Test Time Scaling II

We change the model this time.

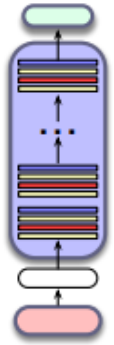
Lecture 10

Agenda

- Fine-tuning
  - Fine-tuning for decoder-only LMs
  - Parameter-efficient fine-tuning (PEFT)
  - Instruction tuning

“Fine-tuning”

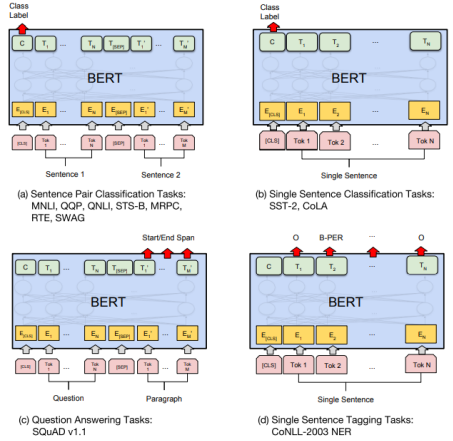
- BERT: the pretrain-finetune paradigm.



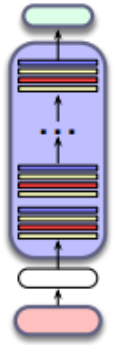
Supervised data from task

Task specific loss

Train a new “bolt-on” classifier



- Fine-tuning as continued pre-training



Data from fine-tuned domain

Next word prediction objective

Train the whole model on a new domain

LLM Generation

Search CTRL K

Web search

Code interpreter

File search and retrieval

More tools

Run and scale

Conversation state

Background mode

Streaming

Webhooks

File inputs

Prompting

Reasoning

Evaluation

Getting started

Working with evals

Supervised fine-tuning

Copy page

Fine-tune models with example inputs and known good outputs for better results and efficiency.

Supervised fine-tuning (SFT) lets you train an OpenAI model with examples for your specific use case. The result is a customized model that more reliably produces your desired style and content.

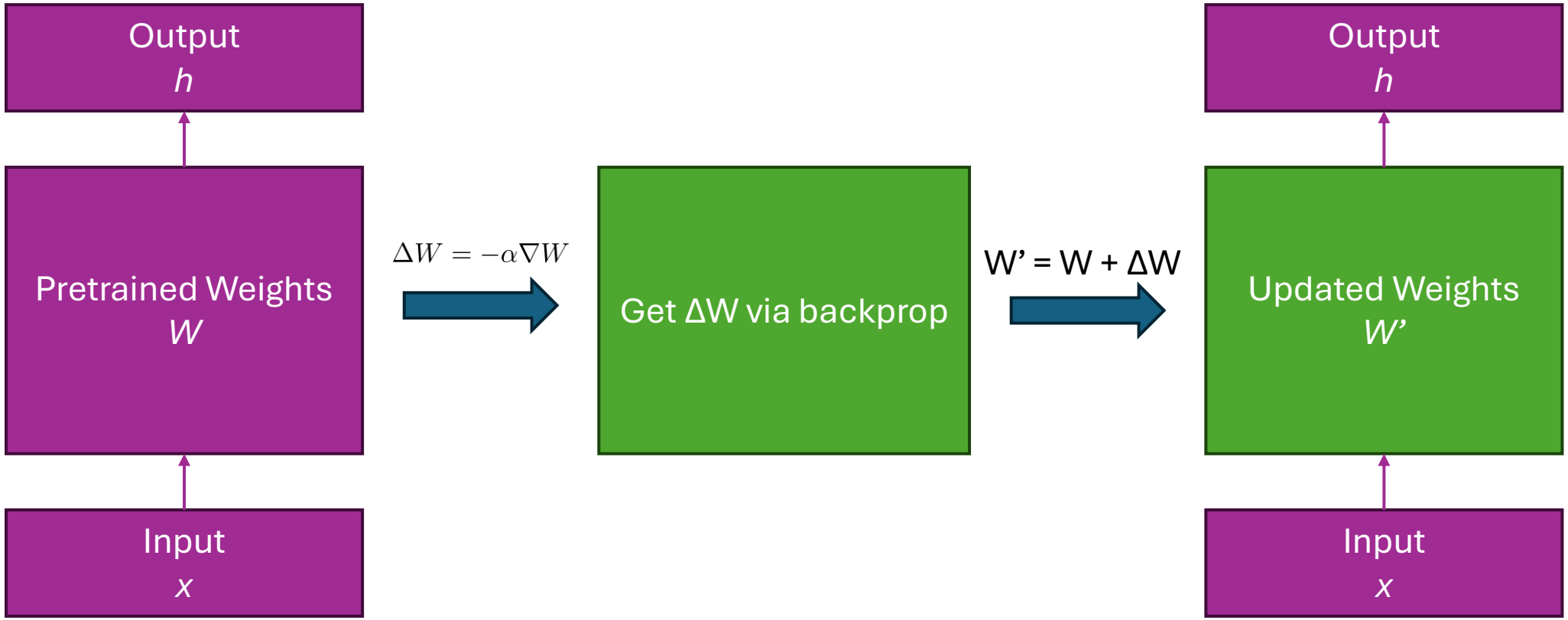
HOW IT WORKS	BEST FOR	USE WITH

Provide examples of correct responses to prompts to guide the model's behavior.
 	
