CS 839: Foundation Models
Specialization: Fine-Tuning, Adaptation, Editing

Fred Sala

University of Wisconsin-Madison

Oct. 3rd, 2024

Announcements

- **Logistics:**
  - Homework 1 is due tonight!
- Class roadmap:

Thursday Oct. 3	Specialization
Tuesday Oct. 8	Alignment
Thursday Oct. 10	Efficient Training
Tuesday Oct. 15	Efficient Inference
Thursday Oct. 17	Prospective Guest Lecture

Language & Foundation Models

Outline

- **Fine-Tuning and Adapter Intro**
  - Fine-tuning vs. prompting, linear probing, etc. Full vs partial fine tuning vs adapting. Popular adapters
- **Cross-Modal Adaptation**
  - Frozen transformers, ORCA, aligning via optimal transport dataset distance
- **Model Editing**
  - Idea, MEND

Outline

- **Fine-Tuning and Adapter Intro**
  - Fine-tuning vs. prompting, linear probing, etc. Full vs partial fine tuning vs adapting. Popular adapters
- **Cross-Modal Adaptation**
  - Frozen transformers, ORCA, aligning via optimal transport dataset distance
- **Model Editing**
  - Idea, MEND

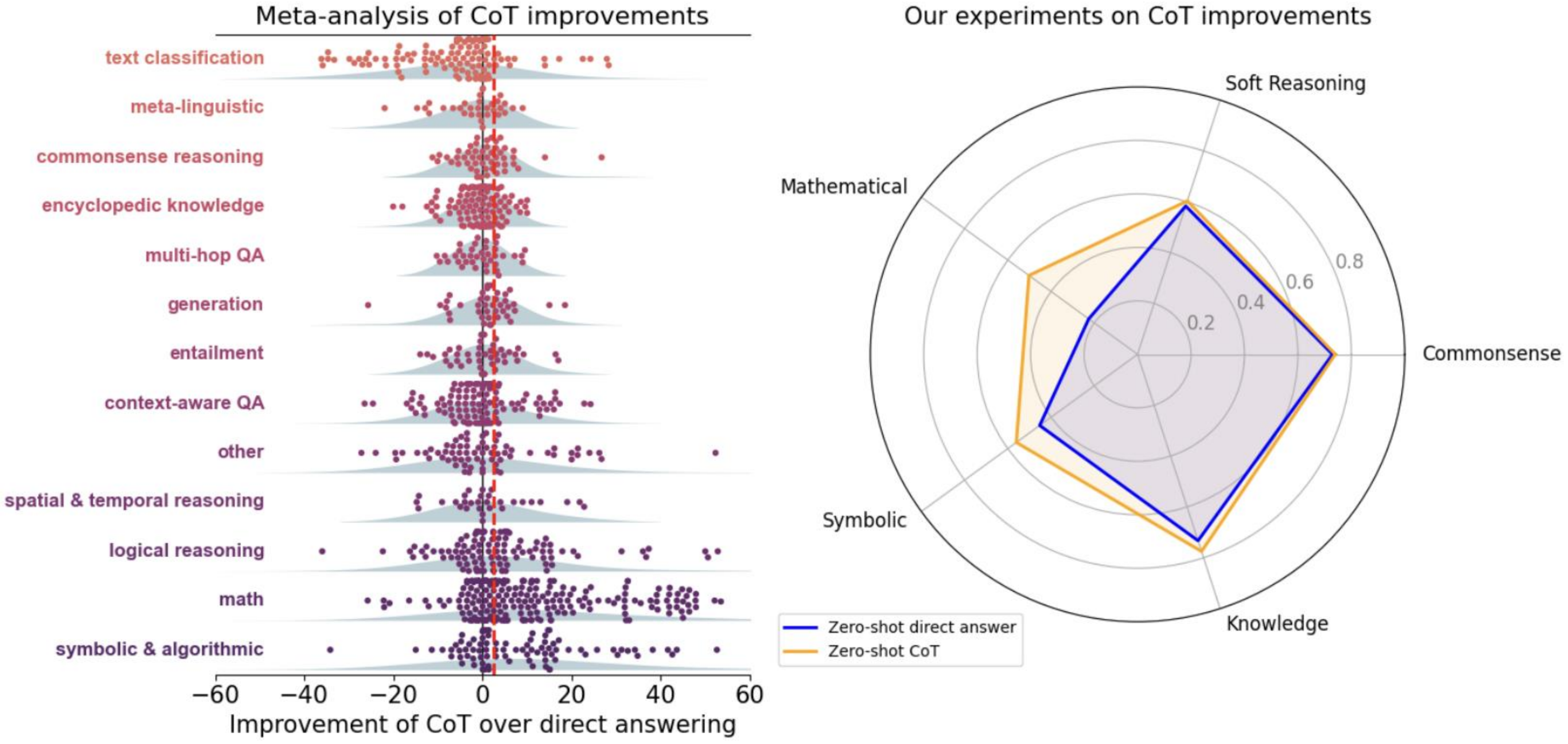
When Does CoT Actually Help?

A: Not always clear

- Mainly on math and symbolic reasoning?

To CoT or not to CoT? Chai
math and symbolic reasoni

Zayne Sprague, Fangcong Yin, Juan Diego I
Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle M



Sprage et al ‘24

From Last Time: Beyond the Unaided Language Models

Even when we do CoT, the language model can get things wrong.

- Often simple things… like **arithmetic**.

- How else can we help it?

