Background

**Model sizes are still growing?**

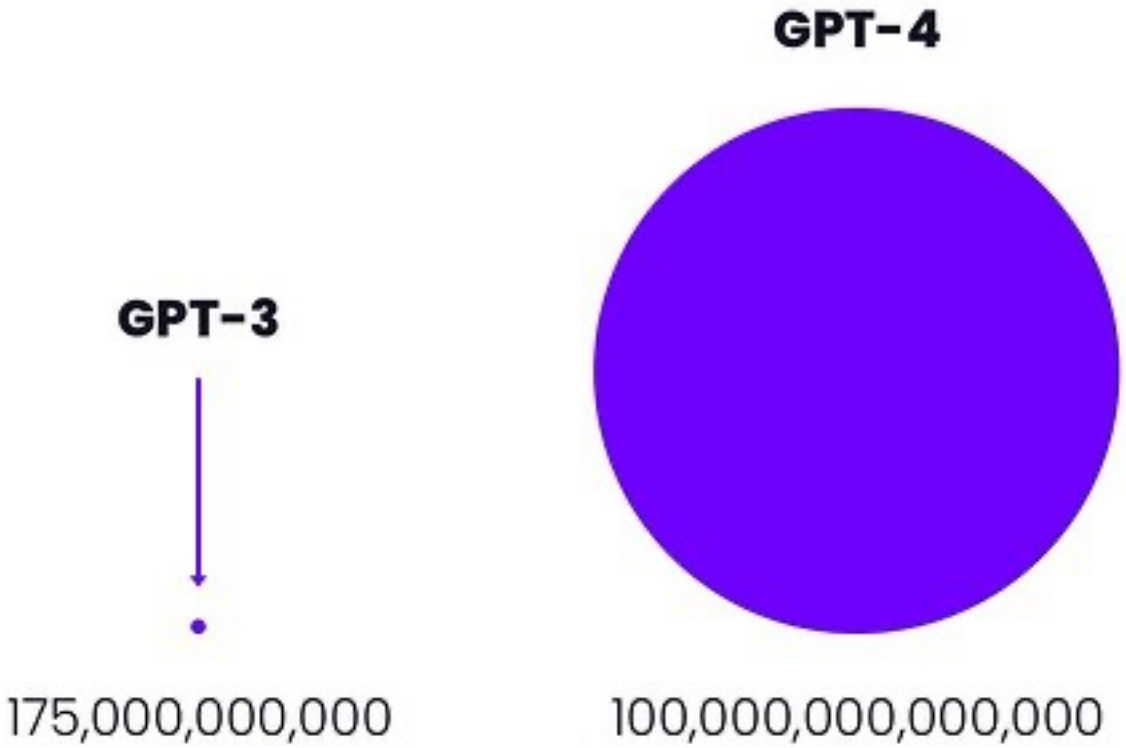



Source

# 1 – Background

**Model sizes are still growing?**

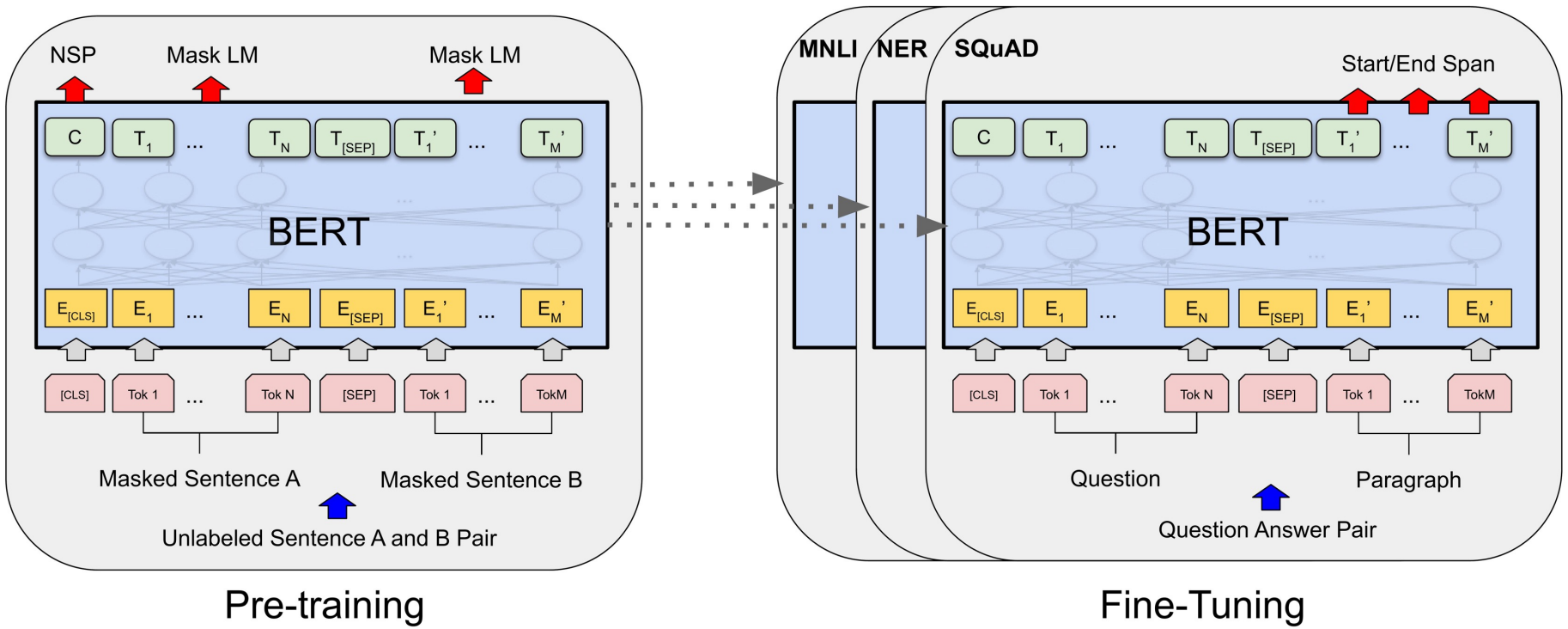
- Model size scales almost two orders of magnitude quicker than single-GPU memory



# 1 – Background

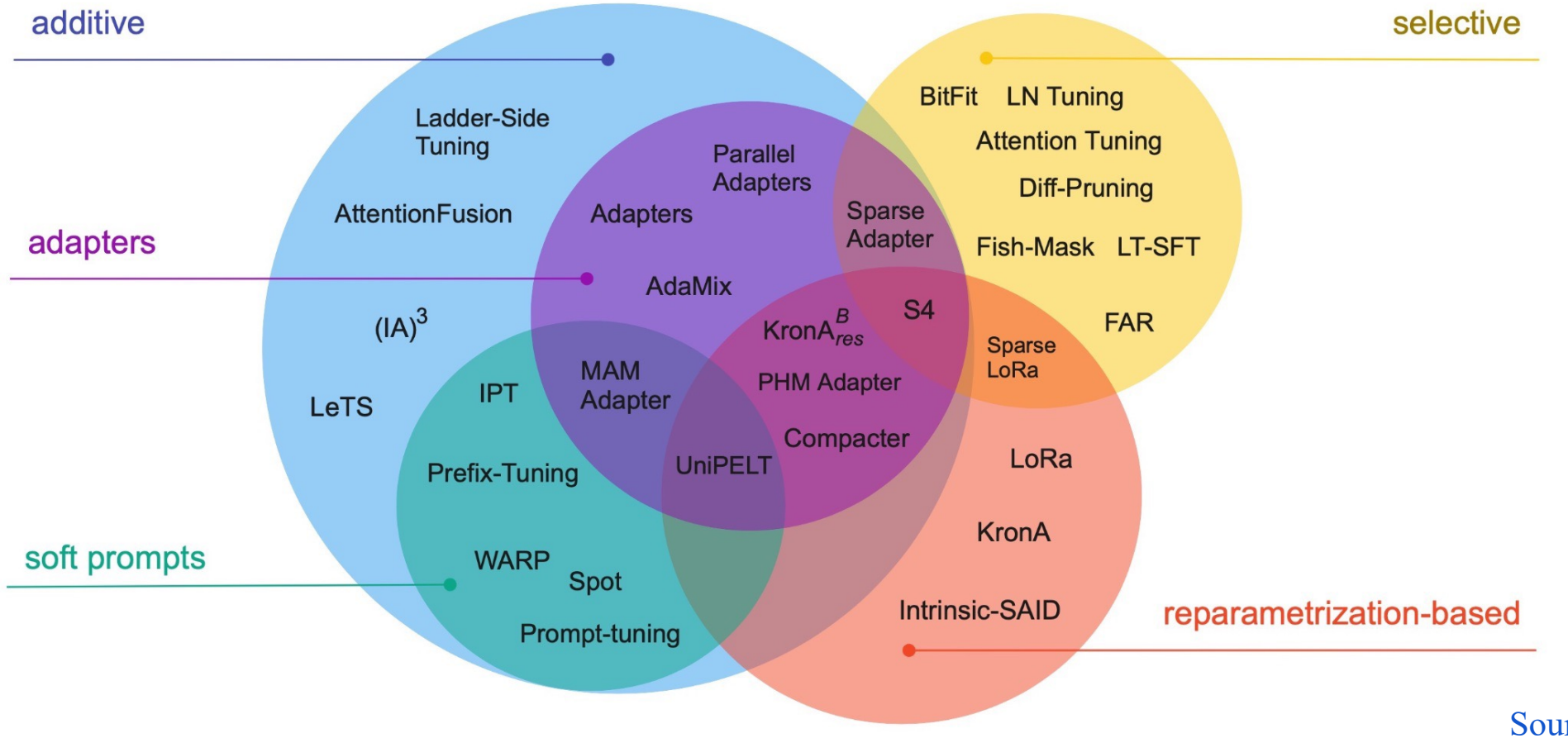
## Parameter-Efficient Fine-Tuning

- Standard fine-tuning: make a new copy of the model for each task
- Parameter-Efficiency: fine tuned a subset of the parameters for each task



