# Parameter-Efficient Fine-Tuning

## CSE 493G, Winter 2026

Tanush Yadav

# The OCR Task

**Backpropagation Applied to Handwritten Zip Code Recognition**

**Y. LeCun**
**B. Boser**
**J. S. Denker**
**D. Henderson**
**R. E. Howard**
**W. Hubbard**
**L. D. Jackel**
*AT&T Bell Laboratories Holmdel, NJ 07733 USA*

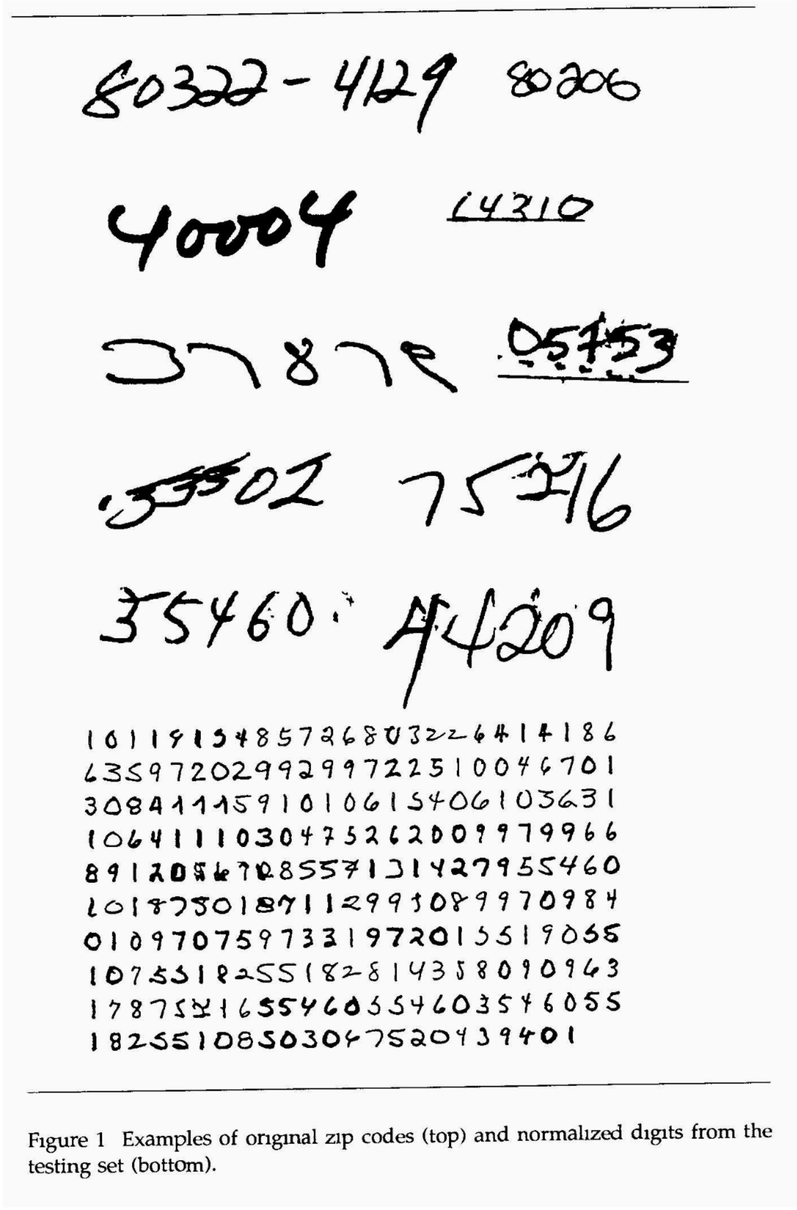
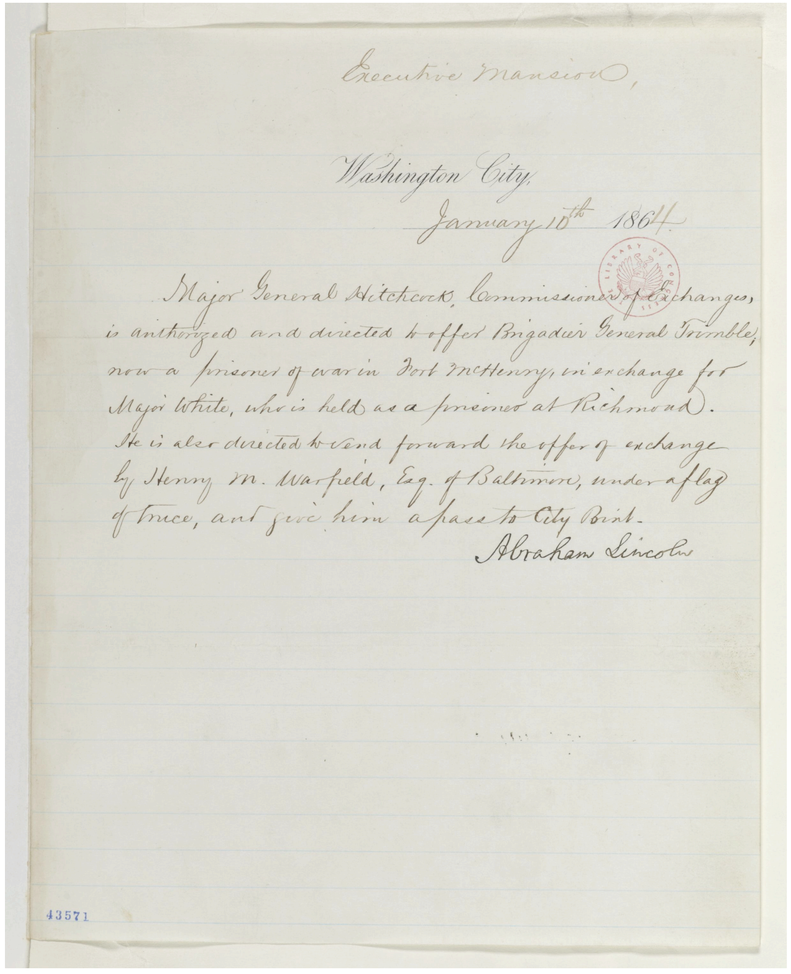


Figure 1 Examples of original zip codes (top) and normalized digits from the testing set (bottom).

LeCun et al.

# The OCR Task



Executive Mansion,

Washington City,
January 15th 1864.

Major General Hitchcock, Commissioner of Exchanges, is authorized and directed to offer Brigadier General Trimble, now a prisoner of war in Fort McHenry, in exchange for Major White, who is held as a prisoner at Richmond. He is also directed to send forward the offer of exchange by Henry M. Warfield, Esq. of Baltimore, under a flag of truce, and give him a pass to City Point.

Abraham Lincoln

43571

```
Executive Mansion,

Washington City,

January 15th, 1864

Major General Hitchcock, Commissioner
of Exchanges, is authorized and
directed to offer Brigadier General
Trimble, now a prisoner of war in Fort
McHenry, in exchange for Major White,
who is held as a prisoner at Richmond.
He is also directed to send forward the
offer of exchange by Henry M. Warfield,
Esq. of Baltimore, under a flag of
truce, and give him a pass to City
Point.

Abraham Lincoln
```

OlmoOCR

# The OCR Task

## An expensive undertaking

- $734k grant to TAMU to OCR 300k documents digitized from microfilm
  - poor image quality (imaged in 70s, microfilm in 80s, digitized in 90s)
  - poor print quality (premodern printing techniques)
  - damaged documents (from 1746 to 1800)

- €11.5 million project by the EU to improve OCR tools

Christy et al. ; UKOLN

# The OCR Task

## A difficult undertaking

"All work currently being performed in the field of cultural analytics on documents published before 1800 … is producing conclusions that are only about 50% reliable."

**OCR results "are poor or even useless"**

Christy et al.

# The OCR Task

## Traditional approaches

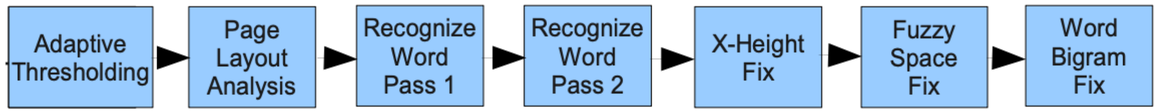



Figure 1. Block diagram of the overall architecture of Tesseract.

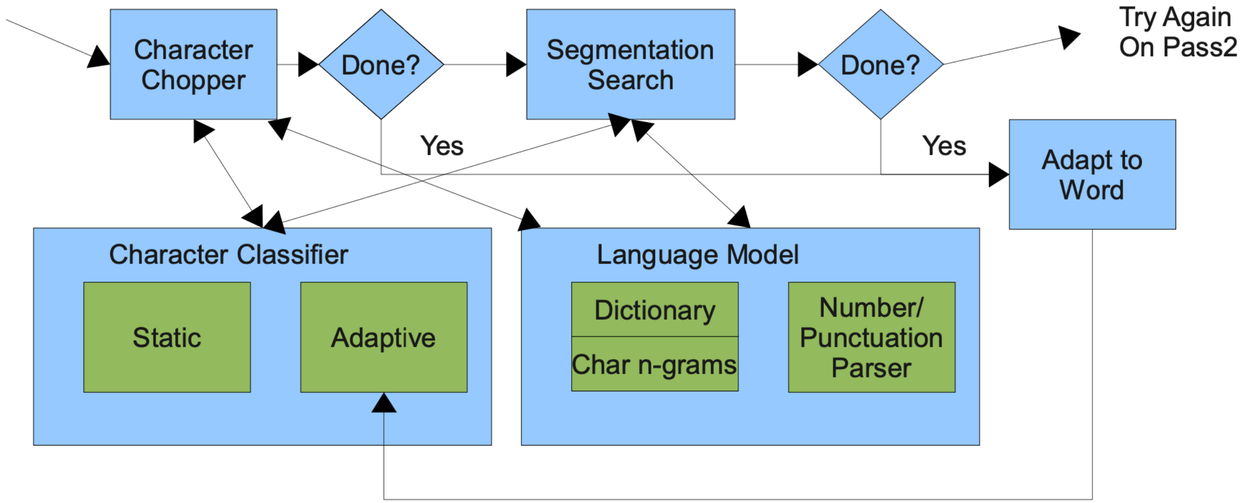



Figure 2. Block diagram of the Tesseract word recognizer.

