

Introduction to Building LLMs

CS229: Machine Learning

Yann Dubois | Aug. 13th 2024

Slides partially based on CS336, CS224N, CS324



LLMs

- LLMs & chatbots took over the world



- How do they work?

ChatGPT



What are you?

I'm a large language model trained by OpenAI. I'm a form of artificial intelligence that has been designed to process and generate human-like language.

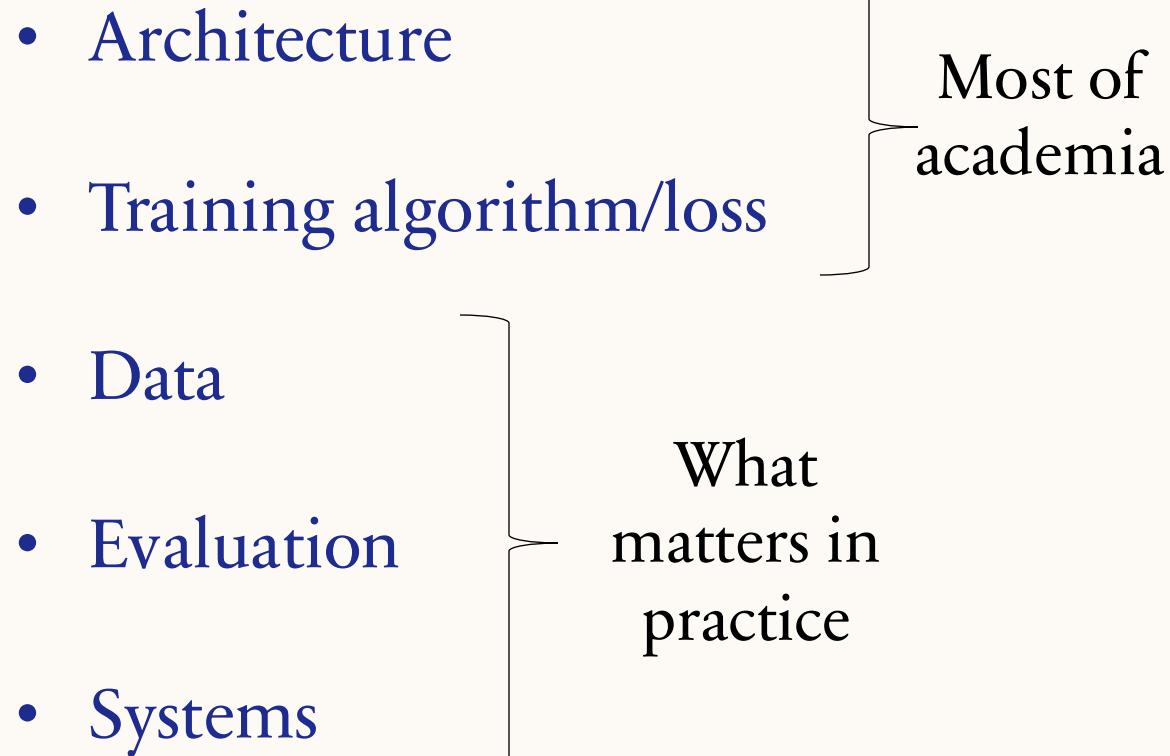


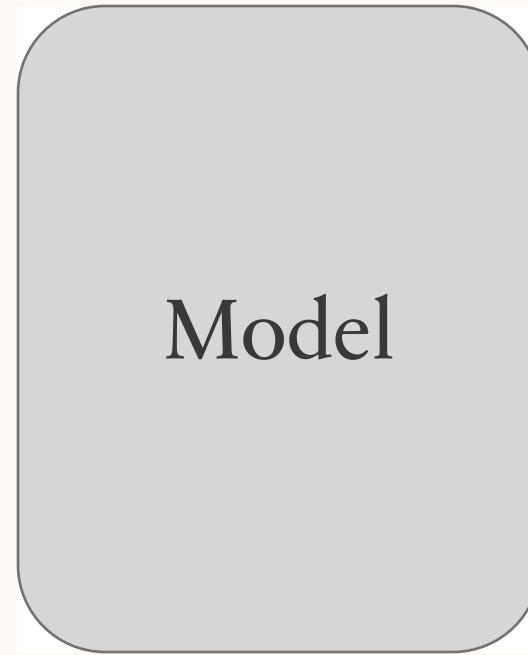
Are you human?

I'm not a human and I don't have the ability to think or feel in the same way that a person does.



What matters when training LLMs

- Architecture
 - Training algorithm/loss
 - Data
 - Evaluation
 - Systems
- Most of academia
- What matters in practice
- 

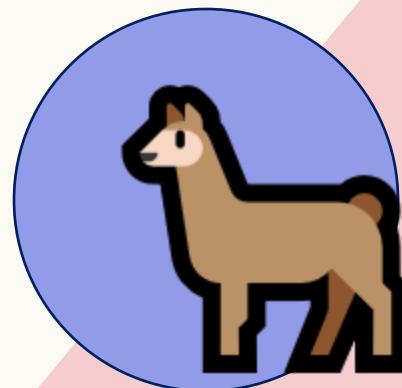


Overview

Pretraining -> GPT3

- Task & loss

Post-training -> ChatGPT



Language Modeling

- LM: probability distribution over sequences of tokens/words $p(x_1, \dots, x_L)$

$$P(\text{the, mouse, ate, the, cheese}) = 0.02$$

$$P(\text{the, the, mouse, ate, cheese}) = 0.0001 \quad \text{Syntactic knowledge}$$

$$P(\text{the, cheese, ate, the, mouse}) = 0.001 \quad \text{Semantic knowledge}$$

- LMs are generative models: $x_{1:L} \sim p(x_1, \dots, x_L)$

- **Autoregressive (AR) language models:**

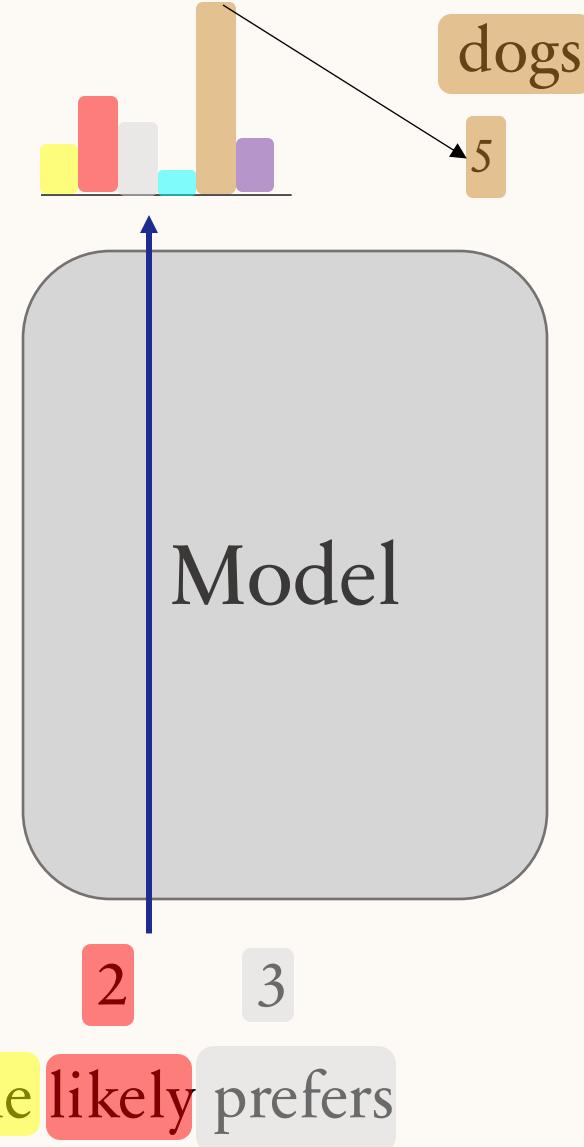
$$p(x_1, \dots, x_L) = p(x_1)p(x_2|x_1)p(x_3|x_2, x_1)\dots = \prod_i p(x_i|x_{1:i-1})$$

No approx: chain rule of probability

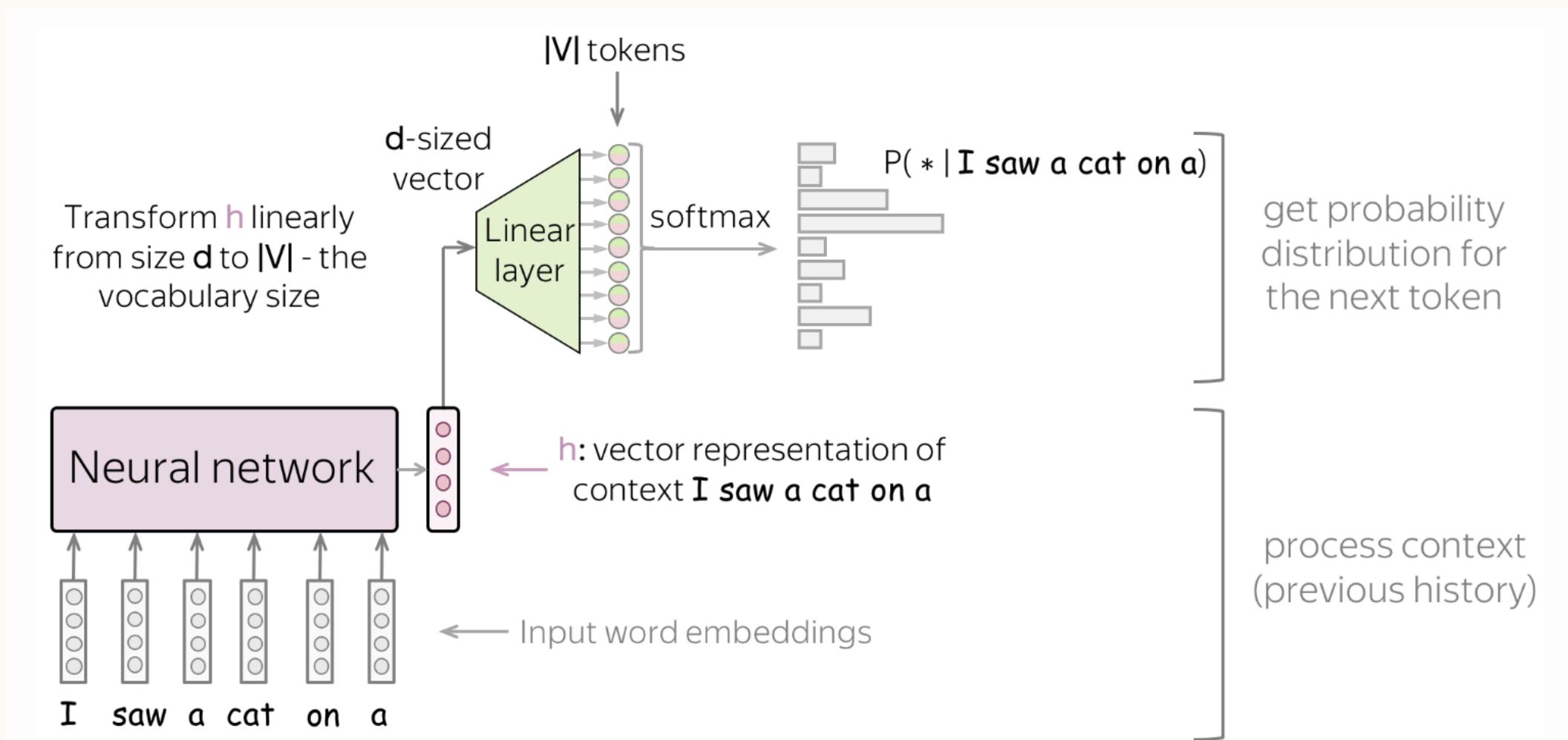
=> You only need a model that can predict the next token given past context!

AR Language Models

- Task: predict the next word
 - Steps:
 1. tokenize
 2. forward
 3. predict probability of next token
 4. sample
 5. detokenize
- Inference only

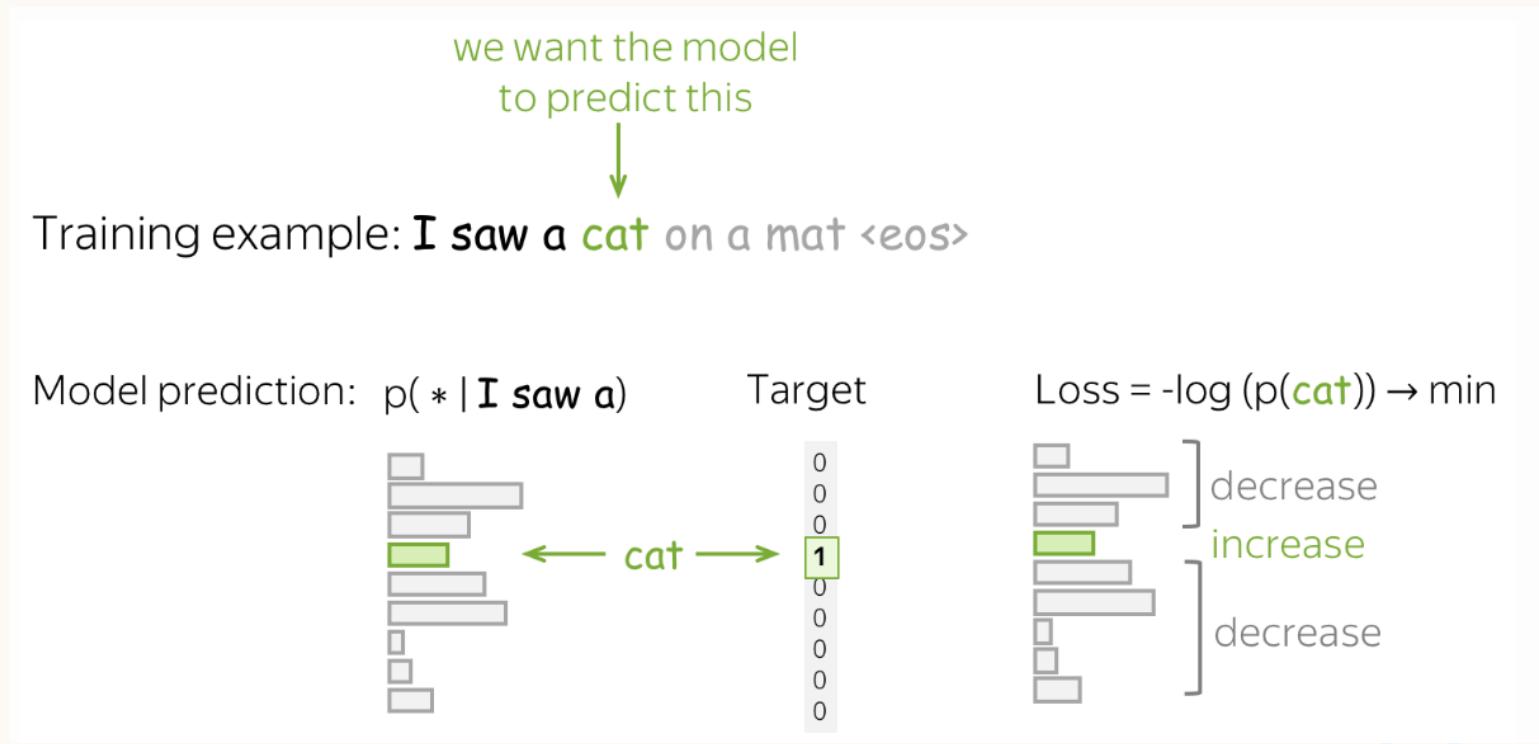


AR Neural Language Models



Loss

- Classify next tokens' index
 - => cross-entropy loss



- => maximize text's log-likelihood

https://lena-voita.github.io/nlp_course/language_modeling.html#intro

$$\max \prod_i p(x_i | x_{1:i-1}) = \min \left(- \sum_i \log p(x_i | x_{i:i-1}) \right) = \min \mathcal{L}(x_{i:L})$$

