

Parameter-efficient fine-tuning

CS 5624: Natural Language Processing

Spring 2025

<https://tuvllms.github.io/nlp-spring-2025>

Tu Vu

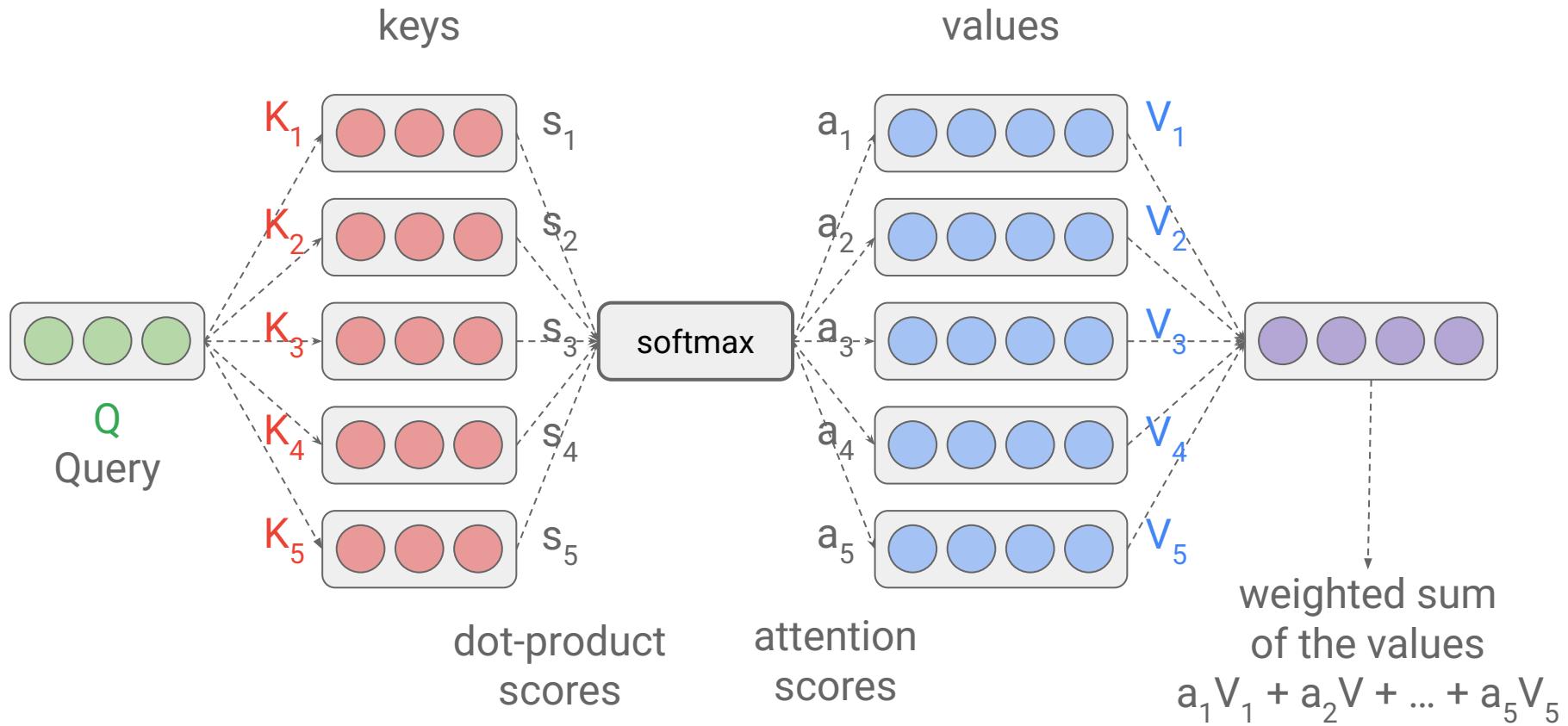


Logistics

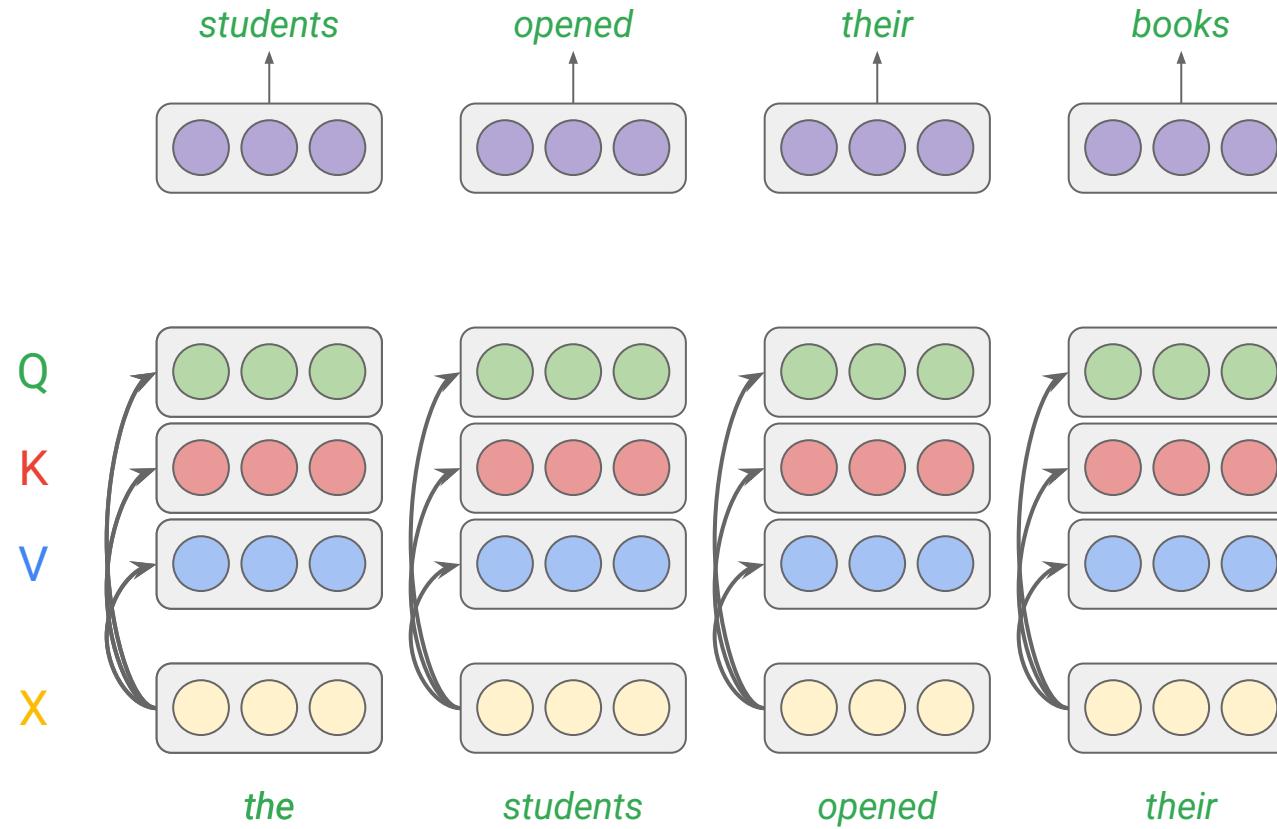
- Quiz 2 will be released tomorrow
- Homework 2 will be released sometime next week
- We are sending out feedback on final project proposals
- Please email us at cs5624instructors@gmail.com

