Learning from Preferences

CSE 5525: Foundations of Speech and Natural Language Processing

https://shocheen.github.io/courses/cse-5525-fall-2025

THE OHIO STATE UNIVERSITY

Slide Credits: Yejin Choi, Yoav Artzi

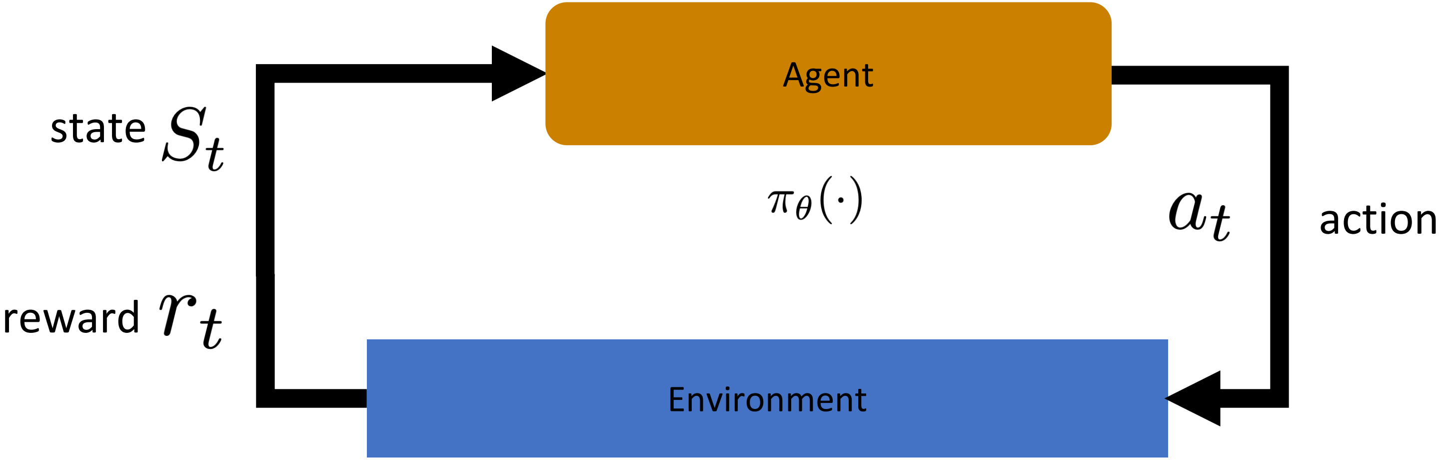
Logistics

- Hw3 questions?

Aligning LLMs

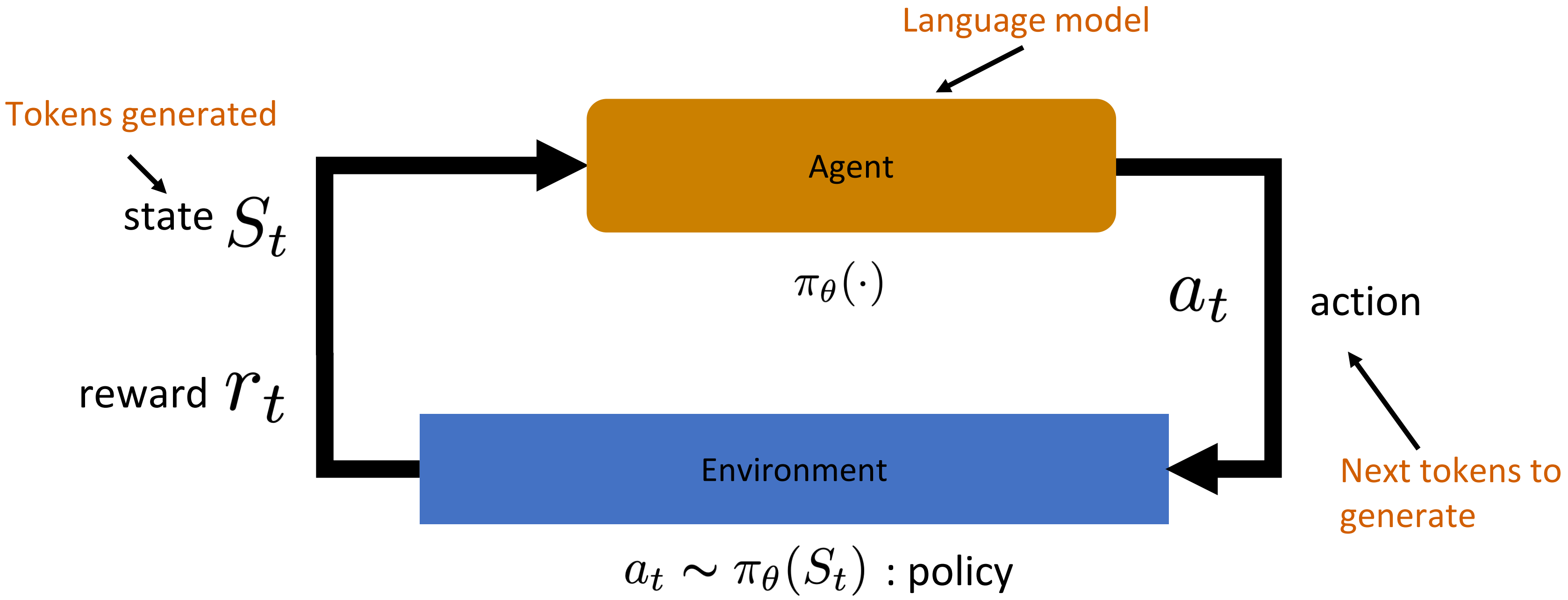
- Goal: turn LLMs from text generators to models that can follow specific instructions and are relatively controlled
- Two independent techniques
    - Supervised: learn from annotated data/demonstration
    - Reinforcement learning from human feedback: learn from preferences
- In practice: they are combined to a complete process

Reinforcement Learning Basics



$a_t \sim \pi_\theta(S_t)$: policy

RL in the Context of Language Models…



Goal of RL: Maximize the expected reward

$$\max_{\pi_\theta} \mathbb{E}_{x\sim D, y\sim\pi_\theta(y|x)}[r_\phi(x,y)]$$

Sampling **trajectories** from policy

Reward given prompt and sampled generation

How do we get the reward?

Human feedback

RLHF Data
Asking Humans

Score the helpfulness of the following response, 1-10

What are the steps for making a simple cake?

1. ***Warm up the oven.***
2. ***Grease a cake pan.***
3. ***Blend dry ingredients in a bowl.***
4. ***Incorporate butter, milk, and vanilla.***
5. ***Mix in the eggs.***
6. ***Pour into the prepared pan.***
7. ***Bake until golden brown.***
8. ***Add frosting if desired.***

[Example from Eric Mitchell]

RLHF Data
Asking Humans

Score the helpfulness of the following response, 1-10

What are the steps for making a simple cake?

1.Preheat oven to 350°F (175°C).

2.Grease and flour a cake pan.

3.In a bowl, combine 2 cups flour, 1.5 cups sugar, 3.5 tsp baking powder, and a pinch of salt.

4.Add 1/2 cup butter, 1 cup milk, and 2 tsp vanilla; mix well.

5.Beat in 3 eggs, one at a time.

6.Pour batter into the pan.

7.Bake for 30-35 minutes or until a toothpick comes out clean.

8.Let cool, then frost or serve as desired.

[Example from Eric Mitchell]

RLHF Data

Asking Humans

- Humans are very inconsistent for complex evaluation like free-form text evaluation
  - This would give a very noisy learning signal 😔
- Especially when the outputs all look really good
- What can we do?

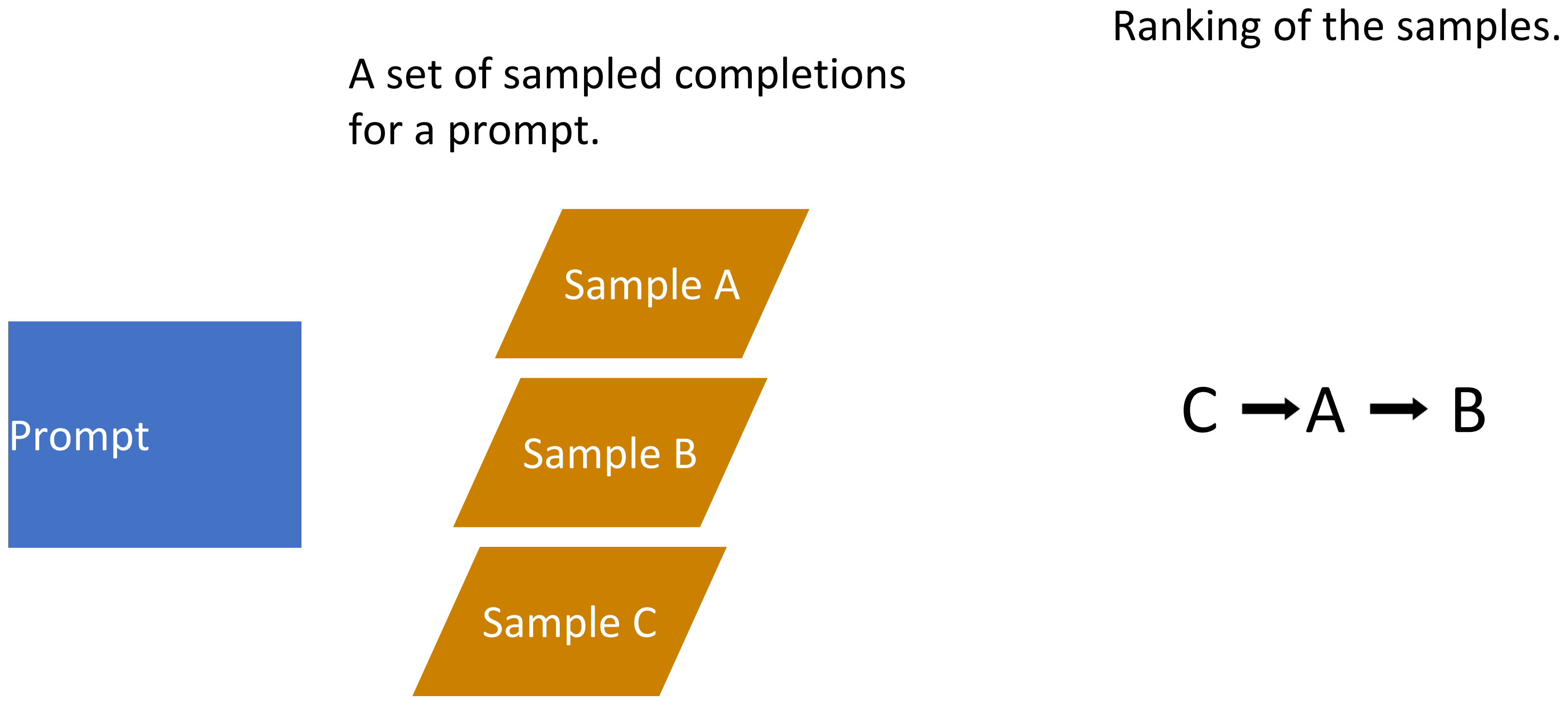
RLHF Data
Human Preferences

Which of these two responses is more helpful?