Often uses human-generated "ground truth" responses to show the model how it should respond.
	• Classification
	• Nuanced translation
	• Generating content in a specific format
	• Correcting instruction-following failures
		`gpt-4.1-2025-04-14`
		`gpt-4.1-mini-2025-04-14`
		`gpt-4.1-nano-2025-04-14`

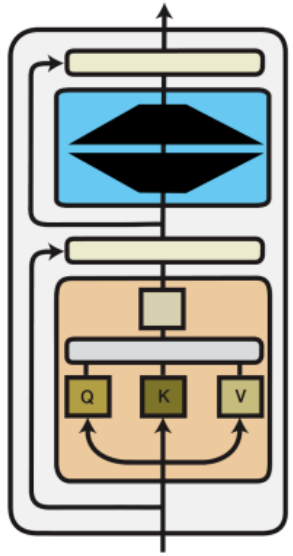
https://platform.openai.com/docs/guides/supervised-fine-tuning

Overview

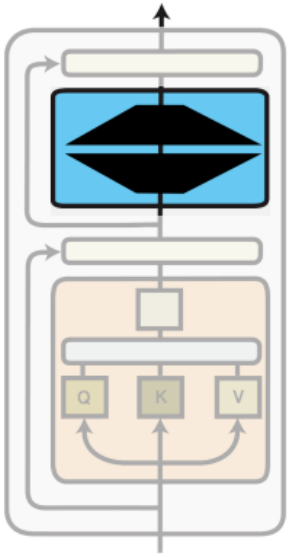
Fine-tuning


Output
h
Pretrained Weights
W
Input
x
$\Delta W = -\alpha \nabla W$
Get ΔW via backprop
W' = W + ΔW
Output
h
Updated Weights
W'
Input
x

Towards Parameter-efficient fine-tuning (PEFT)



Full Fine-tuning
Update **all model parameters**

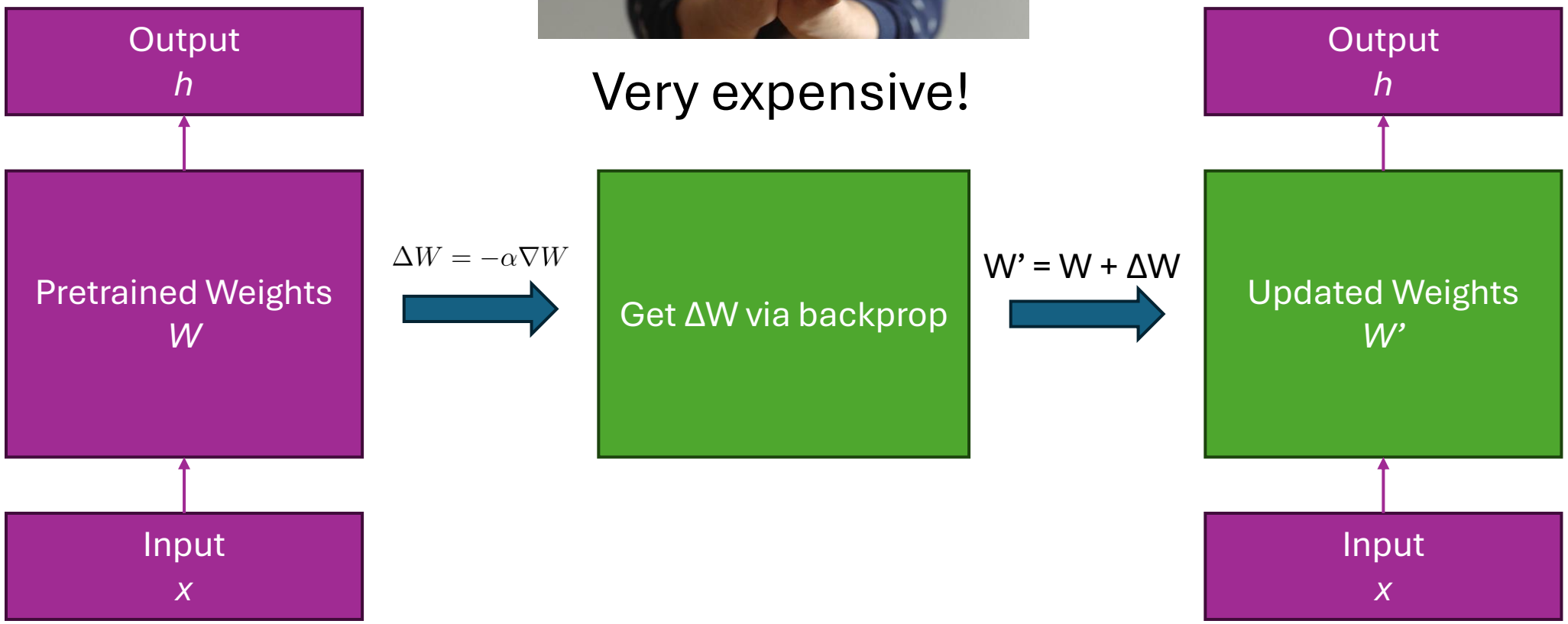


Parameter-efficient Fine-tuning
Update a **small subset** of model parameters

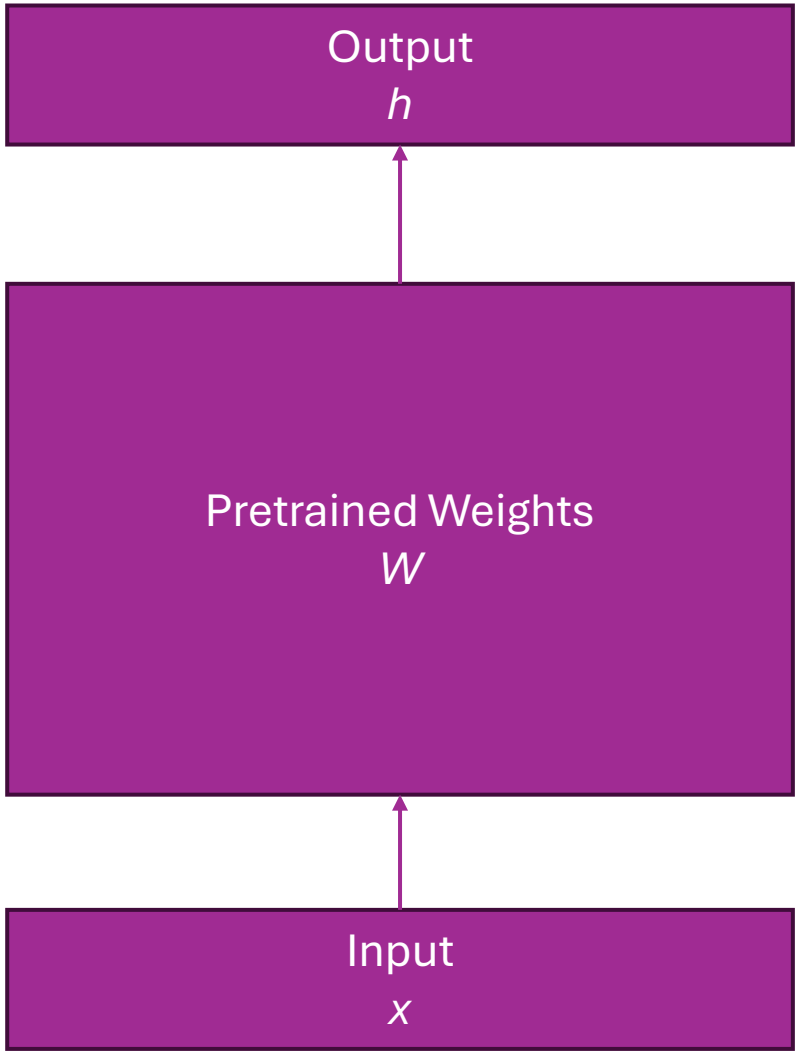
- Why fine-tuning only some parameters?
  - Fine-tuning all parameters is impractical with large models.
  - State-of-the-art models are massively overparameterized → Parameter-efficient finetuning matches performance of full fine-tuning.

