CS 839: Foundation Models
Training

Fred Sala

University of Wisconsin-Madison

Oct. 12, 2023

Announcements

- **Logistics:**
  - Homework 1 deadline **today**!
  - Presentation information out:
    https://pages.cs.wisc.edu/~fredsala/cs839/fall2023/files/presentation_info.pdf
- Class roadmap:

Thursday Oct. 12	Training, Start RLHF
Tuesday Oct. 17	RLHF
Thursday Oct. 19	Data
Tuesday Oct. 24	Multimodal and Specialized Foundation Models
Thursday Oct. 26	Knowledge

Outline

- **Finishing Up Last Time**
  - Fine-tuning, adapting, cross-modal alignment methods, model editing
- **Training**
  - Scale, parallelization, memory optimization, heterogenous training
- **Reinforcement Learning From Human Feedback**
  - Basic idea, goals, mechanisms

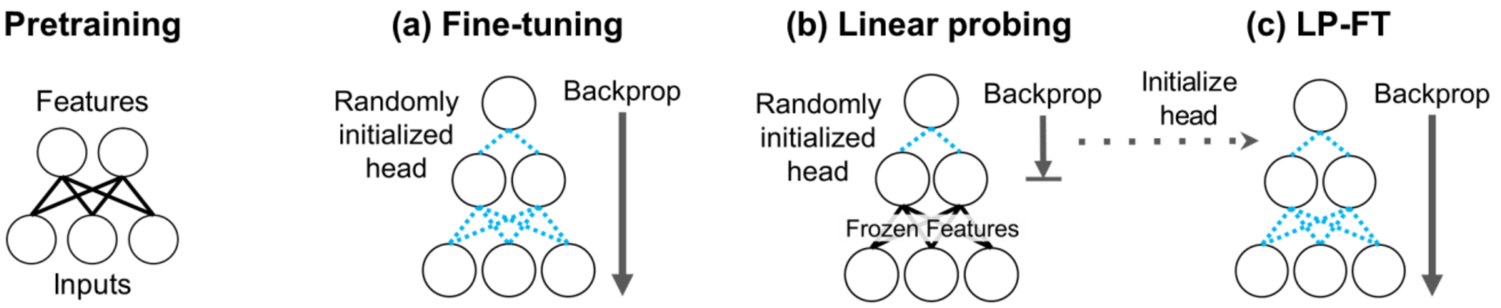
Outline

- **Finishing Up Last Time**
  - Fine-tuning, adapting, cross-modal alignment methods, model editing
- **Training**
  - Scale, parallelization, memory optimization, heterogenous training
- **Reinforcement Learning From Human Feedback**
  - Basic idea, goals, mechanisms

Last time: **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



	(a) Fine-tuning	(b) Linear probing	(c) LP-FT
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

