

Efficient training

CS 6804: Frontier AI Systems
Spring 2026

<https://tuvllms.github.io/ai-seminar-spring-2026/>

Tu Vu



Logistics

- Homework assignments
 - Quiz 0, due **tomorrow 2/17**
 - graded for genuine attempt, not correctness
 - Homework 0 & 1, due **3/3 & 3/10**
 - 5% extra credits each
- Student presentation groups confirmed

AI news

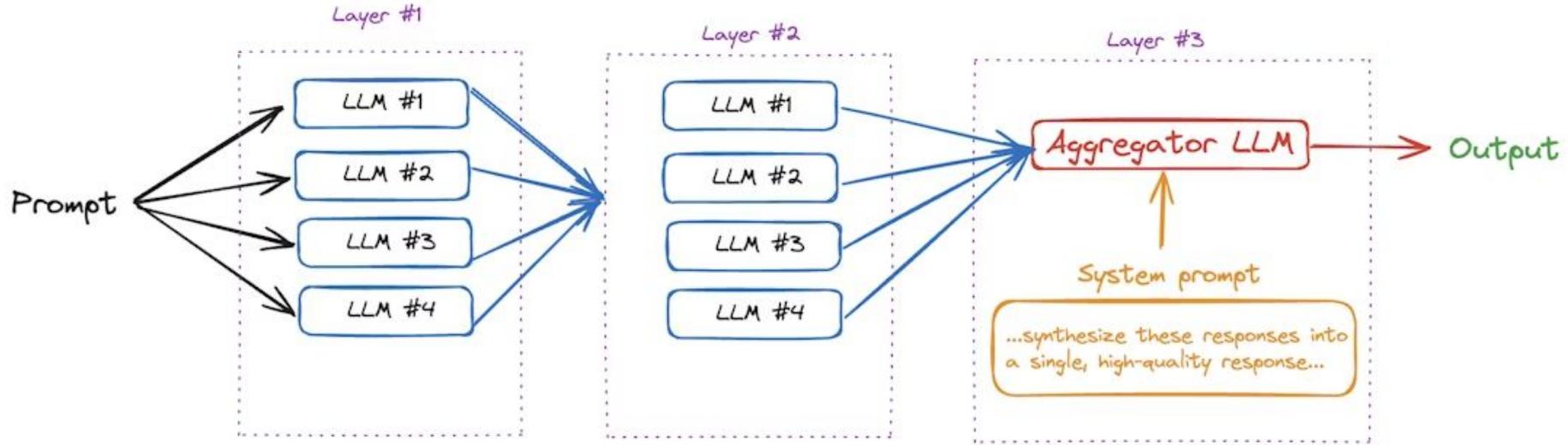


✦ Gemini 3 Deep Think



Deep Think / Parallel Agents / Mixture-of-Agents

Mixture-of-Agents (MoA)
3 layer example

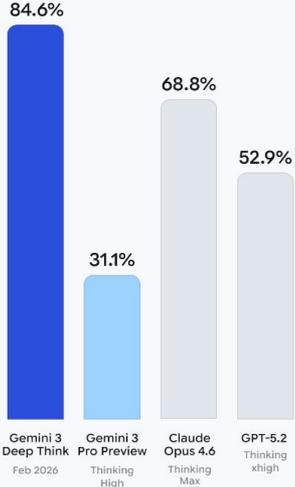


<https://docs.together.ai/docs/mixture-of-agents#advanced-moa-example>

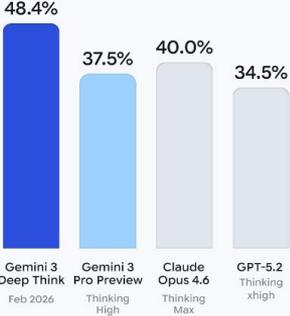
AI news

Gemini 3 Deep Think

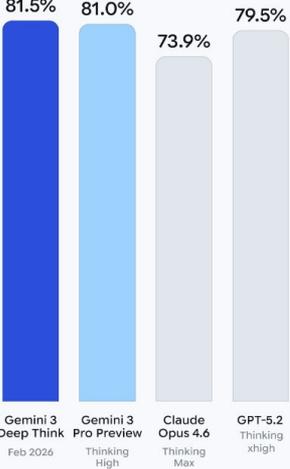
ARC-AGI-2
Reasoning & knowledge
ARC PRIZE VERIFIED



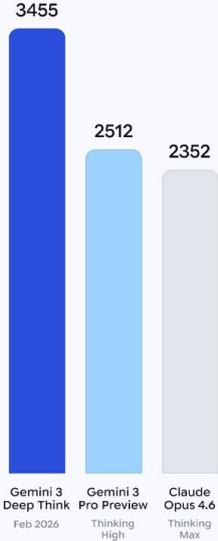
Humanity's Last Exam
Academic reasoning
NO TOOLS



MMMU-Pro
Multimodal understanding and reasoning



Codeforces
Coding and algorithms
NO TOOLS



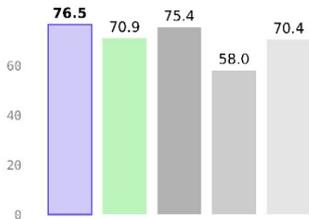
Methodology: deepmind.google/models/evals-methodology/gemini-3-deep-think

Benchmark		Gemini 3 Deep Think Feb 2026	Gemini 3 Pro Preview Thinking High	Claude Opus 4.6 Thinking Max	GPT-5.2 Thinking xhigh
ARC-AGI-2 Abstract reasoning puzzles	ARC prize verified	84.6%	31.1%	68.8%	52.9%
Humanity's Last Exam Academic reasoning (full set, text + MM)	No tools Search + code execution	48.4% 53.4%	37.5% 45.8%	40.0% 53.1%	34.5% 45.5%
MMMU-Pro Multimodal understanding and reasoning	No tools	81.5%	81.0%	73.9%	79.5%
International Math Olympiad 2025 Mathematics		81.5%	14.3%	-	71.4%
Codeforces Coding and algorithms	No tools, Elo	3455	2512	2352	-
International Physics Olympiad 2025 (theory) Physics		87.7%	76.3%	71.6%	70.5%
CMT-Benchmark Condensed matter theory		50.5%	39.5%	17.1%	41.0%
International Chemistry Olympiad 2025, (theory) Chemistry		82.8%	69.6%	-	72.0%

Qwen3.5-397B-A17B

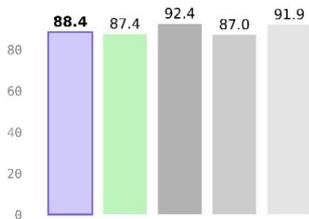
IFBench

Instruction following



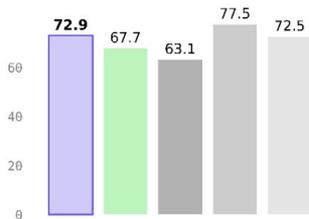
GPQA Diamond

Graduate-level reasoning



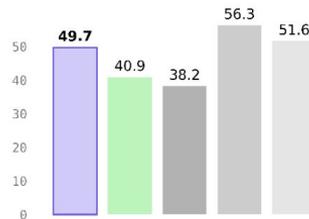
BFCL V4

Agentic tool use



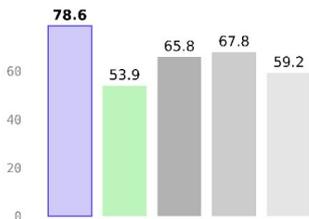
VITA-Bench

Versatile agentic interaction



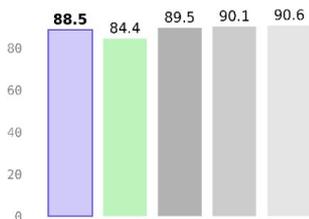
BrowseComp

Agentic search



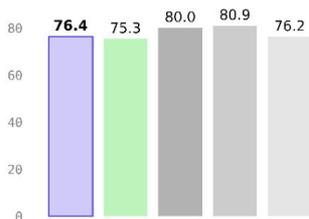
MMMLU

Multilingual knowledge



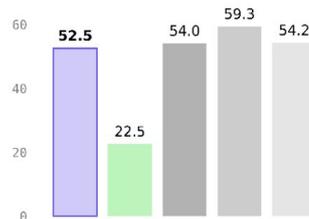
SWE-bench Verified

Agentic coding



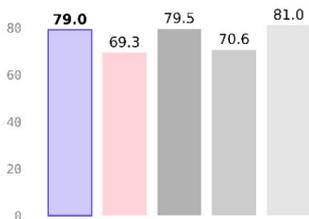
Terminal-Bench 2

Agentic terminal coding



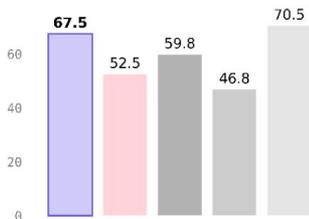
MMMU-Pro

Visual reasoning



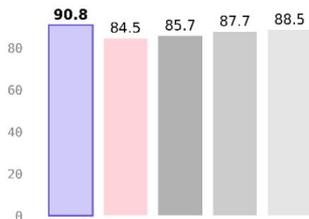
ERQA

Embodied reasoning



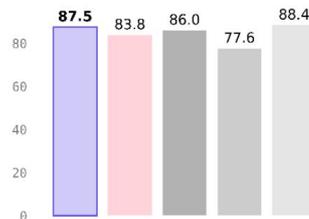
OmniDocBench v1.5

Document recognition & understanding



Video-MME (with subtitle)

Video reasoning



Legend: Qwen3.5-397B-A17B (purple), Qwen3-Max-Thinking (green), Qwen3-VL-235B-A22B (pink), GPT-5.2 (dark grey), Claude Opus 4.5 (medium grey), Gemini 3 Pro (light grey)

Model APIs

- together.ai
- openrouter.ai // unified interface

What are mixture-of-experts?

- The gpt-oss models are autoregressive Mixture-of-Experts (MoE) transformers
 - **gpt-oss-120b**: 36 layers (116.8B total parameters and 5.1B “active” parameters per token)
 - **gpt-oss-20b**: 24 layers (20.9B total and 3.6B active parameters)



Qwen3.5

Model Overview

- Type: Causal Language Model with Vision Encoder
- Training Stage: Pre-training & Post-training
- Language Model
 - Number of Parameters: 397B in total and 17B activated
- Mixture Of Experts
 - Number of Experts: 512
 - Number of Activated Experts: 10 Routed + 1 Shared
 - Expert Intermediate Dimension: 1024

The Bitter Lesson

“The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin.”

Rich Sutton, 2019

Simple architectures—backed by a generous computational budget, data set size and parameter count—surpass more complicated algorithms

OUTRAGEOUSLY LARGE NEURAL NETWORKS: THE SPARSELY-GATED MIXTURE-OF-EXPERTS LAYER

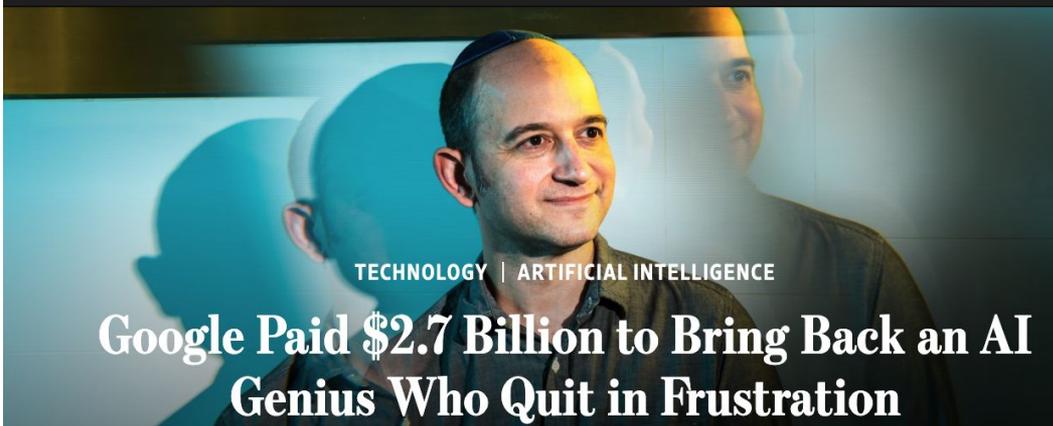
Noam Shazeer¹, Azalia Mirhoseini^{*1}, Krzysztof Maziarz^{*2}, Andy Davis¹, Quoc Le¹, Geoffrey Hinton¹ and Jeff Dean¹

¹Google Brain, {noam,azalia,andydavis,qvl,geoffhinton,jeff}@google.com

²Jagiellonian University, Cracow, krzysztof.maziarz@student.uj.edu.pl

ABSTRACT

The capacity of a neural network to absorb information is limited by its number of parameters. Conditional computation, where parts of the network are active on a per-example basis, has been proposed in theory as a way of dramatically increasing model capacity without a proportional increase in computation. In practice, however, there are significant algorithmic and performance challenges. In this work, we address these challenges and finally realize the promise of conditional



TECHNOLOGY | ARTIFICIAL INTELLIGENCE

Google Paid \$2.7 Billion to Bring Back an AI Genius Who Quit in Frustration

Noam Shazeer

[Google](#)

Verified email at google.com

[Deep Learning](#)



(character.ai)

Sign Up to Chat

Login



Get access to
10M+ Characters

Sign up in just ten seconds

TITLE

CITED BY

YEAR

[Attention is all you need](#)

A Vaswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, AN Gomez, ...
Advances in neural information processing systems 30

209038

2017

[Exploring the limits of transfer learning with a unified text-to-text transformer](#)

C Raffel, N Shazeer, A Roberts, K Lee, S Narang, M Matena, Y Zhou, W Li, ...
Journal of machine learning research 21 (140), 1-67