Fine-tuning

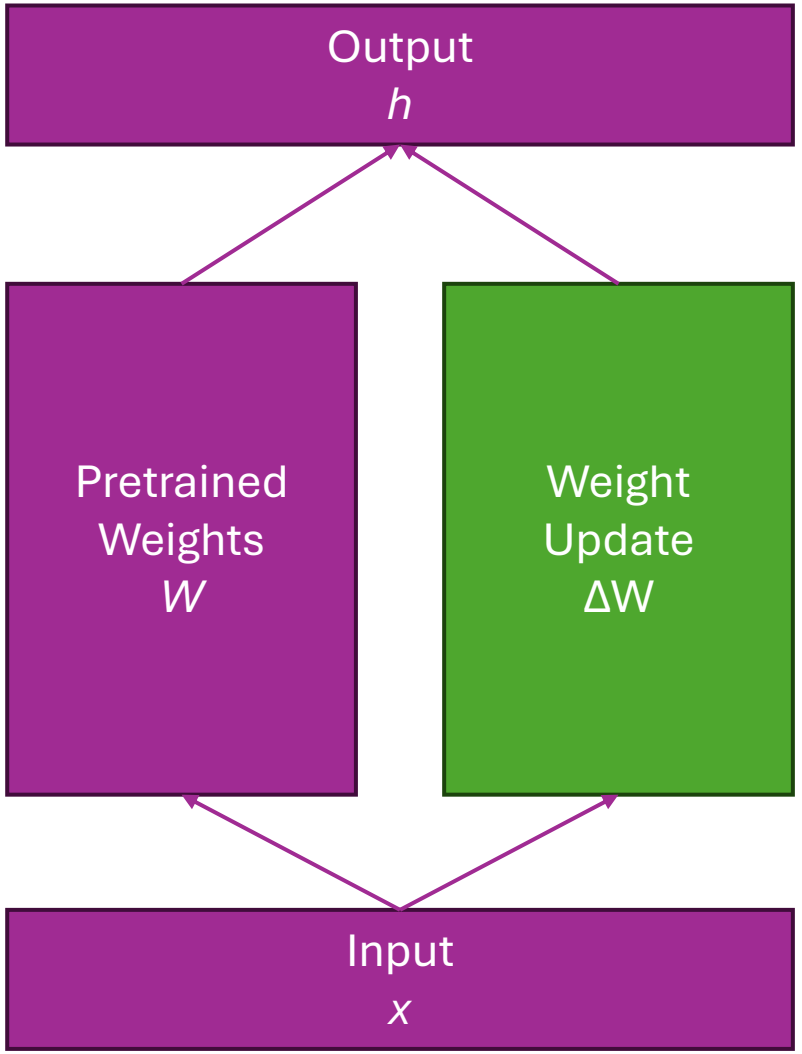
Very expensive!



PEFT



PEFT



$$h = xW + x\Delta W$$

Rank of a Matrix

- We can use a big matrix to represent low dimension data.
- The true dimension of a matrix is its *rank*.
- If a matrix *M* has rank *r*, we can decompose it into the product of two, much smaller matrices.

If

$$M \in \mathbb{R}^{m \times n}, \ \text{rank}(M) = r$$

Then there exist matrices:

$$U \in \mathbb{R}^{m \times r}, \ \text{and } U \in \mathbb{R}^{r \times n}$$

such that

$$M = UV$$

LoRA: Low-Rank Adaptation

- LLM weights are low rank!

INTRINSIC DIMENSIONALITY EXPLAINS THE EFFECTIVENESS OF LANGUAGE MODEL FINE-TUNING

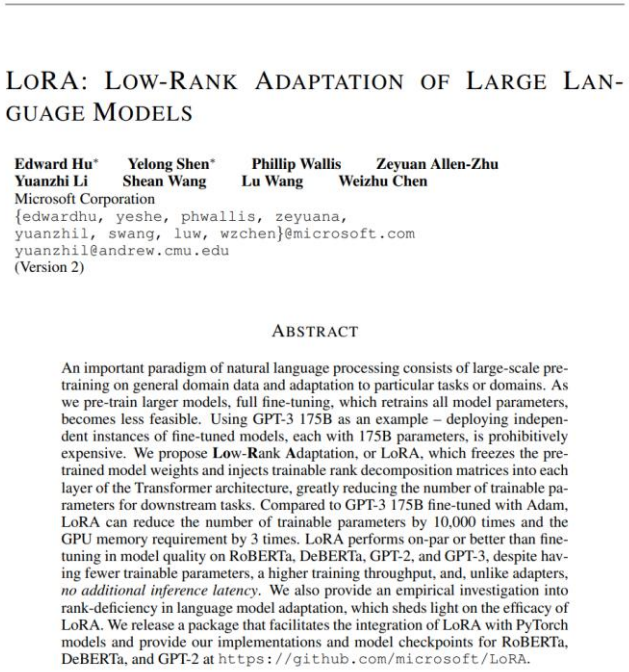
Armen Aghajanyan, Luke Zettlemoyer, Sonal Gupta
Facebook
{armenag,lsz,sonalgupta}@fb.com