# 1 – Background

## Parameter-Efficient Fine-Tuning

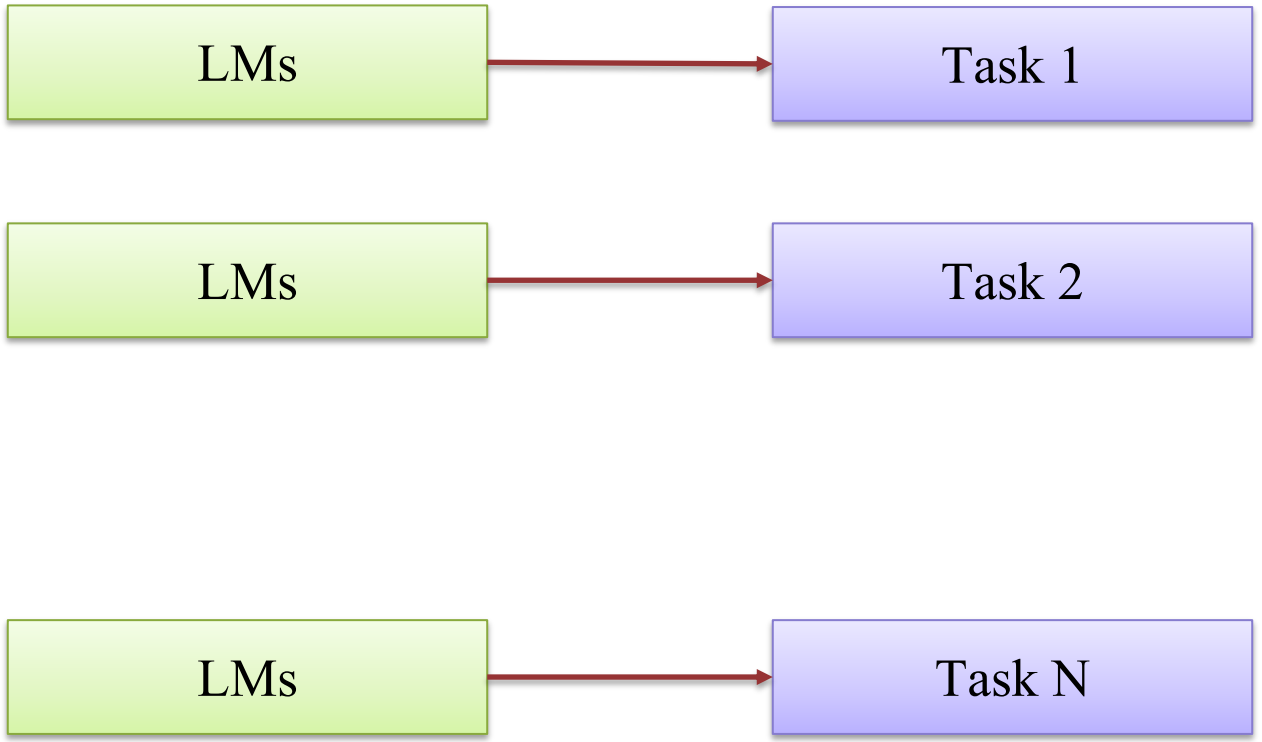



Source

# 2 – Adapter Fine Tuning

## Adapter Layers

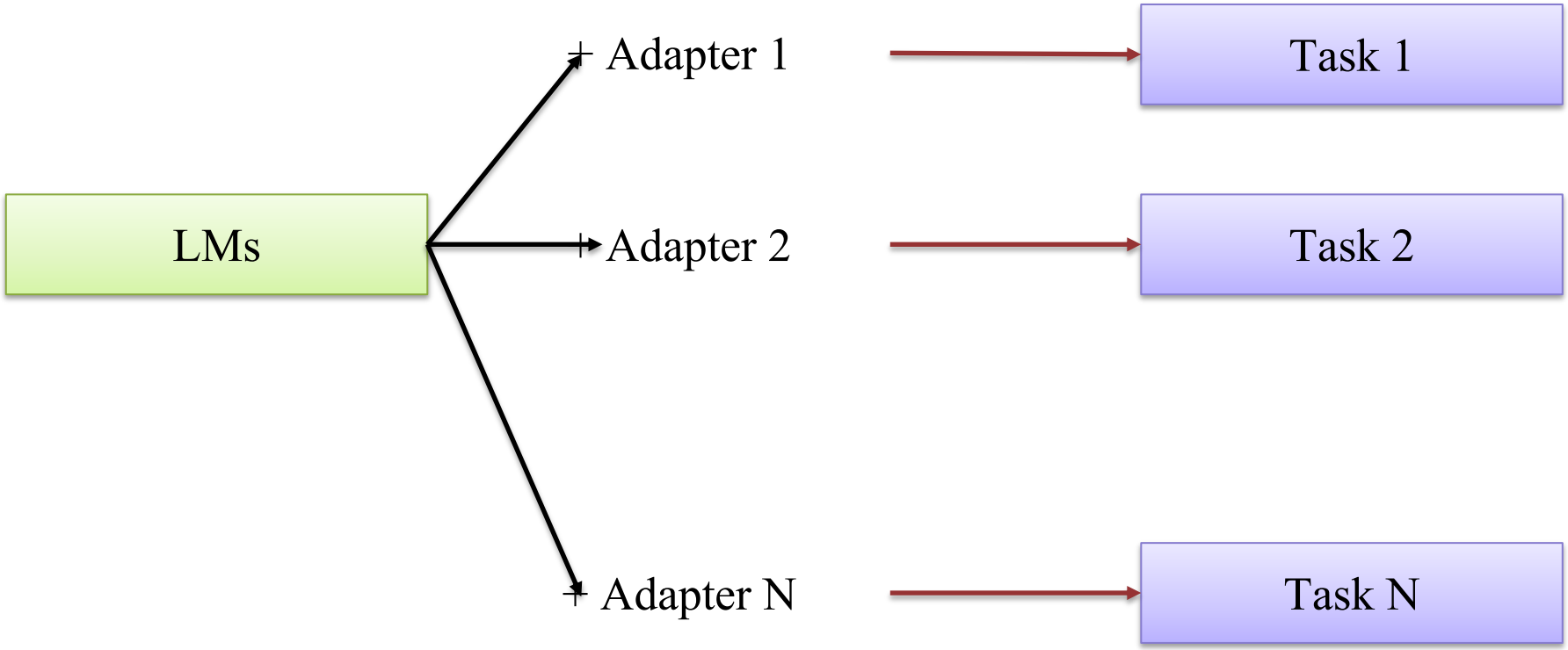
- Add a layer to adapt for downstream tasks



# 2 – Adapter Fine Tuning

## Adapter Layers

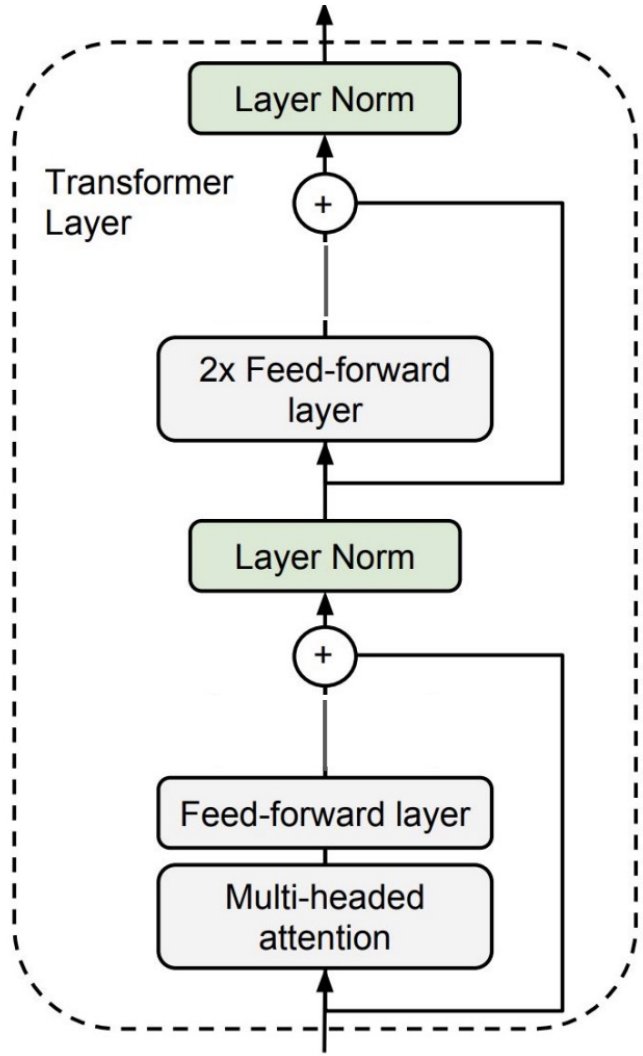
- Add a layer to adapt for downstream tasks



# 2 – Adapter Fine Tuning

## Adapter Layers

- Add adapter layers in between the transformer layers of a large model

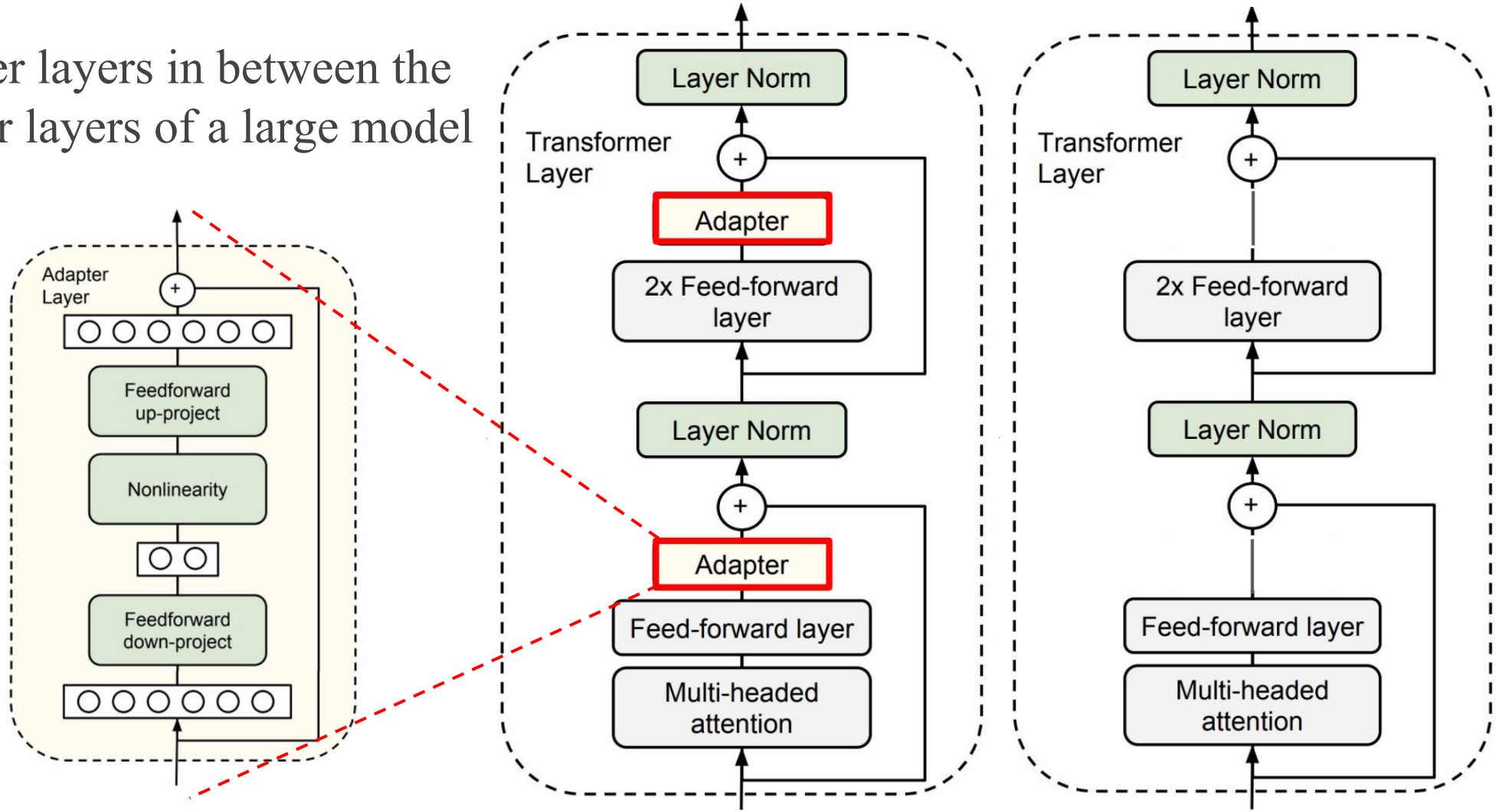



Source

# 2 – Adapter Fine Tuning

## Adapter Layers

- Add adapter layers in between the transformer layers of a large model

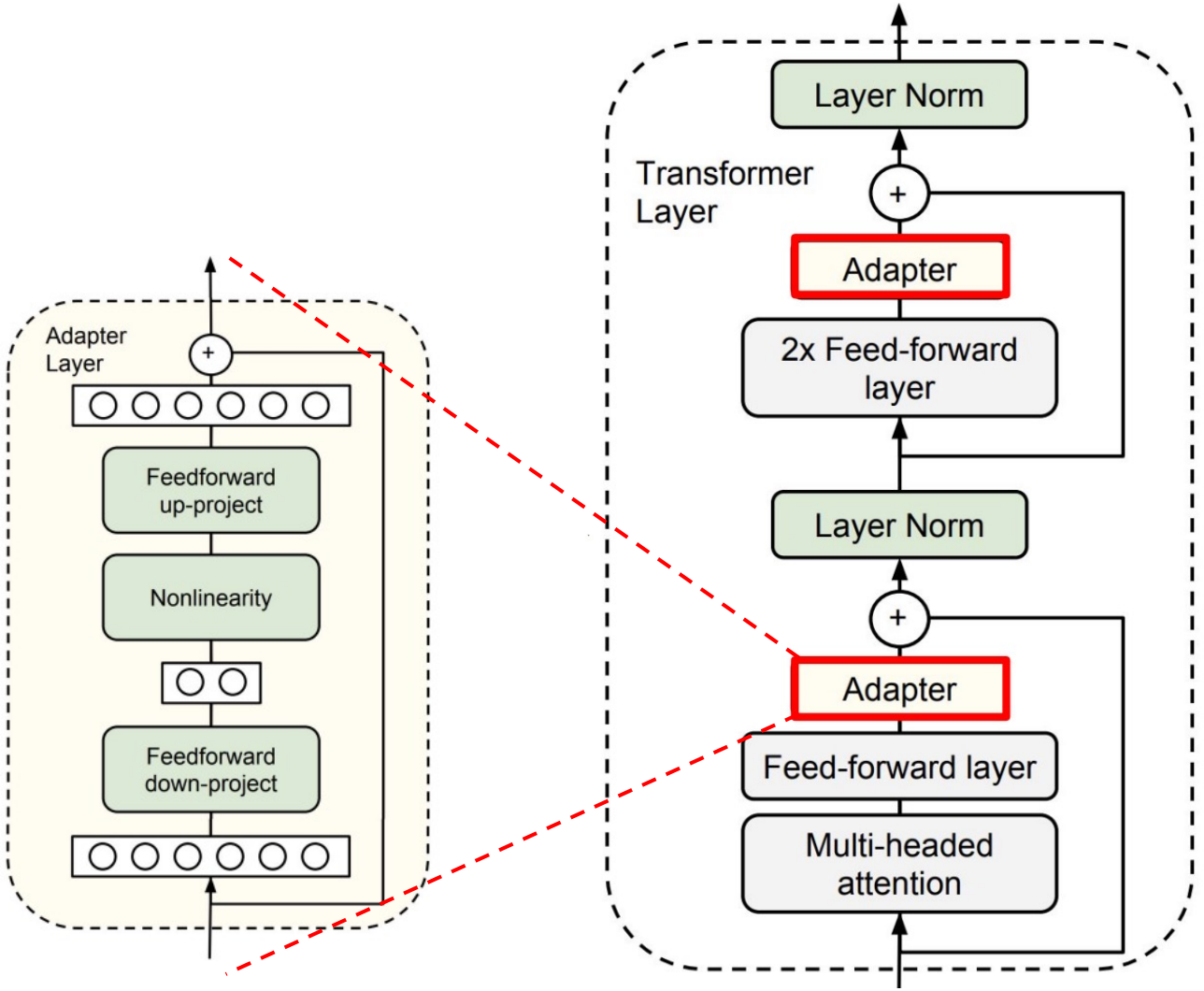



Source

# 2 – Adapter Fine Tuning

## Adapter Layers

- Add adapter layers in between the transformer layers of a large model
- During fine-tuning, fix the original model parameters and only tune the adapter layers

