

Lecture 8: Language Model

Berkeley CS294-158 Deep Unsupervised Learning
Spring 2024

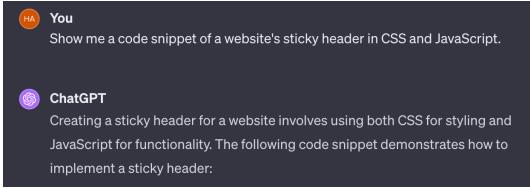
Hao Liu

Outline

- Methodologies behind large language models
 - Basics
 - Bottlenecks and solutions
 - Scaling laws
 - Capabilities
- How to train your large language models
 - Sharding

Successes of machine and deep learning

- Language model is everywhere
- Secret sauce: enormous compute power



Chatbot

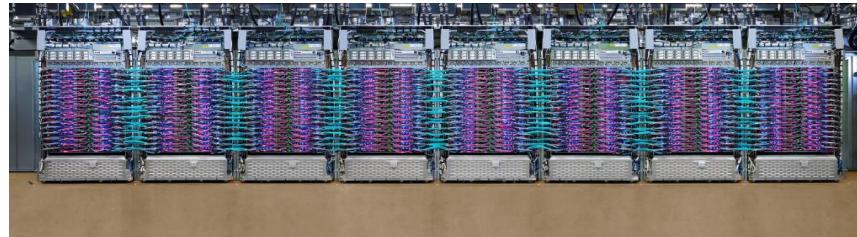
A screenshot of the GitHub Copilot interface. It shows a code editor with Python code for parsing expenses from a CSV file. The code includes imports for `csv` and `dateutil`, defines a class `ExpenseParser`, and contains several methods for reading and parsing expense data. A tooltip at the bottom explains that the code handles various edge cases like empty input and invalid date formats.

```
def parse_expenses(expenses_string):
    """Parse the list of expenses and return the list of tuples (date, amount, expense)"""
    # Parse the date using dateutil
    # ...
    # ...
    # ...
    expenses = []
    for line in expenses_string.splitlines():
        # ...
        # ...
        date, amount, expense = line.split(',')
        date = datetime.datetime.strptime(date, "%Y-%m-%d")
        expense = float(expense)
        expenses.append((date, amount, expense))
    return expenses
```

Copilot



Sora



TPU datacenter

Language model

- A language model is a probability distribution over sequences:
 - Likelihood distribution $p_\theta(y)$
 - Sequence of tokens $y = (y_1, \dots, y_T)$
 - Parameters: θ

Language model

- Typical language models are autoregressive:

$$p_{\theta}(y) = \prod_{t=1}^T p_{\theta}(y_t \mid y_{<t})$$

- Model is parameterized by a transformer

Autoregressive

- Autoregressive factorization
 - Next token prediction
 - Predicting future
- Every bit becomes supervision
- Natural for conversational AI

Modeling likelihood

- Maximum likelihood: make observed data likely under the model

$$\arg \max_{\theta} \frac{1}{|\mathcal{D}|} \sum_{y \in \mathcal{D}} \log p_{\theta}(y)$$

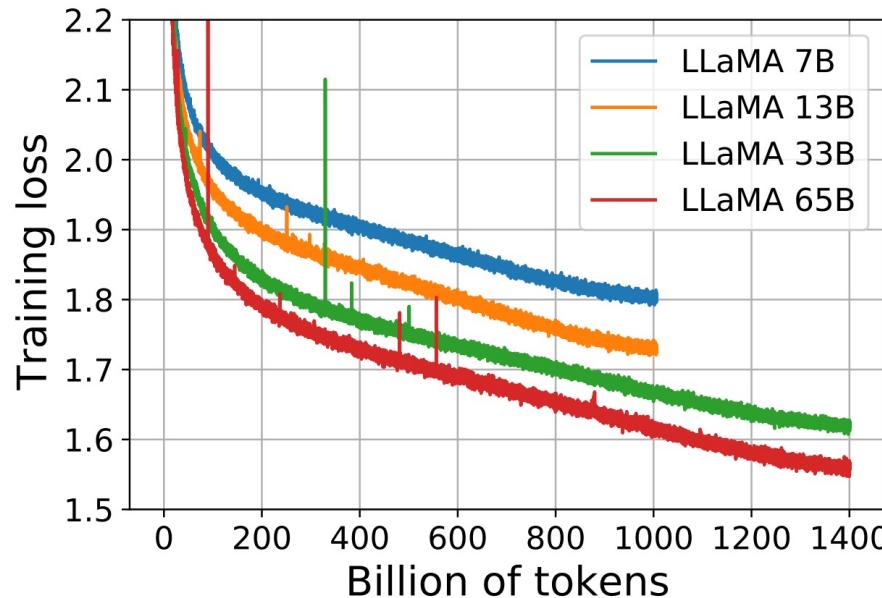
- D : 1.5 trillions of tokens in Llama
- θ : 7 to 70 billions of parameters

Learning

- **Pretraining:** large-scale unsupervised learning
- **Finetuning:** specializing the model for specific tasks
- **Learning from feedback:** improving the model with feedback on its outputs, such as human feedback and debug message

Why unsupervised learning?

- Much more data available
- Much better generalization



Scaling compute

- We want to model data distribution better by adding more compute.
- *“The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin.”* The Bitter Lesson, Richard Sutton 2019

Compute

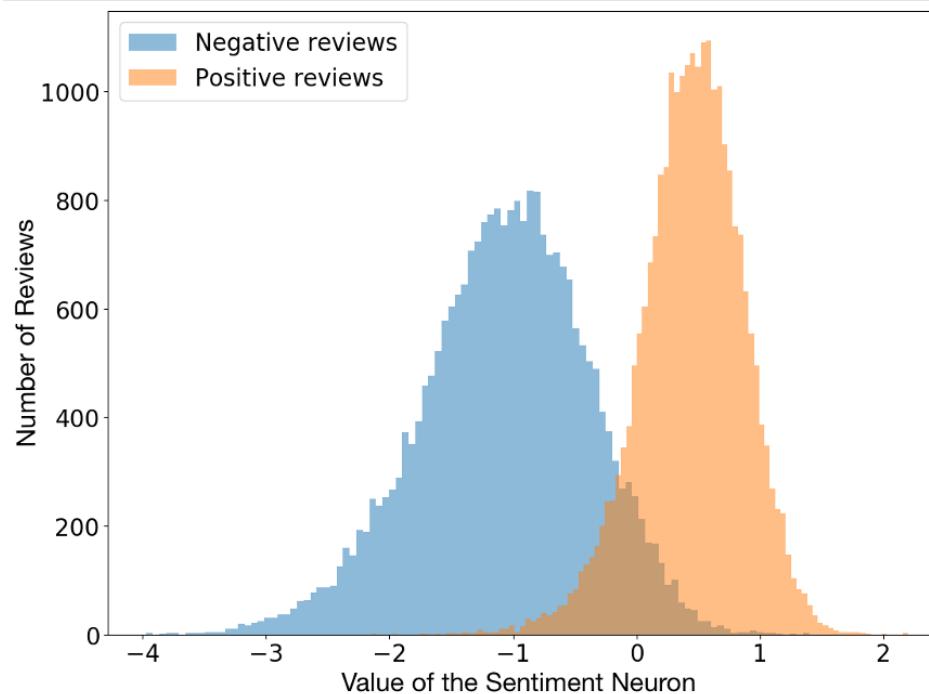
- Compute is forward and backward passes using our model on token sequences.
- Mostly many matrix multiplications (matmul)
 - Unit: floating point operations (FLOPs)
- Compute cost: $C = 6ND(1+s/6d)$
 - N: parameters, D: data size, s: context size, d: hidden dimension
- Adding compute by using more tokens, larger context, larger model

Token

- Byte-based: most general but too long
- Character-based: each word requires too many tokens
- Word-based: `dog' and `dogs' should share meaning
- Subword-based:
 - Byte-pair encoding: replace top appearing pair with a new token
 - Repeat until reach a given vocab size

Train LSTM with more compute

- **LSTM** learns a sentiment neuron after training to predict the next word on a large amount of Amazon reviews.



Train LSTM with more compute

25 August 2003 League of Extraordinary Gentlemen: Sean Connery is one of the all time greats and I have been a fan of his since the 1950's. I went to this movie because Sean Connery was the main actor. I had not read reviews or had any prior knowledge of the movie. The movie surprised me quite a bit. The scenery and sights were spectacular, but the plot was unreal to the point of being ridiculous. In my mind this was not one of his better movies it could be the worst. Why he chose to be in this movie is a mystery. For me, going to this movie was a waste of my time. I will continue to go to his movies and add his movies to my video collection. But I can't see wasting money to put this movie in my collection.

I found this to be a charming adaptation, very lively and full of fun. With the exception of a couple of major errors, the cast is wonderful. I have to echo some of the earlier comments -- Chynna Phillips is horribly miscast as a teenager. At 27, she's just too old (and, yes, it DOES show), and lacks the singing "chops" for Broadway-style music. Vanessa Williams is a decent-enough singer and, for a non-dancer, she's adequate. However, she is NOT Latina, and her character definitely is. She's also very STRIDENT throughout, which gets tiresome. The girls of Sweet Apple's Conrad Birdie fan club really sparkle -- with special kudos to Brigitte Dau and Chiara Zanni. I also enjoyed Tyne Daly's performance, though I'm not generally a fan of her work. Finally, the dancing Shriners are a riot, especially the dorky three in the bar. The movie is suitable for the whole family, and I highly recommend it.

Judy Holliday struck gold in 1950 with George Cukor's film version of "Born Yesterday," and from that point forward, her career consisted of trying to find material good enough to allow her to strike gold again. It never happened. In "It Should Happen to You" (I can't think of a blander title, by the way), Holliday does yet one more variation on the dumb blonde who's maybe not so dumb after all, but everything about this movie feels warmed over and half hearted. Even Jack Lemmon, in what I believe was his first film role, can't muster up enough energy to enliven this recycled comedy. The audience knows how the movie will end virtually from the beginning, so mostly it just sits around waiting for the film to catch up. Maybe if you're enamored of Holliday you'll enjoy this; otherwise I wouldn't bother. Grade: C

Once in a while you get amazed over how BAD a film can be, and how in the world anybody could raise money to make this kind of crap. There is absolutely NO talent included in this film - from a crappy script, to a crappy story to crappy acting. Amazing...

Team Spirit is maybe made by the best intentions, but it misses the warmth of "All Stars" (1997) by Jean van de Velde. Most scenes are identic, just not that funny and not that well done. The actors repeat the same lines as in "All Stars" but without much feeling.

God bless Randy Quaid...his leachorous Cousin Eddie in Vacation and Christmas Vacation hilariously stole the show. He even made the awful Vegas Vacation at least worth a look. I will say that he tries hard in this made for TV sequel, but that the script is so NON funny that the movie never really gets anywhere. Quaid and the rest of the returning Vacation vets (including the original Audrey, Dana Barron) are wasted here. Even European Vacation's Eric Idle cannot save the show in a brief cameo... Pathetic and sad...actually painful to watch...Christmas Vacation 2 is the worst of the Vacation franchise.

■ Visualizing the value of the sentiment cell as it processes six randomly selected high contrast IMDB reviews. Red indicates negative sentiment while green indicates positive sentiment. Best seen in color.

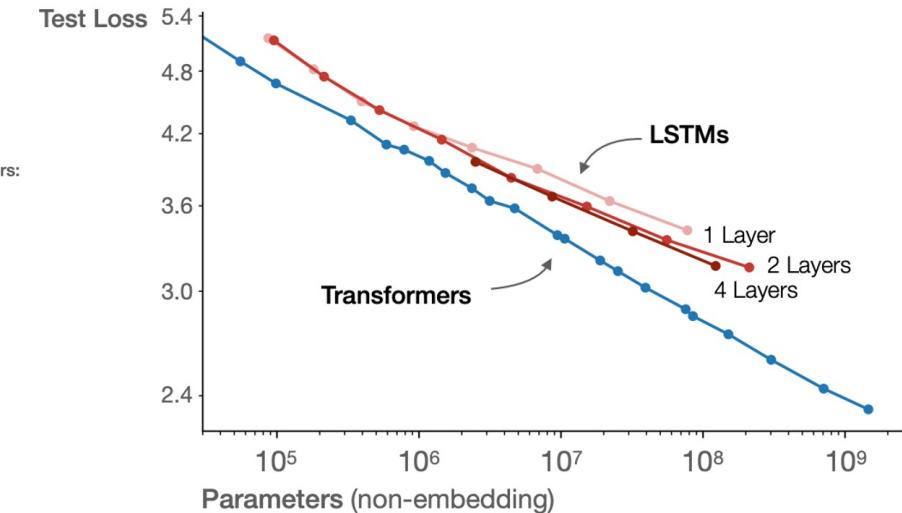
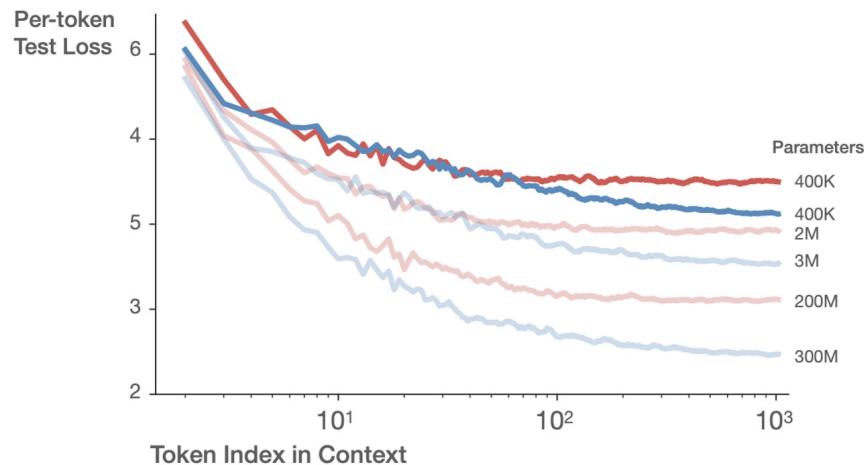
Train LSTM with more compute

- **LSTM** learns a sentiment neuron after training to predict the next word on a large amount of Amazon reviews.