27346

2020

Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity

William Fedus*

LIAMFEDUS@GOOGLE.COM

Barret Zoph*

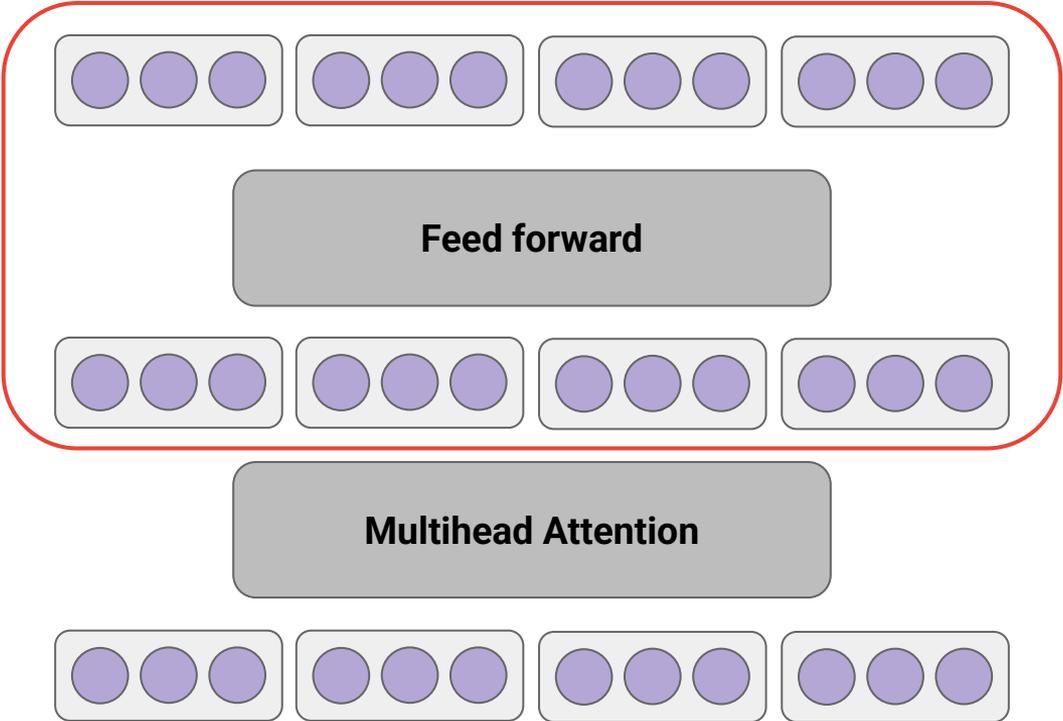
BARRETZOPH@GOOGLE.COM

Noam Shazeer

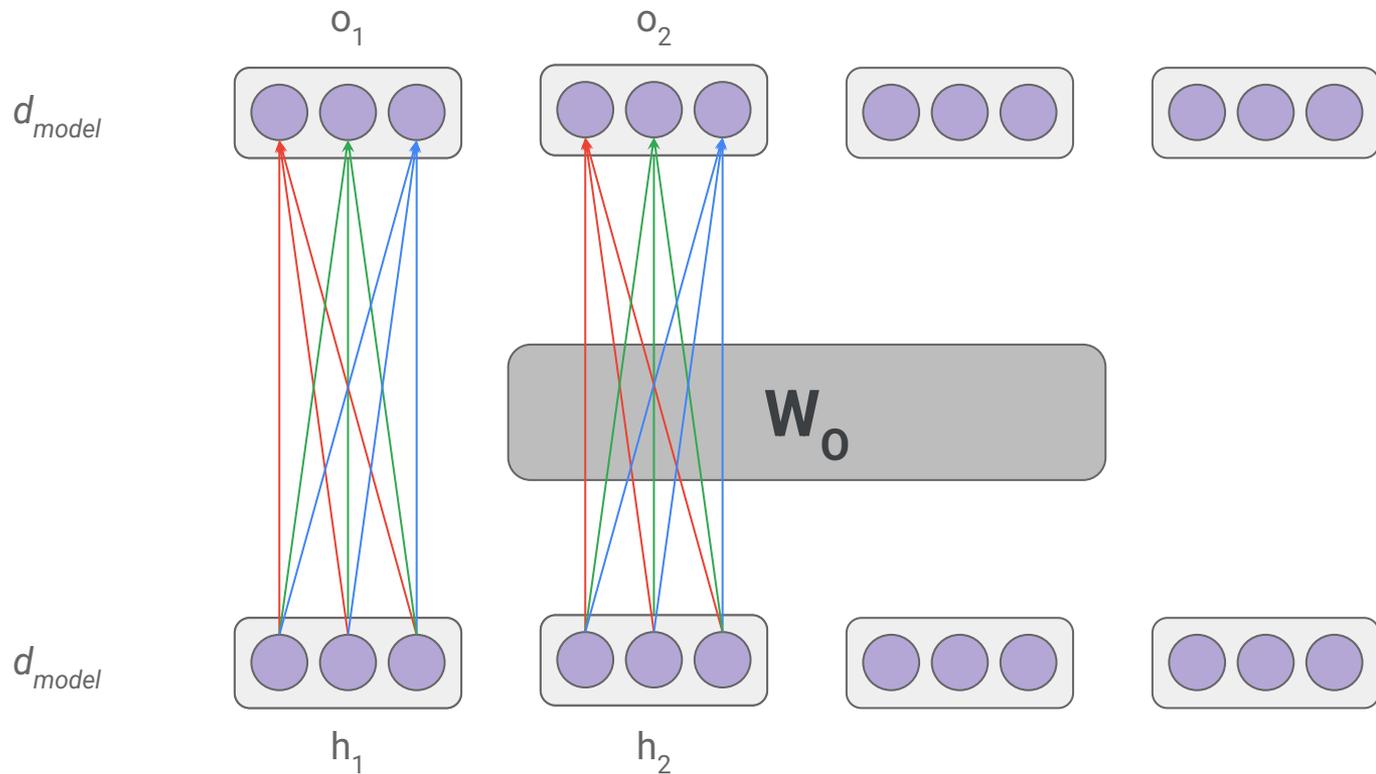
NOAM@GOOGLE.COM

Google, Mountain View, CA 94043, USA

Transformer block recap



Position-wise feedforward networks

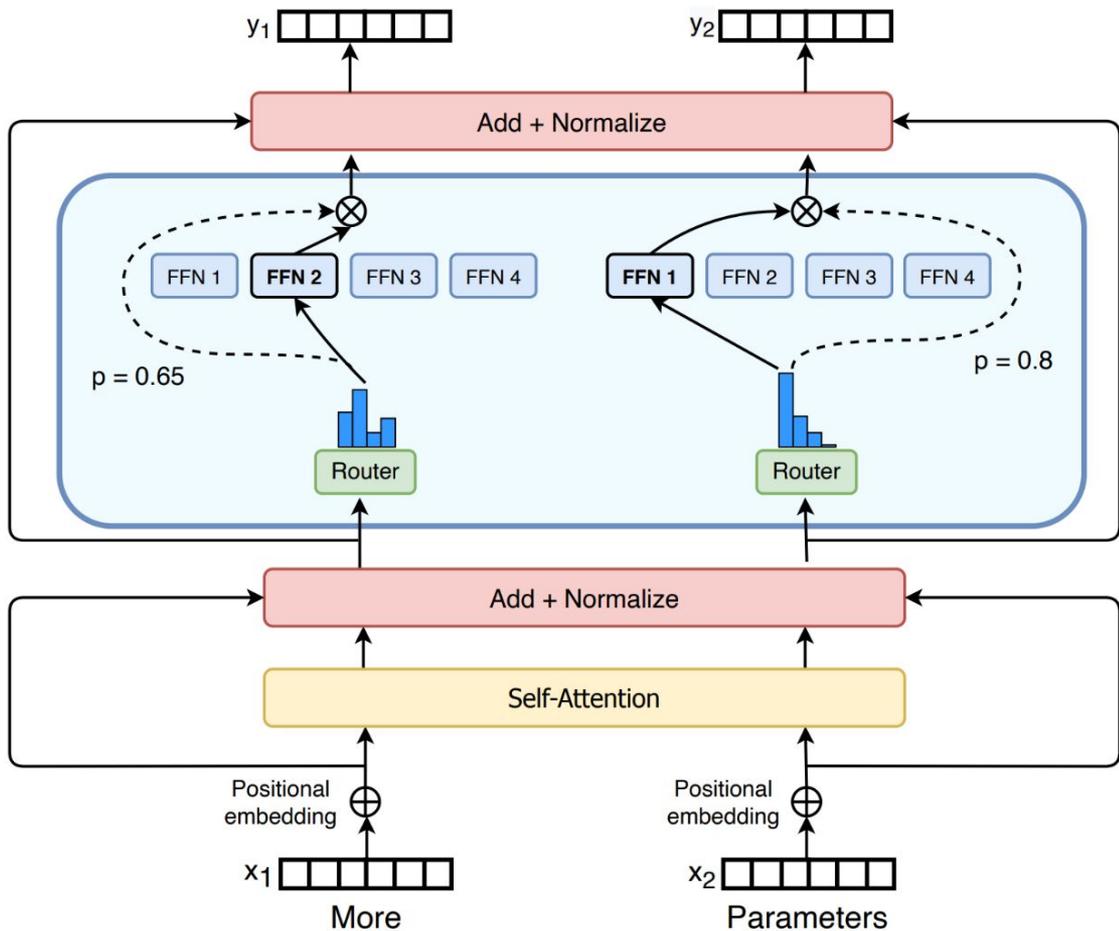
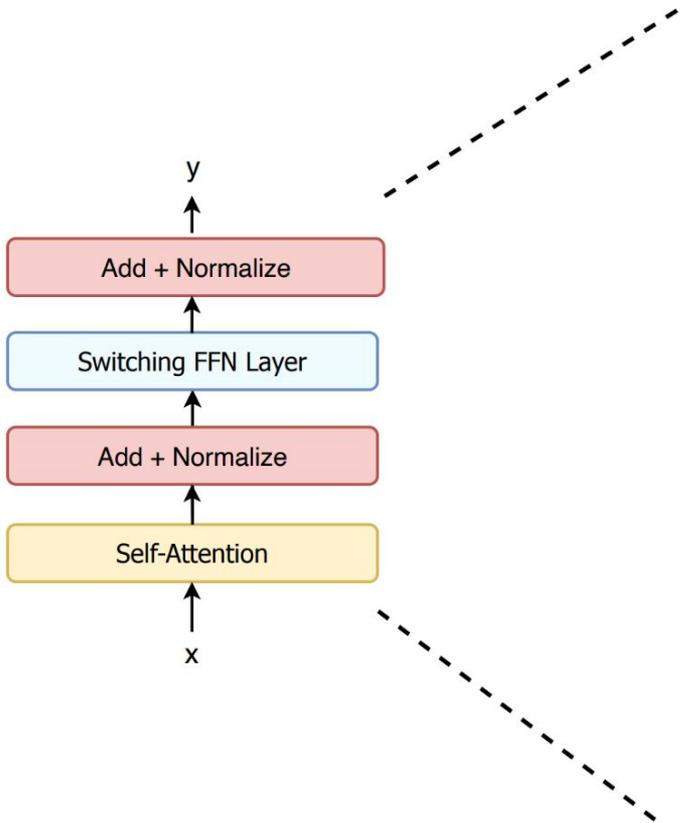


$$O = H \cdot W_0$$

**linear
projections**

Switch Transformers

The layer operates independently on the tokens



Mixture of Expert Routing

The MoE layer takes as an input a token representation x and then routes this to the best determined top- k experts, selected from a set $\{E_i(x)\}_{i=1}^N$ of N experts.

The router variable W_r produces logits $h(x) = W_r \cdot x$, which are normalized via a softmax distribution over the available N experts at that layer. The gate value for expert i is given by:

$$p_i(x) = \frac{e^{h(x)_i}}{\sum_j e^{h(x)_j}}$$

The top- k gate values are selected for routing the token x . If T is the set of selected top- k indices, then the output computation of the layer is the linearly weighted combination of each expert's computation on the token by the gate value:

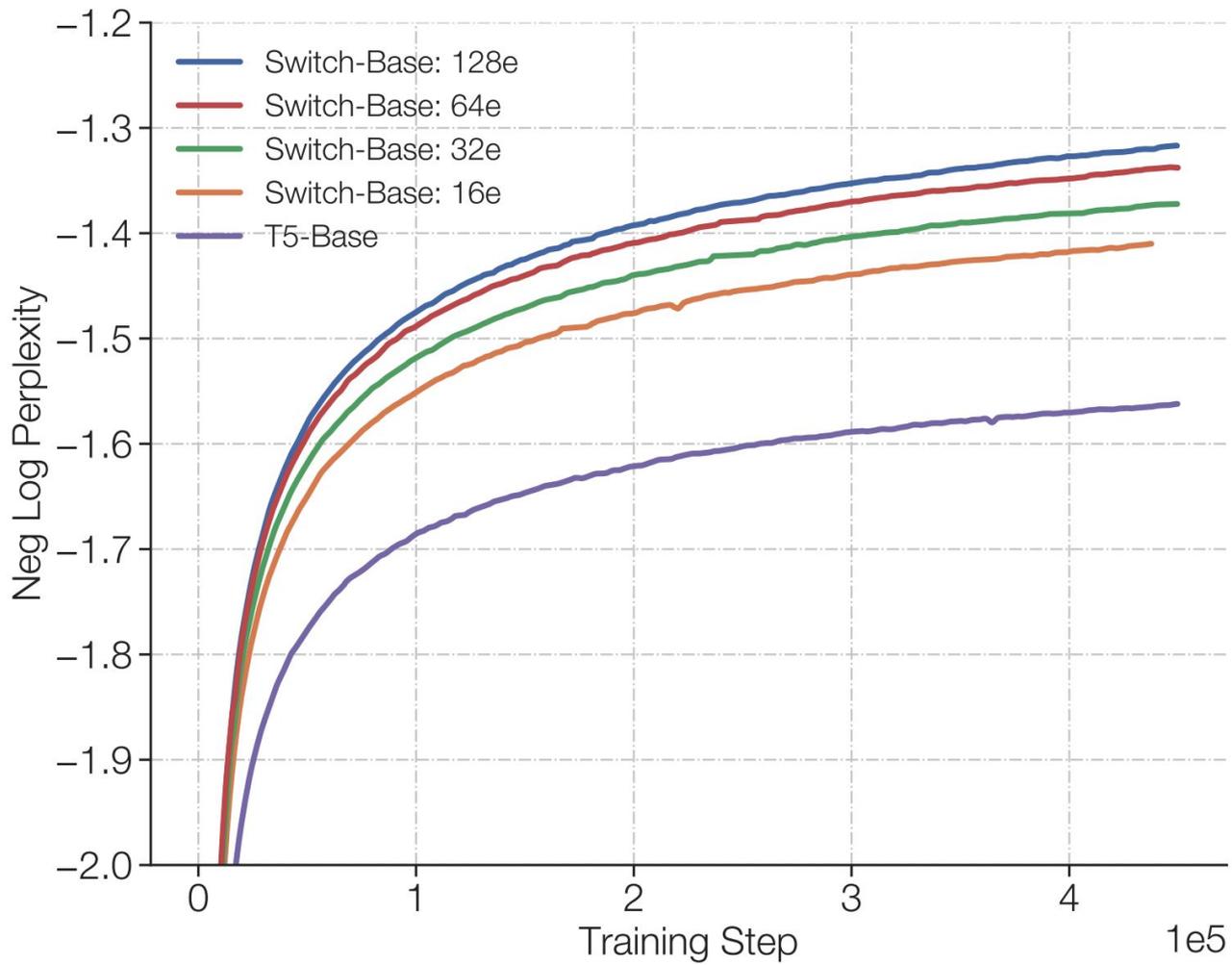
$$y = \sum_{i \in T} p_i(x) E_i(x)$$

Switch Transformers

- Vanilla Transformer
 - densely-activated
- Switch Transformer
 - sparsely-activated expert model
 - with an outrageous number of parameters—but a constant computational cost (!)
 - pretraining up to *trillion* parameter models and achieving a 4x speedup over the T5-XXL (11B)