# The OCR Task

## Modern ML pipeline approaches

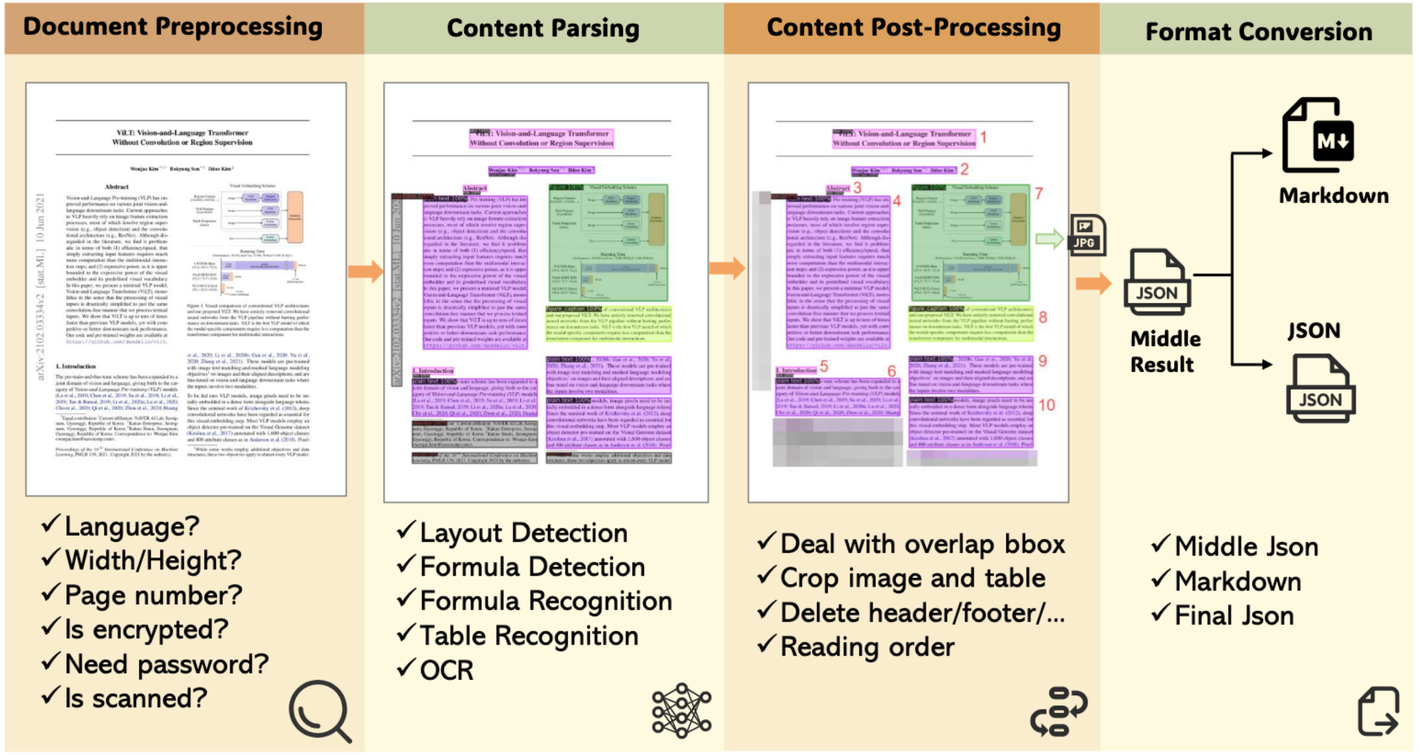



Figure 1: Overview of the MinerU framework processing workflow.

# The OCR Task

## Modern ML pipeline approaches

| | Performance (%) | Cost ($ / mil pages) |
|---|---|---|
| **MinerU** | 61.5 | 596 |

# The OCR Task

## Off-the-shelf VLMs

| | Performance (%) | Cost ($ / mil pages) |
|---|---|---|
| **MinerU** | 61.5 | 596 |
| **Qwen2-VL** | 31.5 | 176 |
| **GPT-4o** | 68.9 | 6240 |

# The OCR Task

## Off-the-shelf solutions force tradeoffs

| | Performance (%) | Cost ($ / mil pages) |
|---|---|---|
| **MinerU** | 61.5 | 596 |
| **Qwen2-VL** | 31.5 | 176 |
| **GPT-4o** | 68.9 | 6240 |
| **Marker** | 70.1 | 1484 |

# The OCR Task

## Off-the-shelf solutions force tradeoffs

| | Performance (%) | Cost ($ / mil pages) |
|---|---|---|
| **MinerU** | 61.5 | 596 |
| **Qwen2-VL** | 31.5 | 176 |
| **GPT-4o** | 68.9 | 6240 |
| **Marker** | 70.1 | 1484 |

**gold standard**

# The OCR Task

## Off-the-shelf solutions force tradeoffs

| | Performance (%) | Cost ($ / mil pages) |
|---|---|---|
| **MinerU** | 61.5 | 596 |
| **Qwen2-VL** | 31.5 | 176 |
| **GPT-4o** | 68.9 | 6240 |
| **Marker** | 70.1 | 1484 |

**small, open-weight model**

**gold standard**

# How could we improve OCR in Qwen2-VL?

***fine-tune* the model on OCR data!**

# Fine-tuning for OCR

## Continue training Qwen2-VL, only on OCR data

| Source | Unique docs | Total pages |
|---|---|---|
| Web crawled PDFs | 96,929 | 240,940 |
| Internet Archive books | 5,896 | 17,701 |
| *Total* | *102,825* | *258,641* |

**Table 1** `olmOCR-mix-0225` composition by source.

| Document type | Fraction |
|---|---|
| Academic | 55.9% |
| Brochure | 11.2% |
| Legal | 10.2% |
| Books | 6.8% |
| Table | 5.6% |
| Diagram | 4.7% |
| Slideshow | 1.9% |
| Other | 3.7% |

**Table 2** `olmOCR-mix-0225` PDFs breakdown by document type.

# The OCR Task

## Off-the-shelf solutions force tradeoffs

| | Performance (%) | Cost ($ / mil pages) |
|---|---|---|
| **MinerU** | 61.5 | 596 |
| **Qwen2-VL** | 31.5 | 176 |
| **GPT-4o** | 68.9 | 6240 |
| **Marker** | 70.1 | 1484 |

**small, open-weight model**

**gold standard**

# The OCR Task

## Off-the-shelf solutions force tradeoffs

| | Performance (%) | Cost ($ / mil pages) |
|---|---|---|
| **MinerU** | 61.5 | 596 |
| **Qwen2-VL** | 31.5 | 176 |
| **olmOCR** | 74.7 | 176 |
| **GPT-4o** | 68.9 | 6240 |
| **Marker** | 70.1 | 1484 |

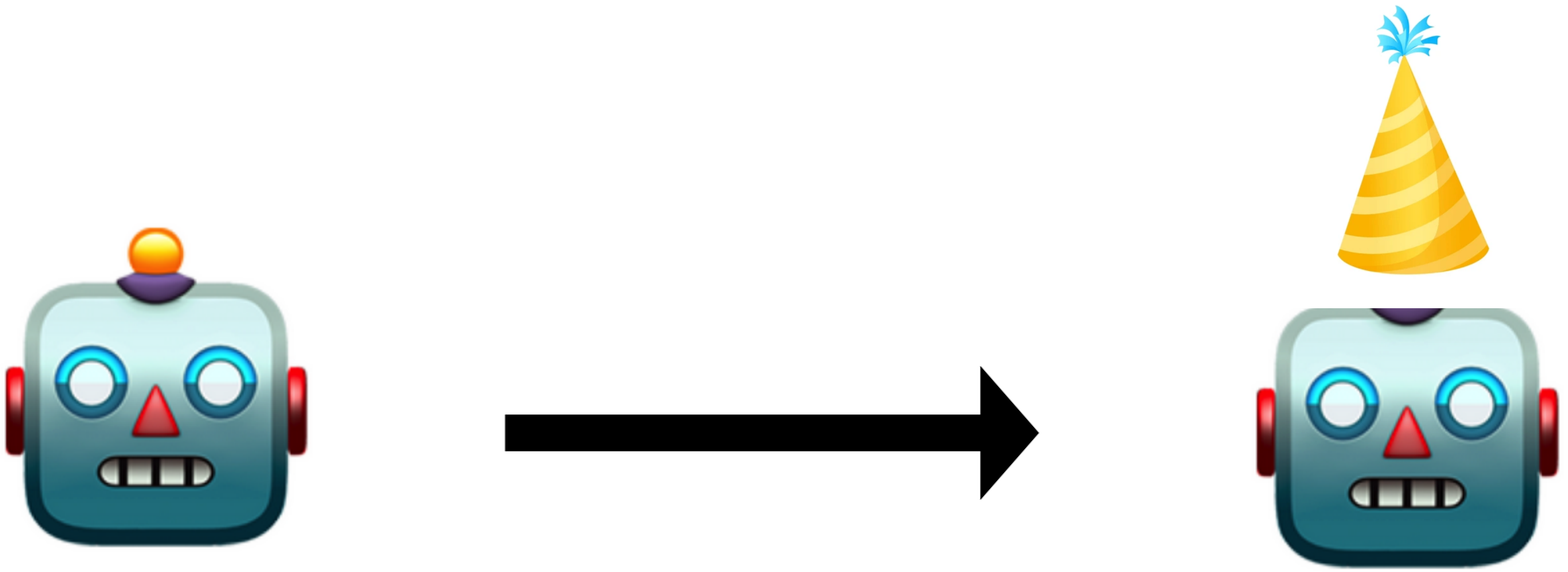
**small, open-weight model**

**small, open-weight model**

**(old) gold standard**

# What is fine-tuning?

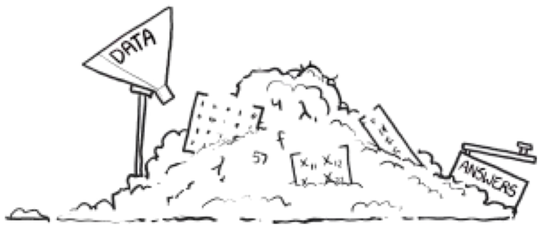
## Take a useful model that already knows a lot and update it slightly



Slide credit to Scott Geng

# Why fine-tune?

## Better, cheaper applications




all of the internet

Pretrain

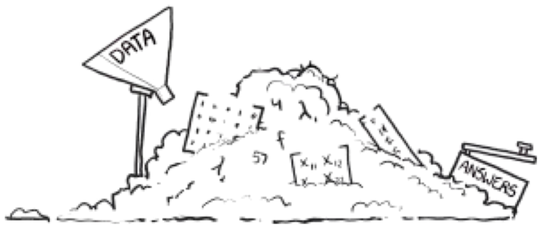
($100M)

Qwen
(base)