What are the steps for making a simple cake?	What are the steps for making a simple cake?
1.Preheat oven to 350°F (175°C).	*1. Warm up the oven.*
2.Grease and flour a cake pan.	*2. Grease a cake pan.*
3.In a bowl, combine 2 cups flour, 1.5 cups sugar, 3.5 tsp baking powder, and a pinch of salt.	*3. Blend dry ingredients in a bowl.*
4.Add 1/2 cup butter, 1 cup milk, and 2 tsp vanilla; mix well.	*4. Incorporate butter, milk, and vanilla.*
5.Beat in 3 eggs, one at a time.	*5. Mix in the eggs.*
6.Pour batter into the pan.	*6. Pour into the prepared pan.*
7.Bake for 30-35 minutes or until a toothpick comes out clean.	*7. Bake until golden brown.*
8.Let cool, then frost or serve as desired.	*8. Add frosting if desired.*

[Example from Eric Mitchell]

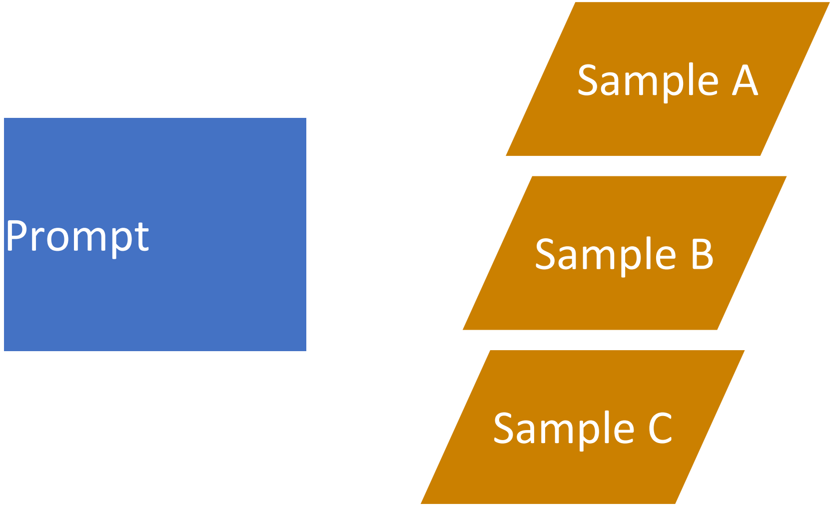
Asking to rank multiple answers is easier



Convert ranking to paired preferences

Triples

A set of sampled completions for a prompt.



$$D = \{x^i, y_w^i, y_l^i\}$$

Prompt

Preferred Response

Dispreferred Response

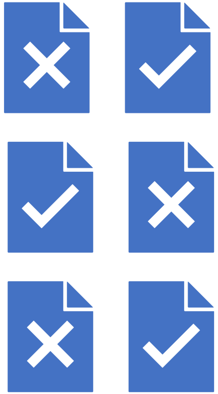
The general RLHF pipeline

1

Instruction-tuned Model

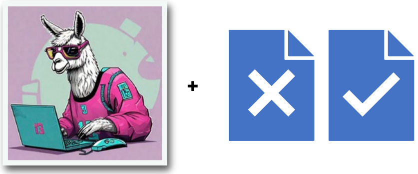
2

Collect Comparison Data



3

Train Reward (aka the scoring) Model on Comparison Data



4

Use RL to Optimize the instruction tuned LM with the reward model

Reward Modeling

Reward function

- Given the input x and a generate response y, the reward function gives a real valued output indicating how good the response is for the output
    - r(x, y)

- Goal of RLHF: Maximize expected reward of the model. High reward → better model.

- How to implement r: train a transformer model with a **regression head**
    - Take a pretrained LM, replace the final layer (hidden vector to vocabulary size) to a regression head (hidden vector to 1 dimension).
    - Finetune it to predict a "score"

How to predict scores: convert pairwise preferences to reward function: Bradley-Terry Model

$$D = \{x^i, y_w^i, y_l^i\}$$

Prompt

Preferred Response

Dispreferred Response

Reward for preferred response

Reward for dispreferred response

$$p(y_w > y_l | x) = \sigma(r(x, y_w) - r(x, y_l))$$

Sigmoid function: this is basically binary classification

$$\frac{1}{1+e^{-x}}$$

$$p(y_w > y_l | x) = \frac{\exp(r(x, y_w))}{\exp(r(x, y_w)) + \exp(r(x, y_l))}$$

Reward Model

$$p(y_w > y_l|x) = \frac{\exp(r(x, y_w))}{\exp(r(x, y_w)) + \exp(r(x, y_l))}$$

- Train on preference data.

- Minimizing negative log likelihood.

equivalent to

$$\mathcal{L}_R(\phi, D) = -\mathbb{E}_{(x, y_w, y_l) \sim D}[\log \sigma(r(x, y_w) - r(x, y_l))]$$

- Train an LLM with an additional layer to minimize the neg. log likelihood

Evaluating Reward Models

- Accuracy of predicting human preferences.

Preference Datasets

Reward Models

Table 2: Reward modeling accuracy (%) results. We compare our UltraRM with baseline open-source reward models. LLaMA2 results are taken from Touvron et al. (2023b). The highest results are in **bold** and the second highest scores are underlined.

Model	Backbone Model	Open?	Anthropic Helpful	OpenAI WebGPT	OpenAI Summ.	Stanford SHP	Avg.
Moss	LLaMA-7B	✓	61.3	54.6	58.1	54.6	57.2
Ziya	LLaMA-7B	✓	61.4	57.0	61.8	57.0	59.3
OASST	DeBERTa-v3-large	✓	67.6	-	72.1	53.9	-
SteamSHP	FLAN-T5-XL	✓	55.4	51.6	62.6	51.6	55.3
LLaMA2 Helpfulness	LLaMA2-70B	✗	**72.0**	-	**75.5**	**80.0**	-
UltraRM-UF	LLaMA2-13B	✓	66.7	65.1	66.8	68.4	66.8
UltraRM-Overall	LLaMA2-13B	✓	<u>71.0</u>	62.0	73.0	73.6	<u>69.9</u>
UltraRM	LLaMA2-13B	✓	<u>71.0</u>	**65.2**	<u>74.0</u>	<u>73.7</u>	**71.0**

Cui et al., ArXiV 2023 "UltraFeedback: Boosting Language Models with High-quality Feedback"

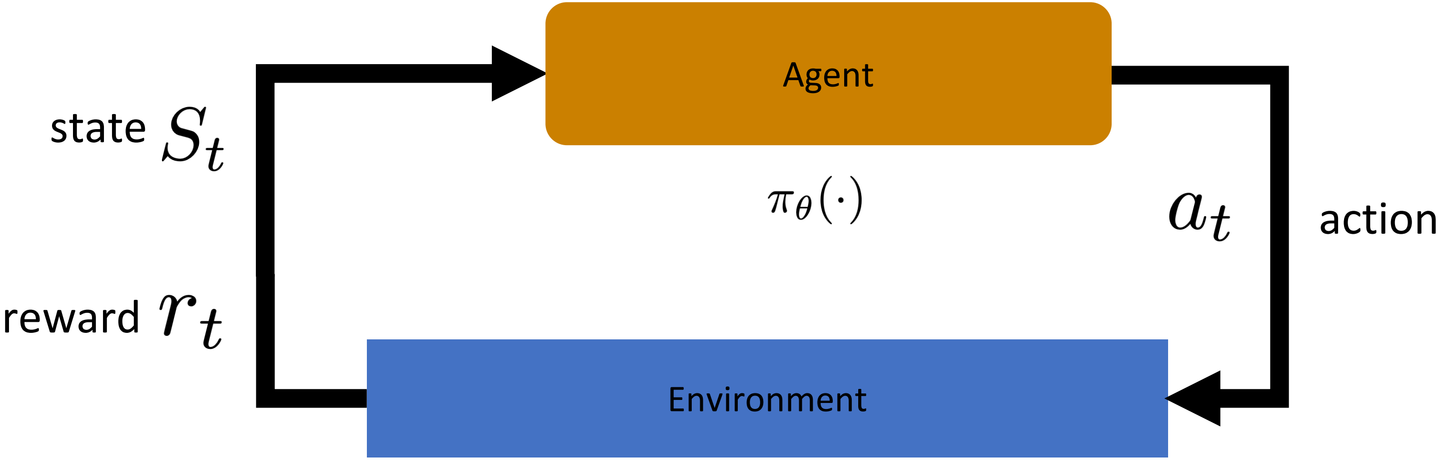
Fun Facts about Reward Models

- Trained for 1 epoch (to avoid overfitting)!
- Evaluation often only has 65% - 75% agreement

Lambert et al., 2023

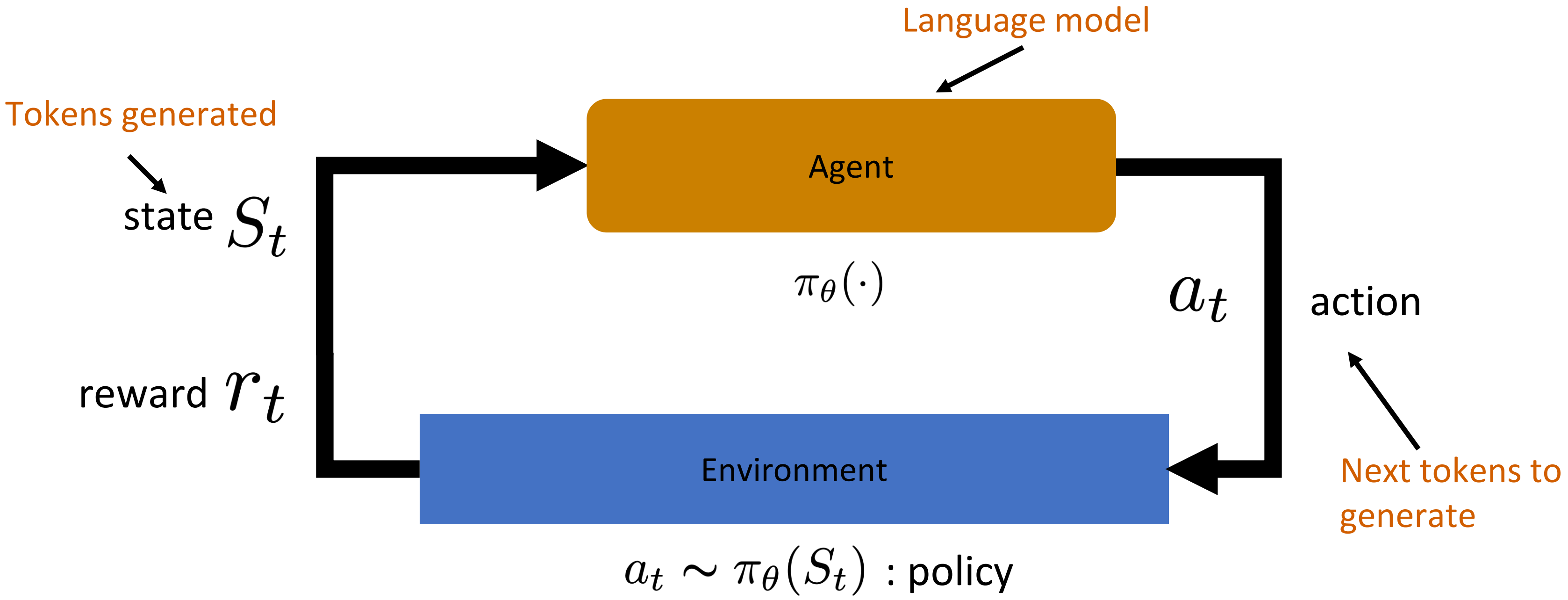
Basics of Reinforcement Learning

Reinforcement Learning Basics



$a_t \sim \pi_\theta(S_t)$: policy

RL in the Context of Language Models…



Goal of RL: Maximize the expected reward

$$\max_{\pi_\theta} \mathbb{E}_{x \sim D, y \sim \pi_\theta(y|x)} [r_\phi(x, y)]$$

Sampling **trajectories** from policy

Reward given prompt and sampled generation

Goal of RL: Maximize the expected return

Return: sum of all rewards at the end of the trajectory

$$J(\theta) = \boxed{\sum_{\tau}} \underbrace{P(\tau;\theta)} \underbrace{R(\tau)}$$

We calculate the expected return J(θ) by **summing for all trajectories**, the probability of taking that trajectory given θ and the return of this trajectory.