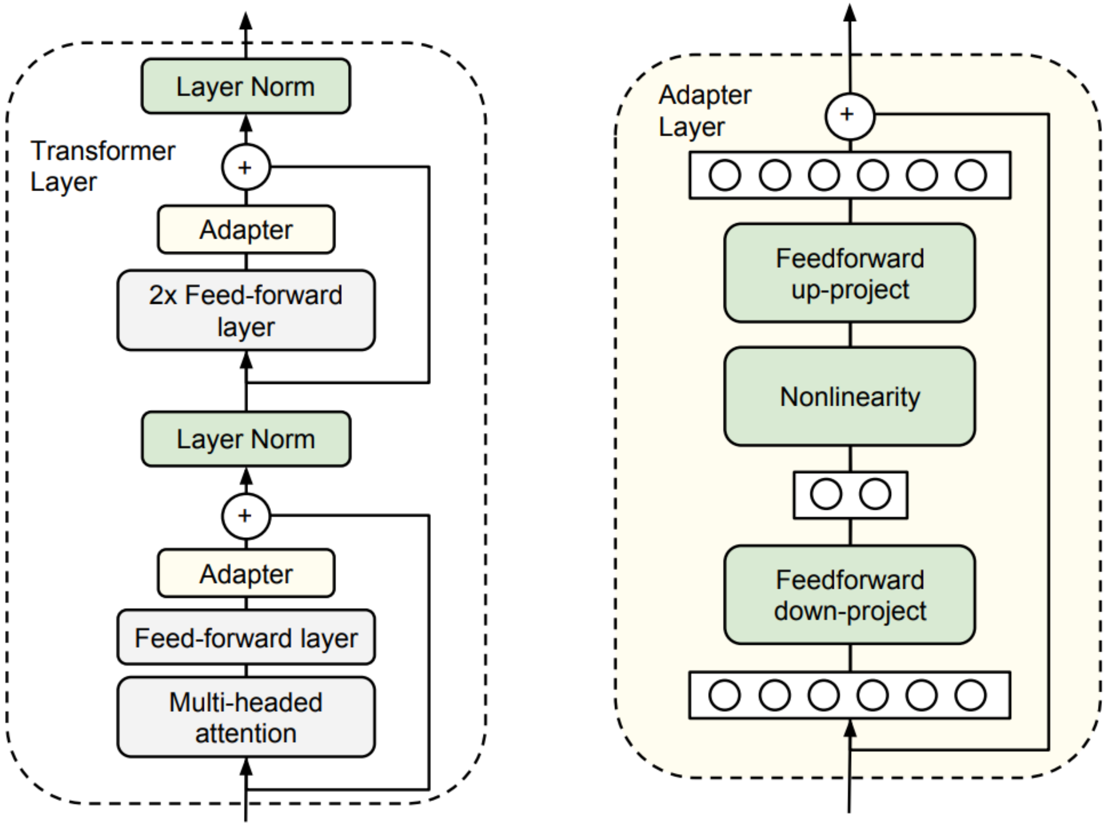
Last time: PEFT: **Adapters**

Want two things in parameter-efficient fine-tuning

- 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



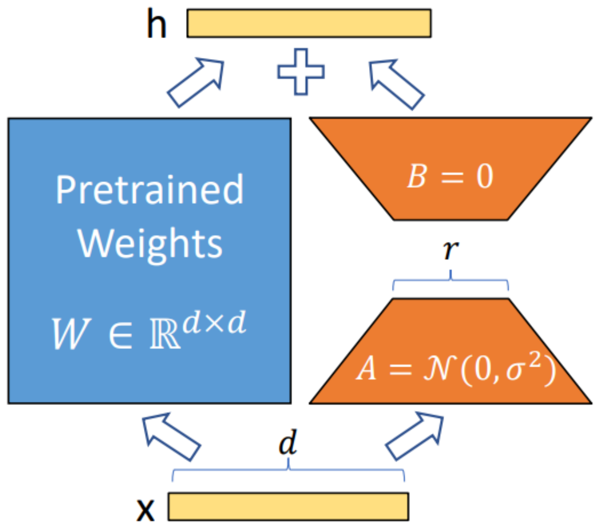
Houlsby et al ‘19

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

Perhaps the most popular variant

- LoRA makes an assumption on adapter layer structure
  - Specifically, should be low-rank
  - Intuition: the weight matrices already live close to a low-rank manifold

- Transformers, apply only to attention weight matrices



Hu et al '22

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

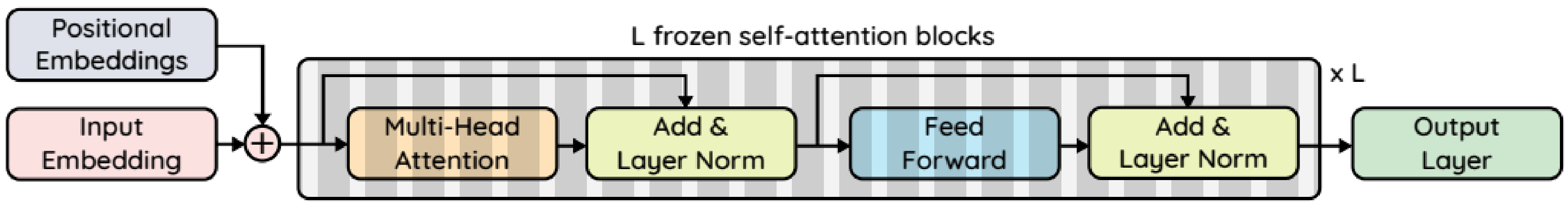


Figure 2: Frozen Pretrained Transformer (FPT). The self-attention & feedforward layers are frozen.