Slide credit to Scott Geng

# Why fine-tune?

## Better, cheaper applications


all of the internet

Pretrain

($100M)

Qwen
(base)

Finetune

($100k)

Qwen
Medical

Slide credit to Scott Geng

# Why fine-tune?

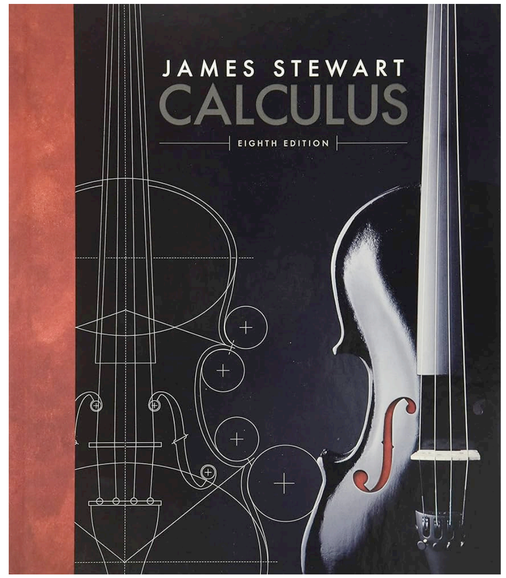
## Better, cheaper applications


all of the internet

Pretrain

($100M)

Qwen
(base)




Finetune

($100k)

Qwen
Math

# Why fine-tune?

## Better, cheaper applications


all of the internet

Pretrain

($100M)

Qwen
(base)

Finetune

($100k)

Qwen
Code

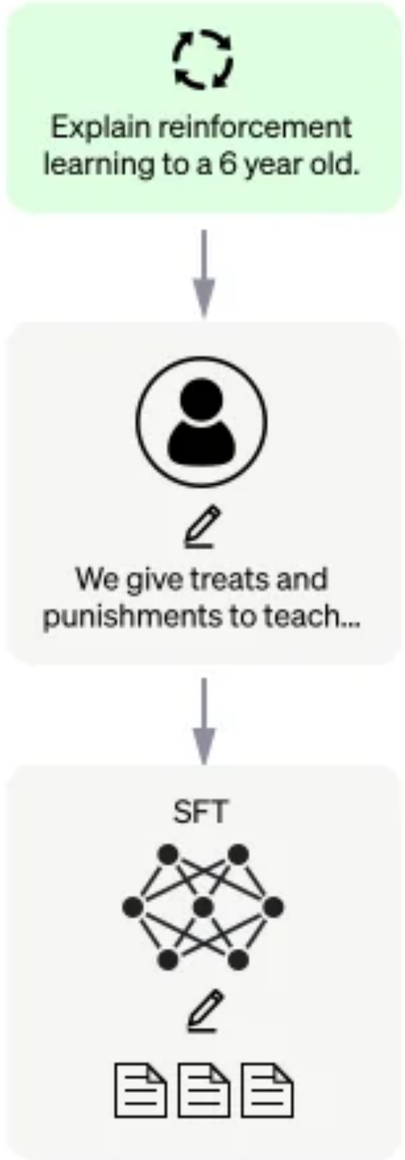
# Why fine-tune?

## Align with human preferences

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3.5 with supervised learning.




Slide credit to Scott Geng

OpenAI

# Why fine-tune?

## Democratize AI

Everyone should be able to easily adapt
a very capable (very big) base model
to whatever task they want

Slide credit to Scott Geng

# Fine-tuning allows specialized models for different tasks or users

**all originating from ONE base model**

# Why not fine-tune?

## It’s expensive!

- During training:
  - must load optimizer state for **entire** model —> high memory overhead
  - meaningfully improving a terabyte-size model requires many GPU hours
- During inference:
  - must load *different* **massive** models for each user
    - can’t cache models
    - takes minutes to load

# The fine-tuning dilemma

- Fine-tuning is often necessary (e.g., OCR, code)
- Fine-tuning is prohibitively expensive

**What should we do ?!?**

# The fine-tuning dilemma

- Fine-tuning is often necessary (e.g., OCR, code)
- Fine-tuning is prohibitively expensive

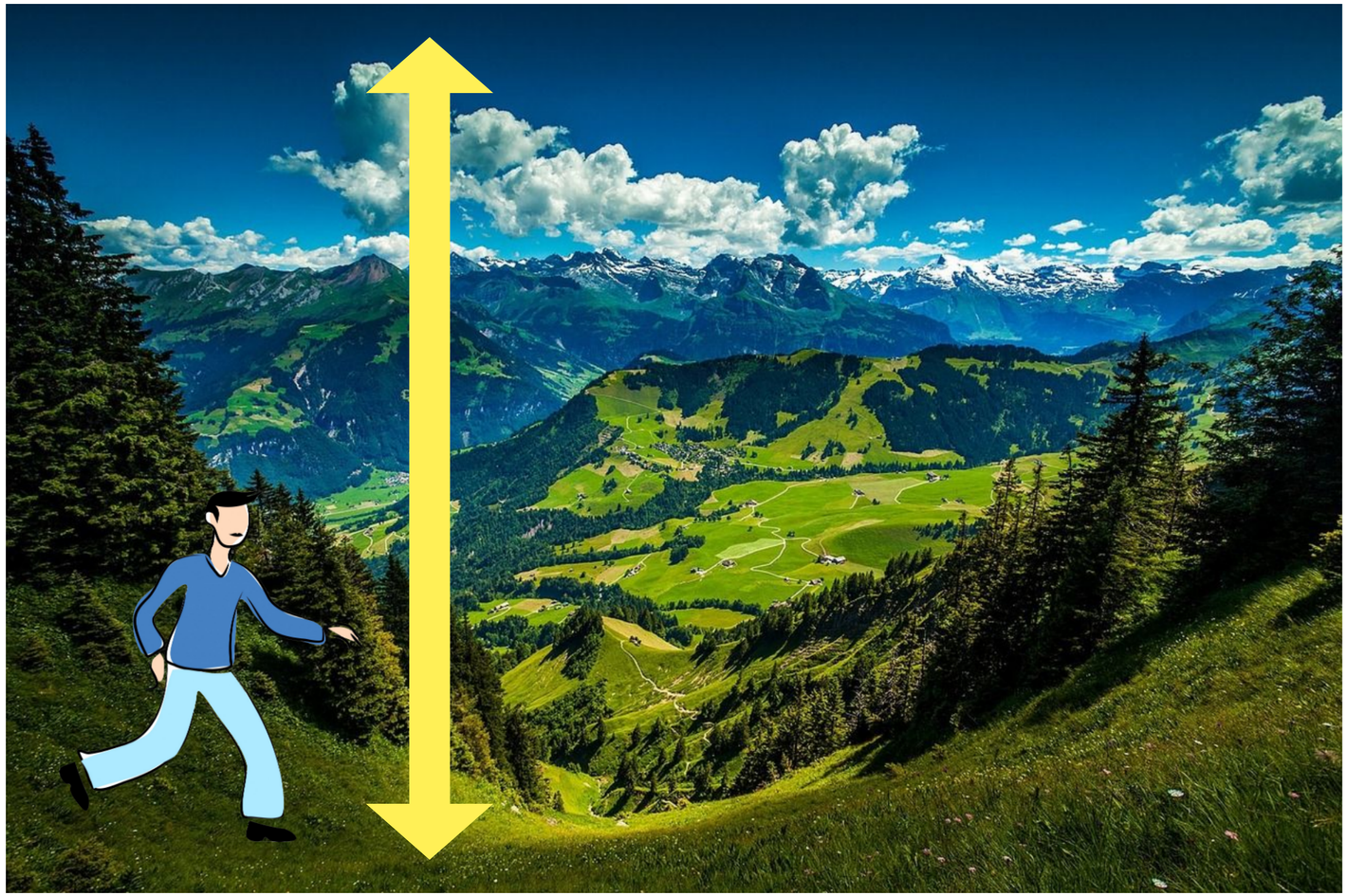
**What if we only “tuned” a very small portion of the parameters?**

# Intrinsic Dimension

## Consider our optimization landscape in $\mathbb{R}^2$

# Intrinsic Dimension

## Restrict ourselves to optimizing in a subspace $\mathbb{R}^1$



Li et al. 2018

# Intrinsic Dimension

## Optimize over a $d$-dimensional subspace of $\mathbb{R}^D$ for $d < D$

- The problem is in $\mathbb{R}^D$
- We will estimate the smallest $d$ such that the problem can be “solved”
- We will call $d$ the problem’s **intrinsic dimension**
- Formally, $d = D - s$ where $s$ is the dimension of the problem’s solution set

Li et al. 2018

# Intrinsic Dimension

## Toy Example

- Let $D = 1000$
- Loss function requires first 100 elements to sum to 1,
  second 100 elements to sum to 2, etc.
- Indeed, from any point with 0-loss we can move in 990 orthogonal directions
  while remaining at 0-loss —> solution set is a 990 dimensional hyperplane
- So $d = D - s = 1000 - 990 = 10$
- Indeed, one only needs to get 10 dimensions “right” to solve this problem

Li et al. 2018

# Intrinsic Dimension

## Estimating $d$ for neural nets

- Aforementioned examples are easy to work out algebraically
- For the mess that is neural networks, we will estimate $d$ by training models

$$\theta^{(D)} = \theta_0^{(D)} + P\theta^{(d)}$$

Li et al. 2018

# Intrinsic Dimension

## Estimating $d$ for neural nets

- Aforementioned examples are easy to work out algebraically
- For the mess that is neural networks, we will estimate $d$ by training models

randomly initialized params

trainable params

$$\theta^{(D)} = \theta_0^{(D)} + P\theta^{(d)}$$

final params

random projection matrix: $d \to D$

Li et al. 2018

# Intrinsic Dimension

## Estimating $d$ for neural nets

- Aforementioned examples are easy to work out algebraically
- For the mess that is neural networks, we will estimate $d$ by training models

$$\theta^{(D)} = \theta_0^{(D)} + P\theta^{(d)}$$

$\theta^{(D)} \in \mathbb{R}^D$, $\theta_0^{(D)} \in \mathbb{R}^D$, $P \in \mathbb{R}^{D \times d}$, $\theta^{(d)} \in \mathbb{R}^d$

Li et al. 2018

# Intrinsic Dimension

## Estimating $d$ for neural nets

- "Baseline solution" is a model allowed to optimize in $\mathbb{R}^D$ space
- Increase $d$ until performance nears the baseline solution