ABSTRACT

Although pretrained language models can be fine-tuned to produce state-of-the-art results for a very wide range of language understanding tasks, the dynamics of this process are not well understood, especially in the low data regime. Why can we use relatively vanilla gradient descent algorithms (e.g., without strong regularization) to tune a model with hundreds of millions of parameters on datasets with only hundreds or thousands of labeled examples? In this paper, we argue that analyzing fine-tuning through the lens of intrinsic dimension provides us with empirical and theoretical intuitions to explain this remarkable phenomenon. We empirically show that common pre-trained models have a very low intrinsic dimension; in other words, there exists a low dimension reparameterization that is as effective for fine-tuning as the full parameter space. For example, by optimizing only 200 trainable parameters randomly projected back into the full space, we can tune a RoBERTa model to achieve 90% of the full parameter performance levels on MRPC. Furthermore, we empirically show that pre-training implicitly minimizes intrinsic dimension and, perhaps surprisingly, larger models tend to have lower intrinsic dimension after a fixed number of pre-training updates, at least in part explaining their extreme effectiveness. Lastly, we connect intrinsic dimensionality with low dimensional task representations and compression based generalization bounds to provide intrinsic-dimension-based generalization bounds that are independent of the full parameter count.

LoRA: Low-Rank Adaptation

- LLM weights are low rank!
- LLM weight updates are also low rank!

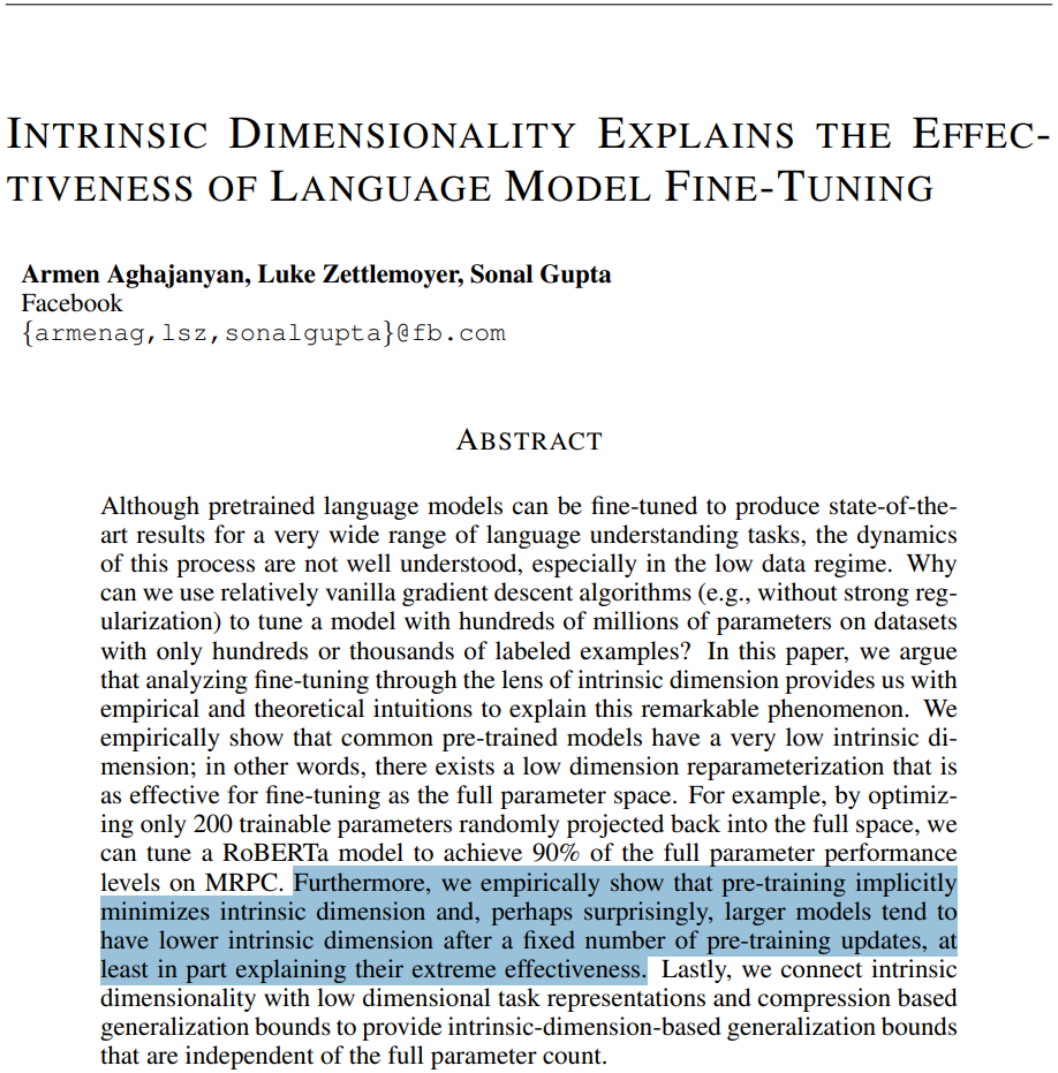


LoRA: Low-Rank Adaptation of Large Language Models

Edward Hu* **Yelong Shen*** **Phillip Wallis** **Zeyuan Allen-Zhu**
Yuanzhi Li **Shean Wang** **Lu Wang** **Weizhu Chen**
Microsoft Corporation
{edwardhu, yeshe, phwallis, zeyuana,
yuanzhil, swang, luw, wzchen}@microsoft.com
yuanzhil@andrew.cmu.edu
(Version 2)