Transformer recap

Attention mechanism



Self-attention



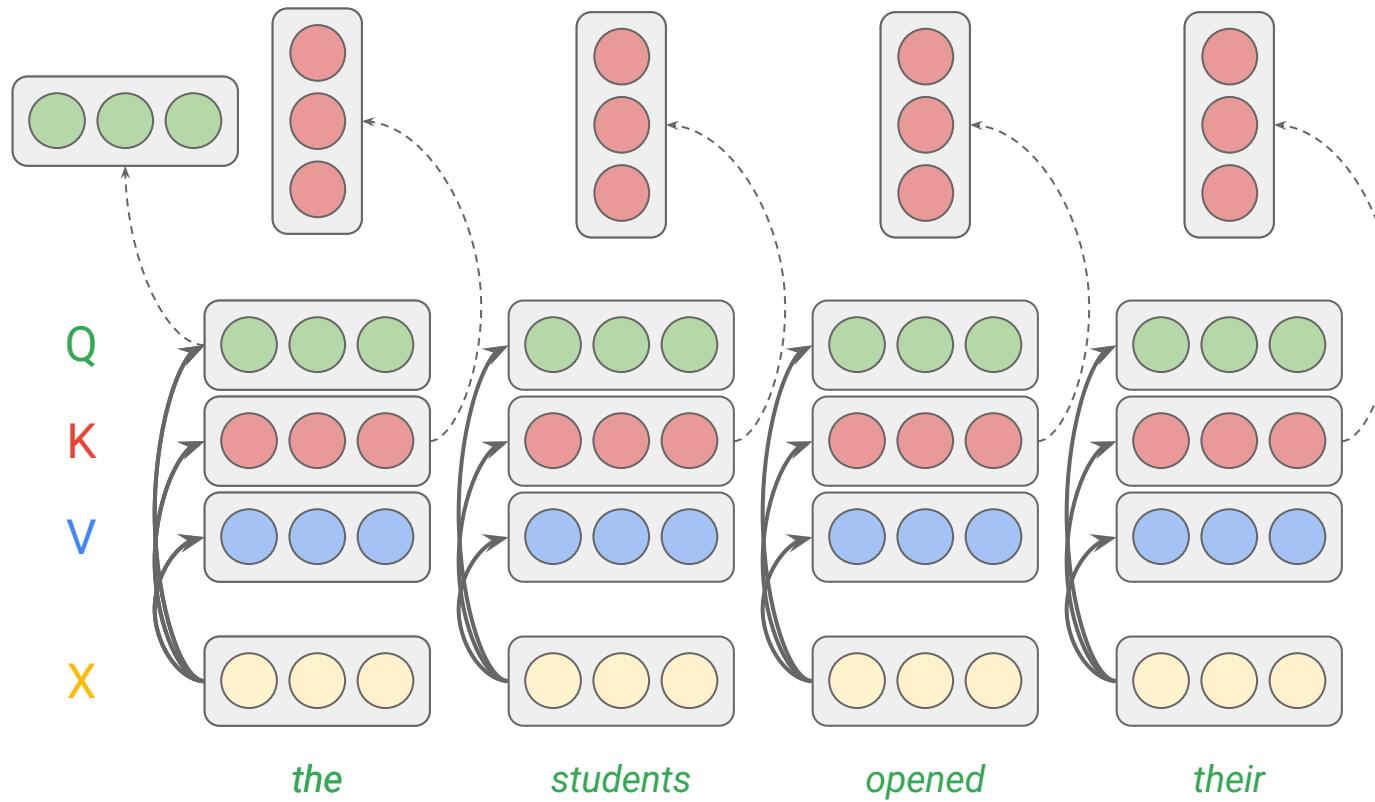
$$Q = X \cdot W_Q$$

$$K = X \cdot W_K$$

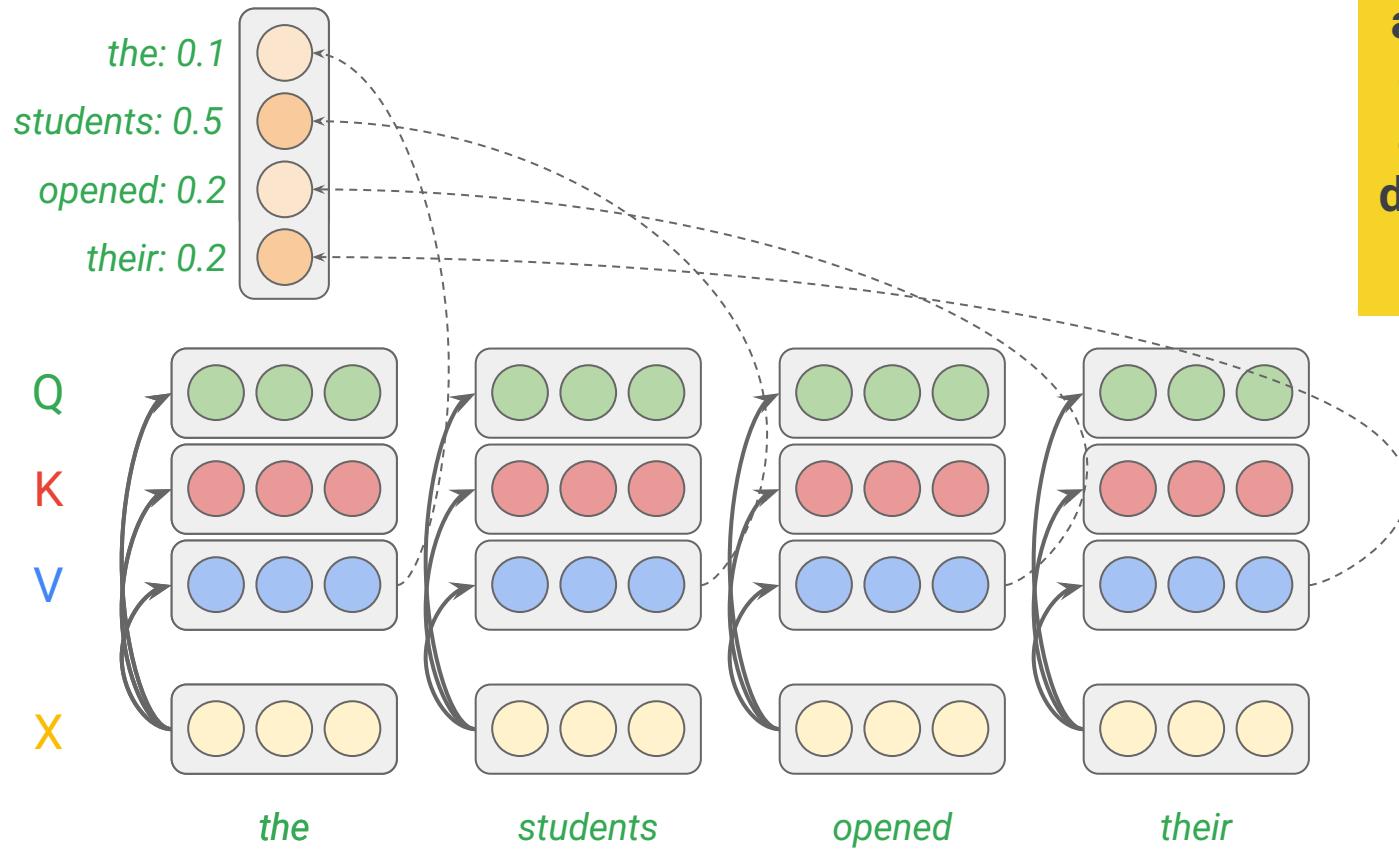
$$V = X \cdot W_V$$

linear
projections

Self-attention (cont'd)



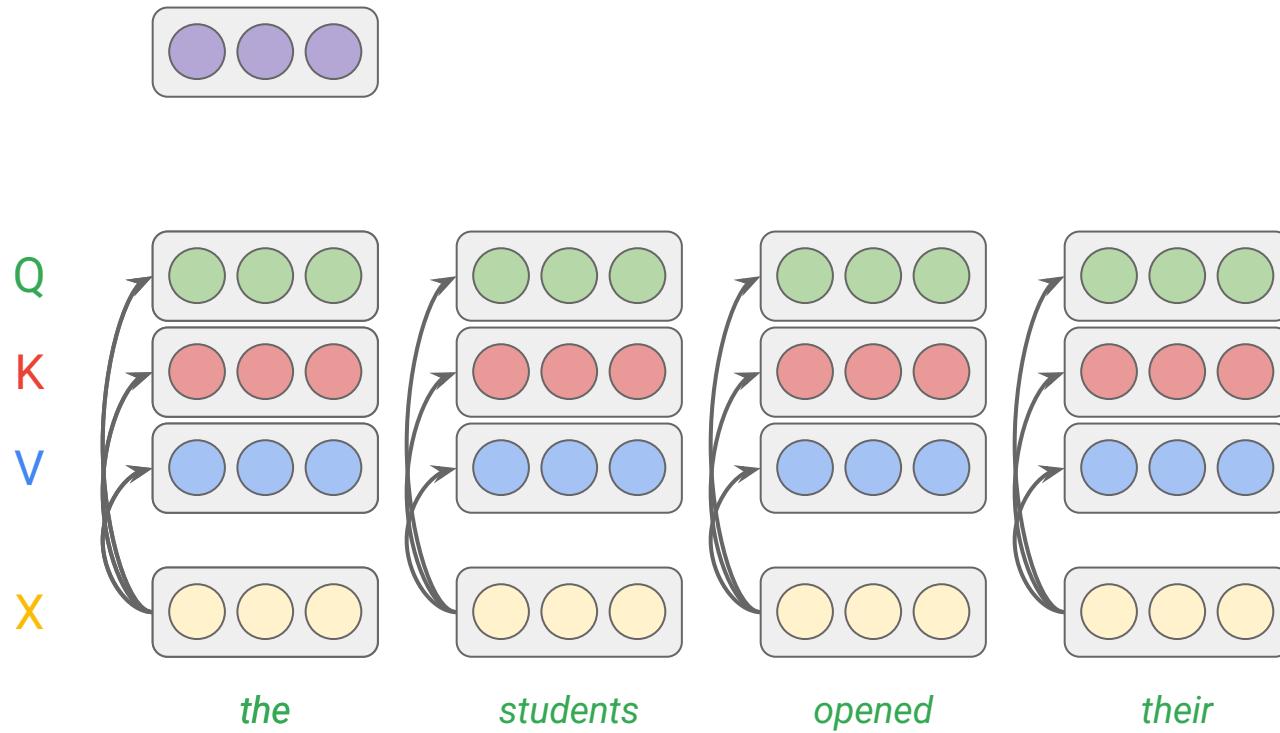
Self-attention (cont'd)



**all computations
are parallelized
during training
and sequential
during inference**

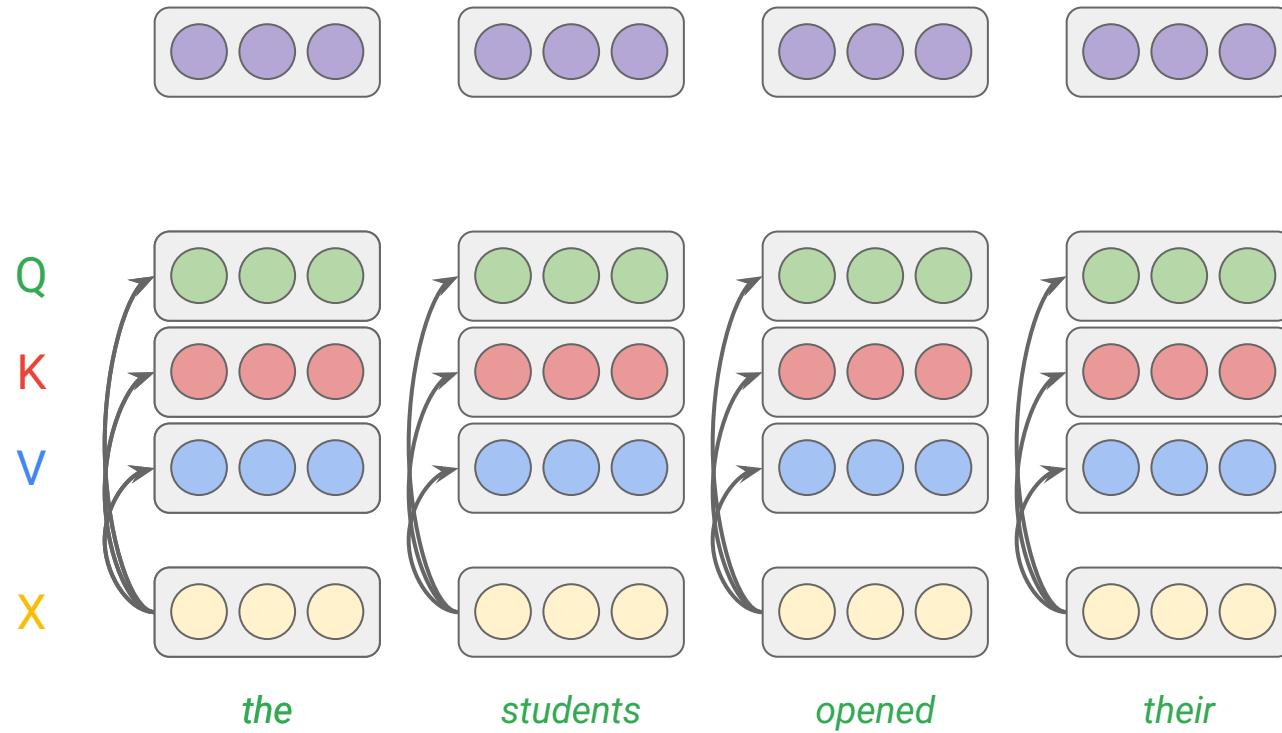
Self-attention (cont'd)

**all computations
are parallelized
during training
and sequential
during inference**



Self-attention (cont'd)

**all computations
are parallelized
during training
and sequential
during inference**



All computations are parallelized

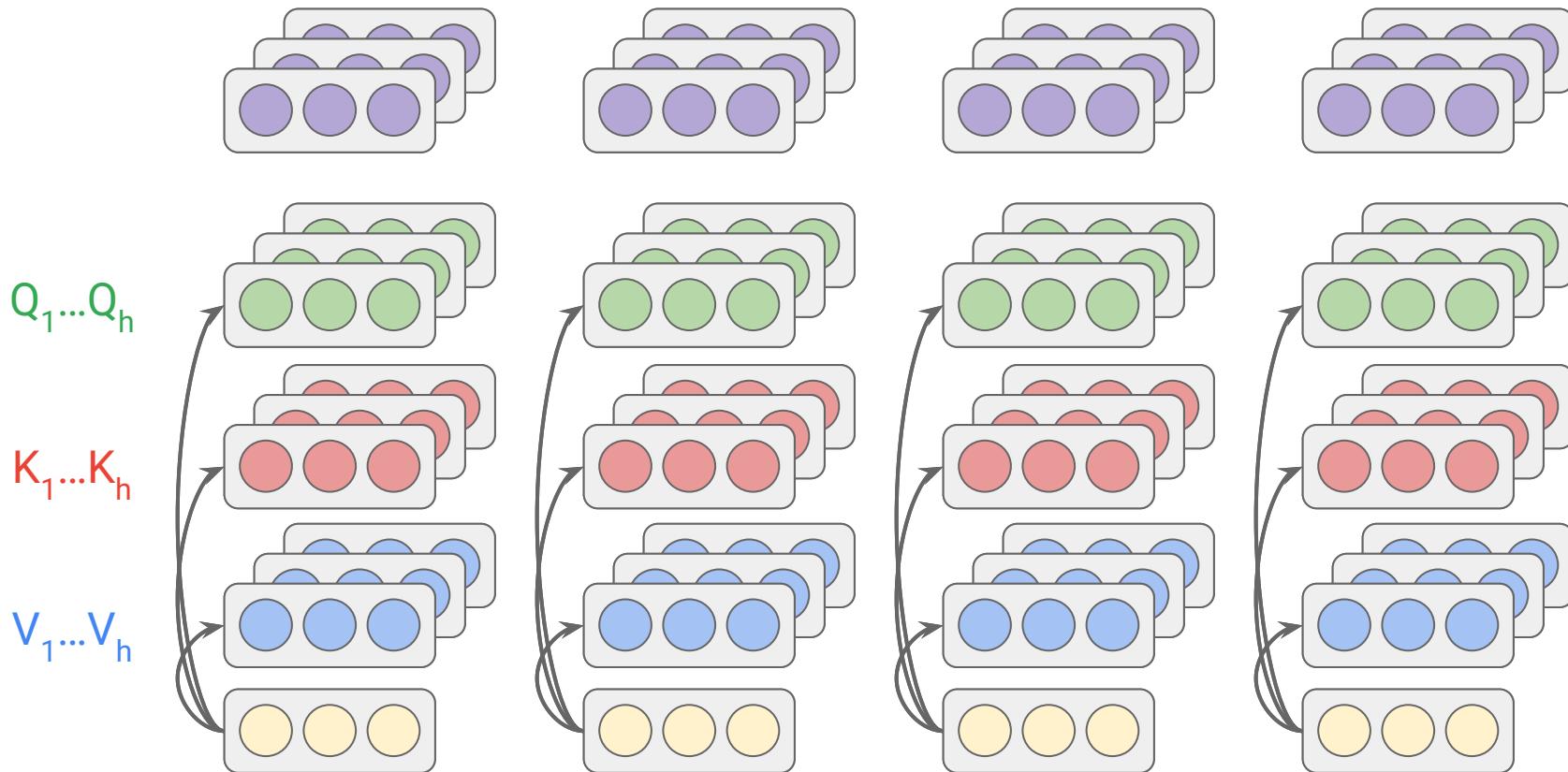
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