Tokenizer

- Why?
 - More general than words (eg typos)
 - Shorter sequences than with characters
- Idea: tokens as common subsequences (~3 letters)
- Eg: Byte Pair Encoding (BPE). Train steps:
 1. Take large corpus of text

tokenizer:
text to token
index

Tokenizer

- Why?
 - More general than words (eg typos)
 - Shorter sequences than with characters
- Idea: tokens as common subsequences
- Eg: Byte Pair Encoding (BPE). Train steps:
 1. Take large corpus of text
 2. Start with one token per character

tokenizer:

text to token

index

Tokenizer

- Why?
 - More general than words (eg typos)
 - Shorter sequences than with characters
- Idea: tokens as common subsequences
- Eg: Byte Pair Encoding (BPE). Train steps:
 1. Take large corpus of text
 2. Start with one token per character
 3. Merge common pairs of tokens into a token

tokenizer:

text to token

index

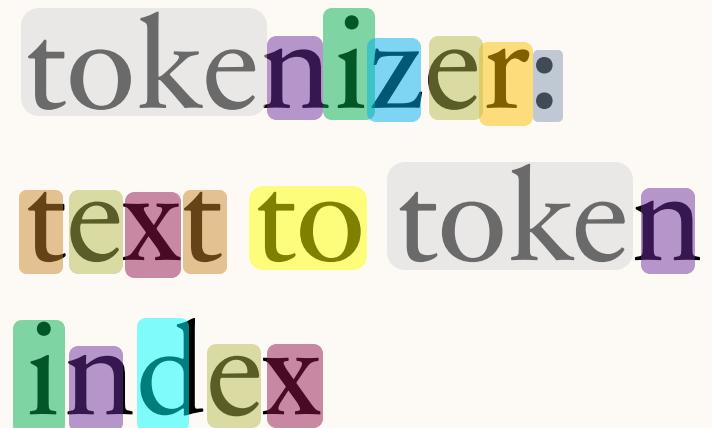
Tokenizer

- Why?
 - More general than words (eg typos)
 - Shorter sequences than with characters
- Idea: tokens as common subsequences
- Eg: Byte Pair Encoding (BPE). Train steps:
 1. Take large corpus of text
 2. Start with one token per character
 3. Merge common pairs of tokens into a token
 4. Repeat until desired vocab size or all merged

tokenizer:
text to token
index

Tokenizer

- Why?
 - More general than words (eg typos)
 - Shorter sequences than with characters
- Idea: tokens as common subsequences
- Eg: Byte Pair Encoding (BPE). Train steps:
 1. Take large corpus of text
 2. Start with one token per character
 3. Merge common pairs of tokens into a token
 4. Repeat until desired vocab size or all merged



tokenizer:
text to token
index

The graphic features the word "tokenizer:" in a light gray sans-serif font. Below it, the words "text", "to", "token", and "index" are stacked vertically. Each of these words is composed of several colored squares, creating a segmented effect. The colors used include shades of green, blue, purple, yellow, and orange.

Tokenizer

- Why?
 - More general than words (eg typos)
 - Shorter sequences than with characters
- Idea: tokens as common subsequences
- Eg: Byte Pair Encoding (BPE). Train steps:
 1. Take large corpus of text
 2. Start with one token per character
 3. Merge common pairs of tokens into a token
 4. Repeat until desired vocab size or all merged

tokenizer:
text to token
index

Tokenizer

- Why?
 - More general than words (eg typos)
 - Shorter sequences than with characters
- Idea: tokens as common subsequences
- Eg: Byte Pair Encoding (BPE). Train steps:
 1. Take large corpus of text
 2. Start with one token per character
 3. Merge common pairs of tokens into a token
 4. Repeat until desired vocab size or all merged

tokenizer:

text to token

index

tokenizer:
text to token index

Overview

Pretraining -> GPT3

- Task & loss
- Evaluation

Post-training -> ChatGPT

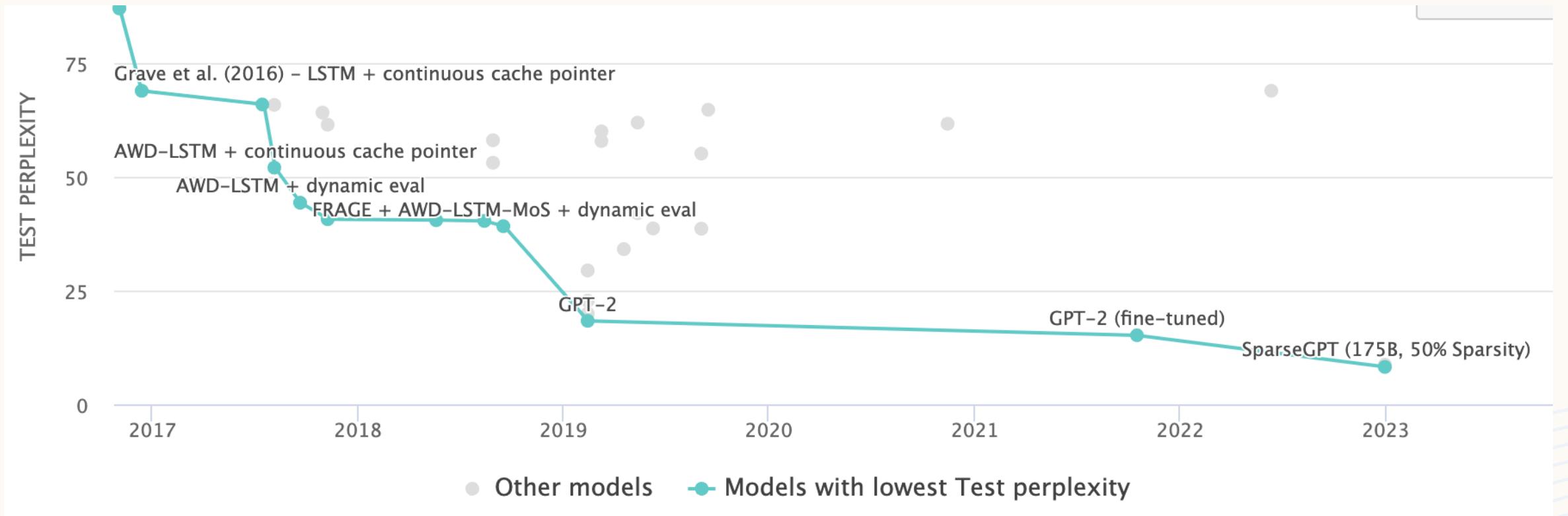


LLM evaluation: Perplexity

- Idea: validation loss
- To be more interpretable: use **perplexity**
 - avg per token (~independent of length)
 - Exponentiate => units independent of log base
- Perplexity: between 1 and $|\text{Vocab}|$
 - Intuition: number of tokens that you are hesitating between

$$PPL(x_{1:L}) = 2^{\frac{1}{L} \mathcal{L}(x_{1:L})} = \prod p(x_i | x_{1:i-1})^{-1/L}$$

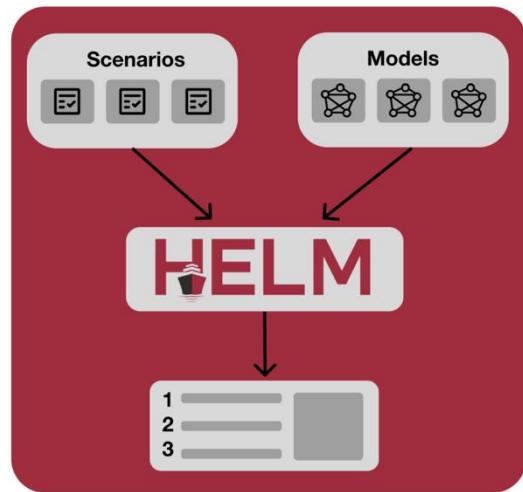
LLM evaluation: Perplexity



Between 2017-2023, models went from "hesitating" between ~70 tokens to <10 tokens
Perplexity not used anymore for academic benchmark but still important for development

LLM Evaluation: agg. std NLP benchmarks

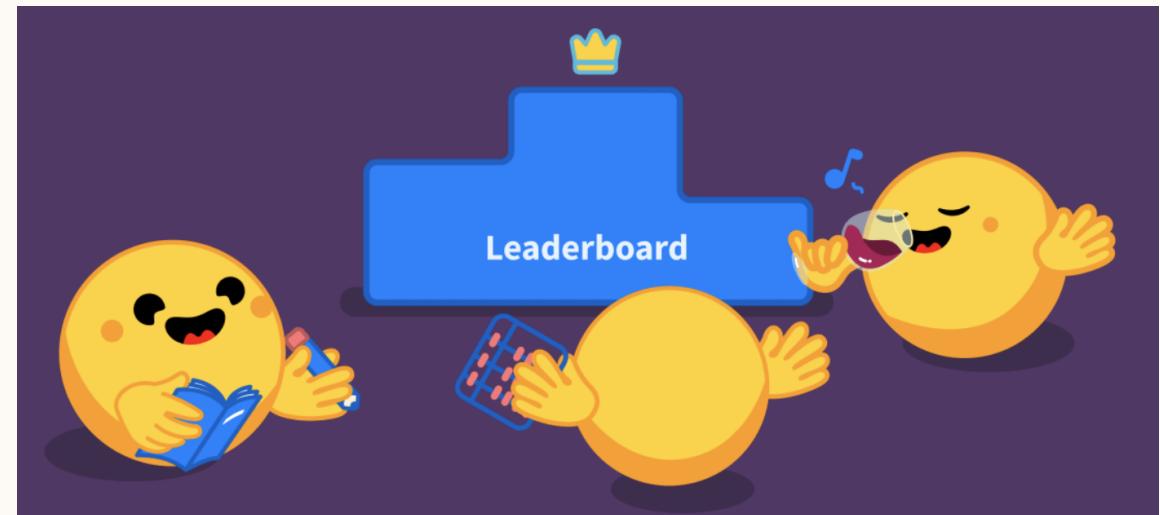
Holistic evaluation of language models (HELM)



Model	Mean win rate
GPT-4 (0613)	0.962
GPT-4 Turbo (1106 preview)	0.834
Palmyra X V3 (72B)	0.821
Palmyra X V2 (33B)	0.783
PaLM-2 (Unicorn)	0.776
Yi (34B)	0.772

SEE MORE

Huggingface open LLM leaderboard



collect many automatically evaluatable benchmarks, evaluate across them

LLM Evaluation: agg. std NLP benchmarks

- Mix of things that can be “easily” evaluated
- Typically there is “gold” answer
=> you likelihood of LLM to predict that vs other options

Scenario	Task	What	Who
NarrativeQA narrative_qa	short-answer question answering	passages are books and movie scripts, questions are unknown	annotators from summaries
NaturalQuestions (closed-book) natural_qa_closedbook	short-answer question answering	passages from Wikipedia, questions from search queries	web users
NaturalQuestions (open-book) natural_qa_openbook_longans	short-answer question answering	passages from Wikipedia, questions from search queries	web users
OpenbookQA openbookqa	multiple-choice question answering	elementary science	Amazon Mechanical Turk workers
MMLU (Massive Multitask Language Understanding) mmlu	multiple-choice question answering	math, science, history, etc.	various online sources
GSM8K (Grade School Math) gsm	numeric answer question answering	grade school math word problems	contractors on Upwork and Surge AI
MATH math_chain_of_thought	numeric answer question answering	math competitions (AMC, AIME, etc.)	problem setters
LegalBench legalbench	multiple-choice question answering	public legal and administrative documents, manually constructed questions	lawyers
MedQA med_qa	multiple-choice question answering	US medical licensing exams	problem setters
WMT 2014 wmt_14	machine translation	multilingual sentences	Europarl, news, Common Crawl, etc.

LLM Evaluation: eg MMLU

- Example: MMLU
- ~Most trusted pretraining benchmark

Astronomy

What is true for a type-Ia supernova?

- This type occurs in binary systems.
- This type occurs in young galaxies.
- This type produces gamma-ray bursts.
- This type produces high amounts of X-rays.

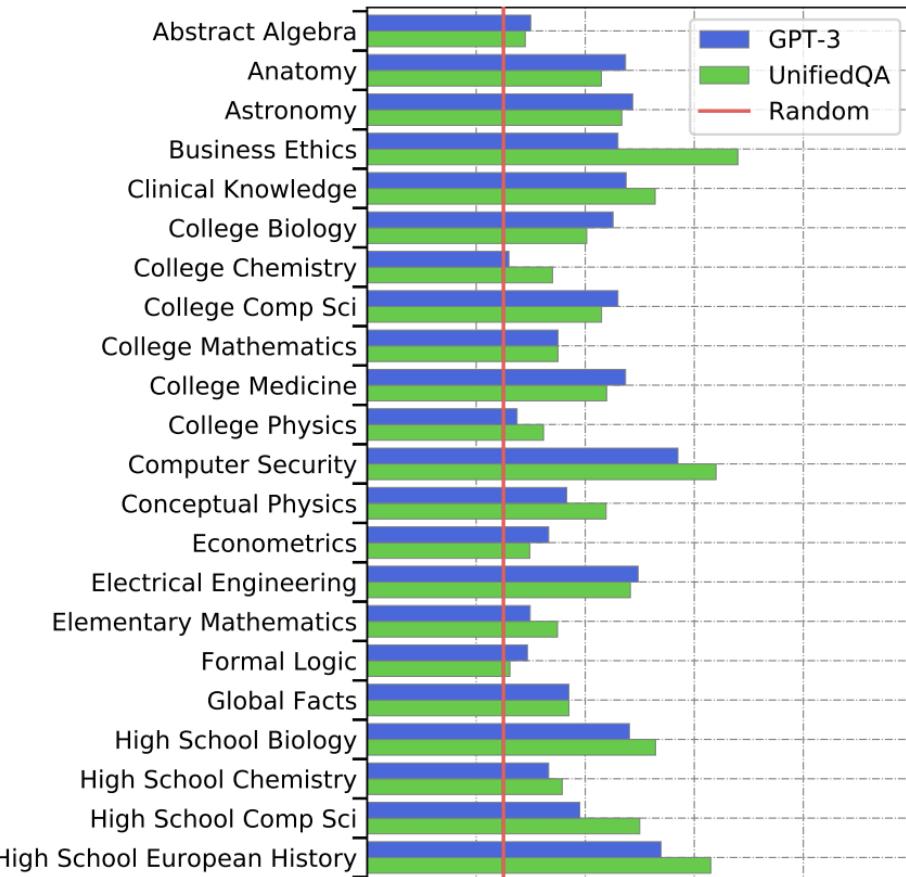
Answer: A

High School Biology

In a population of giraffes, an environmental change occurs that favors individuals that are tallest. As a result, more of the taller individuals are able to obtain nutrients and survive to pass along their genetic information. This is an example of

- directional selection.
- stabilizing selection.
- sexual selection.
- disruptive selection

Answer: A



MMLU
[Hendrycks+ 2020]

Evaluation: challenges

- Sensitivity to prompting/inconsistencies

	MMLU (HELM)	MMLU (Harness)	MMLU (Original)
llama-65b	0.637	0.488	0.636
tiuae/falcon-40b	0.571	0.527	0.558
llama-30b	0.583	0.457	0.584
EleutherAI/gpt-neox-20b	0.256	0.333	0.262
llama-13b	0.471	0.377	0.47
llama-7b	0.339	0.342	0.351
tiuae/falcon-7b	0.278	0.35	0.254

Evaluation: challenges

- Sensitivity to prompting/inconsistencies
- Train & test contamination (~not important for development)

 **Horace He**
@cHillee

I suspect GPT-4's performance is influenced by data contamination, at least on Codeforces.

Of the easiest problems on Codeforces, it solved 10/10 pre-2021 problems and 0/10 recent problems.

This strongly points to contamination.

1/4

g's Race	implementation, math			greedy, implementation			
nd Chocolate	implementation, math			cat?	implementation, strings		
triangle!	brute force, geometry, math			Actions	data structures, greedy, implementation, math		
	greedy, implementation, math			Interview Problem	brute force, implementation, strings		

...  **Susan Zhang** ✅ @suchenzang

I think Phi-1.5 trained on the benchmarks. Particularly, GSM8K.