Rethinking Mixture-of-Experts

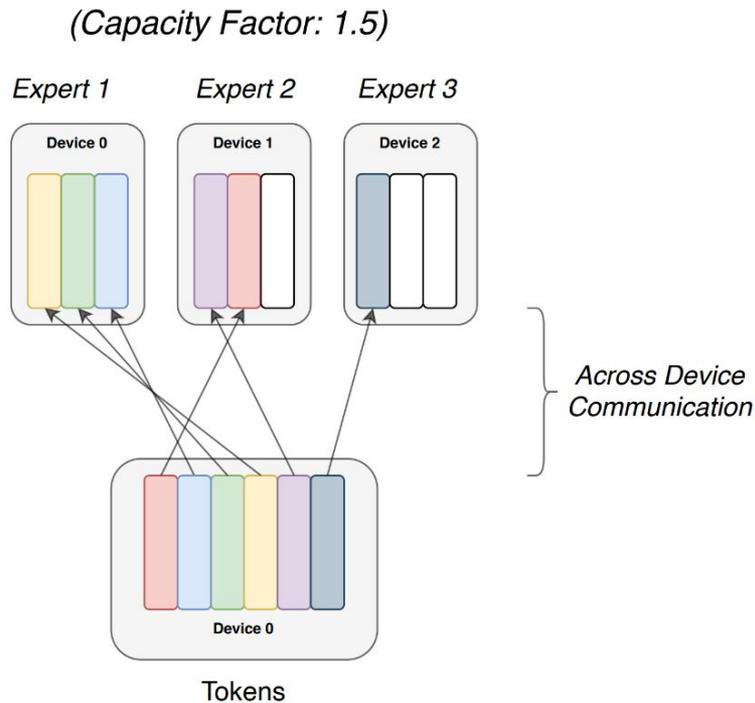
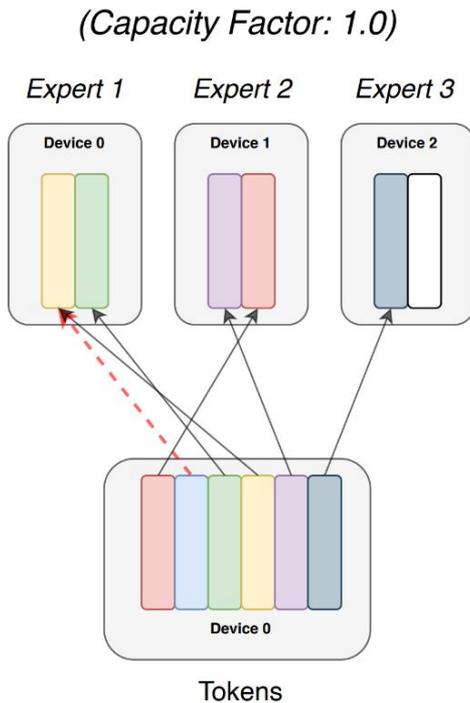
- [Shazeer et al. \(2017\)](#)
 - Each token is processed by every expert, and their outputs are combined
 - This increases computational cost linearly with the number of experts, so adding experts makes training and inference more expensive
- Switch layer
 - Only a small subset of experts are activated per token
 - This allows the model to scale to many more experts while keeping the computational cost per token roughly constant



$$\text{expert capacity} = \left(\frac{\text{tokens per batch}}{\text{number of experts}} \right) \times \text{capacity factor}$$

Terminology

- **Experts:** Split across devices, each having their own unique parameters. Perform standard feed-forward computation.
- **Expert Capacity:** Batch size of each expert. Calculated as $(\text{tokens_per_batch} / \text{num_experts}) * \text{capacity_factor}$
- **Capacity Factor:** Used when calculating expert capacity. Expert capacity allows more buffer to help mitigate token overflow during routing.

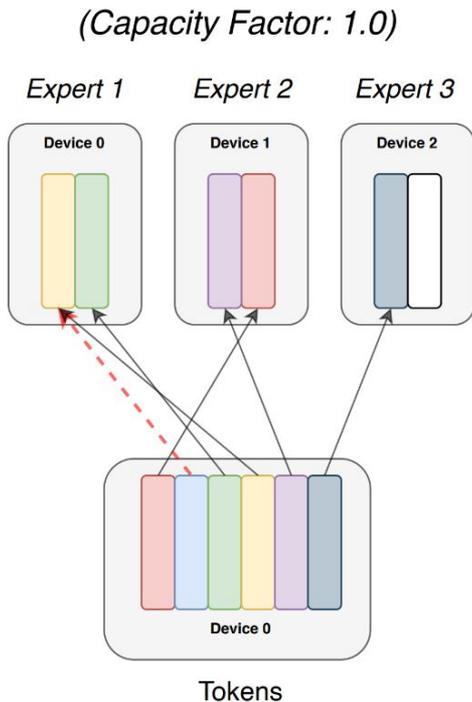


What would happen with the blue (dropped) token?

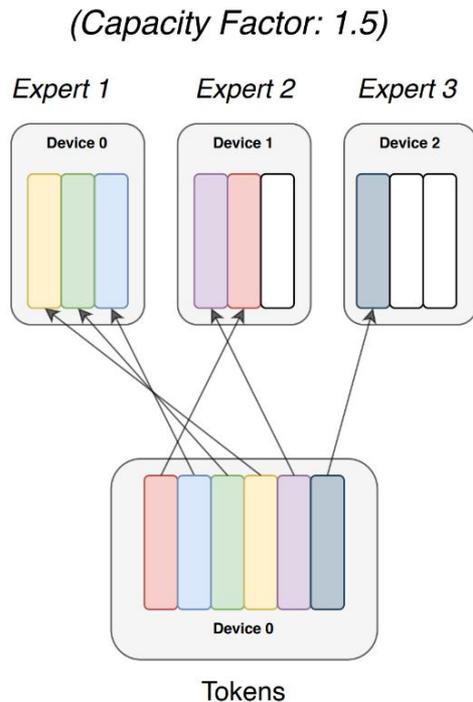
$$\text{expert capacity} = \left(\frac{\text{tokens per batch}}{\text{number of experts}} \right) \times \text{capacity factor}$$

Terminology

- **Experts:** Split across devices, each having their own unique parameters. Perform standard feed-forward computation.
- **Expert Capacity:** Batch size of each expert. Calculated as $(\text{tokens_per_batch} / \text{num_experts}) * \text{capacity_factor}$
- **Capacity Factor:** Used when calculating expert capacity. Expert capacity allows more buffer to help mitigate token overflow during routing.



overflow



wasted computation & memory

An auxiliary load balancing loss

Given N experts indexed by $i = 1$ to N and a batch B with T tokens, the auxiliary loss is computed as the scaled dot-product between vectors f and P :

$$\text{loss} = \alpha \cdot N \sum_{i=1}^N f_i \cdot P_i$$

f_i : fraction of tokens to expert i

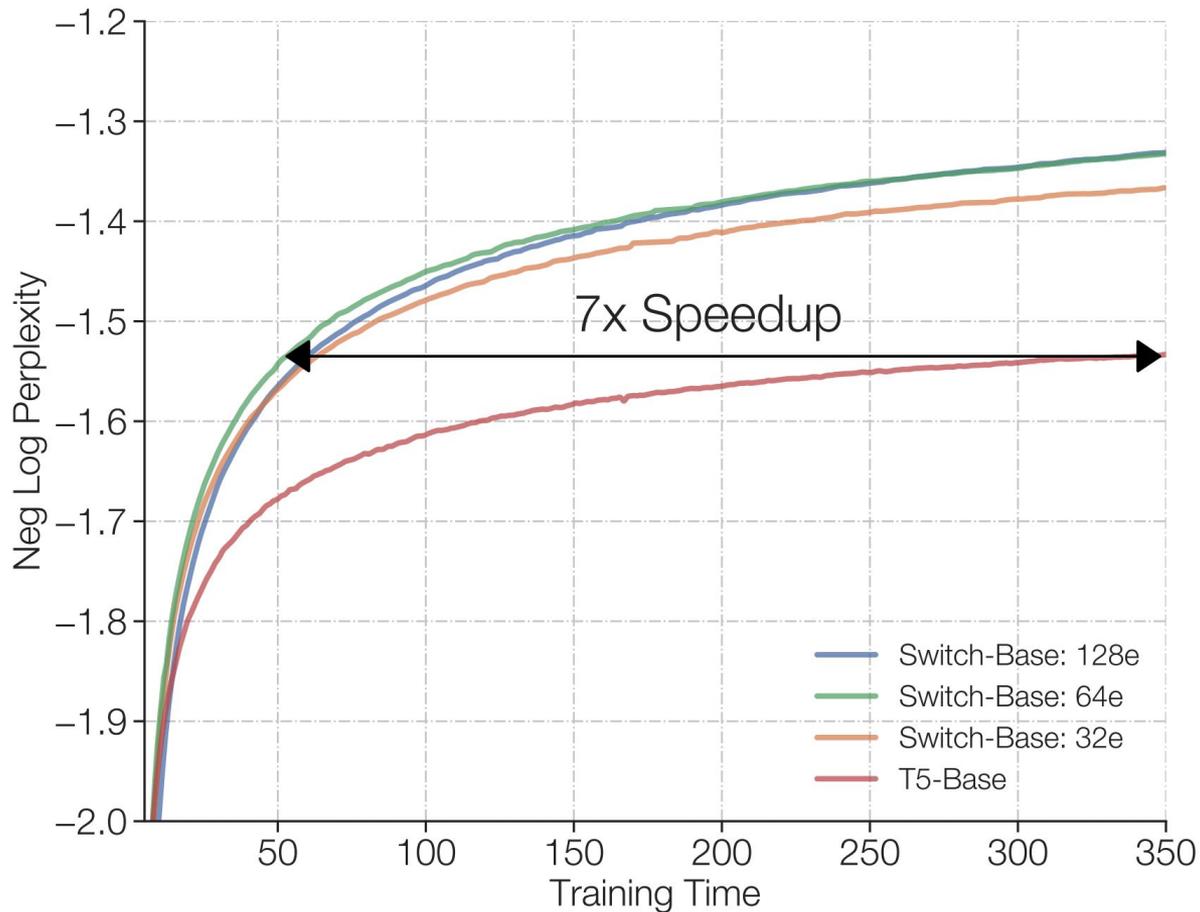
- For each token in the batch, check which expert got chosen.
- Count how many times expert i was picked.
- Divide by the total number of tokens T to get the fraction.

P_i : router probability for expert i

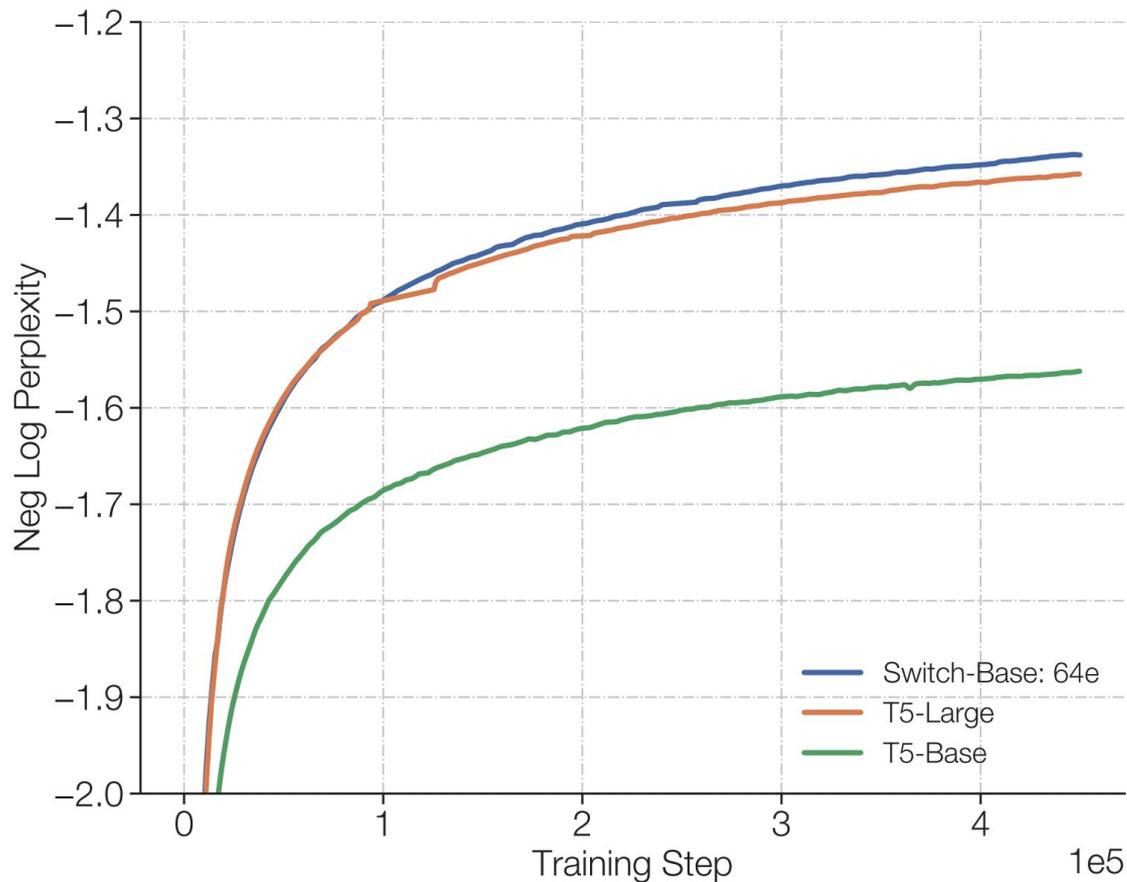
- The router assigns probabilities to each expert for every token.
- For each token, take the probability that expert i was preferred.
- Average over all tokens.