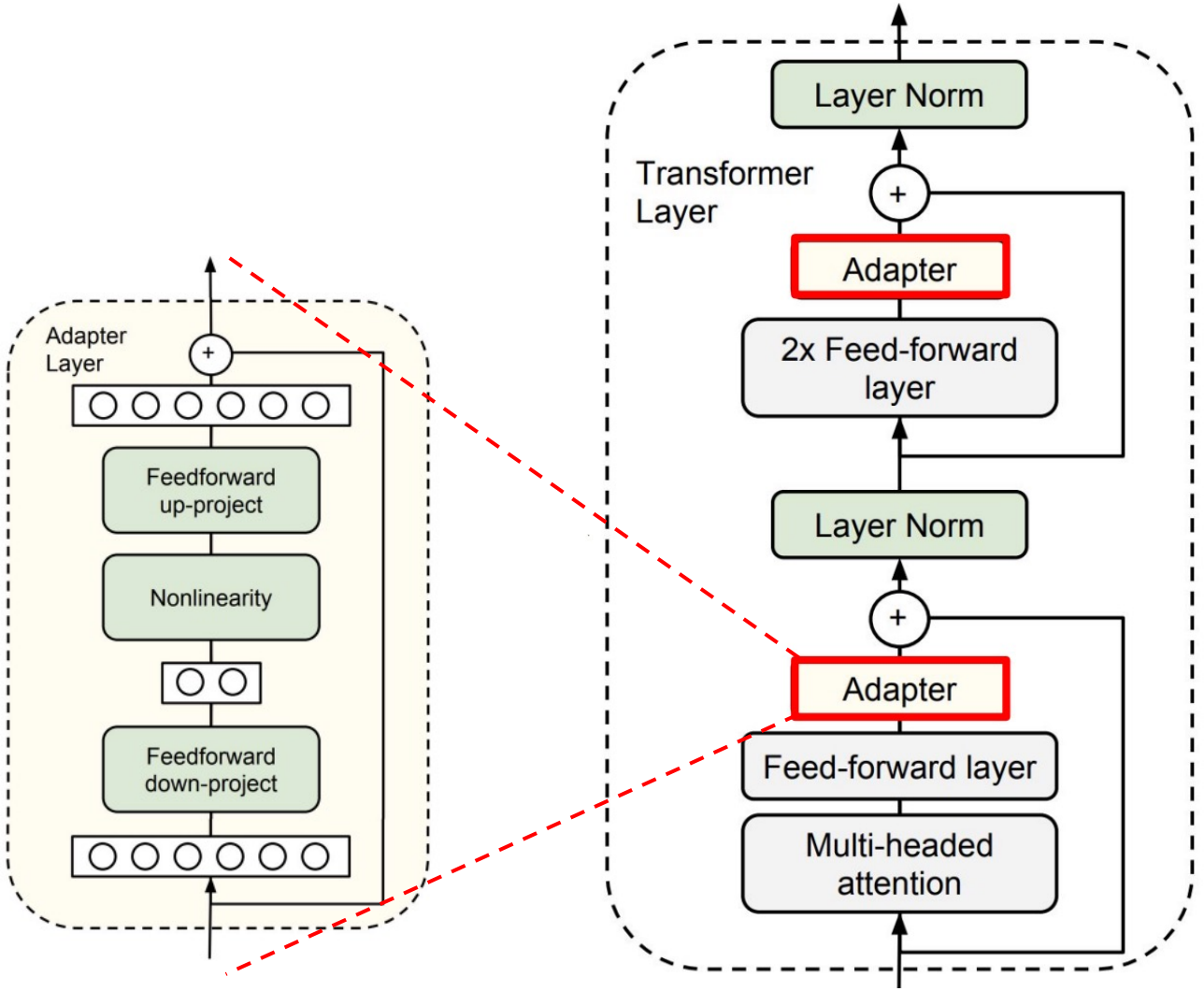



Source

# 2 – Adapter Fine Tuning

## Adapter Layers

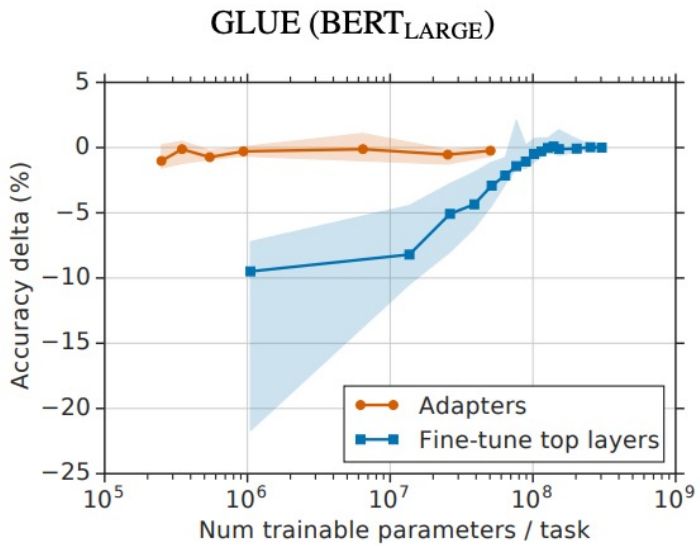
- Add adapter layers in between the transformer layers of a large model
- During fine-tuning, fix the original model parameters and only tune the adapter layers
- 3.6 % of parameters needed

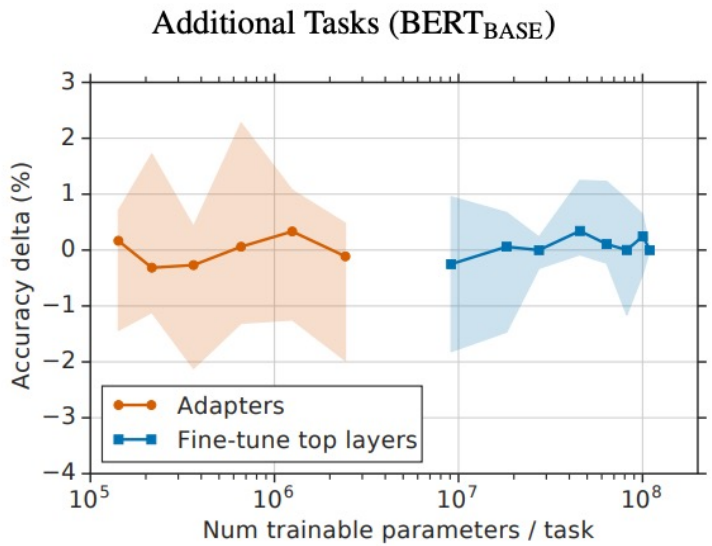



Source

# 2 – Adapter Fine Tuning

## Results on GLEU Benchmark



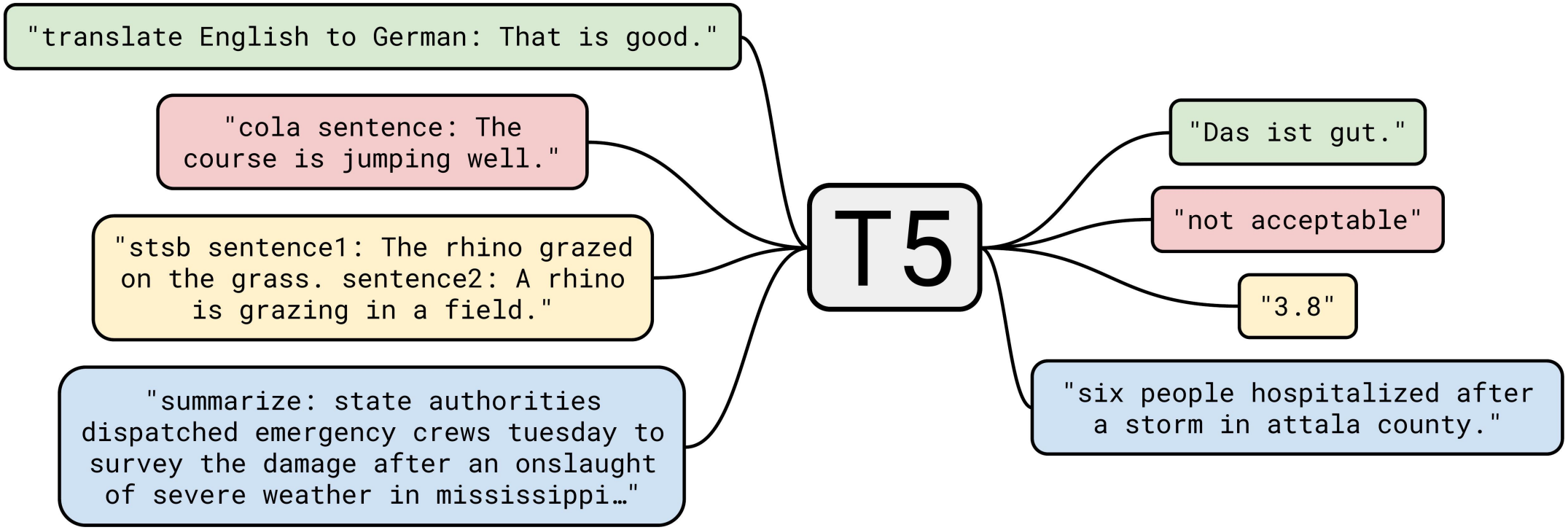

| | Total num params | Trained params / task | CoLA | SST | MRPC | STS-B | QQP | MNLI$_m$ | MNLI$_{mm}$ | QNLI | RTE | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BERT$_{LARGE}$ | 9.0× | 100% | 60.5 | 94.9 | 89.3 | 87.6 | 72.1 | 86.7 | 85.9 | 91.1 | 70.1 | 80.4 |
| Adapters (8-256) | 1.3× | 3.6% | 59.5 | 94.0 | 89.5 | 86.9 | 71.8 | 84.9 | 85.1 | 90.7 | 71.5 | 80.0 |
| Adapters (64) | 1.2× | 2.1% | 56.9 | 94.2 | 89.6 | 87.3 | 71.8 | 85.3 | 84.6 | 91.4 | 68.8 | 79.6 |