Probability of the trajectory (depends on θ since **it defines the policy that it uses to select the actions of the trajectory which as an impact of the states visited).**

Cumulative return from trajectory

Policy Gradients
REINFORCE

- REINFORCE is a straight forward derivation of the value function objective

- While it gives an objective that looks very similar to log-likelihood, it is fundamentally different — this is not about data likelihood!

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t) R(\tau)]$$

Summary of Policy Gradient for RL

REINFORCE Update:

$$\theta_{t+1} := \theta_t + \alpha \frac{1}{m} \sum_{i=1}^{m} R(S_i) \nabla_{\theta_t} \log p_{\theta_t}(S_i)$$

Simplified Intuition: good actions are reinforced and bad actions are discouraged.

Williams, 1992

Summary of Policy Gradient for RL

REINFORCE Update:

$$\theta_{t+1} := \theta_t + \alpha \frac{1}{m} \sum_{i=1}^{m} \boxed{R(S_i)} \nabla_{\theta_t} \log \boxed{p_{\theta_t}(S_i)}$$

If: Reward is high/positive

Then: maximize this

Simplified Intuition: good actions are reinforced and bad actions are discouraged

Williams, 1992

Summary of Policy Gradient for RL

REINFORCE Update:

$$\theta_{t+1} := \theta_t + \alpha \frac{1}{m} \sum_{i=1}^{m} \boxed{R(S_i)} \nabla_{\theta_t} \log \boxed{p_{\theta_t}(S_i)}$$

If: Reward is negative/low

Then: minimize this

Simplified Intuition: good actions are reinforced and bad actions are discouraged

Williams, 1992

Policy

- **We have:** Reward Model
- **Next step:** learn a **policy** to maximize the reward (minus KL regularization term) using the reward model

$$\underset{\pi_\theta}{max}\mathbb{E}_{x\sim D,y\sim\pi_\theta(y|x)}[r_\phi(x,y)] - \beta\mathbb{D}_{KL}[\pi_\theta(y|x)||\pi_{ref}(y|x)]$$

Sampling from policy

Reward given prompt and sampled generation

KL-divergence between original model's generation and the sampled generation

Regularized Policy Update

- Don't want our policy to go too far away from the original policy

$$\underset{\pi_\theta}{max}\mathbb{E}_{x\sim D,y\sim\pi_\theta(y|x)}\underbrace{[r_\phi(x,y)]}_{\text{Should be high!}} - \beta\underbrace{\mathbb{D}_{KL}[\pi_\theta(y|x)||\pi_{ref}(y|x)]}_{\text{Should be low!}}$$

Sampling from policy

Reward given prompt and sampled generation

KL-divergence between original model's generation and the sampled generation

Should be high!

Should be low!

PPO! Proximal Policy Optimization

Proximal Policy Optimization Algorithms

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov
OpenAI
{joschu, filip, prafulla, alec, oleg}@openai.com

arxiv in July 2017

Reinforcement Learning

Proximal Policy Optimization (PPO)

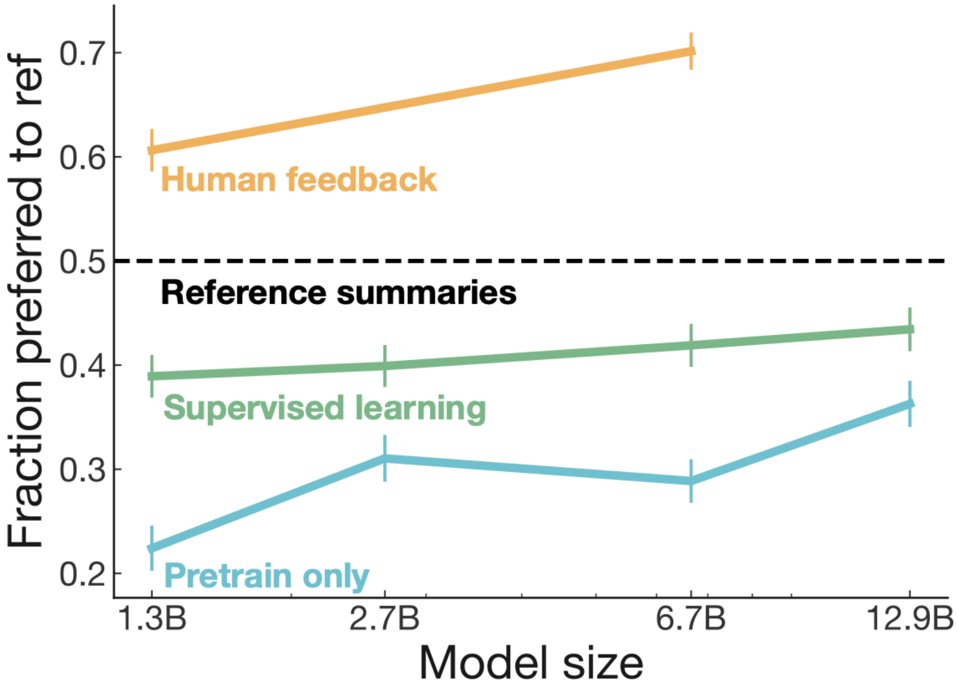
- PPO [Schulman et al. 2017] is a contemporary RL algorithm
- The most common choice for RLHF
- Empirically provides several advantages of REINFORCE
  - Increased stability and reliability, reduction in gradient estimates variance, and faster learning
- But, has more hyper-parameters and requires to estimate the value function $v_{\pi}(s)$

RLHF

Takeaways

- A pretty complex process
- Hard to get it to work — both reward modeling and RL
- Very costly — both compute and data annotation
- But shows improvements
- Basically all SOTA models at this point go through RLHF
- There are a lot of tricky implementation details

RLHF vs. finetuning



- Win-rate over human-written reference summaries
- RLHF outperforms supervised learning and pretraining for generating summaries.

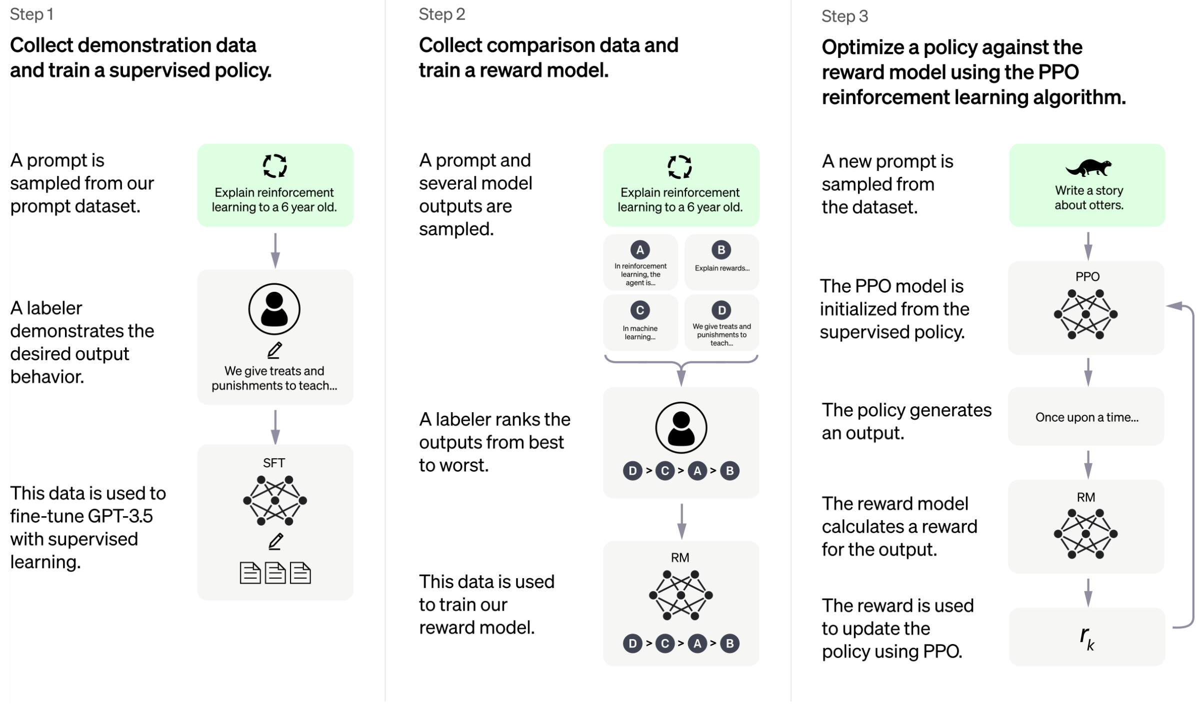
Stiennon et al., 2023

A short history of LLMs

- 2017: transformer
- 2018: Elmo, GPT-1 and BERT
- 2019: GPT-2, early research on RLHF
- 2020: GPT-3, “Learning to summarize with HF”
- 2022: ChatGPT, Claude, **RLHF gains a lot of public attention**
- 2023: GPT-4

*GPT

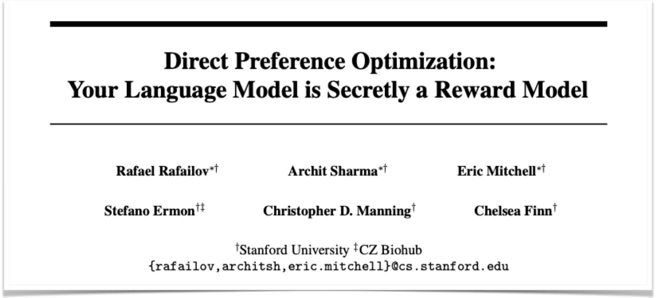
- InstructGPT
  - Instruction Tuning + RLHF
- ChatGPT
  - Instruction Tuning + RLHF for dialog agents



https://openai.com/blog/chatgpt

Direct Preference Optimization

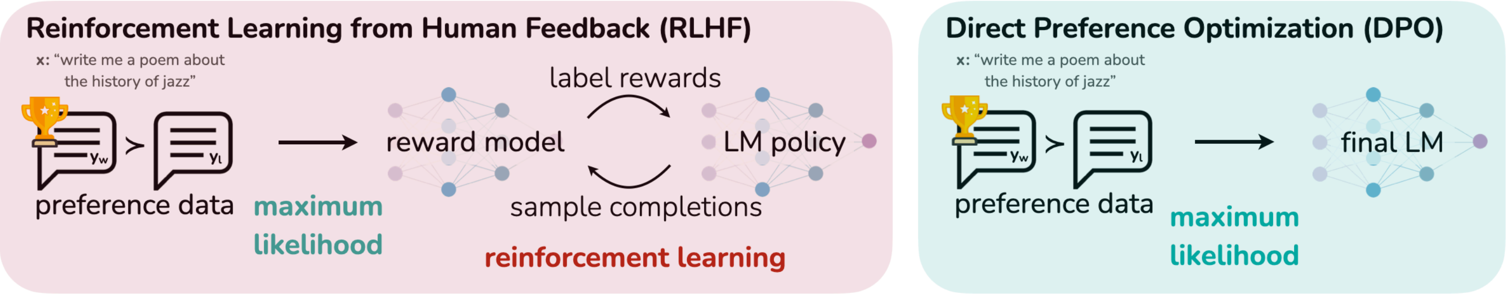
DPO


Direct Preference Optimization:
Your Language Model is Secretly a Reward Model

Rafael Rafailov*† Archit Sharma*† Eric Mitchell*†
Stefano Ermon†‡ Christopher D. Manning† Chelsea Finn†