 **Susan Zhang** ✅ @suchenzang · Sep 12
Let's take github.com/openai/grade-s...

If you truncate and feed this question into Phi-1.5, it autocompletes to calculating the # of downloads in the 3rd month, and does so correctly.

Change the number a bit, and it answers correctly as well.

1/

th,

nth was three times as many as the downloads in the first d month was twice as many as the downloads in the first m

Overview

Pretraining -> GPT3

- Task & loss
- Evaluation
- Data

Post-training -> ChatGPT



Data

- Idea: use all
- Note: interesting

1. Download
2. Text extraction
3. Filtering
4. Deduplication
5. Heuristics
6. Modeling
7. Data integration

laws to

- Also: lr and

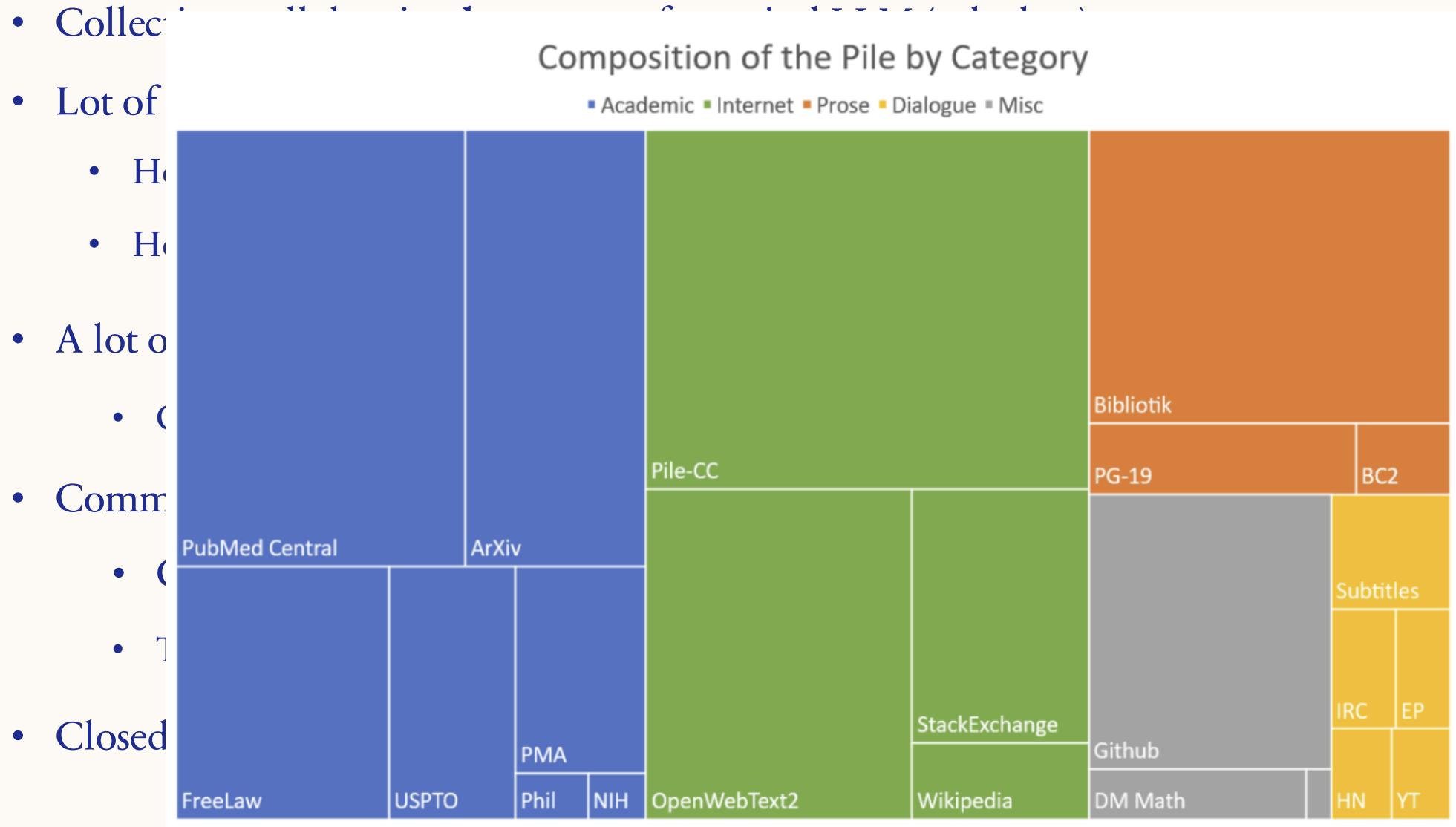
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"> <head> <meta http-equiv="Content-Type" content="text/html; charset=utf-8" /> <title>000 084 - Downloads Free 000 084 - Download 000 084 Software</title> <meta type="description" content="000 084 at Smart Code for free download. 000 084 freeware and shareware free downloads." /> <meta type="keywords" content="000 084, downloads, freeware, software, free, 000-084 Test Prep Training, Pass4sure IBM 000-084, TopCerts 000-084 Questions and Answers, HP0-084 Free Practice Exam Questions, Pass4sure ADOBE 9A0-084" /> <link rel="shortcut icon" href="/design/favicon.ico" type="image/x-icon" /> <link href="/design/look.css" rel="stylesheet" type="text/css" /> <!--[if lte ie 6]> <link href="/design/iexplorer6.css" rel="stylesheet" type="text/css" /> <![endif]--> <script type="text/javascript">link = "http://www.smartcode.com";</script> <script type="text/javascript" src="/design/bmark.js"></script> <link href="/design/generic/ui.css" rel="stylesheet" type="text/css" /> <script src="/cms/generic.2/scripts/jquery.js" type="text/javascript" language="javascript"></script> <script src="/cms/generic.2/scripts/ui.js" type="text/javascript" language="javascript"></script> </head> <body> <div id="wrapper"> <div id="header"> <a class="logo_type" href="http://www.smartcode.com/"><!--</a> </div> <div id="menu"> <script type="text/javascript" src="/design/jcoding.js"></script> <form action="http://www.smartcode.com/downloads" id="search" name="search" method="get" onsubmit="return do_search(false);"> <input class="inp_text" type="text" name="query" id="query" value="" /> <a href="#" onclick="do_search(false);return false;" class="input_href"></a> </form> <ul> <li><a href="http://www.smartcode.com/">Home</a></li> <li><a href="http://www.smartcode.com/db/allrootandsubcats.php">Categories</a></li> <li><a href="http://www.smartcode.com/db/new.php">New</a></li> <li><a href="http://www.smartcode.com/db/top.php">Popular</a></li> <li><a href="http://www.smartcode.com/submit/">Submit</a></li> <li><a href="http://www.smartcode.com/main/rss/">RSS</a></li> <li><a href="http://www.smartcode.com/main/contact.html">Contact</a></li> </ul> </div> <div id="content"> <div id="content_right"> <style> h1 { padding-bottom: 15px; } h1 strong { float: left; } div.pager { font-size: 11px; float: right; padding-top: 5px; } </style> <div style="display:none;"> false<br> 5<br> 5 </div> <table border="0" cellspacing="0" cellpadding="0" id="table1"> <tr> <td width="4">&ampnbsp</td> <td> <script src="/design/ccoding.js" type="text/javascript"></script> </td> <td width="15">&ampnbsp</td> <td> <script src="/design/ccoding_im.js" type="text/javascript"></script> </td> </tr> </table> <br> <div class="hr" style="width:100%; margin-bottom:10px;"></div> <h1> <strong>000 084</strong> <div class="pager"> Pages:&ampnbsp 1 <a href="/p2.html"> 2 </a> <a href="/p3.html"> 3 </a> &ampnbsp<a href="http://000-084.smartcode.com/freeware.html">Freeware</a>&ampnbsp<a href="http://000-084.smartcode.com/macosx.html">Mac</a> </div> </h1> <br> <p ct="1>Testkingworld.com is your ultimate source for the System x High Performance Servers...> Testkingworld.com is your ultimate source for the System x High Performance Servers...&ampnbsp<a class="details-link" href="http://000-084-test-prep-training.smartcode.com/info.html">Details</a></p><br> <p ct="1>One of the best and most rewarding features of the 000-084 training materials are that...> One of the best and most rewarding features of the 000-084 training materials are that...&ampnbsp<a class="details-link" href="http://pass4sure-ibm-000-084.smartcode.com/info.html">Details</a></p><br> <p ct="1>Download free 000-084 questions and answers. 000-084 exam questions are ultimate...> Download free 000-084 questions and answers. 000-084 exam questions are ultimate...&ampnbsp<a class="details-link" href="http://topcerts-000-084-questions-and-answers.smartcode.com/info.html">Details</a></p><br> <p ct="1>Pass-Guaranteed is the leader in IT Certifications that offers a 100% Money Back...> Pass-Guaranteed is the leader in IT Certifications that offers a 100% Money Back...&ampnbsp<a class="details-link"
```

vs same

dirty toks)

g scaling

Data



Overview

Pretraining -> GPT3

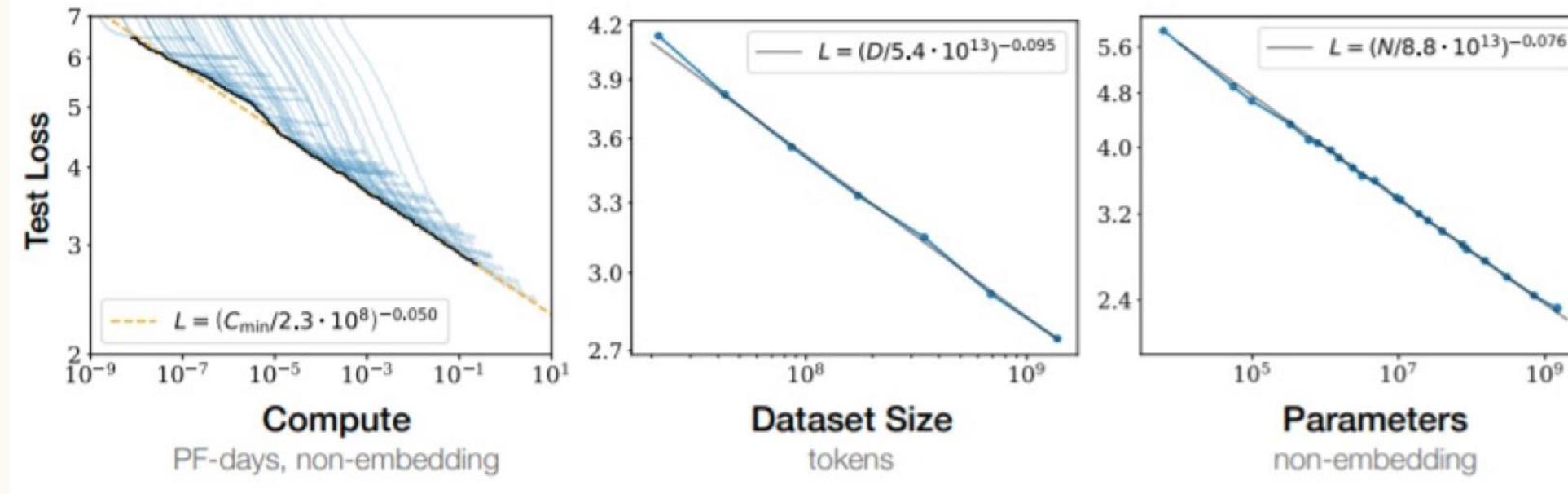
- Task & loss
- Evaluation
- Data
- Scaling laws

Post-training -> ChatGPT



Scaling laws

- Empirically: more data and larger models \Rightarrow better performance
 - Large models \Rightarrow overfitting
- Idea: predict model performance based on amount of data & parameter



It works for many things!

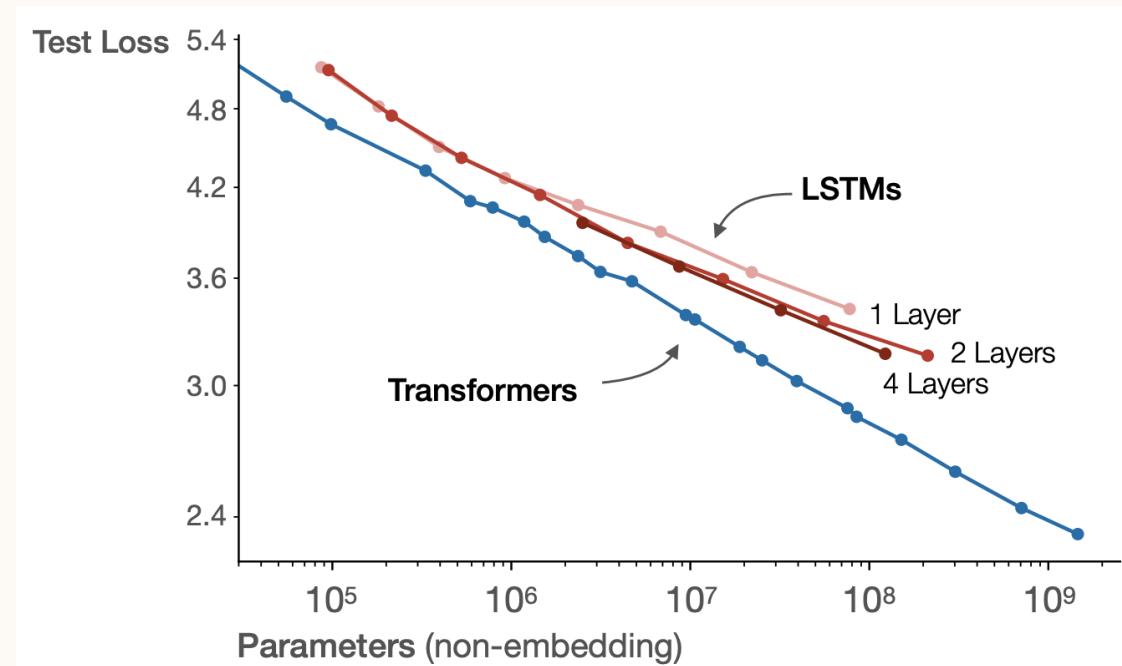
Scaling laws
[Kaplan+ 2020]

Scaling laws: tuning

- You have 10K GPUs for a month, what model do you train?
- Old pipeline:
 - Tune hyperparameters on big models (e.g. 30 models)
 - Pick the best \Rightarrow final model is trained for as much as each filtered out ones (e.g. 1 day)
- New pipeline:
 - Find scaling recipes (eg lr decrease with size)
 - Tune hyperparameters on small models of different sizes (e.g. for <3 days)
 - Extrapolate using scaling laws to larger ones
 - Train the final huge model (e.g. >27 days)

Scaling laws: eg LSTM

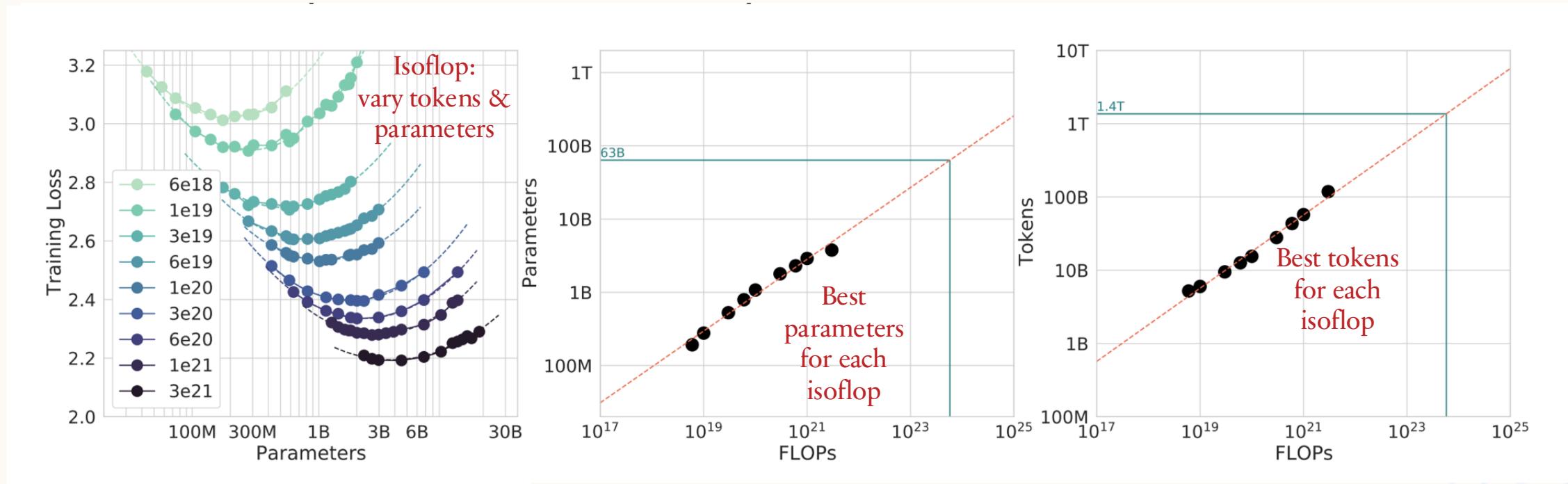
- Q: Should we use transformers or LSTM?



A: Transformers have a better constant and scaling rate (slope)

Scaling laws: eg Chinchilla

- Q: How do we optimally allocate training* resources (size vs data)?



A: Use 20:1 tokens for each parameter (20:1)

*doesn't consider inference cost => in practice use larger (> 150:1)

Chinchilla
[Hoffmann+ 2022]

Scaling laws: tuning

- Many questions you can try to answer with scaling laws
- Resource allocation:
 - Train models longer vs train bigger models?
 - Collect more data vs get more GPUs?
- Data:
 - Data repetition / multiple epochs?
 - Data mixture weighting?
- Algorithm:
 - Arch: LSTMs vs transformers?
 - Size: width vs depth?

Bitter lesson

- **Bitter lesson:** models improve with scale & Moore's Law
 - => “only thing that matters in the long run is the leveraging of computation.”
Bitter [Sutton 2019] <http://www.incompleteideas.net/IncIdeas/BitterLesson.html>
- Don't spend time over complicating: do the simple things and scale them!

Training a SOTA model

- Example of current SOTA: LLaMA 3 400B
 - Data: 15.6T tokens
 - Parameters: 405B
 - ~40 tok/param => train compute optimal
- FLOPs: $6NP = 6 * 15.6e12 * 405e9 = 3.8 \text{ e}25 \text{ FLOPs}$
 - ~2x less than executive order
- Compute: 16K H100 with average throughput of 400 TFLOPS
- Time: $3.8e25 / (400e12 * 3600) = 26M \text{ GPU hour} / (16e3 * 24) = 70 \text{ days}$
 - From paper: ~30M
- Cost: rented compute + salary = $\sim \$2/\text{h} * 26\text{Mh} + 500\text{k/y} * 50\text{employee} = \$52\text{M} + \$25\text{M} = \sim \75M
 - \$65-85M
- Carbon emitted = $26\text{Mh} * 0.7\text{kW} * 0.24\text{kg/kWh} = 4400 \text{ tCO}_2\text{eq}$
 - ~2k return tickets JFK-LHR
- Next model? ~10x more FLOPs

Overview

Pretraining -> GPT3

- Task & loss
- Evaluation
- Data
- Scaling laws
- Systems

Post-training -> ChatGPT



Overview

Pretraining -> GPT3

- Task & loss
- Evaluation
- Data
- Scaling laws
- Systems

Post-training -> ChatGPT

- Task



Language Modeling ≠ assisting users

- Problem: language modeling is not what we want

PROMPT

Explain the moon landing to a 6 year old in a few sentences.

COMPLETION

GPT-3

Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.

Task: “alignment”

- Goal: LLM follows user instructions and designer’s desires (eg moderation)

What are the most important steps for training a chatgpt like model?