Source

# 3 – Prefix Tuning

## Prompt Design

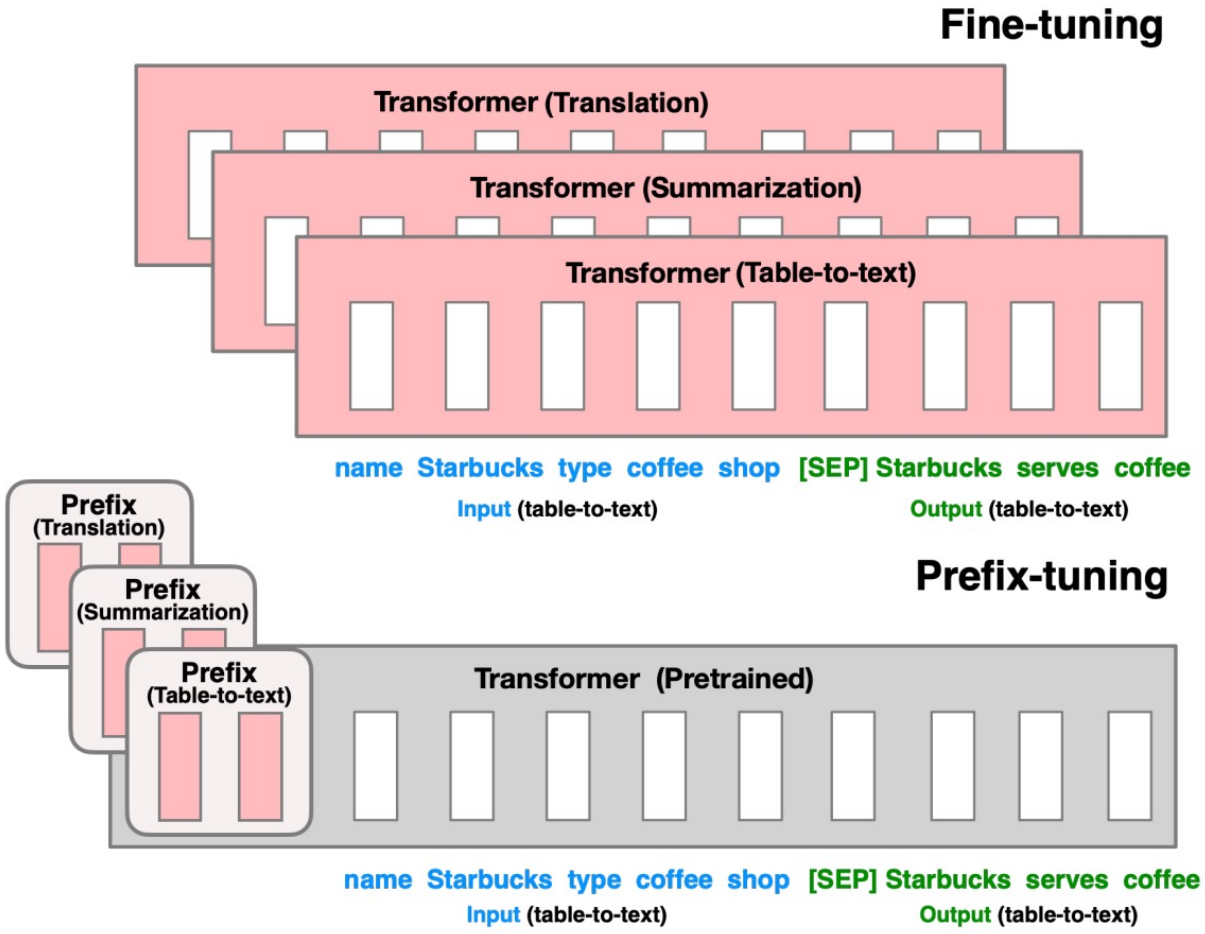
- For prompt design, the discrete prompts is optimized manually
- Optimization in discrete space is hard!



# 3 – Prefix Tuning

**Prefix-Tuning: Optimizing Continuous Prompts for Generation**

- Optimization in the continuous embedding space
- Learn an optimal prefix for each task

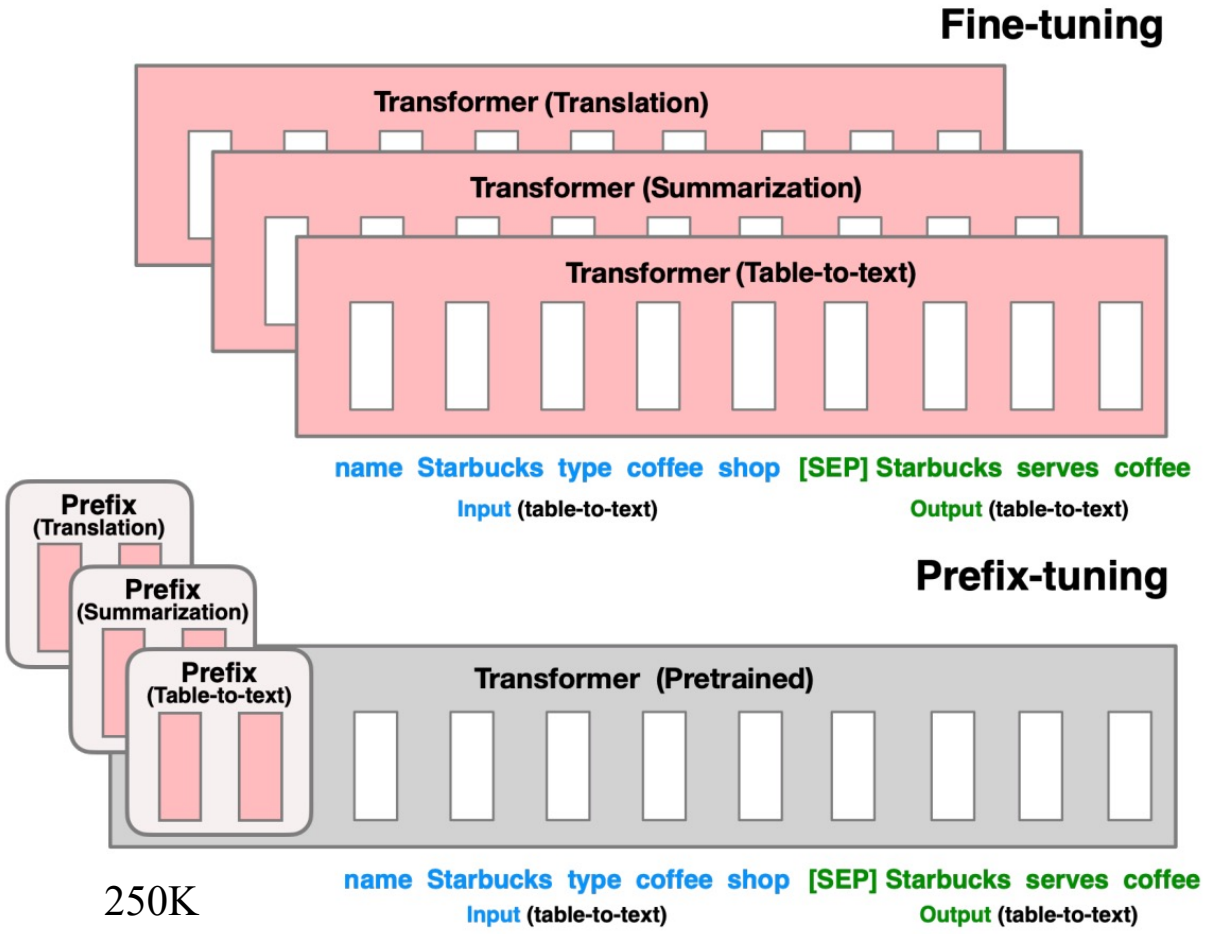



Source

# 3 – Prefix Tuning

**Prefix-Tuning: Optimizing Continuous Prompts for Generation**

- Optimization in the continuous embedding space
- Learn an optimal prefix for each task
- Only 0.1% of parameters need to be tuned