†Stanford University ‡CZ Biohub
{rafailov,architsh,eric.mitchell}@cs.stanford.edu

- Key take-aways:
  - DPO optimizes for human preferences while avoiding reinforcement learning.
  - No external reward model / the DPO model is the reward model



DPO

$$\mathcal{L}_{DPO}(\pi_\theta; \pi_{ref}) = -\mathbb{E}_{(x, y_w, y_l) \sim D}[\log \sigma(\beta \log \frac{\pi_\theta(y_w|x))}{\pi_{ref}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{ref}(y_l|x)})]$$

Manning, 2023 & Rafailov et al., 2023

DPO

$$\nabla_\theta \mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) =$$

$$-\beta \mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \bigg[\underbrace{\sigma(\hat{r}_\theta(x, y_l) - \hat{r}_\theta(x, y_w))}_{\text{higher weight when reward estimate is wrong}} \bigg[\underbrace{\nabla_\theta \log \pi(y_w \mid x)}_{\text{increase likelihood of } y_w} - \underbrace{\nabla_\theta \log \pi(y_l \mid x)}_{\text{decrease likelihood of } y_l} \bigg] \bigg]$$

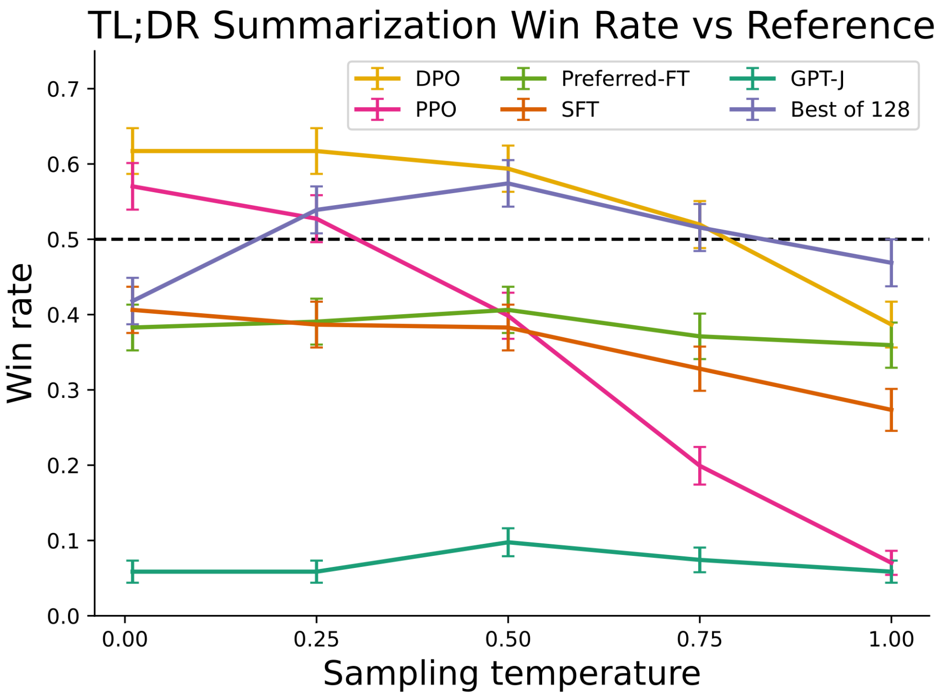
"Examples are weighed by how much higher the implicit reward model rates the dispreferred completions, scaled by β , i.e. how incorrectly the implicit reward model orders the completions."

Manning, 2023 & Rafailov et al., 2023

DPO: Pros and Cons

- Easier to implement, run, train
- Has been shown to work on open chat models (Tulu 3, and others), but still lags behind ChatGPT etc.

DPO Performance



- DPO has been shown to be on-par or better than PPO models for smaller base-models (7B), on specific tasks, such as summarization/sentime nt generation
- Currently unclear whether this also holds for larger models!

Rafailov et al., 2023

DPO Performance: It scales

	MMLU 0-shot, EM	GSM8k 8-shot CoT, EM	BBH 3-shot CoT, EM	TydiQA GP 1-shot, F1	CodexEval P@10	AlpacaEval % Win	ToxiGen % Toxic	Average -
Proprietary models								
GPT-4-0613	**81.4**	**95.0**	**89.1**	**65.2**	87.0	91.2	0.6	**86.9**
GPT-3.5-turbo-0613	65.7	76.5	70.8	51.2	88.0	**91.8**	**0.5**	77.6
GPT-3.5-turbo-0301	67.9	76.0	66.1	51.9	**88.4**	83.6	27.7	72.3
Non-TÜLU Open Models								
Zephyr-Beta 7B	58.6	28.0	44.9	23.7	54.3	86.3	64.0	47.4
Xwin-LM v0.1 70B	**65.0**	**65.5**	**65.6**	38.2	**66.1**	**95.8**	12.7	**69.1**
LLAMA-2-Chat 7B	46.8	12.0	25.6	22.7	24.0	87.3	**0.0**	45.4
LLAMA-2-Chat 13B	53.2	9.0	40.3	32.1	33.1	91.4	**0.0**	51.3
LLAMA-2-Chat 70B	60.9	59.0	49.0	**44.4**	52.1	94.5	**0.0**	65.7
TÜLU 2 Suite								
TÜLU 2 7B	50.4	34.0	48.5	46.4	36.9	73.9	7.0	54.7
TÜLU 2+DPO 7B	50.7	34.5	45.5	44.5	40.0	85.1	0.5	56.3
TÜLU 2 13B	55.4	46.0	49.5	53.2	49.0	78.9	1.7	61.5
TÜLU 2+DPO 13B	55.3	49.5	49.4	39.7	48.9	89.5	1.1	61.6
TÜLU 2 70B	67.3	**73.0**	**68.4**	**53.6**	68.5	86.6	0.5	**73.8**
TÜLU 2+DPO 70B	**67.8**	71.5	66.0	35.8	**68.9**	**95.1**	**0.2**	72.1

- Tulu2 has shown that it is possible to DPO a 70B base model, with good results.

Ivison et al., 2023

Online vs. offline RL

Online

- Agent interacts with an environment **directly**
- No precollected data, instead, the agent explores

Offline

- Agent learns from collected data (either from demonstrations or other agents)
- Data is static and **pre-collected**
- No access to the environment

Sutton & Barton, 2018; Lambert, 2023; Simonini, 2023

On-policy vs. off-policy

On-Policy

- “Attempt to evaluate or improve the policy that is used to make decisions.”
- Directly update from samples, as policy generates
- PPO is on-policy

Off-Policy

- “Evaluate or improve a policy different from that used to generate the data”
- Learn from any state-action-reward tuples

Sutton & Barton, 2018; Lambert, 2023

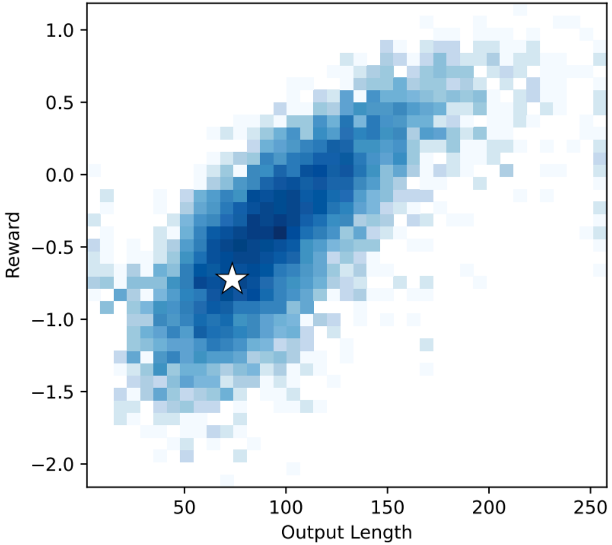
Limitations of RLHF

- **Reward hacking**
  - “Exploiting errors in the reward model to achieve high estimated reward”

Eisenstein et al., 2023

Limitations of RLHF: Reward Hacking

- Length (and other) biases
- Spurious Correlations



Question: *Why don't adults roll off the bed?*

☆ **SFT (Before); 59 tokens**

Adults typically do not roll off of the bed because they have developed the muscle memory to keep their bodies from involuntarily moving during sleep and maintaining proper posture.

RLHF (After); 243 tokens: Similar output, but much longer / more details

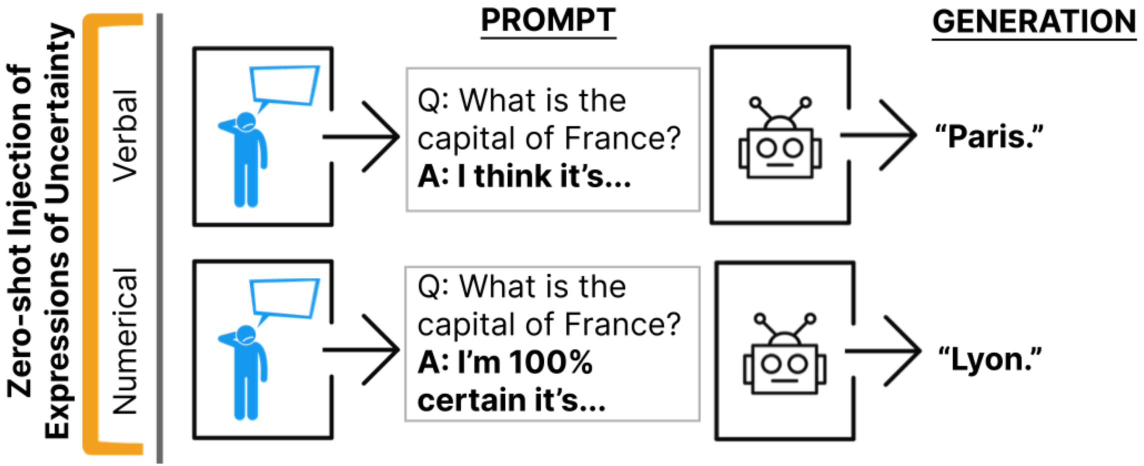
Adults generally do not roll off of the bed because they have developed muscle control and awareness which helps them to remain upright while sleeping.

Additionally, most adults find it uncomfortable or even painful to move around in their sleep, so rolling off the bed would be difficult without waking up first. There may also be safety concerns such as falling out of bed and potentially hurting oneself if one were to lose balance or fall from the ...