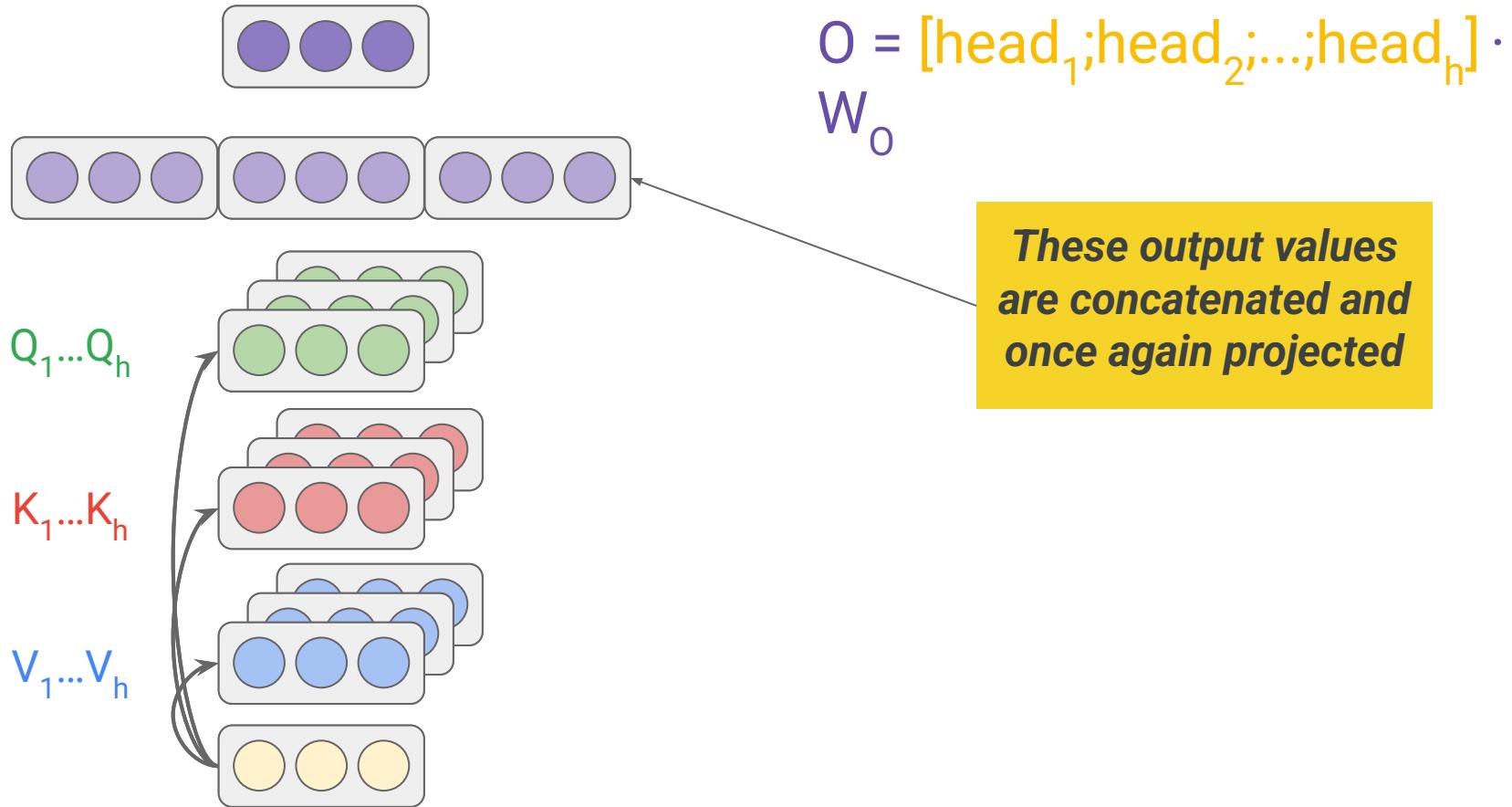
d_k : scaling factor

large products push the softmax function into regions where it has extremely small gradients

Multi-head attention



Multi-head attention (cont'd)



$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W_O$$

where

$$\text{head}_i = \text{Attention}(QW_{Q_i}, KW_{K_i}, VW_{V_i})$$

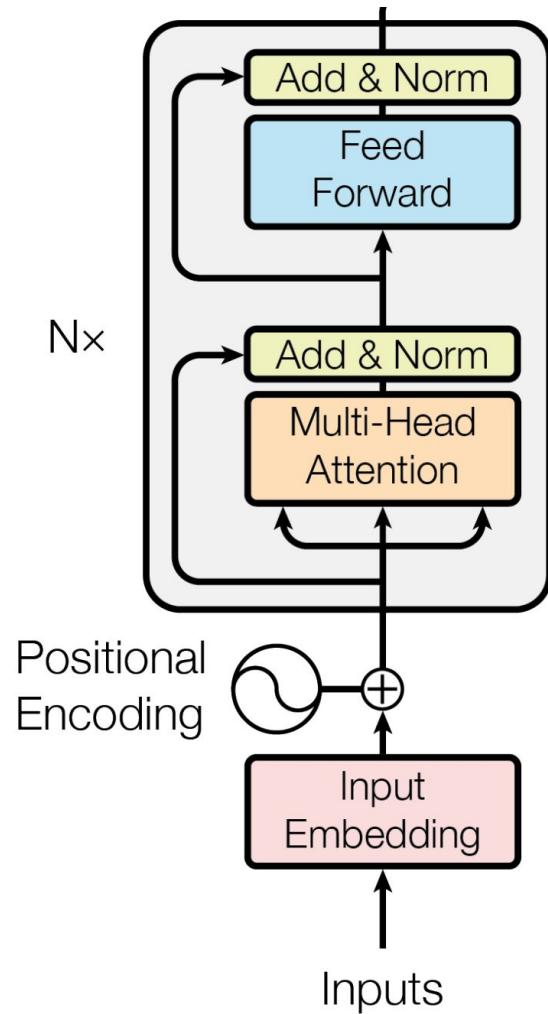
The projections are parameter matrices:

$$W_{Q_i} \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_{K_i} \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_{V_i} \in \mathbb{R}^{d_{\text{model}} \times d_v}$$

and

$$W_O \in \mathbb{R}^{hd_v \times d_{\text{model}}}.$$

In the Transformer paper, they employ $h = 8$ parallel attention layers, or heads. For each of these, they use $d_k = d_v = \frac{d_{\text{model}}}{h} = 64$. Due to the reduced dimension of each head, the total computational cost is similar to that of single-head attention with full dimensionality.



Bias term

$$h = \sigma(Wx + b)$$

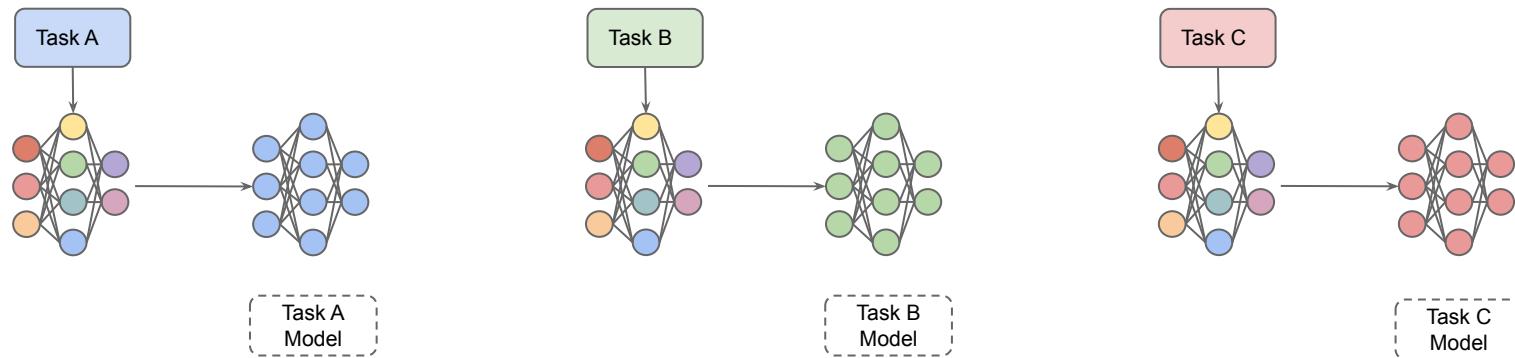
bias term

Model parameters (weights)