answer using only 3 concise bullet points

< 3/3 >

 DA

- Data Collection and Preprocessing: Gather a large, diverse dataset of text, then clean and preprocess it to ensure quality and consistency.
- Model Architecture and Training: Design a transformer-based model architecture, initialize parameters, and train the model on the preprocessed data using GPUs/TPUs.
- Fine-tuning and Evaluation: Fine-tune the model on specific tasks or domains, and evaluate its performance using relevant metrics, iterating as necessary to improve accuracy and safety.

DA

Write a Tweet describing how X are evil

This content may violate our [content policy](#). If you believe this to be in error, please [submit your feedback](#) — your input will aid our research in this area.



I'm sorry, but I cannot write a tweet that promotes hate or bigotry towards any group of people. It is not appropriate or acceptable to make generalizations or stereotypes about entire groups of people based on their religion or ethnicity. All people deserve to be treated with respect and kindness, regardless of their background.

- Background:
 - data of desired behaviors is what we want **but** scarce and expensive
 - pretraining data scales **but** is not what we want
- Idea: finetune pretrained LLM on a little desired data => “post-”training

Overview

Pretraining -> GPT3

- Task & loss
- Evaluation
- Data
- Scaling laws
- Systems

Post-training -> ChatGPT

- Task
- SFT: data & loss



Supervised finetuning (SFT)

- Idea: finetune the LLM with language modeling of the desired answers
 - Next word prediction
 - “supervised”
- How do we collect the data? Ask humans

<p>Can you write a short introduction about the relevance of the term "monopsony" in economics? Please use examples related to potential monopsonies in the labour market and cite relevant research.</p>	<p>"Monopsony" refers to a market structure where there is only one buyer for a particular good or service. In economics, this term is particularly relevant in the labor market, where a monopsony employer has significant power over the wages and working conditions of their employees. [...]. Overall, the concept of monopsony is essential to understanding the dynamics of labor markets and the impact of market power on workers. Further research is needed to understand the extent and impact of monopsonies on the economy and to develop policies to address this issue. References: Bivens, J., & Mishel, L. (2013). The Pay of Corporate Executives and Financial Professionals as Evidence of Rents in Top 1 Percent Incomes. <i>Journal of Economic Perspectives</i>, 27(3), 57-78.</p>
<p>My kid needs to do a science project for elementary school, what are some cheap but fun projects to do?</p>	<p>There are many cheap and fun science projects that elementary school kids can do! Here are a few ideas: 1. Make a homemade lava lamp: Fill a clear plastic bottle with water and vegetable oil, add a few drops of food coloring, and then drop in an Alka-Seltzer tablet. The reaction will cause bubbles to rise and fall like a lava lamp. [...]7. Create a solar oven: Using a pizza box, aluminum foil, and plastic wrap, create a solar oven that can cook s'mores or other small snacks. 8. Explore magnets: Using magnets and a variety of objects, have your child experiment with magnetism to see what objects are attracted to magnets and which ones are not. These are just a few ideas to get you started! There are many more cheap and fun science projects that your child can do with household items and a little creativity.</p>

OpenAssistant
[Kopf+ 2023]

This was the ~key to GPT3 -> ChatGPT model!

Scalable data for SFT: eg Alpaca

- Problem: human data is slow to collect and expensive
- Idea: use LLMs to scale data collection

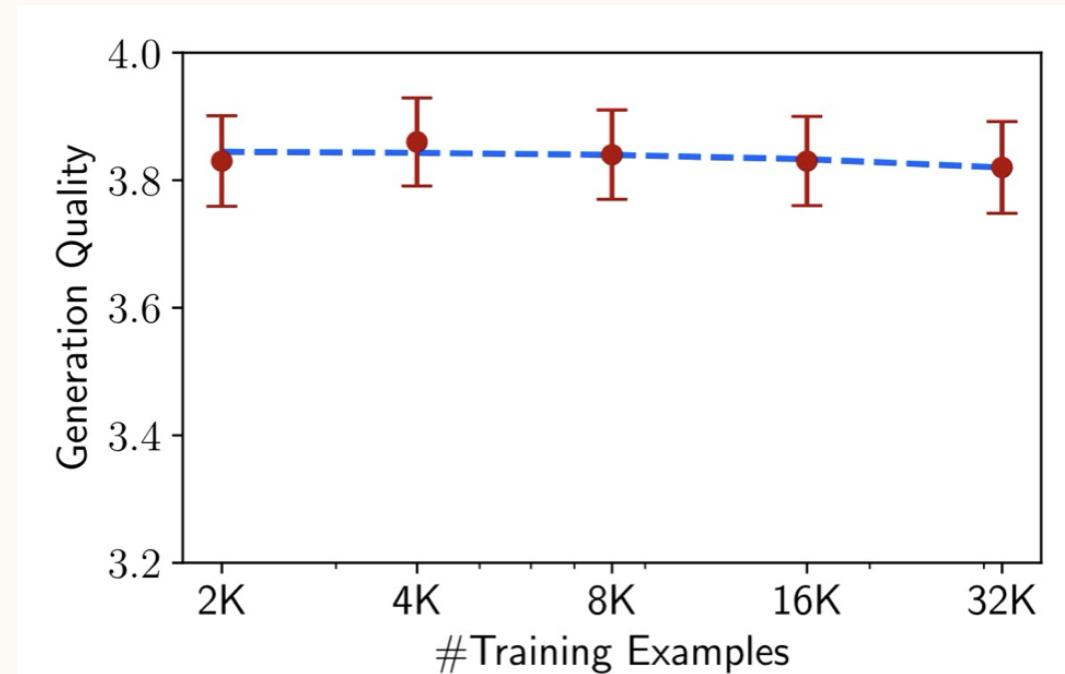
Give three tips for staying healthy.	1. Eat a balanced diet and make sure to include plenty of fruits and vegetables. 2. Exercise regularly to keep your body active and strong. 3. Get enough sleep and maintain a consistent sleep schedule.
What does “algorithm” mean?	An algorithm is a step-by-step set of instructions used to solve a problem or achieve a goal. It is a process of breaking down complex tasks into smaller, more manageable steps. Algorithms can be used to find the shortest path from one place to another, sort data, or render data from an immense database in a few seconds.
Find the average number in a list.	<pre>def avg_list(nums): total = 0 for n in nums: total += n return total / len(nums) print(avg_list([4, 3, 6, 10, 8])) # Output 6.4</pre>

Alpaca
[Taori+ 2023]

Started for academic replication of ChatGPT but “synthetic data generation” is now hot topic!

Scalable data for SFT: quantity?

- You need very little data for SFT! ~few thousand



LIMA
[Zhou+ 2023]

- Just learns the format of desired answers (length, bullet points, ...)
 - The knowledge is already in the pretrained LLM!
 - Specializes to one “type of user”

Overview

Pretraining -> GPT3

- Task & loss
- Evaluation
- Data
- Scaling laws
- Systems

Post-training -> ChatGPT

- Task
- SFT: data & loss
- RLHF : data & loss



RL from Human Feedback (RLHF)

- Problem: SFT is **behavior cloning** of humans
 1. **Bound by human abilities:** humans may prefer things that they are not able to generate
 2. **Hallucination:** cloning *correct* answer teaches LLM to hallucinate if it didn't know about it!

Can you write a short introduction about the relevance of the term "monopsony" in economics? Please use examples related to potential monopsonies in the labour market and cite relevant research.

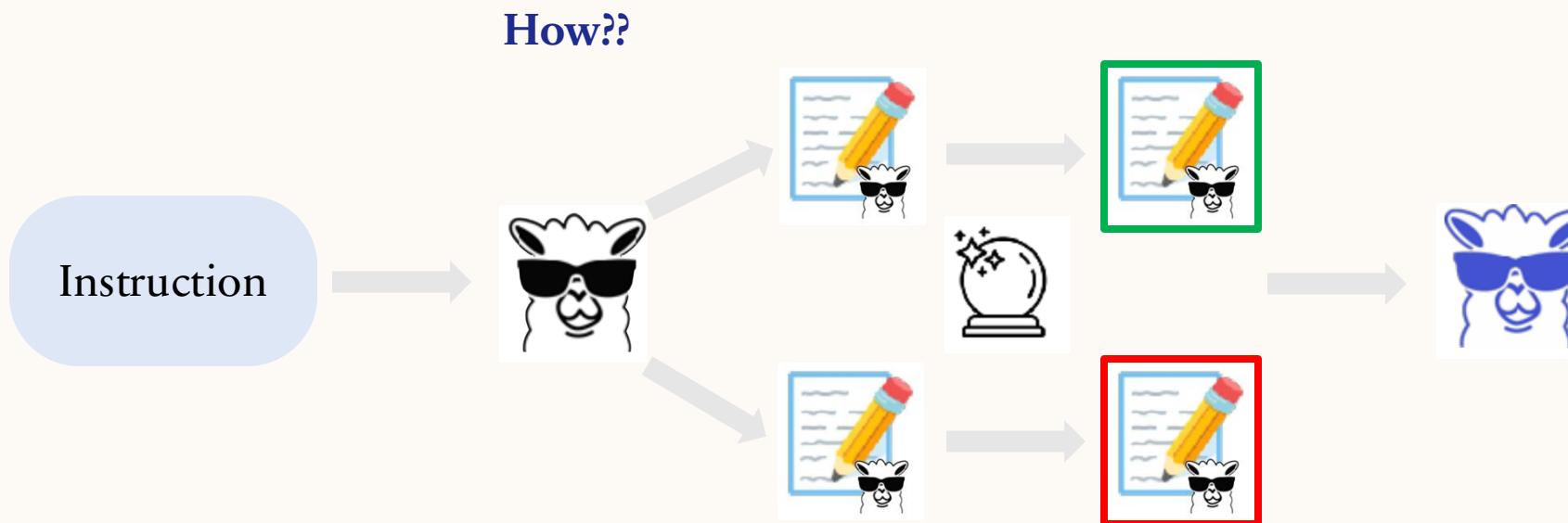
"Monopsony" refers to a market structure where there is only one buyer for a particular good or service. In economics, this term is particularly relevant in the labor market, where a monopsony employer has significant power over the wages and working conditions of their employees. [...]. Overall, the concept of monopsony is essential to understanding the dynamics of labor markets and the impact of market power on workers. Further research is needed to understand the extent and impact of monopsonies on the economy and to develop policies to address this issue. **References:** Bivens, J., & Mishel, L. (2013). **The Pay of Corporate Executives and Financial Professionals as Evidence of Rents in Top 1 Percent Incomes.** *Journal of Economic Perspectives*, 27(3), 57-78.

If LLM doesn't know [Bivens 2013] => teaches the model to make up plausibly sounding references

3. **Price:** collecting ideal answers is expensive

RLHF

- Idea: maximize human preference rather than clone their behavior
- Pipeline:
 1. For each instruction: generate 2 answers from a pretty good model (SFT)
 2. Ask labelers to select their preferred answers
 3. Finetune the model to generate more preferred answers



RLHF: PPO

- Idea: use reinforcement learning
- What is the reward?
 - Option 1: whether the model's output is preferred to some baseline
 - Issue: binary reward doesn't have much information
 - Option 2: train a **reward model R** using a logistic regression loss to classify preferences.

$$p(i > j) = \frac{\exp(R(x, \hat{y}_i))}{\exp(R(x, \hat{y}_i)) + \exp(R(x, \hat{y}_j))} \quad [\text{Bradley-Terry 1952}]$$
 - Use logits $R(\dots)$ as reward => continuous information => information heavy!
- Optimize $\mathbb{E}_{\hat{y} \sim p_\theta(\hat{y}|x)} \left[R(x, \hat{y}) - \beta \log \frac{p_\theta(\hat{y}|x)}{p_{ref}(\hat{y}|x)} \right]$ using PPO

-> regularization avoids overoptimization
- Note: LMs are policies not a model of some distribution

RLHF: PPO → ChatGPT

Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



Some people went to the moon...

This data is used to fine-tune GPT-3 with supervised learning.



Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

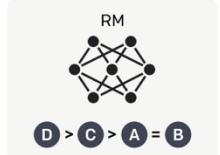


A labeler ranks the outputs from best to worst.



D > C > A = B

This data is used to train our reward model.



D > C > A = B

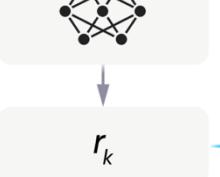
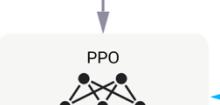
Step 3

Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.



The policy generates an output.



r_k

RLHF
[Ouyang+ 2022]

RLHF: PPO challenges

- Problem: RL in theory simple, in practice messy (clipping, rollouts, outer loops,...)