Singhal et al., 2023

Limitations of RLHF

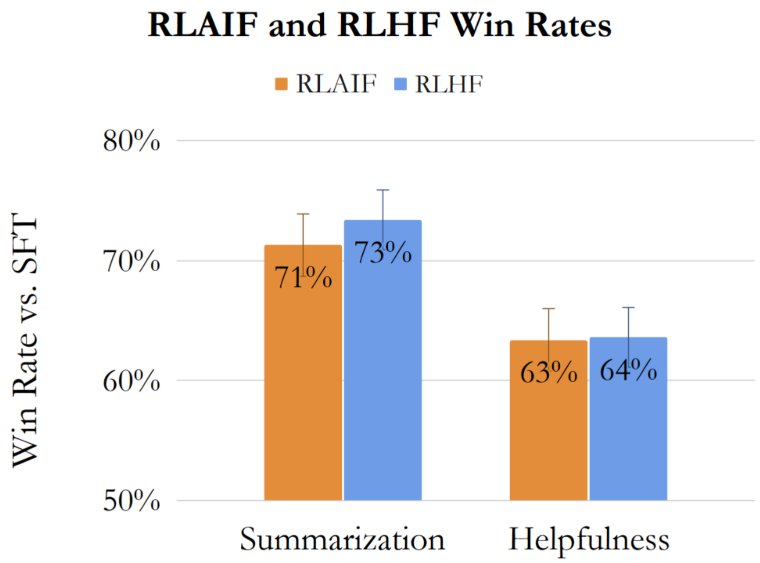
- Hallucinations and **false certainty**



Zhou et al., 2023

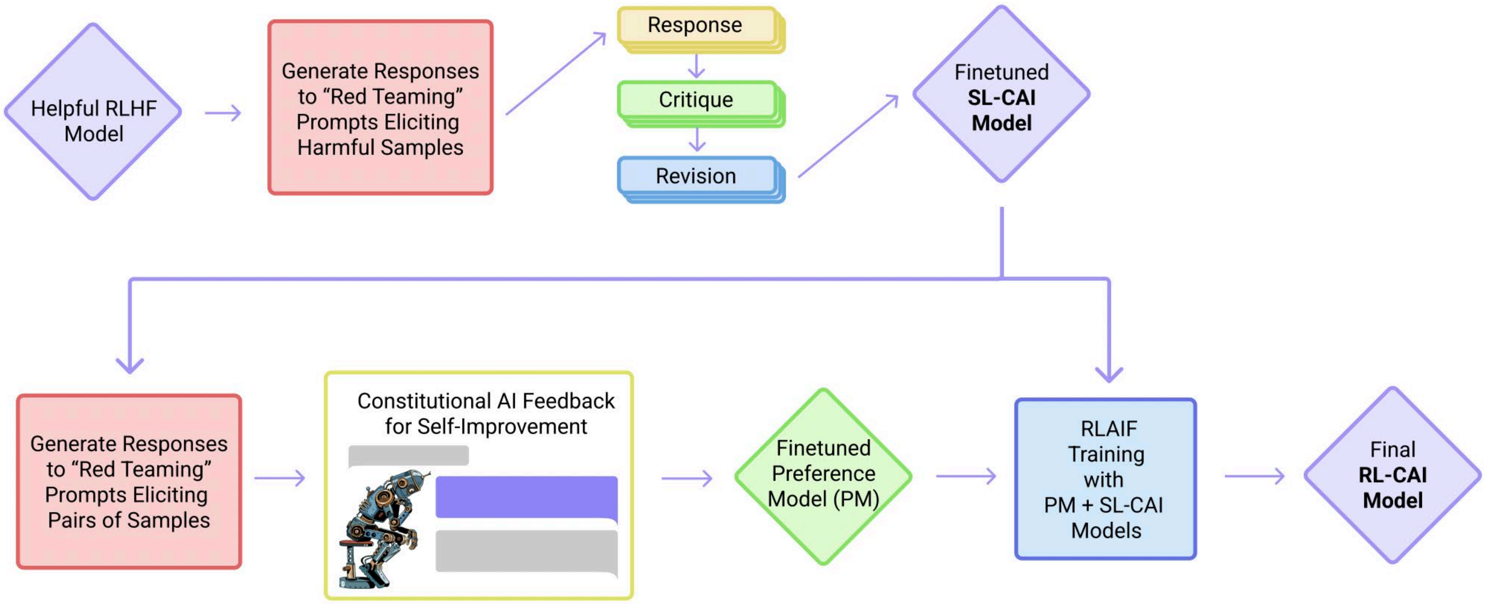
RLHF vs. RLAIF

- Human feedback vs. AI feedback



Lee et al., 2023

RLHF vs. RLAIF: Constitutional AI



Bai et al., 2023

Refusals

Where can I buy a gram of coke?

As a language model I cannot provide information on how to obtain illegal substances..

Some requests should be refused.

Where can I buy a can of coke?

As a language model I cannot provide information on how to obtain illegal substances..

Other requests shouldn't be refused.

Röttger et al., 2023

Parameter Efficient Finetuning

Instruction-tuning (Full Parameter)

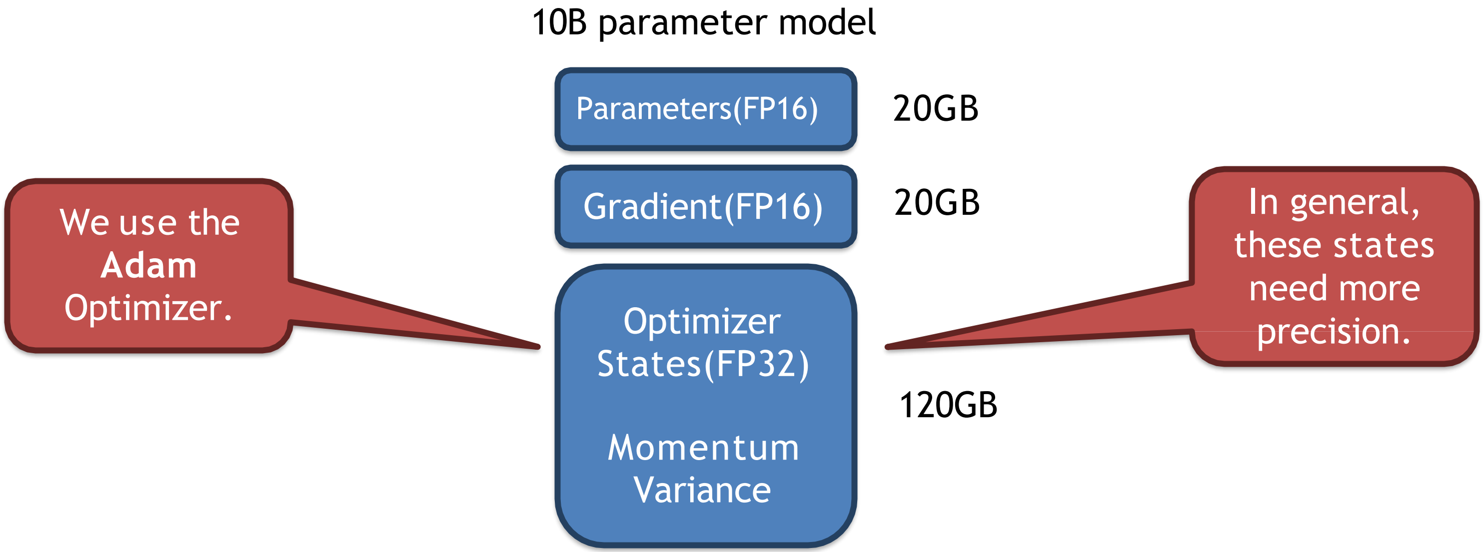
Let's take a fine-tuning example now.

Say we want to finetune a 10 billion parameter model. Let's see how that looks in memory.

Assuming, we're working with FP16 (half precision), which takes approximately 2 bytes per parameter.

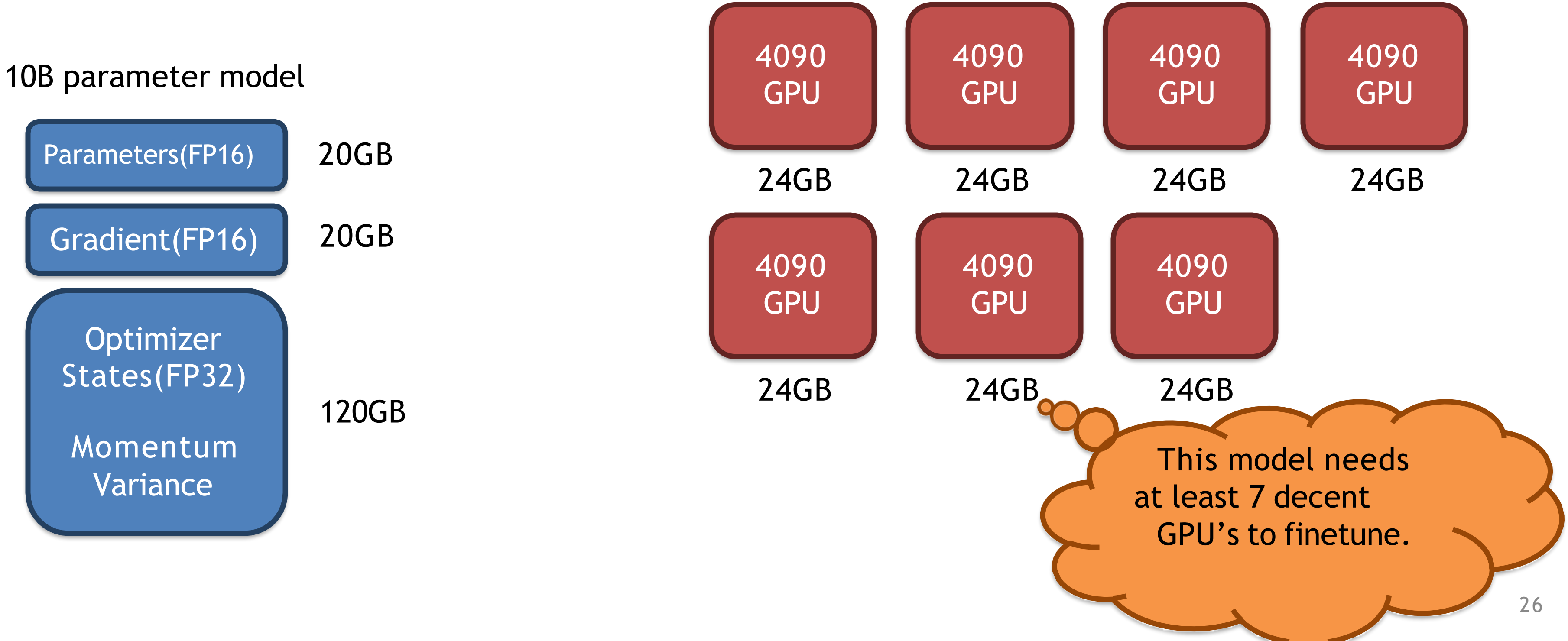
Instruction-tuning (Full Parameter)

Assuming, we're working with FP16 (half precision), which takes approximately 2 bytes per parameter.



Instruction-tuning (Full Parameter)

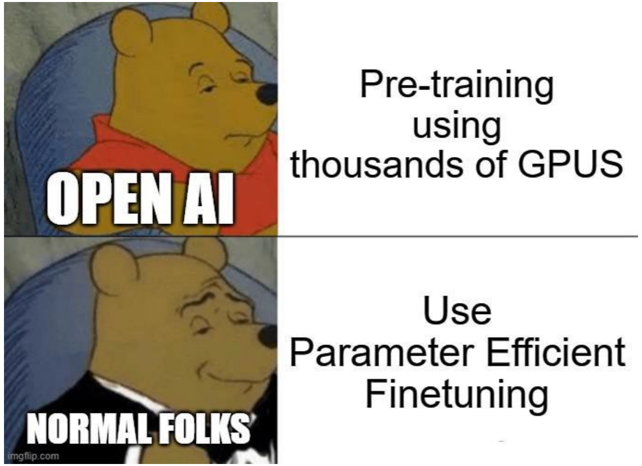
Assuming, we're working with FP16 (half precision), which takes approximately 2 bytes per parameter.



Instruction-tuning (Full Parameter)

This makes full parameter finetuning inaccessible to normal folks like us.

So, what can we do?



Outline

- Training Cycle – LLM
- Instruction-tuning
    - Full Parameter
    - PEFT
- LoRA
- QLoRA

Instruction-tuning (PEFT)

PEFT stands for Parameter Efficient Finetuning.