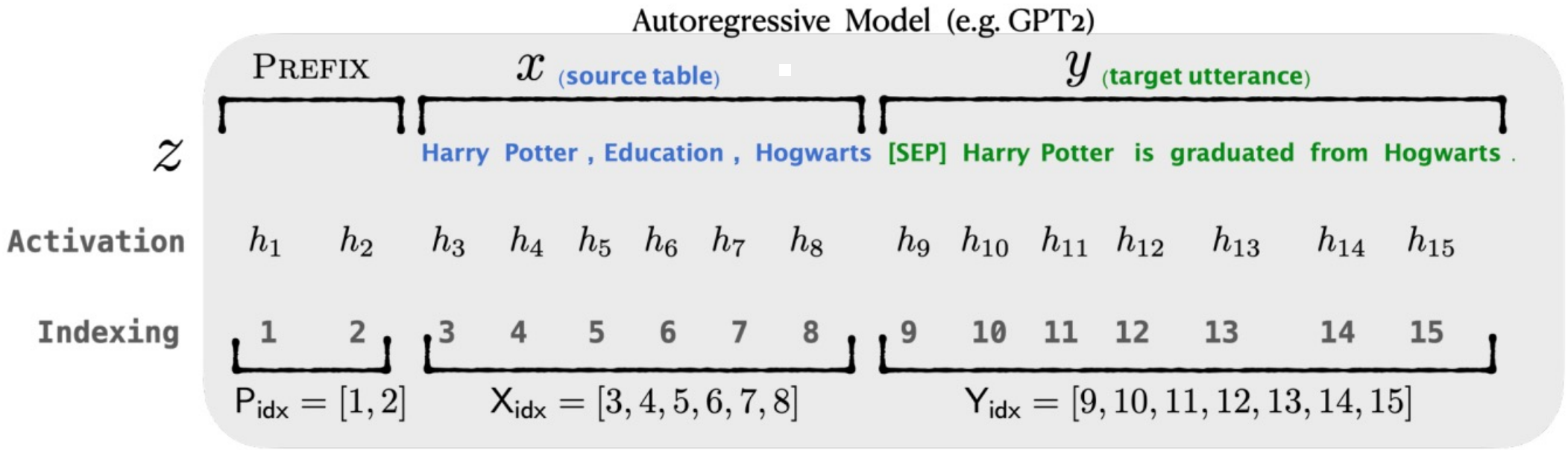

250K parameters

Source

# 3 – Prefix Tuning

**Prefix-Tuning: Optimizing Contiunous Prompts for Generation**

- Prefix-Tuning using an autoregressive LM

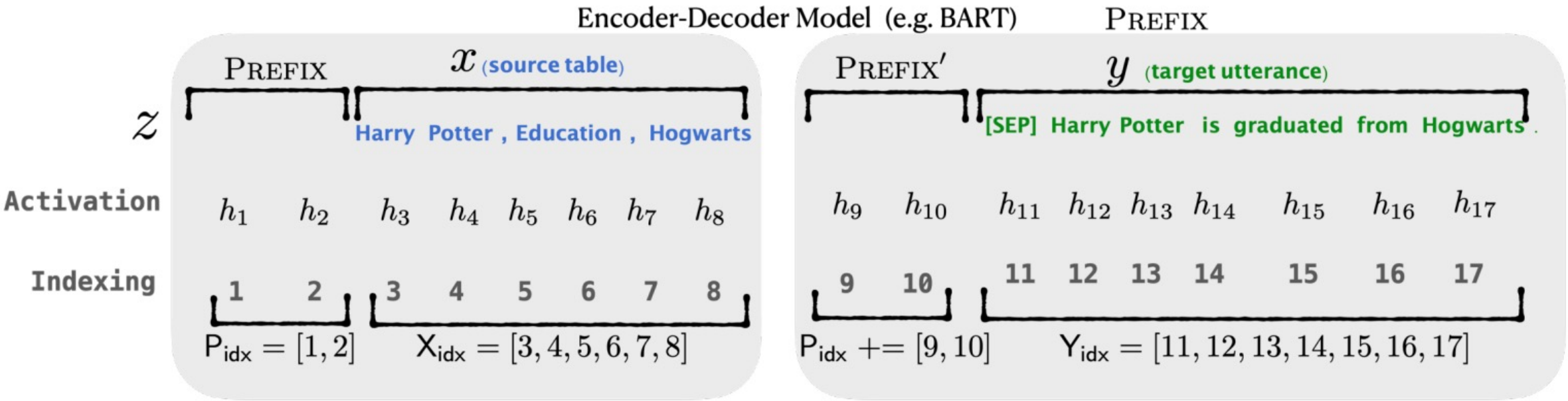



Source

# 3 – Prefix Tuning

**Prefix-Tuning: Optimizing Continuous Prompts for Generation**

- Prefix-Tuning using an encoder-decoder model

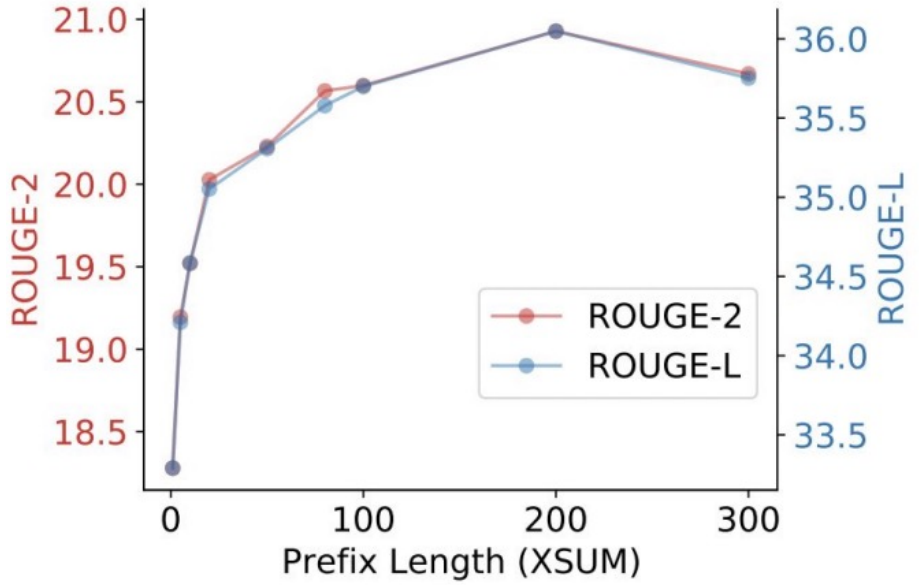


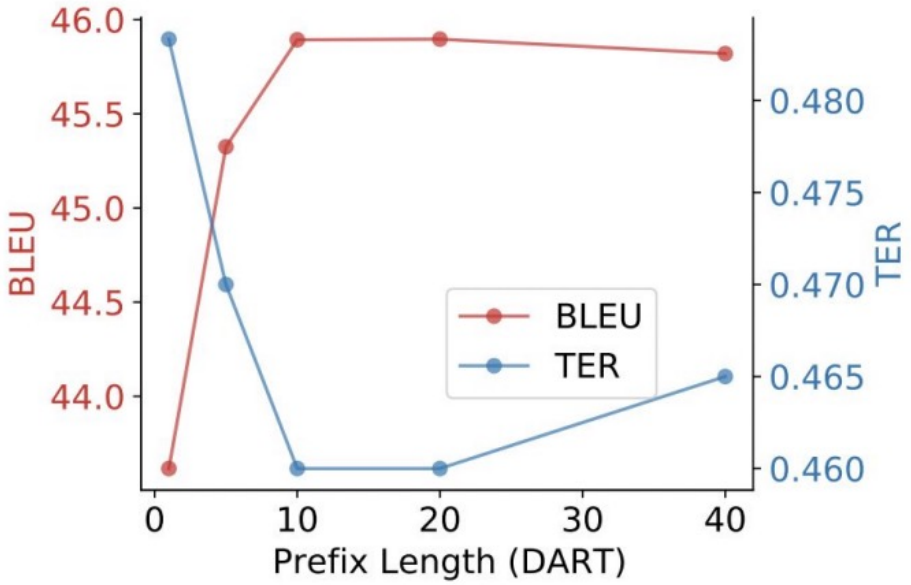

Source

# 3 – Prefix Tuning

## Prefix Length

- As the tunable prefix-length increases, performance increases, with diminishing returns
- Optimal length for table to text is 10 tokens, for summarization it seems closer to 200 tokens




Source

# 3 – Prefix Tuning

## Prefix Tuning and Infix Tuning

- Instead of tuning the prefix, tune a portion at the end of the input and before the output
- Infix tuning is worse than prefix tuning, since input embeddings cannot attend to infix

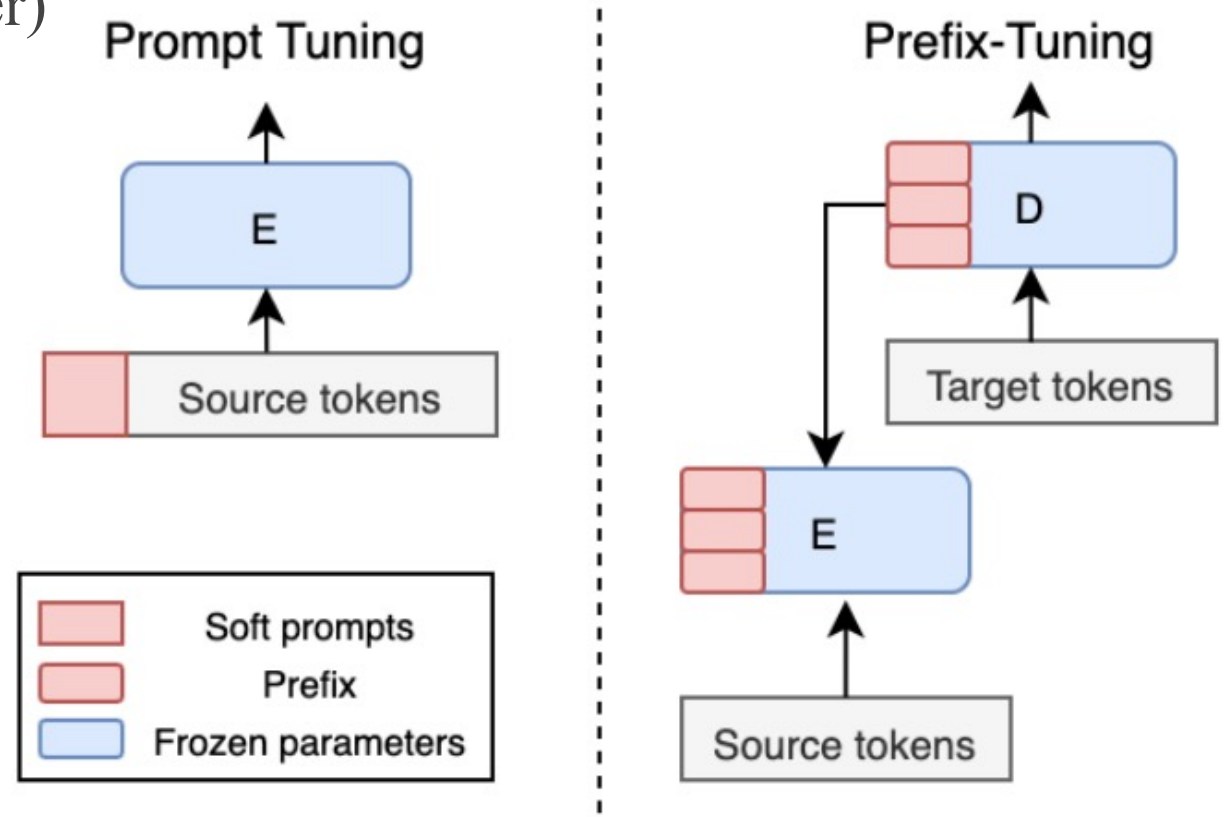
| | E2E | | | | |
|---|---|---|---|---|---|
| | BLEU | NIST | MET | ROUGE | CIDEr |
| PREFIX | 69.7 | 8.81 | 46.1 | 71.4 | 2.49 |
| Embedding-only: EMB-{PrefixLength} | | | | | |
| EMB-1 | 48.1 | 3.33 | 32.1 | 60.2 | 1.10 |
| EMB-10 | 62.2 | 6.70 | 38.6 | 66.4 | 1.75 |
| EMB-20 | 61.9 | 7.11 | 39.3 | 65.6 | 1.85 |
| Infix-tuning: INFIX-{PrefixLength} | | | | | |
| INFIX-1 | 67.9 | 8.63 | 45.8 | 69.4 | 2.42 |
| INFIX-10 | 67.2 | 8.48 | 45.8 | 69.9 | 2.40 |
| INFIX-20 | 66.7 | 8.47 | 45.8 | 70.0 | 2.42 |

Source

# 4 - Prompt Tuning

**From prefix-tuning to prompt-tuning**

- Prefix-tuning learn a sequence of prefixes (are prepended at every transformer layer)
- Prompt-tuning uses a single prompt representation is prepended to the embedded input



# 4 - Prompt Tuning

## Prompt Tuning

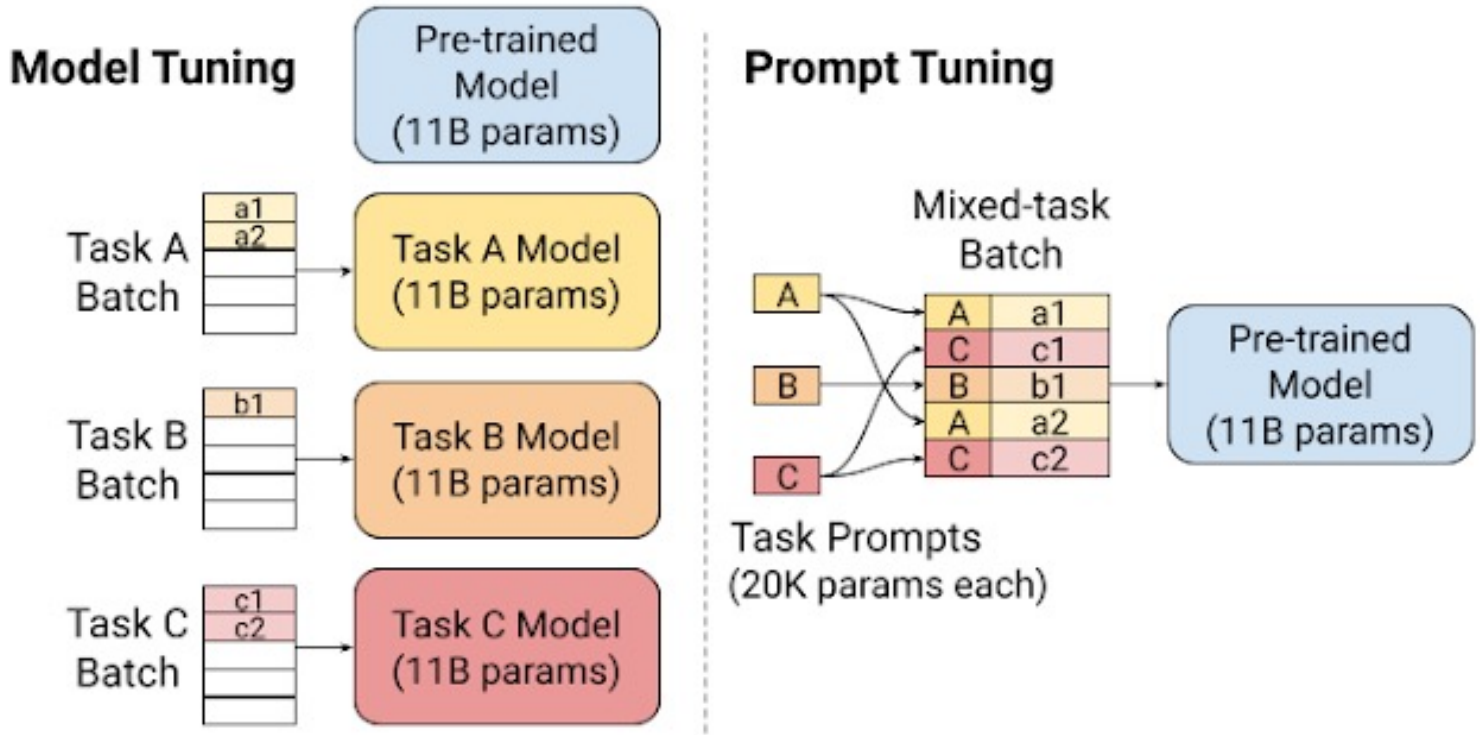
- Prepend virtual tokens to input
- Pre-trained

$$P_{r_\theta}(Y|X)$$

- Fine-tuned

$$P_{r_{\theta;\theta_p}}(Y|[P;X])$$

fixed learnable



# 4 - Prompt Tuning

## Design Decision

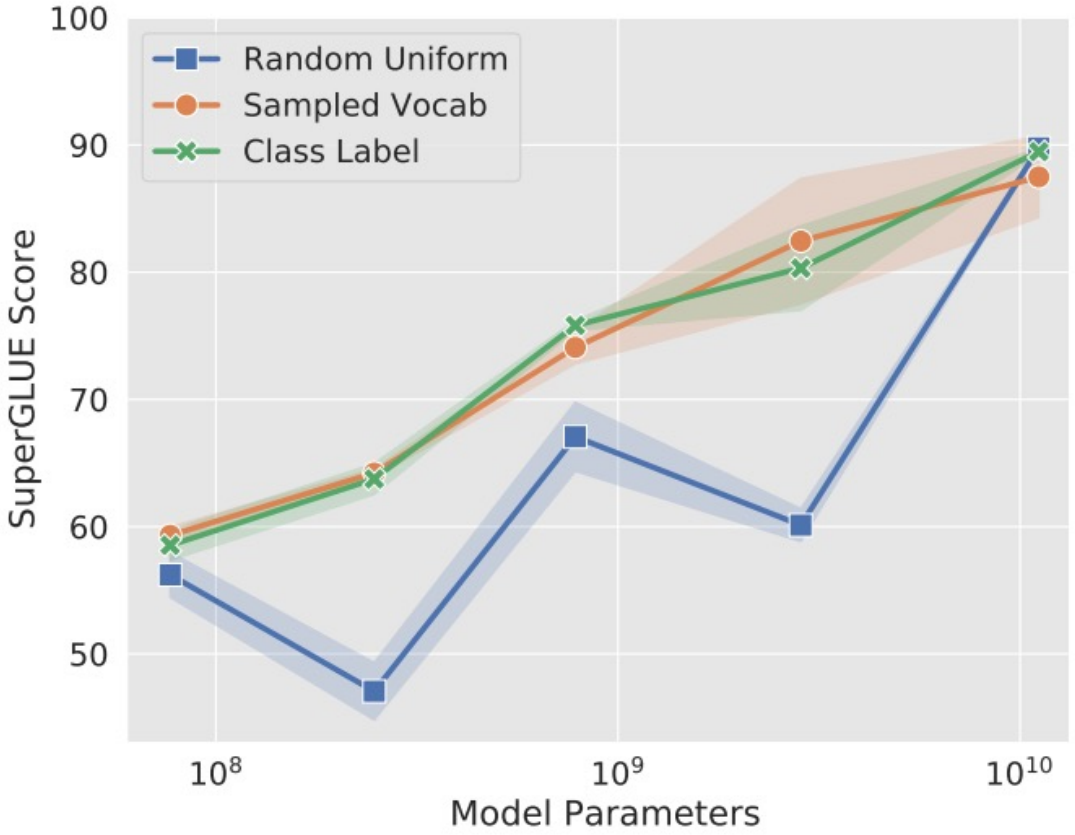
- Prompt initialization method
- Prompt length
- Pre-training method
- LM adaptation steps