The auxiliary loss encourages uniform routing since it is minimized under a uniform distribution

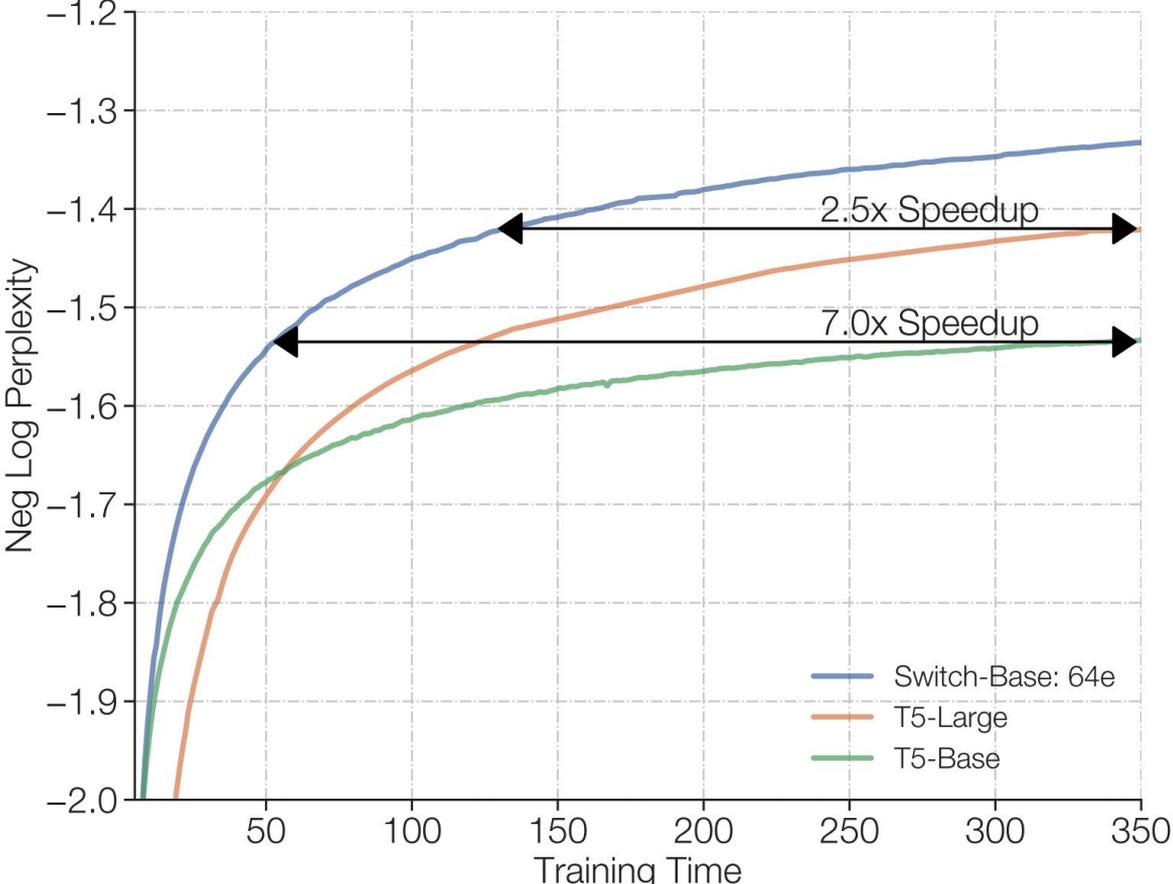
Speed advantage of Switch Transformer



Switch-Base is more sample efficient than T5-Large



Switch-Base is faster than T5-Large (2.5x speedup)



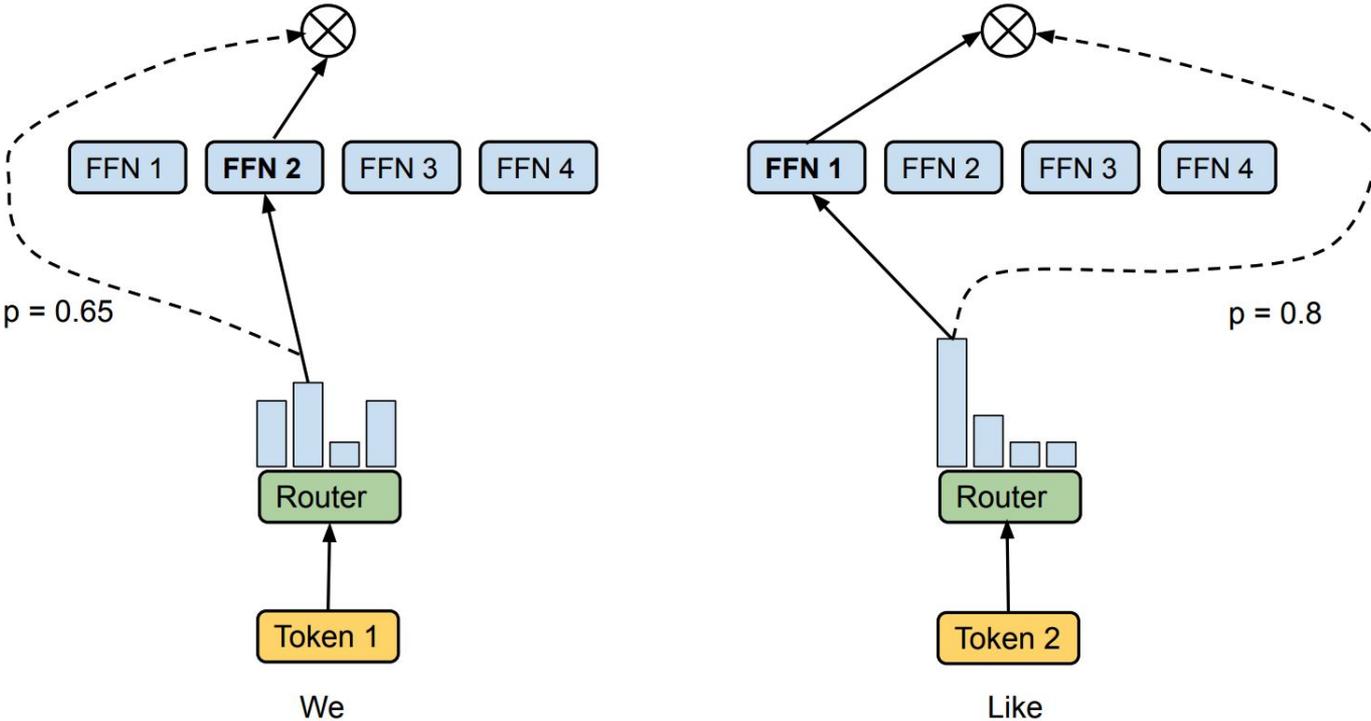
... and significant downstream improvements

Model	GLUE	SQuAD	SuperGLUE	Winogrande (XL)
T5-Base	84.3	85.5	75.1	66.6
Switch-Base	86.7	87.2	79.5	73.3
T5-Large	87.8	88.1	82.7	79.1
Switch-Large	88.5	88.6	84.7	83.0

Model	XSum	ANLI (R3)	ARC Easy	ARC Chal.
T5-Base	18.7	51.8	56.7	35.5
Switch-Base	20.3	54.0	61.3	32.8
T5-Large	20.9	56.6	68.8	35.5
Switch-Large	22.3	58.6	66.0	35.5

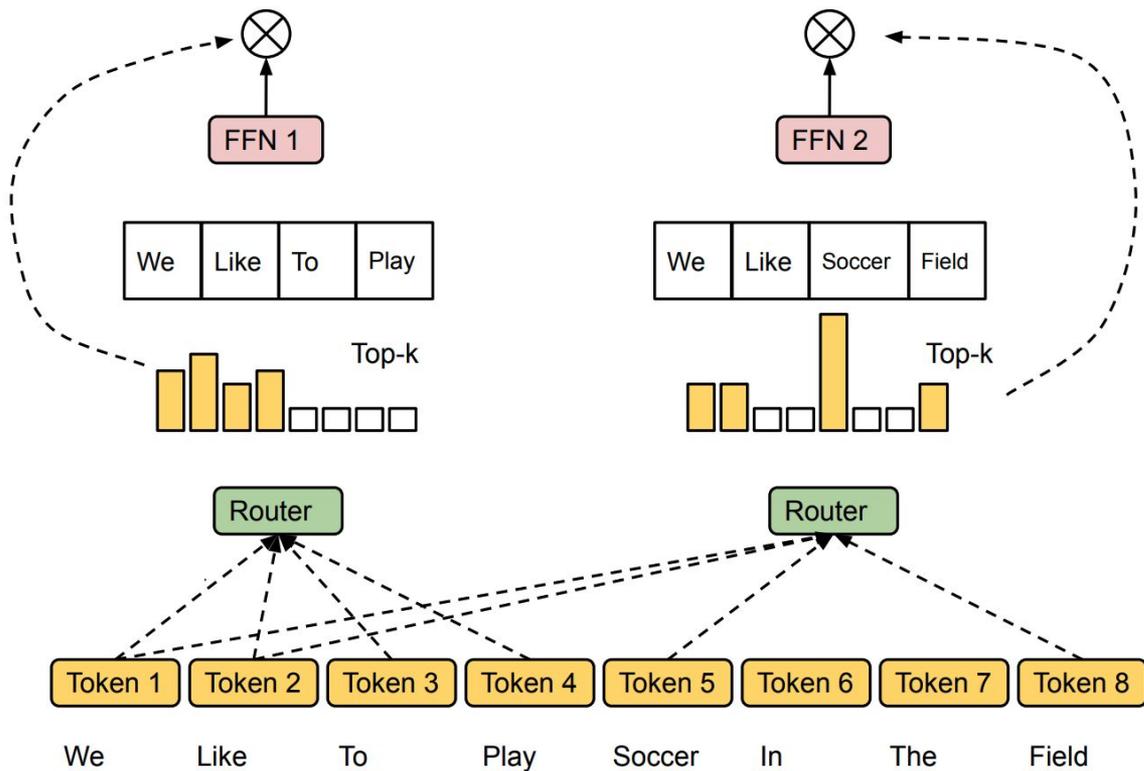
Model	CB Web QA	CB Natural QA	CB Trivia QA
T5-Base	26.6	25.8	24.5
Switch-Base	27.4	26.8	30.7
T5-Large	27.7	27.6	29.5
Switch-Large	31.3	29.5	36.9

Token-choice routing



[“Mixture-of-Experts with Expert Choice Routing” by Zhou et al. \(2022\)](#)

Expert-choice routing



When to use sparse MoEs vs dense models?

Experts are useful for high throughput scenarios with many machines. Given a fixed compute budget for pretraining, a sparse model will be more optimal. For low throughput scenarios with little VRAM, a dense model will be better.

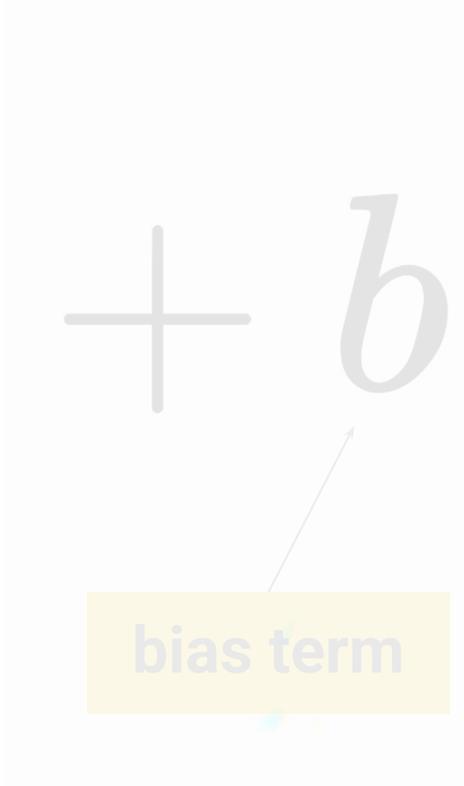
Parameter-efficient fine-tuning (PEFT)

Model parameters (weights)

- Embedding matrices
 - Word embeddings, positional embeddings, segment embeddings
- Weight matrices
 - E.g., W_Q , W_K , W_V , W_O
- Bias terms

Bias term

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



bias term

Updating model parameters

$$w_{t+1} = w_t - \eta \cdot \frac{\partial L}{\partial w_t}$$

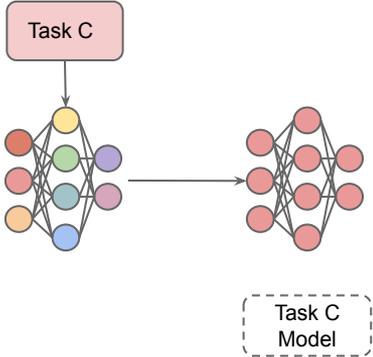
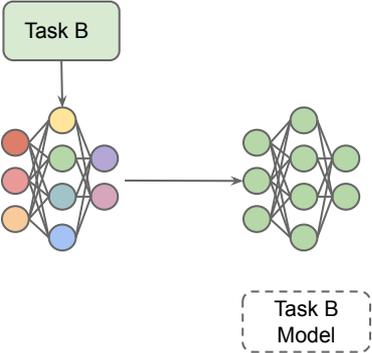
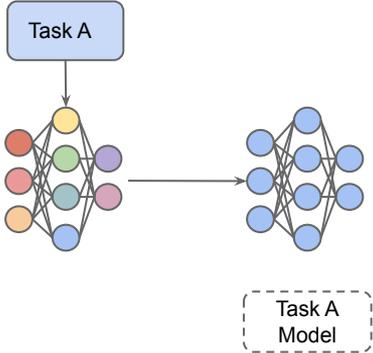
Where:

- w_t is the parameter at the current time step.
- w_{t+1} is the updated parameter after applying the gradient.
- η is the learning rate, which controls the step size.
- $\frac{\partial L}{\partial w_t}$ is the gradient of the loss function L with respect to the parameter w_t , representing how the loss changes as the parameter changes.