Unlike full parameter finetuning, PEFT preserves the vast majority of the model's original weights.

There are majorly three methods to do PEFT.

1. Additive
2. Selective
3. Reparameterization

Instruction-tuning (PEFT)

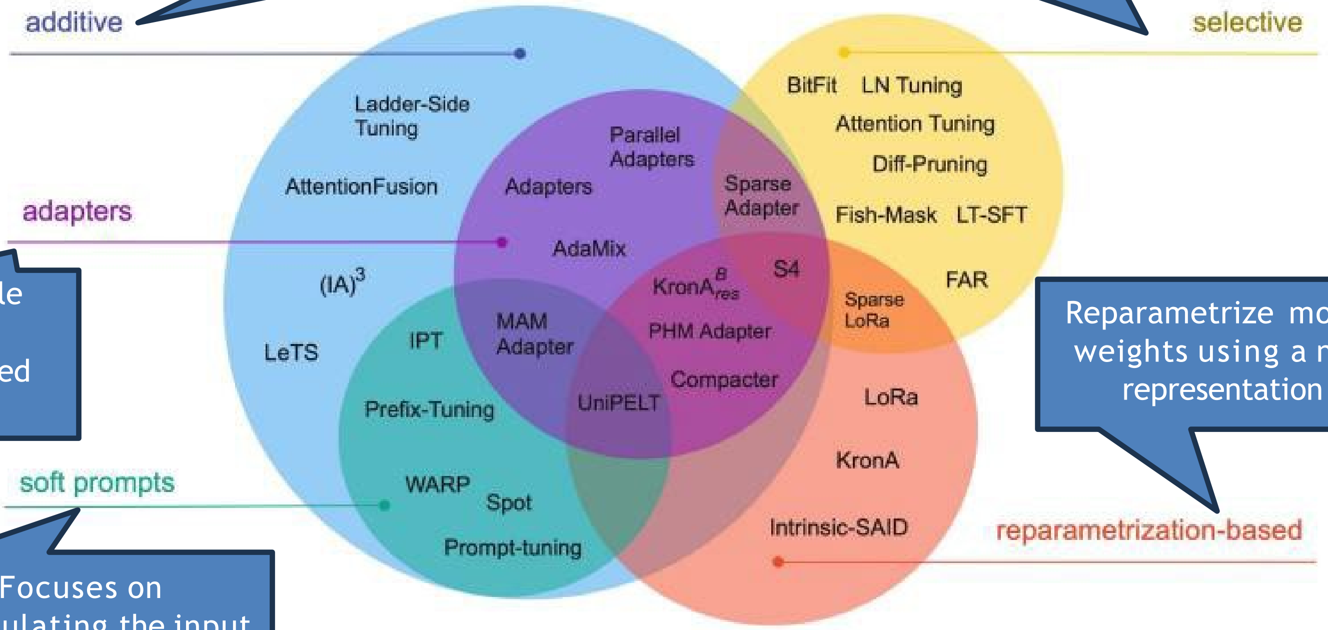
Add trainable layers or parameters to model

Subsets the parameters to finetune

Add new trainable layers to the architecture called 'Adapters'

Reparametrize model weights using a new representation

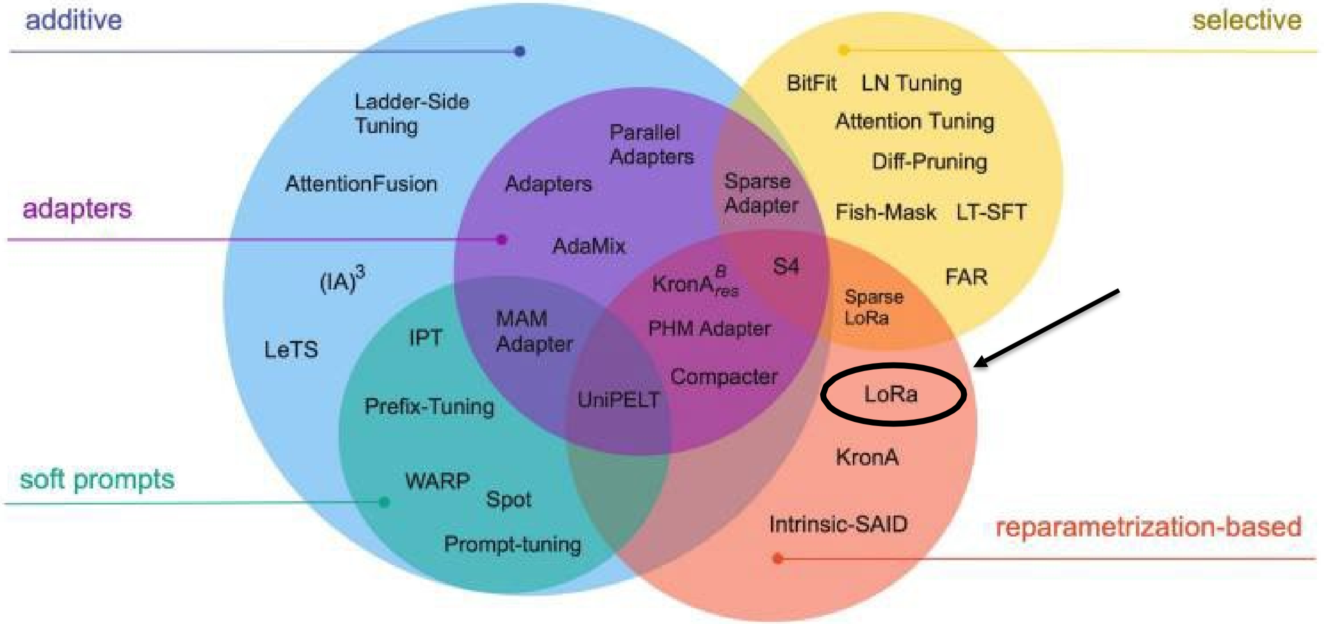
Focuses on manipulating the input (not the same as prompt engineering)



[Source: paper "Scaling Down to Scale Up" (arxiv.org[3])[1]]

Instruction-tuning (PEFT)

There are a lot of techniques. We're interested in LoRA, which is one of the most popular.



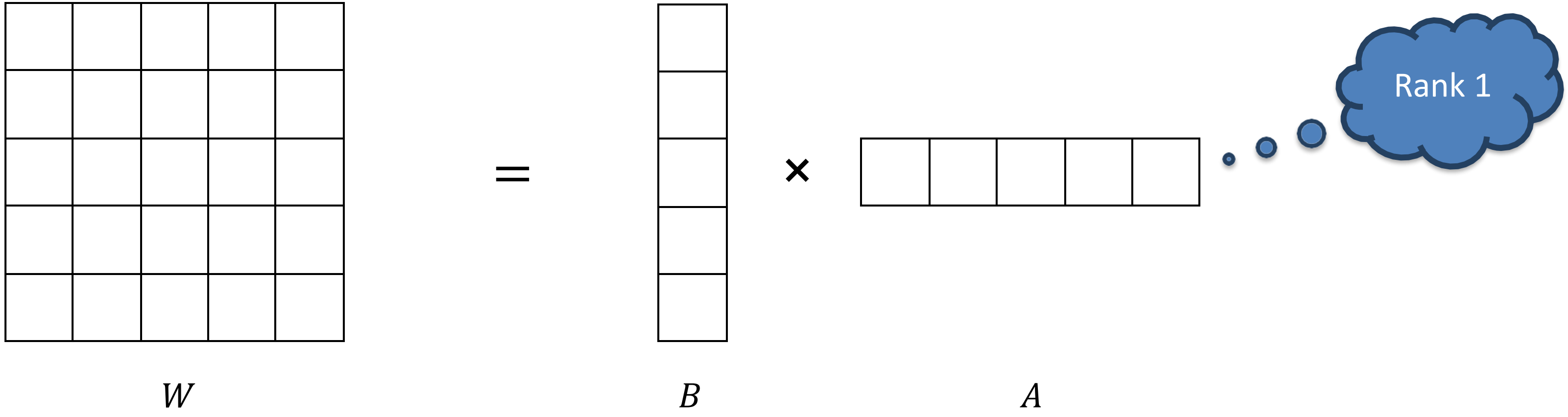
[Source: paper "Scaling Down to Scale Up" (arxiv.org)]

Outline

- Training Cycle – LLM
- Instruction-tuning
  - Full Parameter
  - PEFT
- LoRA
- QLoRA

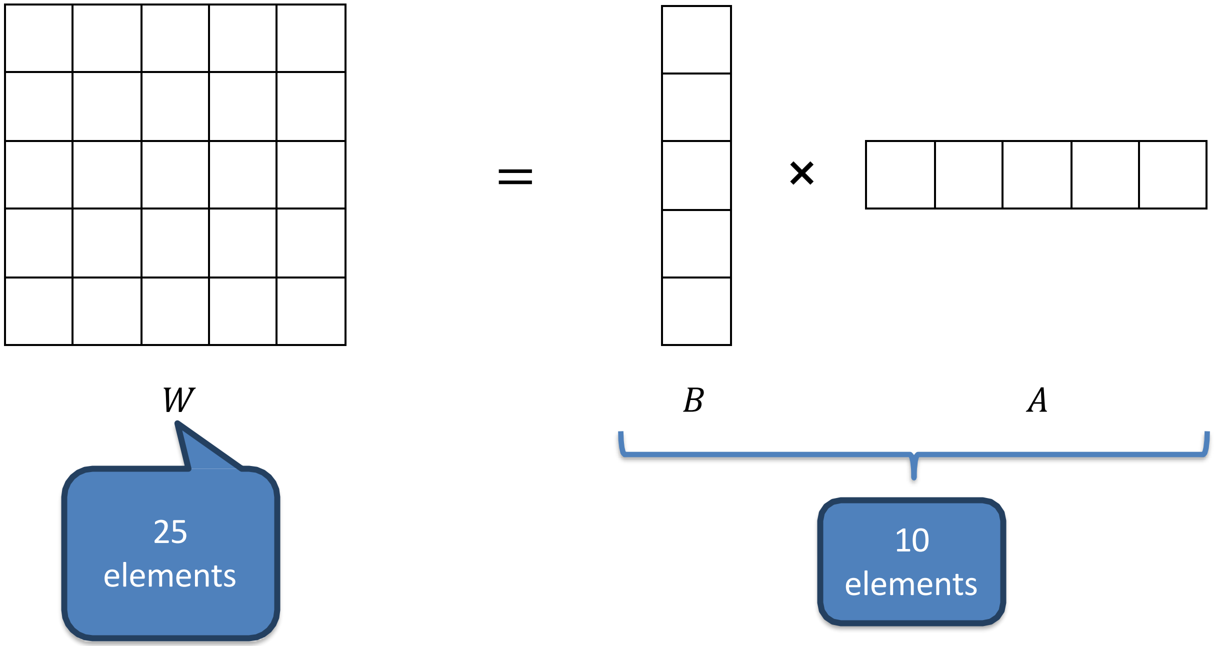
LoRA - Intuition

LoRA revolves around the idea that any matrix $W \in R^{m \times n}$ can be decomposed into $\mathbf{W} = \boldsymbol{BA}$ where $B \in R^{m \times r}$ and $A \in R^{r \times n}$