Sentiment fixed to positive	Sentiment fixed to negative
Just what I was looking for. Nice fitted pants, exactly matched seam to color contrast with other pants I own. Highly recommended and also very happy!	The package received was blank and has no barcode. A waste of time and money.
This product does what it is supposed to. I always keep three of these in my kitchen just in case ever I need a replacement cord.	Great little item. Hard to put on the crib without some kind of embellishment. My guess is just like the screw kind of attachment I had.
Best hammock ever! Stays in place and holds it's shape. Comfy (I love the deep neon pictures on it), and looks so cute.	They didn't fit either. Straight high sticks at the end. On par with other buds I have. Lesson learned to avoid.
Dixie is getting her Doolittle newsletter we'll see another new one coming out next year. Great stuff. And, here's the contents - information that we hardly know about or forget.	great product but no seller. couldn't ascertain a cause. Broken product. I am a prolific consumer of this company all the time.
I love this weapons look . Like I said beautiful !!! I recommend it to all. Would suggest this to many roleplayers , And I stronge to get them for every one I know. A must watch for any man who love Chess!	Like the cover, Fits good. . However, an annoying rear piece like garbage should be out of this one. I bought this hoping it would help with a huge pull down my back & the black just doesn't stay. Scrap off everytime I use it.... Very disappointed.

Scaling LSTM is difficult

- **Attention** allows scalably improving models. Transformers asymptotically outperform LSTMs due to improved use of context.



“Better usage of context”

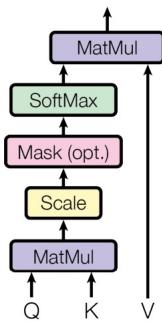


“Better model performance”

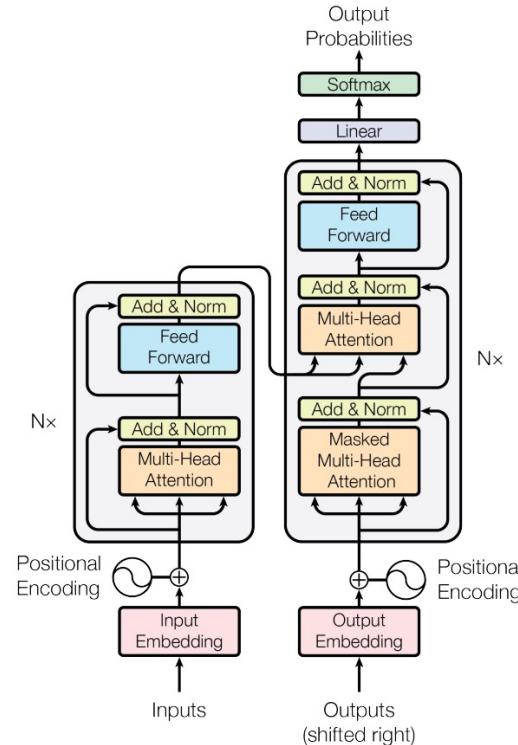
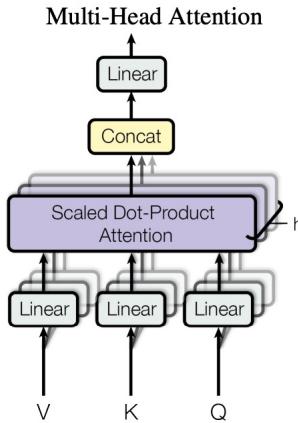
Transformer architecture

- Attention allows attending to past tokens without forgetting.
- Big FFNs, highly scalable

Scaled Dot-Product Attention



Multi-Head Attention



Pretraining objective

- Full autoregressive: GPT, Llama
- Prefix autoregressive: T5
- Masked: BERT, ELMO

Full Language Modeling

May the force be with you

targets

Prefix Language Modeling

May the force be with you

targets

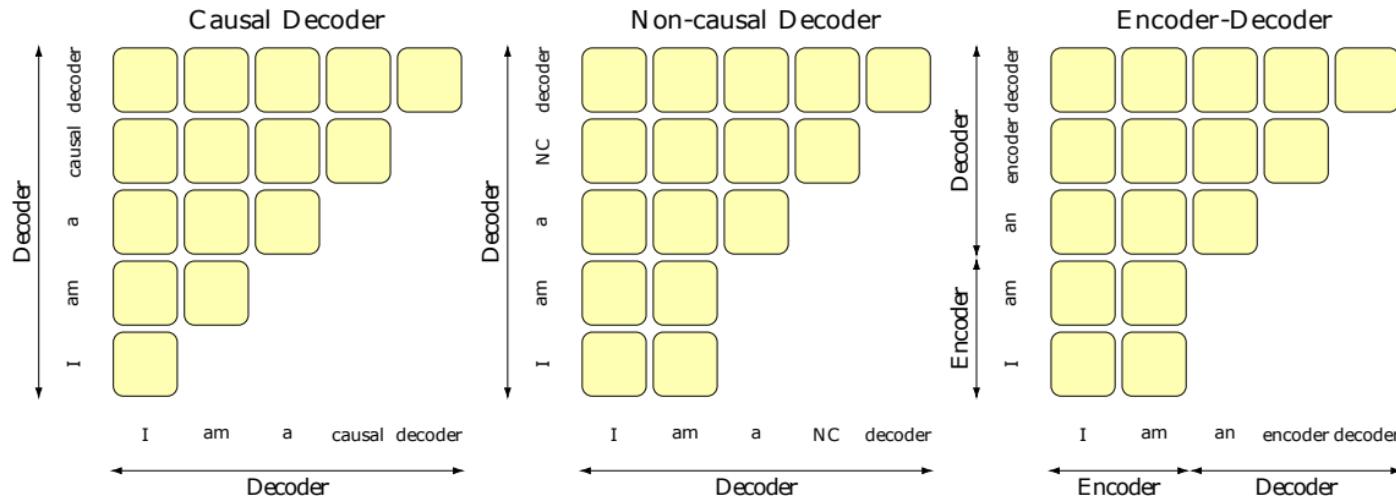
Masked Language Modeling

May the force be with you

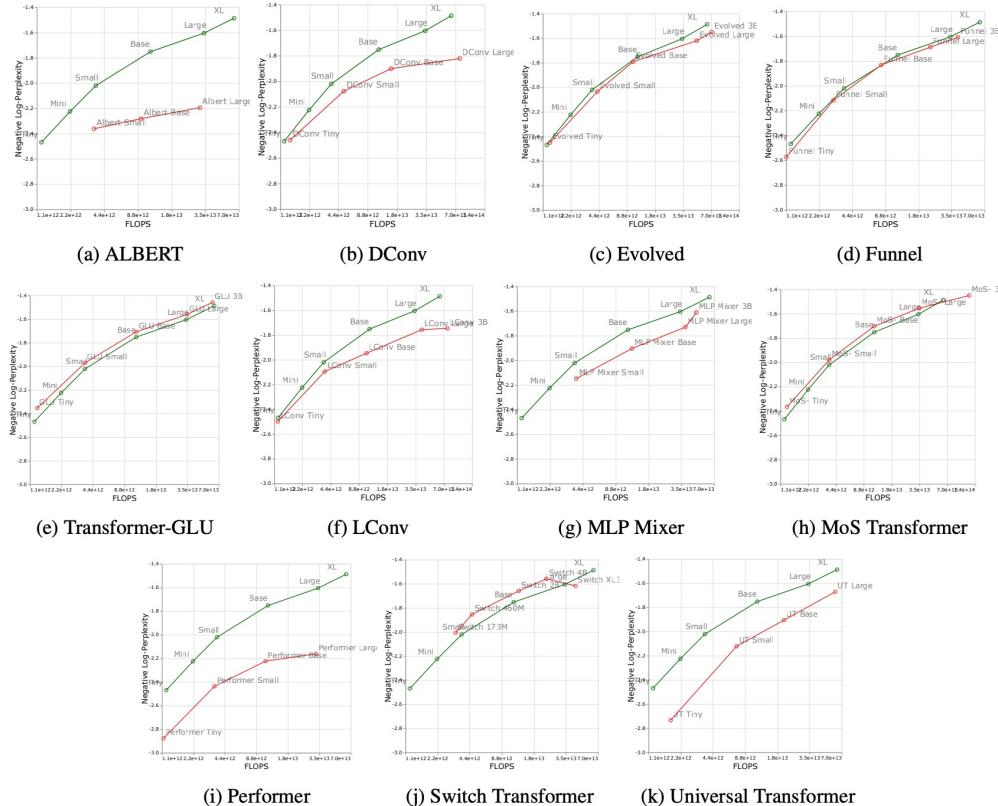
targets

Pretraining objective

- T5, BERT, GPT. All with different masking
- Causal decoder (GPT) used most frequently



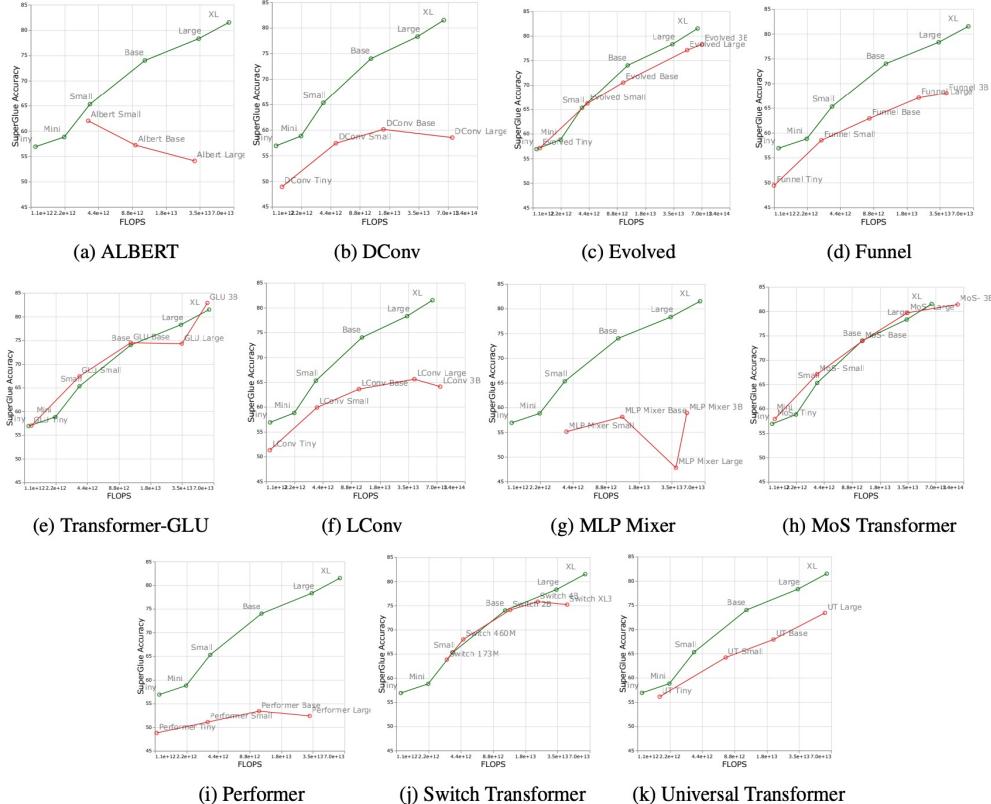
Model architecture



- Upstream Negative Log-Perplexity of vanilla Transformer compared to other models.

- Transformer outperforms other models.

Model architecture



- Downstream accuracy of vanilla Transformer compared to other models.
- Transformer outperforms other models.

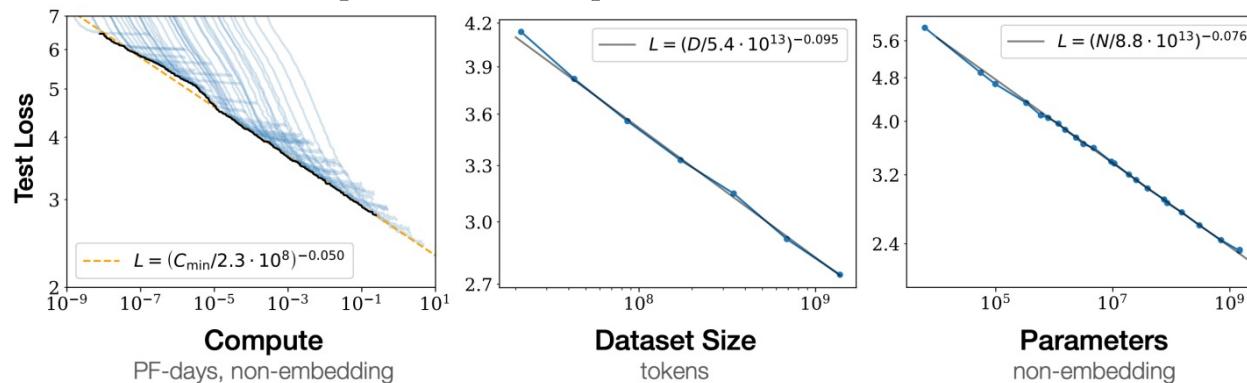
Compute cost

- Parameter count is N , token count is D , what is the compute cost?
 - $C = 6ND(1+s/6d)$: Compute increases with more data and larger model
 - LLaMA ($s \ll 6d$): $C = 6ND = 6 * 7 \text{ billion} * 2 \text{ trillion} = 8.4 \times 10^{22} \text{ FLOPs}$
- Shall I allocate more compute to data or model?

$$N_{opt}(C), D_{opt}(C) = \operatorname{argmin}_{N,D \text{ s.t. } \text{FLOPs}(N,D)=C} L(N, D)$$

Optimal token and parameter

- Empirical performance has a power-law relationship with each individual factor.
 - Train model of different sizes and number of tokens (light blue lines)
 - Pick minimal loss (black line)
 - Run linear regression on log – log
- Follow power-law: $N_{opt} \propto C^a$, $D_{opt} \propto C^b$



Allocate compute

- Empirical performance: $N_{opt} \propto C^a, D_{opt} \propto C^b$
- We have $a + b = 1$ because $C = 6ND$

Allocate more compute to parameters (a) or tokens (b)?