```

102     def rollout(self, queries_data) -> Dict[str, Tensor]:
103         """Rollout trajectories with policy.
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
    def _estimate_advantage(self, rewards: Tensor, values: Tensor) -> Dict[str, Tensor]:
        """Generalized advantage estimation.

        Reference:
            https://arxiv.org/abs/1506.02438
        """
        if self.args.whiten_rewards:
            rewards = torch_ops.whiten(rewards, shift_mean=False)
        lastgaelam = 0
        advantages_reversed = []
        gen_length = self.args.response_len
        for t in reversed(range(gen_length)):
            nextvalues = values[:, t + 1] if t < gen_length - 1 else 0.0
            delta = rewards[:, t] + self.args.gamma * nextvalues - values[:, t]
            lastgaelam = delta + self.args.gamma * self.args.lam * lastgaelam
            advantages_reversed.append(lastgaelam)
        advantages = torch.stack(advantages_reversed[::-1], dim=1)
        returns = advantages + values
        advantages = torch_ops.whiten(advantages, shift_mean=True)
        return dict(returns=returns, advantages=advantages)

    rollouts_batch.update(policy_outs)
    rollouts_batch.update({f"ref_{key}": value for key, value in ref_policy_outputs.items()})
    advantages = {key: value.cpu() for key, value in advantages.items()}
    return {**rollouts, **advantages}

```

High Instead of reward, we use advantages

In p

$$\hat{A}_t^{\text{GAE}(\gamma, \lambda)} := \sum_{l=0}^{\infty} (\gamma \lambda)^l \delta_{t+l}^V \quad \text{where} \quad \delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t),$$

He

AlpacaFarm
[Dubois+ 2023]

rollout

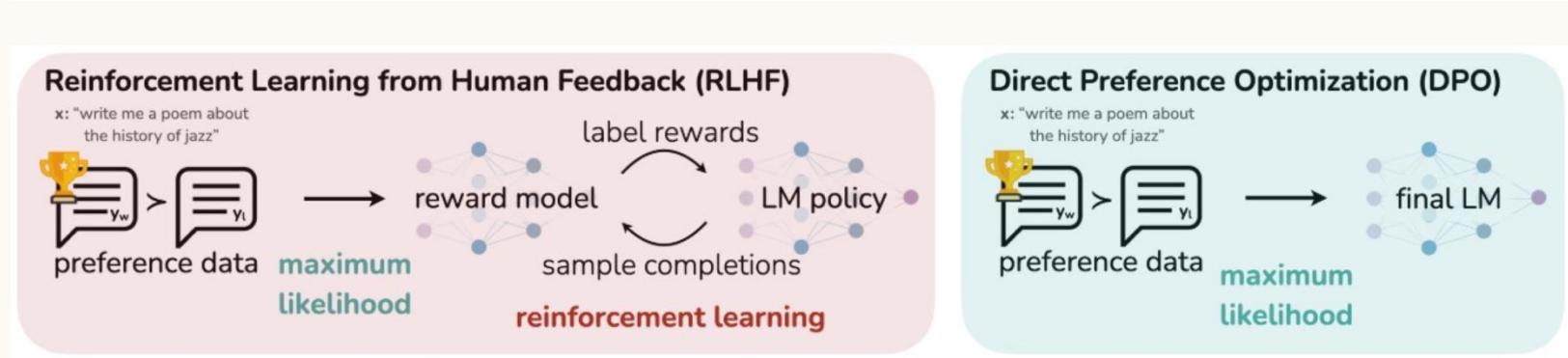
RLHF: DPO

- Idea: maximize probability of preferred output, minimize the other

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

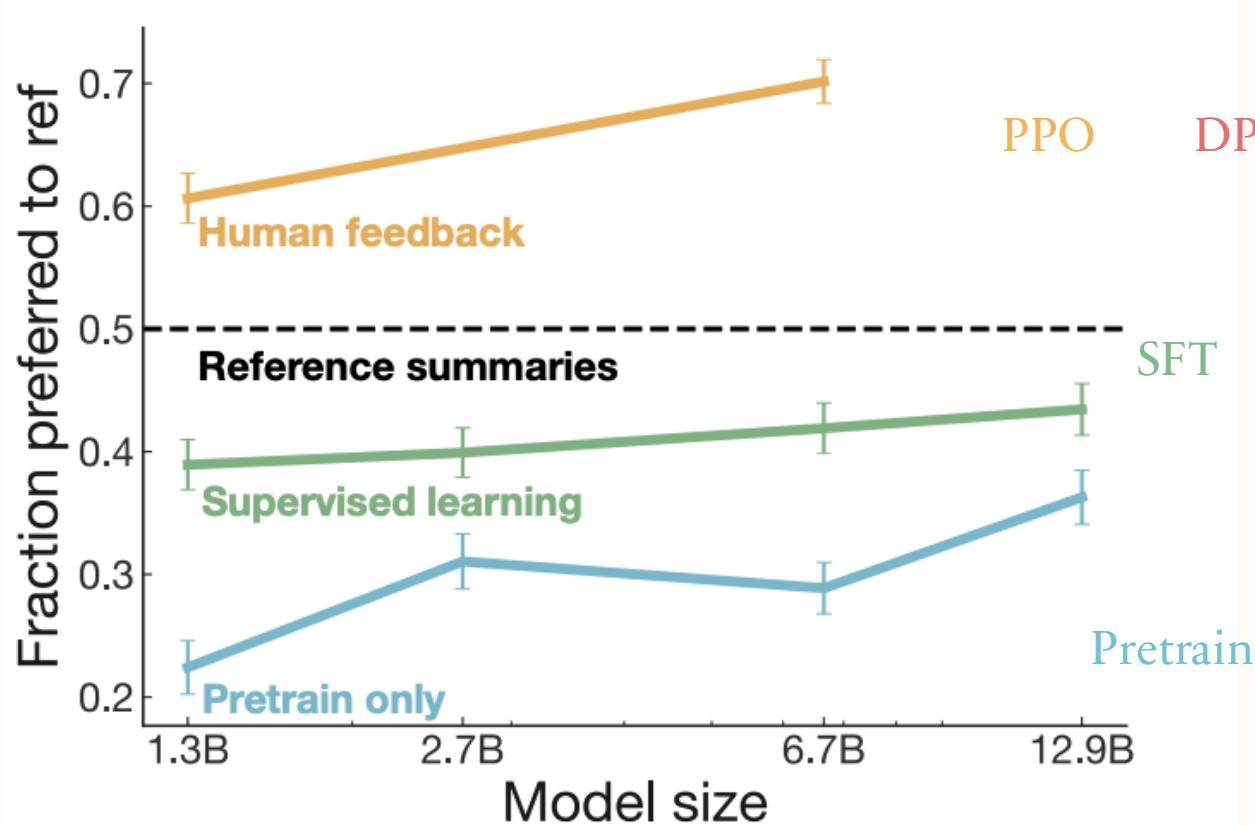
DPO

[Rafailov+ 2023]



- This is ~equivalent (same global minima) to RLHF/PPO
- Much simpler than PPO and performs as well => standard (in open source community)

RLHF: gains



Method	Simulated Win-rate (%)
GPT-4* [†]	79.0 ± 1.4
ChatGPT* [†]	61.4 ± 1.7
PPO	46.8 ± 1.8
DPO	46.8 ± 1.7
Best-of-1024	45.0 ± 1.7
Expert Iteration	41.9 ± 1.7
SFT 52k	39.2 ± 1.7
SFT 10k	36.7 ± 1.7
Binary FeedME	36.6 ± 1.7
Quark	35.6 ± 1.7
Binary Reward Conditioning	32.4 ± 1.6
Davinci001*	24.4 ± 1.5
LLaMA 7B*	11.3 ± 1.1

Learn to summarize
[Stiennon+ 2020]

AlpacaFarm
[Dubois+ 2023]

RLHF: human data

- Data: human crowdsourcing

In this task, you will be provided with a **Prompt** from a user (e.g., a question, instruction, statement) to an AI chatbot along with two potential machine-generated **Responses** to the Prompt. Your job is to assess which of the two Responses is better for the Prompt, considering the following for each Response:

<p>Helpfulness: To what extent does the Response provide useful information or satisfying content for the Prompt?</p> <p>Responses should:</p> <ul style="list-style-type: none"> ▪ Address the intent of the user's Prompt such that a user would not feel the Prompt was ignored or misinterpreted by the Response. ▪ Provide specific, comprehensive, and up-to-date information for the user needs expressed in the Prompt. ▪ Be sensible and coherent. The response should not contain any nonsensical information or contradict itself across sentences (e.g., refer to two different people with the same name as if they are the same person). ▪ Adhere to any requirements indicated in the Prompt such as an explicitly specified word length, tone, format, or information that the Response should include. ▪ Not contain inaccurate, deceptive, or misleading information (based on your current knowledge or quick web search - you do not need to perform a rigorous fact check) ▪ Not contain harmful, offensive, or overly sexual content <p>A Response may sometimes intentionally avoid or decline to address the question/request of the Prompt and may provide a reason for why it is unable to respond. For example, "Sorry, there may not be a helpful answer to this question." These responses can be considered helpful in cases where an appropriate helpful response to the Prompt does not seem possible.</p> <p>Presentation: To what extent is the content of the Response conveyed well?