Abstract

An important paradigm of natural language processing consists of large-scale pre-training on general domain data and adaptation to particular tasks or domains. As we pre-train larger models, full fine-tuning, which retrains all model parameters, becomes less feasible. Using GPT-3 175B as an example – deploying independent instances of fine-tuned models, each with 175B parameters, is prohibitively expensive. We propose **Lo**w-**R**ank **A**daptation, or LoRA, which freezes the pre-trained model weights and injects trainable rank decomposition matrices into each layer of the Transformer architecture, greatly reducing the number of trainable parameters for downstream tasks. Compared to GPT-3 175B fine-tuned with Adam, LoRA can reduce the number of trainable parameters by 10,000 times and the GPU memory requirement by 3 times. LoRA performs on-par or better than fine-tuning in model quality on RoBERTa, DeBERTa, GPT-2, and GPT-3, despite having fewer trainable parameters, a higher training throughput, and, unlike adapters, *no additional inference latency*. We also provide an empirical investigation into rank-deficiency in language model adaptation, which sheds light on the efficacy of LoRA. We release a package that facilitates the integration of LoRA with PyTorch models and provide our implementations and model checkpoints for RoBERTa, DeBERTa, and GPT-2 at https://github.com/microsoft/LoRA.



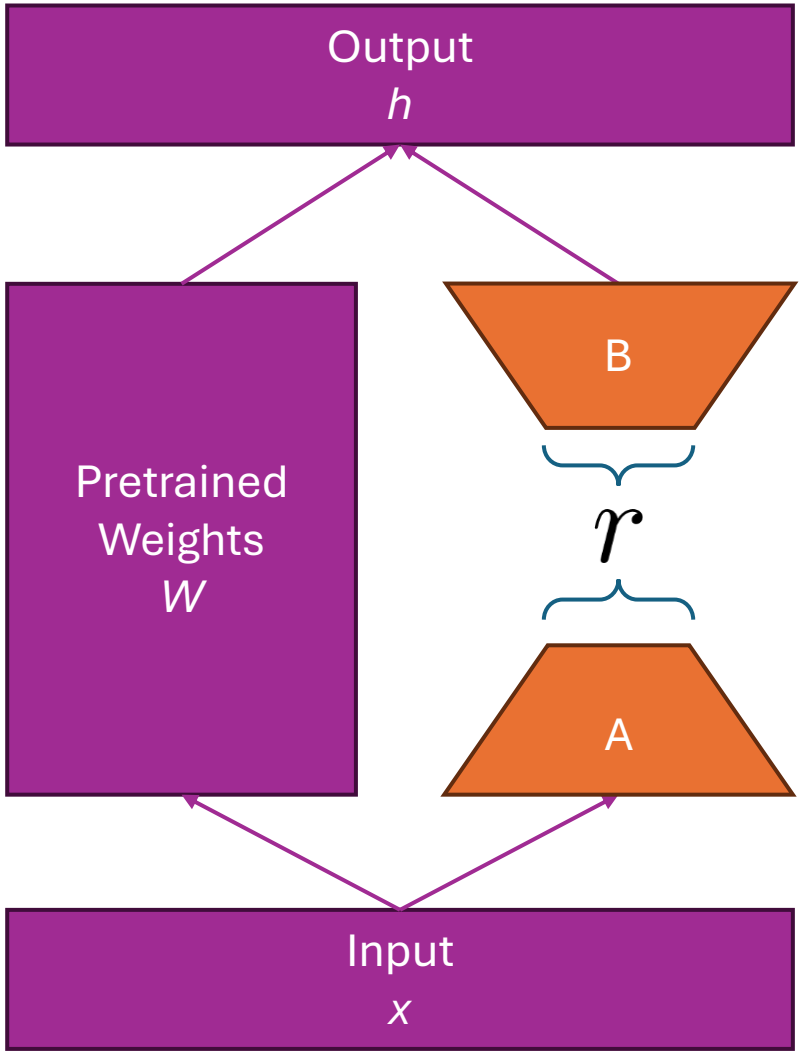
Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning

Armen Aghajanyan, Luke Zettlemoyer, Sonal Gupta
Facebook
{armenag,lsz,sonalgupta}@fb.com