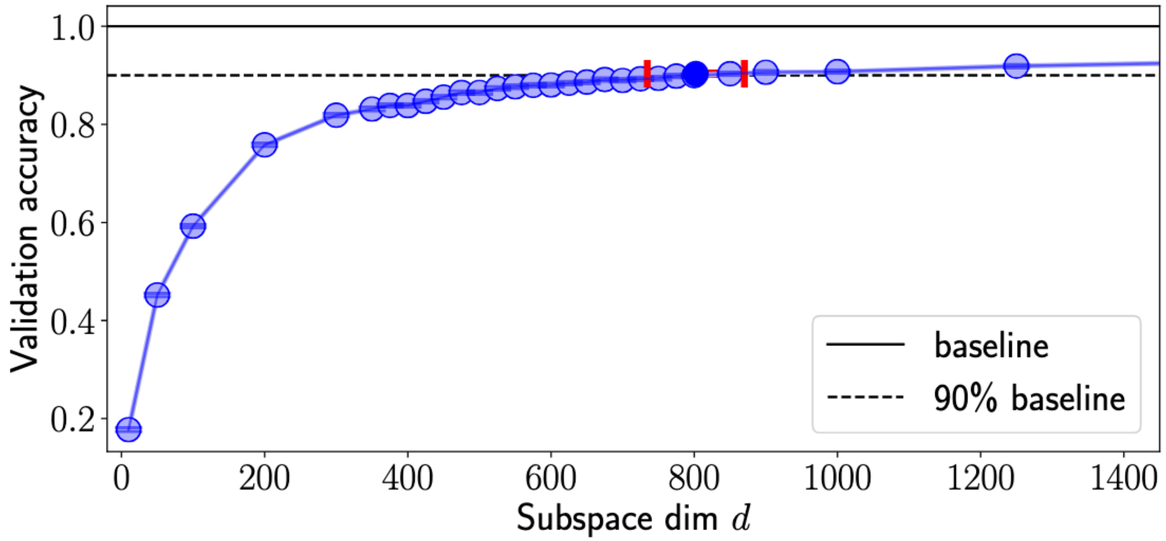
$$\theta^{(D)} = \theta_0^{(D)} + P\theta^{(d)}$$

Li et al. 2018

# Intrinsic Dimension

## Estimating $d$ for MNIST

- Architecture: FCNet with two hidden layers of width 200
- $D = 199{,}210$ but we reach 90% of baseline near $d = 750$

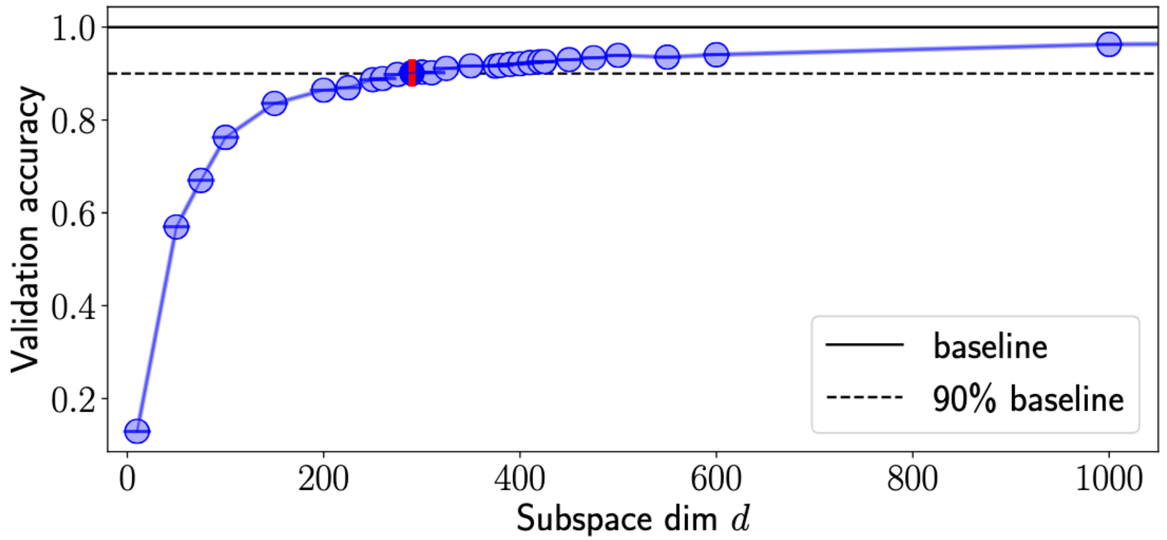



Li et al. 2018

# Intrinsic Dimension

## Estimating $d$ for MNIST

- Architecture: LeNet
- $D = 44{,}426$ but we reach 90% of baseline near $d = 290$




Li et al. 2018

# Intrinsic Dimension

## Estimating $d$ for classic datasets

| **Dataset** | **CIFAR-10** | | **ImageNet** |
|---|---|---|---|
| **Network Type** | **FC** | **LeNet** | **SqueezeNet** |
| Parameter Dim. $D$ | 656,810 | 62,006 | 1,248,424 |
| Intrinsic Dim. $d_{\text{int90}}$ | 9,000 | 2,900 | $>$ 500k |

Li et al. 2018

# Intrinsic Dimension

## Takeaways

- Classic computer vision problems require far fewer dimensions to solve than the size of our models would make you think
- “once a parameter space is large enough to solve a problem, extra parameters serve directly to increase the dimensionality of the solution manifold”
- With larger models, solutions have greater redundancy
—> “cover” more space —> easier to train

Li et al. 2018

# Intrinsic Dimension

## Extended to fine-tuning

- Fine-tuning transformers for classic NLP tasks has a similar phenomenon
- Main change in formulation: $\theta_0^{(D)}$ is now our pre-trained base model
  - (We'll ignore some changes necessary for complexity of large models)

$$\theta^{(D)} = \theta_0^{(D)} + P\theta^{(d)}$$

Gupta et al. 2020

# Intrinsic Dimension

## Extended to fine-tuning

- Fine-tuning transformers for classic NLP tasks has a similar phenomenon

SAID:
small changes
we glossed over

DID:
exactly what we learned

| Model | SAID | DID |
|---|---|---|
| BERT-Base | | |
| BERT-Large | | |
| RoBERTa-Base | | |
| RoBERTa-Large | | |

Gupta et al. 2020

# Intrinsic Dimension

## Extended to fine-tuning

- Fine-tuning transformers for classic NLP tasks has a similar phenomenon

SAID:
small changes
we glossed over

DID:
exactly what we learned

| Model | SAID | | DID | |
|---|---|---|---|---|
| | MRPC | QQP | MRPC | QQP |
| BERT-Base | | | | |
| BERT-Large | | | | |
| RoBERTa-Base | | | | |
| RoBERTa-Large | | | | |

Gupta et al. 2020

# Intrinsic Dimension

## Extended to fine-tuning

- Fine-tuning transformers for classic NLP tasks has a similar phenomenon

SAID: small changes we glossed over

DID: exactly what we learned

| # params | Model | SAID MRPC | SAID QQP | DID MRPC | DID QQP |
|---|---|---|---|---|---|
| 110M | BERT-Base | 1608 | 8030 | 1861 | 9295 |
| 340M | BERT-Large | 1037 | 1200 | 2493 | 1389 |
| 125M | RoBERTa-Base | 896 | 896 | 1000 | 1389 |
| 355M | RoBERTa-Large | **207** | **774** | 322 | **774** |

Gupta et al. 2020

# LoRA

## Low-Rank Adaptation

- Previous papers: models “reside on a low intrinsic dimension”

- LoRA extends this notion to fine-tuning
  - “hypothesize that the change in weights during model adaptation also has a low ‘intrinsic rank’”

Hu et al. 2021

Forget the previous definitions of $d$, $D$, etc.