- **A:** Use external tools

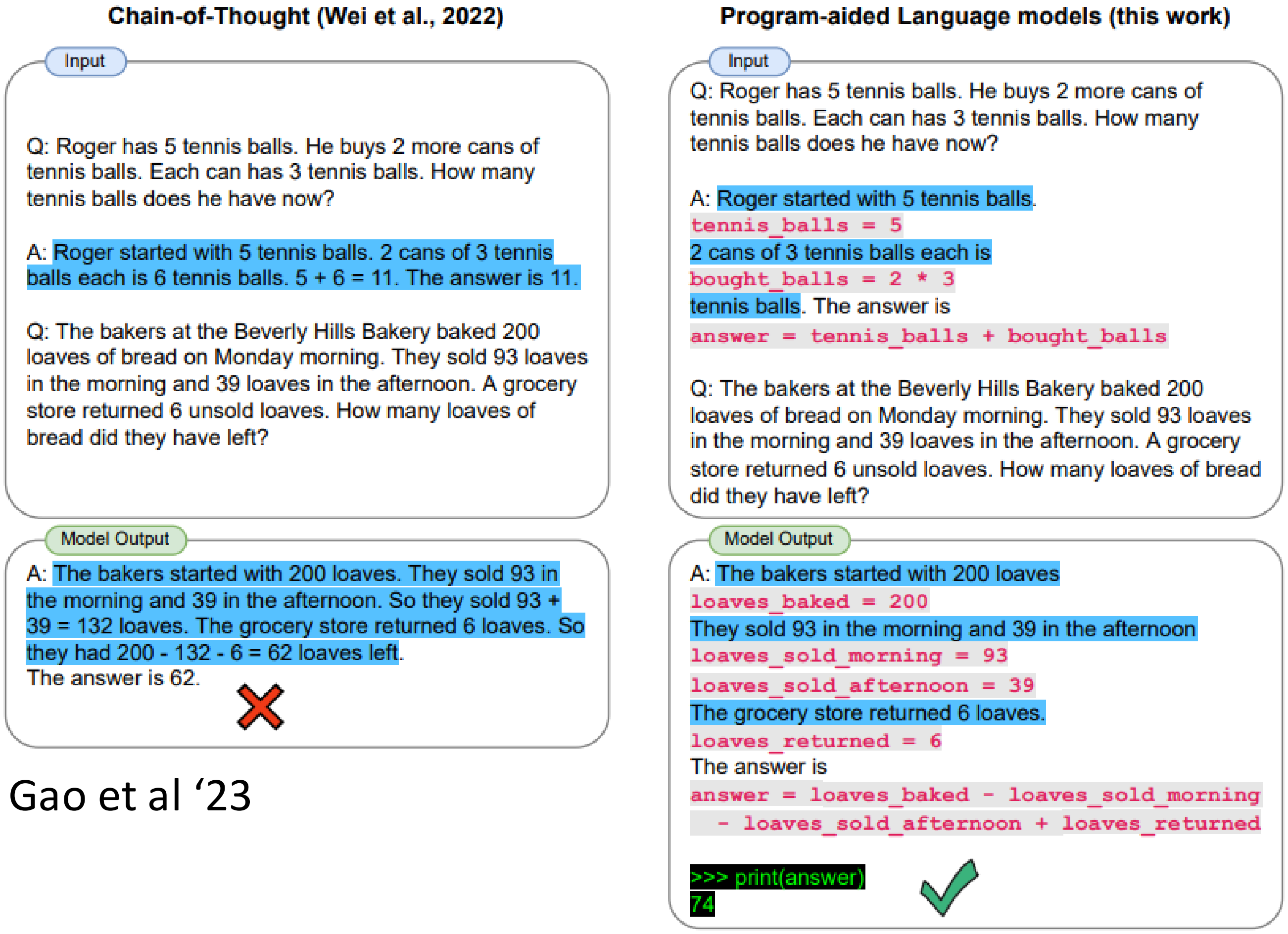
Tools: **Program-aided LMs**

Use external tools:

- Python interpreter

- How? *Interleave* the text explanations in CoT steps with lines of Python code

- LMs can already output code
  - Just need to *prompt* the right way



Gao et al '23

Tools: Programs-aided LMs **Prompts**

Just need to *prompt* the right way

- Craft examples that interleave code and text

Q: Olivia has $23. She bought five bagels for $3 each. How much money does she have left?

```
money_initial = 23
bagels = 5
bagel_cost = 3
money_spent = bagels * bagel_cost
money_left = money_initial - money_spent
answer = money_left
```

Figure 3: Example prompt for the mathematical reasoning tasks, from the GSM8K benchmark.

Q: On the table, you see a bunch of objects arranged in a row: a purple paperclip, a pink stress ball, a brown keychain, a green scrunchiephone charger, a mauve fidget spinner, and a burgundy pen. What is the color of the object directly to the right of the stress ball?

```
...
stress_ball_idx = None
for i, object in enumerate(objects):
    if object[0] == 'stress ball':
        stress_ball_idx = i
        break
# Find the directly right object
direct_right = objects[stress_ball_idx+1]
# Check the directly right object's color
answer = direct_right[1]
```

Tools: **Program-of-Thoughts**

Similar idea: program-of-thoughts

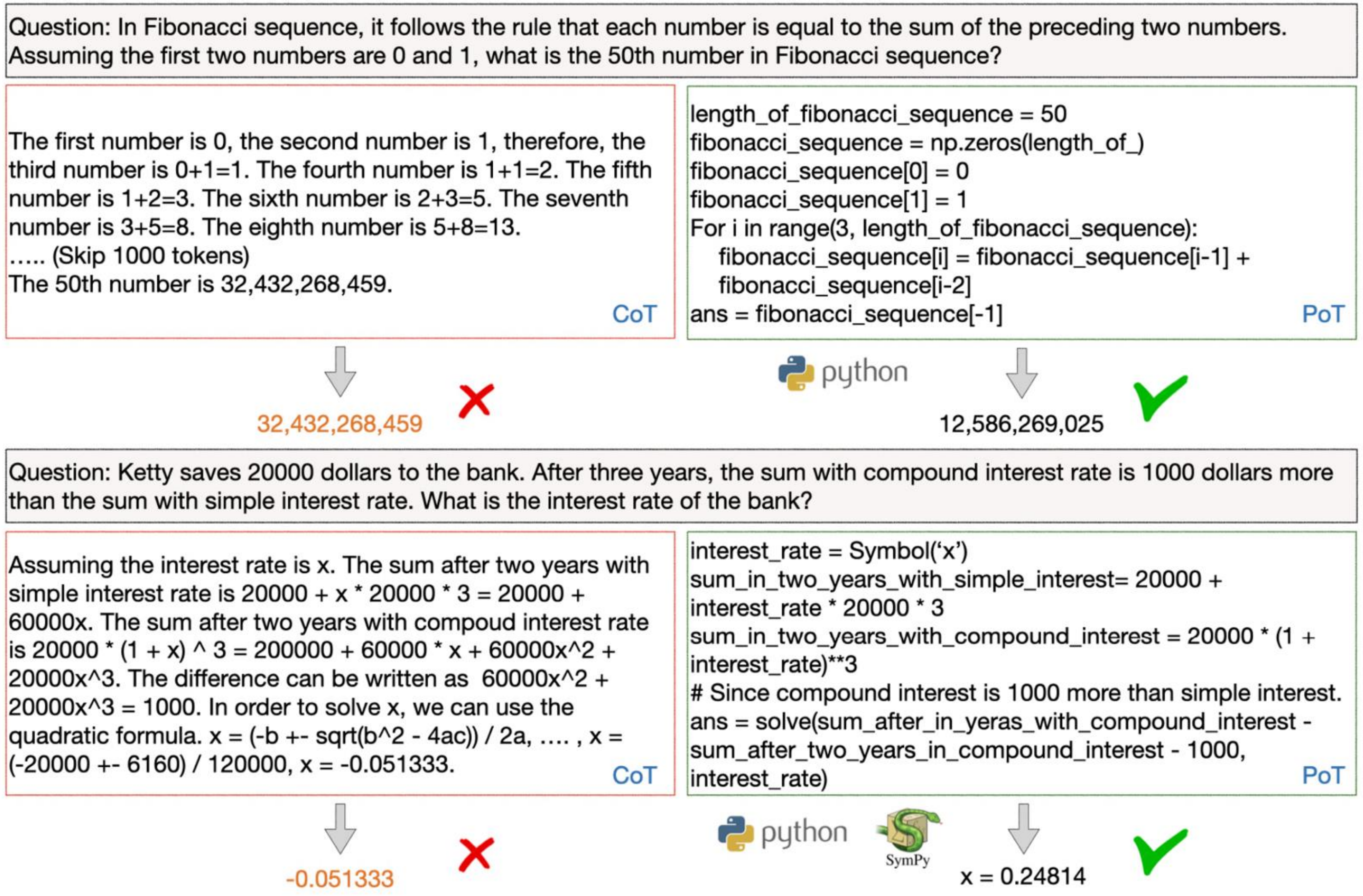


Figure 1: Comparison between Chain of Thoughts and Program of Thoughts.

Chen et al '22

Tools: **More General Tools**

Ideally, use more general external tools

- Without lots of human annotation
- Model should decide on its own which tool to use

- **Toolformer**: introduces API calls into the model
  - But these API calls aren't already there… so need to fine-tune

Your task is to add calls to a Question Answering API to a piece of text. The questions should help you get information required to complete the text. You can call the API by writing "[QA(question)]" where "question" is the question you want to ask. Here are some examples of API calls:

Input: Joe Biden was born in Scranton, Pennsylvania.

Output: Joe Biden was born in **[QA("Where was Joe Biden born?")]** Scranton, **[QA("In which state is Scranton?")]** Pennsylvania.

Input: Coca-Cola, or Coke, is a carbonated soft drink manufactured by the Coca-Cola Company.

Output: Coca-Cola, or **[QA("What other name is Coca-Cola known by?")]** Coke, is a carbonated soft drink manufactured by **[QA("Who manufactures Coca-Cola?")]** the Coca-Cola Company.