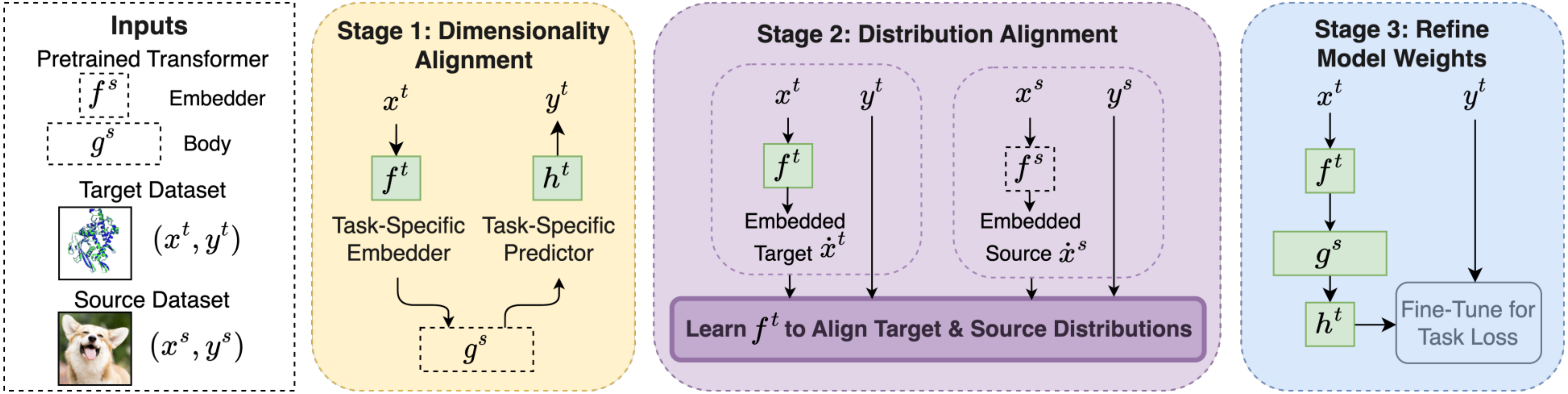
Lu et al, 21

Cross-Modal: **ORCA**

Performance bottleneck in FPTs

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

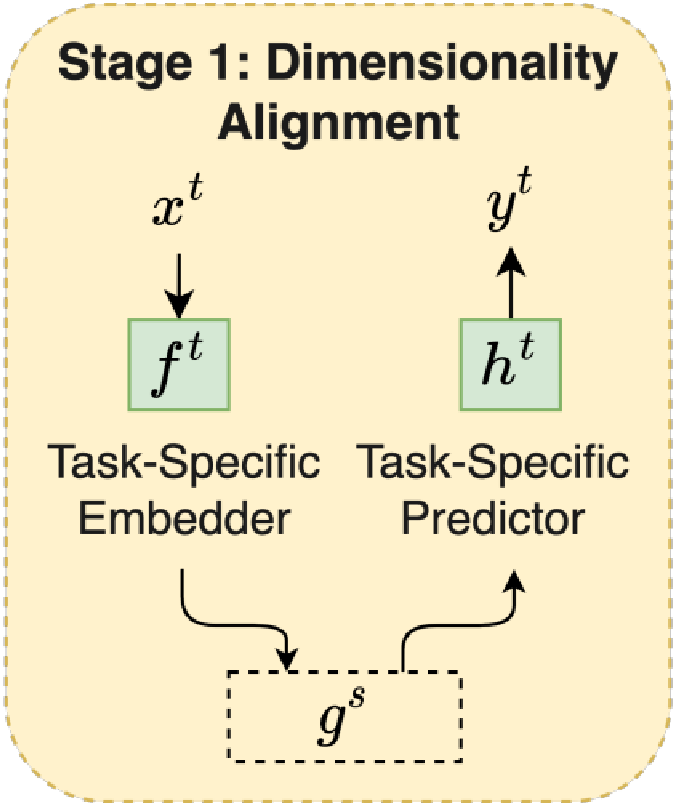
- Adds: distribution alignment step (align then refine)