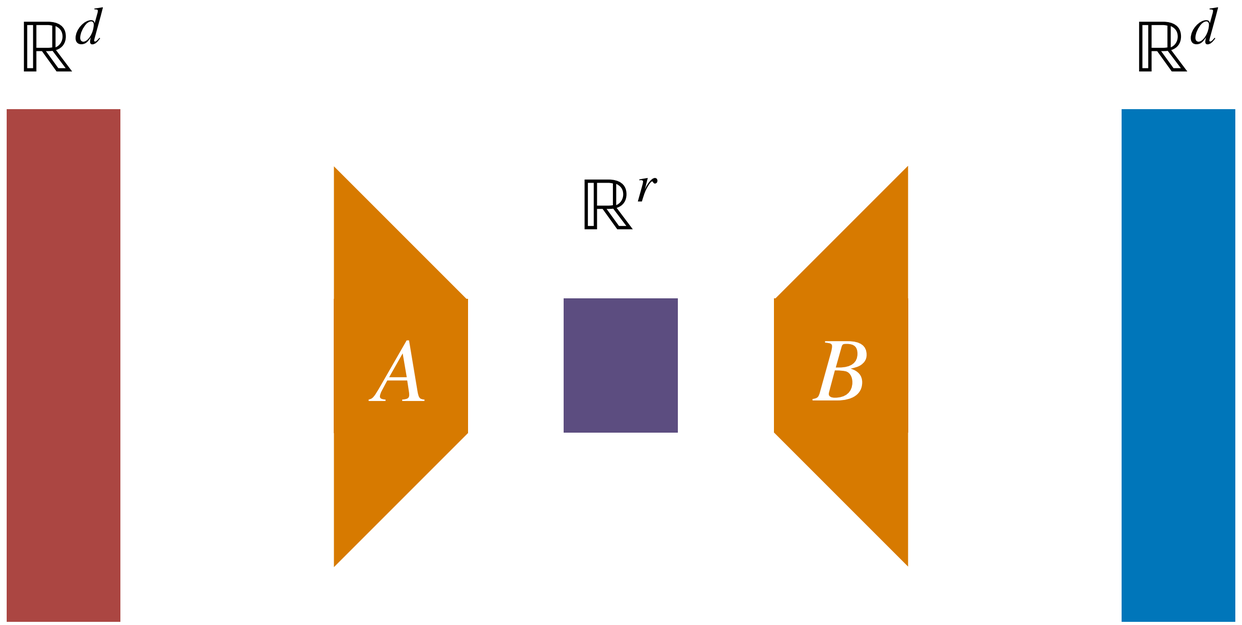
# LoRA

## Linear Transformation of a Full Rank Matrix ( rank = $d$ )


Hu et al. 2021

# LoRA

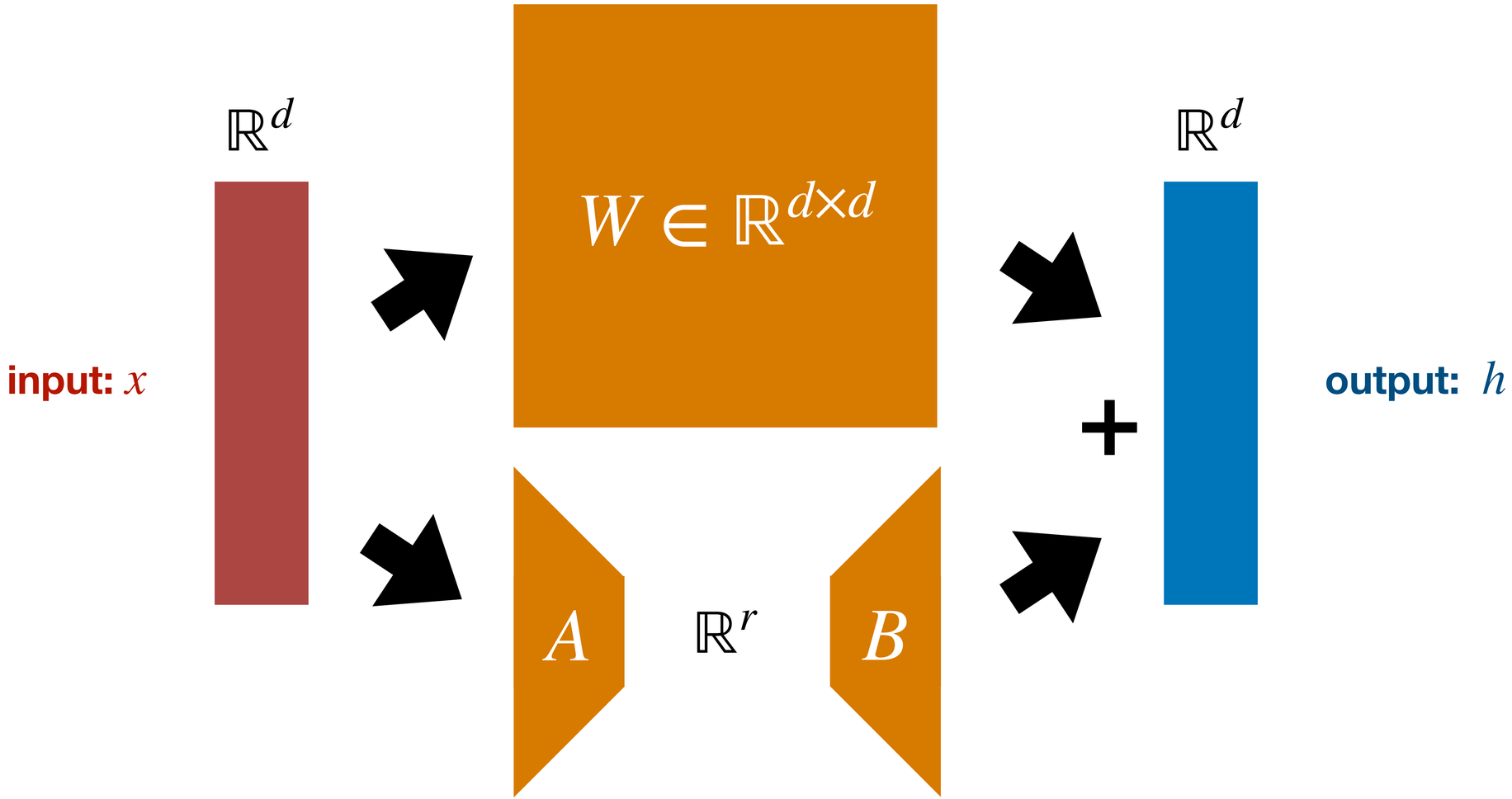
## Linear Transformations via an Intermediate Matrix ( rank = $r << d$ )




Hu et al. 2021

# LoRA

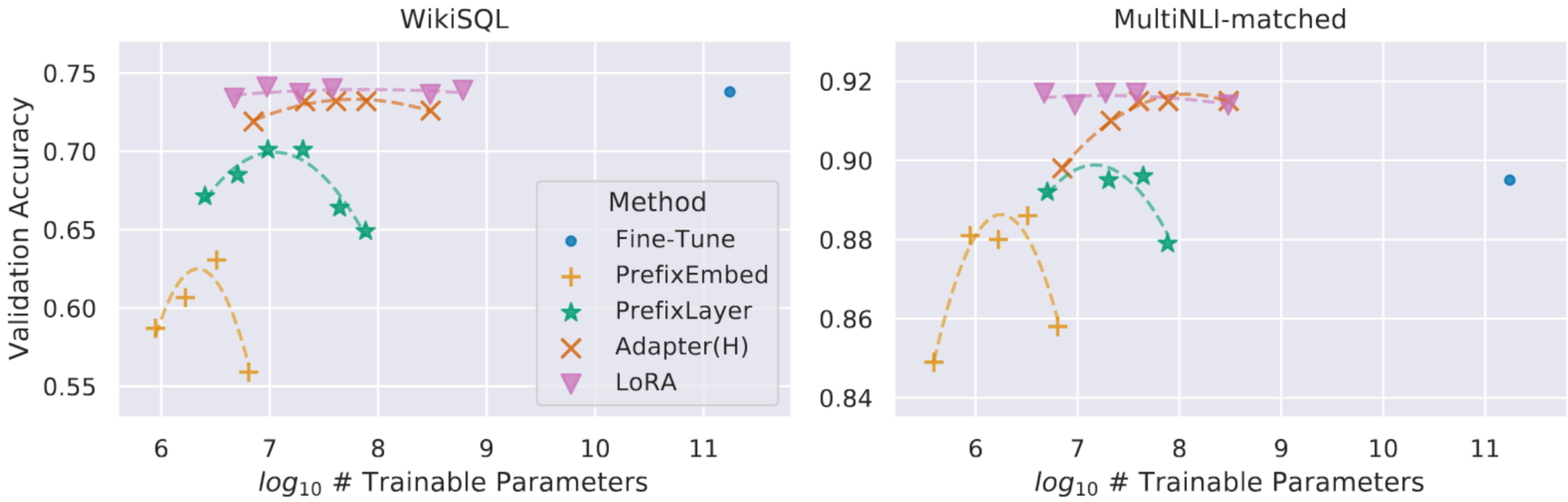
## On a pre-trained model: freeze $W$, train $A, B$




Hu et al. 2021

# LoRA

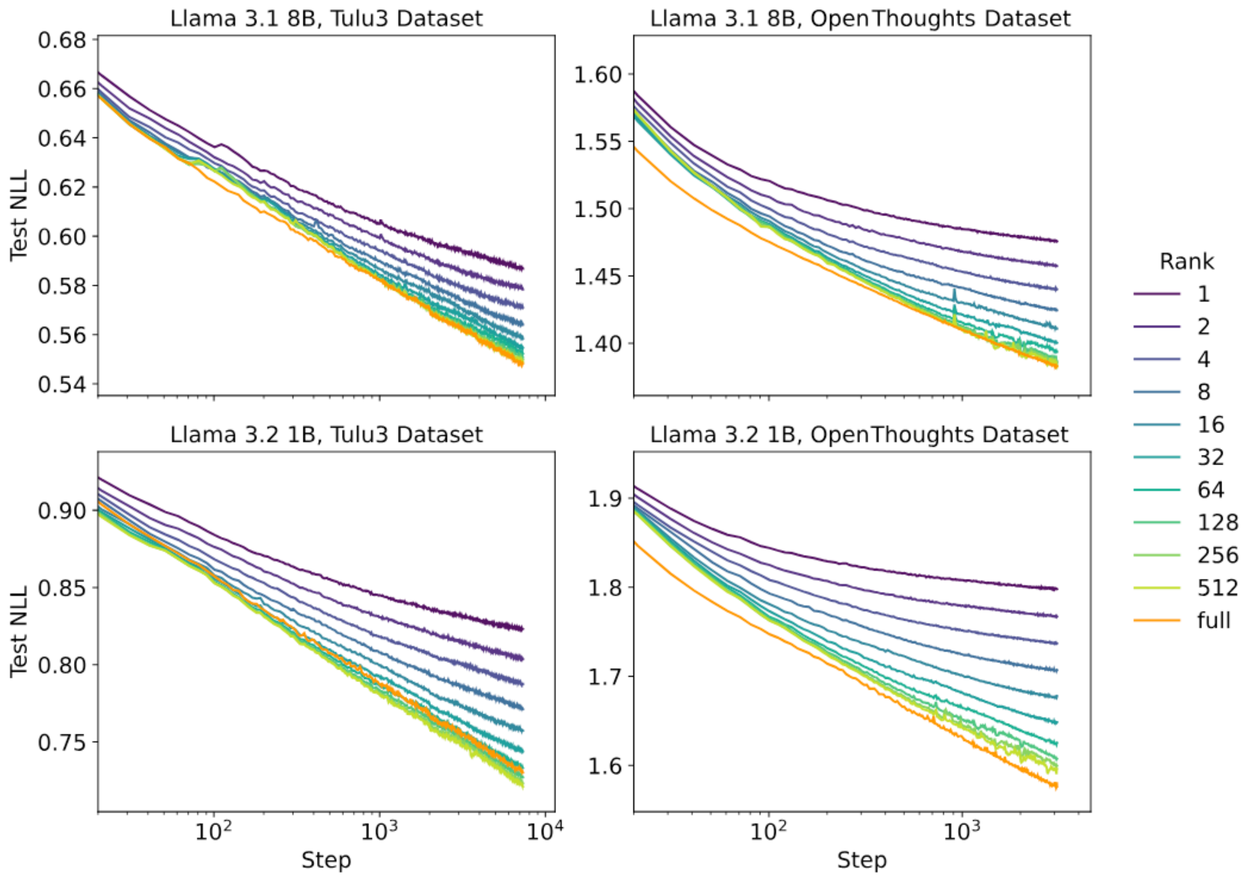
## Compare pink triangles vs. the blue dot