- Weight matrices
 - E.g., W_Q , W_K , W_V , W_O
- Bias terms

Limitations of full model tuning

Limitations of full model tuning



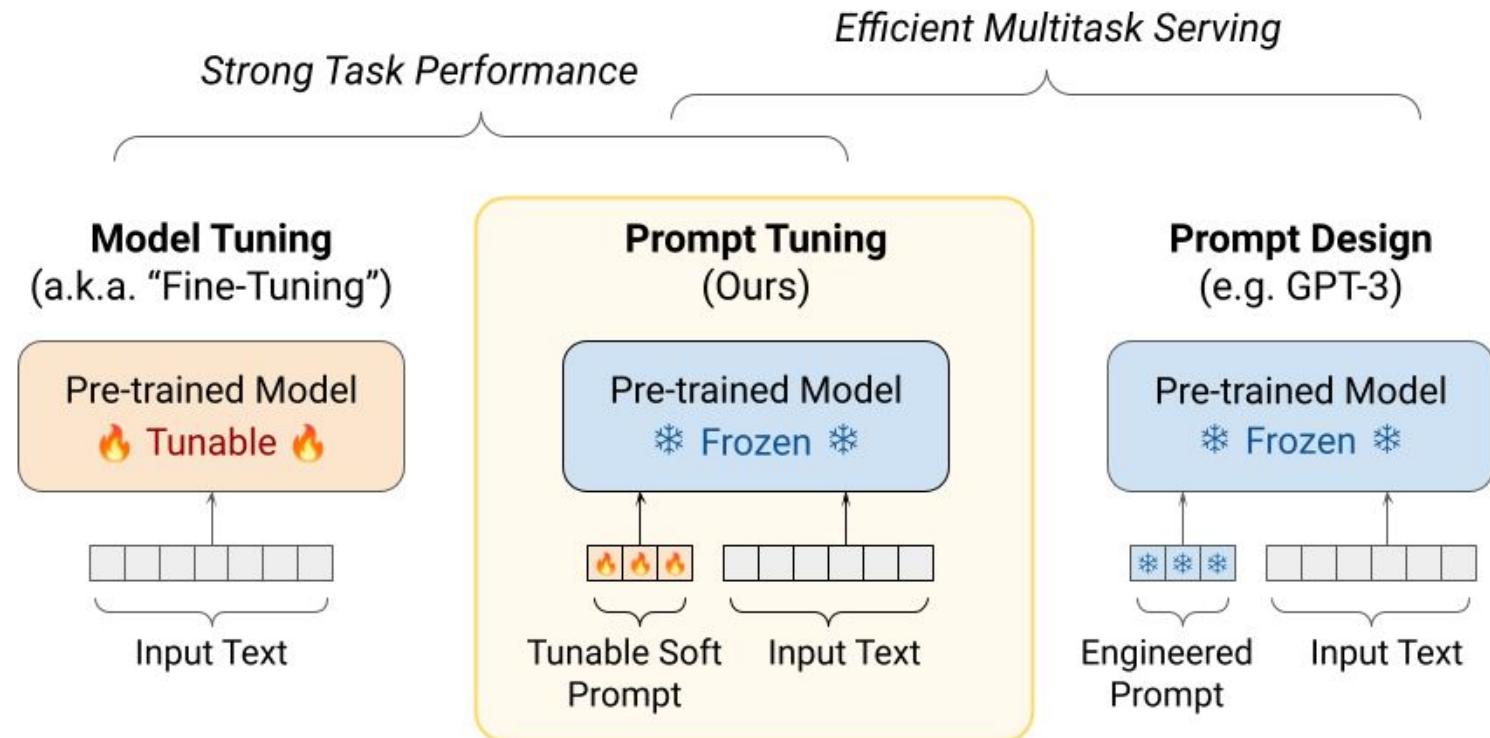
The Power of Scale for Parameter-Efficient Prompt Tuning

Brian Lester* **Rami Al-Rfou** **Noah Constant**

Google Research

{brianlester, rmyeid, nconstant}@google.com

Soft prompt tuning



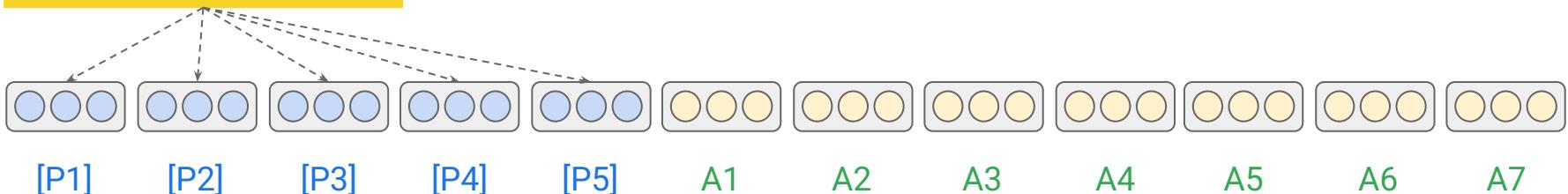
["The Power of Scale for Parameter-Efficient Prompt Tuning" by Lester et al. \(2021\)](#)

Soft prompt



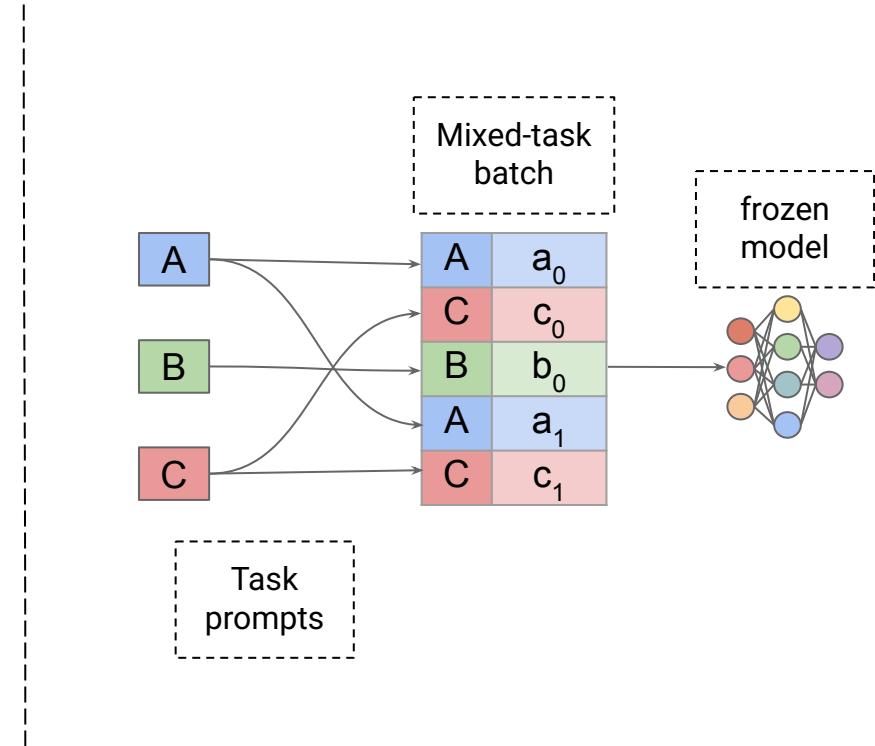
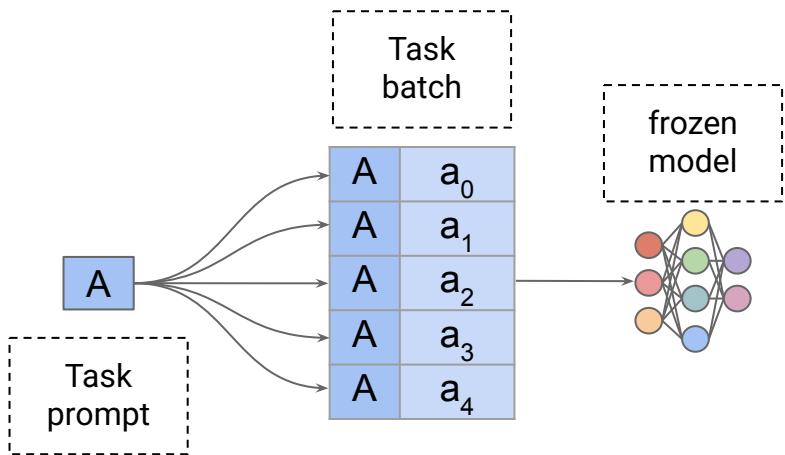
***soft tokens are
added in the
embedding layer***