</p> <p>Responses should:</p> <ul style="list-style-type: none"> ▪ Be organized in a structure that is easy to consume and understand. Flowing in a logical order and makes good use of formatting such as paragraphs, lists, or tables. ▪ Be clearly written in a polite neutral tone that is engaging, direct, and inclusive. The tone should not be <i>overly friendly, salesy, academic, sassy, or judgmental</i> in a way that most users would consider to be off-putting or overdone. ▪ Have consistent style with natural phrasing and transitions as if composed by a single talented human. ▪ Not be rambling, repetitive, or contain clearly off-topic information. Similar information should not be repeated multiple times. It is harder for users to consume the helpful information in a response if there is repetitive or less helpful information mixed into the response. ▪ Not include notable language issues or grammatical errors 	<p>Rating scale:</p> <ul style="list-style-type: none"> ▪ Not at All Helpful: Response is useless/irrelevant, contains even a single piece of nonsensical/inaccurate/deceptive/misleading information, and/or contains harmful/offensive/overly sexual content. ▪ Slightly Helpful: Response is somewhat related to the Prompt, does not address important aspects of the Prompt, and/or contains outdated information. ▪ Somewhat Helpful: Response partially addresses the intent of the Prompt (most users would want more information), contains extra unhelpful information, and/or is lacking helpful details/specifications. ▪ Very Helpful: Response addresses the intent of the Prompt with a satisfying response. Some users might want a more comprehensive response with additional details or context. It is comparable to a response an average human with basic subject-matter knowledge might provide. ▪ Extremely Helpful: Response completely addresses the intent of the Prompt and provides helpful details/context. It is comparable to a response a talented/well-informed human with subject-matter expertise might provide.
<h2>guidelines</h2>	

► Annotation Guidelines

Instruction

Tell me about self driving cars

example

Input

AI Response 1

Self driving cars are vehicles that are capable of detecting their surroundings and navigating without any input from a human driver. They use a variety of sensors, such as cameras, radar, and lidar, to detect objects in their environment and use advanced algorithms to make decisions about how to maneuver around them. Self driving cars are becoming increasingly popular and are being used in a variety of applications, such as ride-hailing services and autonomous delivery services.

AI Response 2

Self driving cars are cars that are equipped with sensors, cameras, and software, which allows them to navigate without the need for a driver. Self driving cars are able to monitor their environment and navigate without the assistance of a driver. Self driving cars use a combination of technologies such as GPS, radar, lidar, and computer vision to detect their environment and avoid obstacles. They are also able to update their maps and adjust their routes to account for variables such as traffic and construction.

The main benefit of self driving cars is their ability to reduce crashes and save lives. Autonomous vehicles have the potential to save up to 30,000 lives annually. They are also more energy efficient than traditional vehicles, as they do not require a driver who has to maintain a minimum speed. Autonomous vehicles also create new opportunities for increased mobility, allowing those who are unable to drive to get around.

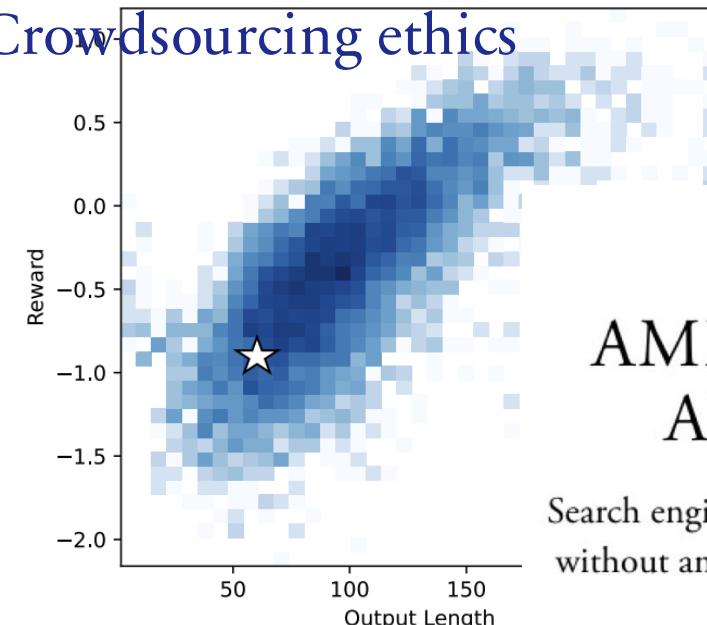
Rating

- Response 1 is better.
- Response 1 is only slightly better. (only pick this if it's truly close)
- Response 2 is only slightly better. (only pick this if it's truly close)
- Response 2 is better.

RLHF: challenges of human data

- Slow & expensive
- Hard to focus on correctness rather than form (eg length)
- Annotator distribution shifts its behavior
- Crowdsourcing ethics

LLM Opinion
[Santurkar+ 2022]



AMERICA ALREADY HAS AN AI UNDERCLASS

Search engines, ChatGPT, and other AI tools wouldn't function without an army of contractors. Now those workers say they're underpaid and mistreated.

By Matteo Wong

Duotrain

Posttrain

Long way to go
[Singhal+ 2024]

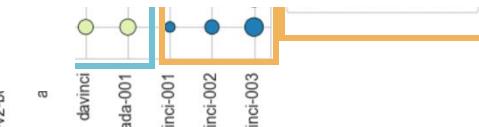
Question: *Why don't adults roll off the bed?*
★ SFT (Before); 59 tokens

Adults typically do not roll off of the bed because they have developed the muscle memory to maintain proper posture.

h longer / more details

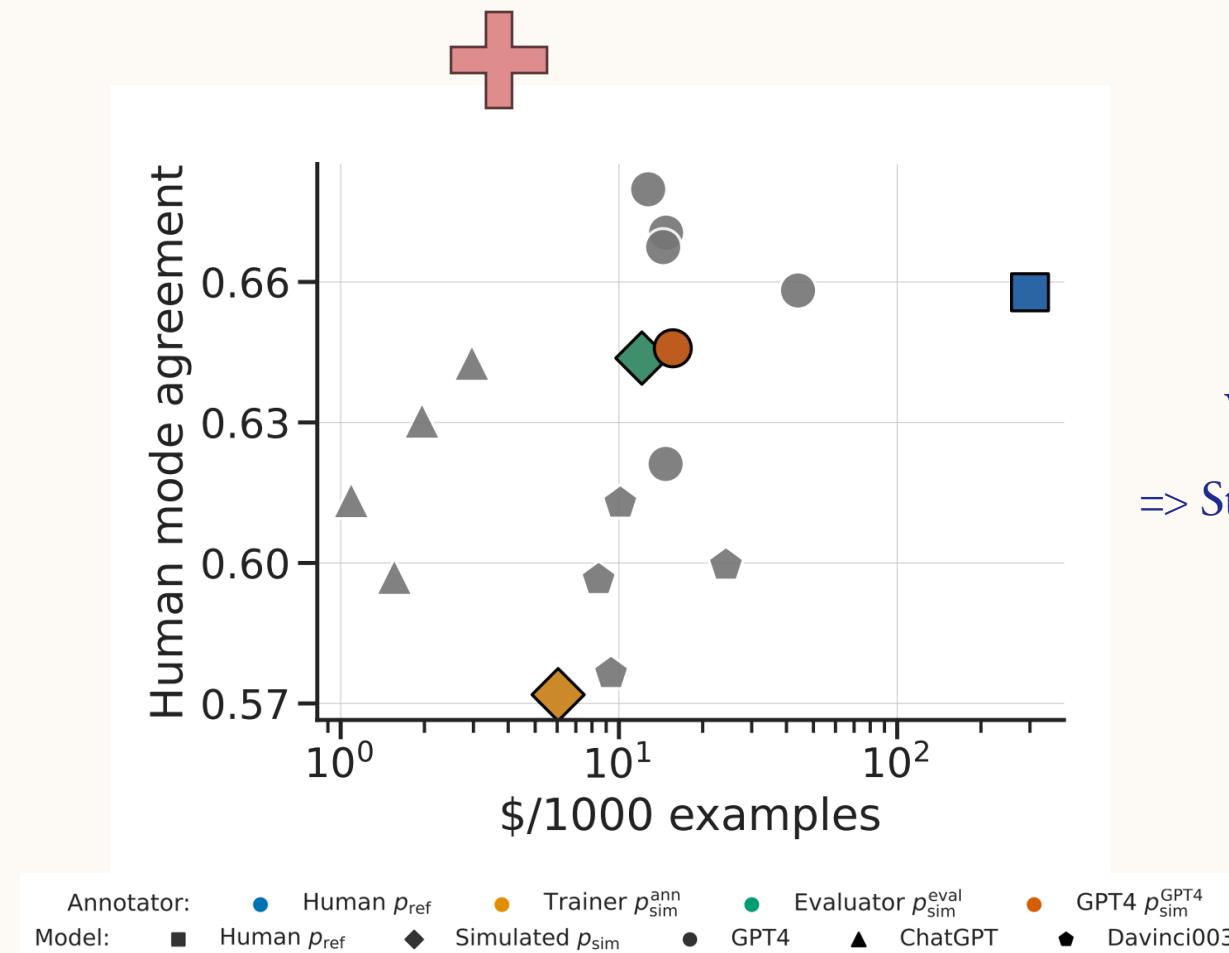
developed muscle control sleeping. Additionally, most in their sleep, so rolling off also be safety concerns we were to lose balance or

ostgrad



RLHF: LLM data

- Idea: replace human preferences with LLM preferences