Hu et al. 2021

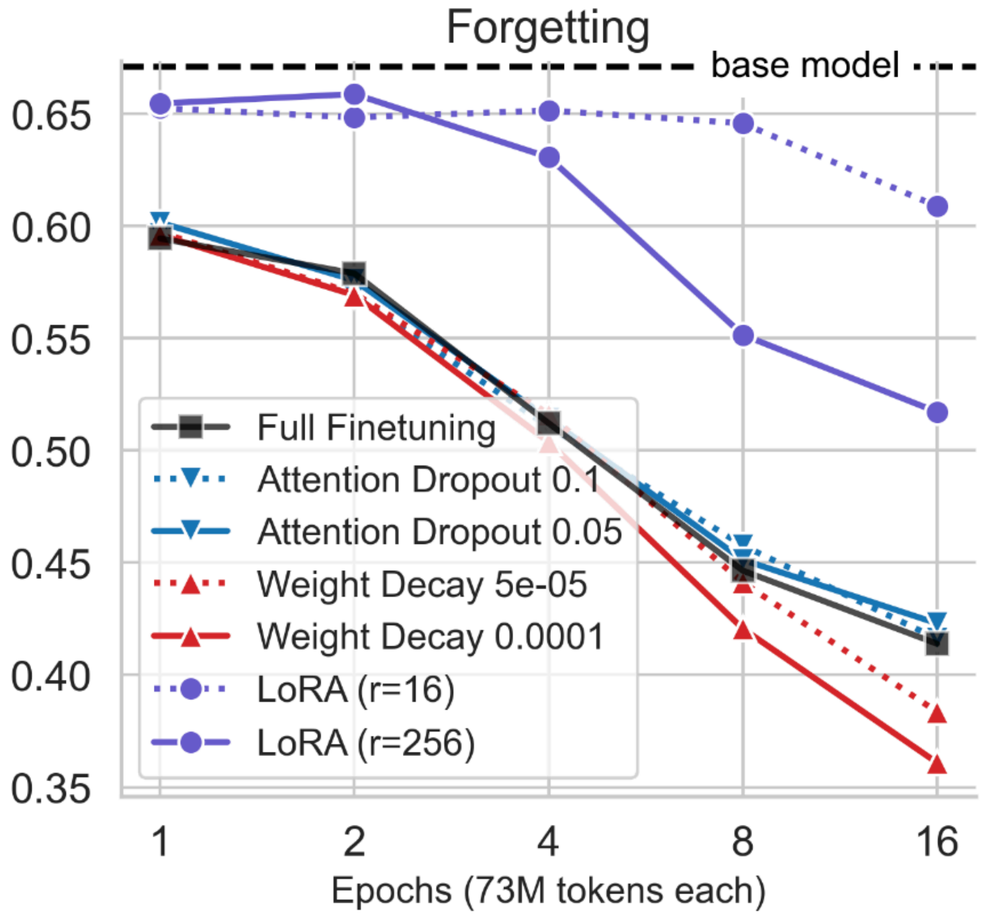
# LoRA

$8 \leq r \leq 512$ **is common in literature (where, often,** $d = 4096$**)**



# LoRA

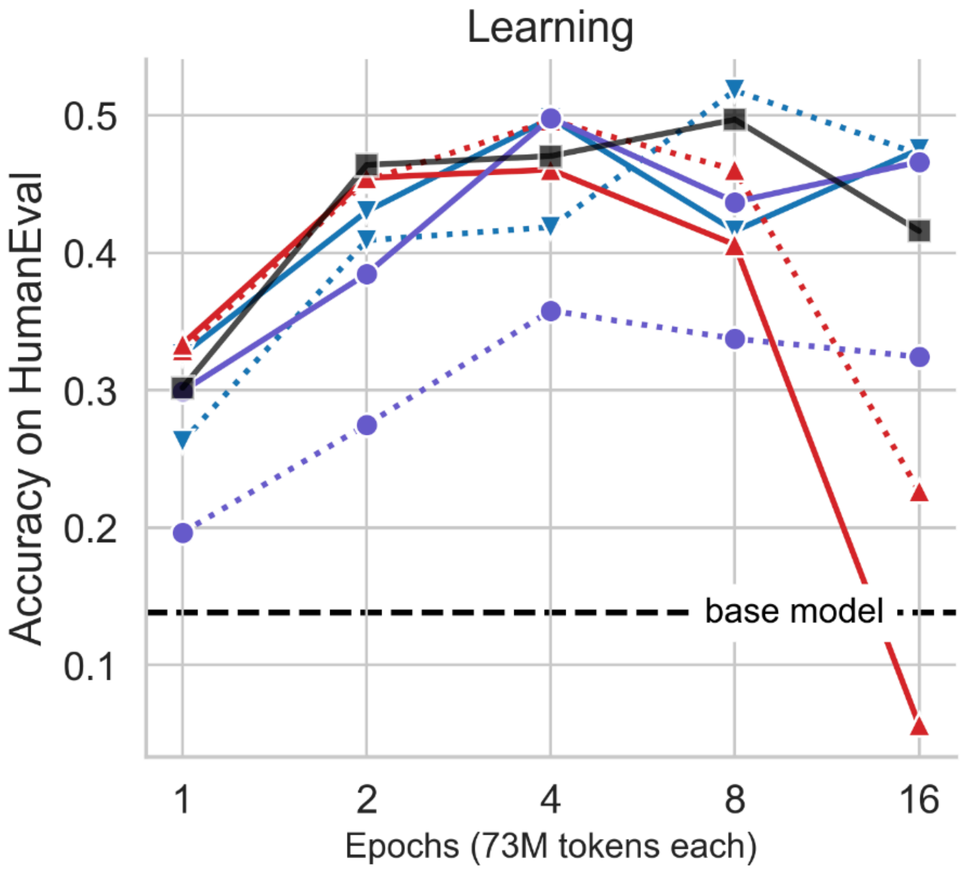
## Less “forgetting” of source domain




Biderman et al. 2024

# LoRA

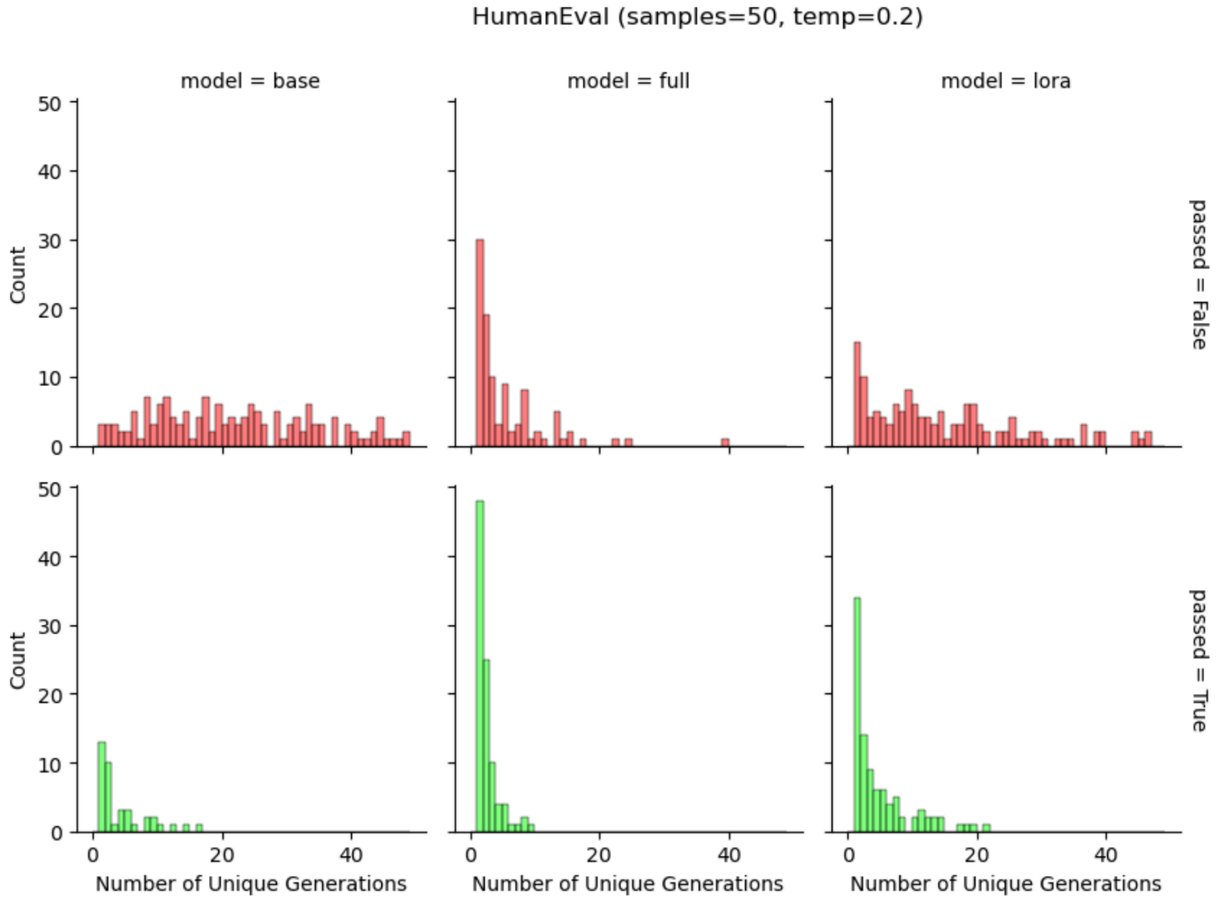
## But still learns (approximately) as much