Source

# 4 - Prompt Tuning

## Design Decision: Prompt Initilization

- Random initialization
- Sampled vocabulary: initialize each prompt token to an embedding drawn from the model's vocabulary
- Class label: initialize the prompt with embeddings that enumerate the output classes

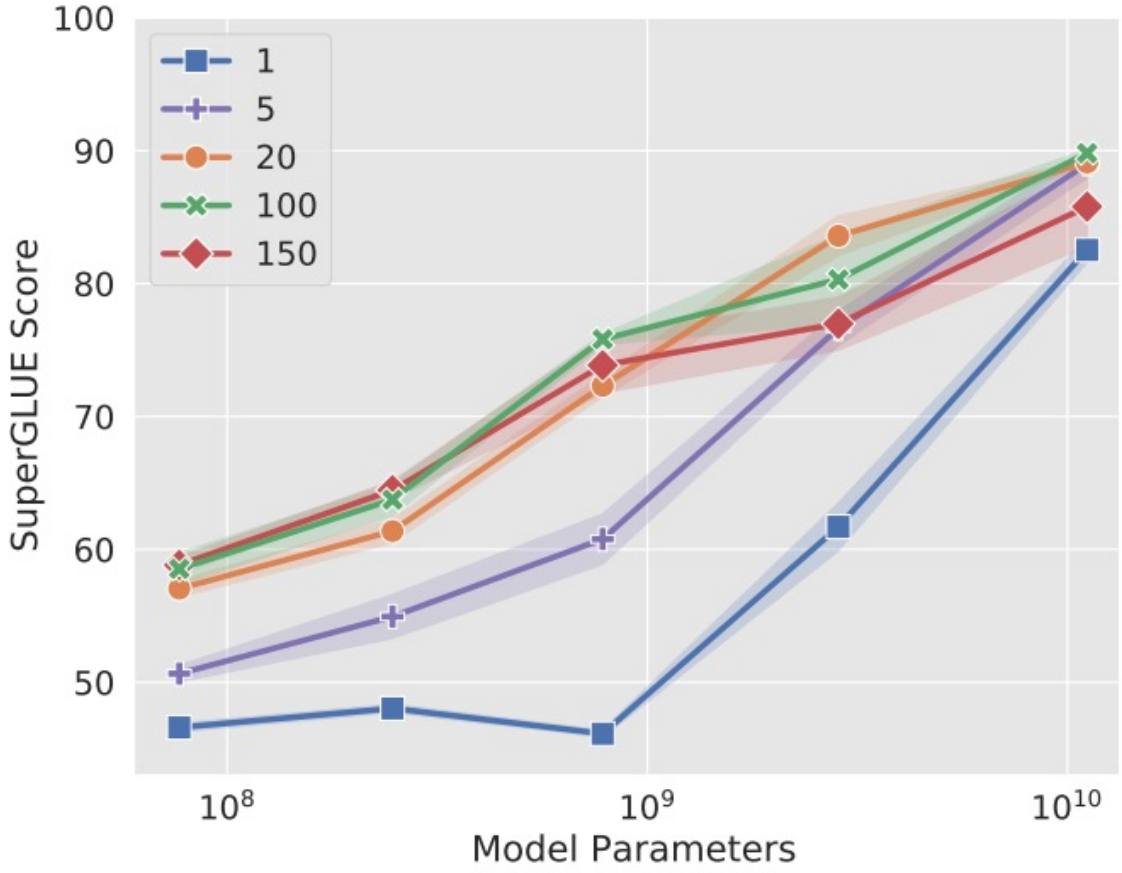



Source

# 4 - Prompt Tuning

## Design Decision: Prompt Length

- The shorter the prompt, the fewer parameters must be tuned

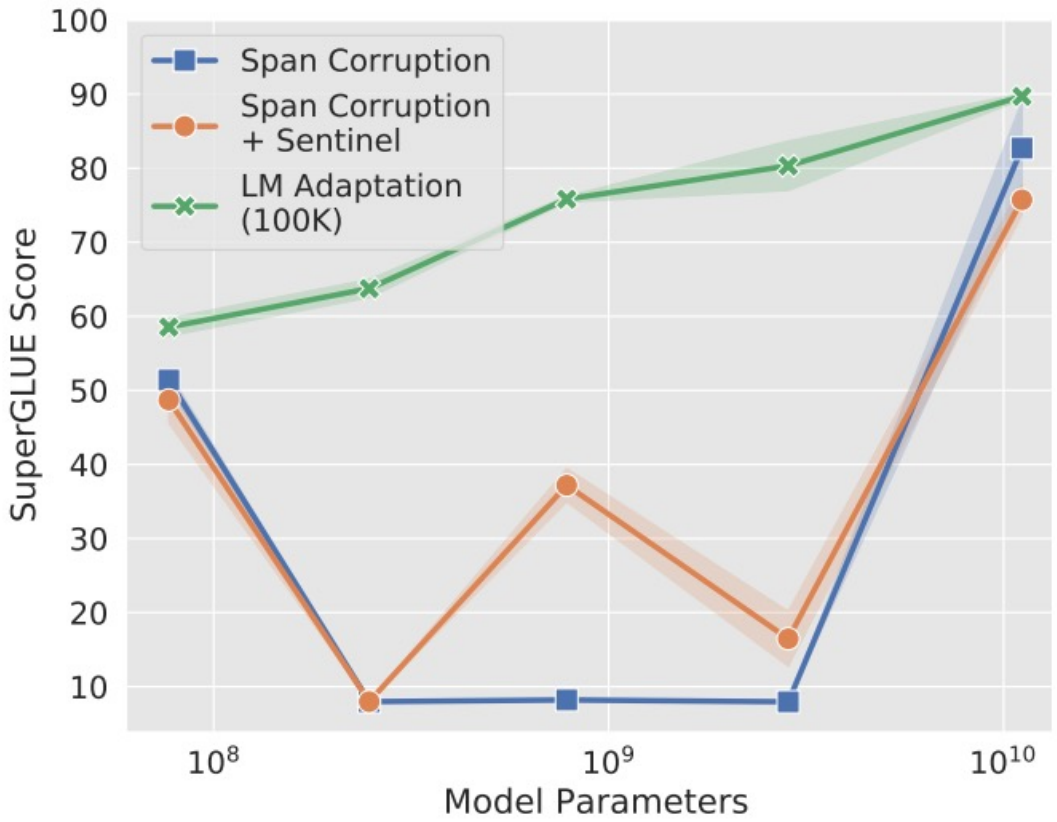



Source

# 4 - Prompt Tuning

**Design Decision: Pre-training Method**

- Span Corruption: reconstructing masked span in the input text
- Span Corruption + Sentinel: prepend all downstream targets with a sentinel
- ” LM Adaptation”: as T5 objective function

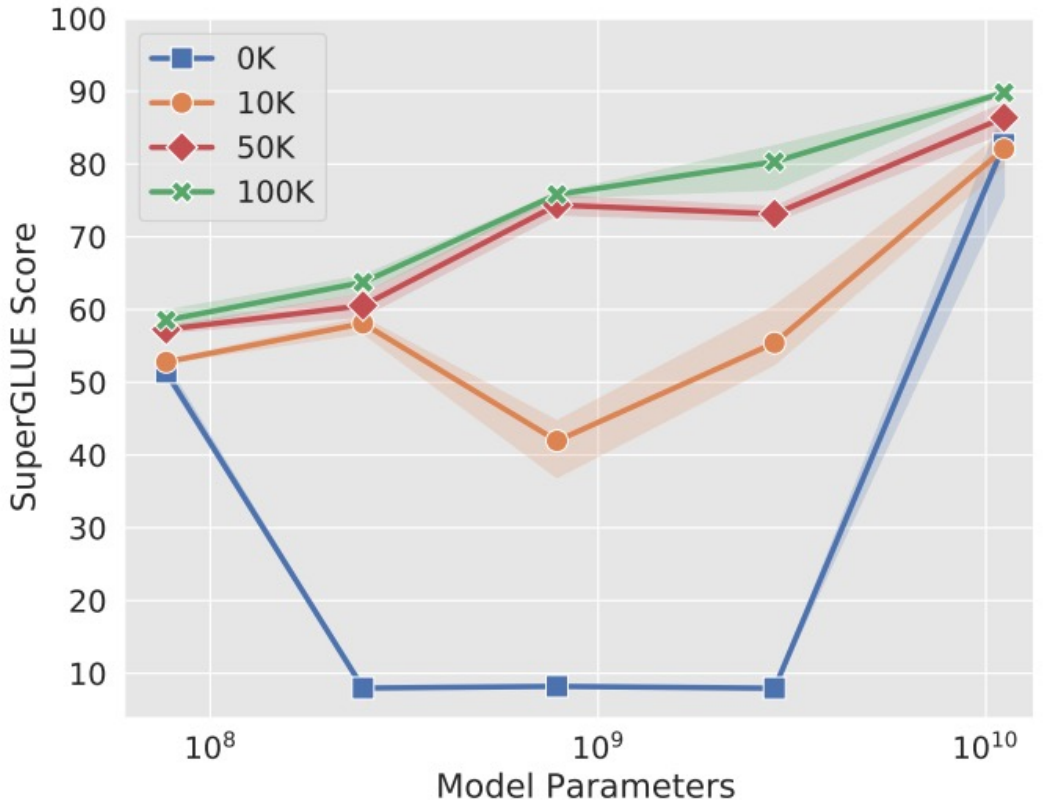



Source

# 4 - Prompt Tuning

**Design Decision: Pre-training Method**

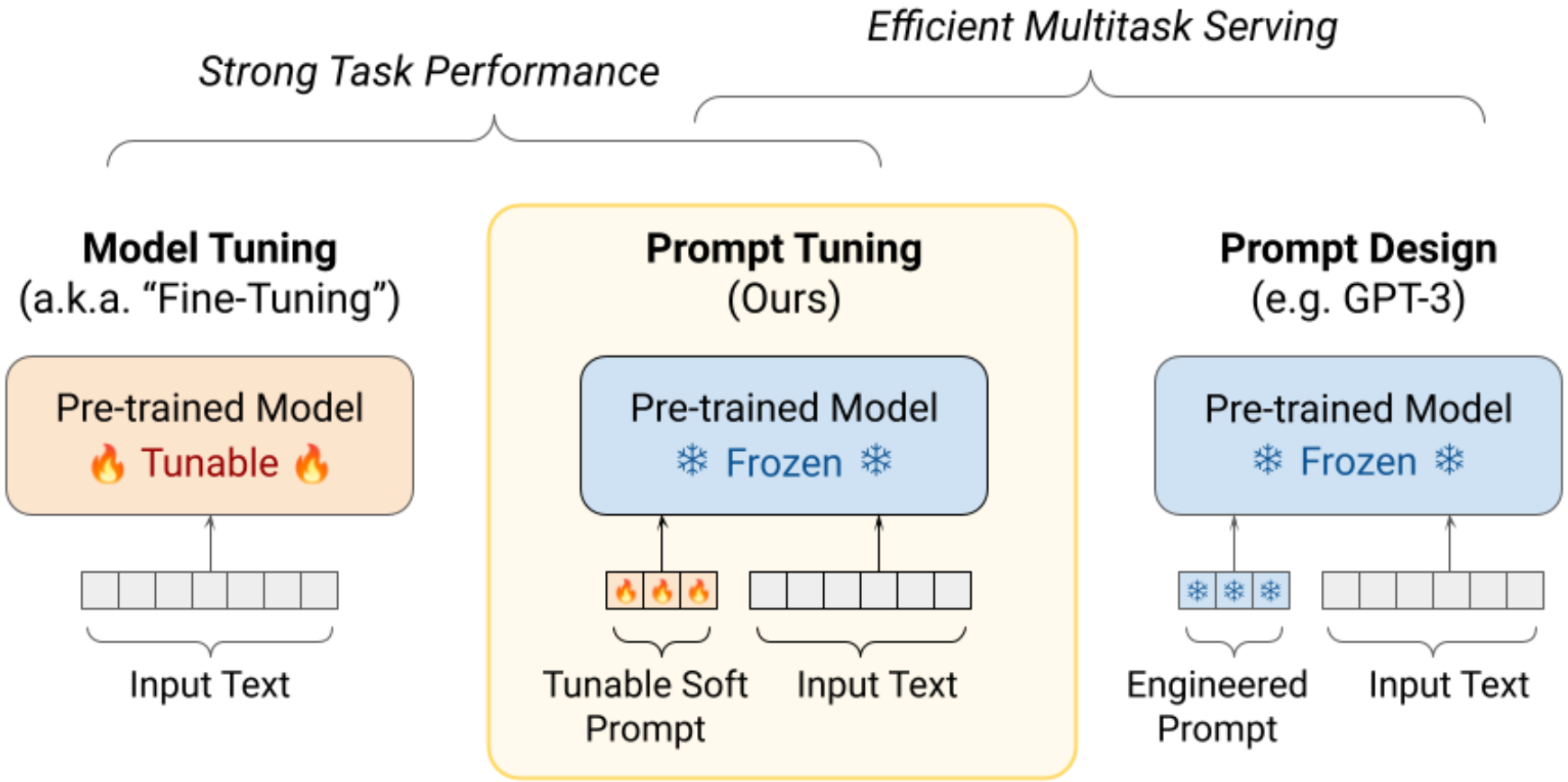
- Longer adaptation provides additional gains, up to 100K steps
- At the largest model size, the gains from adaptation are quite modest