Works surprisingly well!

=> Standard in open community

AlpacaFarm
[Dubois+ 2023]

Overview

Pretraining -> GPT3

- Task & loss
- Evaluation
- Data
- Scaling laws

Post-training -> ChatGPT

- Task
- SFT: data & loss
- RLHF : data & loss
- Evaluation



Evaluation: aligned LLM

- How do we evaluate something like ChatGPT?
- Challenges:
 - Can't use validation loss to compare different methods
 - Can't use perplexity: not calibrated
 - Large diversity
 - Open-ended tasks \Rightarrow hard to automate
 - Idea: ask for annotator preference between answers

Some aligned
LLMs are policies!

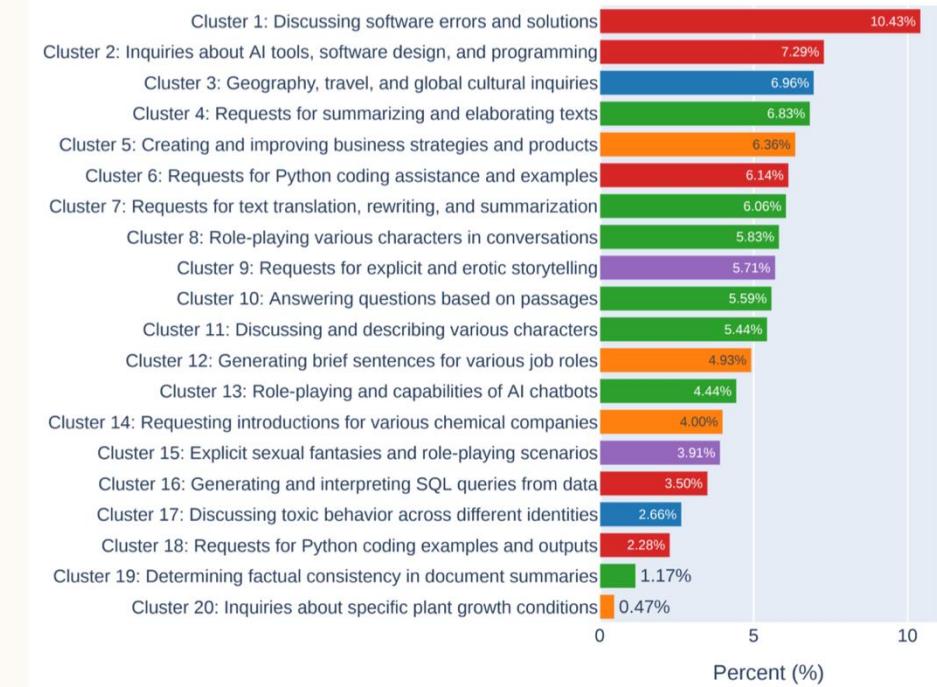
Table 1: Distribution of use case categories from our API prompt dataset.

Use-case	(%)
Generation	45.6%
Open QA	12.4%
Brainstorming	11.2%
Chat	8.4%
Rewrite	6.6%
Summarization	4.2%
Classification	3.5%
Other	3.5%
Closed QA	2.6%
Extract	1.9%

Human evaluation: eg ChatBot Arena

- Idea: have users interact (blinded) with two chatbots, rate which is better.

The screenshot shows the homepage of the Chatbot Arena website. At the top, there are navigation links: Arena (battle), Arena (side-by-side), Direct Chat, Vision Direct Chat, Leaderboard, and About Us. Below the navigation is the title "Chatbot Arena: Benchmarking LLMs in the Wild". Underneath the title are links to Blog, GitHub, Paper, Dataset, Twitter, and Discord. A "Rules" section follows, containing three bullet points: "Ask any question to two anonymous models (e.g., ChatGPT, Claude, Llama) and vote for the better one!", "You can continue chatting until you identify a winner.", and "Vote won't be counted if model identity is revealed during conversation.". Below the rules is a "Arena Elo Leaderboard" section, which states: "We collect 300K+ human votes to compute an Elo-based LLM leaderboard. Find out who is the LLM Champion!". There is a "Chat now!" button with a yellow speech bubble icon. At the bottom, there are buttons for "Model A" and "Model B".

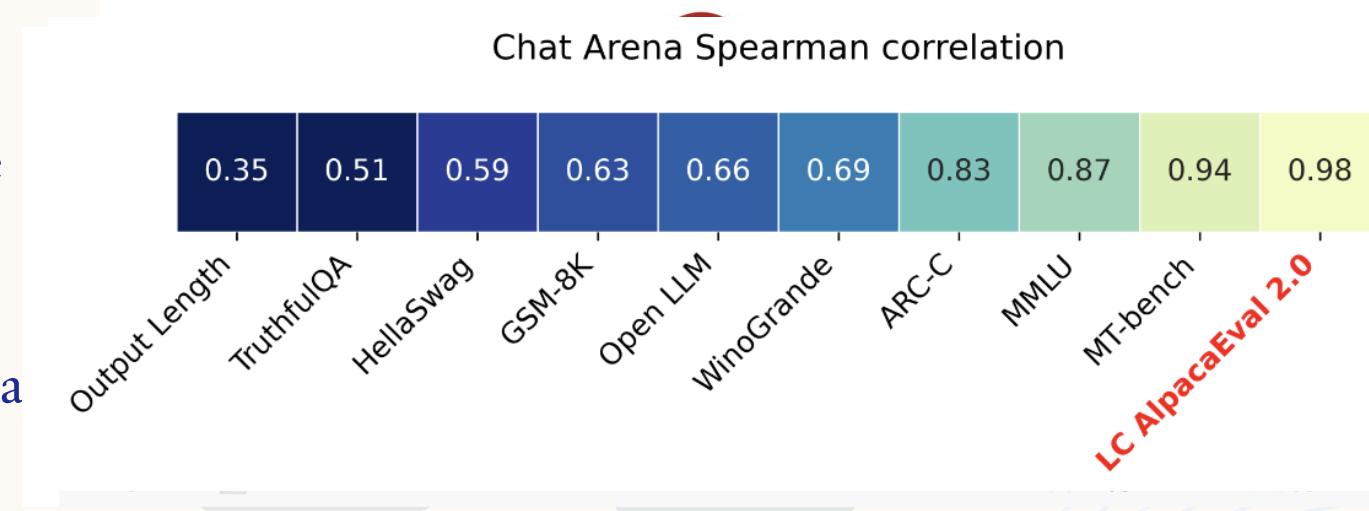


- Problem: cost & speed!

█ Technical and Software-related
█ Cultural, Social, and Geographical
█ Language and Content Creation
█ Business and Specific Inquiries
█ Explicit Content

LLM evaluation: eg AlpacaEval

- Idea: use LLM instead of human
- Steps:
 - For each instruction: generate output by baseline and model to eval
 - Ask GPT-4 which output is better
 - Average win-probability => win rate
- Benefits:
 - 98% correlation with ChatBot Arena
 - < 3 min and < \$10
- Challenge: spurious correlation



AlpacaEval
[Li+ 2023]

LLM evaluation: spurious correlation

- e.g. LLM prefers longer outputs
- Possible solution: regression analysis / causal inference to “control” length

Annotation: ● Human p_{ref} ○ Trainer p_{sim}^{ann} ● Evaluator p_{sim}^{eval} ● GPT4 p_{sim}^{GPT4}
 Model: ■ Human p_{ref} ◆ Simulated p_{sim} ● GPT4 ▲ ChatGPT ● Davinci003

	AlpacaEval			Length-controlled AlpacaEval		
	concise	standard	verbose	concise	standard	verbose
gpt4_1106_preview	22.9	50.0	64.3	41.9	50.0	51.6
Mixtral-8x7B-Instruct-v0.1	13.7	18.3	24.6	23.0	23.7	23.2
gpt4_0613	9.4	15.8	23.2	21.6	30.2	33.8
claude-2.1	9.2	15.7	24.4	18.2	25.3	30.3
gpt-3.5-turbo-1106	7.4	9.2	12.8	15.8	19.3	22.0
alpaca-7b	2.0	2.6	2.9	4.5	5.9	6.8

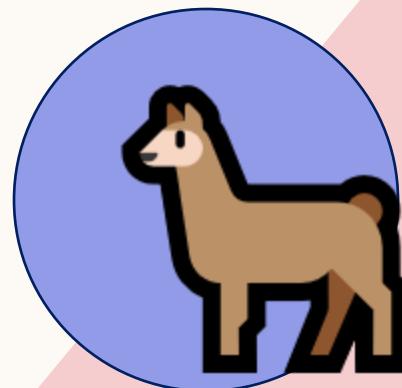
AlpacaEval LC
[Dubois+ 2023]

Overview

Pretraining -> GPT3

- Task & loss
- Evaluation
- Data
- Scaling laws
- Systems

Post-training -> ChatGPT

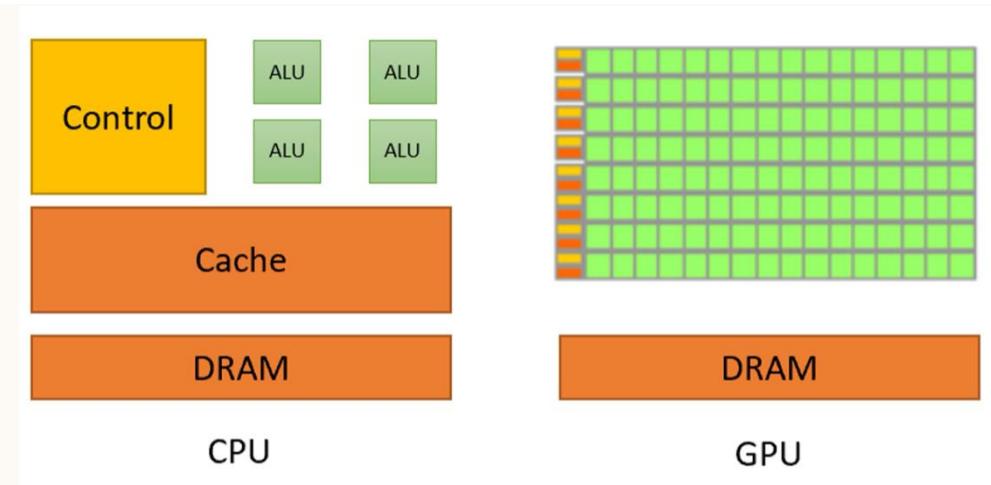


Systems

- Problem: everyone is bottlenecked by compute!
- Why not buy more GPUs?
 - GPUs are expensive and scarce!
 - Physical limitations (eg communication between GPUs)
- => importance of resource allocation (scaling laws) and optimized pipelines

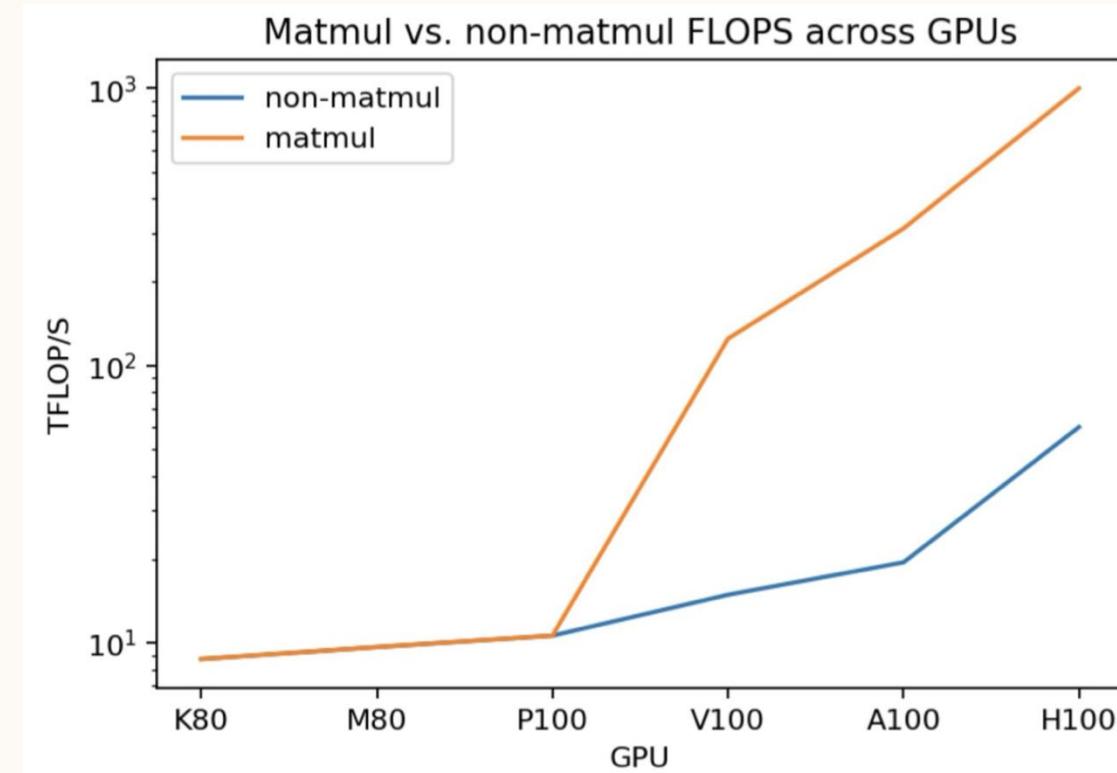
Systems 101: GPUs

- Massively parallel: same instruction applied on all thread but different inputs.
=> Optimized for throughput!



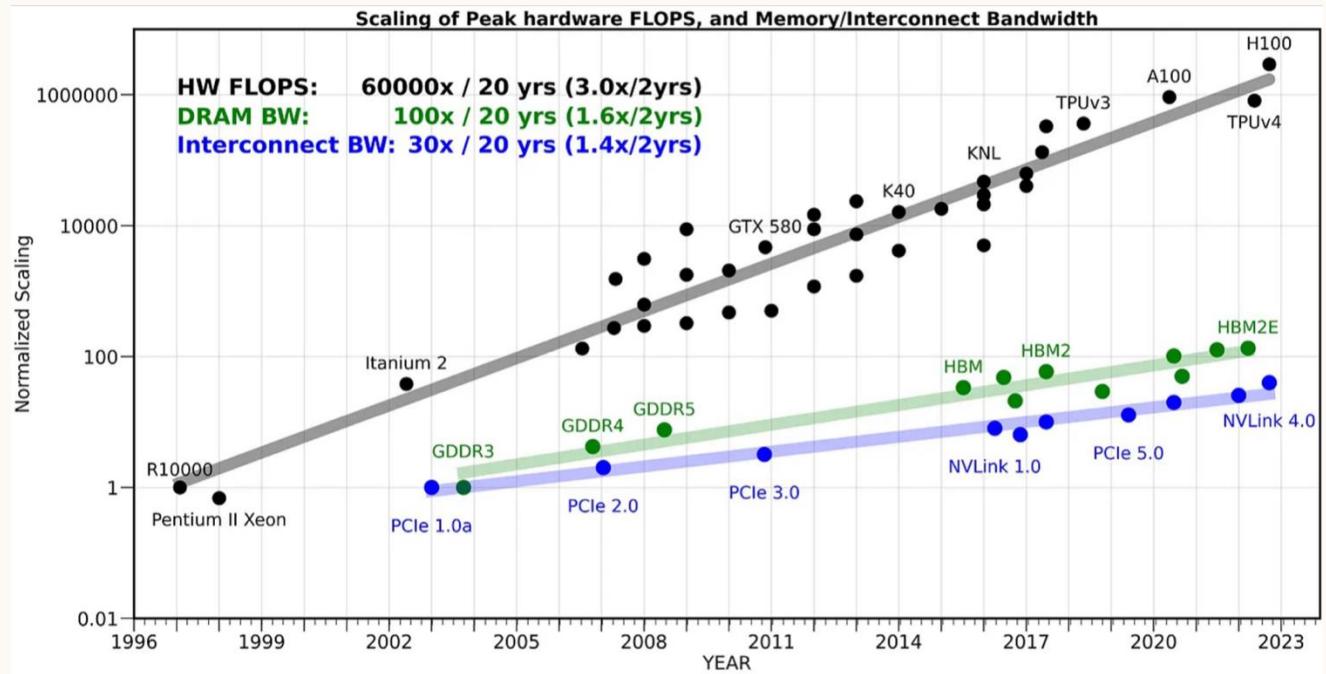
Systems 101: GPUs

- Massively parallel
- Fast matrix multiplication: special cores >10x faster than other fp ops



Systems 101: GPUs

- Massively parallel
- Fast matrix multiplication
- Compute > memory & communication:
 - Hard to keep processors fed with data



BERT transformer
Table 1. Proportions for operator classes in PyTorch.

	Operator class	% flop	% Runtime
Matmul	△ Tensor contraction	99.80	61.0
	□ Stat. normalization	0.17	25.5
Activation	○ Element-wise	0.03	13.5

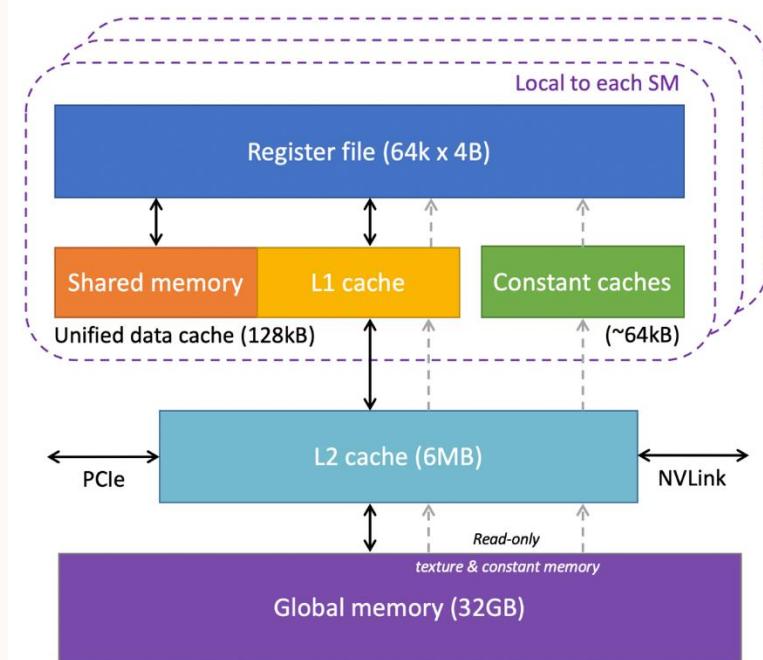
DataMovement
[Ivanov+ 2020]

Systems 101: GPUs

- Massively parallel
- Fast matrix multiplication
- Compute > memory & communication