Biderman et al. 2024

# LoRA

## Preserves token diversity, which full fine-tuning is known to lose




Biderman et al. 2024

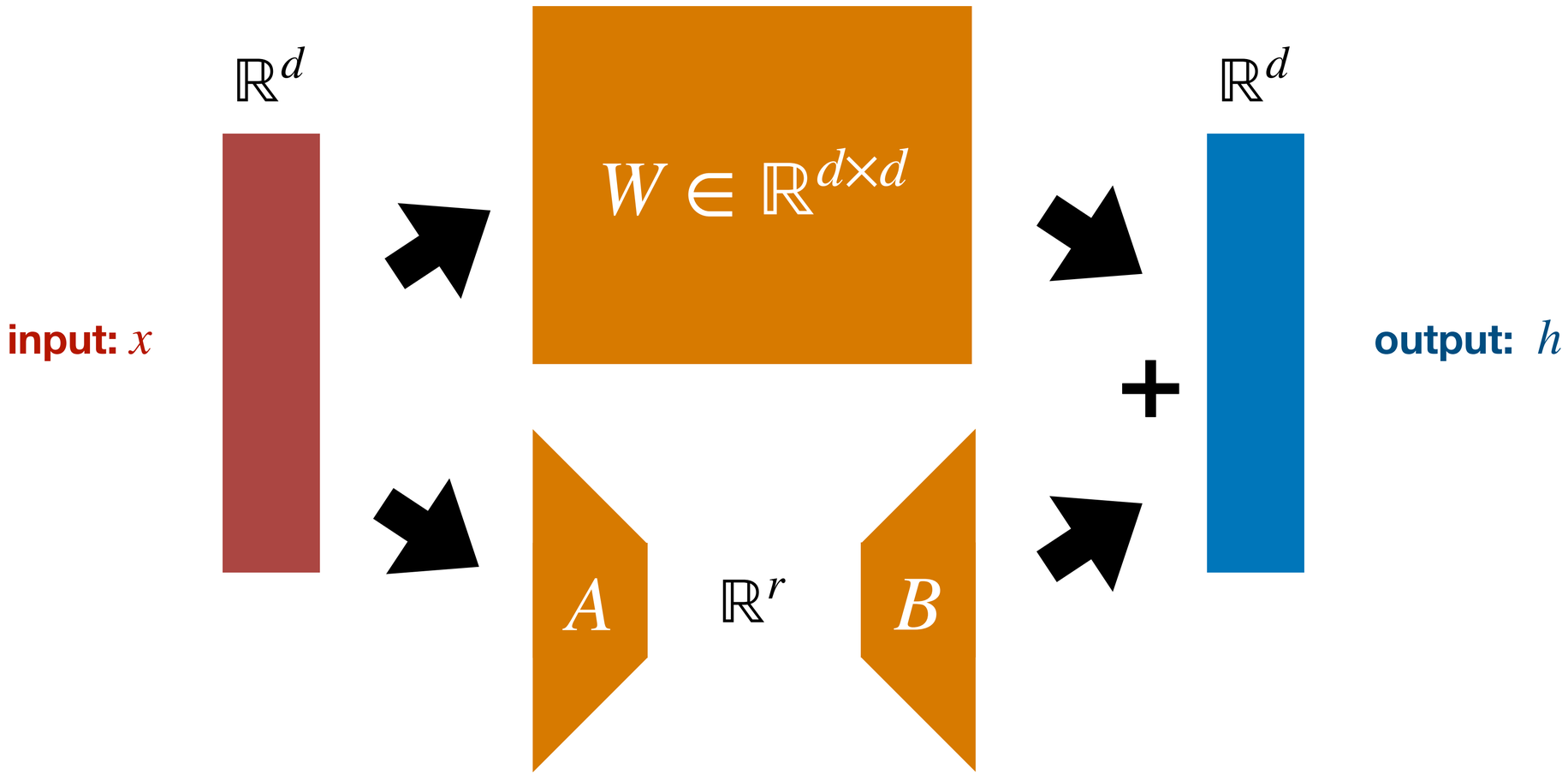
# LoRA

## Too weak sometimes

- olmOCR used full fine-tuning, not LoRA
- OpenThoughts used full fine-tuning, not LoRA