Input: $\mathbf{x}$

Output:

Schick et al '23

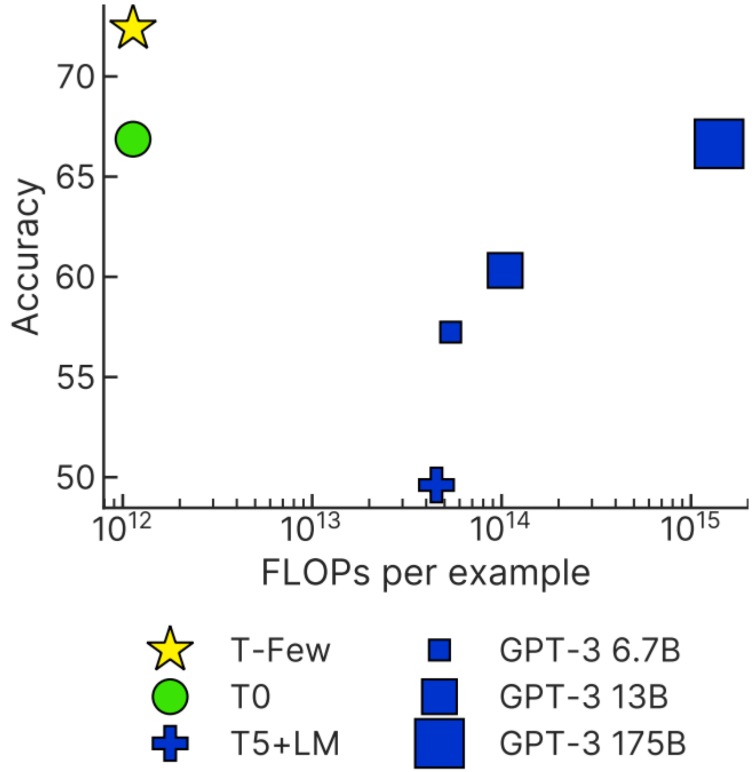
Before: **Prompting**

With prompting, we didn't change the model

- To improve performance, we used few-shot/ICL
- But, this might be **worse** than changing our model weights

Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning

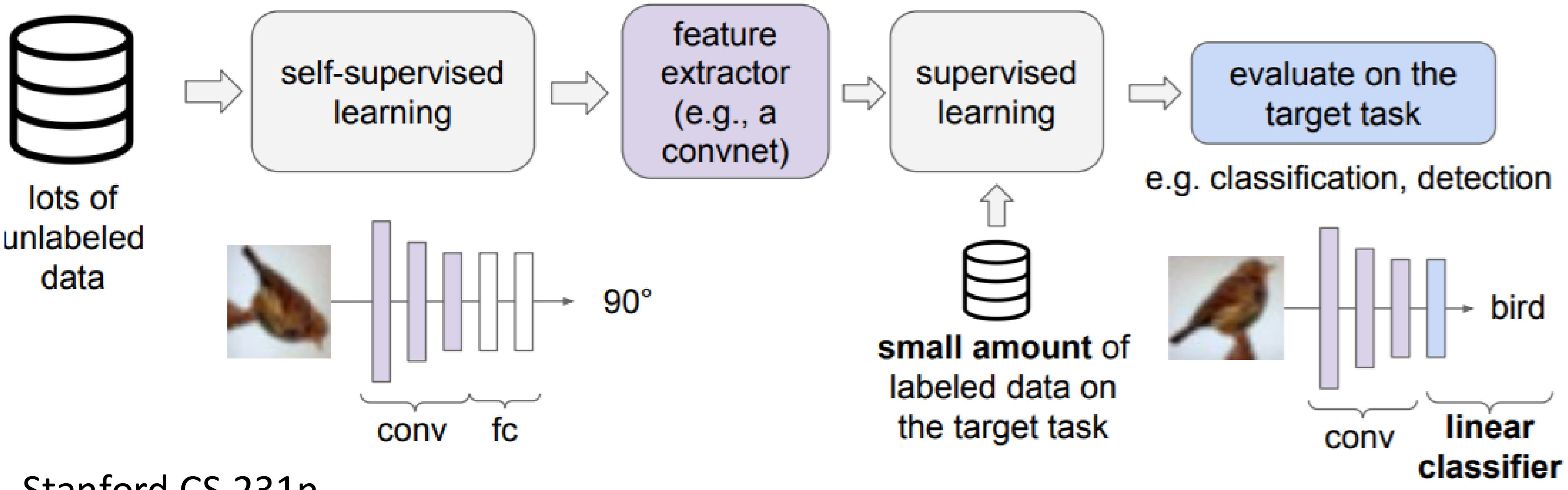
Liu et al '22



Before: **Frozen Models/Linear Probing**

We previously discussed freezing our model, and using just some trainable heads

- E.g., a linear model on top (called **linear probing**)
- Our self-supervised learning example



Stanford CS 231n

Full Fine-Tuning

Performance might still be bottlenecked,

- Frozen representations might not be suitable for task
- Might need lots of capacity on top to adapt
- **Change all the weights!**

```
>>> from transformers import AutoModelForSequenceClassification

>>> model = AutoModelForSequenceClassification.from_pretrained("bert-base-cased", num_labels=5)
```

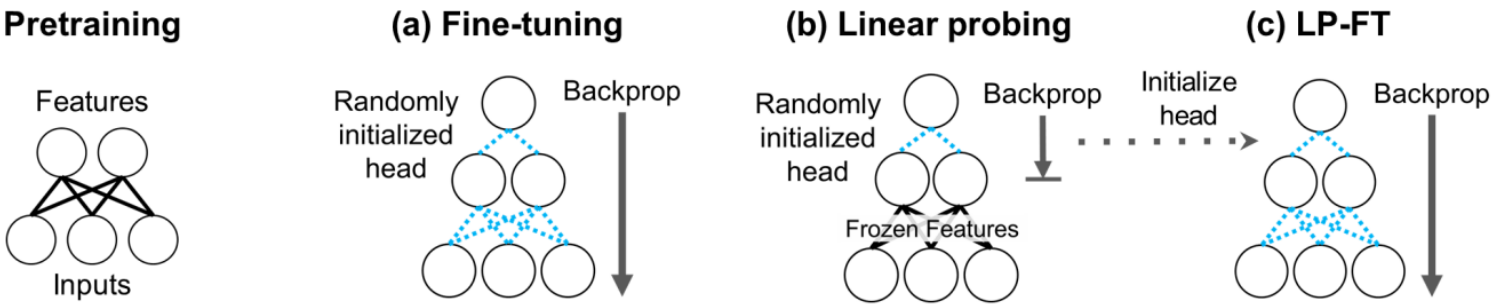
```
>>> trainer.train()
```

https://huggingface.co/docs/transformers/training

Full Fine-Tuning: **Downsides**

Fine-tuning all parameters is tough:

1. **Expensive:** just like training a full model
2. **Known to cause issues on OOD data…**
   - Fine-Tuning can Distort Pretrained Features and Underperform Out-of-Distribution