Abstract

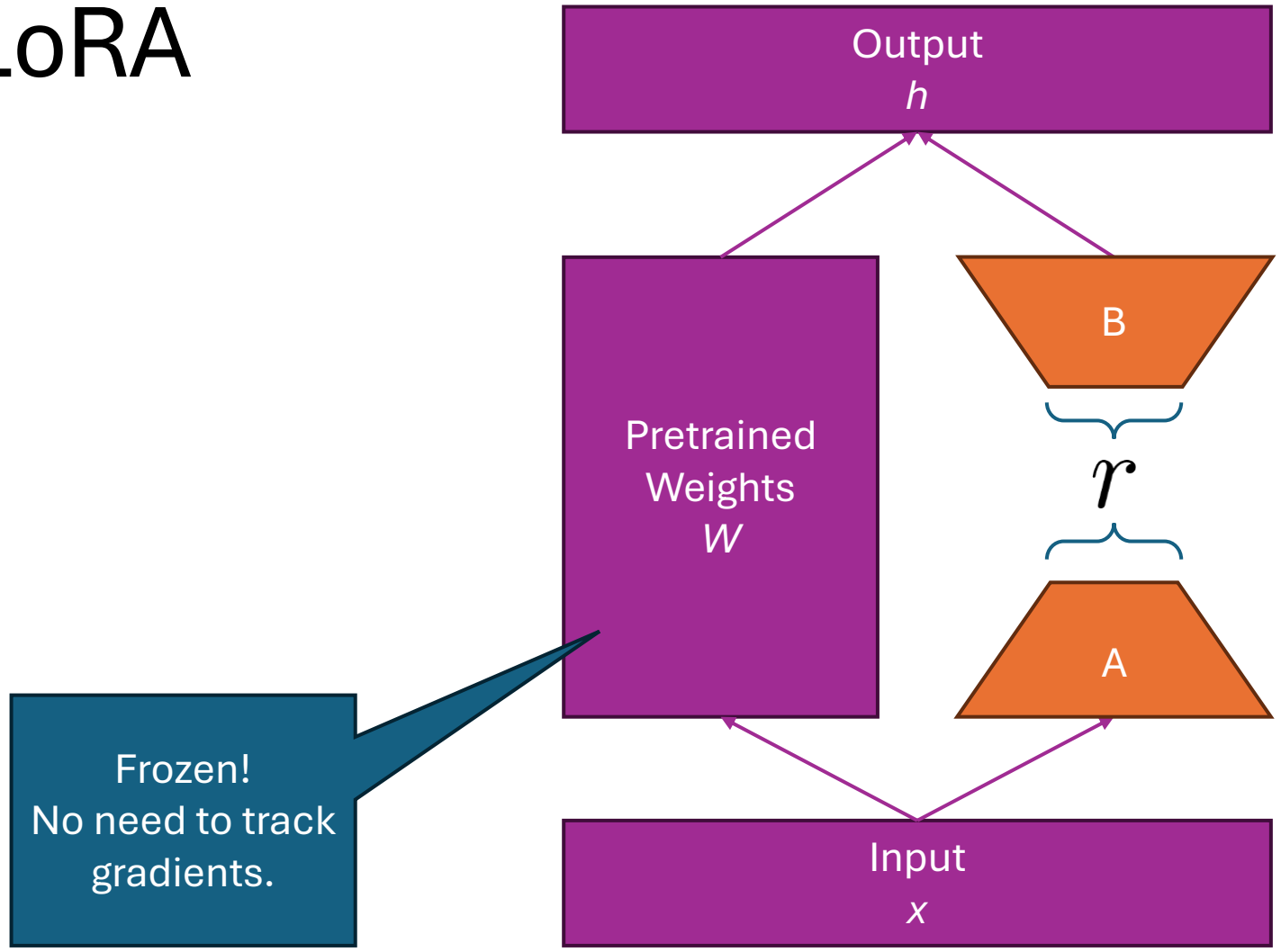
Although pretrained language models can be fine-tuned to produce state-of-the-art results for a very wide range of language understanding tasks, the dynamics of this process are not well understood, especially in the low data regime. Why can we use relatively vanilla gradient descent algorithms (e.g., without strong regularization) to tune a model with hundreds of millions of parameters on datasets with only hundreds or thousands of labeled examples? In this paper, we argue that analyzing fine-tuning through the lens of intrinsic dimension provides us with empirical and theoretical intuitions to explain this remarkable phenomenon. We empirically show that common pre-trained models have a very low intrinsic dimension; in other words, there exists a low dimension reparameterization that is as effective for fine-tuning as the full parameter space. For example, by optimizing only 200 trainable parameters randomly projected back into the full space, we can tune a RoBERTa model to achieve 90% of the full parameter performance levels on MRPC. Furthermore, we empirically show that pre-training implicitly minimizes intrinsic dimension and, perhaps surprisingly, larger models tend to have lower intrinsic dimension after a fixed number of pre-training updates, at least in part explaining their extreme effectiveness. Lastly, we connect intrinsic dimensionality with low dimensional task representations and compression based generalization bounds to provide intrinsic-dimension-based generalization bounds that are independent of the full parameter count.

LoRA



$$h = xW + xAB$$

LoRA


Output
h
Pretrained
Weights
W
B
r
A
Input
x
Frozen!
No need to track
gradients.

$$h = xW + xAB$$

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

Table 4: Performance of different adaptation methods on GPT-3 175B. We report the logical form validation accuracy on WikiSQL, validation accuracy on MultiNLI-matched, and Rouge-1/2/L on SAMSum. LoRA performs better than prior approaches, including full fine-tuning. The results on WikiSQL have a fluctuation around $\pm 0.5\%$, MNLI-m around $\pm 0.1\%$, and SAMSum around $\pm 0.2/\pm 0.2/\pm 0.1$ for the three metrics.

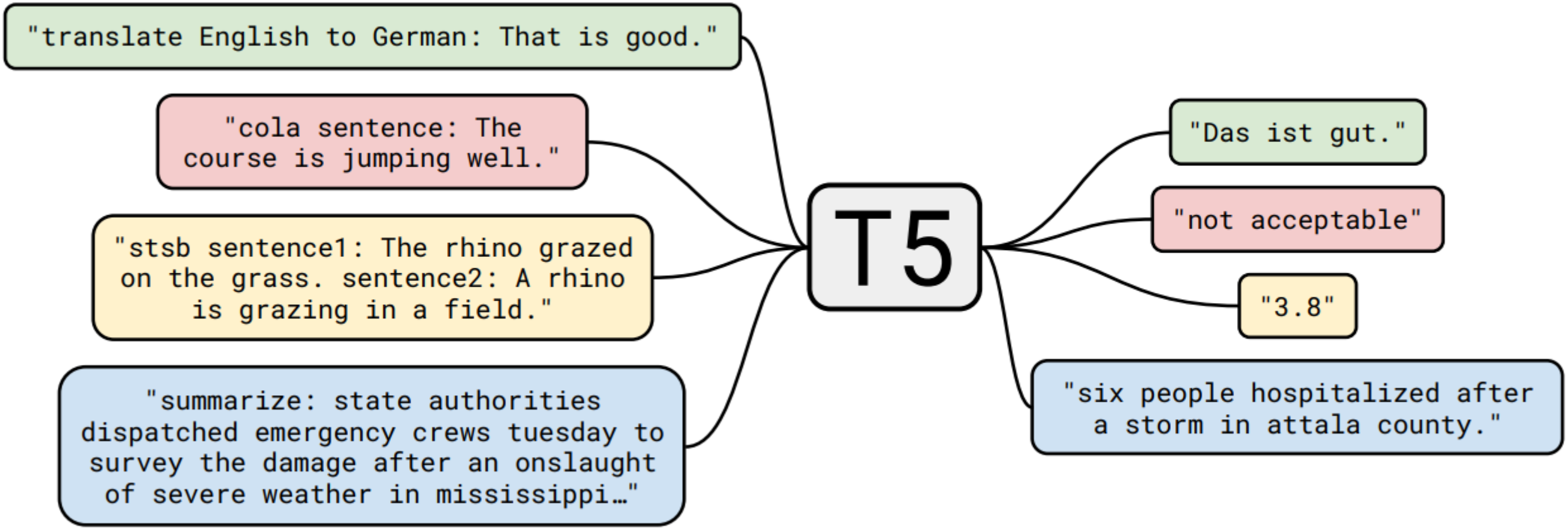
7.2 What is the Optimal Rank r for LoRA?