**Multi-head Self-attention
(unmasked)**

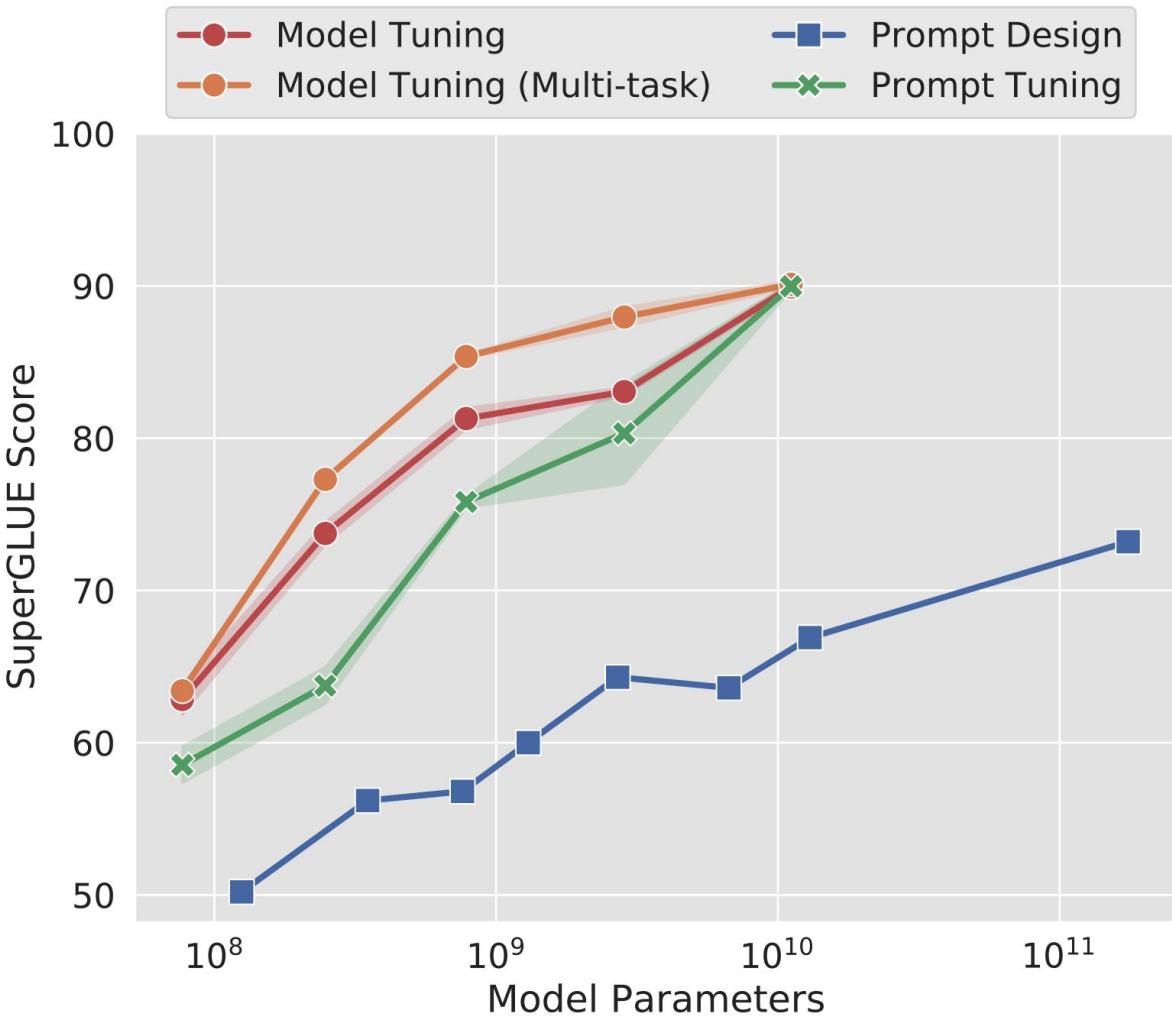


Advantages of soft prompt tuning

Parameter-efficient tuning & mixed-task inference



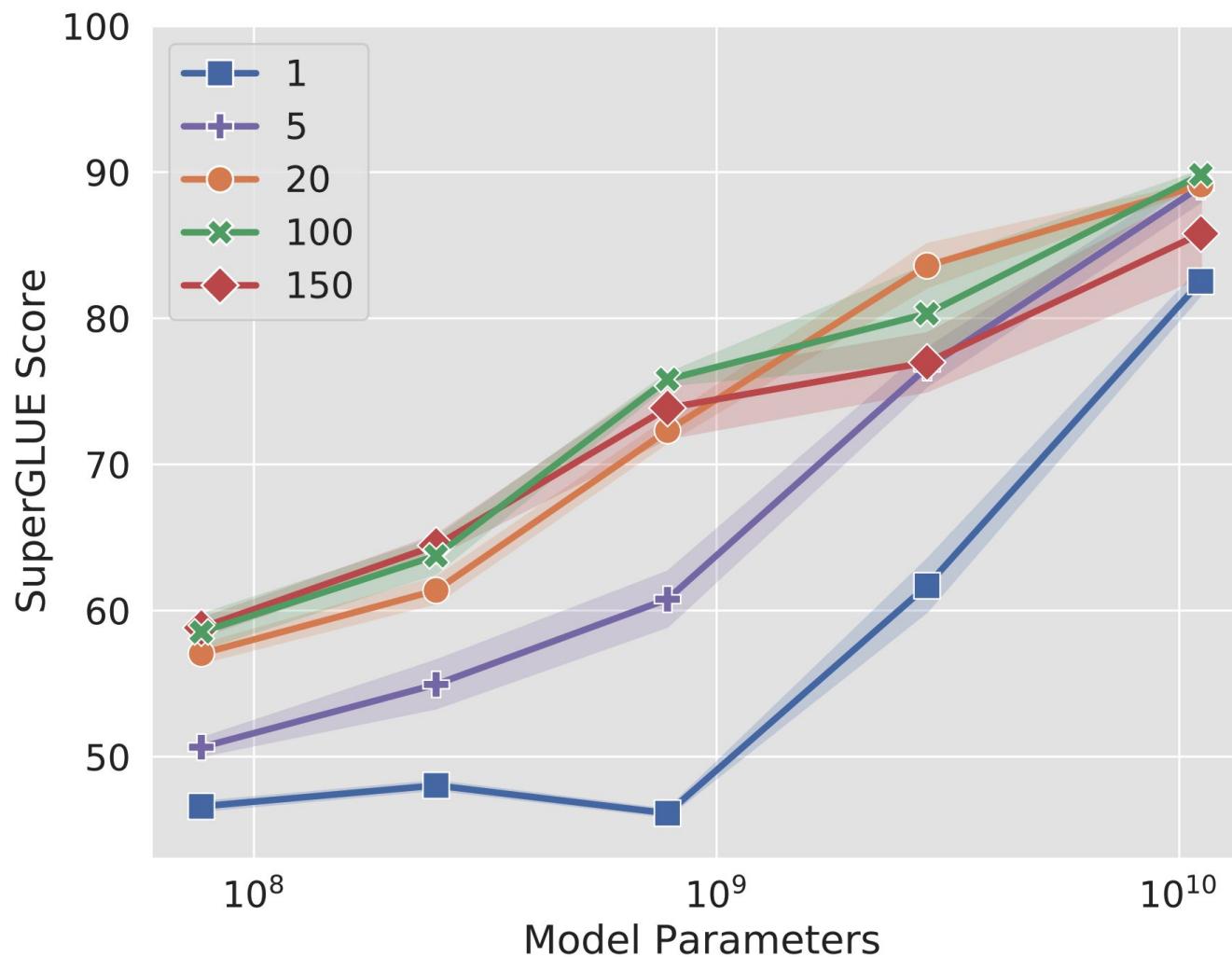
Improvement with Scale



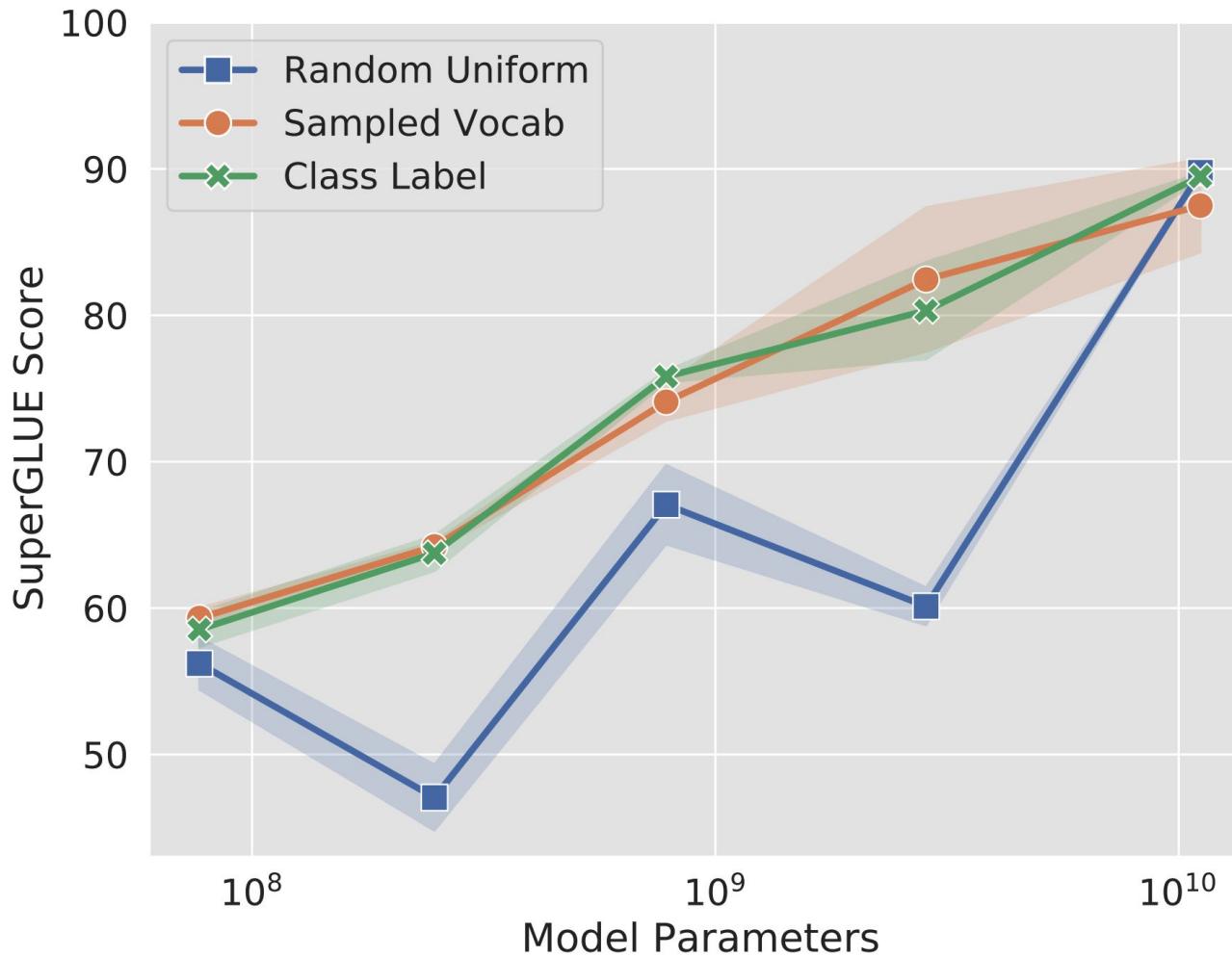
Resilience to domain shift

| Train | Eval | Tuning | Accuracy | F1 |
|-------|------|--------|----------------------------------|----------------------------------|
| QQP | MRPC | Model | 73.1 ± 0.9 | 81.2 ± 2.1 |
| | | Prompt | 76.3 ± 0.1 | 84.3 ± 0.3 |
| MRPC | QQP | Model | 74.9 ± 1.3 | 70.9 ± 1.2 |
| | | Prompt | 75.4 ± 0.8 | 69.7 ± 0.3 |