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



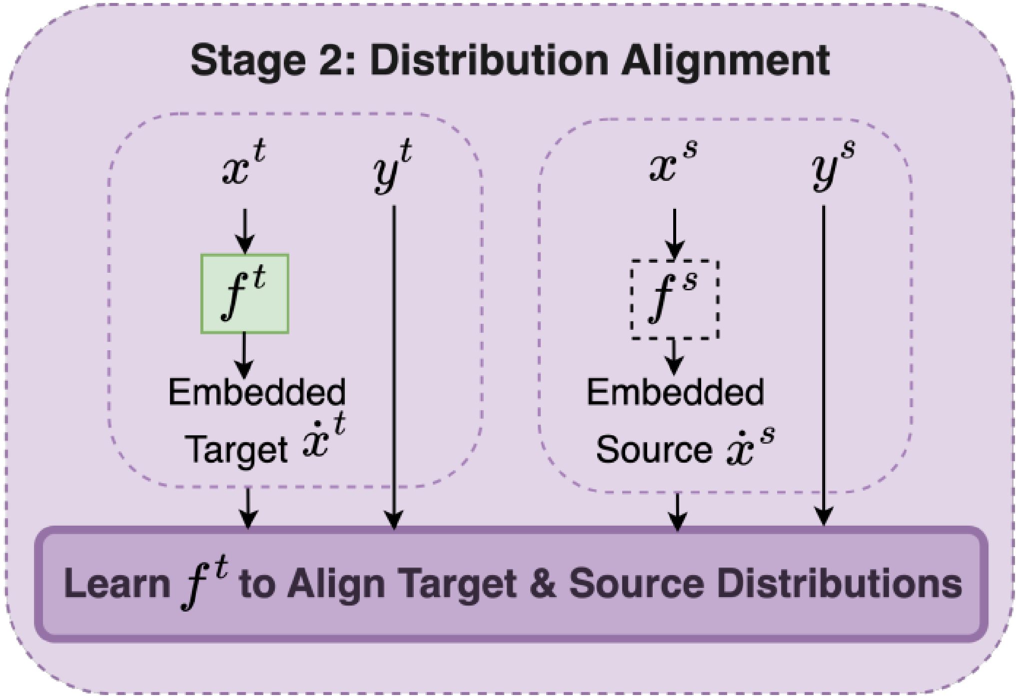
ORCA: **Stage 2**

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**

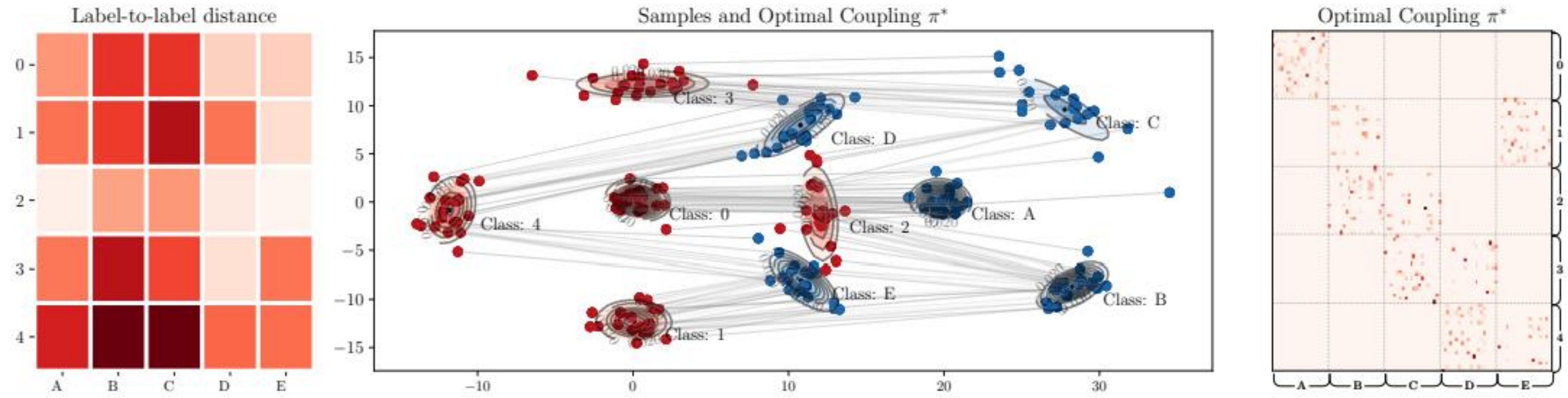
  $(f^t(x^t), y^t)$ and $(f^s(x^s), y^s)$



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
- Let's use the **optimal transport dataset distance** (OTDD)



Alvarez-Melis and Fusi, '20

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

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

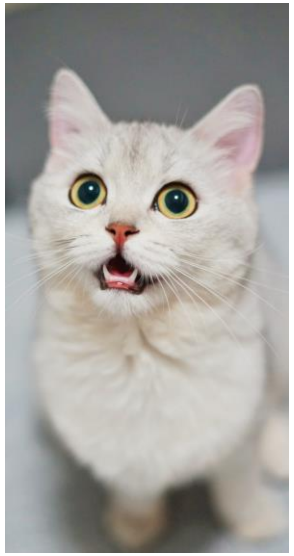
- Cost given by **distance**: Wasserstein distance
- Gives a distance on distributions, i.e.,

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

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)

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

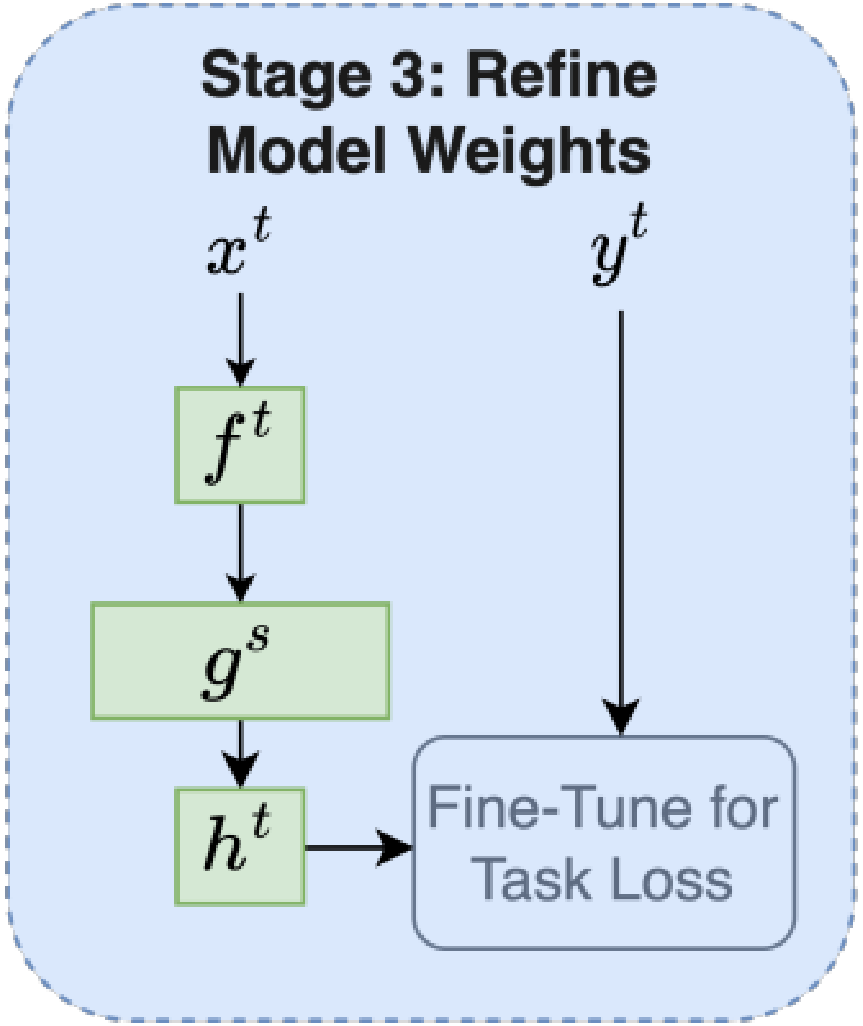
i.e., Euclidean distance

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

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

Break & Questions

Outline

- **Finishing Up Last Time**
  - Fine-tuning, adapting, cross-modal alignment methods, model editing
- **Training**
  - Scale, parallelization, memory optimization, heterogenous training
- **Reinforcement Learning From Human Feedback**
  - Basic idea, goals, mechanisms