Updating model parameters (cont'd)

$$W' = W + \Delta W$$

Full model fine-tuning (Full FT)



Limitations of full model tuning

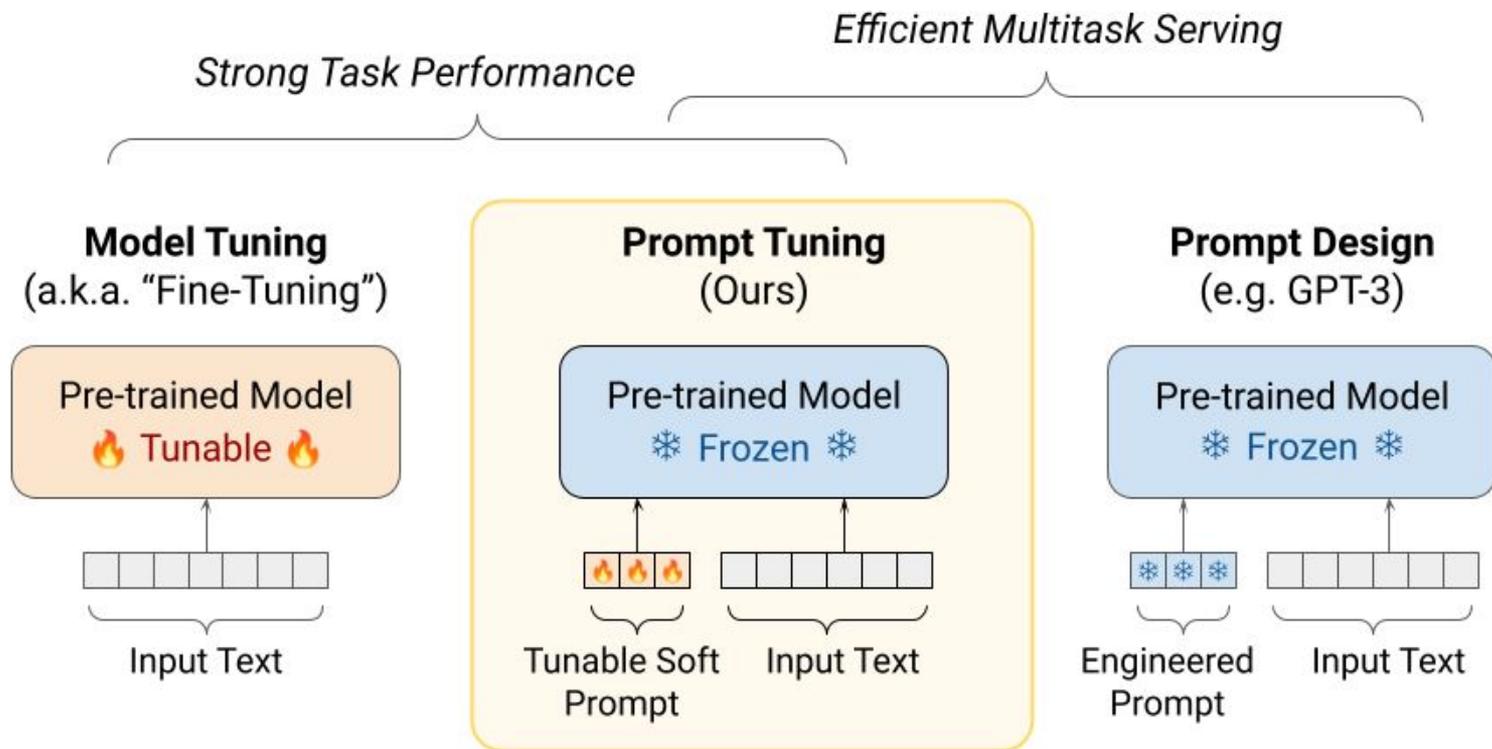
The Power of Scale for Parameter-Efficient Prompt Tuning

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

Google Research

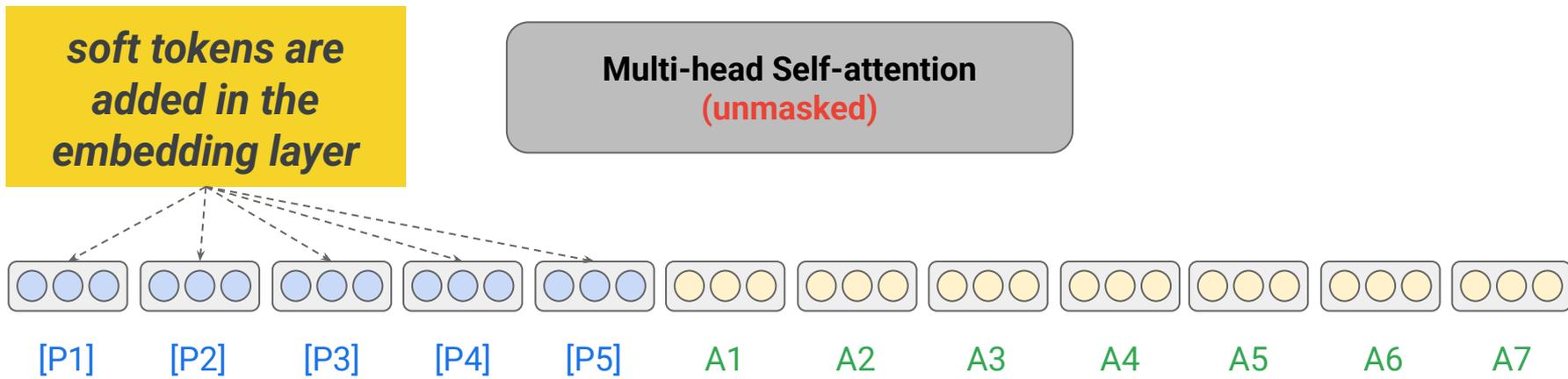
`{brianlester, rmyeid, nconstant}@google.com`

Soft prompt tuning



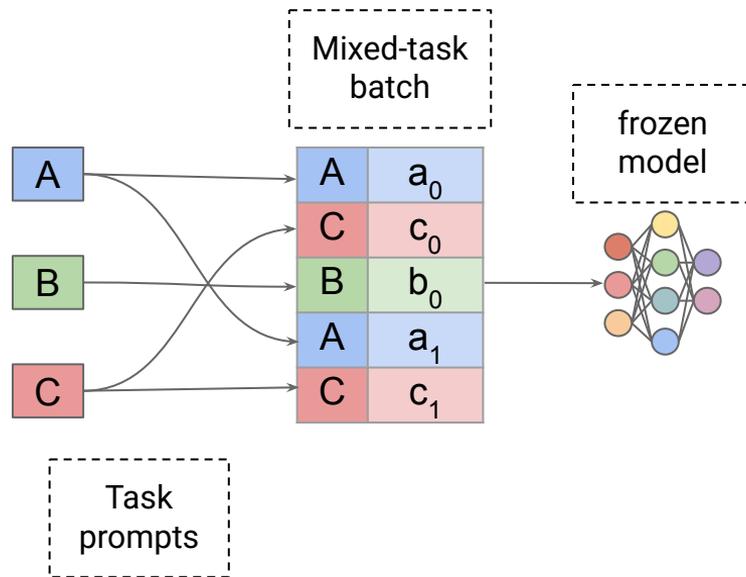
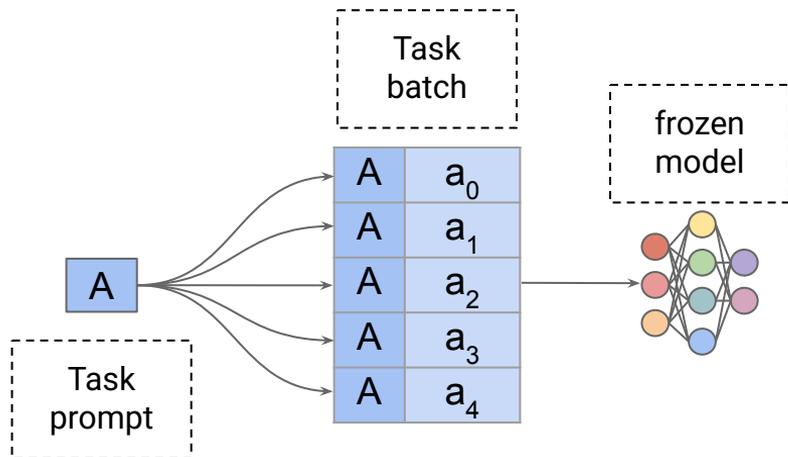
["The Power of Scale for Parameter-Efficient Prompt Tuning" by Lester et al. \(2021\)](#)