Source

# 4 - Prompt Tuning

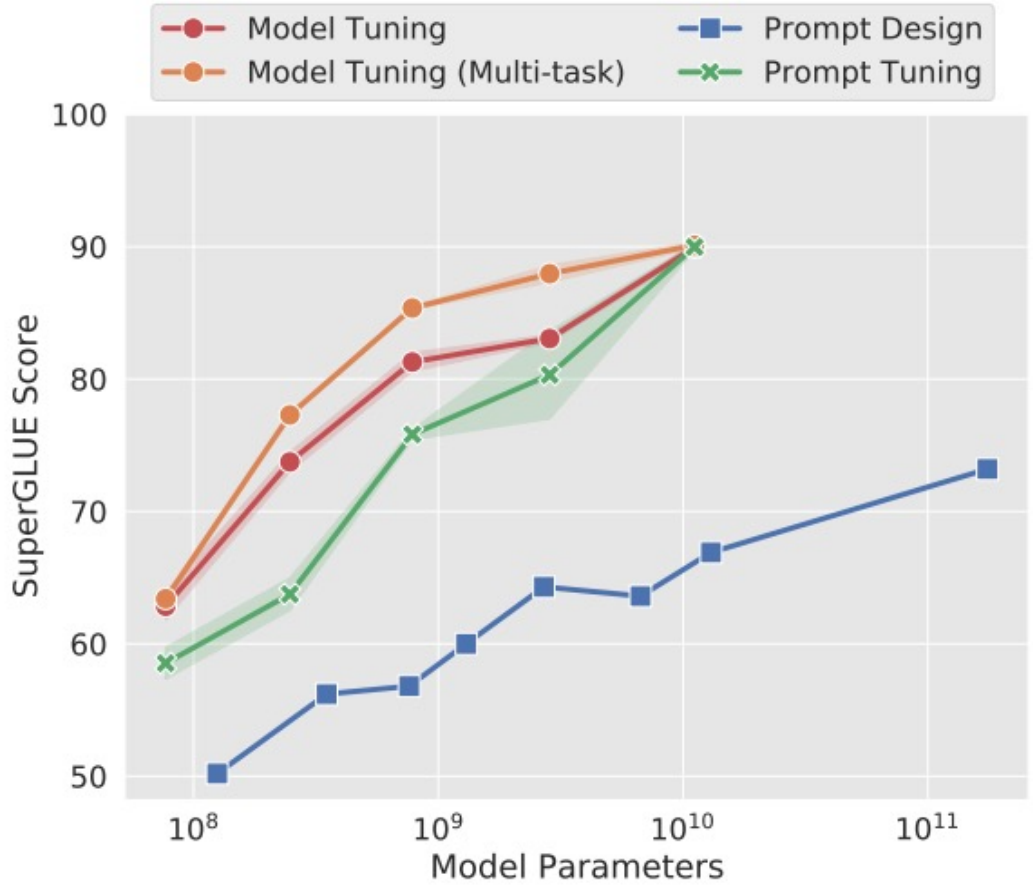
**Comparing: model-tuning, prompt-tuning, prompt-design**




Source

# 4 - Prompt Tuning

**Comparing: model-tuning, prompt-tuning, prompt-design**



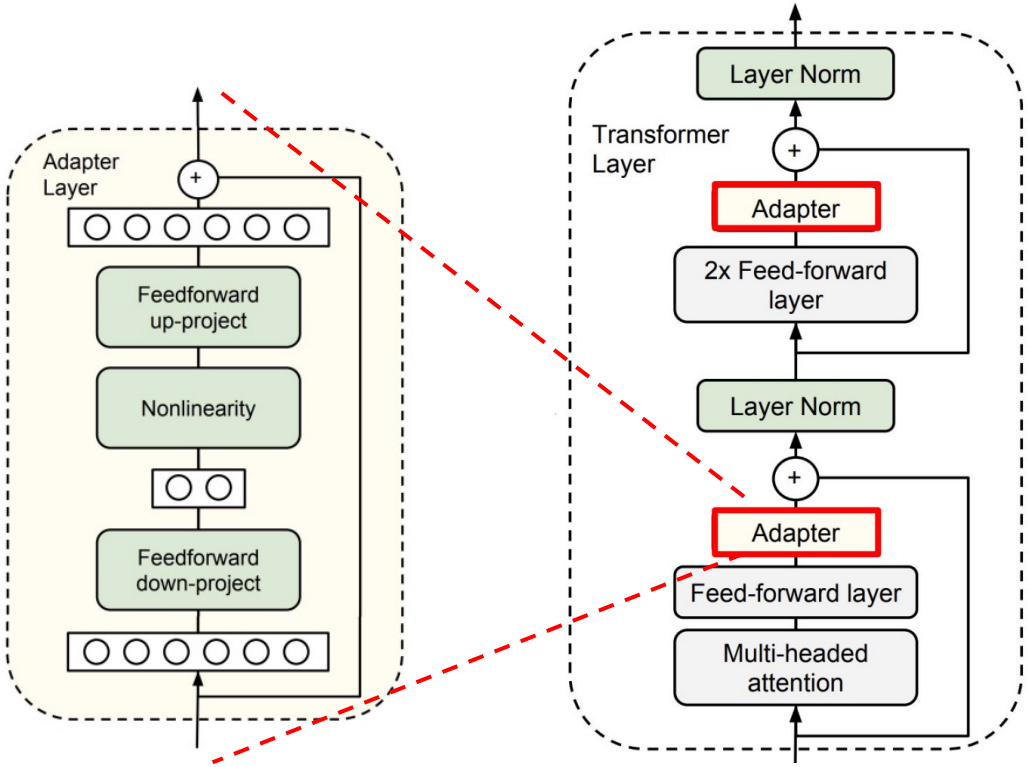

| Dataset | Domain | Model | Prompt | Δ |
|---|---|---|---|---|
| SQuAD | Wiki | 94.9 ±0.2 | 94.8 ±0.1 | −0.1 |
| TextbookQA | Book | 54.3 ±3.7 | **66.8** ±2.9 | +12.5 |
| BioASQ | Bio | 77.9 ±0.4 | **79.1** ±0.3 | +1.2 |
| RACE | Exam | 59.8 ±0.6 | **60.7** ±0.5 | +0.9 |
| RE | Wiki | 88.4 ±0.1 | **88.8** ±0.2 | +0.4 |
| DuoRC | Movie | **68.9** ±0.7 | 67.7 ±1.1 | −1.2 |
| DROP | Wiki | **68.9** ±1.7 | 67.1 ±1.9 | −1.8 |

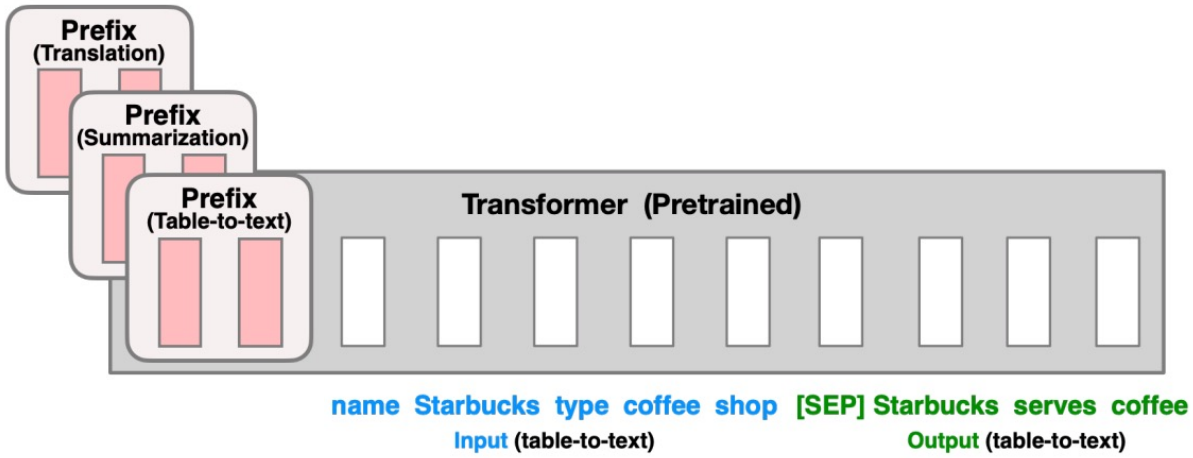
Source

# 5 – Low-Rank Adaptation

## Adapter Tuning and Prefix Tuning

- Adapter Tuning: High-quality, but adds latency
- No latency, but suboptimal quality