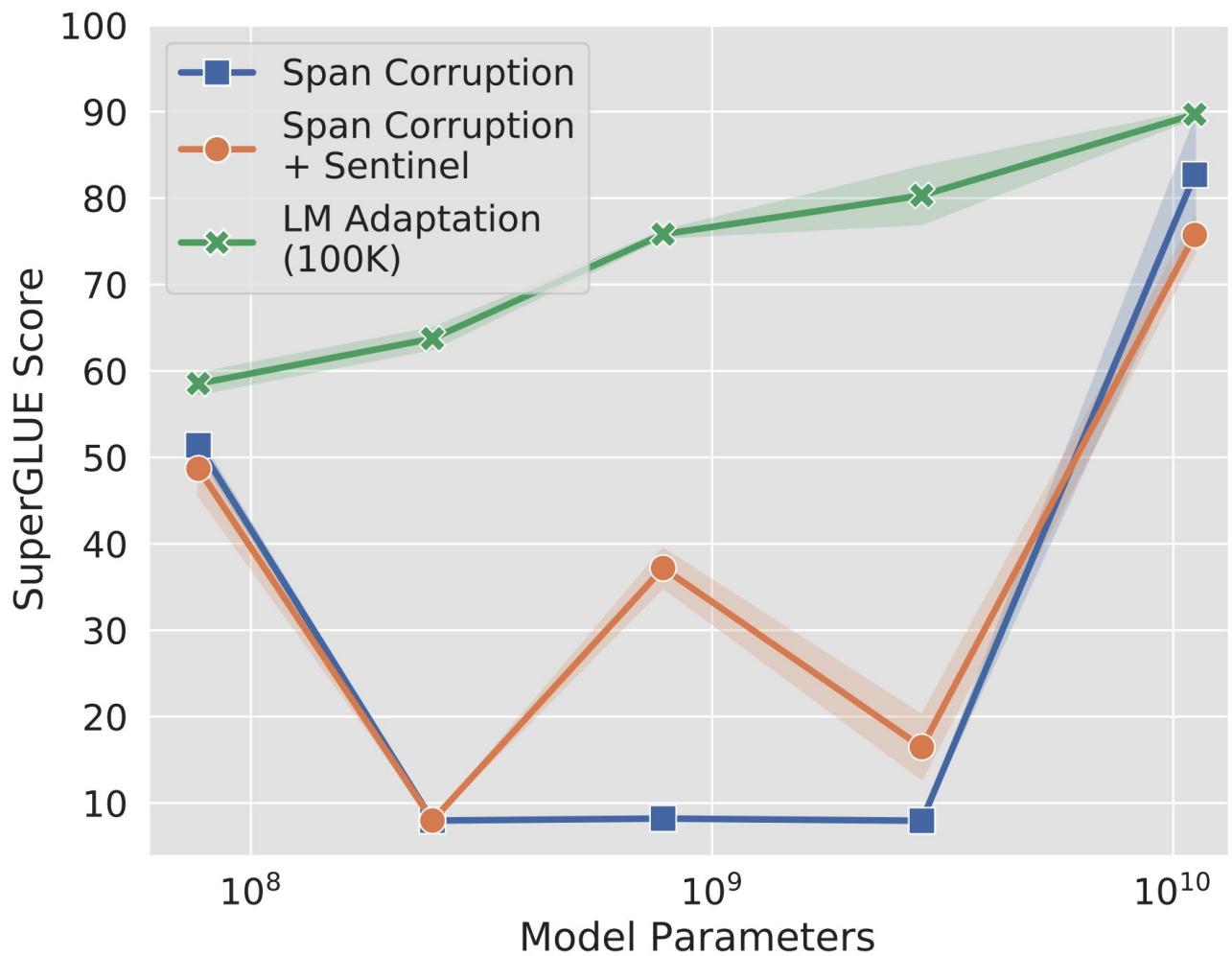
Effect of prompt length



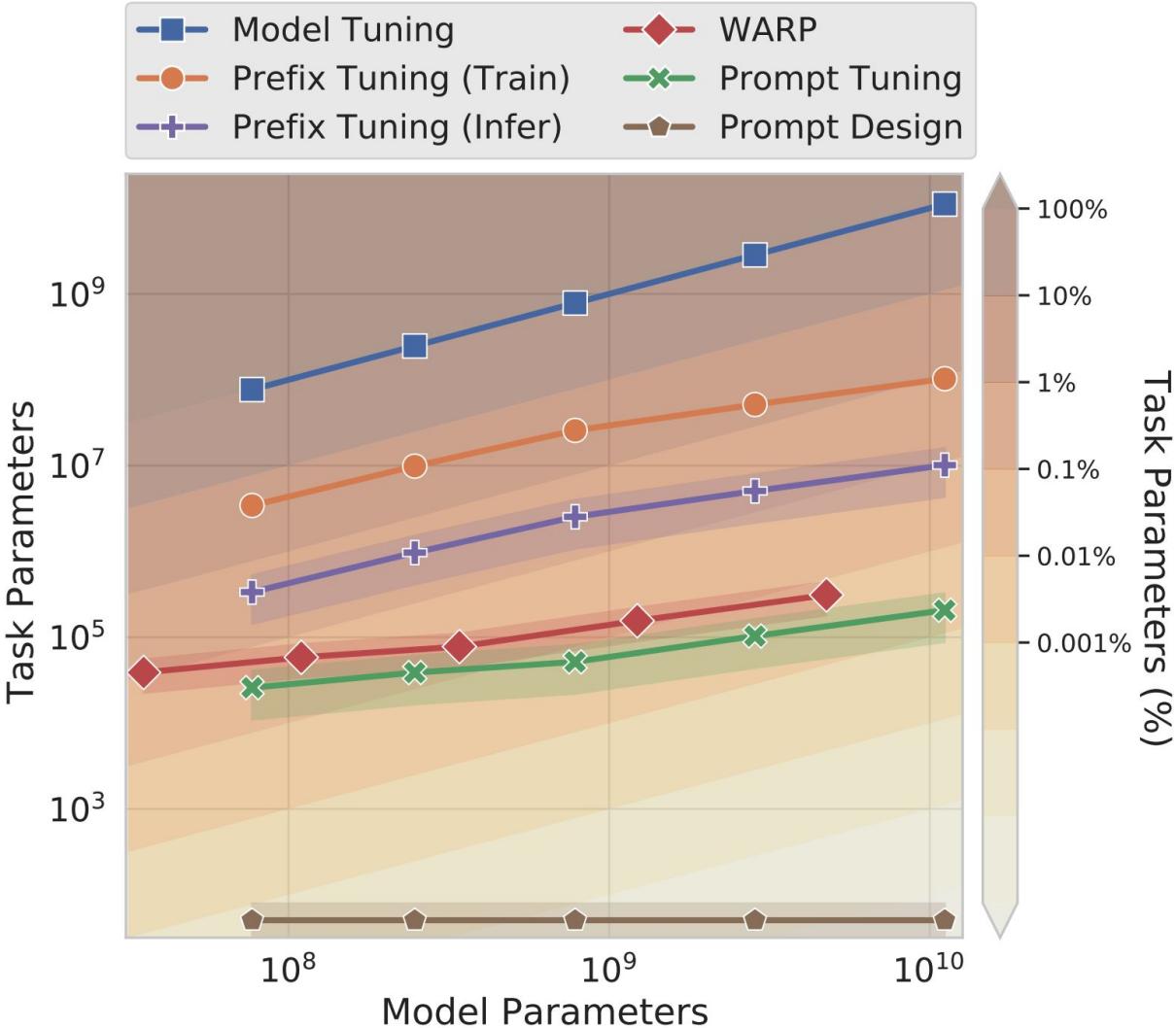
Effect of prompt initialization



Effect of pretraining method



Parameter usage



Interpretability

- the learned prompts taken as sequences show little interpretability

Limitations of soft prompt tuning

SPoT: Better Frozen Model Adaptation through Soft Prompt Transfer

Tu Vu^{1,2★}

Brian Lester¹

Noah Constant¹

Rami Al-Rfou¹

Daniel Cer¹

Google Research¹

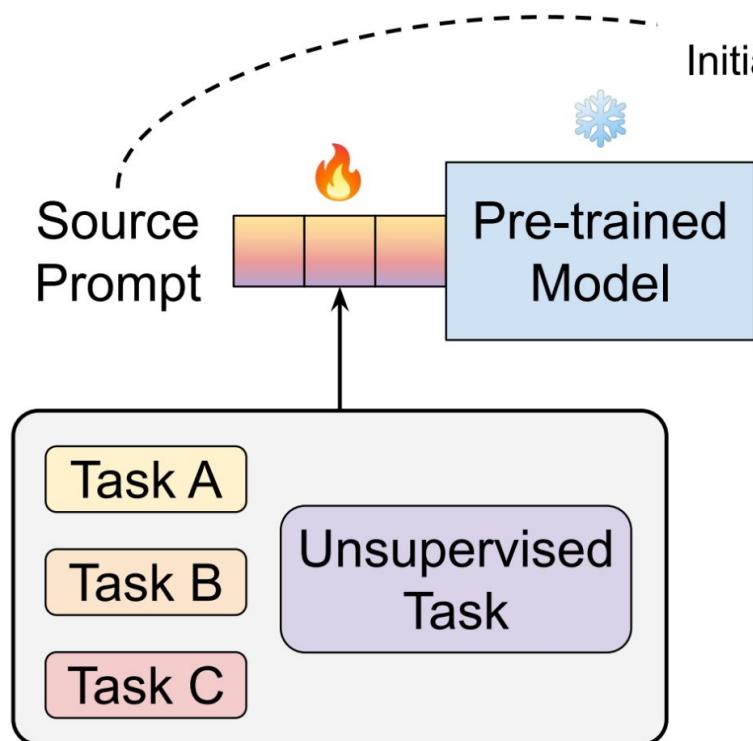
University of Massachusetts Amherst²

{ttvu,brianlester,nconstant,rmyeid,cer}@google.com

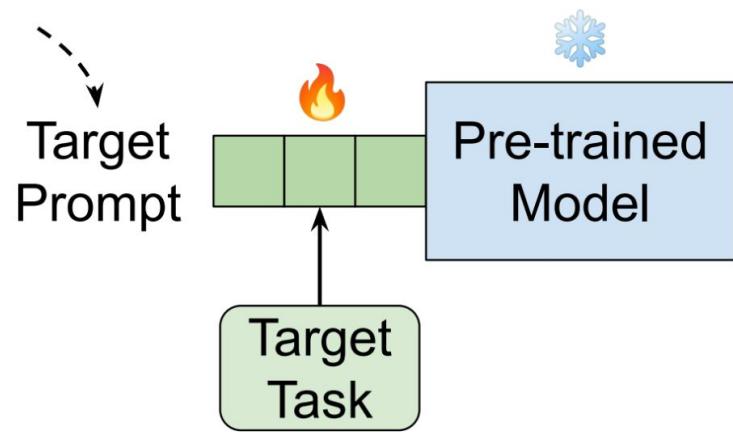
tuvu@cs.umass.edu

Generic SPoT

Source Prompt Tuning

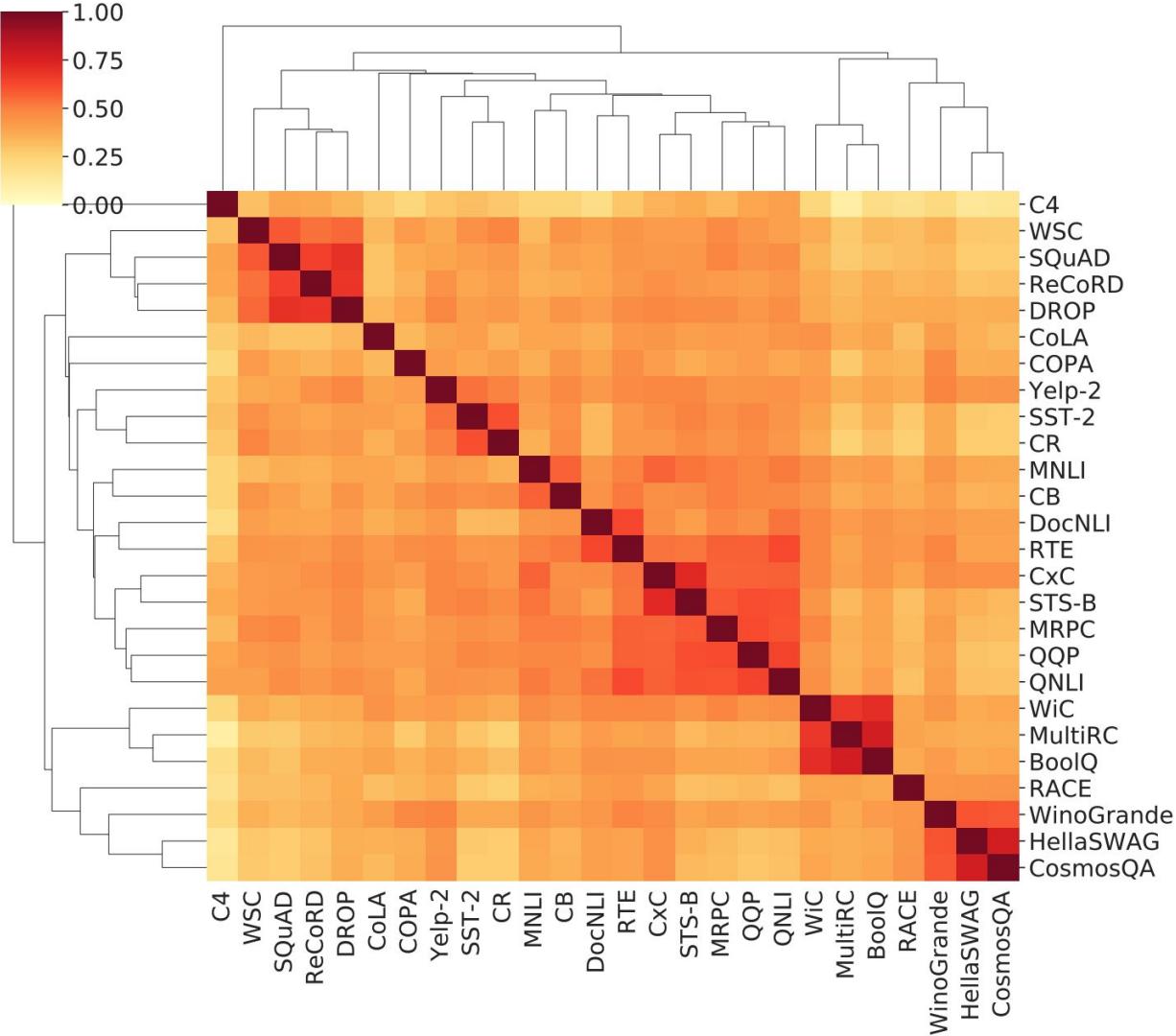


Target Prompt Tuning

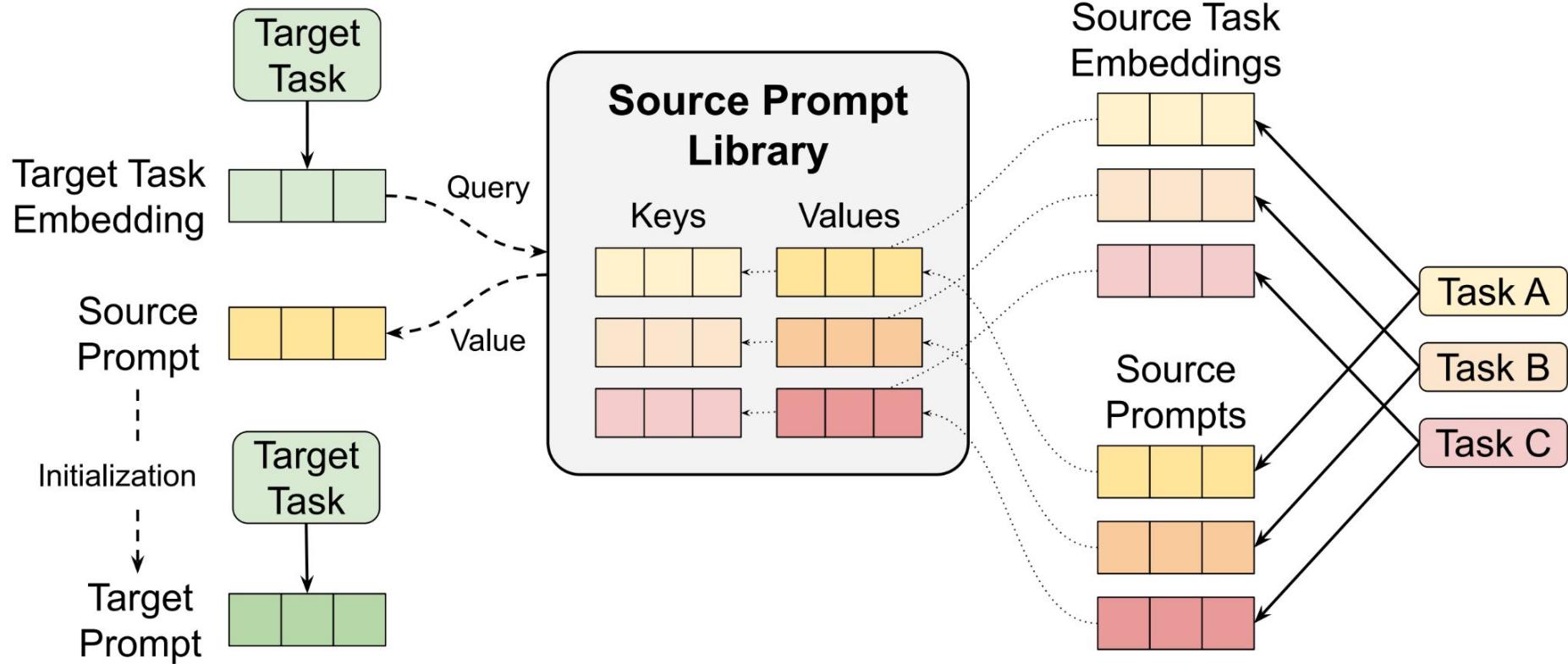


🔥 tuned
❄️ frozen

Prompt-based task embeddings capture task relationships



Targeted SPoT



Overcoming Catastrophic Forgetting in Zero-Shot Cross-Lingual Generation

Tu Vu^{1,2}★, Aditya Barua¹, Brian Lester¹, Daniel Cer¹, Mohit Iyyer², Noah Constant¹
Google Research¹

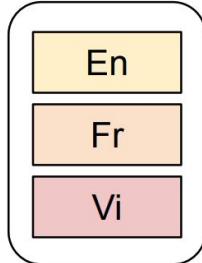
University of Massachusetts Amherst²

{ttvu, adityabaru, brianlester, cer, nconstant}@google.com
{tuvu, miyyer}@cs.umass.edu

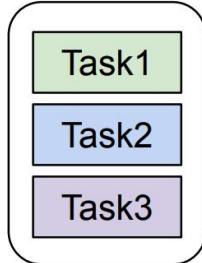
Factorized prompts

1) Train factorized prompts on all language / task combinations

Language Sub-Prompts



Task Sub-Prompts



Factorized Prompt Training Batch

| | | |
|----|-------|------------|
| Vi | Task2 | example #1 |
| Fr | Task1 | example #2 |
| En | Task1 | example #3 |
| Fr | Task3 | example #4 |

2) Train downstream task prompt (keeping En sub-prompt frozen)