ID test	85.1%	82.9%	**85.7%**
OOD test	59.3%	66.2%	**68.9%**

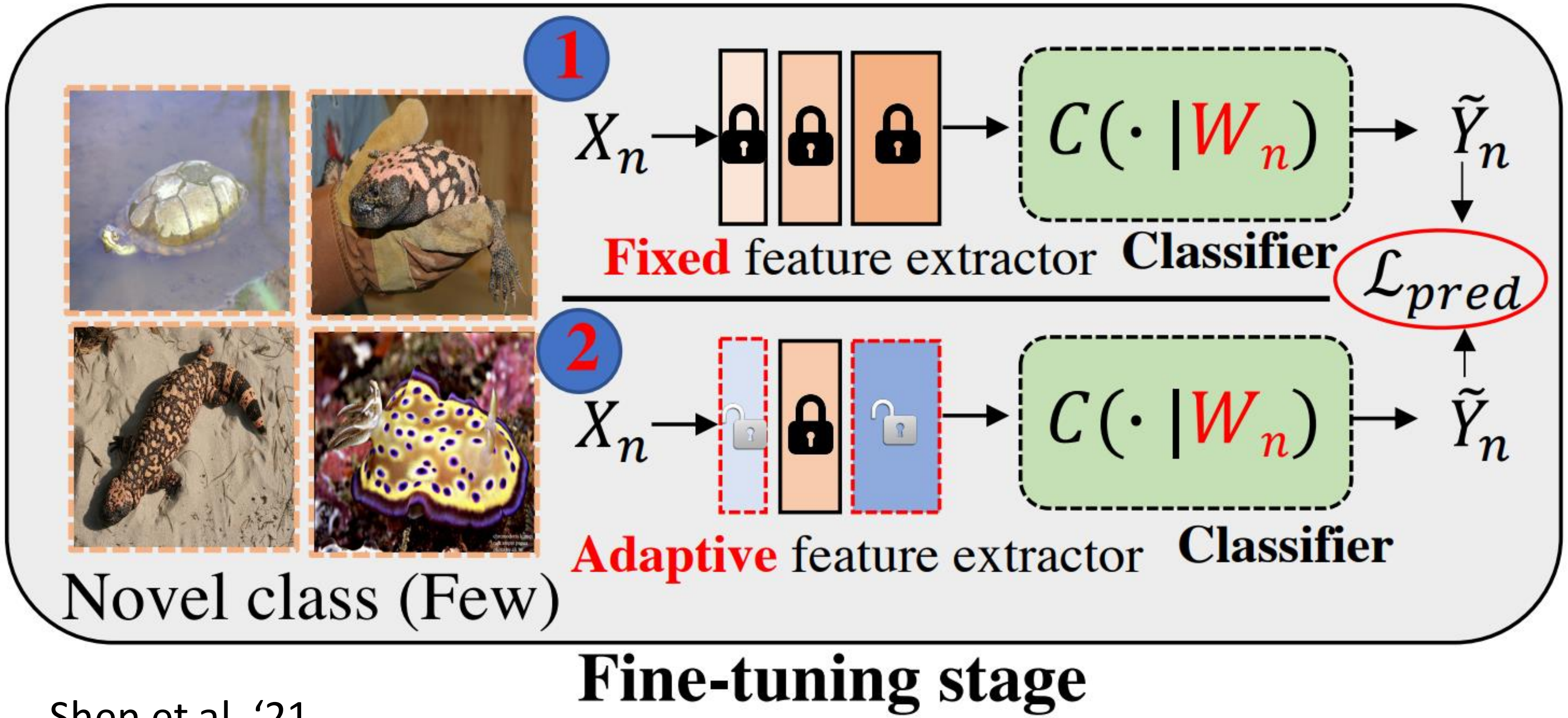
Average accuracies (10 distribution shifts)

Kumar et al '22

Partial Fine-Tuning

Full fine-tuning might be expensive

- Partial fine-tuning might be a good choice
- Only some layers change

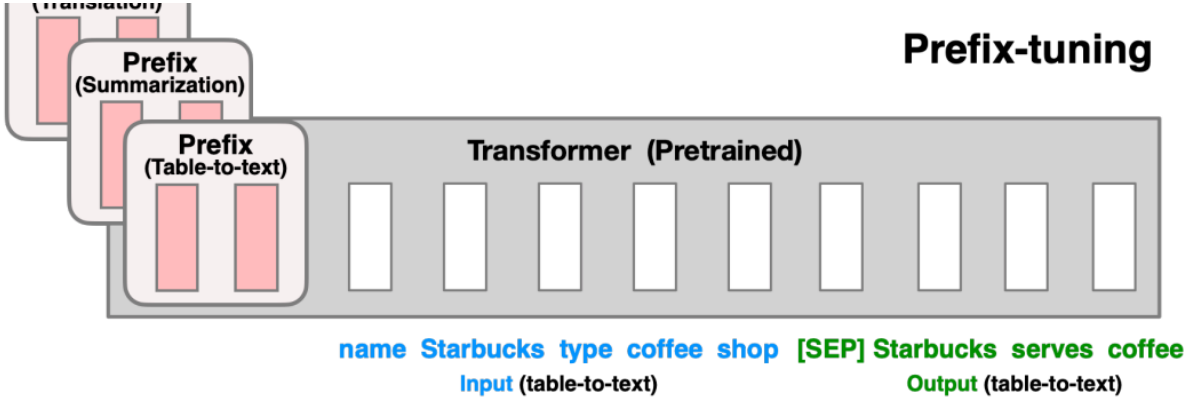


Shen et al, '21

Prefix-Tuning

Recall this *soft prompting* method.

- Prefixes are trainable parameters
- Train one for each goal task, only store these new parameters
- Enables cheap **adaptation** of frozen language model

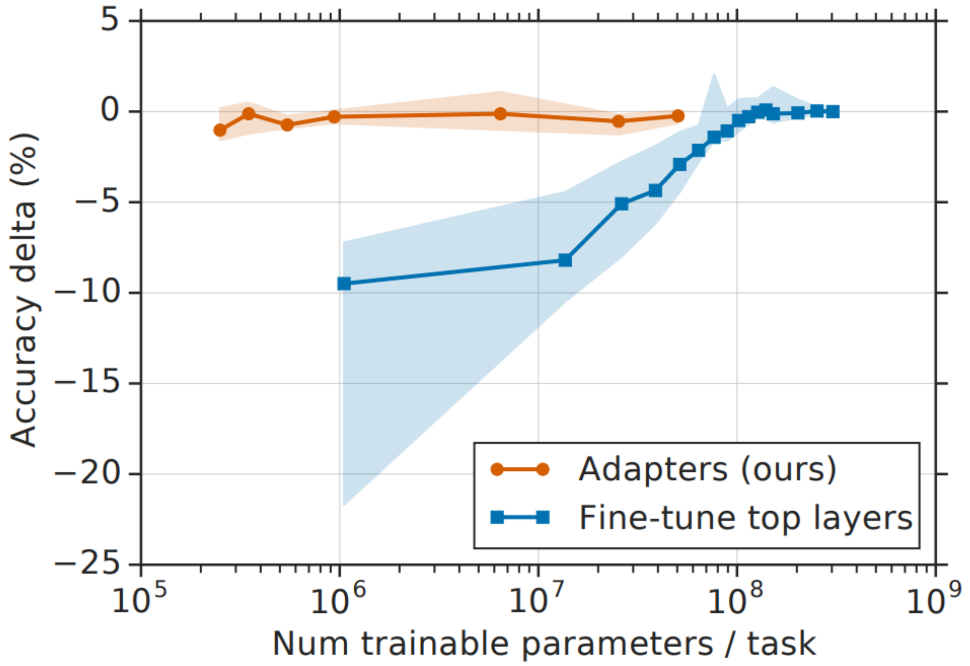


Li and Liang '21

Parameter-Efficient Fine-Tuning (PEFT)

None of these methods were full satisfying

- Have to figure out what layers to train, have to interpolate with prompts, etc.
  - Lots of choices!
- If we fine-tune too many parameters, that gets expensive…
  - But top only, performance isn't **great**
- Houlsby et al '19:

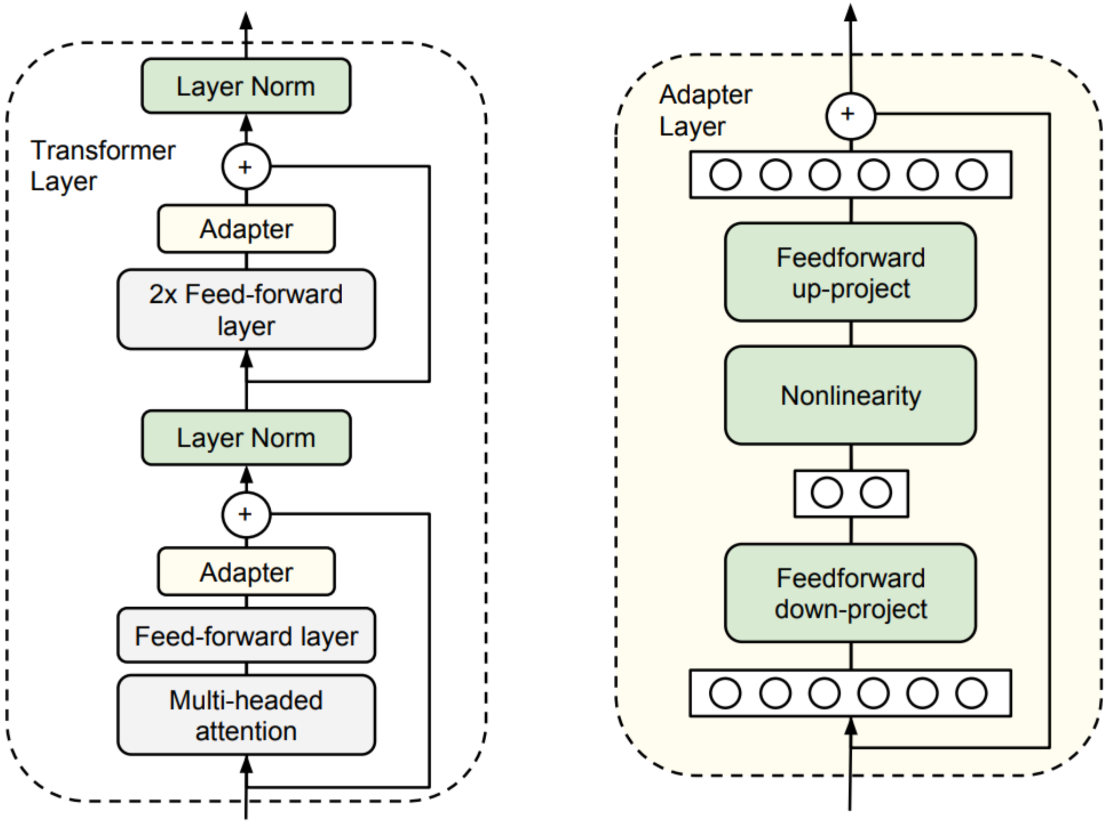


PEFT: **Adapters**

Want two things in PEFT
- Good performance (accuracy, etc.)
- Parameter efficiency

- Solution: **Adapters**
  - Small modules, inserted in between model and trained

Another **advantage:** no change to model, new modules for tasks

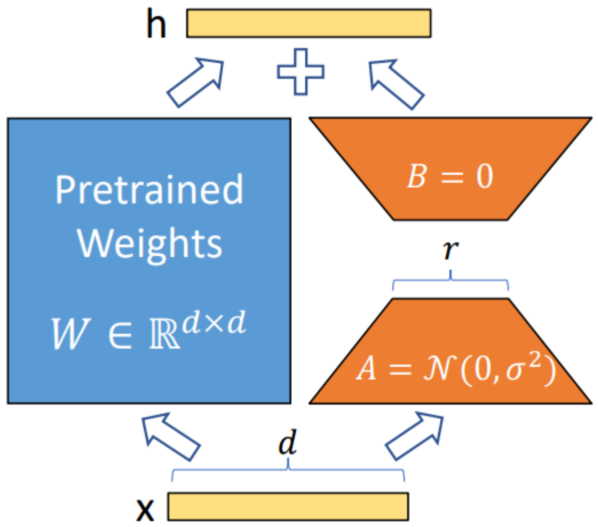


Houlsby et al '19

PEFT: **Low-Rank Adapters (LoRA)**

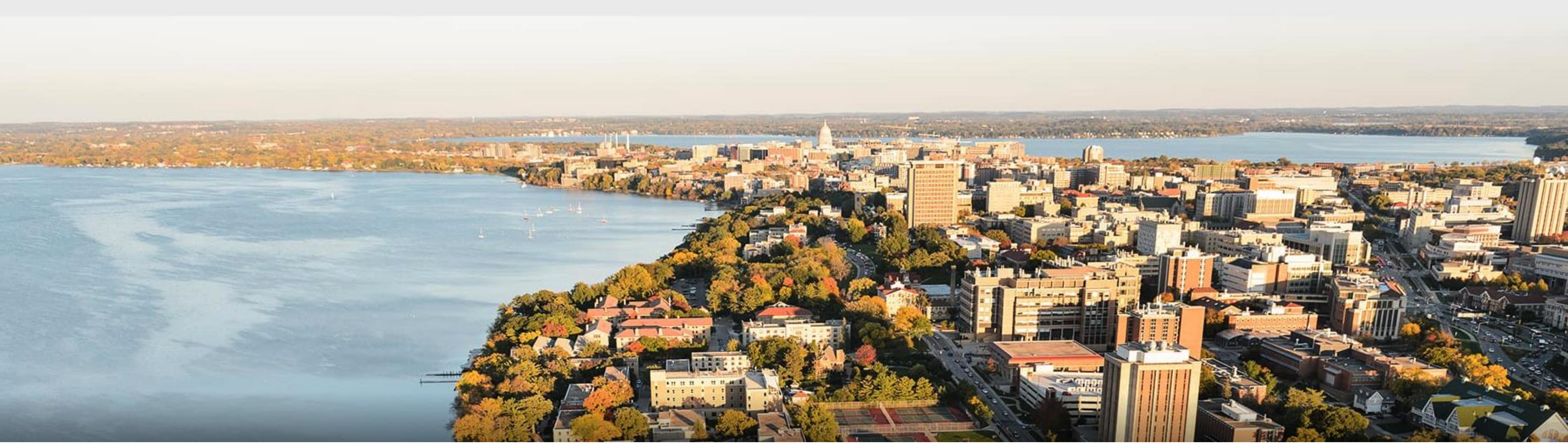
Perhaps the most popular variant

- LoRA suggests **adding** directly to pretrained weights
  - Instead of placing in a new module
  - The matrix to be added should be low-rank
  - Intuition: the weight matrices already live close to a low-rank manifold



- Transformers, initially applied only to a Attention weight matrices
- Now everywhere

Hu et al '22



Break & Questions

Outline

- **Fine-Tuning and Adapter Intro**
  - Fine-tuning vs. prompting, linear probing, etc. Full vs partial fine tuning vs adapting. Popular adapters
- **Cross-Modal Adaptation**
  - Frozen transformers, ORCA, aligning via optimal transport dataset distance
- **Model Editing**
  - Idea, MEND

What About Other Modalities?

So far, mostly talked about language models.

- Suppose we want tasks that are not directly language-based
- Could just train a new model… but harder

Can we adapt language models? Lots of c**hallenges:**

- Must change data types
- How do we know modalities are usable together?

Cross-Modal: **FPTs**

Frozen language-pretrained transformers (Lu et al '21)

Basic idea:

- Change the **input/output layers** (here, linear)
- Layer norm parameters
- Everything else frozen

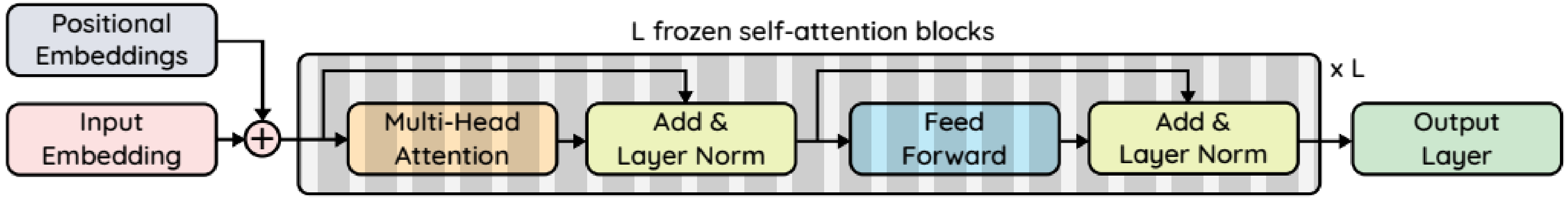


Figure 2: Frozen Pretrained Transformer (FPT). The self-attention & feedforward layers are frozen.