Training Foundation Models: **Scale**

Llama family of models,

- "*we estimate that we used 2048 A100-80GB for a period of approximately 5 months to develop our models*"

OPT (Open Pre-trained Transformers),

- "*training OPT-175B on 992 80GB A100 GPUs*"

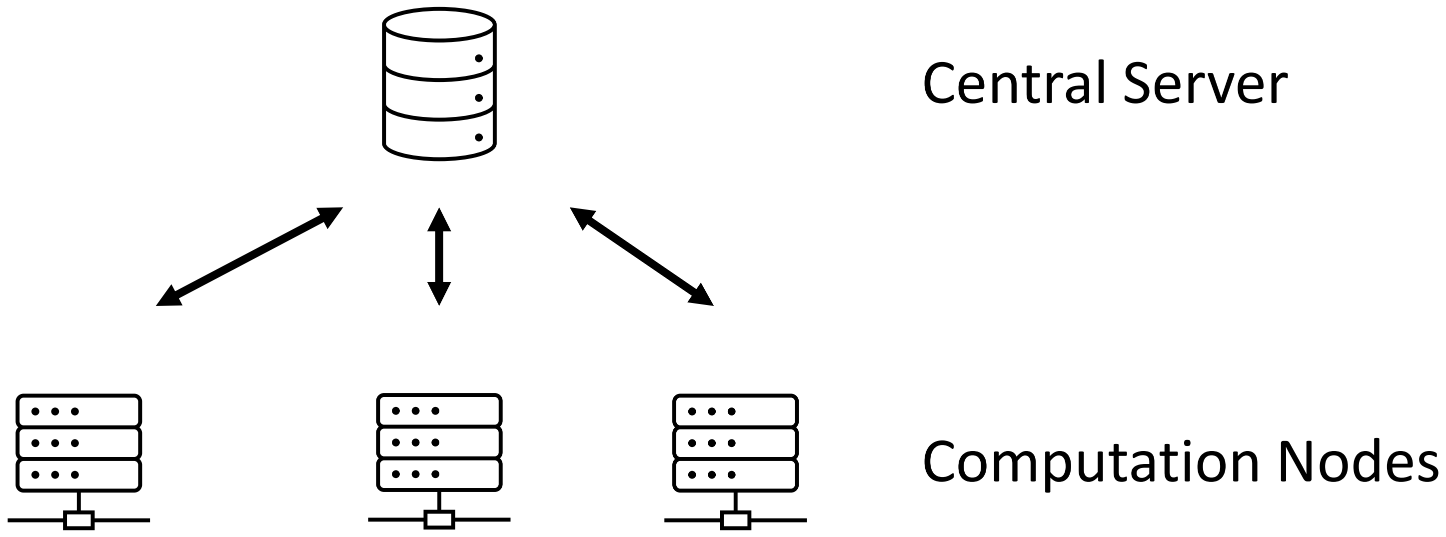
	GPU Type	GPU Power consumption	GPU-hours	Total power consumption	Carbon emitted (tCO_2eq)
OPT-175B	A100-80GB	400W	809,472	356 MWh	137
BLOOM-175B	A100-80GB	400W	1,082,880	475 MWh	183
LLaMA-7B	A100-80GB	400W	82,432	36 MWh	14
LLaMA-13B	A100-80GB	400W	135,168	59 MWh	23
LLaMA-33B	A100-80GB	400W	530,432	233 MWh	90
LLaMA-65B	A100-80GB	400W	1,022,362	449 MWh	173

Touvron et al, 23

Training Foundation Models: **Parallelization**

Traditional approach is to **distribute** training loads

- Classic centralized distributed training
  - Synchronize each local gradient update
  - Send synchronized vector back to each node (lots of communication!)



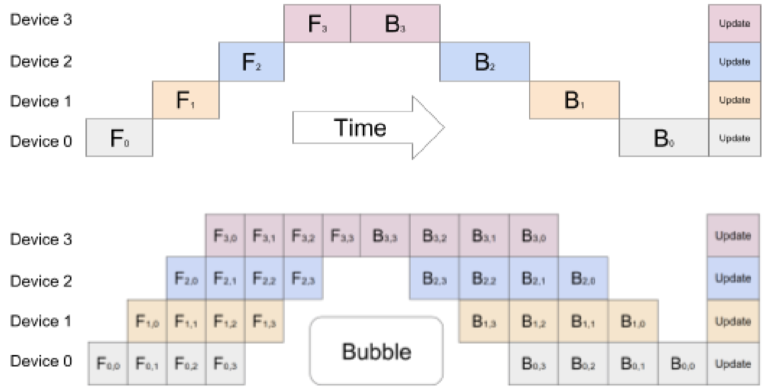
Training Foundation Models: **Parallelization**

Traditional approach is to **distribute** training loads

- This is by itself impossible (each node *can't* handle full model for large models)
- Need further parallelism:
  - **Data**: each node sees a different slice of data
  - **Weights/tensors**: chunks so no GPU sees whole model
  - **Pipeline**: only a few layers per GPU