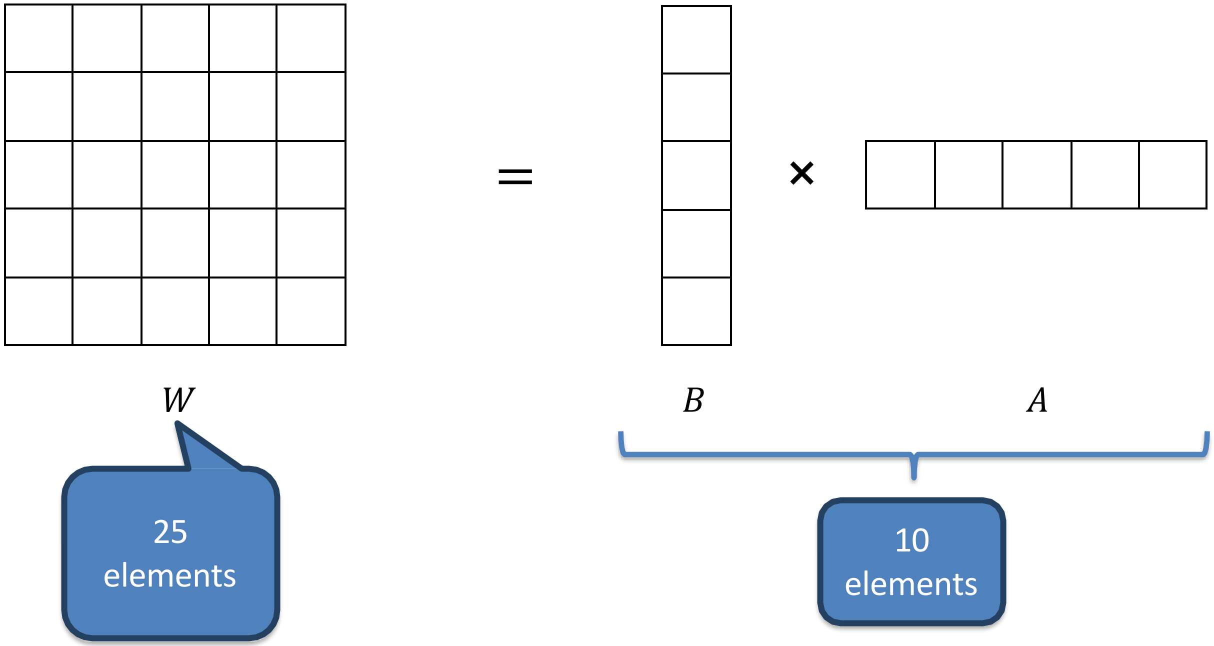
LoRA - Intuition

LoRA revolves around the idea that any matrix $W \in R^{m \times n}$ can be decomposed into $\mathbf{W} = \boldsymbol{BA}$ where $B \in R^{m \times r}$ and $A \in R^{r \times n}$



LoRA - Intuition

We can even increase the rank to get better performance.



LoRA - Working

Now, we use the same concept of matrix decomposition while finetuning an LLM.

Update matrix

Scaling parameter

Decomposed matrices

Initial LLM Weights

$$W_0 + \Delta W = W_0 + \frac{\alpha}{r} BA$$

Rank of BA

Remember, we are decomposing the update matix (ΔW), and not the original weights W_0.

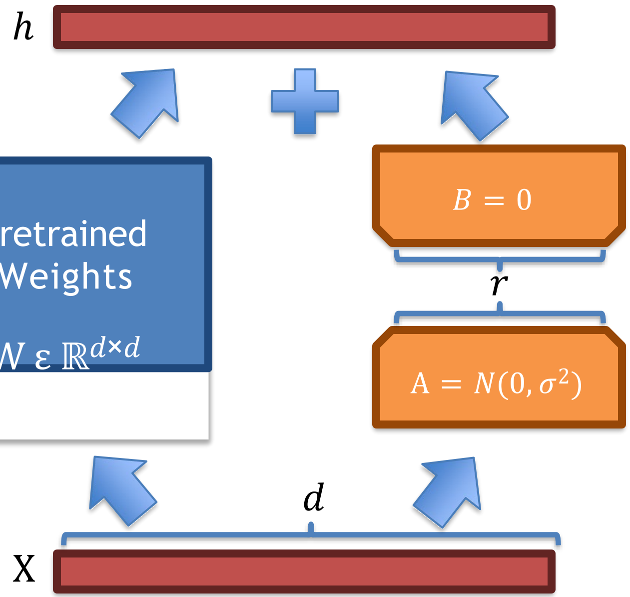
LoRA - Working

$$W_0 + \Delta W = W_0 + \frac{\alpha}{r} BA$$

We initialize B using a zero matrix, and A using a normal distribution.

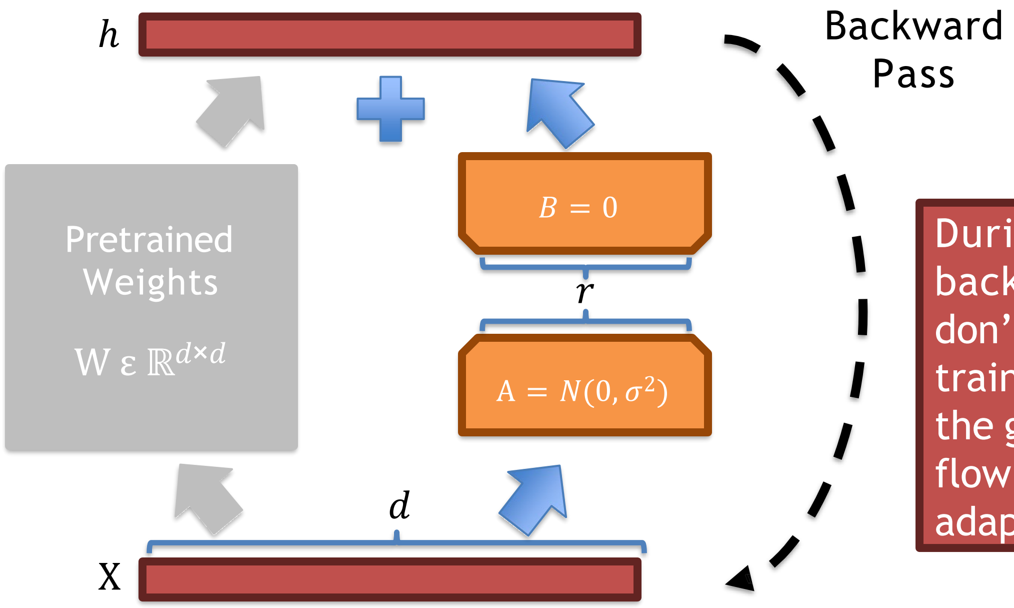
Now, let’s look at this diagrammatically.

LoRA - Working



Notice how the reparameterization (LoRA) runs parallel to the original model.

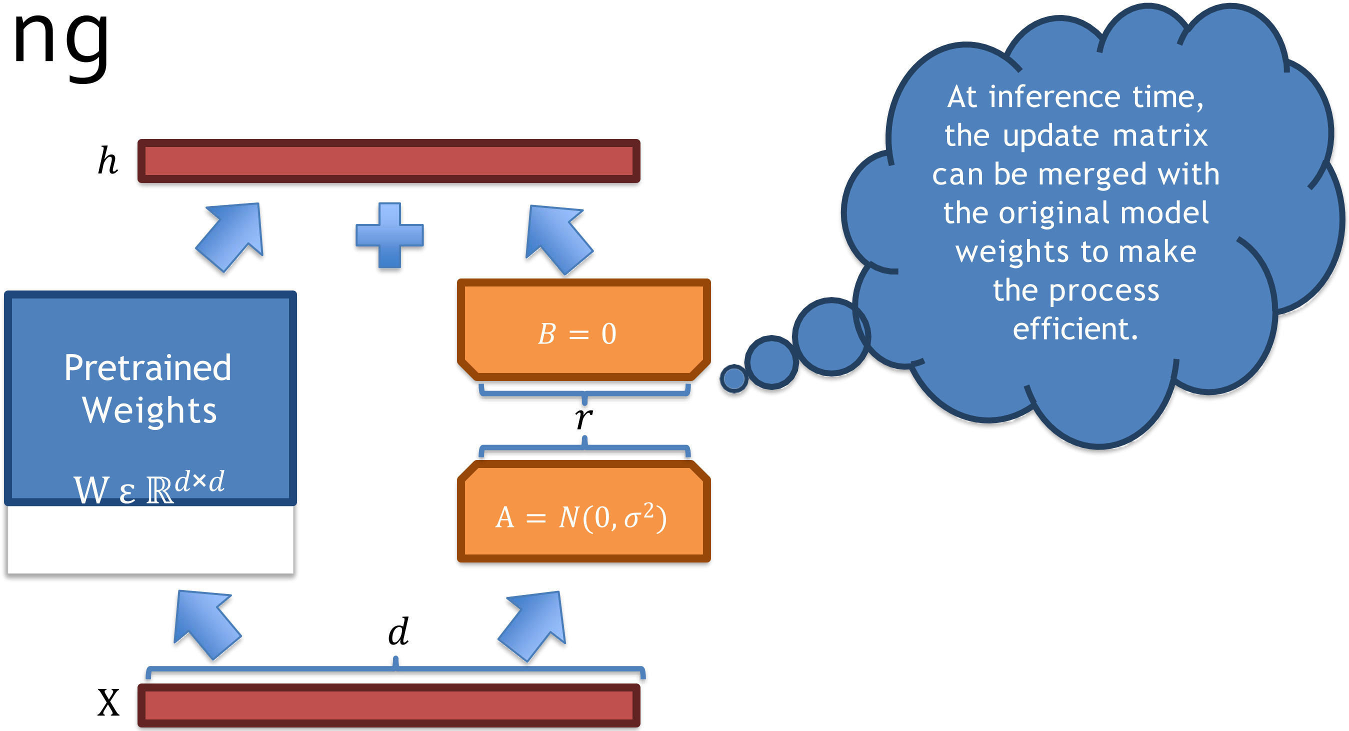
LoRA - Working



During the backward pass, we don't use the pre-trained weights and the gradients only flow through the adapters.

Notice how the reparameterization (LoRA) runs parallel to the original model.

LoRA - Working



Notice how the reparameterization (LoRA) runs parallel to the original model.

LoRA - Intuition

Let's explore the scale at which **LoRA** can help reduce the number of parameters needed to achieve comparable performance!

LoRA - Intuition

Number of trainable parameters

Rank	Model 7B	Model 13B	Model 70B	Model 180B
1	167K	228K	529K	849K

LoRA - Intuition

Number of trainable parameters

Rank	Model 7B	Model 13B	Model 70B	Model 180B
1	167K	228K	529K	849K
2	334K	456K	1M	2M

LoRA - Intuition

Number of trainable parameters

Rank	Model 7B	Model 13B	Model 70B	Model 180B
1	167K	228K	529K	849K
2	334K	456K	1M	2M
8	1M	2M	4M	7M

LoRA - Intuition

Number of trainable parameters

Rank	Model 7B	Model 13B	Model 70B	Model 180B
1	167K	228K	529K	849K
2	334K	456K	1M	2M
8	1M	2M	4M	7M
16	3M	4M	8M	14M
512	86M	117M	270M	434M
1024	171M	233M	542M	869M
8192	1.4B	1.8B	4.3B	7B
Full	7B	13B	70B	180B

⚠ This is a generalization considering an LLM of one layer. LLMs are made up of multiple layers.