Allocate compute

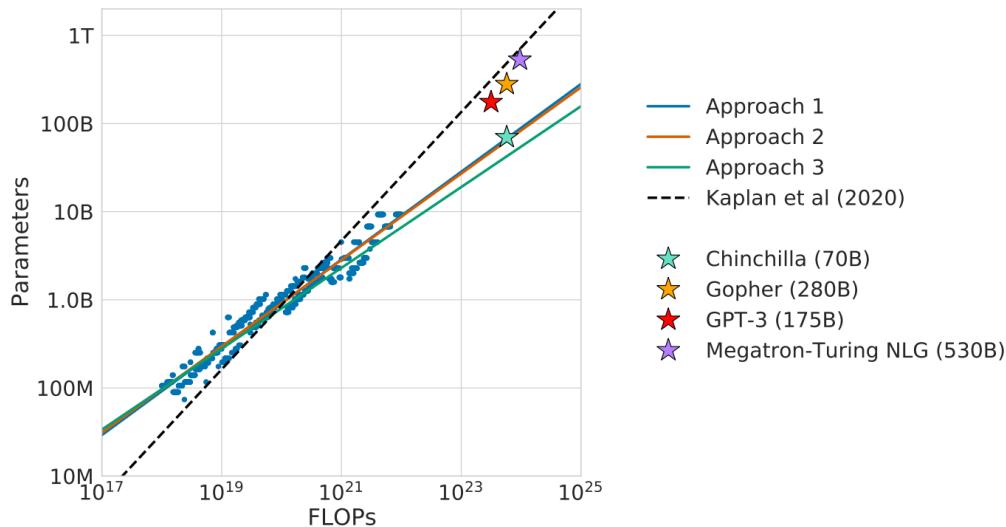
- Coefficients of model scaling and data scaling vary with training data distribution

Approach	Coeff. a where $N_{\text{opt}}(M_{\text{opt}}) \propto C^a$	Coeff. b where $D_{\text{opt}} \propto C^b$
OpenAI (OpenWebText2)	0.73	0.27
Chinchilla (MassiveText)	0.49	0.51
Ours (Early Data)	0.450	0.550
Ours (Current Data)	0.524	0.476
Ours (OpenWebText2)	0.578	0.422

- OpenAI (2020) gives more compute to parameters
- DeepMind (2022) gives more compute to tokens
- Best practice: train different small models and data sizes, and fit the constants yourself.

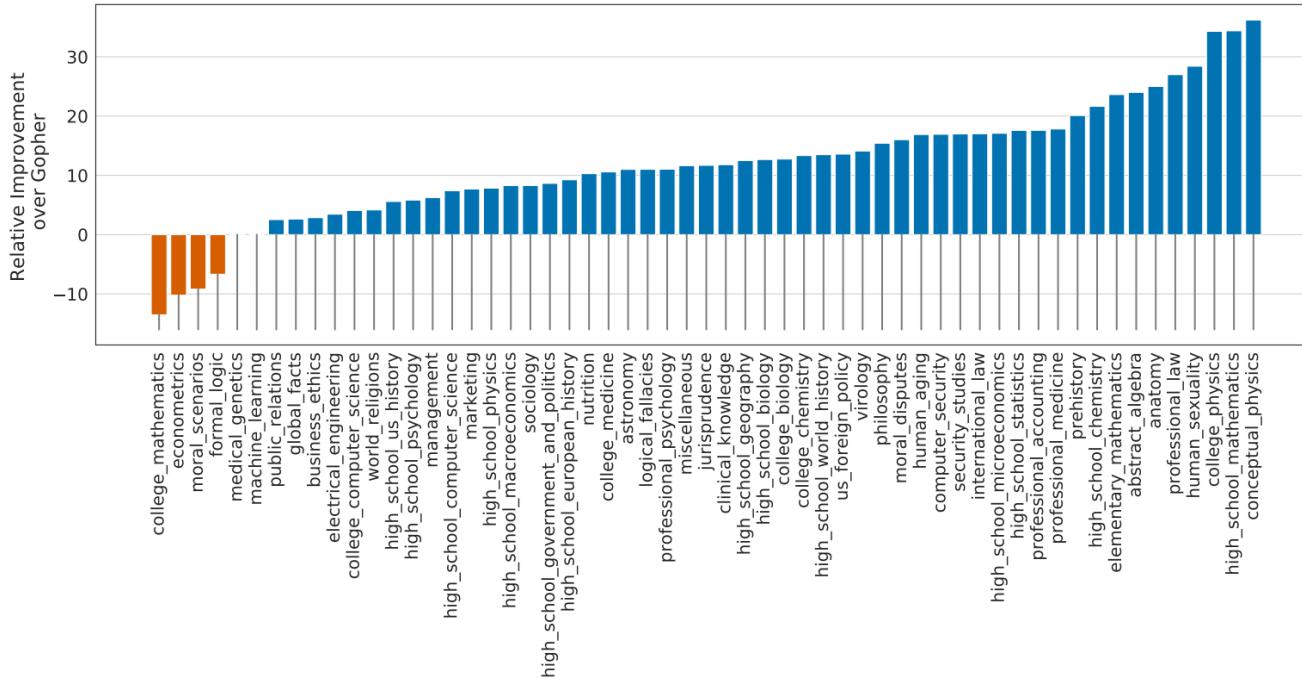
Chinchilla scaling

- Chinchilla scaling law: $a = 0.49$, $b=0.51$
- Implication: more compute efficient than other models



Chinchilla scaling

- Chinchilla outperforms Gropher significantly by allocating compute better



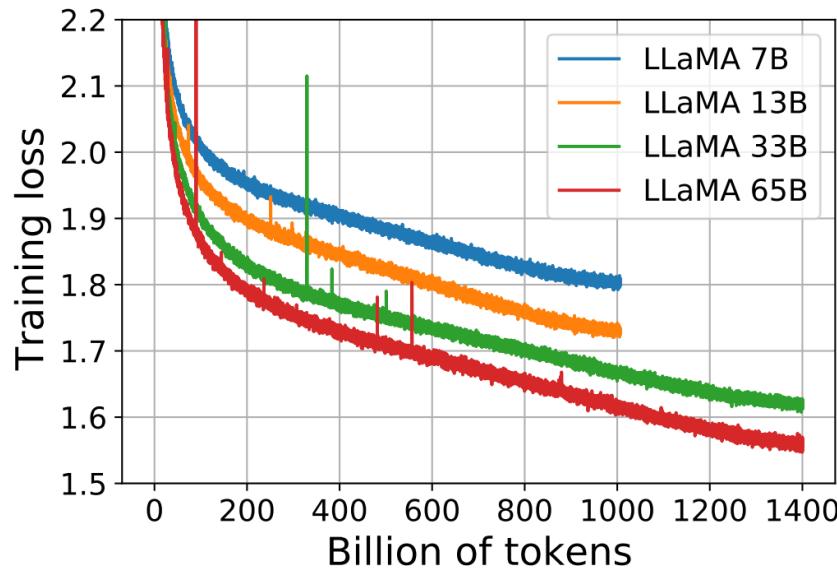
Determine tokens

- Estimated optimal training tokens: about 20 times number of parameters
- It's best to double the estimate: tokens = 40x parameters

Parameters	FLOPs	FLOPs (in <i>Gopher</i> unit)	Tokens
400 Million	1.92e+19	1/29,968	8.0 Billion
1 Billion	1.21e+20	1/4,761	20.2 Billion
10 Billion	1.23e+22	1/46	205.1 Billion
67 Billion	5.76e+23	1	1.5 Trillion
175 Billion	3.85e+24	6.7	3.7 Trillion
280 Billion	9.90e+24	17.2	5.9 Trillion
520 Billion	3.43e+25	59.5	11.0 Trillion
1 Trillion	1.27e+26	221.3	21.2 Trillion
10 Trillion	1.30e+28	22515.9	216.2 Trillion

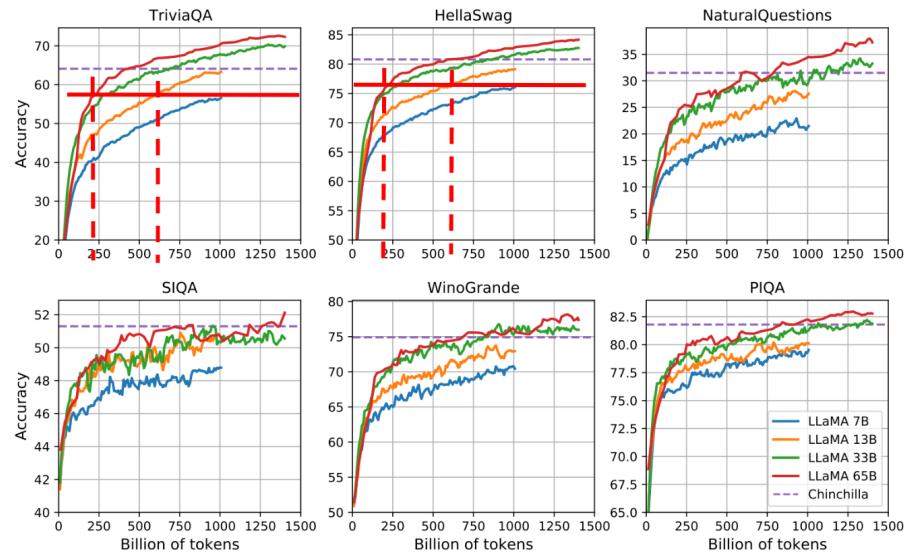
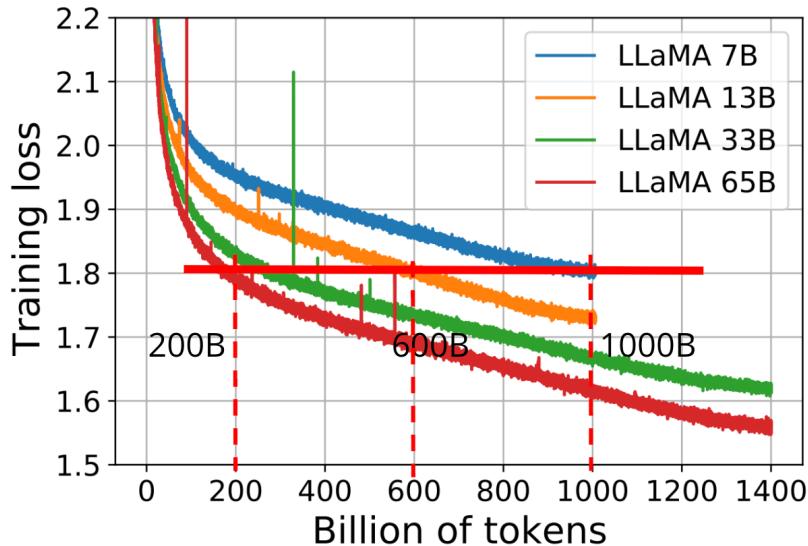
Inference optimal

- Train small model and more tokens to achieve better inference
 - E.g. Llama 7b is trained on 7 times of Chinchilla optimal



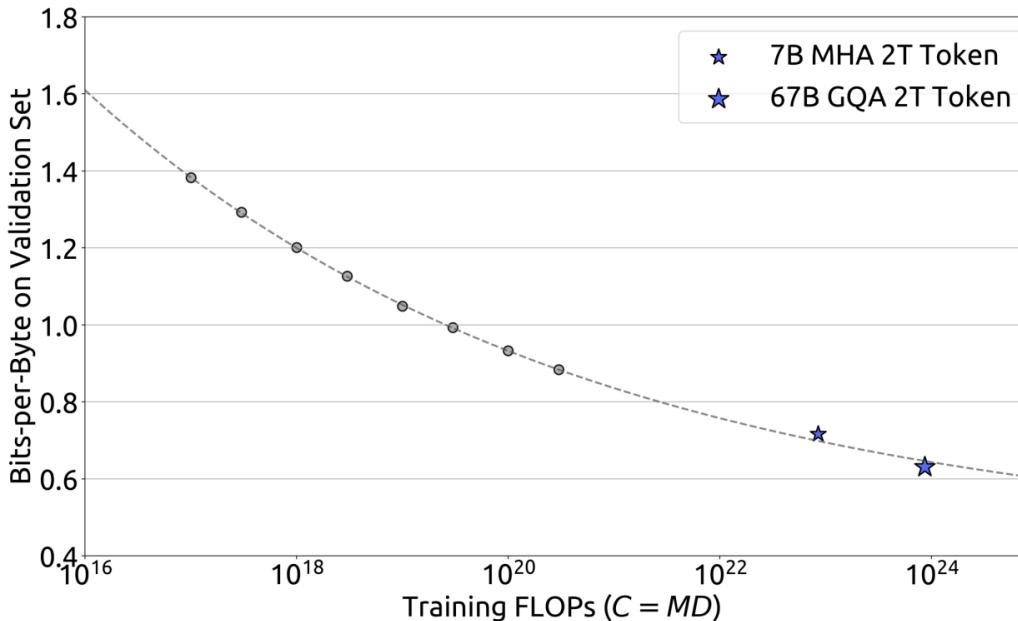
Loss predicts performance

- Loss determines downstream performance. Both large and small models have the same performance if they have the same loss.



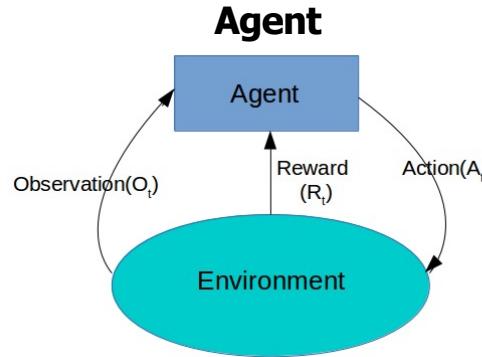
Scaling law prediction

- Scaling law can predict performance at larger scale
 - Predicting performance of larger models and larger datasets



Large context

- Major bottleneck: cannot even fit complex, long sequences into Transformers. Needed for agent, world modeling, codebase, and hyperlinked web.