We turn our attention to the effect of rank r on model performance. We adapt $\{W_q, W_v\}$, $\{W_q, W_k, W_v, W_c\}$, and just W_q for a comparison.

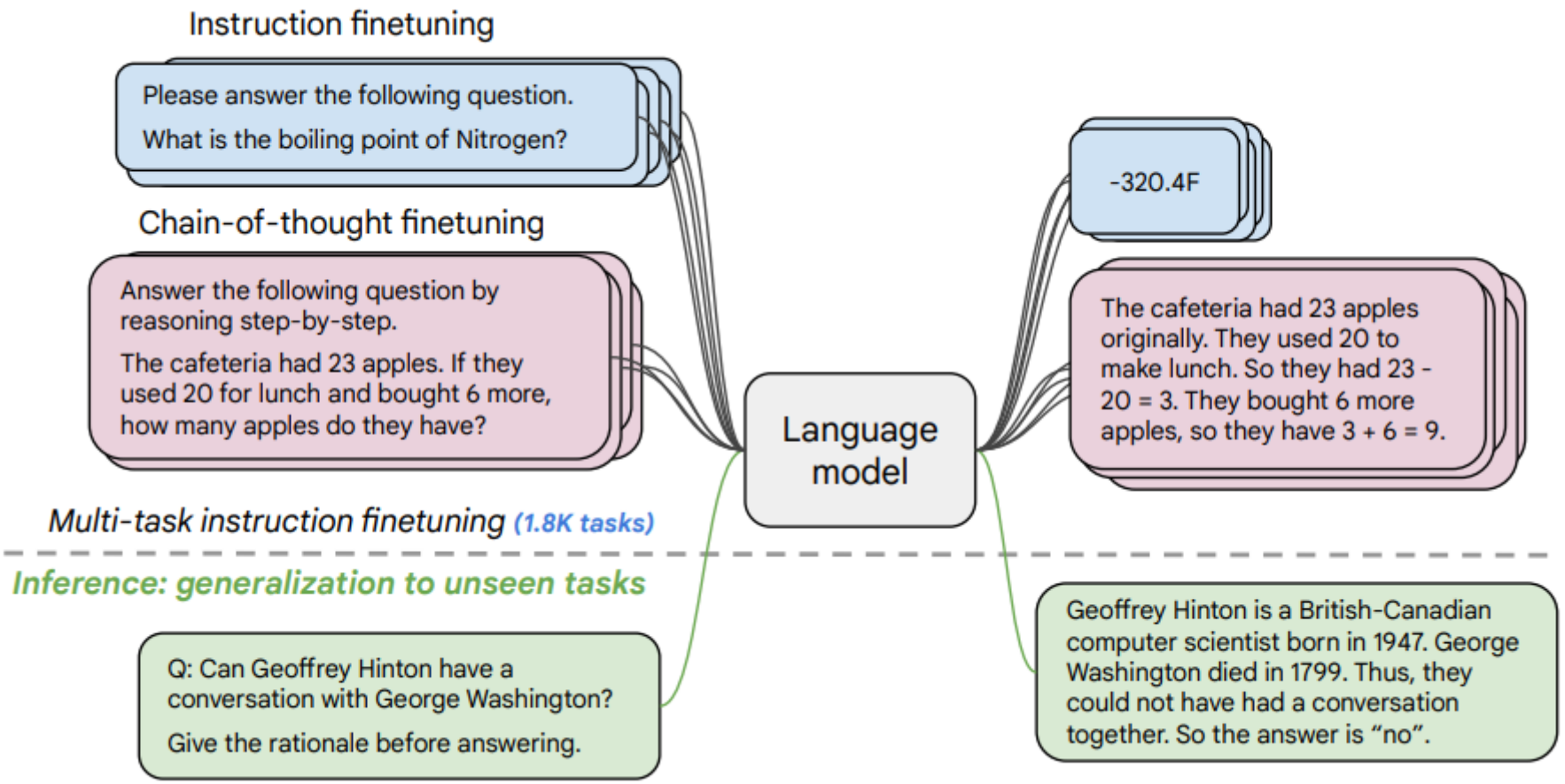
	Weight Type	$r = 1$	$r = 2$	$r = 4$	$r = 8$	$r = 64$
WikiSQL($\pm 0.5\%$)	W_q	68.8	69.6	70.5	70.4	70.0
	W_q, W_v	73.4	73.3	73.7	73.8	73.5
	W_q, W_k, W_v, W_o	74.1	73.7	74.0	74.0	73.9
MultiNLI ($\pm 0.1\%$)	W_q	90.7	90.9	91.1	90.7	90.7
	W_q, W_v	91.3	91.4	91.3	91.6	91.4
	W_q, W_k, W_v, W_o	91.2	91.7	91.7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

Table 6: Validation accuracy on WikiSQL and MultiNLI with different rank r. To our surprise, a rank as small as one suffices for adapting both W_q and W_v on these datasets while training W_q alone needs a larger r. We conduct a similar experiment on GPT-2 in Section H.2.

Instruction Tuning



Instruction Tuning



Chung et al. 2022. Scaling Instruction-Finetuned Language Models.