| | | |
|----|------|------------|
| En | Summ | example #1 |
| En | Summ | example #2 |
| En | Summ | example #3 |

3) Swap language sub-prompts at inference time

| | | |
|----|------|------------|
| Fr | Summ | example #1 |
| Vi | Summ | example #2 |
| En | Summ | example #3 |

BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models

Elad Ben-Zaken¹ Shauli Ravfogel^{1,2} Yoav Goldberg^{1,2}

¹Computer Science Department, Bar Ilan University

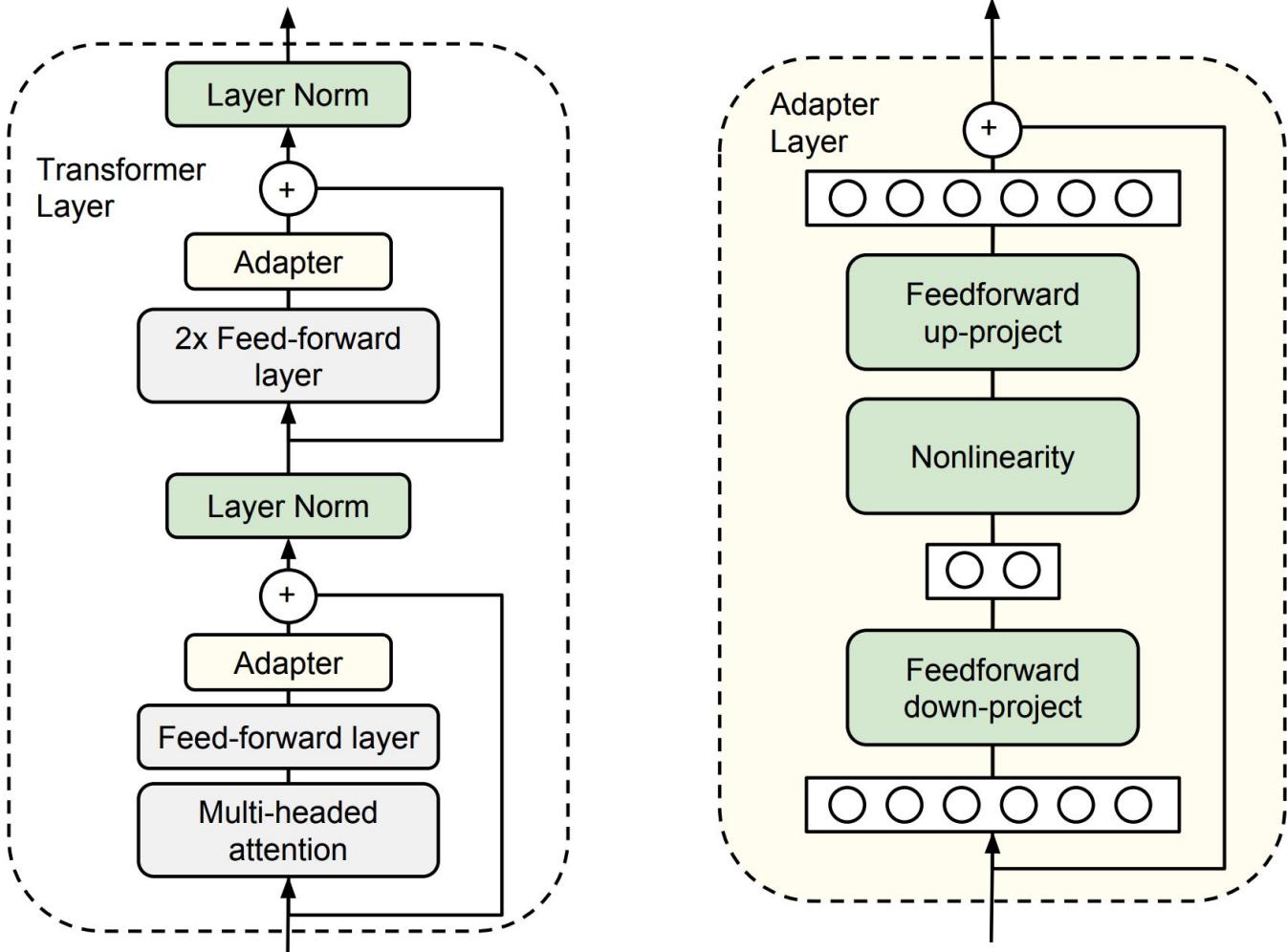
²Allen Institute for Artificial Intelligence

{benzakenelad, shauli.ravfogel, yoav.goldberg}@gmail.com

Parameter-Efficient Transfer Learning for NLP

Neil Houlsby¹ Andrei Giurgiu^{1*} Stanisław Jastrzębski^{2*} Bruna Morrone¹ Quentin de Laroussilhe¹
Andrea Gesmundo¹ Mona Attariyan¹ Sylvain Gelly¹

Adapters



Prefix-Tuning: Optimizing Continuous Prompts for Generation

Xiang Lisa Li

Stanford University

`xlisali@stanford.edu`

Percy Liang

Stanford University

`pliang@cs.stanford.edu`

LoRA

Algebra review

- The rank of a matrix is the number of linearly independent rows or columns (whichever is smaller)
- A ***full-rank*** matrix refers to a matrix that does not have any constraints on its rank. In other words, it has the maximum possible rank, meaning all of its rows and columns are linearly independent.

LORA: LOW-RANK ADAPTATION OF LARGE LANGUAGE MODELS

Edward Hu*

Yelong Shen*

Yuanzhi Li

Shean Wang

Microsoft Corporation

{edwardhu, yeshe, phwallis, zeyuana,

yuanzhil, swang, luw, wzchen}@microsoft.com

yuanzhil@andrew.cmu.edu

Phillip Wallis

Zeyuan Allen-Zhu

Lu Wang

Weizhu Chen

Weight changes during model adaptation has a low “intrinsic rank”