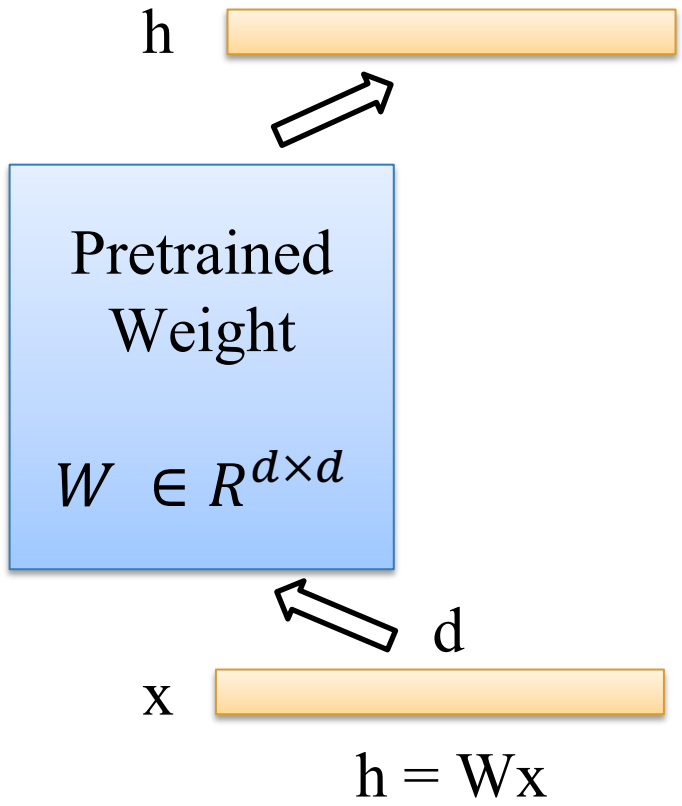


# 5 – Low-Rank Adaptation

## Low-Rank Adaptation (LoRA)

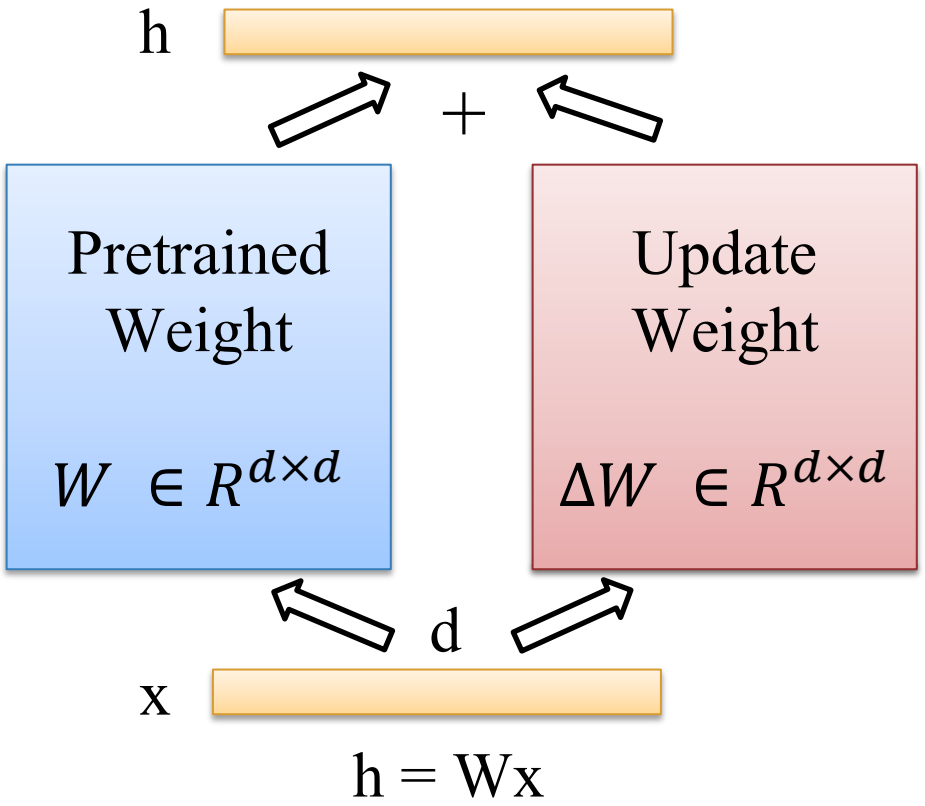
- Freezes the pretrained model weights and injects trainable rank decomposition matrices into each layer of the Transformer architecture



# 5 – Low-Rank Adaptation

**Low-Rank Adaptation (LoRA)**

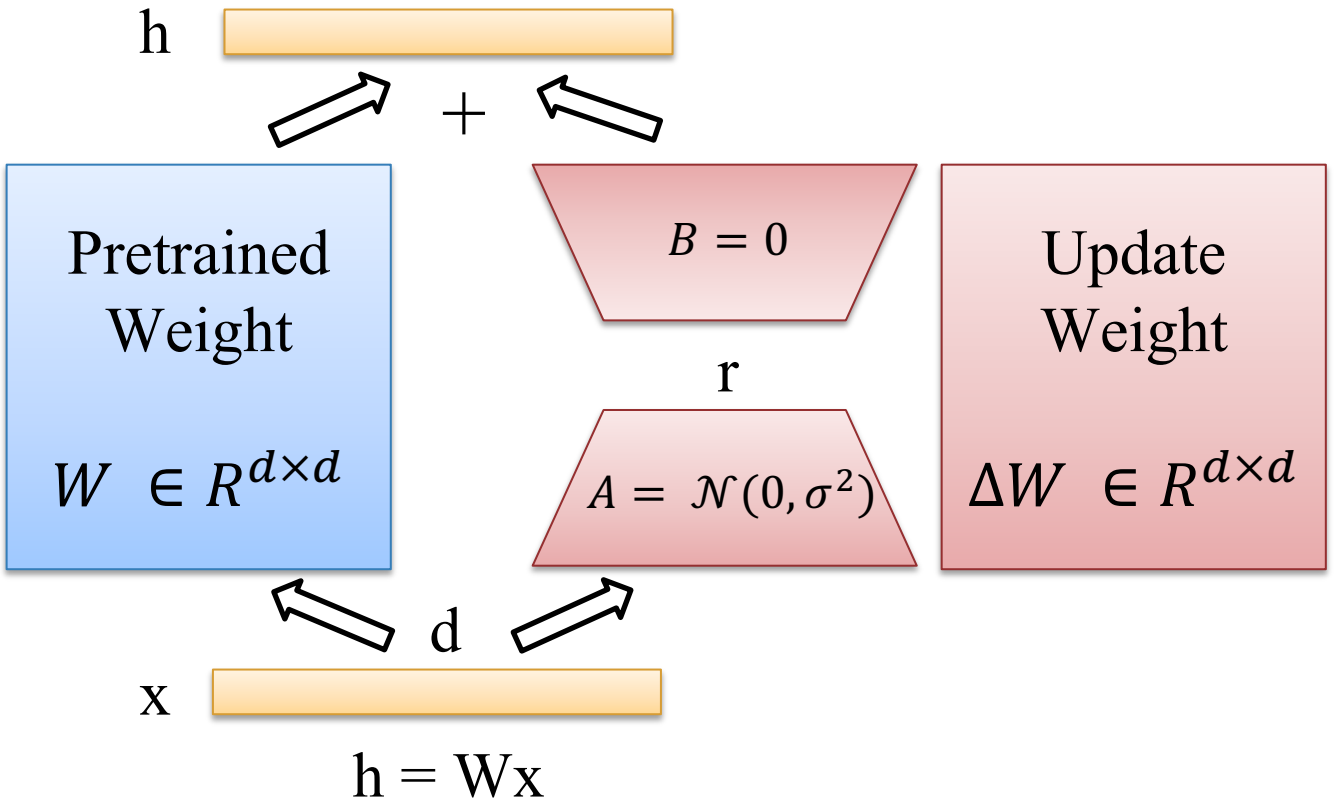
- Freezes the pretrained model weights and injects trainable rank decomposition matrices into each layer of the Transformer architecture



# 5 – Low-Rank Adaptation

**Low-Rank Adaptation (LoRA)**

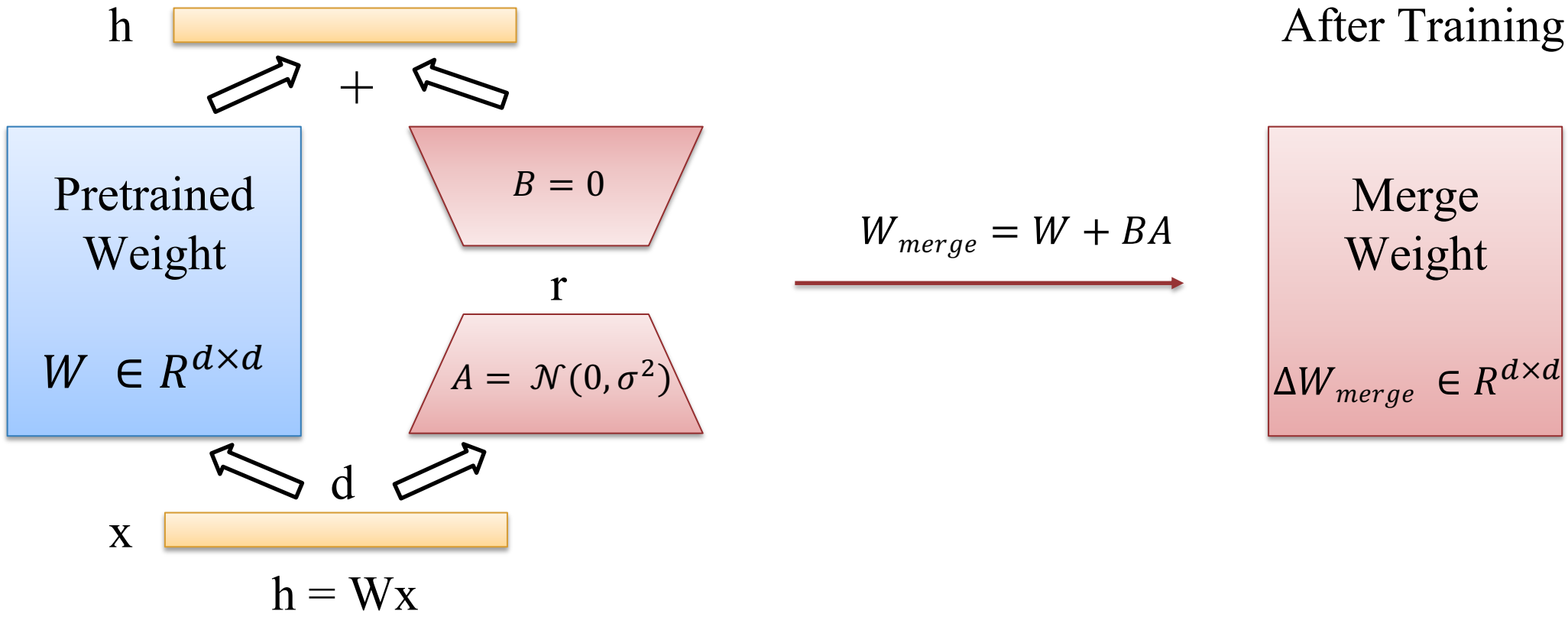
- Freezes the pretrained model weights and injects trainable rank decomposition matrices into each layer of the Transformer architecture



# 5 – Low-Rank Adaptation

## Low-Rank Adaptation (LoRA)

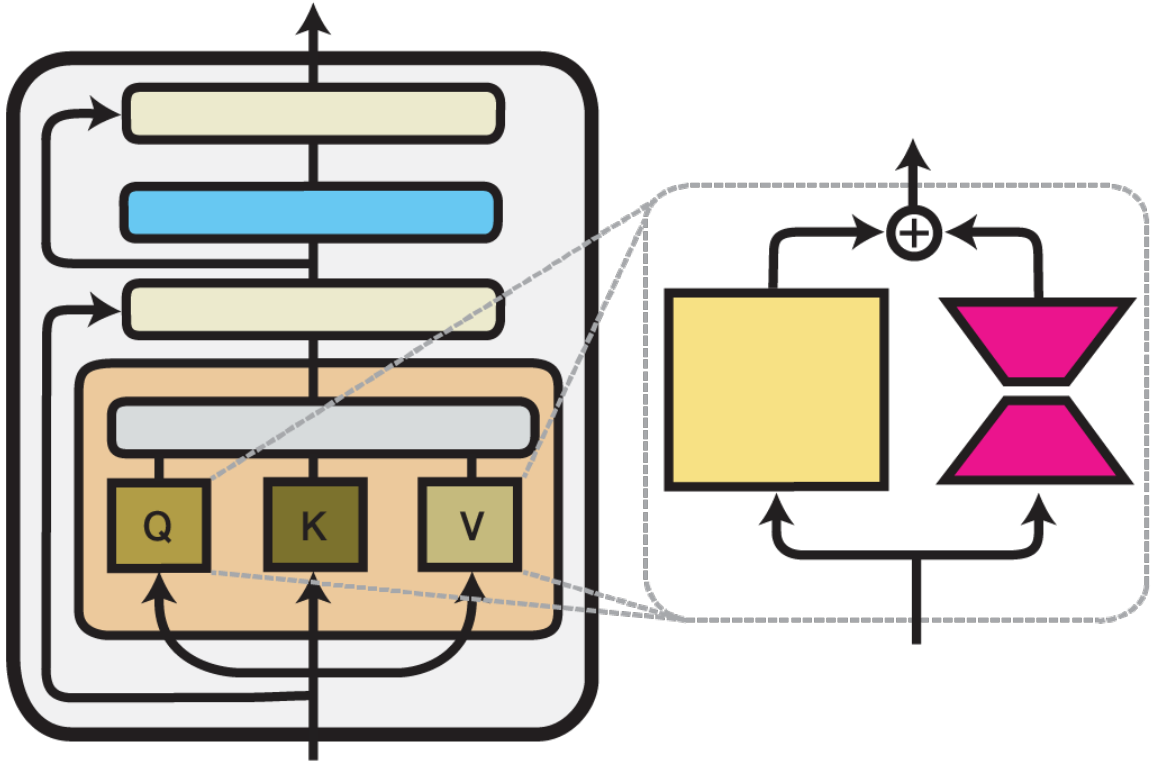
- Freezes the pretrained model weights and injects trainable rank decomposition matrices into each layer of the Transformer architecture



# 5 – Low-Rank Adaptation

## Low-Rank Adaptation (LoRA)

- Adapts the attention weights (query and value) of the Transformer self-attention sub-layer with LoRA



# 5 – Low-Rank Adaptation

**Result on the GLEU benchmark**

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

AI VIET NAM
@aivietnam.edu.vn

# 6 - Experiment

!

**Source code**

# Reference


- **COS 597G (Fall 2022): Understanding Large Language Models**
- https://adapterhub.ml/blog/2022/09/updates-in-adapter-transformers-v3-1/
- https://arxiv.org/pdf/2303.15647.pdf

AI VIET NAM

@aivietnam.edu.vn

# Thanks!

## Any questions?