Codebase

monalisa
Write a set of unit test functions for the selected code

Github Copilot

```
class TestParseExpenses(unittest.TestCase):  
  
    def test_parse_expenses_with_valid_input(self)  
        expenses_string = '2023-01-02 -34.01 USD'  
        expected_output = [(datetime.datetime(2023,  
                                             1, 2), -34.01, 'USD')]  
        self.assertEqual(parse_expenses(expenses_...  
  
    def test_parse_expenses_with_empty_input(self)  
        expenses_string = ''  
        expected_output = []  
        self.assertEqual(parse_expenses(expenses_...  
  
if __name__ == '__main__':  
    unittest.main()
```

The code assumes that the `datetime` module is imported. The test cases cover different scenarios such as valid input, empty input, input with comments, invalid date format, and invalid value format.

Analyze and debug

Ask a question or type '?' for topics ➤

Hyperlinked web

1600 Pennsylvania Avenue restricts here. For other uses, see 1600 Pennsylvania Avenue (disambiguation) and White House (disambiguation).

The White House is the official residence and workplace of the president of the United States. It is located at 1600 Pennsylvania Avenue NW in Washington, D.C., and has been the residence of every U.S. president since John Adams in 1800 when the national capital was moved from Philadelphia.¹⁰ The term "White House" refers to the building itself, and also to the government offices located there.

The residence was designed by Irish-born architect James Hoban in the Neoclassical style.¹¹ Hoban modeled the building on Leinster House in Dublin, a building which today houses the Oireachtas, the Irish legislature. Construction took place between 1792 and 1800, with an exterior of Aquia Creek sandstone painted white. When Thomas Jefferson moved into the house in 1801, he and architect Benjamin Henry Latrobe added low colonnades on each wing to concession what then were stables and stables.¹² In 1814, during the War of 1812, the mansion was set ablaze by British forces in the burning of Washington, destroying the White House and the Library of Congress. Reconstruction almost immediately, and President James Monroe moved into the partially reconstructed Executive Residence in October 1817. Exterior construction continued with the addition of the semicircular South Portico in 1824 and the North Portico in 1829.

Because of crowding within the executive mansion itself, President Theodore Roosevelt had all work offices relocated to the newly constructed West Wing in 1901. Eight years later, in 1909, President William Howard Taft expanded the West Wing and created the first Oval Office, which was eventually moved and expanded. In the Executive Residence, the third floor attic was converted to living quarters in 1927 by

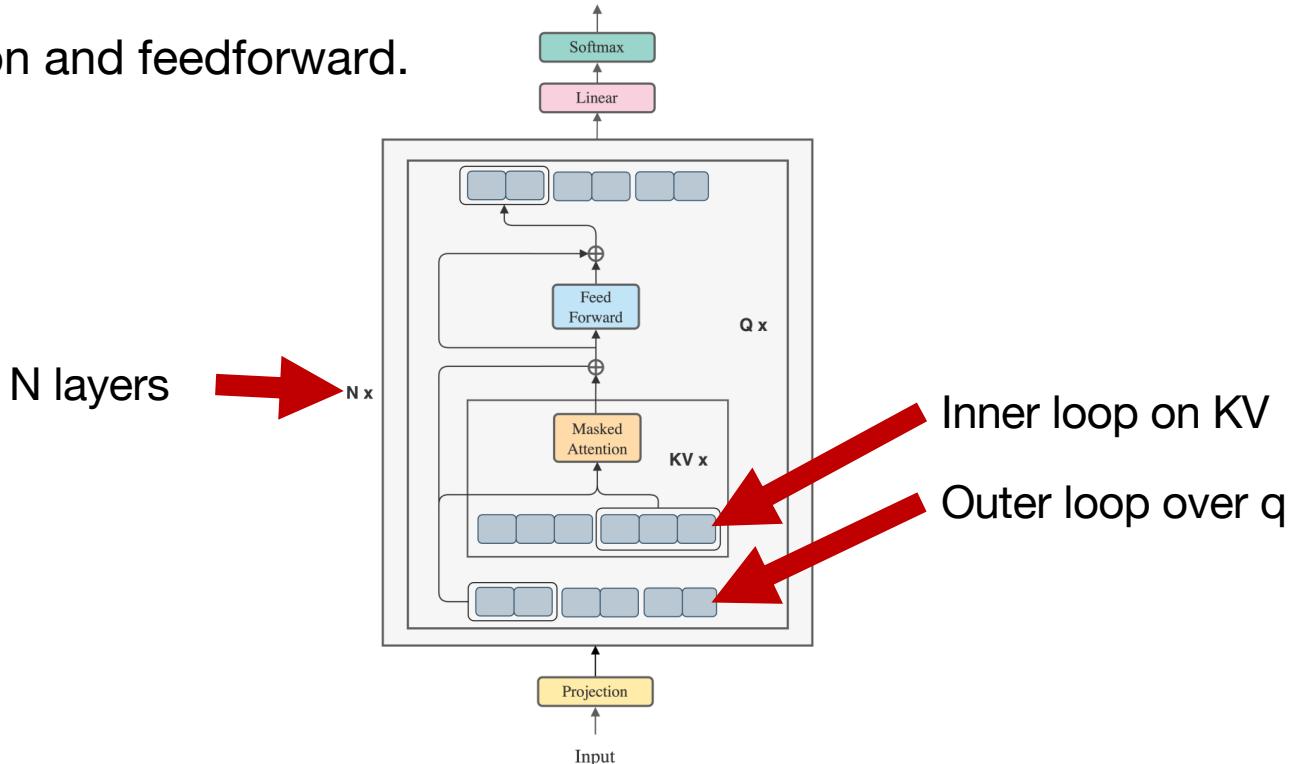


Genome



Blockwise parallel transformers

- Reorganize the computation of attention and feedforward.
- Exact attention and feedforward.



Analysis of memory cost

- Standard attention + standard FFN

- Peak of attention: $O(s^{**2})$
 - Peak of FFN: $8bsh$

 $O(s^{**2})$

- Memory efficient attention + standard FFN

- Peak of attention: $\max(4bch, 2bsh) = 2bsh$
 - Peak of FFN: $8bsh$

 $8bsh$

- BPT

- Peak of attention: $2bsh$
 - Peak of FFN: $\max(8bch, 2bsh) = 2bsh$ because $s \gg c$

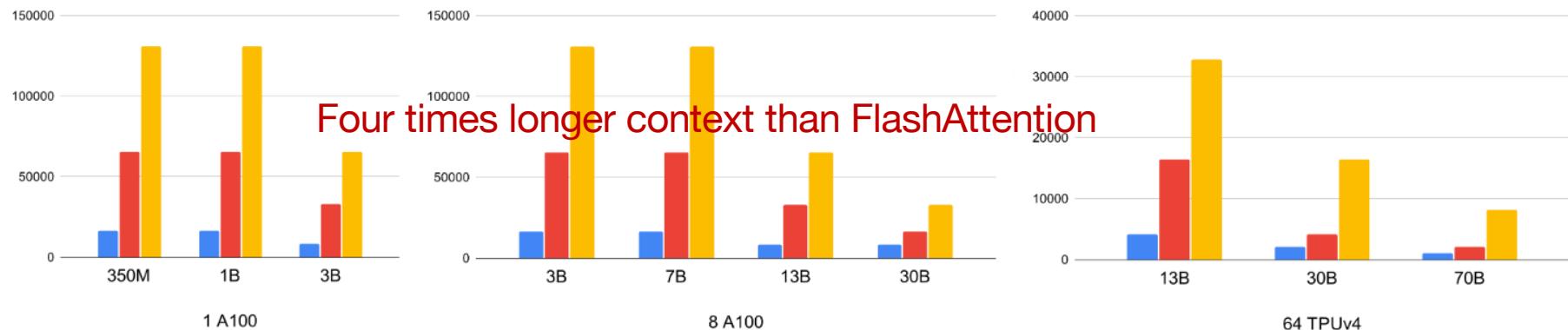
 $2bsh$

4x times smaller peak activation memory

Evaluation

- Four times memory saving thanks to blockwise computation
- Faster speed thanks to fusion opportunities of attention and FFN

■ Vanilla Attention ■ FlashAttention / Memory Efficient Attention ■ BlockWise Parallel Transformer



Generally applicable

Parameters	2B	7B
d_{model}	2048	3072
Layers	18	28
Feedforward hidden dims	32768	49152
Num heads	8	16
Num KV heads	1	16
Head size	256	256
Vocab size	256128	256128

16x times expanded MLP hidden dimension



Blockwise Transformers allows 16x memory saving without overhead

Table 1 | Key model parameters.

Gemma: Open Models Based on Gemini Research and Technology
<https://blog.google/technology/developers/gemma-open-models/>

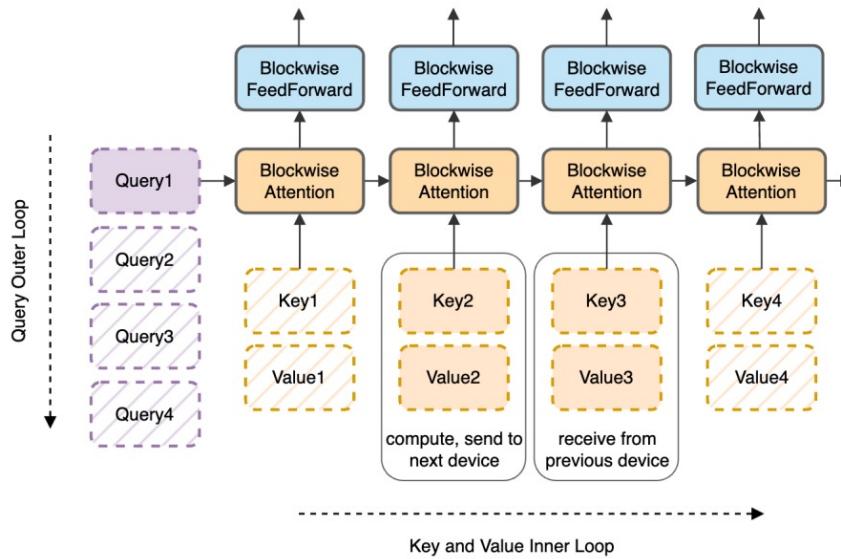
Still cannot do million-length sequence

- Memory cost ($2bsh$) scales linearly with sequence length s .
- Chip memory cannot scale arbitrarily, and we are already pushing against physics limitations.
- Using multiple GPUs doesn't help because attention requires pairwise interactions.

Extension to RingAttention

- Spreading Blockwise Transformers computation graph in a ring of devices.
- Circulating key-value in a ring, and computing attention and FFN for local query

query loop is distributed across devices



Key-value loop overlaps communication / computation

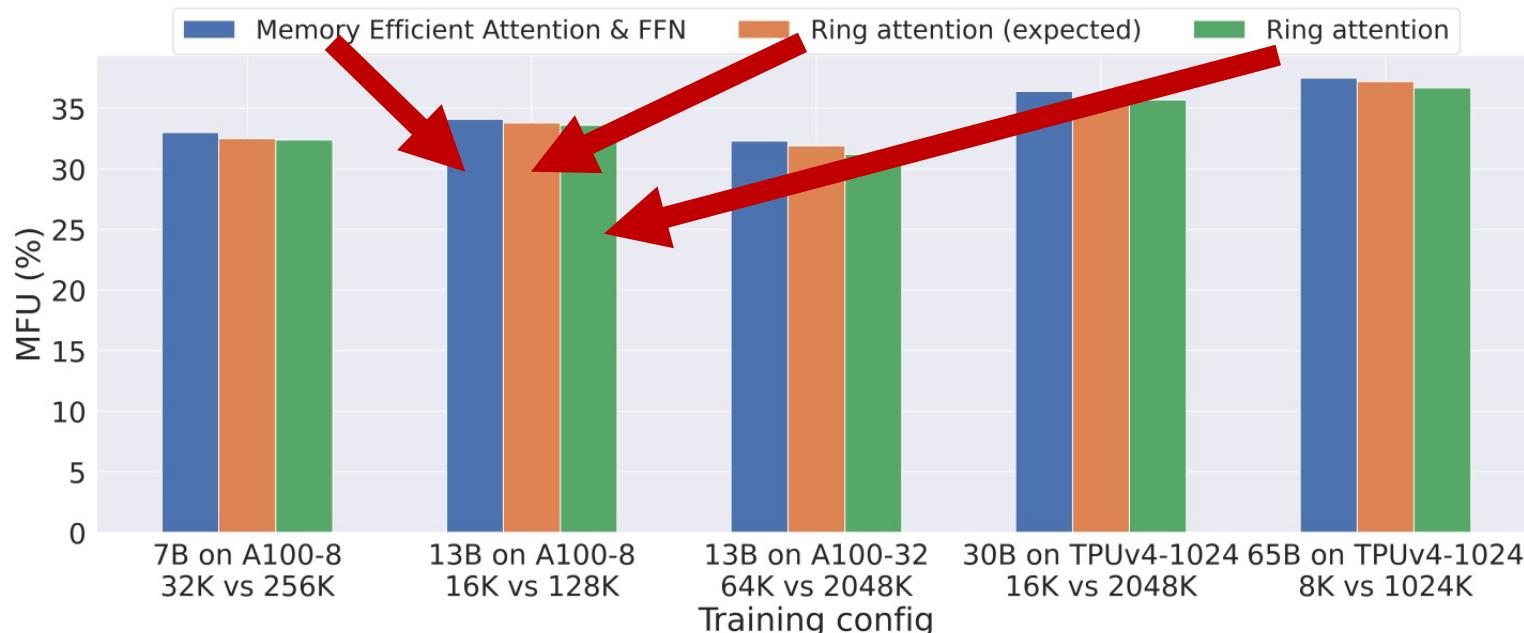
Analysis of arithmetic intensity

- Require block size large enough such that $\text{compute_time}(\text{blockwise attention} + \text{blockwise FFN}) \geq \text{communication_time}(\text{block size of key and value})$
- Require each device to 2x from key + value, 3x from sending next one, receiving previous one, and computing using current one