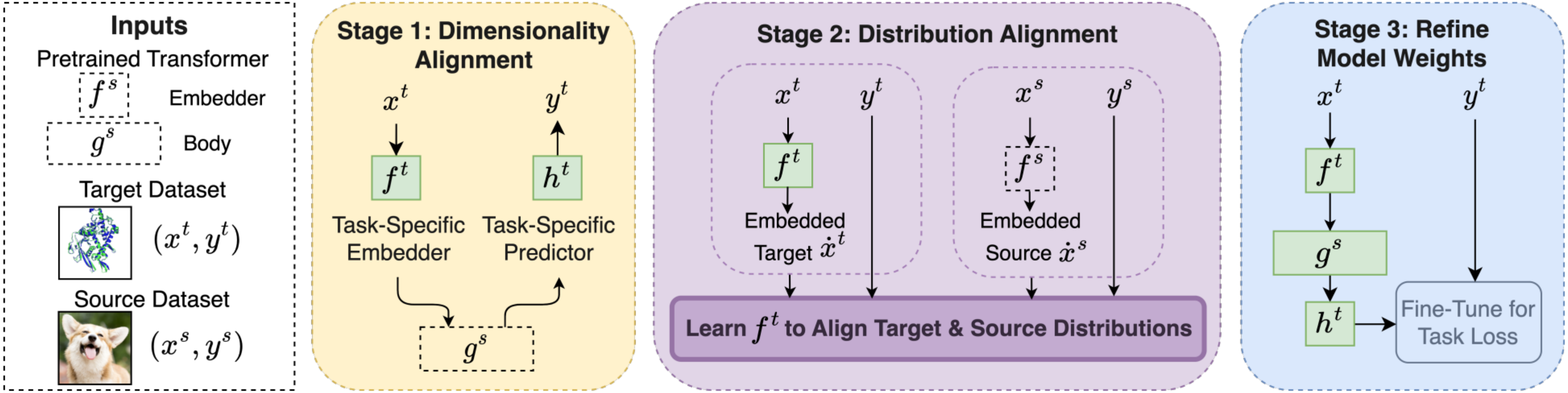
Lu et al, 21

Cross-Modal: **ORCA**

Performance bottleneck in FPTs

A more powerful approach: ORCA (Shen et al ‘23)

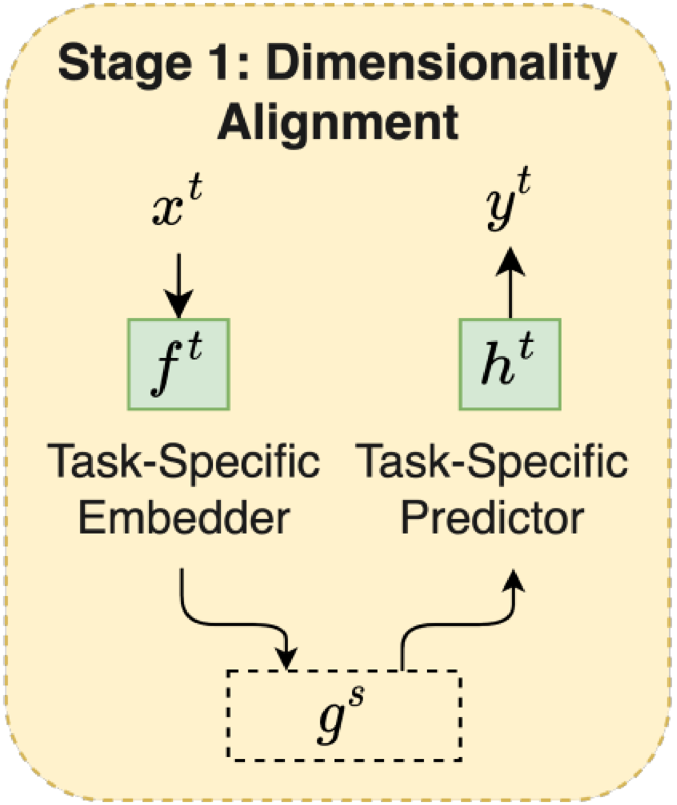
- Adds: distribution alignment step (align then refine)



ORCA: **Stage 1**

Let’s understand each stage of ORCA

- Stage 1: compatibility for inputs and outputs
- Custom input and output embedders that depend on the task
  - Input example: convolutional layers for image settings
  - Output example: average pooling+linear layer for classification



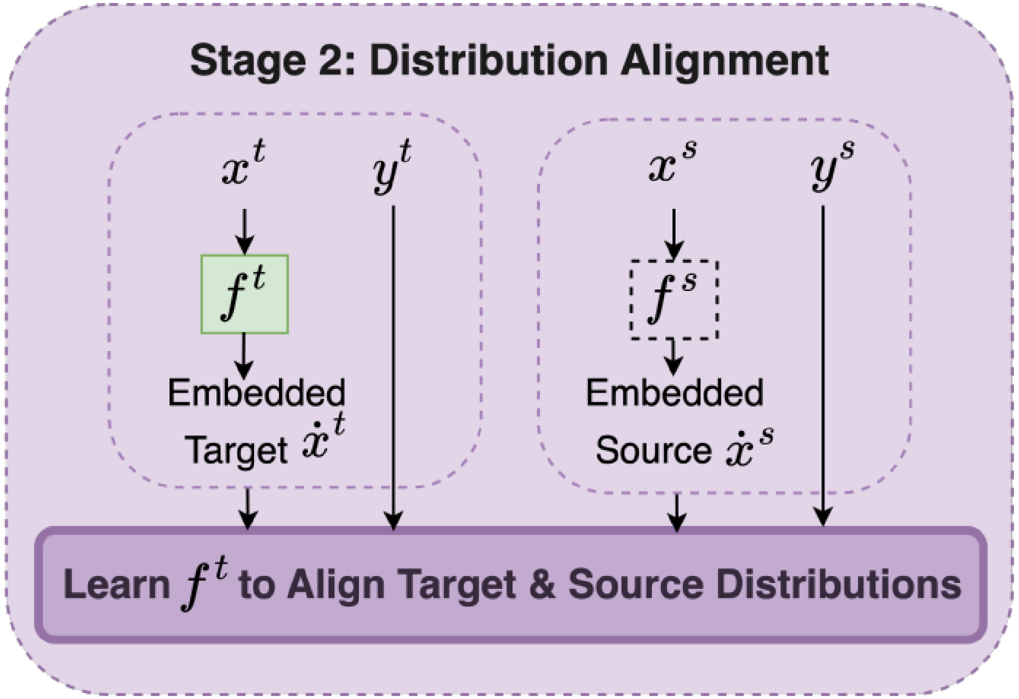
ORCA: **Stage 2**

Let's understand each stage of ORCA

- Stage 2: distribution alignment
- Intuition:
  - Change embeddings so target features **resemble** source features