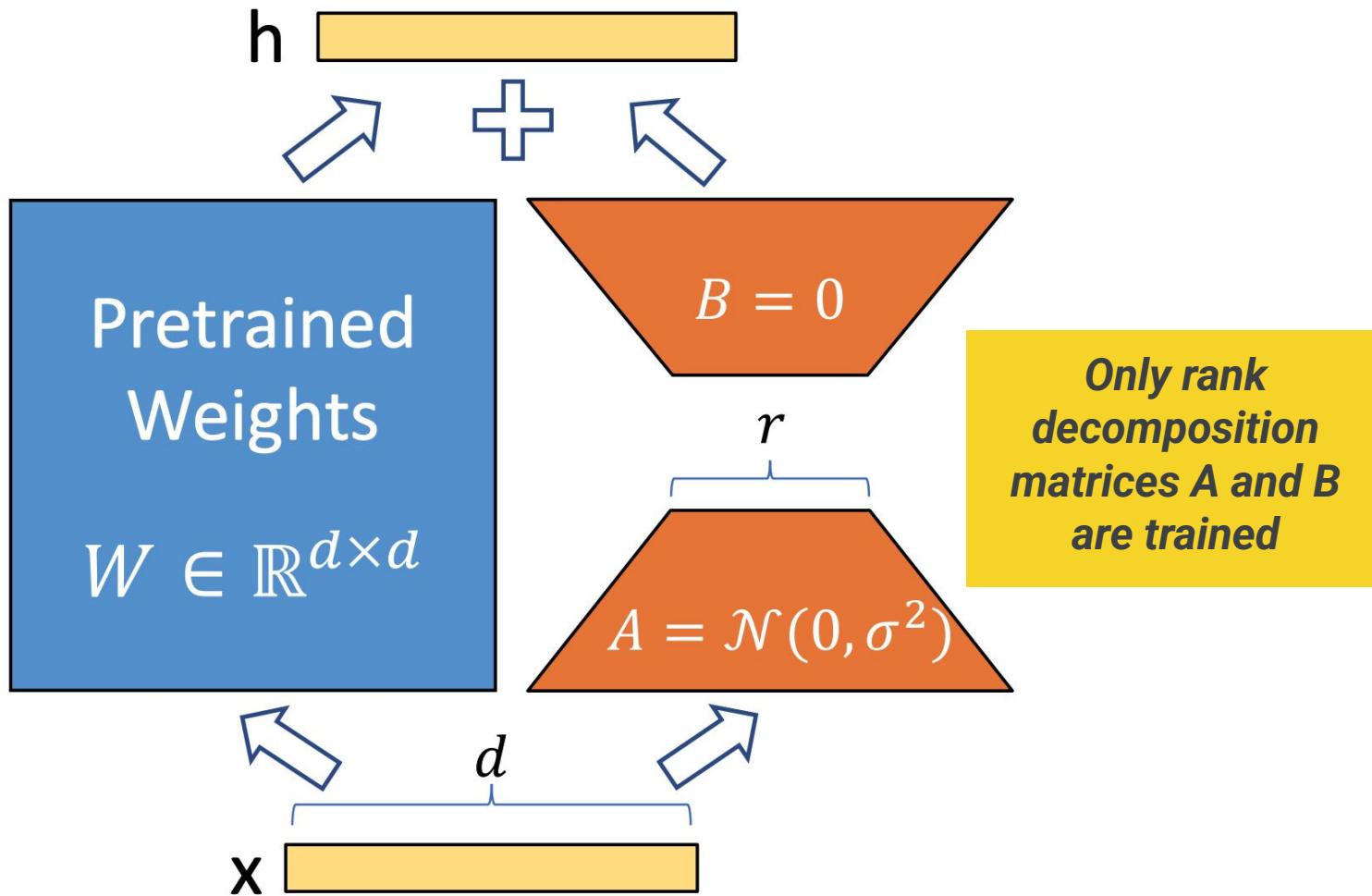
- The learned over-parametrized models in fact reside on a low intrinsic dimension
 - intrinsic dimension: the minimal number of variables needed to describe the essential variations in the data
- Many real-world high-dimensional datasets actually lie on or near a lower-dimensional manifold embedded in the high-dimensional space
- If a model or function resides in a low intrinsic dimension, then it may be possible to approximate it well with fewer parameters or a lower-dimensional representation, leading to improved generalization and efficiency

For a pre-trained weight matrix $W_0 \in \mathbb{R}^{d \times k}$, we constrain its update by representing the latter with a low-rank decomposition $W_0 + \Delta W = W_0 + BA$, where $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times k}$, and the rank $r \ll \min(d, k)$. During training, W_0 is frozen and does not receive gradient updates, while A and B contain trainable parameters. Note both W_0 and $\Delta W = BA$ are multiplied with the same input, and their respective output vectors are summed coordinate-wise.

For $h = W_0x$, our modified forward pass yields:

$$h = W_0x + \Delta Wx = W_0x + BAx$$

LoRA

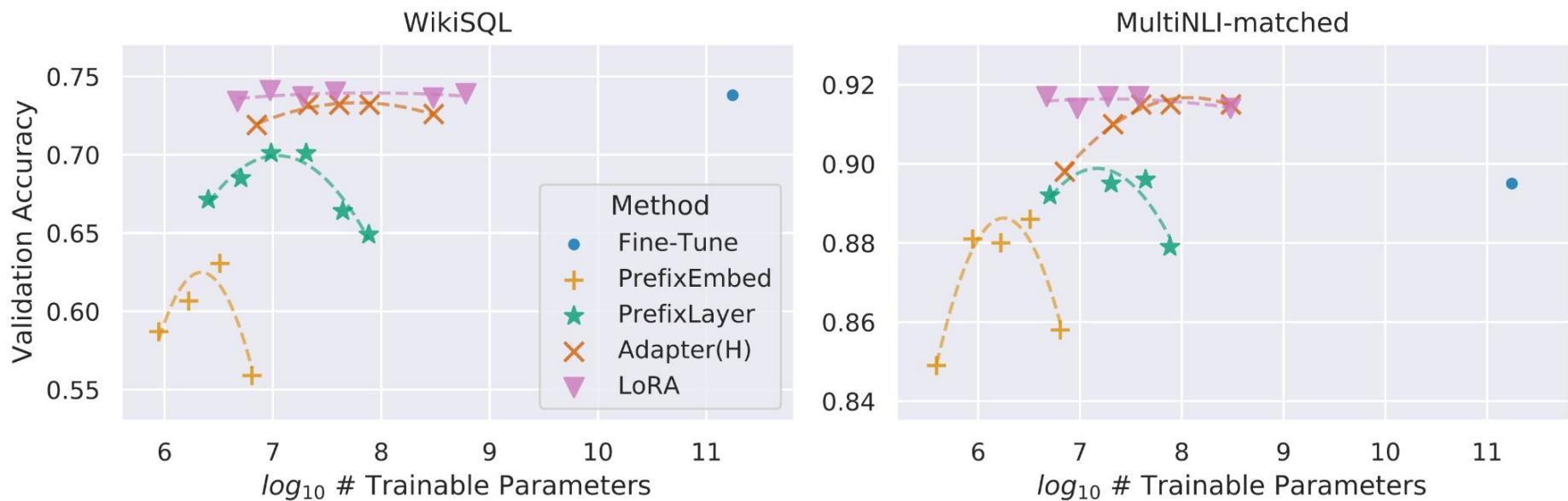


Advantages of LoRA

| Model & Method | # Trainable Parameters | MNLI | SST-2 | MRPC | CoLA | QNLI | QQP | RTE | STS-B | Avg. |
|--|------------------------|----------------------------|----------------------------|-----------------------------|-----------------------------|----------------------------|----------------------------|-----------------------------|----------------------------|-------------|
| RoB _{base} (FT)* | 125.0M | 87.6 | 94.8 | 90.2 | 63.6 | 92.8 | 91.9 | 78.7 | 91.2 | 86.4 |
| RoB _{base} (BitFit)* | 0.1M | 84.7 | 93.7 | 92.7 | 62.0 | 91.8 | 84.0 | 81.5 | 90.8 | 85.2 |
| RoB _{base} (Adpt ^D)* | 0.3M | 87.1 _{±.0} | 94.2 _{±.1} | 88.5 _{±1.1} | 60.8 _{±.4} | 93.1 _{±.1} | 90.2 _{±.0} | 71.5 _{±2.7} | 89.7 _{±.3} | 84.4 |
| RoB _{base} (Adpt ^D)* | 0.9M | 87.3 _{±.1} | 94.7 _{±.3} | 88.4 _{±.1} | 62.6 _{±.9} | 93.0 _{±.2} | 90.6 _{±.0} | 75.9 _{±2.2} | 90.3 _{±.1} | 85.4 |
| RoB _{base} (LoRA) | 0.3M | 87.5 _{±.3} | 95.1 _{±.2} | 89.7 _{±.7} | 63.4 _{±1.2} | 93.3 _{±.3} | 90.8 _{±.1} | 86.6 _{±.7} | 91.5 _{±.2} | 87.2 |
| RoB _{large} (FT)* | 355.0M | 90.2 | 96.4 | 90.9 | 68.0 | 94.7 | 92.2 | 86.6 | 92.4 | 88.9 |
| RoB _{large} (LoRA) | 0.8M | 90.6 _{±.2} | 96.2 _{±.5} | 90.9 _{±1.2} | 68.2 _{±1.9} | 94.9 _{±.3} | 91.6 _{±.1} | 87.4 _{±2.5} | 92.6 _{±.2} | 89.0 |
| RoB _{large} (Adpt ^P)† | 3.0M | 90.2 _{±.3} | 96.1 _{±.3} | 90.2 _{±.7} | 68.3 _{±1.0} | 94.8 _{±.2} | 91.9 _{±.1} | 83.8 _{±2.9} | 92.1 _{±.7} | 88.4 |
| RoB _{large} (Adpt ^P)† | 0.8M | 90.5 _{±.3} | 96.6 _{±.2} | 89.7 _{±1.2} | 67.8 _{±2.5} | 94.8 _{±.3} | 91.7 _{±.2} | 80.1 _{±2.9} | 91.9 _{±.4} | 87.9 |
| RoB _{large} (Adpt ^H)† | 6.0M | 89.9 _{±.5} | 96.2 _{±.3} | 88.7 _{±2.9} | 66.5 _{±4.4} | 94.7 _{±.2} | 92.1 _{±.1} | 83.4 _{±1.1} | 91.0 _{±1.7} | 87.8 |
| RoB _{large} (Adpt ^H)† | 0.8M | 90.3 _{±.3} | 96.3 _{±.5} | 87.7 _{±1.7} | 66.3 _{±2.0} | 94.7 _{±.2} | 91.5 _{±.1} | 72.9 _{±2.9} | 91.5 _{±.5} | 86.4 |
| RoB _{large} (LoRA)† | 0.8M | 90.6 _{±.2} | 96.2 _{±.5} | 90.2 _{±1.0} | 68.2 _{±1.9} | 94.8 _{±.3} | 91.6 _{±.2} | 85.2 _{±1.1} | 92.3 _{±.5} | 88.6 |
| DeB _{XXL} (FT)* | 1500.0M | 91.8 | 97.2 | 92.0 | 72.0 | 96.0 | 92.7 | 93.9 | 92.9 | 91.1 |
| DeB _{XXL} (LoRA) | 4.7M | 91.9 _{±.2} | 96.9 _{±.2} | 92.6 _{±.6} | 72.4 _{±1.1} | 96.0 _{±.1} | 92.9 _{±.1} | 94.9 _{±.4} | 93.0 _{±.2} | 91.3 |

| Model&Method | # Trainable Parameters | WikiSQL | MNLI-m | SAMSum |
|-------------------------------|------------------------|-------------|-------------|-----------------------|
| | | Acc. (%) | Acc. (%) | 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 (Adapter ^H) | 7.1M | 71.9 | 89.8 | 53.0/28.9/44.8 |
| GPT-3 (Adapter ^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 exhibits better scalability and task performance



Given a limited parameter budget, which weight matrices should we apply LoRA to?

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

The effect of rank r on model performance

| | 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 |

practical recommendations

of training examples

- < 20: LoRA is difficult to train
- 50: LoRA w/ careful settings can be better than full model fine-tuning; $r=1$ or 4
- $O(100)$: e.g., 200-500, LoRA is recommended; $r=1$ or 4
- $O(10K)$: should compare LoRA vs. full model fine-tuning
- Very large ($>100K$): LoRA can get decent quality to match full model fine-tuning when r is large, e.g., 128 or 512

Limitations of parameter-efficient tuning methods

LoRA Learns Less and Forgets Less

Dan Biderman^{1,2}, Jacob Portes², Jose Javier Gonzalez Ortiz², Mansheej Paul², Philip Greengard¹, Connor Jennings², Daniel King², Sam Havens², Vitaliy Chiley², Jonathan Frankle², Cody Blakeney², John P. Cunningham¹

¹Columbia University {db3236, pg2118, jpc2181}@columbia.edu

²Databricks Mosaic Research {jacob.portes, j.gonzalez, mansheej.paul, connor.jennings, daniel.king, sam.havens, vitaliy.chiley, jfrankle, cody.blakeney}@databricks.com



LoraHub: Efficient Cross-Task Generalization via Dynamic LoRA Composition

Chengsong Huang^{†§*}, Qian Liu^{†*}, Bill Yuchen Lin^{◊*}, Tianyu Pang[†], Chao Du[†], Min Lin[†]

[†]Sea AI Lab, Singapore

[§]Washington University in St. Louis, MO, USA

[◊]Allen Institute for AI, Seattle, WA, USA

Thank you!