- Great resource:

https://huggingface.co/blog/bloom-megatron-deepspeed

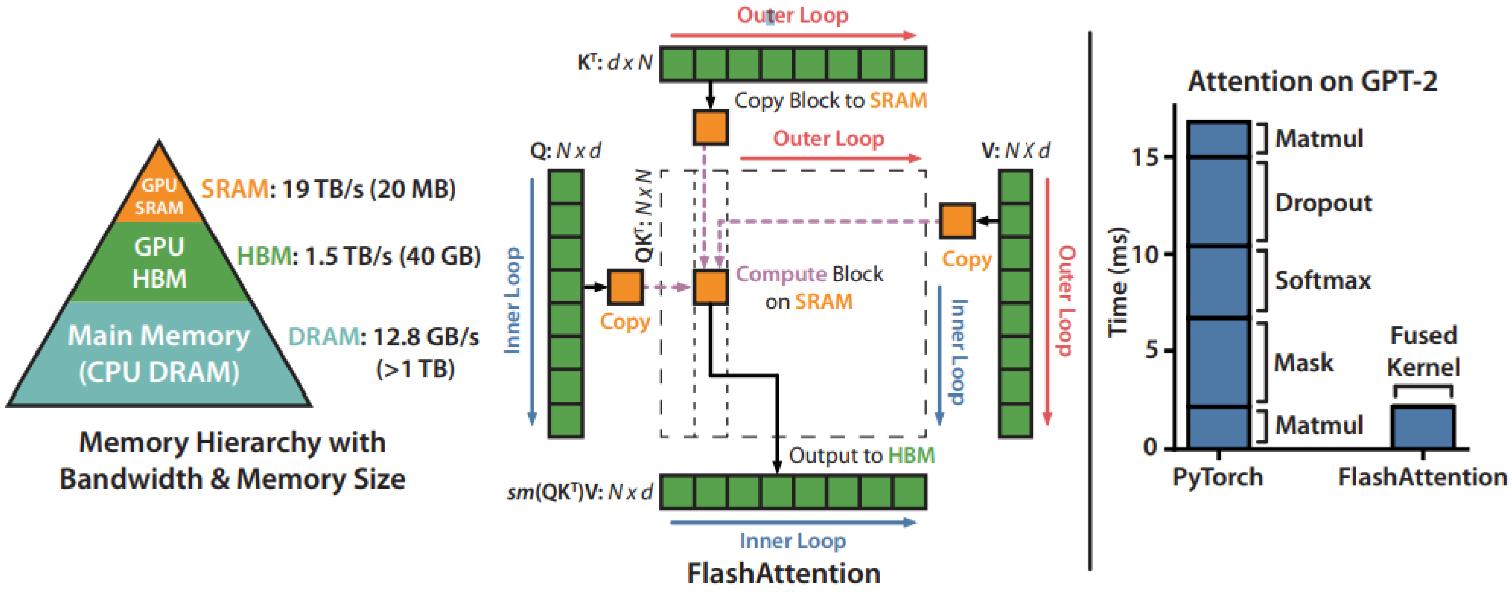


Top: *The naive model parallelism strategy leads to severe underutilization due to the sequential nature of the network. Only one accelerator is active at a time.* ***Bottom:*** *GPipe divides the input mini-batch into smaller micro-batches, enabling different accelerators to work on separate micro-batches at the same time.*

Training Foundation Models: **GPU Usage**

Even for each GPU, there's additional considerations

- A little bit of fast memory, lots of slower memory
- Avoid using slow memory when possible
  - FlashAttention: Tiling + computing tricks



Dao et al '22

Break & Questions

Outline

- **Finishing Up Last Time**
    - Fine-tuning, adapting, cross-modal alignment methods, model editing
- **Training**
    - Scale, parallelization, memory optimization, heterogenous training
- **Reinforcement Learning From Human Feedback**
    - Basic idea, goals, mechanisms