Soft prompt

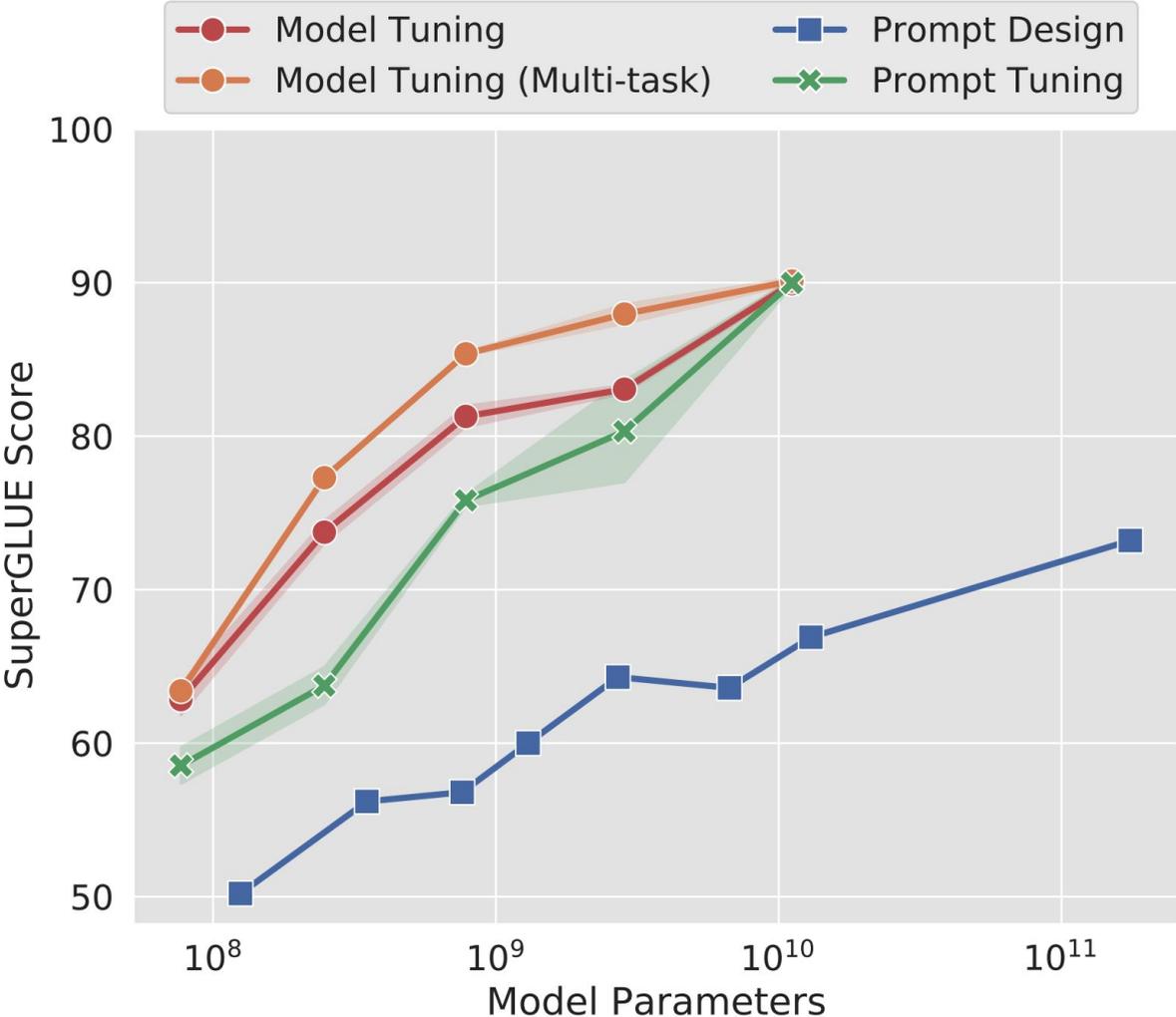


Advantages of soft prompt tuning

Parameter-efficient tuning & mixed-task inference



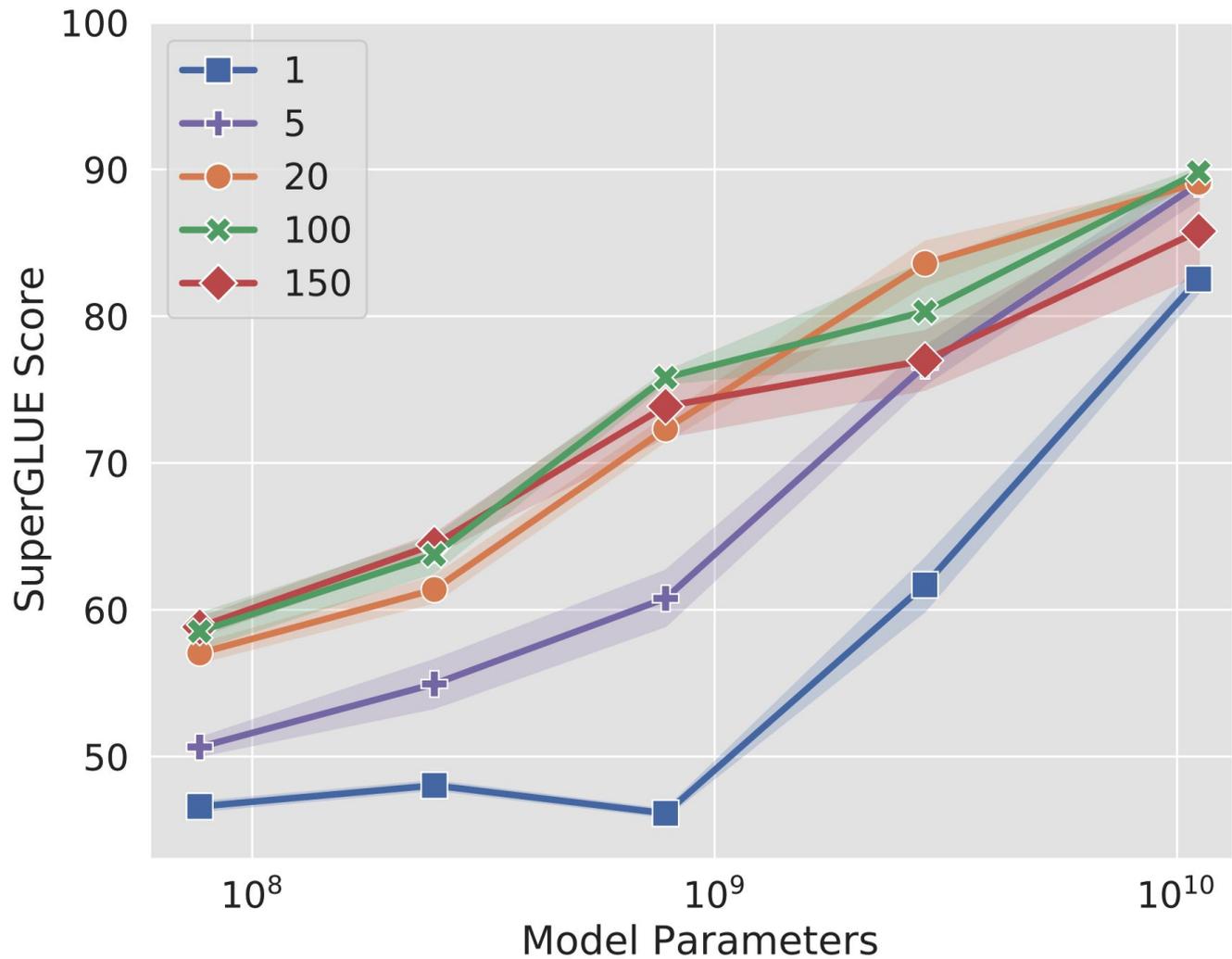
Improvement with Scale



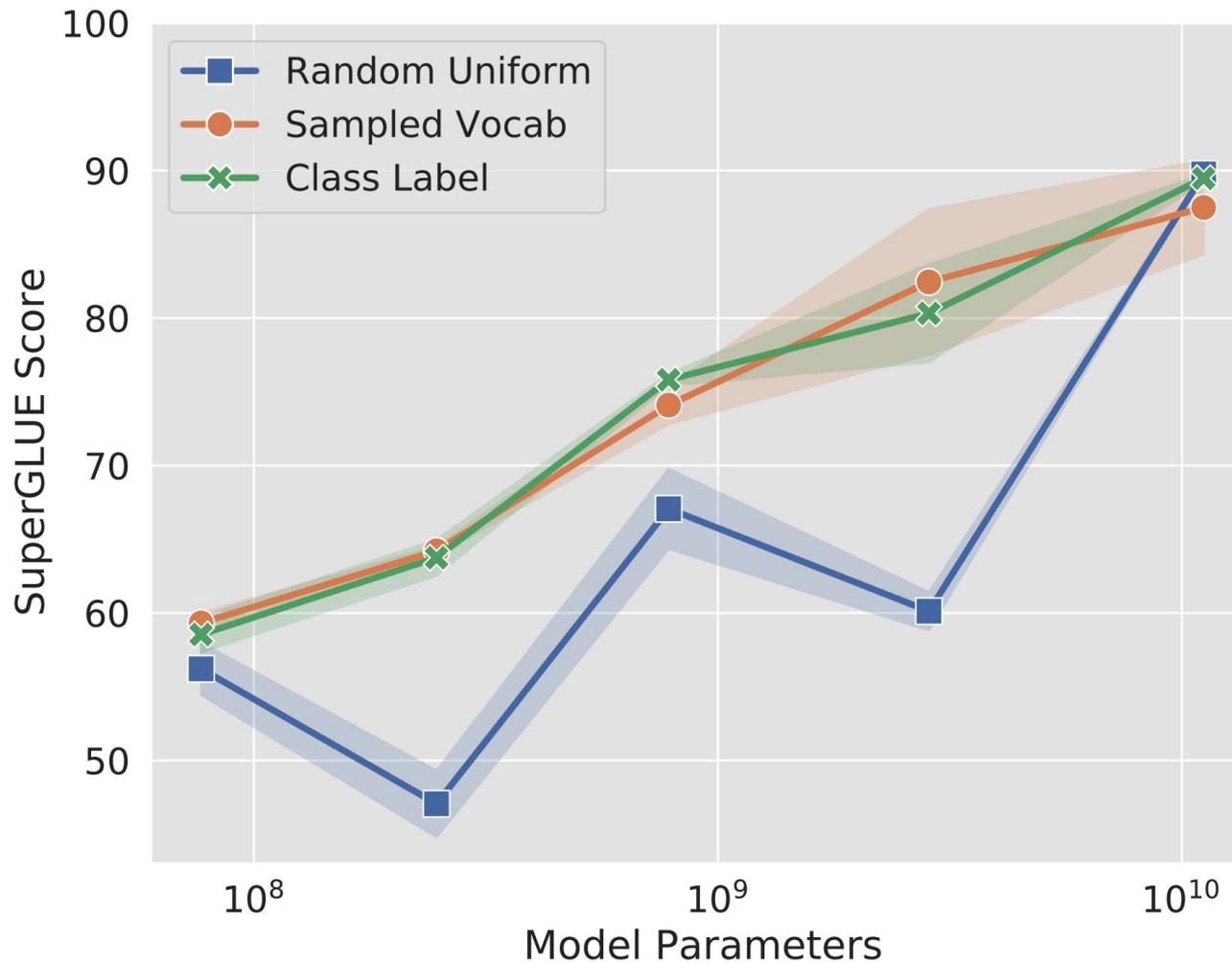
Resilience to domain shift

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

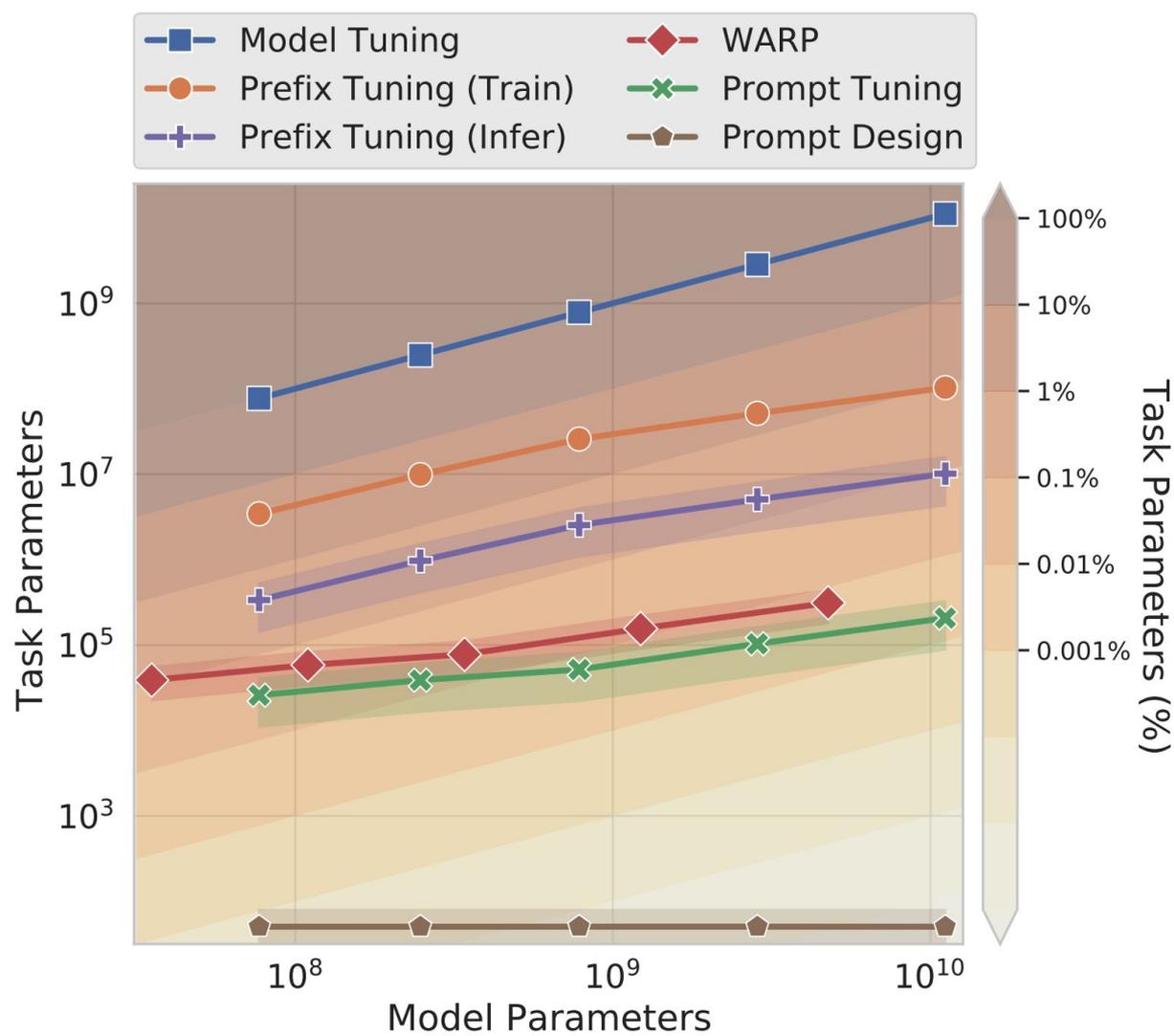
Effect of prompt length



Effect of prompt initialization



Parameter usage



Interpretability

- the learned prompts taken as sequences show little interpretability

SPoT: Better Frozen Model Adaptation through Soft Prompt Transfer

Tu Vu^{1,2}★ **Brian Lester**¹ **Noah Constant**¹ **Rami Al-Rfou**¹ **Daniel Cer**¹