- Learn the function f^t **that minimizes distance between**

  $(f^t(x^t), y^t)$ and $(f^s(x^s), y^s)$

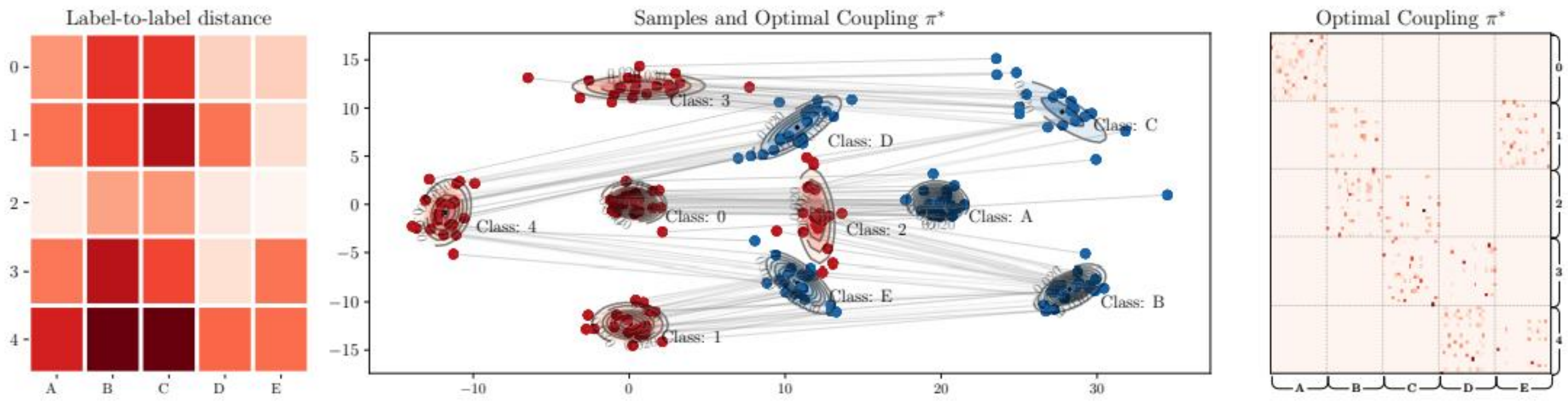


ORCA: **Distributional Distances**

Want: learn the function f^t **that minimizes distance between**

$(f^t(x^t), y^t)$ and $(f^s(x^s), y^s)$

- How?
- Need a distance function on these distributions
- Here, **optimal transport dataset distance** (OTDD)



Interlude: **Optimal Transport**

In optimal transport, we solve

$$\inf\left\{\int_{X\times Y} c(x,y)\,\mathrm{d}\gamma(x,y)\,\middle|\,\gamma\in\Gamma(\mu,\nu)\right\},$$

Cost or **distance** of moving x to y

The two **marginals** we care about, i.e., on x and y

- Want to “move” distribution on *x* to one on *y*
  - Output is a joint distribution with the original source and target
- But there’s a cost to moving x to y, given by c(x,y)

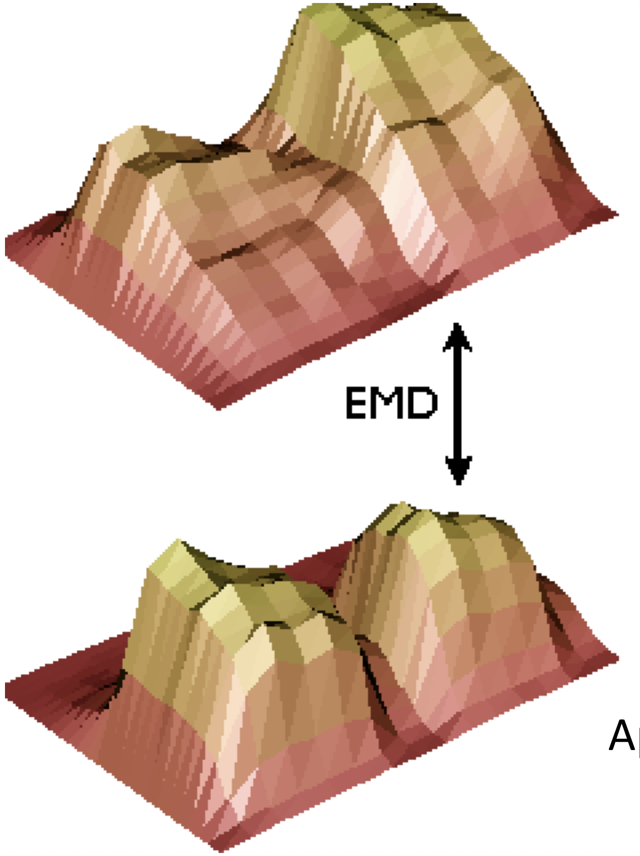
Interlude: **Optimal Transport**

In optimal transport, we solve

$$\inf\left\{\int_{X\times Y} c(x,y)\,\mathrm{d}\gamma(x,y)\,\middle|\,\gamma\in\Gamma(\mu,\nu)\right\},$$

Cost or **distance** of moving x to y

The two **marginals** we care about, i.e., on x and y



Applegate et al ‘11

Interlude: **Optimal Transport**

In optimal transport, we solve

$$\inf\left\{\int_{X\times Y} c(x,y)\,\mathrm{d}\gamma(x,y) \,\middle|\, \gamma\in\Gamma(\mu,\nu)\right\},$$

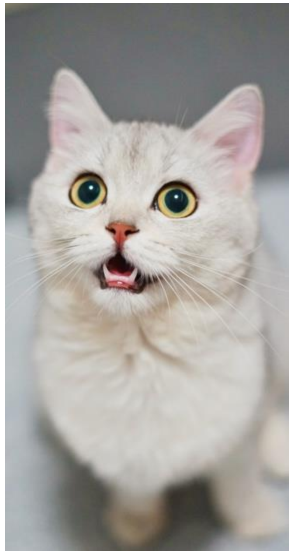
- Cost given by **distance**: Wasserstein distance
- Gives a distance on distributions, i.e.,

$$W_p(\mu,\nu) = \left(\inf_{\gamma\in\Gamma(\mu,\nu)} \mathbf{E}_{(x,y)\sim\gamma} d(x,y)^p\right)^{1/p}$$

Interlude: **Dataset Distance**

What should this cost/distance c(x,y) be for us?

- For inputs x, pretty easy: feature vectors in spaces that have distances, e.g., $||x-x'||$
- For outputs y, not so easy



–

- A clever idea:
  - Replace y with $P(X|y)$
- Even harder? No, just use Wasserstein: $W(P(X|y),P(X|y'))$
  - Approximate this with a Gaussian: closed form too!

ORCA: **Distributional Distances**

Want: learn the function f^t **that minimizes distance between** $(f^t(x^t), y^t)$ and $(f^s(x^s), y^s)$

- Need a distance function on these distributions
- Here, **optimal transport dataset distance** (OTDD)

$$d_{\mathcal{Z}}\big((x,y),(x',y')\big) \triangleq \big(d_{\mathcal{X}}(x,x')^p + \mathbf{W}_p^p(\alpha_y,\alpha_{y'})\big)^{1/p}$$