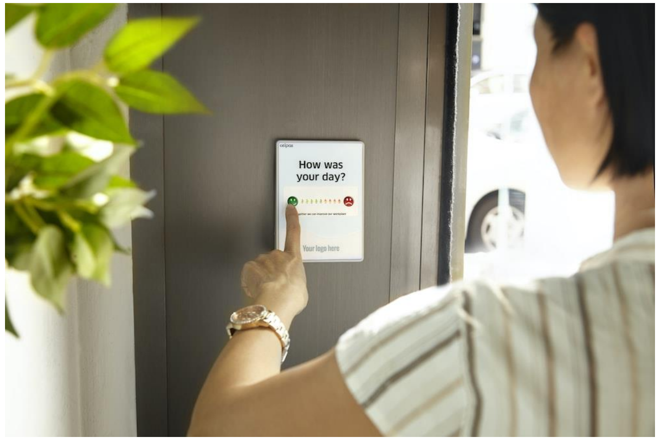
RLHF: **Basic Motivation**

Goal: produce language model outputs that users like better…

- **Hard** to specify exactly what this means,
- **Easy** to query users

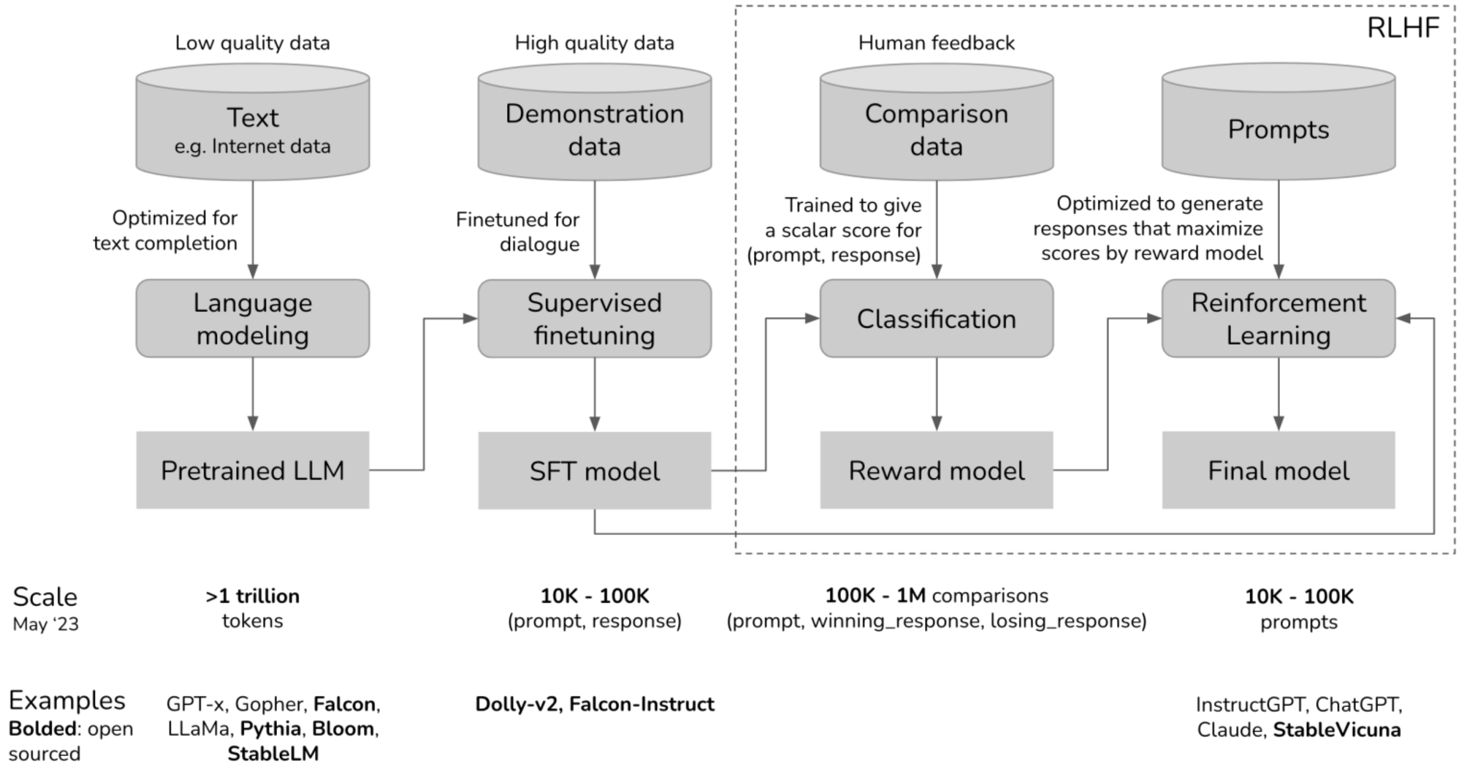
Collect human feedback and use it to change the model

- Can do this by fine-tuning, especially with instructions
- Doesn’t quite capture what users want



RLHF: **Setup**

Goal: produce language model outputs that users like better…



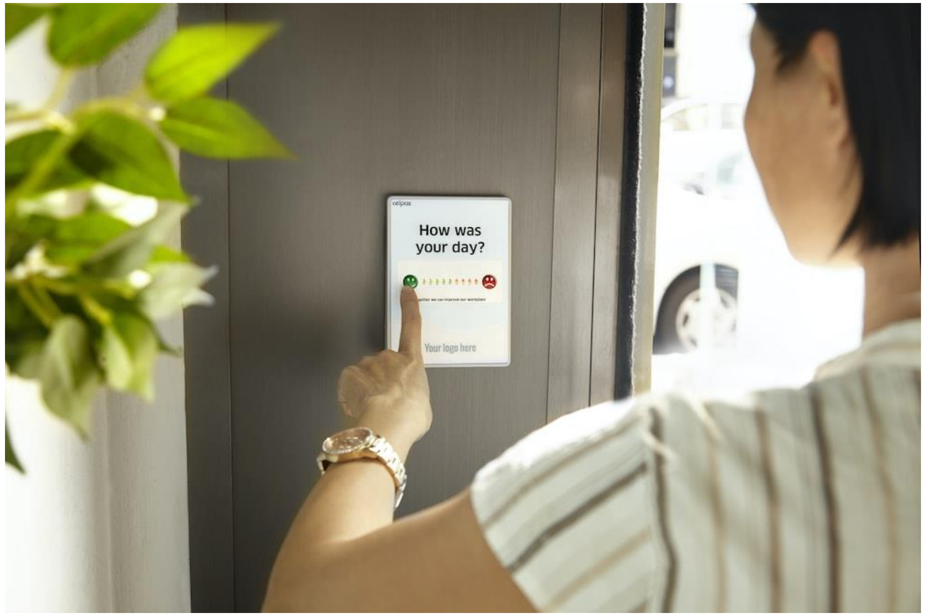
Chip Huyen

RLHF: **Feedback**

First stage: get **human feedback** to train reward model

- Fix a set of prompts
- Take two language models and produce outputs for each prompt