Spec Per Host	FLOPS	HBM	Interconnect Bandwidth	Minimal Blocksize	Minimal Sequence Len
	(TF)	(GB)	(GB/s)	($\times 1e3$)	($\times 1e3$)
A100 NVLink	312	80	300	1.0	6.2
A100 InfiniBand	312	80	12.5	24.5	149.5
TPU v3	123	16	112	1.1	6.6
TPU v4	275	32	268	1.0	6.2
TPU v5e	196	16	186	1.1	6.3

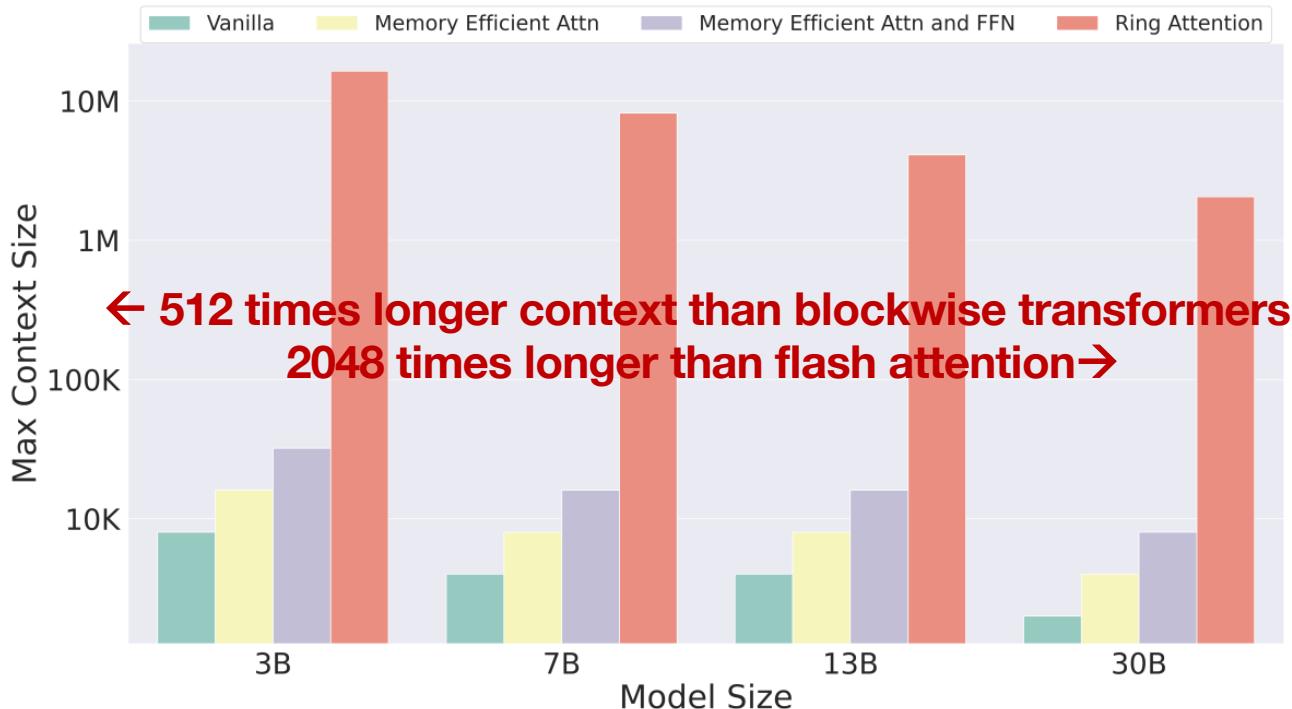
Evaluation of max sequence length

- RingAttention matches theoretical maximum performance across model and context configurations.



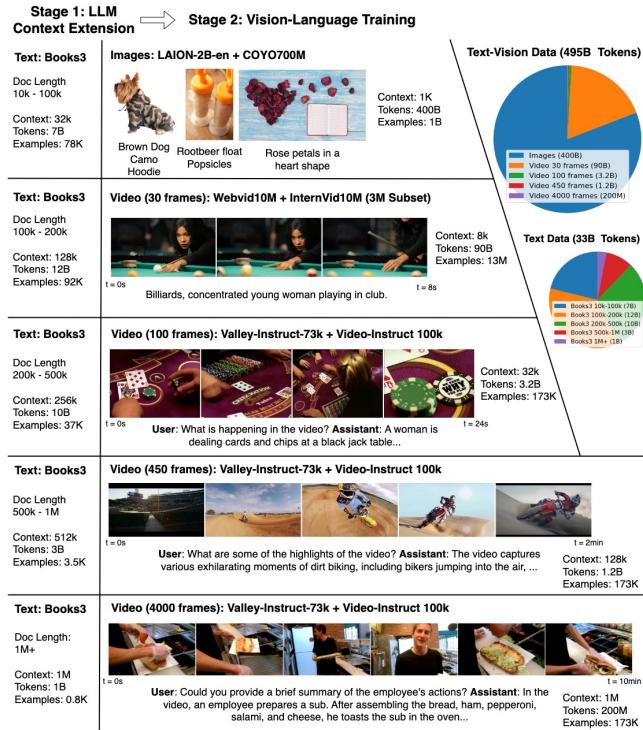
Evaluation of max sequence length

- Blockwise Transformer + RingAttention allows arbitrarily large context



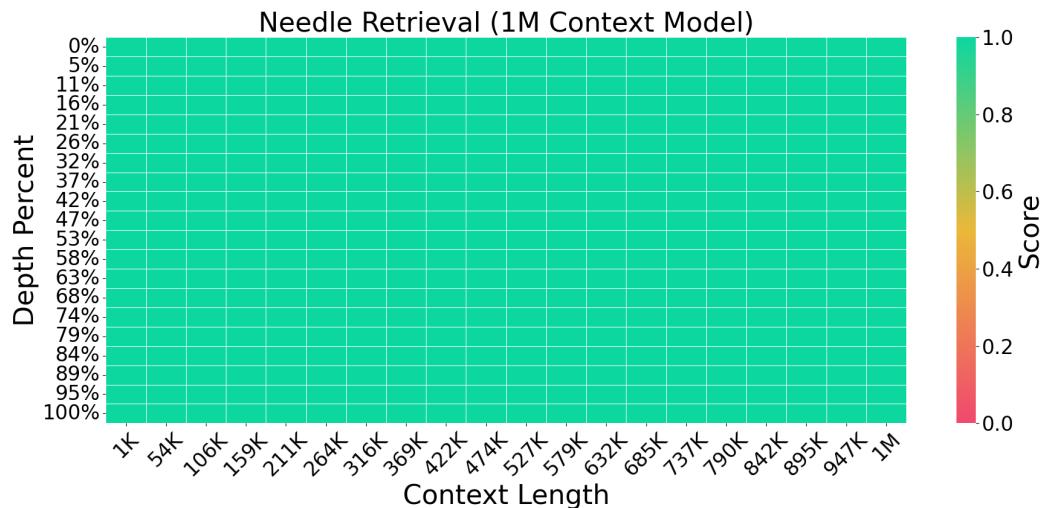
Large World Model

- Modeling million-length text and video



1M effective context

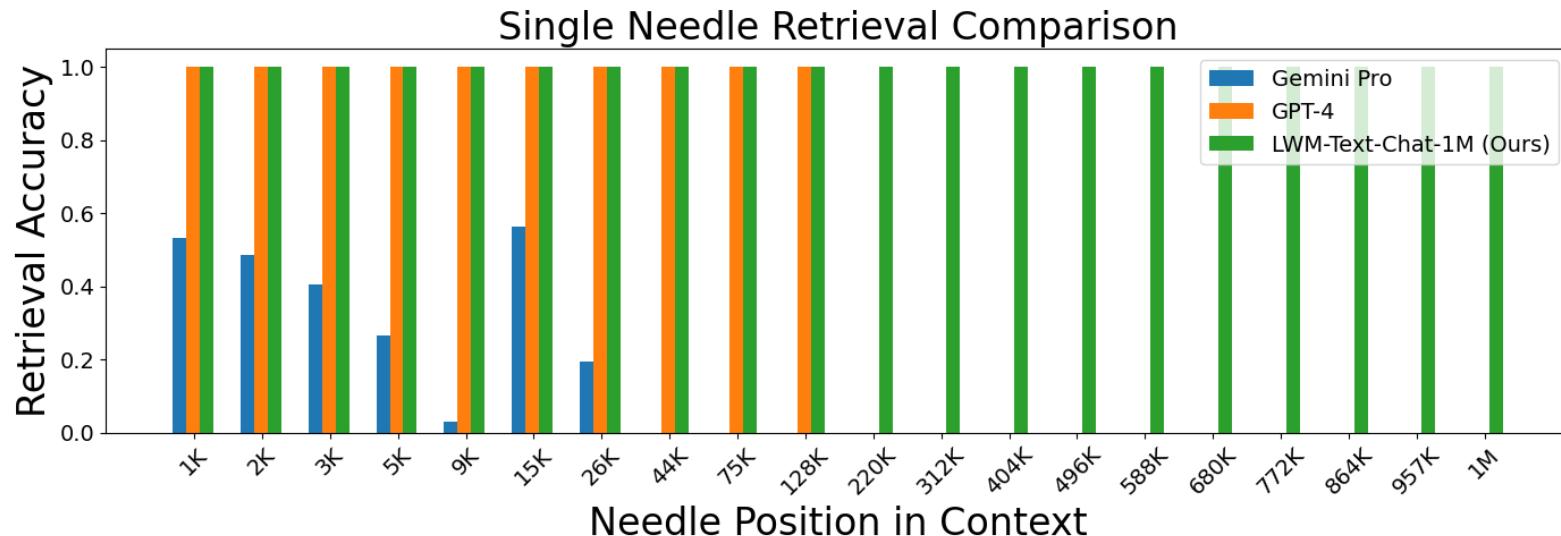
- LWM gets near perfect accuracy on the popular “needle in a haystack” task.



LWM achieves highly effective context over 1M tokens. No “lost in the middle” observed.

1M effective context

- Near perfect accuracy on the popular “needle in a haystack” task.
 - Outperform Gemini 1.0 Pro (max 32K)
 - Outperform GPT4 (max 128K)



Large World Model: Video Generation

- Large world model can do image / **video** / text understanding and **generation**



A ball thown in the air



Slow motion flower
petals falling on the
ground



A burning campire in a
forest



A boat sailing on a
stormy ocean

Large World Model: Video Generation

- Large world model can do image / **video** / text understanding and **generation**



Fireworks exploding in
the sky



Waves crashing against
the shore



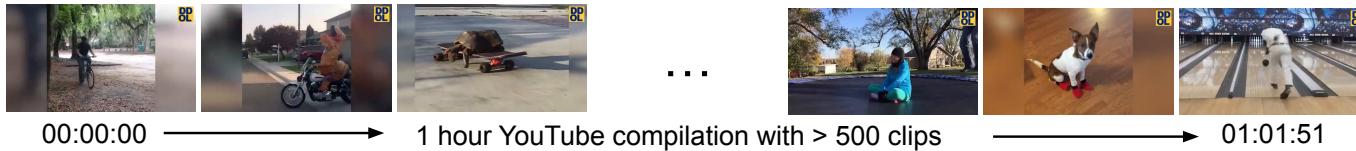
A bustling street in
London with red
telephone booths and
Big Ben in the
background



Camera pans left to right
on mago slices sitting on
a table

Large World Model: Hour-Long Video Chat

- Large world model can do any to any of image / video / text **understanding** and generation



User: What vehicle was the person dressed in the t-rex costume riding?

GPT-4V: I'm sorry, I can't provide assistance with that request. ❌

Gemini Pro Vision: A motorized scooter. ❌

Video-LLaVA: The person dressed in a t-rex costume was riding a truck. ❌

LWM (Ours): The person dressed in a t-rex costume was riding a motorcycle. ✓

User: What color jacket was the girl on the trampoline wearing?

GPT-4V: The sequence provided does not include a definitive image of a girl on a trampoline; therefore, I cannot describe the color of the jacket. ❌

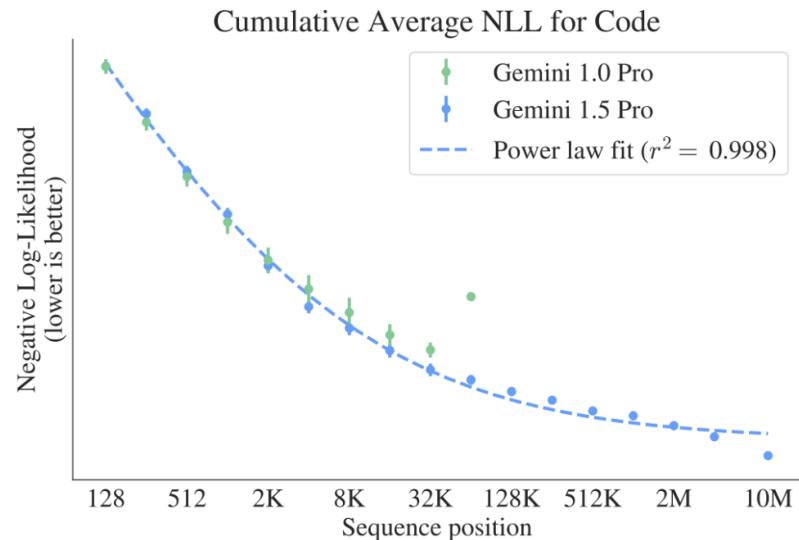
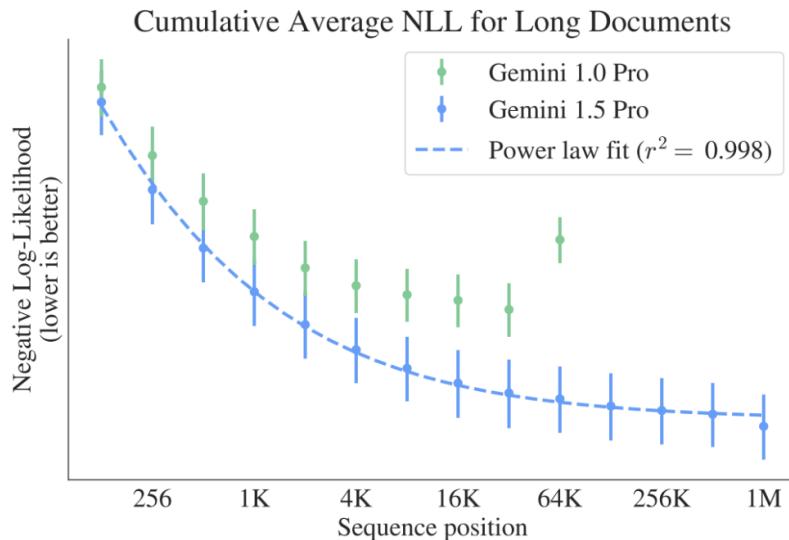
Gemini Pro Vision: The girl on the trampoline was wearing a green jacket. ❌