Google Research¹

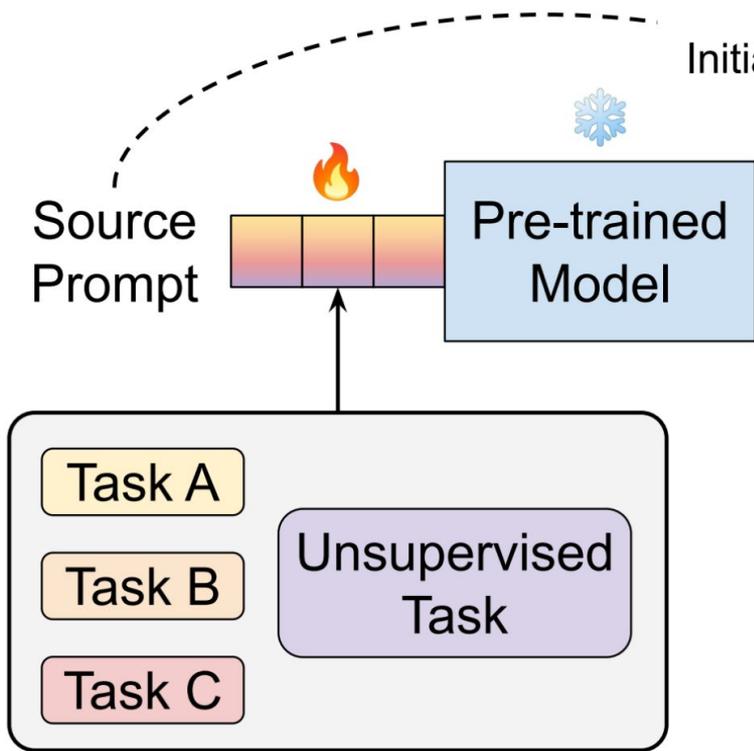
University of Massachusetts Amherst²

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

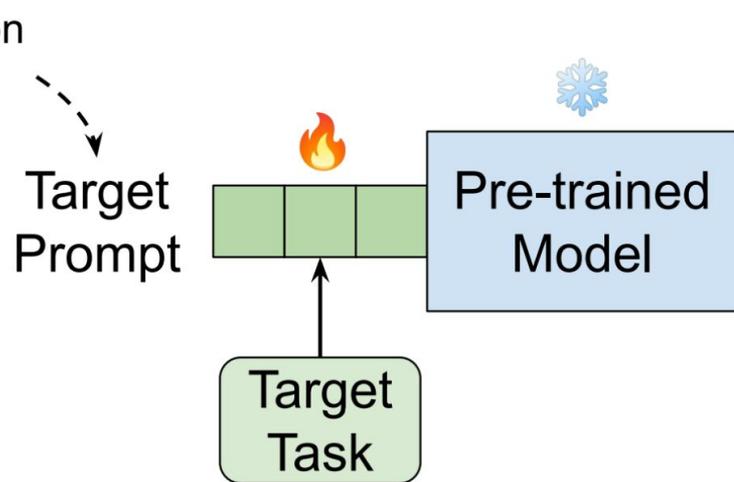
tuvu@cs.umass.edu

Generic SPoT

Source Prompt Tuning

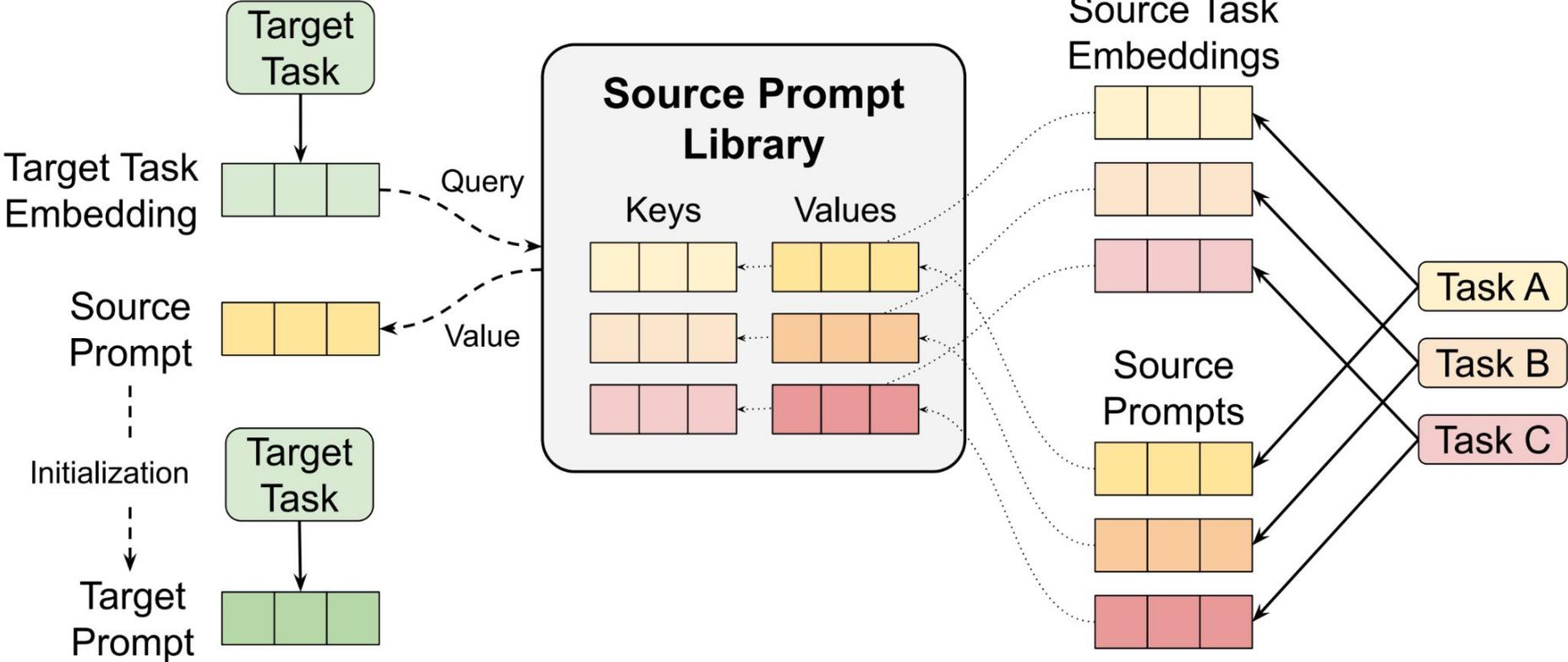


Target Prompt Tuning



 tuned
 frozen

Targeted SPoT



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

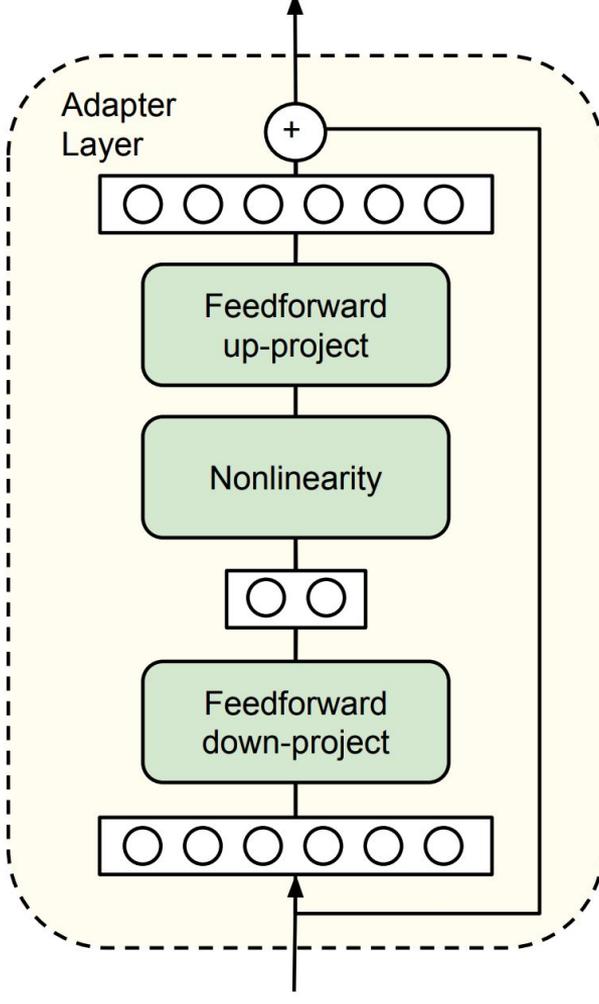
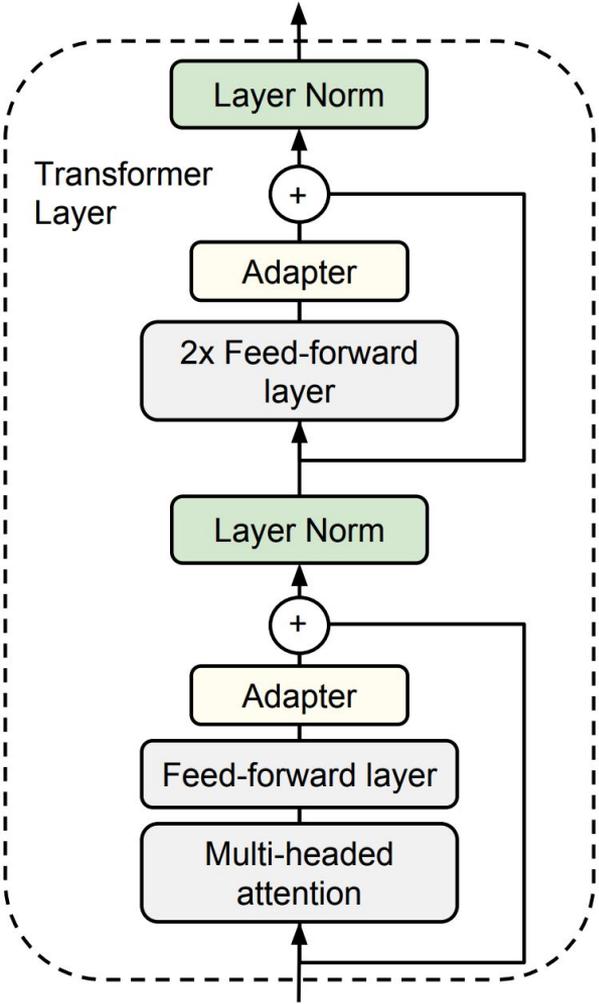
²Allen Institute for Artificial Intelligence

{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*** **Phillip Wallis** **Zeyuan Allen-Zhu**
Yuanzhi Li **Shean Wang** **Lu Wang** **Weizhu Chen**

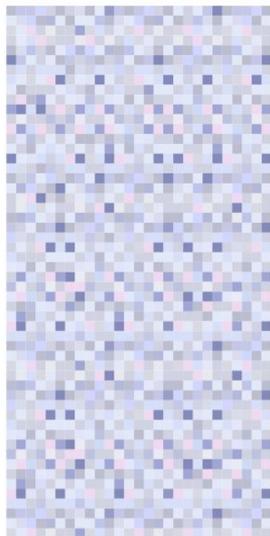
Microsoft Corporation

{edwardhu, yeshe, phwallis, zeyuana,
yuanzhil, swang, luw, wzchen}@microsoft.com
yuanzhil@andrew.cmu.edu

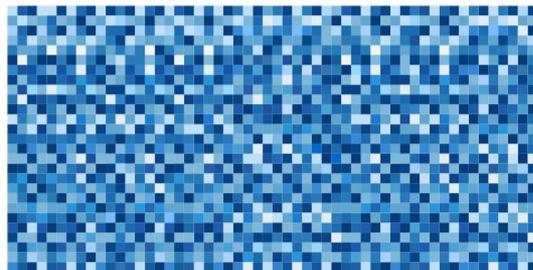
Weight changes during model adaptation have a low “intrinsic rank”

- 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

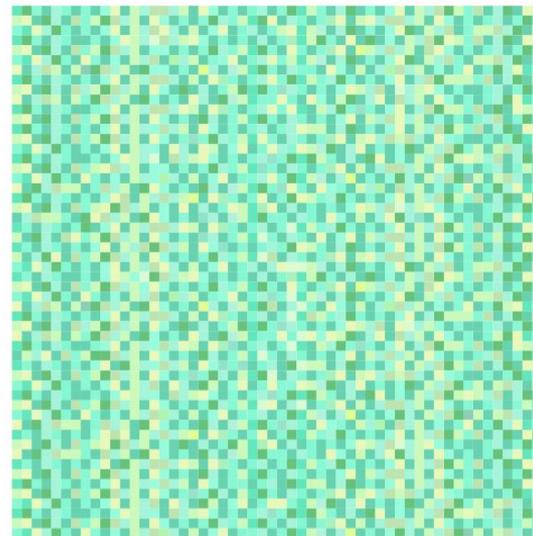
LoRA



B
 $d \times r$



A
 $r \times k$



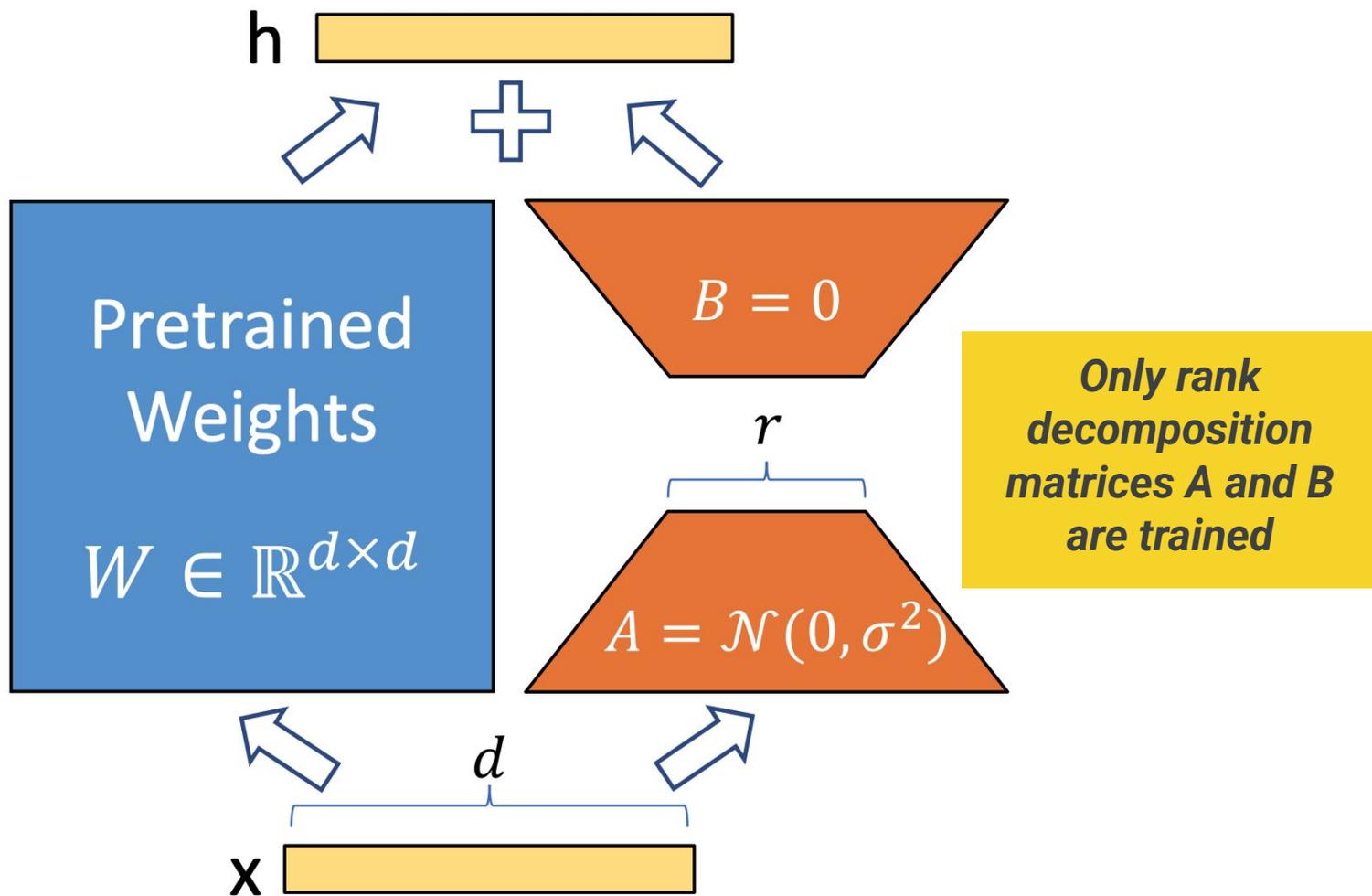
$\Delta W = \gamma \times B \times A$
 $d \times k$