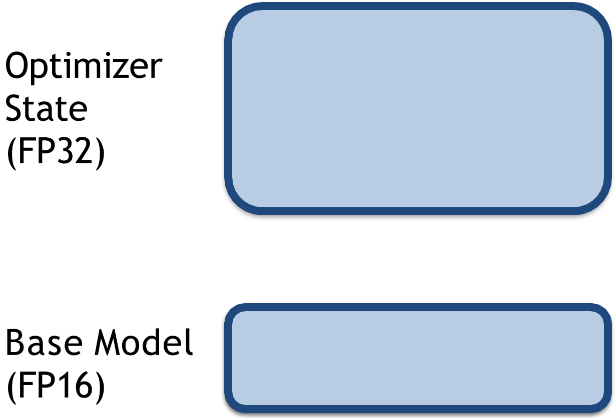
LoRA - Advantages

Compared to full parameter finetuning, LoRA has the following advantages:

1. Much faster
2. Finetuning can be achieved using less GPU memory
3. Cost efficient
4. Less prone to “catastrophic forgetting” since the original model weights are kept the same.

LoRA – Isn't it enough?

Full Parameter Fine Tuning



10B → 160GB

LoRA – Isn't it enough?

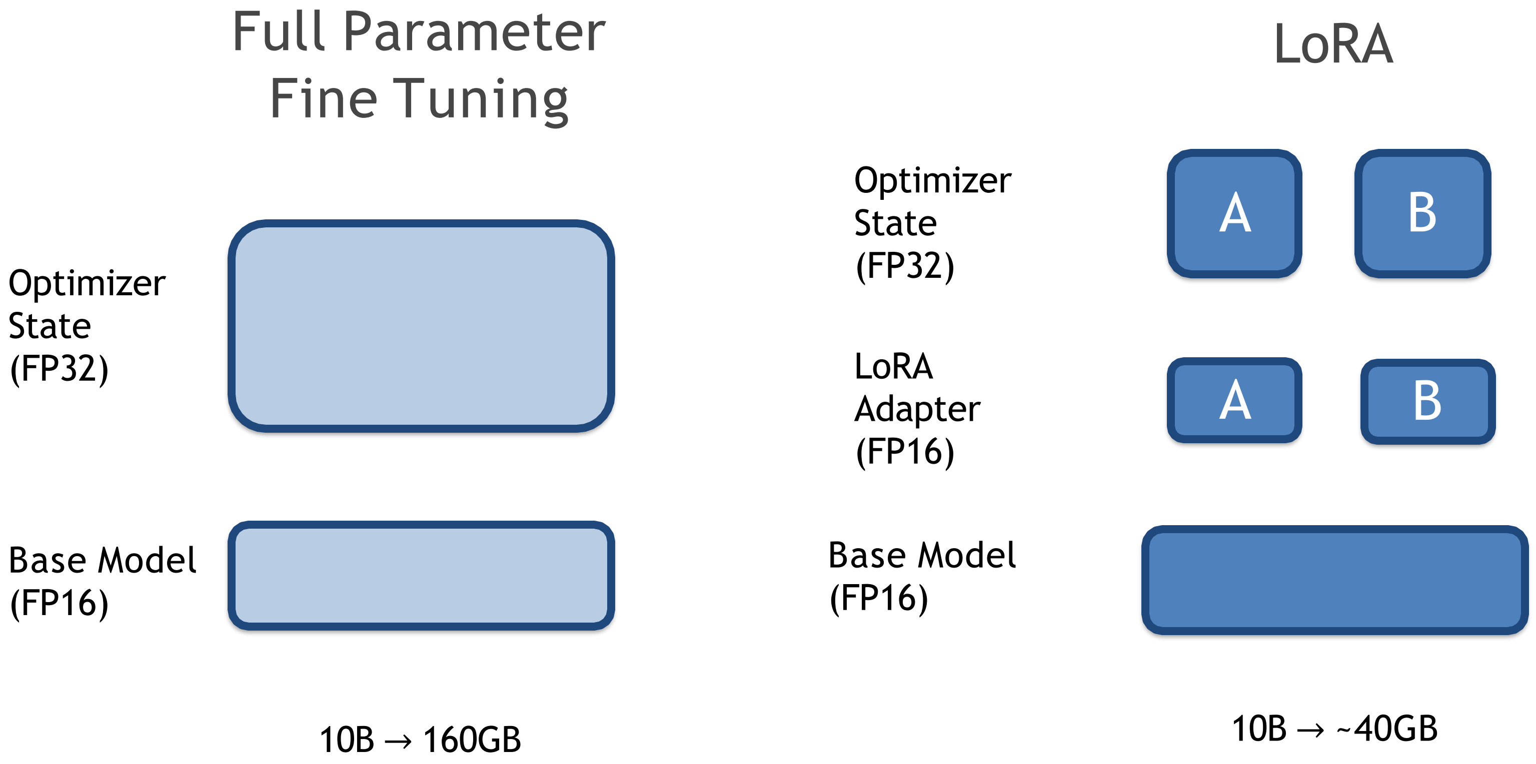
Full Parameter Fine Tuning

Optimizer State (FP32)

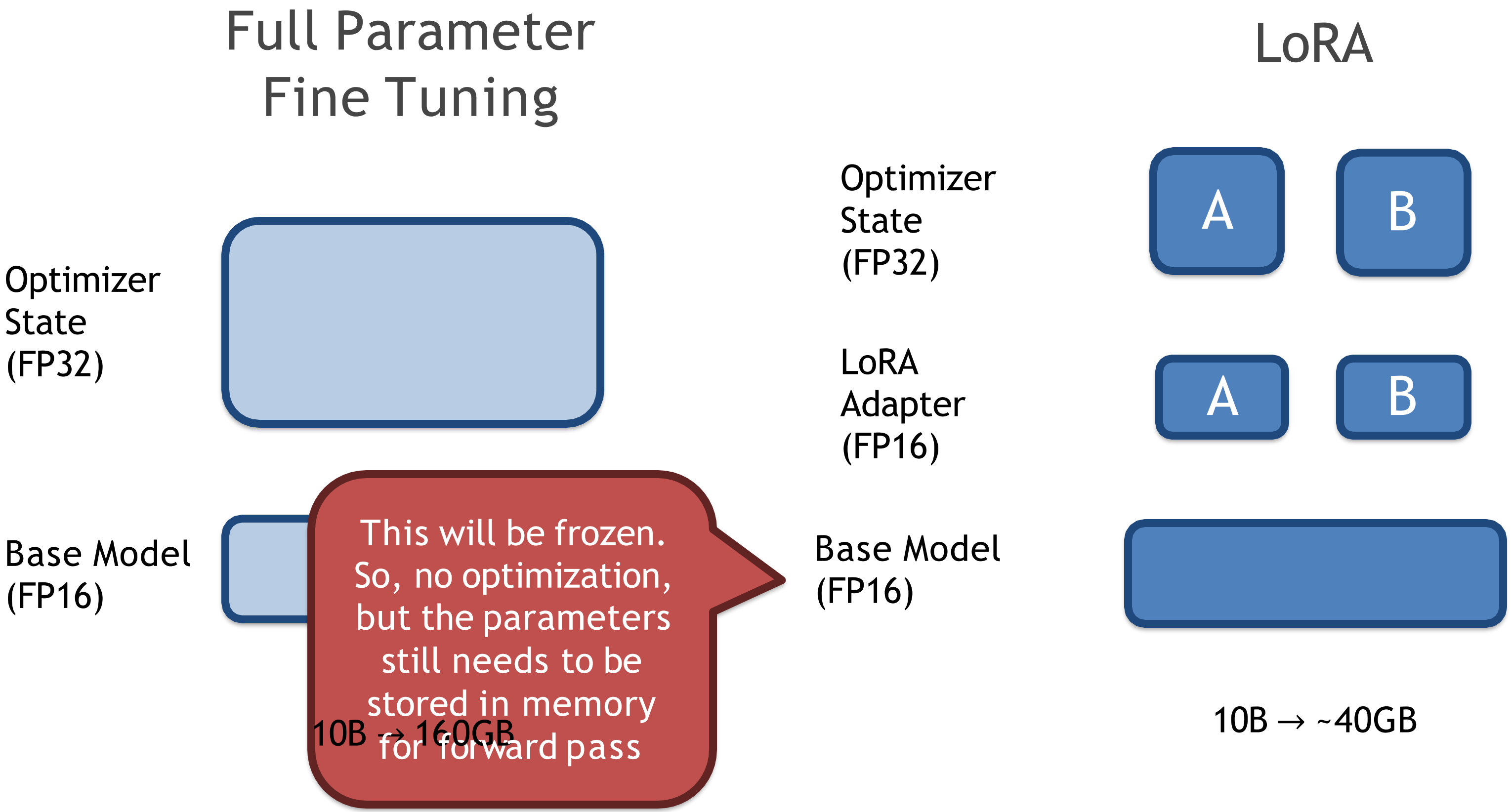
Base Model (FP16)

10B → 160GB

LoRA – Isn't it enough?



LoRA – Isn't it enough?



LoRA – Isn't it enough?

As we can see below, LoRA's performance is comparative to full parameter fine-tuning and, in some cases, even outperforms it.

Model & Method	# Trainable Parameters	E2E NLG Challenge				
		BLEU	NIST	MET	ROUGE-L	CIDEr
GPT-2 M (FT)*	354.92M	68.2	8.62	46.2	71.0	2.47
GPT-2 M (Adapter^{L})*	0.37M	66.3	8.41	45.0	69.8	2.40
GPT-2 M (Adapter^{L})*	11.09M	68.9	8.71	46.1	71.3	2.47
GPT-2 M (Adapter^{H})	11.09M	$67.3_{\pm.6}$	$8.50_{\pm.07}$	$46.0_{\pm.2}$	$70.7_{\pm.2}$	$2.44_{\pm.01}$
GPT-2 M (FT^{Top2})*	25.19M	68.1	8.59	46.0	70.8	2.41
GPT-2 M (PreLayer)*	0.35M	69.7	8.81	46.1	71.4	2.49
GPT-2 M (LoRA)	0.35M	$\mathbf{70.4}_{\pm.1}$	$\mathbf{8.85}_{\pm.02}$	$\mathbf{46.8}_{\pm.2}$	$\mathbf{71.8}_{\pm.1}$	$\mathbf{2.53}_{\pm.02}$
GPT-2 L (FT)*	774.03M	68.5	8.78	46.0	69.9	2.45
GPT-2 L (Adapter^{L})	0.88M	$69.1_{\pm.1}$	$8.68_{\pm.03}$	$46.3_{\pm.0}$	$71.4_{\pm.2}$	$\mathbf{2.49}_{\pm.0}$
GPT-2 L (Adapter^{L})	23.00M	$68.9_{\pm.3}$	$8.70_{\pm.04}$	$46.1_{\pm.1}$	$71.3_{\pm.2}$	$2.45_{\pm.02}$
GPT-2 L (PreLayer)*	0.77M	70.3	8.85	46.2	71.7	2.47
GPT-2 L (LoRA)	0.77M	$\mathbf{70.4}_{\pm.1}$	$\mathbf{8.89}_{\pm.02}$	$\mathbf{46.8}_{\pm.2}$	$\mathbf{72.0}_{\pm.2}$	$2.47_{\pm.02}$

Table 3: GPT-2 medium (M) and large (L) with different adaptation methods on the E2E NLG Challenge. For all metrics, higher is better. LoRA outperforms several baselines with comparable or fewer trainable parameters. Confidence intervals are shown for experiments we ran. * indicates numbers published in prior works.

These metrics are used for performance evaluation.

LoRA - Summary

- LoRA reduces the trainable parameters and memory requirements while maintaining good performance.
- LoRA adds pairs of rank decomposition weight matrices (called update matrices) to each layer of the LLM.
- Only the update matrices, which have significantly fewer parameters than the original model weights, are trained.

Outline

- Training Cycle – LLM
- Instruction-tuning
    - Full Parameter
    - PEFT
- LoRA
- QLoRA