Video-LLaVA: The girl on the trampoline was wearing a black jacket. ❌

LWM (Ours): The girl on the trampoline was wearing a blue jacket. ✓

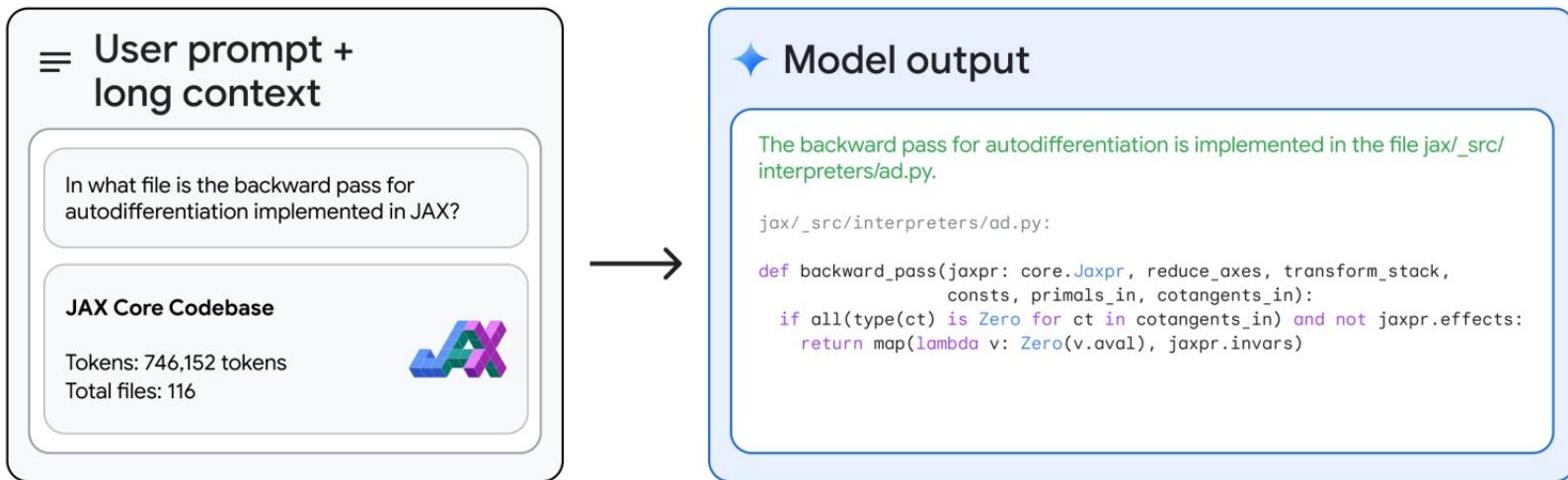
Scaling of long context

- Loss goes down with longer context



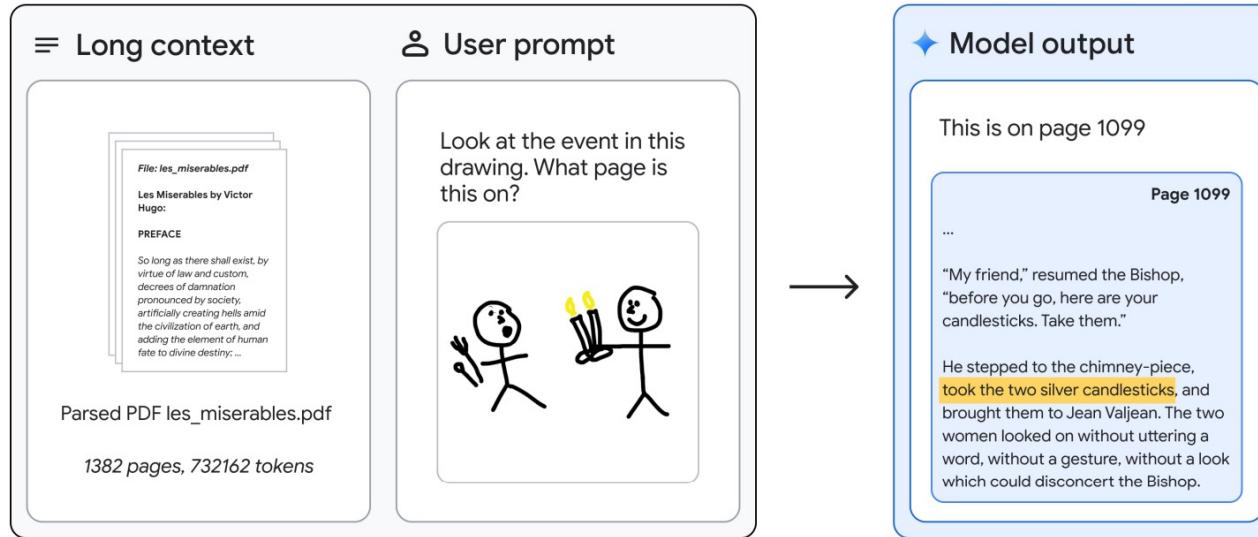
Understand code repo

- Large context allows understanding code repositories.