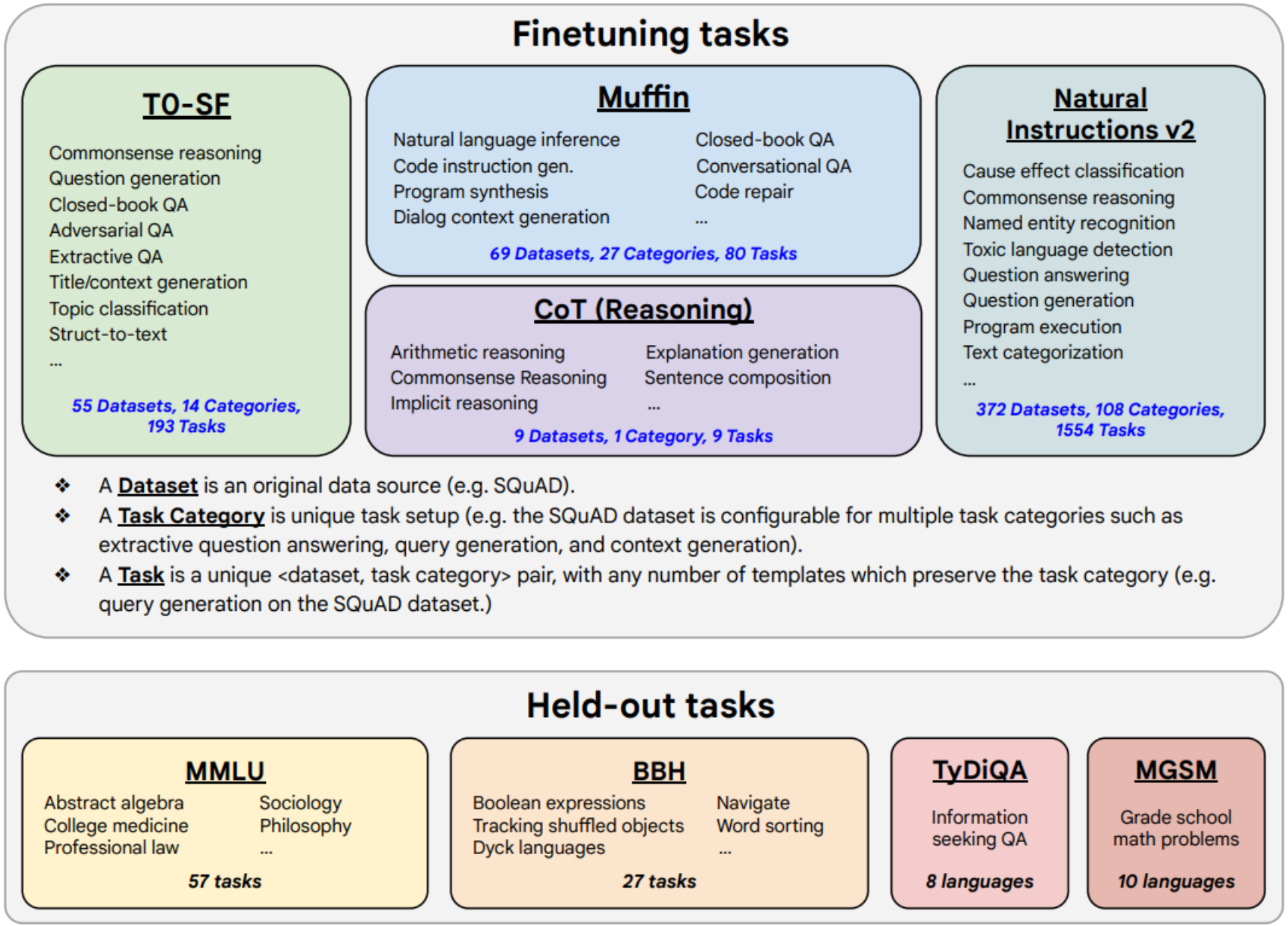


Figure 2: Our finetuning data comprises 473 datasets, 146 task categories, and 1,836 total tasks. Details for the tasks used in this paper is given in Appendix F.

Params	Model	Norm. avg.
80M	T5-Small	-9.2
	Flan-T5-Small	-3.1 (+6.1)
250M	T5-Base	-5.1
	Flan-T5-Base	6.5 (+11.6)
780M	T5-Large	-5.0
	Flan-T5-Large	13.8 (+18.8)
3B	T5-XL	-4.1
	Flan-T5-XL	19.1 (+23.2)
11B	T5-XXL	-2.9
	Flan-T5-XXL	23.7 (+26.6)
8B	PaLM	6.4
	Flan-PaLM	21.9 (+15.5)
62B	PaLM	28.4
	Flan-PaLM	38.8 (+10.4)
540B	PaLM	49.1
	Flan-PaLM	58.4 (+9.3)
62B	cont-PaLM	38.1
	Flan-cont-PaLM	46.7 (+8.6)
540B	U-PaLM	50.2
	Flan-U-PaLM	59.1 (+8.9)

Model input (Disambiguation QA)

Q: In the following sentences, explain the antecedent of the pronoun (which thing the pronoun refers to), or state that it is ambiguous.

Sentence: The reporter and the chef will discuss their favorite dishes.

Options:
(A) They will discuss the reporter's favorite dishes
(B) They will discuss the chef's favorite dishes
(C) Ambiguous

A: Let's think step by step.

Before instruction finetuning

The reporter and the chef will discuss their favorite dishes.
The reporter and the chef will discuss the reporter's favorite dishes.
The reporter and the chef will discuss the chef's favorite dishes.
The reporter and the chef will discuss the reporter's and the chef's favorite dishes.

✖ **(doesn’t answer question)**

Model input (Disambiguation QA)

Q: In the following sentences, explain the antecedent of the pronoun (which thing the pronoun refers to), or state that it is ambiguous.

Sentence: The reporter and the chef will discuss their favorite dishes.

Options:
(A) They will discuss the reporter's favorite dishes
(B) They will discuss the chef's favorite dishes
(C) Ambiguous

A: Let's think step by step.

After instruction finetuning

The reporter and the chef will discuss their favorite dishes does not indicate whose favorite dishes they will discuss. So, the answer is (C). ✅


"See You Next Year" isn't a funny joke

See you next year. Have fun and stay safe.