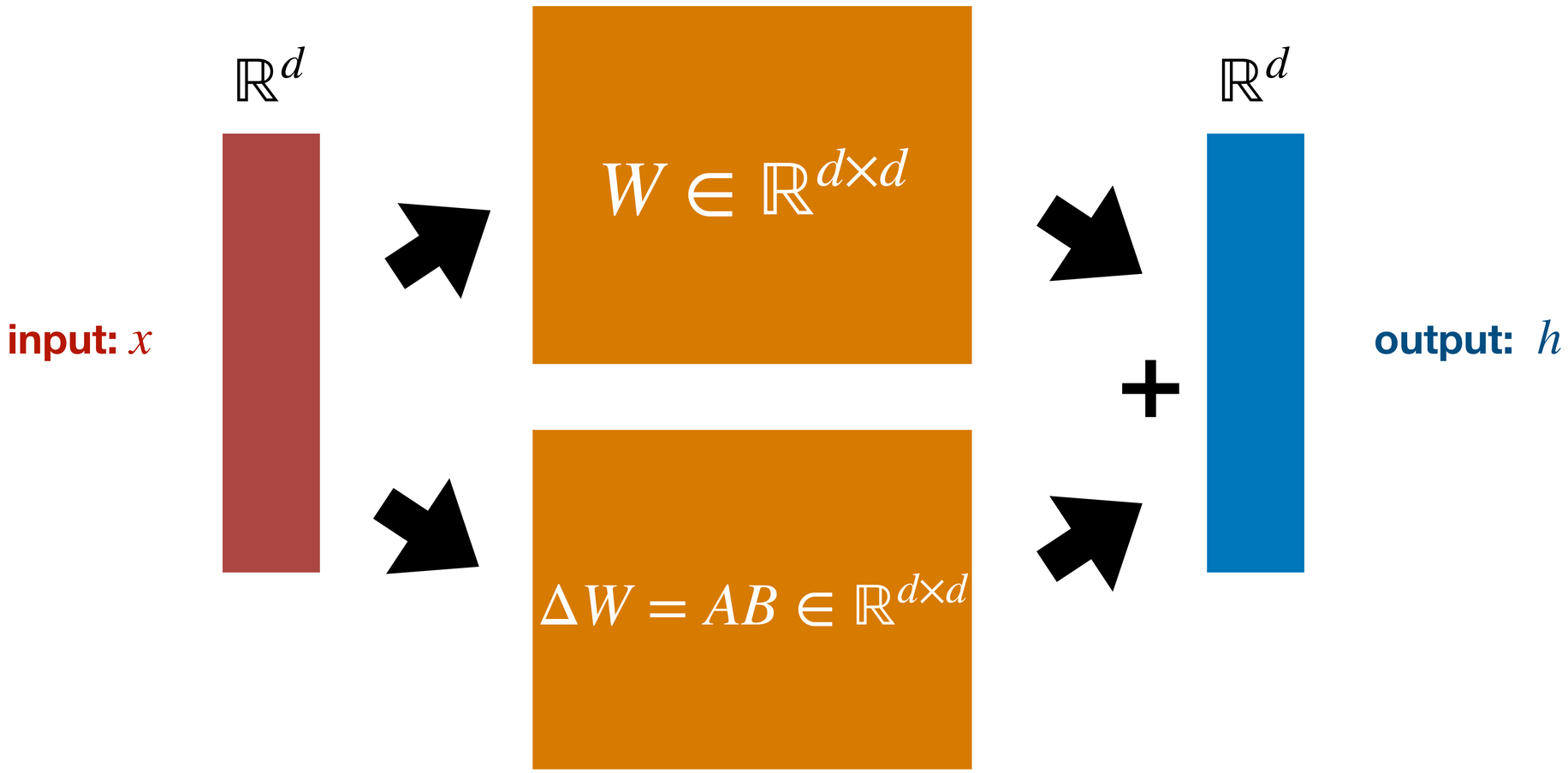
# LoRA

## During deployment



# LoRA

## During deployment

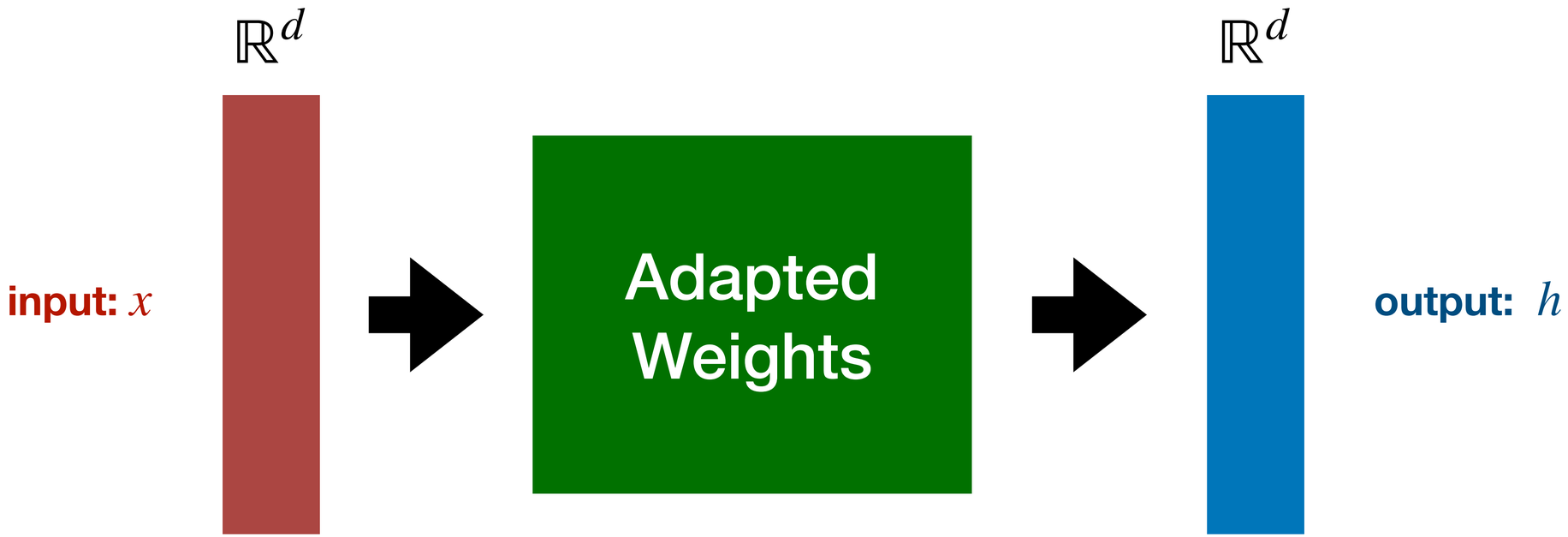


# LoRA

## During deployment

# LoRA

## During deployment



# LoRA

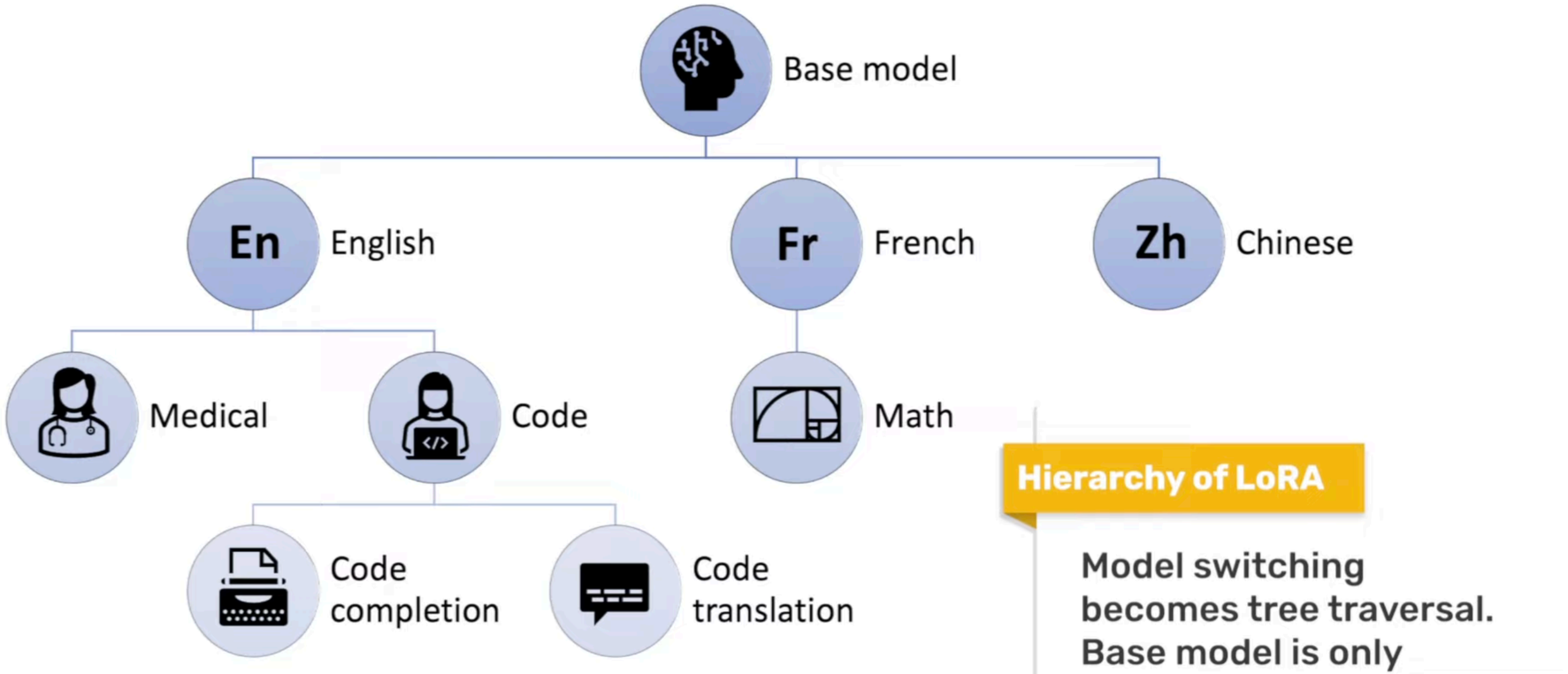
## Switching models

- Subtracting your adapter $\Delta W = AB$ from the updated weight matrix yields your **original** weights!
- Simplifies deploying many fine-tuned models:
  - don’t need to keep multi-terabyte copies of fine-tuned models
  - instead, keep one master copy of the full base model, and then one copy for each of your (small) adapters

Edward Hu

# LoRA

## Hierarchy of modules

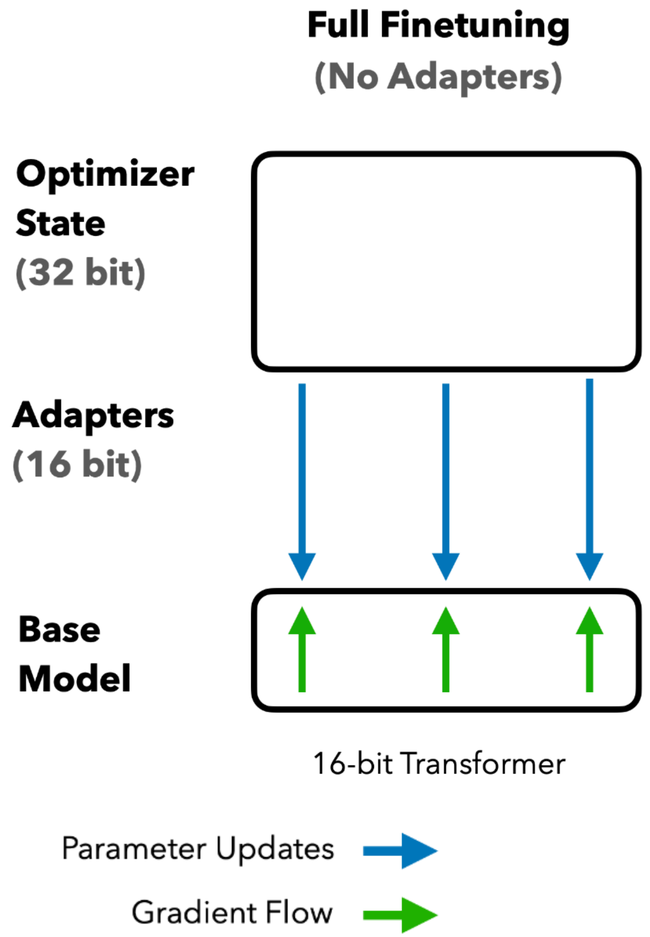


thinkingmachines.ai/blog/lora

# Full FT is expensive

## OK at an industry scale … what about everyone else?

- Cost per param:
  - Weight: 2 byte
  - Weight gradient: 2 byte
  - Optimizer state: 8 byte
  - Total: 12 **bytes** per parameter
- 70B model -> 840 GB of GPU memory
  -> 18x data center GPUs

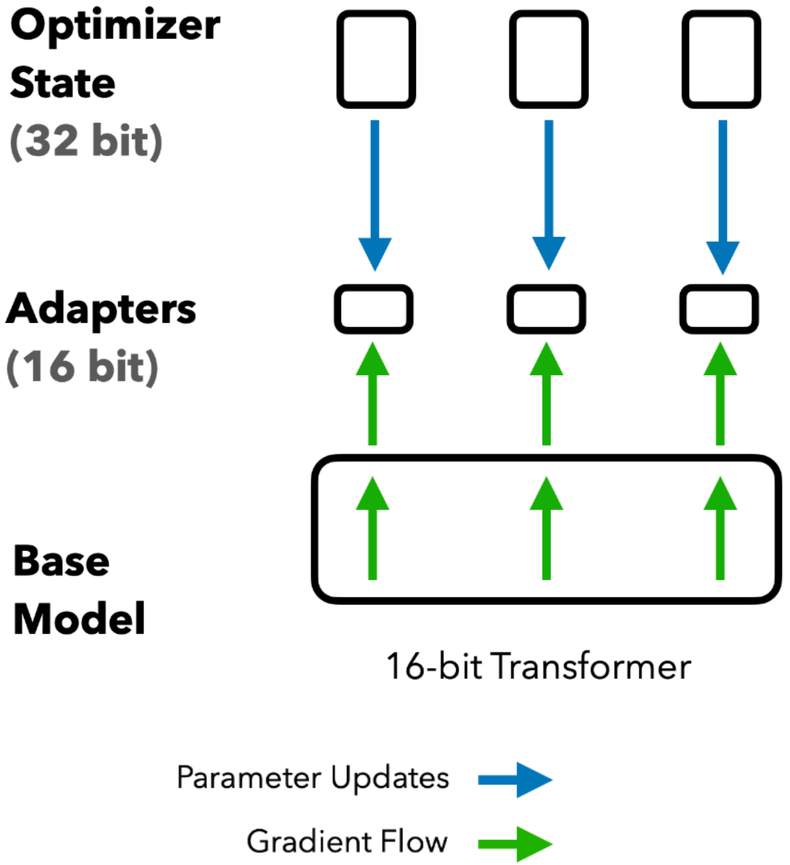



Slide Credit to Tim Dettmers

# LoRA is expensive

## OK at an R1 scale … what about everyone else?

- Cost per param:
  - Weight: 16 bits
  - Weight gradient: ~0.4 bit
  - Optimizer state: ~0.8 bit
  - Adapter weights: ~0.4 bit
  - Total: 17.6 **bits** per parameter
- 70B model -> 154 GB of GPU memory
  -> 4x data center GPUs
  OR 8x consumer GPUs

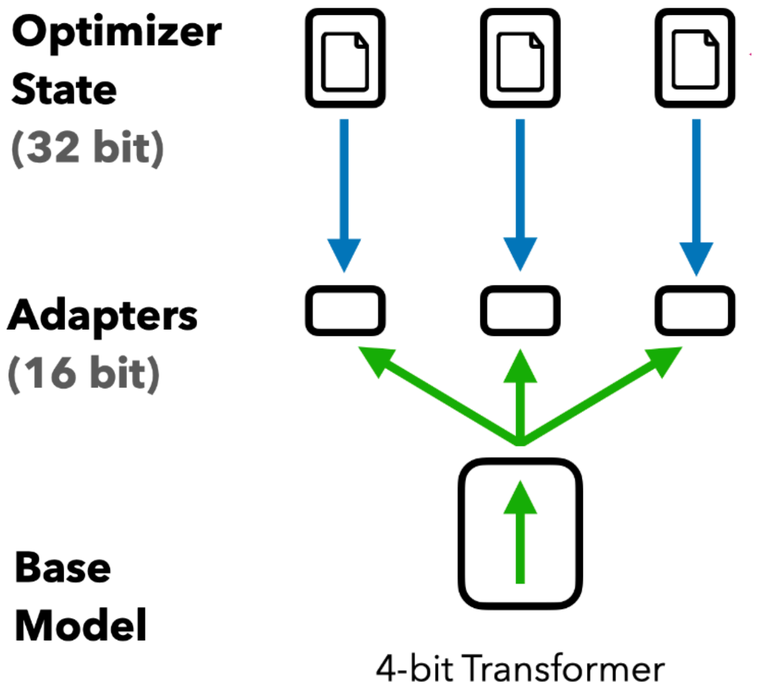



Slide Credit to Tim Dettmers

# QLoRA is affordable

## Fine-tuning for everyone!

- Cost per param:
  - Weight: **4 bits**
  - Weight gradient: ~0.4 bit
  - Optimizer state: ~0.8 bit
  - Adapter weights: ~0.4 bit
  - Total: 5.2 **bits** per parameter
- 70B model -> 46 GB of GPU memory
  -> 1x data center GPUs
  OR 2x consumer GPUs




Slide Credit to Tim Dettmers

# QLoRA

## Core Components

- **Double quantization**: reduce space taken up by quantization constants by quantizing those constants themselves!
  - saves 0.4 bits of space per weight
- **Quantize model to 4 bits**: uses new data type (NF4) that is optimal for normal distribution
- **Paged optimizers**: engineering trick for GPU memory spikes; we skip this

Dettmers et al. 2023

# QLoRA

## Produces LoRA Adapters on Fewer GPUs

- 4-bit quantized weights used *during* QLoRA
  - not required to keep them quantized during serving
- QLoRA yields **16 bit** adapters
  - could add/merge these with original weights then re-quantize
  - could swap them into original weights

# Summary

- Great practical utility in fine-tuning base models for specific applications
- Full fine-tuning is prohibitively expensive to conduct and annoying to serve
- LoRA only adapts a certain subset of the model's parameters
  - Motivated by notion of a model's intrinsic dimension being small
  - Simplifies model serving since adapters can be easily "swapped" in/out
- QLoRA reduces the memory overhead when training LoRA adapters