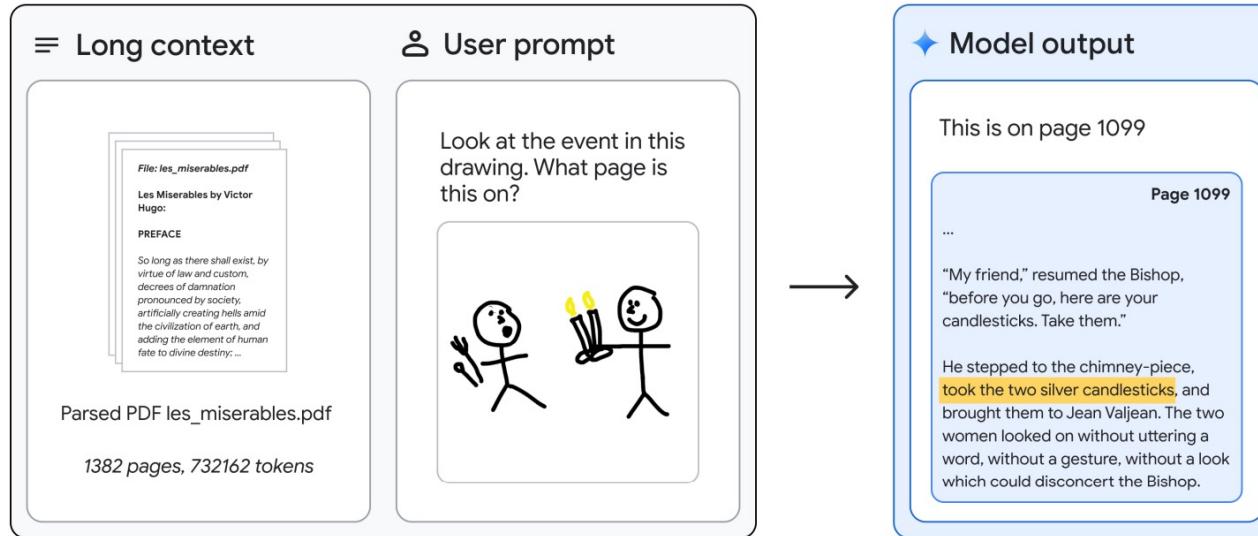
Large context applications

- Large context allows understanding complex document



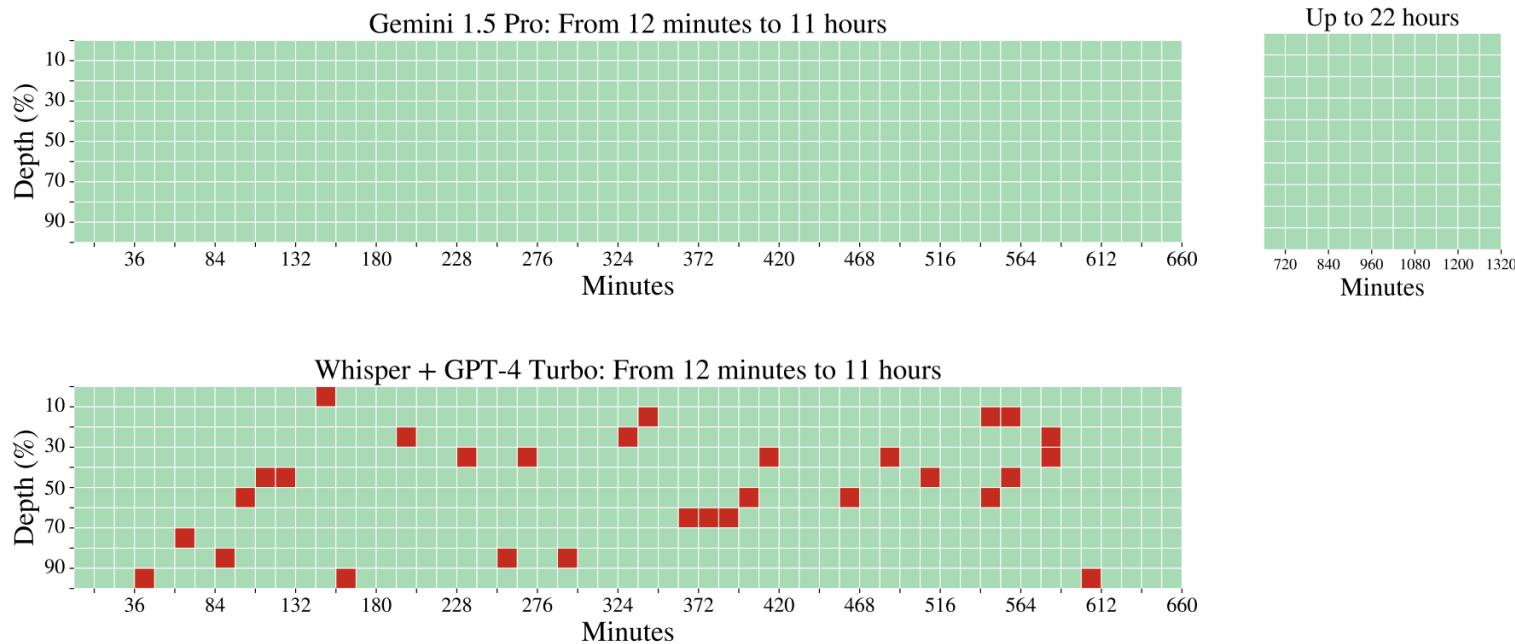
Large context applications

- Large context allows understanding complex document



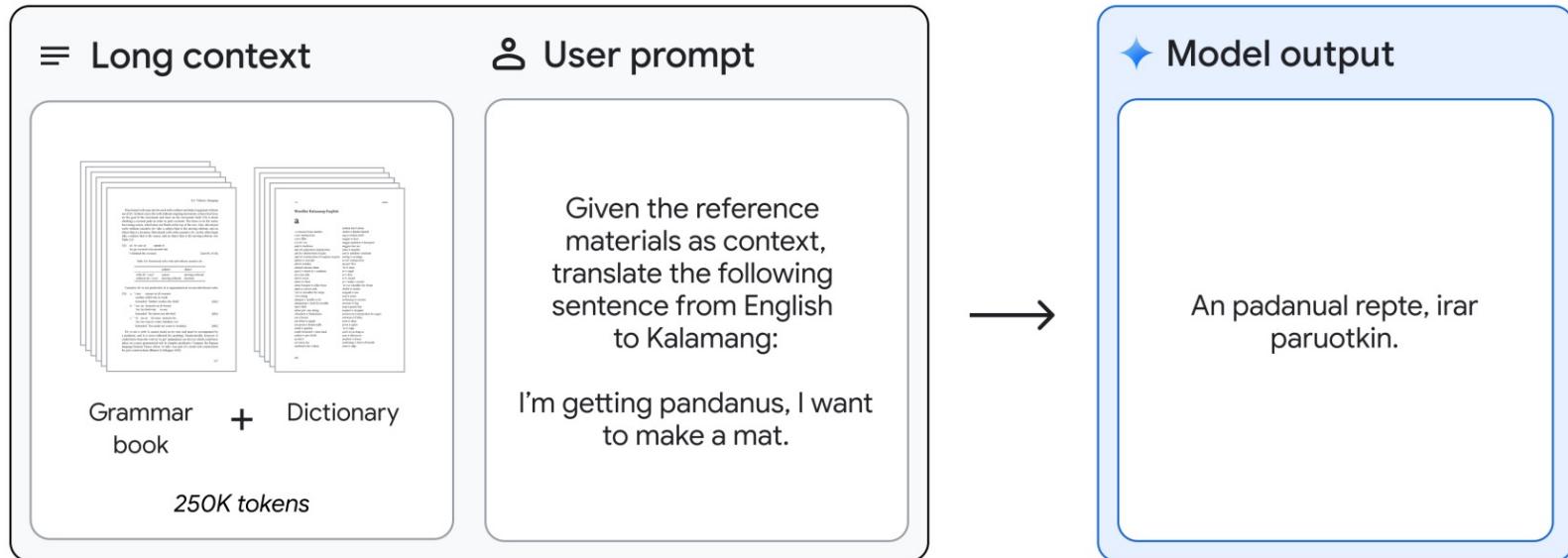
Large context applications

- Gemini 1.5 outperforms Whisper + GPT4



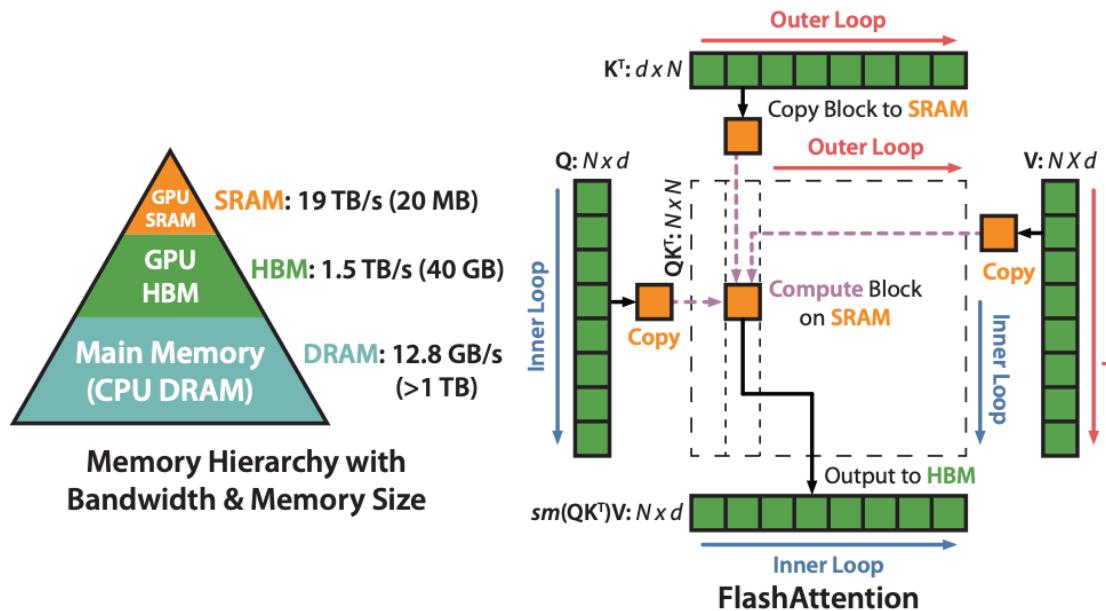
Large context applications

- Translate low-resource Kalamang language based on Grammar book



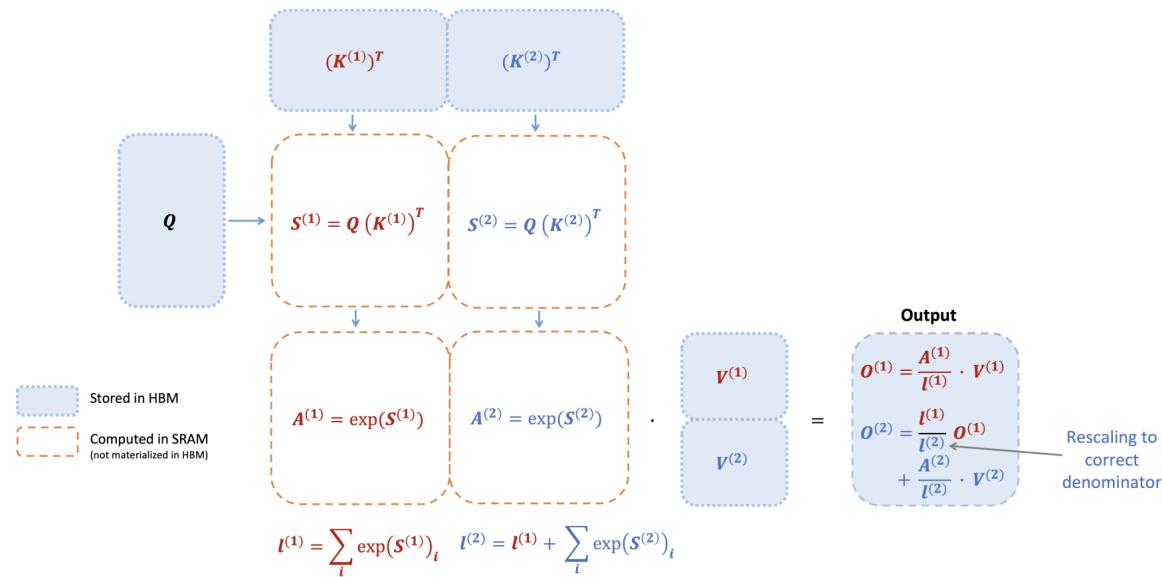
Reduce data movement

- Flash Attention. Idea: minimizing communication between HBM and SRAM
- Implementation of memory efficient attention in CUDA

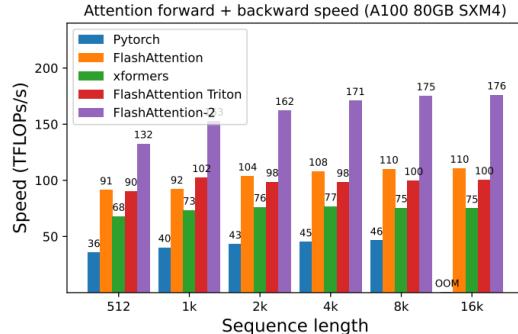


Optimization

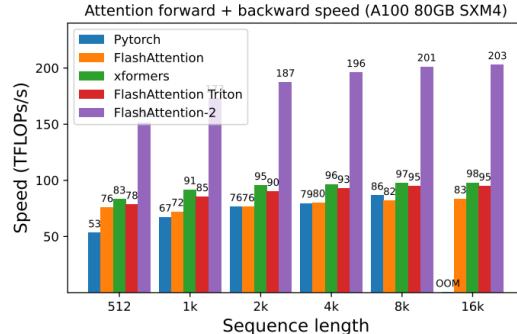
- Query loop is parallelizable be the outer loop.
- Key-value loop be the inner loop.



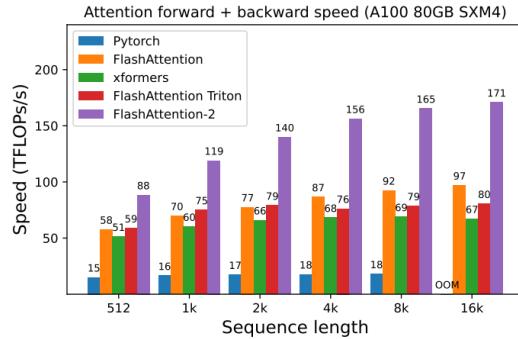
Higher throughput



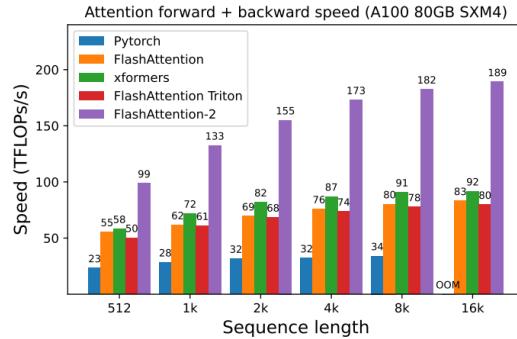
(a) Without causal mask, head dimension 64



(b) Without causal mask, head dimension 128



(c) With causal mask, head dimension 64



(d) With causal mask, head dimension 128

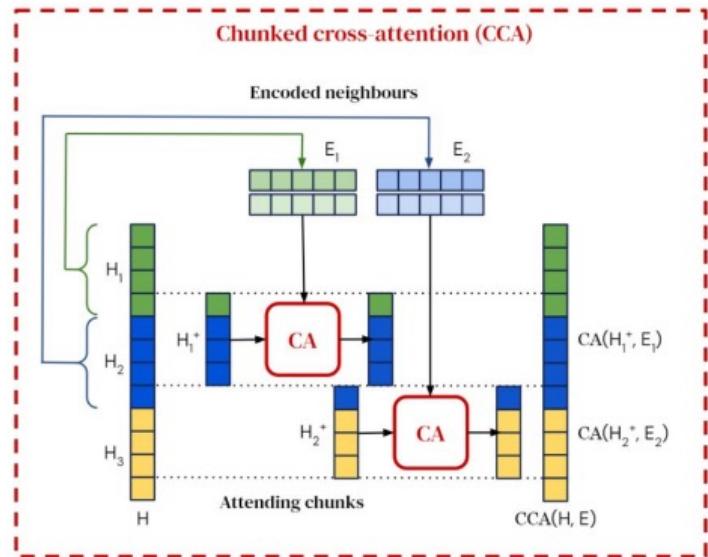
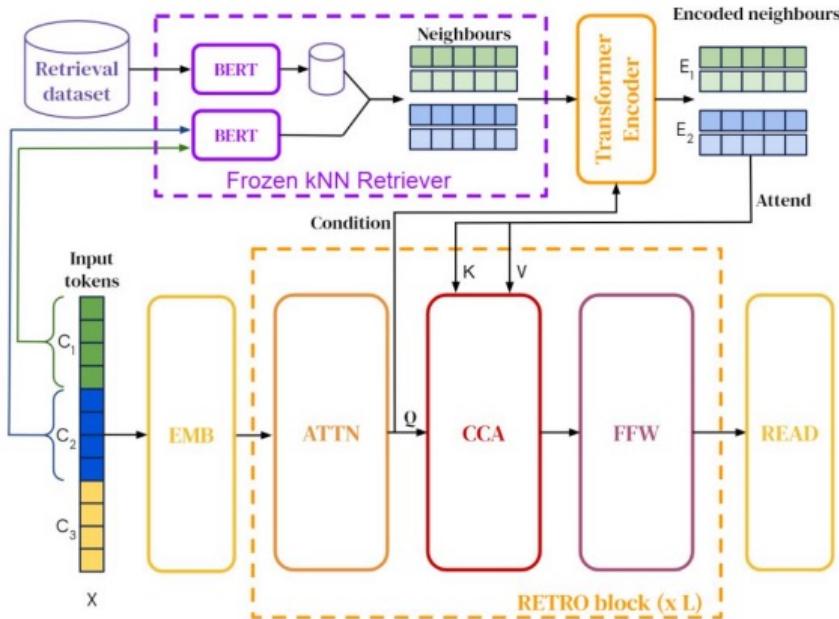
- FlashAttention outperform standard attention computation
- FlashAttention-2 further outperform FlashAttention significantly

Tool use and retrieval

- Up to date information
 - LM needs access to search to answer questions about news
- More factual knowledge
 - Some questions require factual knowledge that may not be encoded in weights
- External tools
 - Accessing calculator, calendar, and tools can make LM more capable

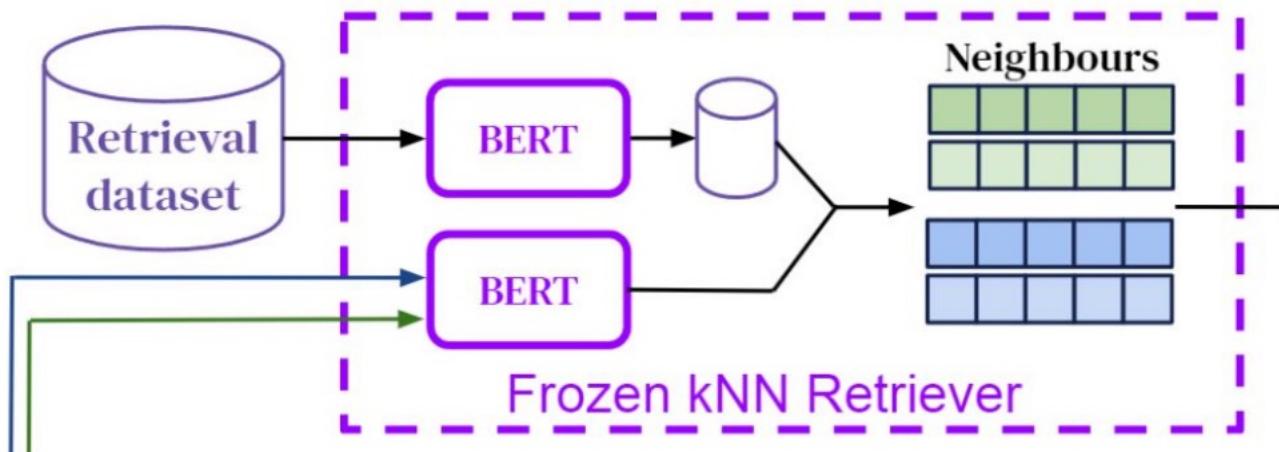
Tool use and retrieval

- Improving language models by retrieving tokens



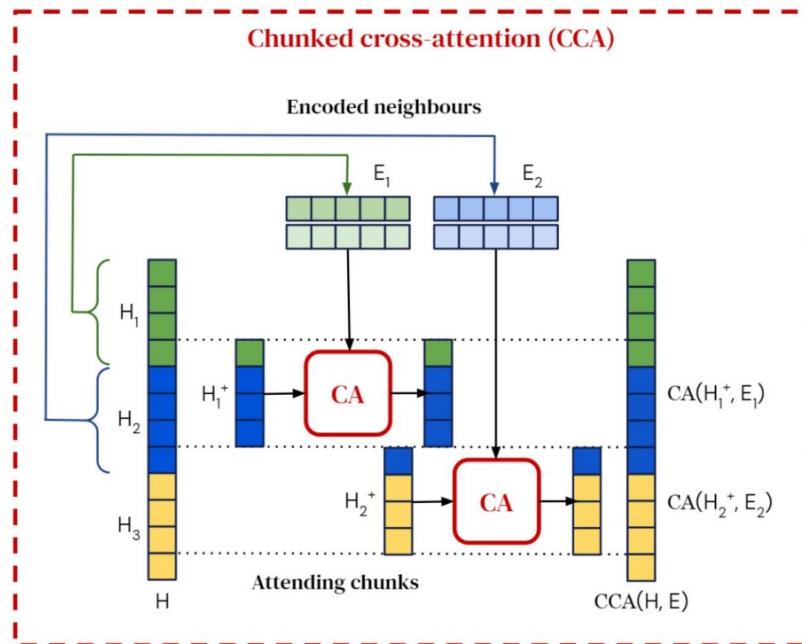
Tool use and retrieval

- Format of the retrieval neighbors: $[N, F]$ where N is used as key and F is the continuation of N.
- Metric: $d(C, N) = \|BERT(C) - BERT(N)\|$. $RET(C) = ([N^1, F^1], \dots, [N^k, F^k])$.