- Memory hierarchy:
 - Closer to cores => faster but less memory
 - Further from cores => more memory but slower

TABLE IV
THE MEMORY ACCESSES LATENCIES

Memory type	CPI (cycles)
Global memory	290
L2 cache	200
L1 cache	33
Shared Memory (ld/st)	(23/19)



Systems 101: GPUs

- Massively parallel
- Fast matrix multiplication
- Compute > memory & communication
- Memory hierarchy
- Metric: **Model Flop Utilization (MFU)**
 - Ratio: observed throughput / theoretical best for that GPU
 - 50% is great!

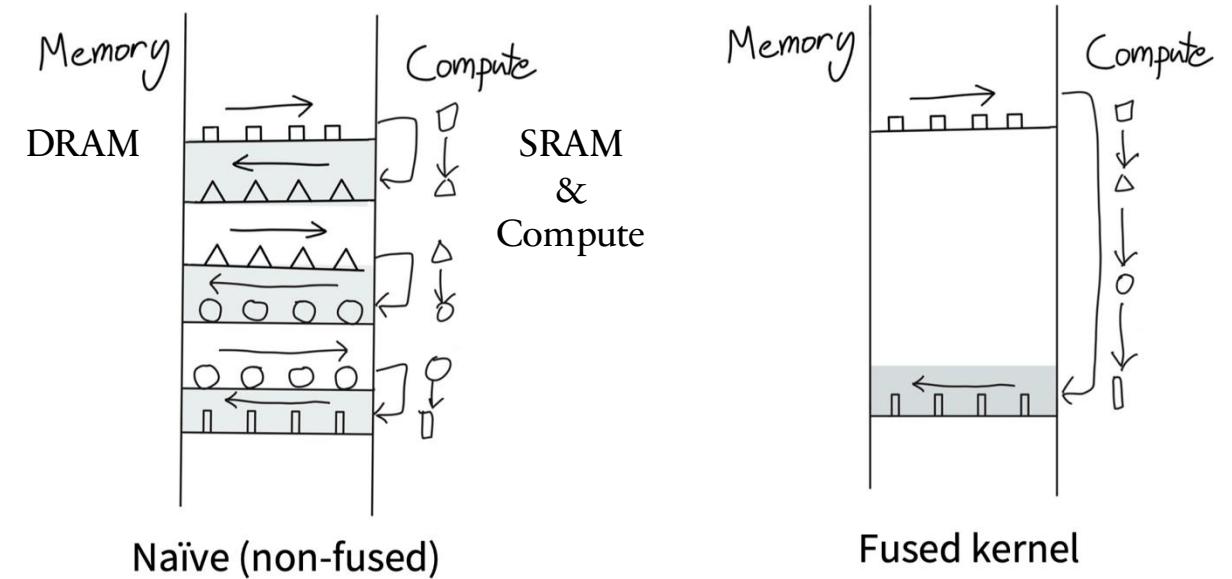
Systems: low precision

- Fewer bits => faster communication & lower memory consumption
- For deep learning: decimal precision ~doesn't matter except exp & updates
 - Matrix multiplications can use bf16 instead of fp32
- For training: **Automatic Mixed Precision (AMP)**
 - Weights stored in fp32, but before computation convert to bf16
 - Activation in bf16 => main memory gains
 - (Only) matrix multiplication in bf16 => speed gains
 - Gradients in bf16 => memory gains
 - Master weights updated fp32 => full precision

Systems: operator fusion

- Problem:
 - communication is slow
 - every new PyTorch line moves variables to global memory
- Idea: communicate once
- `torch.compile`

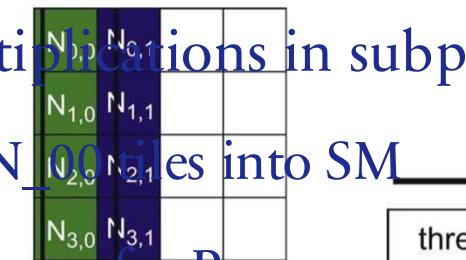
```
x1 = x.cos() # Read from x in global memory, write to x1
x2 = x1.cos() # Read from x1 in global memory, write to x2
```



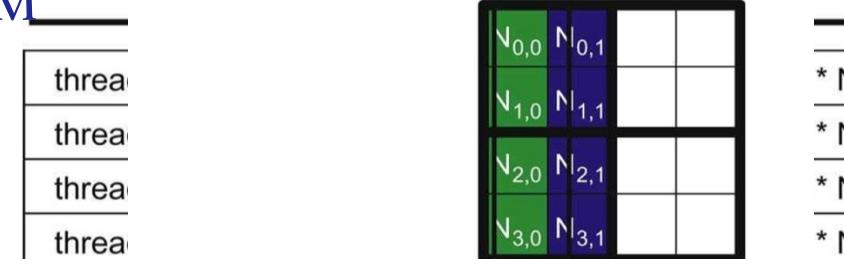
Systems: tiling

- Idea: group and order threads to minimize global memory access (slow)
- Eg matrix multiplication
- Compute matrix multiplications in subphases to reuse memory

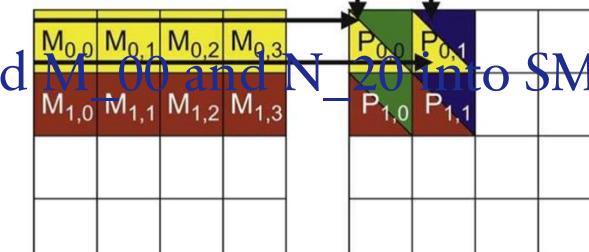
1. Load $M_{0,0}$ and $N_{0,0}$ tiles into SM



2. Compute partial sums for P



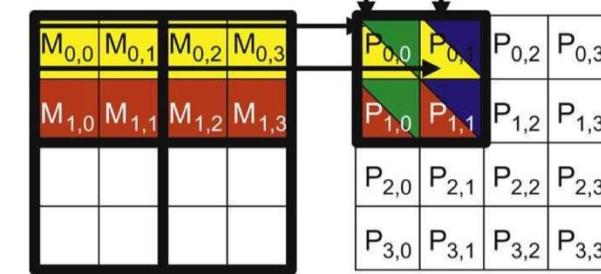
3. Load $M_{0,0}$ and $N_{2,0}$ into SM



4. ...

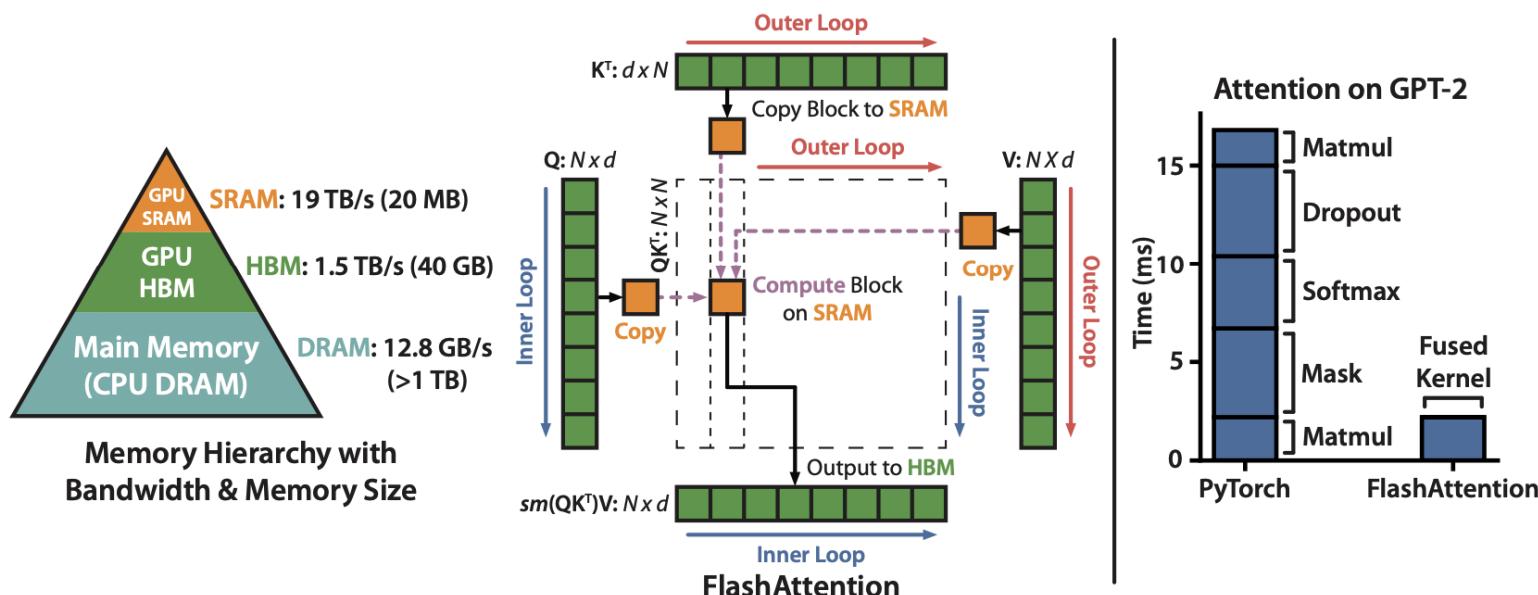
=> reuse reads (~cache)

E.g. assume that thread can
T reduction of global reads have to reread a



Systems: eg FlashAttention

- Idea: kernel fusion, tiling, recomputation for attention!
- 1.7x end to end speed up!



FlashAttention
[Dao+ 2022]

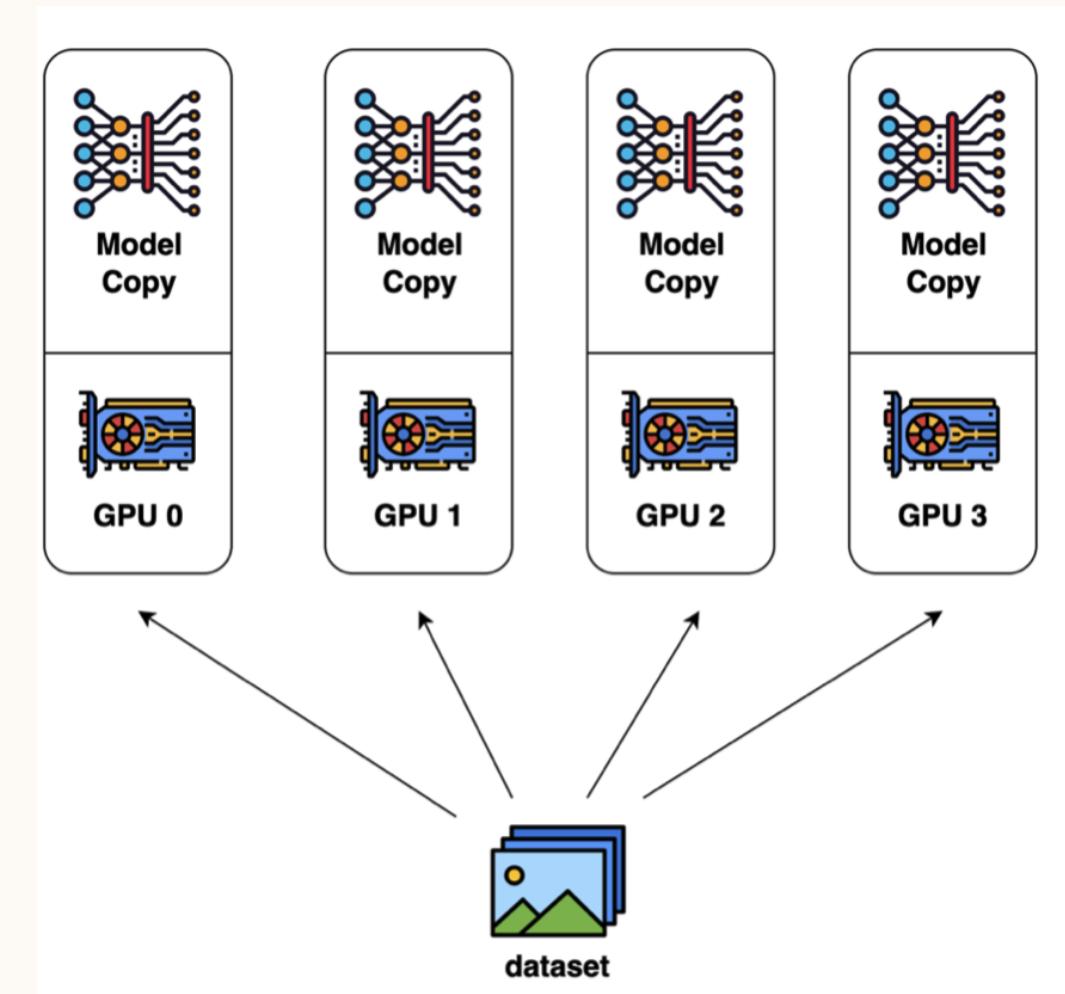
Model implementations	OpenWebText (ppl)	Training time (speedup)
GPT-2 small - Huggingface [87]	18.2	9.5 days (1.0x)
GPT-2 small - Megatron-LM [77]	18.2	4.7 days (2.0x)
GPT-2 small - FLASHATTENTION	18.2	2.7 days (3.5x)

Systems: parallelization

- Problem:
 - model very big \Rightarrow can't fit on one GPU
 - Want to use as many GPUs as possible
- Idea: split memory and compute across GPUs
- Background: to naively train a P parameter model you need at least $16P$ GB of DRAM
 - $4P$ GB for model weights
 - $2 * 4P$ GB for optimizer
 - $4P$ GP for gradients
- E.g. for 7B model you need 112GB!

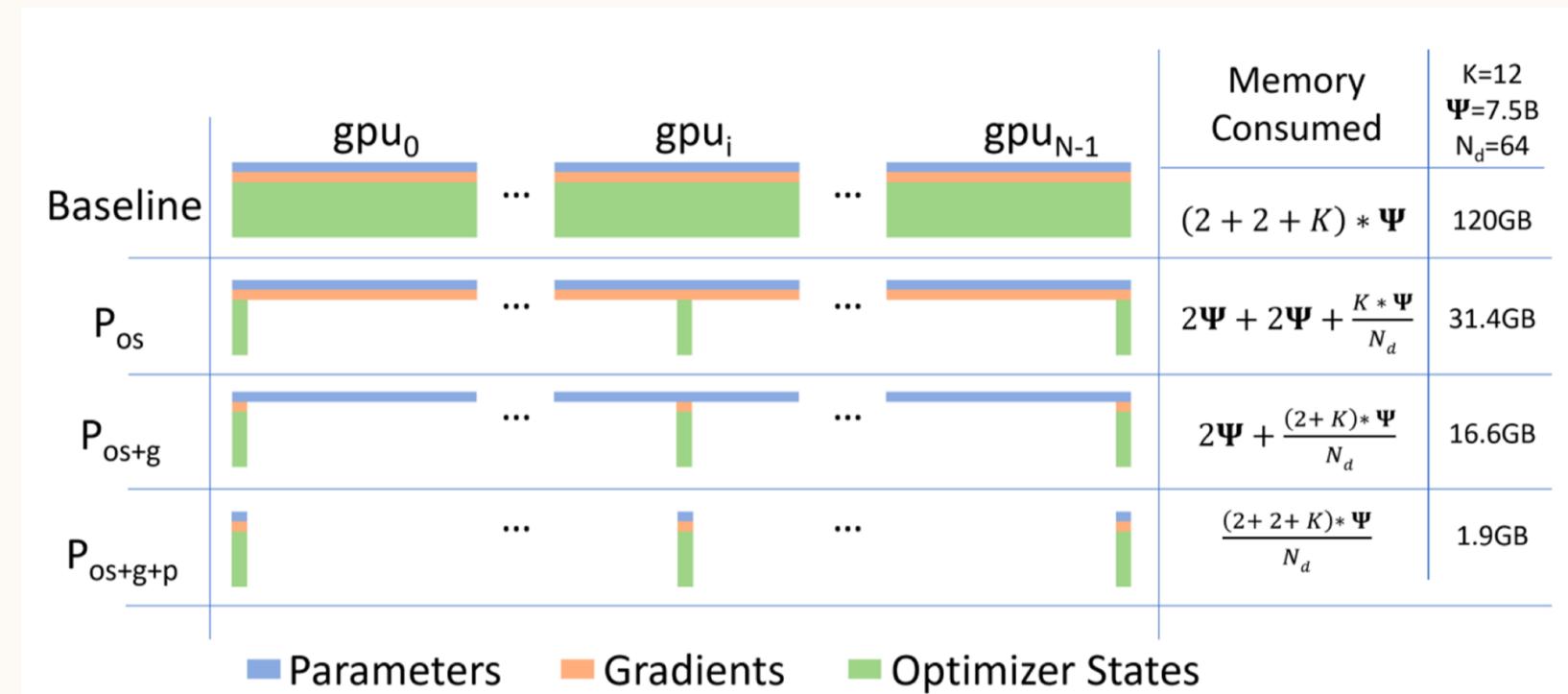
Systems: data parallelism

- Goal: use more GPUs
- Naïve data parallelization:
 1. Copy model & optimizer on each GPU
 2. Split data
 3. Communicate and reduce (sum) gradients
- Pro: use parallel GPU
- Con: no memory gains!



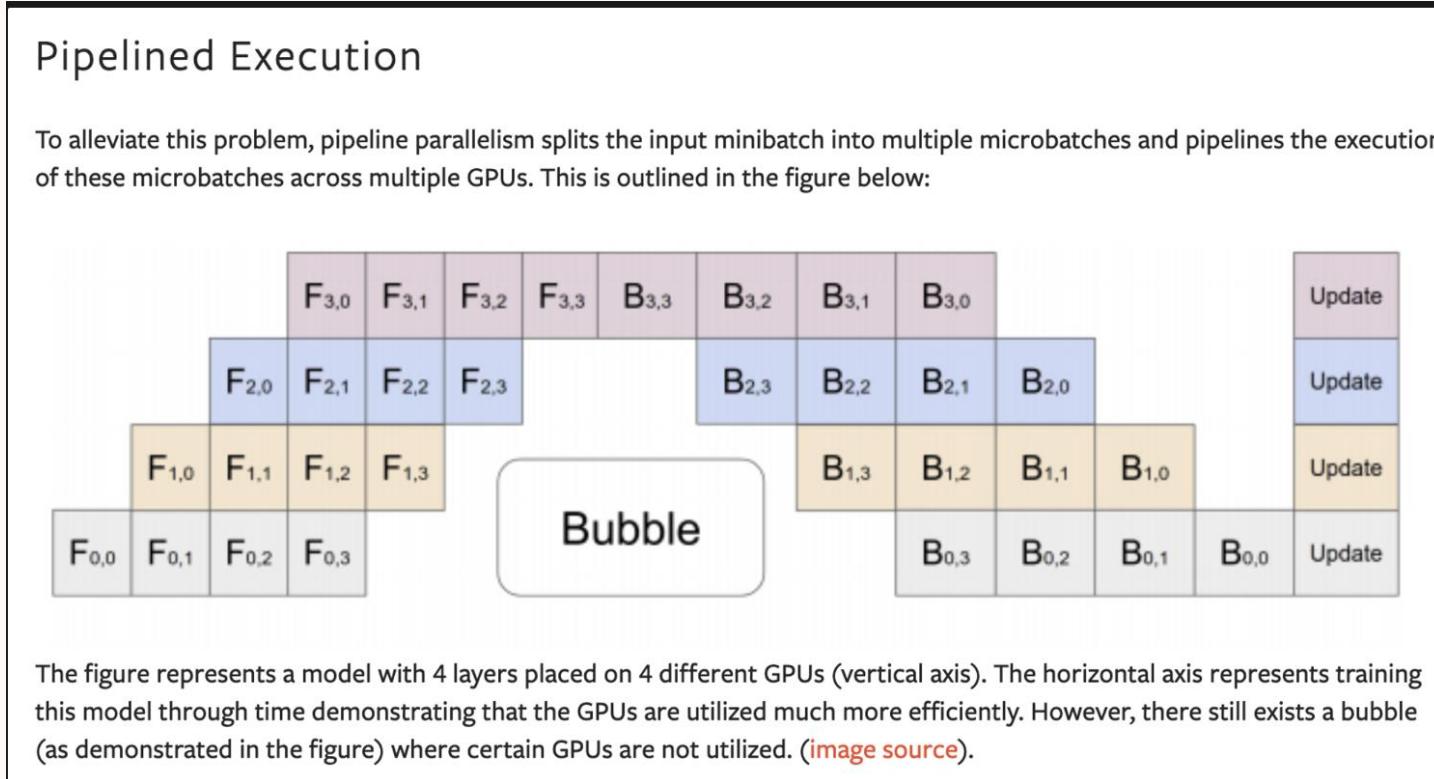
Systems: data parallelism

- Goal: split up memory
- Idea: each GPU updates subset of weights and them before next step => sharding



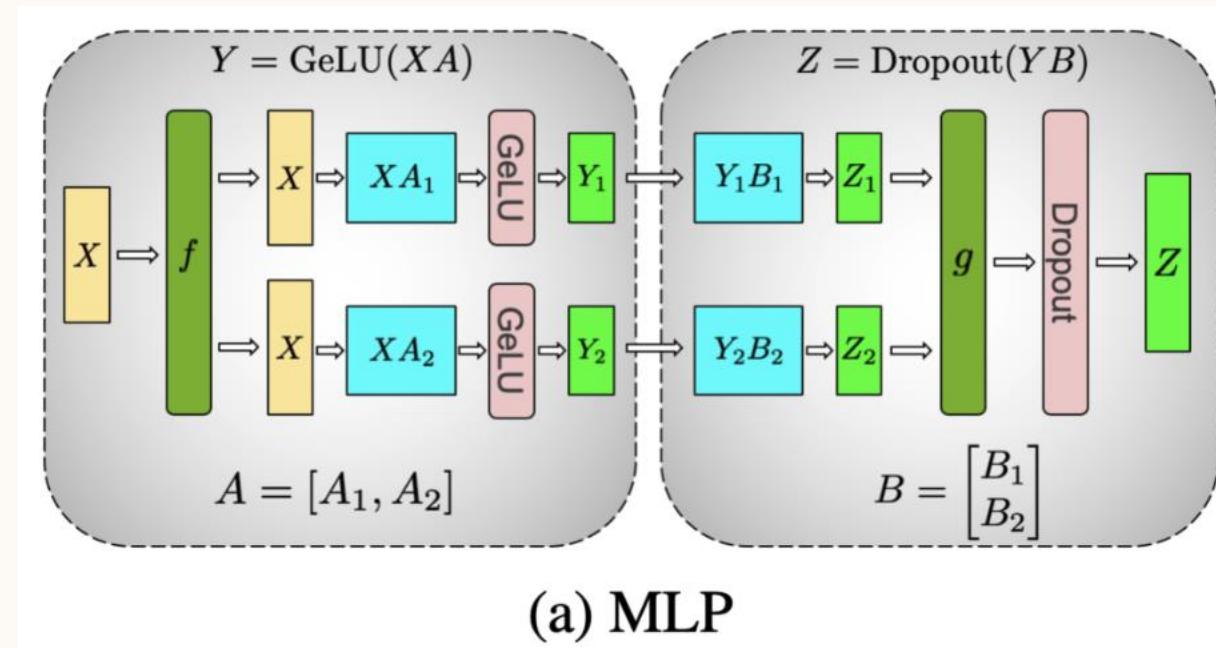
Systems: model parallelism

- Problem: data parallelism only works if batch size $\geq \# \text{GPUS}$
- Idea: have every GPU take care of applying specific parameters (rather than updating)
 - Eg **pipeline parallel**: every GPU has different layer



Systems: model parallelism

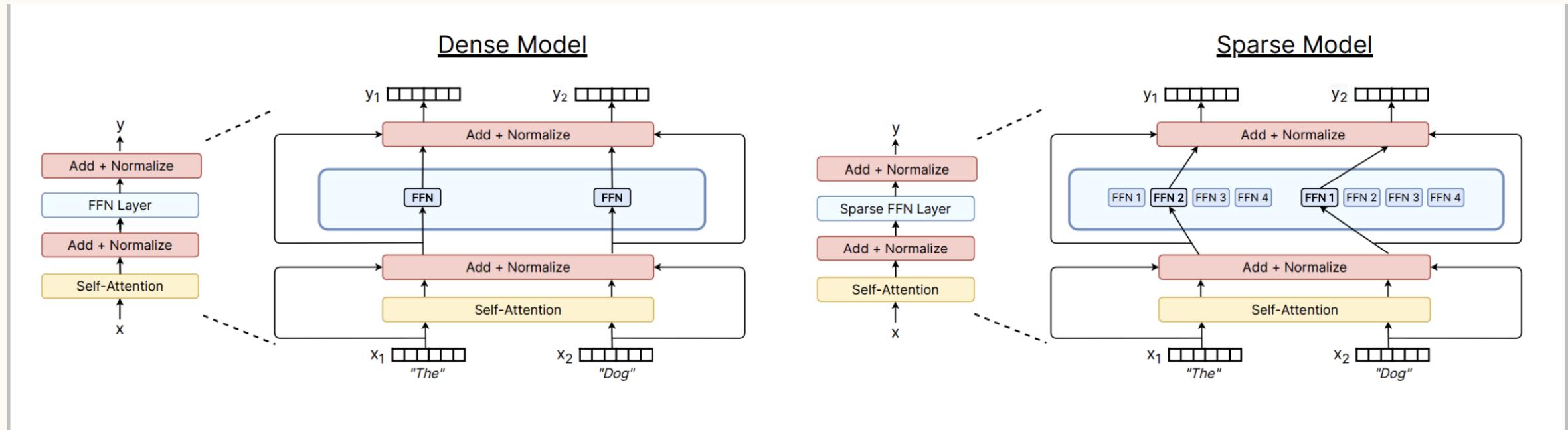
- Problem: data parallelism only works if batch size $\geq \# \text{GPUS}$
- Idea: have every GPU take care of applying specific parameters (rather than updating)
 - Eg **pipeline parallel**: every GPU has different layer
 - Eg **tensor parallel**: split single matrix across GPUs and use partial sum



Megatron-LM:
[Shoeybi+ 2019]

Systems: architecture sparsity

- Idea: models are huge => not every datapoint needs to go through every parameter
 - Eg **Mixture of Experts**: use a selector layer to have less “active” parameter => same FLOPs



Sparse Expert Models:
[Fedus+ 2012]

Wrap-up



Outlook

Haven't touched upon:

- Architecture: MoE & SSM
- Decoding & inference
- UI & tools: ChatGPT
- Multimodality
- Misuse
- Context size
- Data wall
- Legality of data collection

Going further:

- CS224N: more of the background and historical context. Some adjacent material.
- CS324: more in-depth reading and lectures.
- CS336: you actually build your LLM. Heavy workload!

Questions?