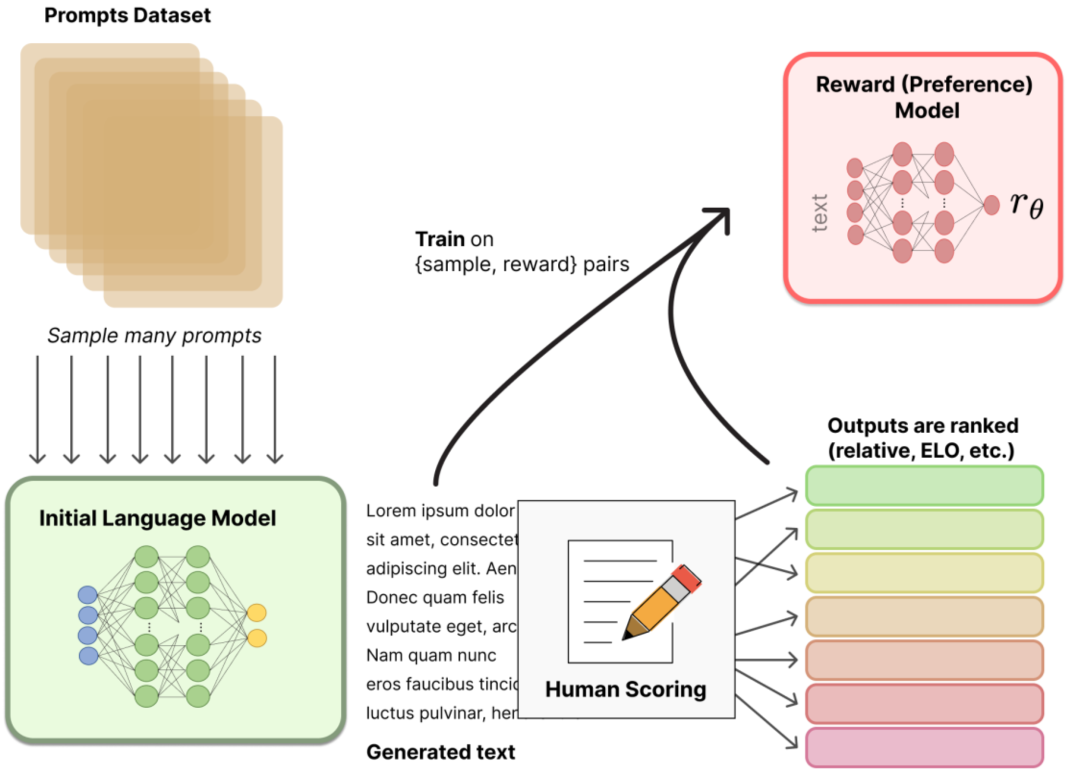
- Ask human users **which is better**
  - **Binary output**



RLHF: **Reward/Preference Model**

Second stage: train reward model

- Use the human feedback to train/fine-tune another model to reproduce the metric
- **Preference model**



https://huggingface.co/blog/rlhf

RLHF: **Fine-Tuning with RL**

Third stage: RL

- Use an RL algorithm
- **Goal:** produce outputs that have high reward

RL formulation:

- **Action space**: all the tokens possible to output
- **State space**: all the sequences of tokens
- **Reward function**: the trained model (some variations)
- **Policy**: the new version of the LM, taking in prompts and returning output

Bibliography

- 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)
- 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)
- Hu et al ’22: Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, “LoRA: Low-Rank Adaptation of Large Language Models” (https://arxiv.org/abs/2106.09685)
- Lu et al ’21: Kevin Lu, Aditya Grover, Pieter Abbeel, Igor Mordatch , “Pretrained Transformers as Universal Computation Engines” (https://arxiv.org/abs/2103.05247)
- Shen et al ’23: Junhong Shen, Liam Li, Lucio M. Dery, Corey Staten, Mikhail Khodak, Graham Neubig, Ameet Talwalkar, “Cross-Modal Fine-Tuning: Align then Refine” (https://arxiv.org/abs/2302.05738)
- Alvarez-Melis and Fusi, ’20: David Alvarez-Melis, Nicolo Fusi, “Geometric Dataset Distances via Optimal Transport” (https://arxiv.org/abs/2002.02923)
- Touvron et al ‘23: Hugo Touvron and many others, “LLaMA: Open and Efficient Foundation Language Models” (https://arxiv.org/abs/2302.13971)
- Stas Bekman: https://huggingface.co/blog/bloom-megatron-deepspeed
- Dao et al ‘22: Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, Christopher Ré, “FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness” (https://arxiv.org/abs/2205.14135)
- Chip Huyen: https://huyenchip.com/2023/05/02/rlhf.html
- Nathan Lambert et al: https://huggingface.co/blog/rlhf

Thank You!