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

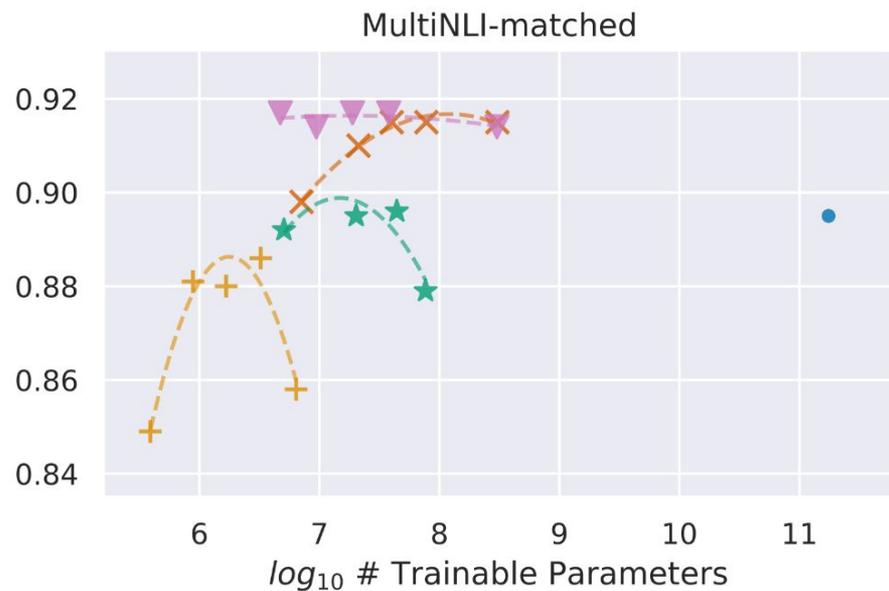
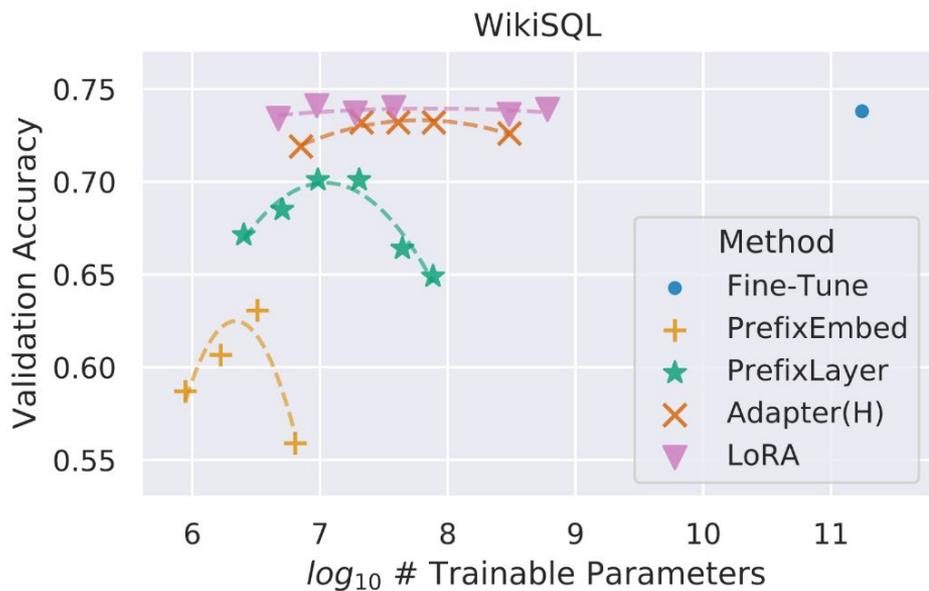


Advantages of LoRA

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 \pm 0	94.2 \pm 1	88.5 \pm 1.1	60.8 \pm 4	93.1 \pm 1	90.2 \pm 0	71.5 \pm 2.7	89.7 \pm 3	84.4
RoB _{base} (Adpt ^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 \pm 2.2	90.3 \pm 1	85.4
RoB _{base} (LoRA)	0.3M	87.5 \pm 3	95.1\pm2	89.7 \pm 7	63.4 \pm 1.2	93.3\pm3	90.8 \pm 1	86.6\pm7	91.5\pm2	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\pm2	96.2 \pm 5	90.9\pm1.2	68.2\pm1.9	94.9\pm3	91.6 \pm 1	87.4\pm2.5	92.6\pm2	89.0
RoB _{large} (Adpt ^P)†	3.0M	90.2 \pm 3	96.1 \pm 3	90.2 \pm 7	68.3\pm1.0	94.8\pm2	91.9\pm1	83.8 \pm 2.9	92.1 \pm 7	88.4
RoB _{large} (Adpt ^P)†	0.8M	90.5\pm3	96.6\pm2	89.7 \pm 1.2	67.8 \pm 2.5	94.8\pm3	91.7 \pm 2	80.1 \pm 2.9	91.9 \pm 4	87.9
RoB _{large} (Adpt ^H)†	6.0M	89.9 \pm 5	96.2 \pm 3	88.7 \pm 2.9	66.5 \pm 4.4	94.7 \pm 2	92.1 \pm 1	83.4 \pm 1.1	91.0 \pm 1.7	87.8
RoB _{large} (Adpt ^H)†	0.8M	90.3 \pm 3	96.3 \pm 5	87.7 \pm 1.7	66.3 \pm 2.0	94.7 \pm 2	91.5 \pm 1	72.9 \pm 2.9	91.5 \pm 5	86.4
RoB _{large} (LoRA)†	0.8M	90.6\pm2	96.2 \pm 5	90.2\pm1.0	68.2 \pm 1.9	94.8\pm3	91.6 \pm 2	85.2\pm1.1	92.3\pm5	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\pm2	96.9 \pm 2	92.6\pm6	72.4\pm1.1	96.0\pm1	92.9\pm1	94.9\pm4	93.0\pm2	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 8	W_k 8	W_v 8	W_o 8	W_q, W_k 4	W_q, W_v 4	W_q, W_k, W_v, W_o 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

The effect of rank r on model performance

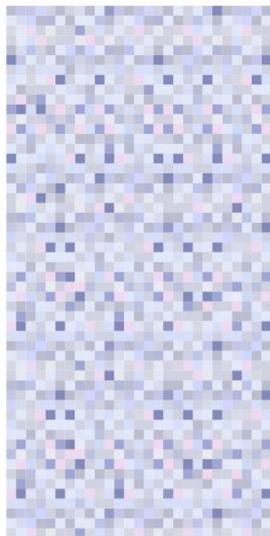
	Weight Type	$r = 1$	$r = 2$	$r = 4$	$r = 8$	$r = 64$
WikiSQL($\pm 0.5\%$)	W_q	68.8	69.6	70.5	70.4	70.0
	W_q, W_v	73.4	73.3	73.7	73.8	73.5
	W_q, W_k, W_v, W_o	74.1	73.7	74.0	74.0	73.9
MultiNLI ($\pm 0.1\%$)	W_q	90.7	90.9	91.1	90.7	90.7
	W_q, W_v	91.3	91.4	91.3	91.6	91.4
	W_q, W_k, W_v, W_o	91.2	91.7	91.7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

practical recommendations

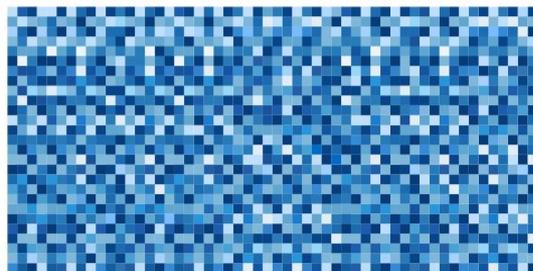
of training examples

- < 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

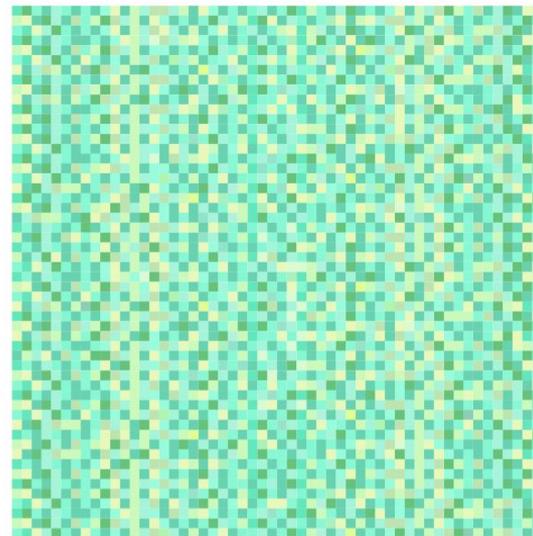
LoRA Without Regret



B
 $d \times r$



A
 $r \times k$



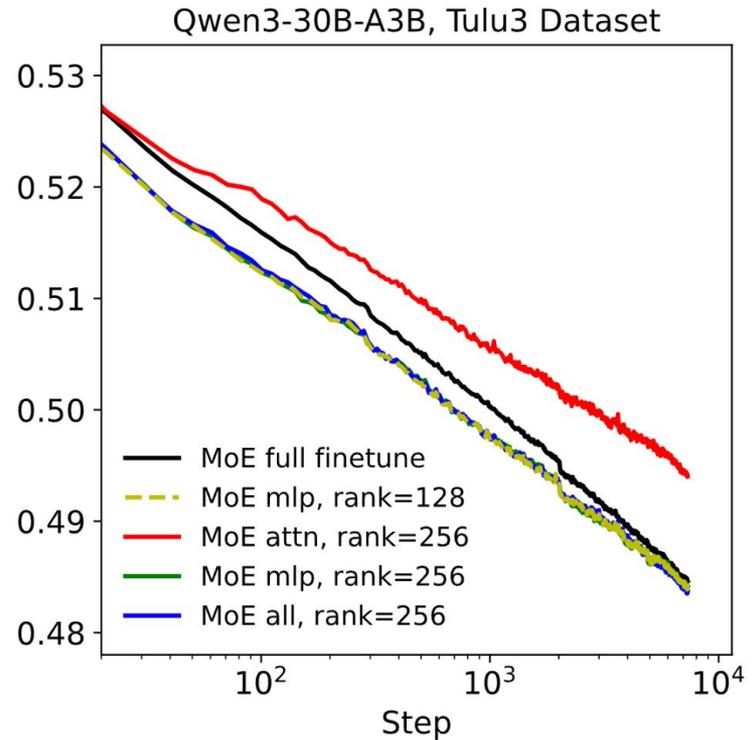
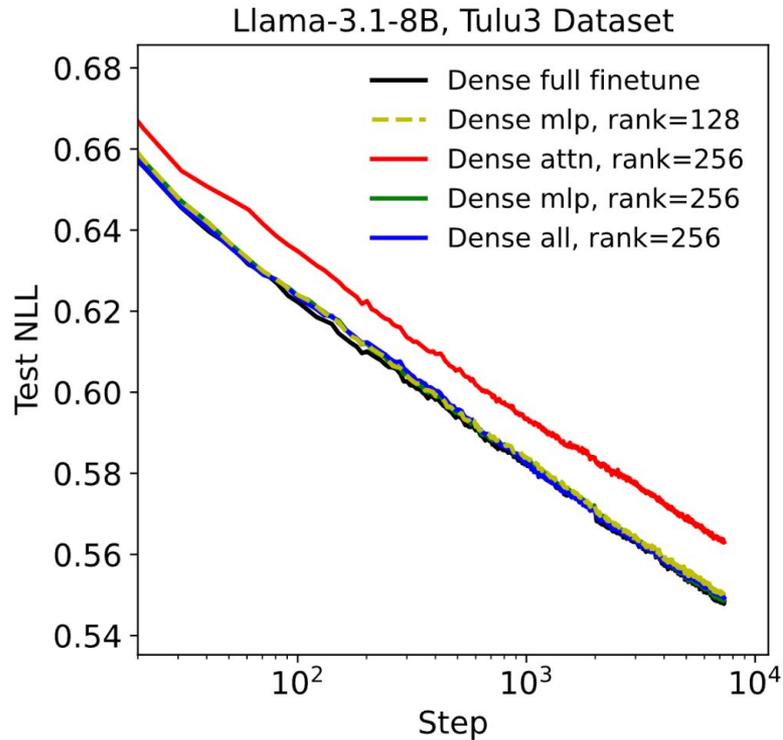
$\Delta W = \gamma \times B \times A$
 $d \times k$

LoRA Without Regret

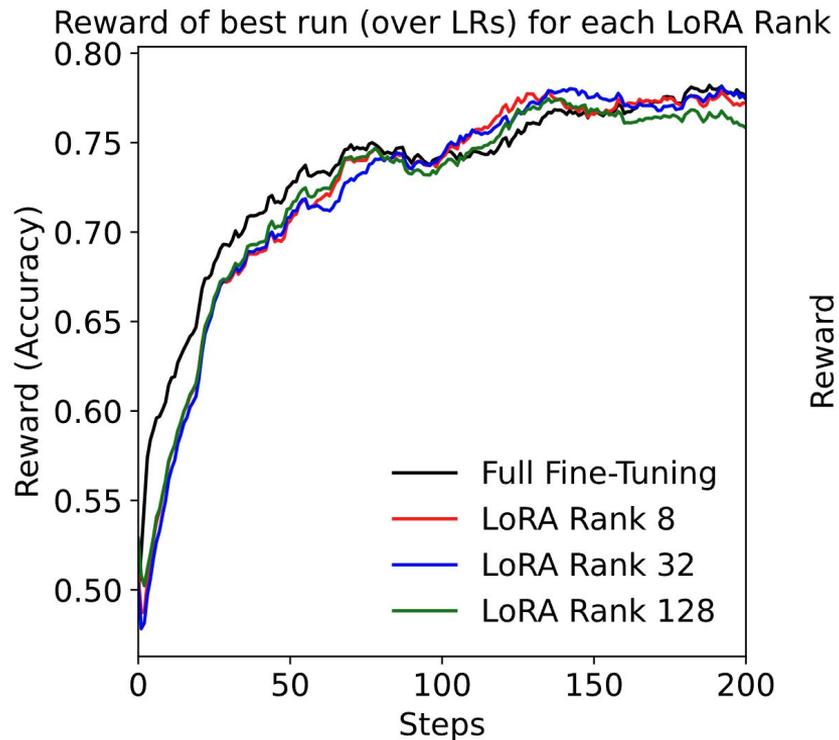
- LoRA can match full fine-tuning – both in sample efficiency and final performance – so long as two conditions hold:
 - You apply LoRA to all the weight matrices (especially MLP / MoE layers, not just attention).
 - The adapter has enough capacity relative to the amount of information to learn

If the adapter rank is too small, LoRA becomes bottlenecked and cannot capture all the necessary updates (GPT-style models have a capacity of approximately 3.6 bits per parameter)

Attention-only LoRA significantly underperforms MLP-only LoRA



LoRA vs. Full Fine-tuning



<https://thinkingmachines.ai/blog/lora/>

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

Published as a conference paper at COLM 2024

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

Limitations of parameter-efficient tuning methods

Thank you!