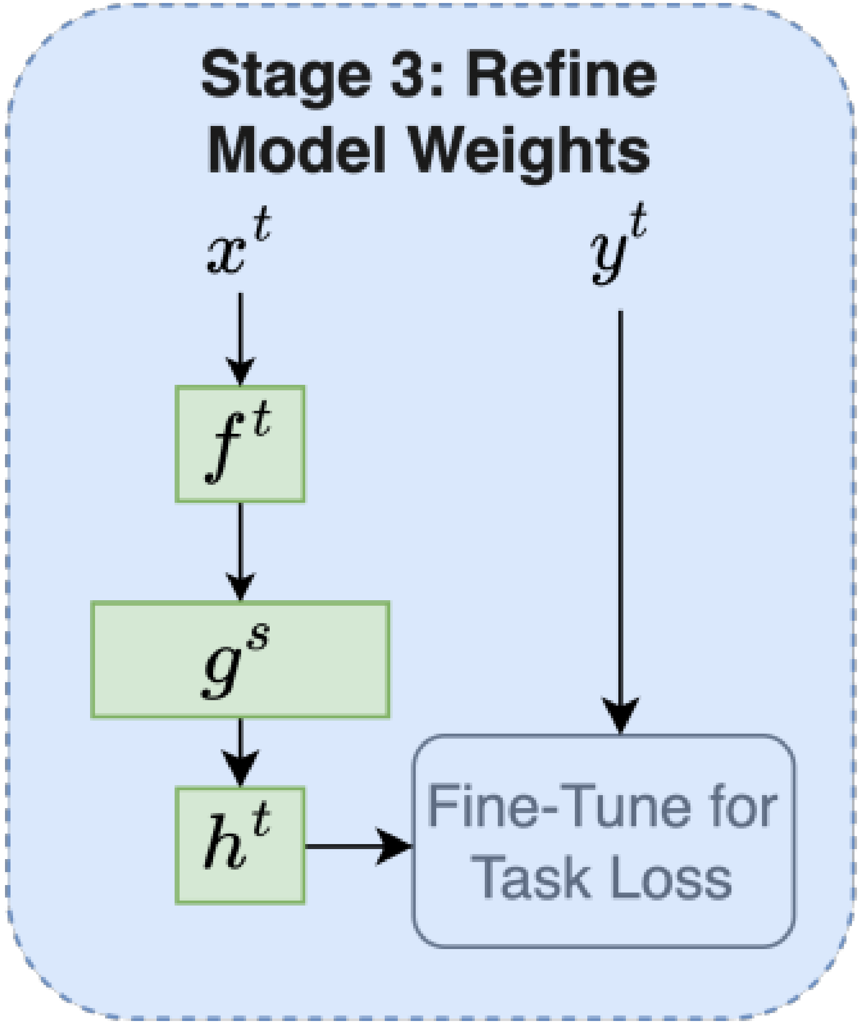
i.e., Euclidean distance

p-**Wasserstein distance** on P(x|y)

ORCA: **Stage 3**

Let's understand each stage of ORCA

- Stage 3: fine-tune the input and output network weights
  - For particular tasks
  - Or, could do any other variant of what we've talked about…



ORCA: **Results**

Extremely good, even against state-of-the-art results

- Compare to Neural Architecture Search (NAS)
  - Produces custom architectures that hit sota for various tasks
  - Same procedure on many types of tasks works well:

	CIFAR-100 0-1 error (%)	Spherical 0-1 error (%)	Darcy Flow relative ℓ_2	PSICOV MAE_8	Cosmic 1-AUROC	NinaPro 0-1 error (%)	FSD50K 1- mAP	ECG 1 - F1 score	Satellite 0-1 error (%)	DeepSEA 1- AUROC
Hand-designed	19.39	67.41	8E-3	3.35	**0.127**	8.73	0.62	**0.28**	19.80	0.30
NAS-Bench-360	23.39	48.23	2.6E-2	2.94	0.229	7.34	0.60	0.34	12.51	0.32
DASH	24.37	71.28	7.9E-3	3.30	0.19	**6.60**	0.60	0.32	12.28	**0.28**
Perceiver IO	70.04	82.57	2.4E-2	8.06	0.485	22.22	0.72	0.66	15.93	0.38
FPT	10.11	76.38	2.1E-2	4.66	0.233	15.69	0.67	0.50	20.83	0.37
ORCA	**6.53**	**29.85**	**7.28E-3**	**1.91**	0.152	7.54	**0.56**	**0.28**	**11.59**	0.29

Break & Questions

Outline

- **Fine-Tuning and Adapter Intro**
  - Fine-tuning vs. prompting, linear probing, etc. Full vs partial fine tuning vs adapting. Popular adapters
- **Cross-Modal Adaptation**
  - Frozen transformers, ORCA, aligning via optimal transport dataset distance
- **Model Editing**
  - Idea, MEND

Model Editing

So far, adapting to new tasks

- But what if we just want to change the model?

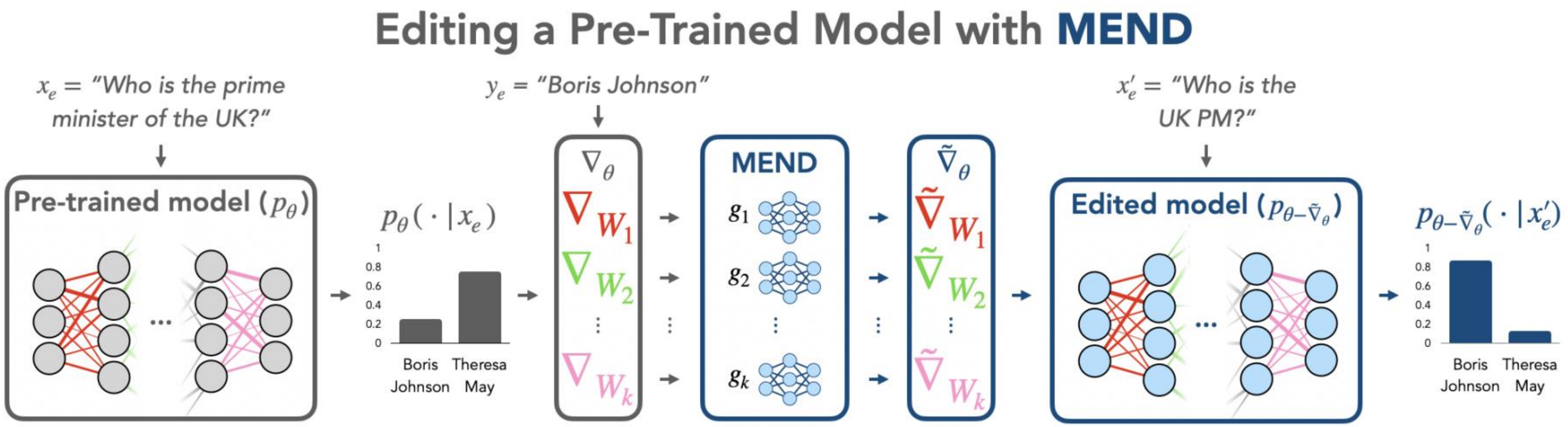
Why?

- Models have outdated (or wrong!) information in them
- Need to update these facts… but fine-tuning on just one point can be hard
  - Overfit to the point
  - May change other aspects

Model Editing: **MEND**

Fast editing with Model Editor Networks with Gradient Decomposition (MEND)

- Mitchell et al '22



Bibliography

- Sprague et al ‘24: Zayne Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, Greg Durrett “To CoT or not to CoT? Chain-of-thought helps mainly on math and symbolic reasoning” (https://arxiv.org/abs/2409.12183)
- Gao et al ’23: Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, Graham Neubig, “PAL: Program-aided Language Models” (https://arxiv.org/abs/2211.10435)
- Chen et al ‘22: Wenhu Chen, Xueguang Ma, Xinyi Wang, William W. Cohen, “Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks” (https://arxiv.org/abs/2211.12588)
- Schick et al ’23: Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, Thomas Scialom, “Toolformer: Language Models Can Teach Themselves to Use Tools” (https://arxiv.org/abs/2302.04761)
- Liu et al ‘22, Haokun Liu, Derek Tam, Muqeeth Mohammed, Jay Mohta, Tenghao Huang, Mohit Bansal, Colin Raffel, “Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning”. (https://openreview.net/forum?id=rBCvMG-JsPd)
- Kumar et al ’22, Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, Percy Liang, “Fine-Tuning can Distort Pretrained Features and Underperform Out-of-Distribution” (https://openreview.net/pdf?id=UYneFzXSJWh)
- Shen et al ’21, Zhiqiang Shen1, Zechun Liu, Jie Qin, Marios Savvides, and Kwang-Ting Cheng, “Partial Is Better Than All: Revisiting Fine-tuning Strategy for Few-shot Learning”. (https://arxiv.org/pdf/2102.03983.pdf)
- Li and Liang ‘21, Lisa Li and Percy Liang, “Prefix-Tuning: Optimizing Continuous Prompts for Generation” (https://arxiv.org/abs/2101.00190)
- Houlsby et al ’19, Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, Sylvain Gelly, “Parameter-Efficient Transfer Learning for NLP” (https://arxiv.org/abs/1902.00751)

Thank You!