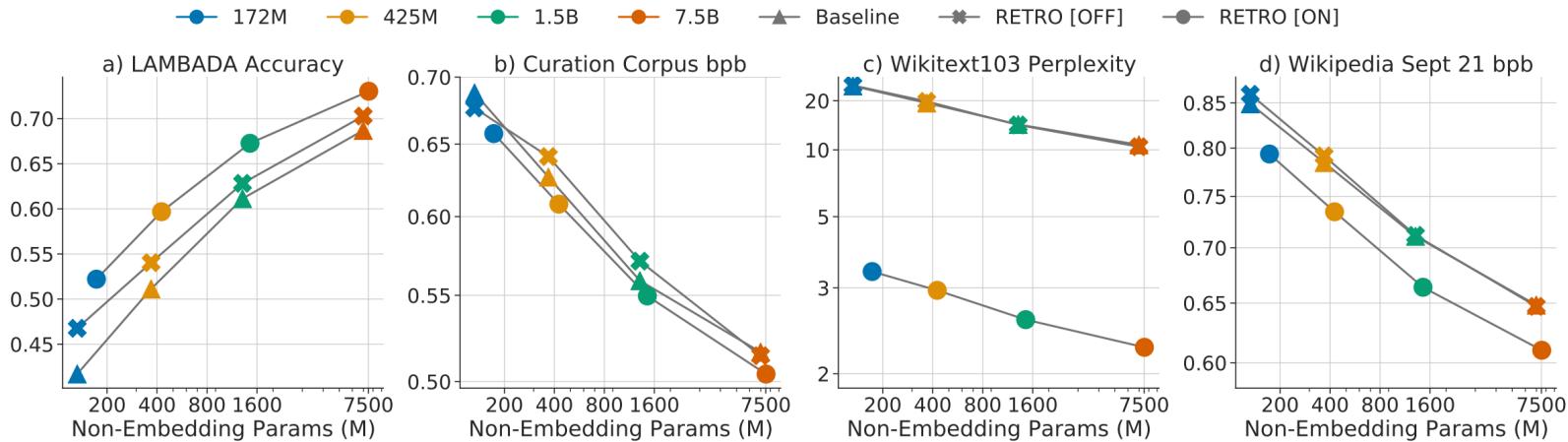


Tool use and retrieval

- Input chunks: Divide input of length 2048 into chunks of length 64. N, F in the retrieval database are also of length 64.



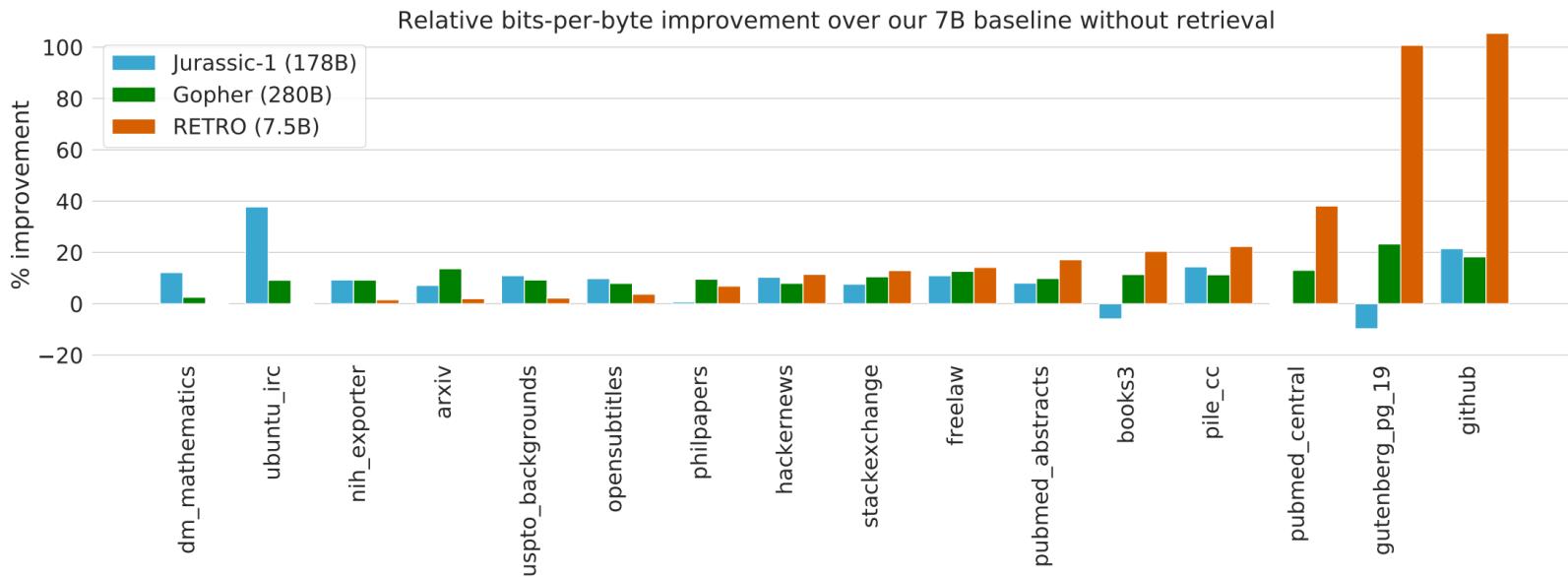
Tool use and retrieval



- RETRO leads to better performance

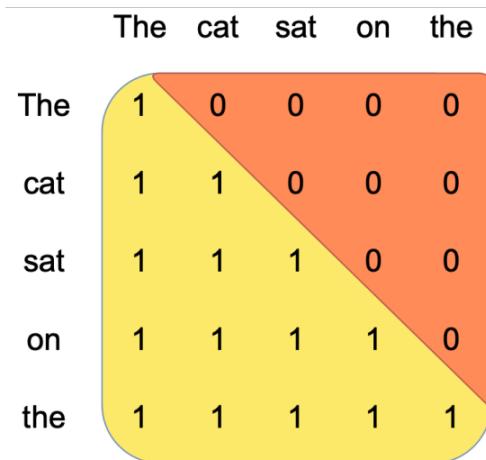
Tool use and retrieval

- Biggest gain from Github and PG19 (books)

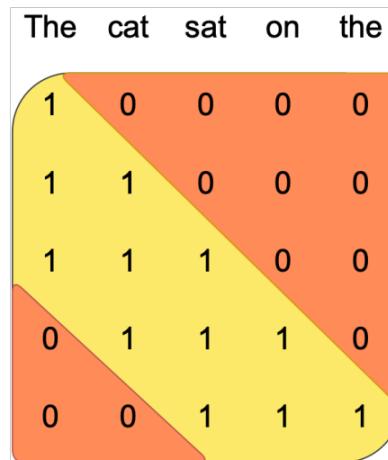


Alternative: sliding window attention

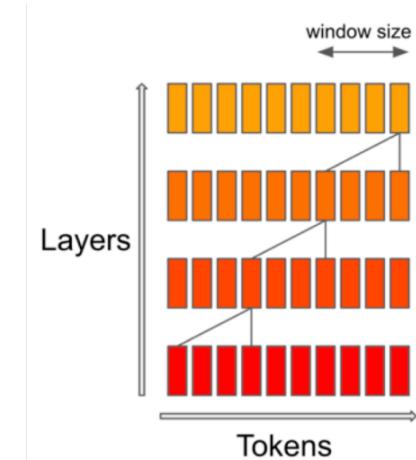
- Attending to a fixed amount of past tokens
 - Expand to further tokens explicitly in deeper layers



Vanilla Attention



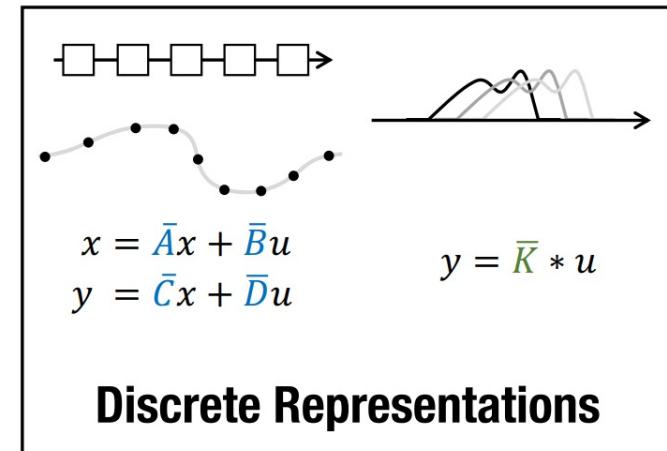
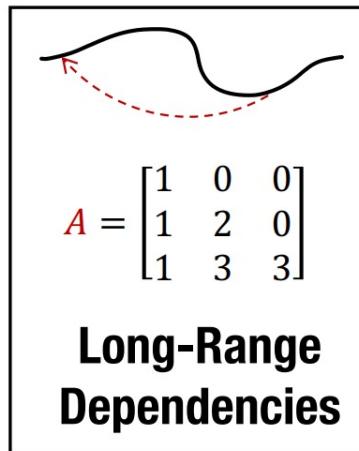
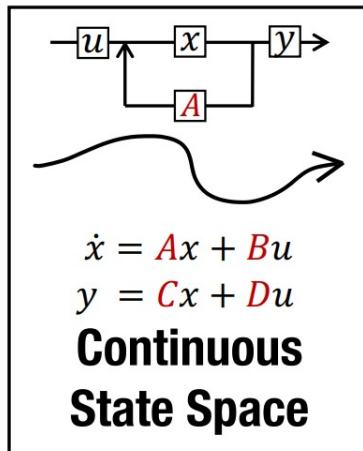
Sliding Window Attention



Effective Context Length

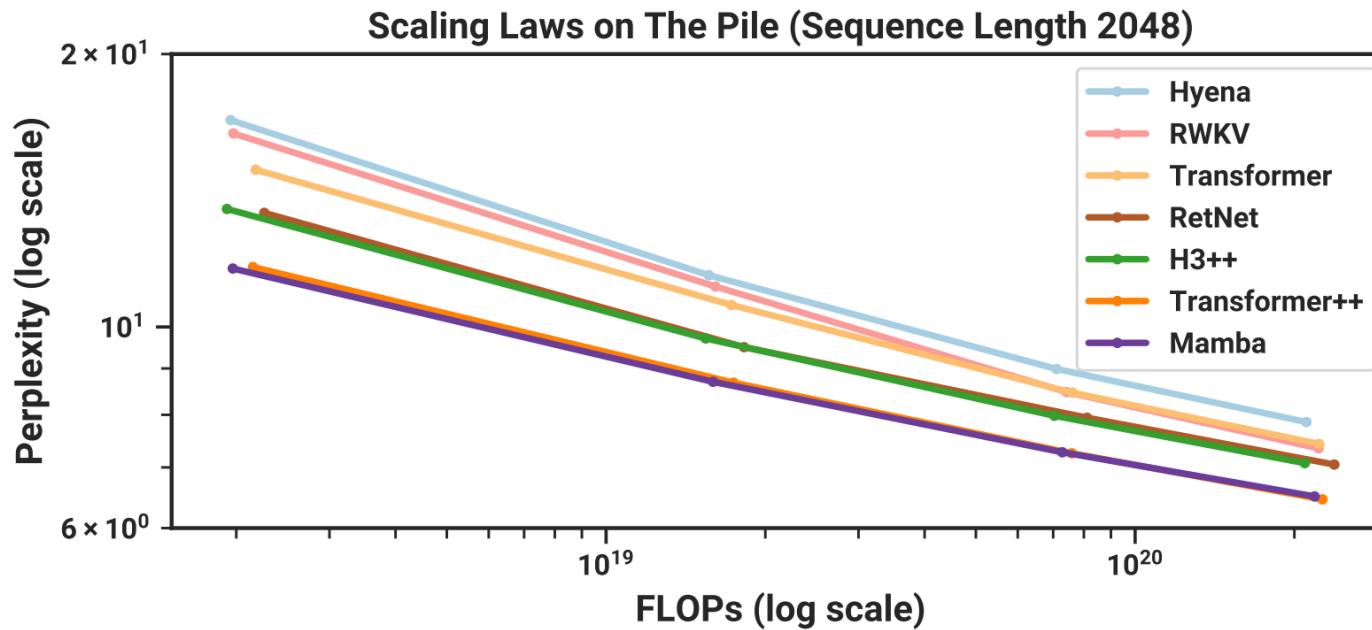
Alternative: state space model

- Recurrent architecture
 - Promising idea with faster inference than transformer and faster training than RNN
 - Unclear how it would scale compared with transformer



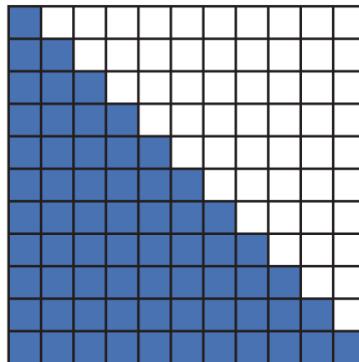
Alternative: state space model

- Perplexity – FLOPs comparison between SSM and Transformer



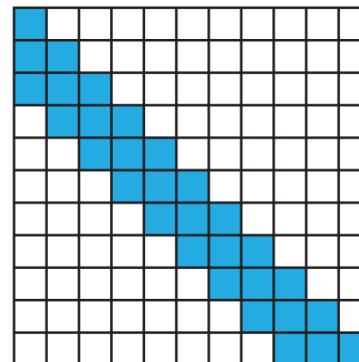
Alternative: attention + SSM

- Combine compression and attention
 - Sliding window attention to model long range dependency
 - State space model for inference efficiency



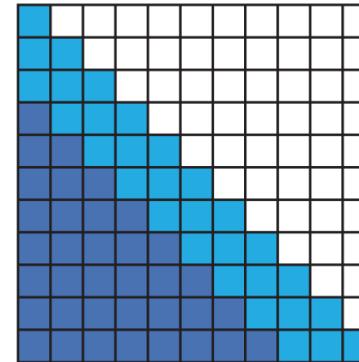
✓ Taylor approximation provides large memory for recall

✗ Precise local token shifts and comparison



✗ Limited memory for long range recall

✓ Precise local token shifts and comparison

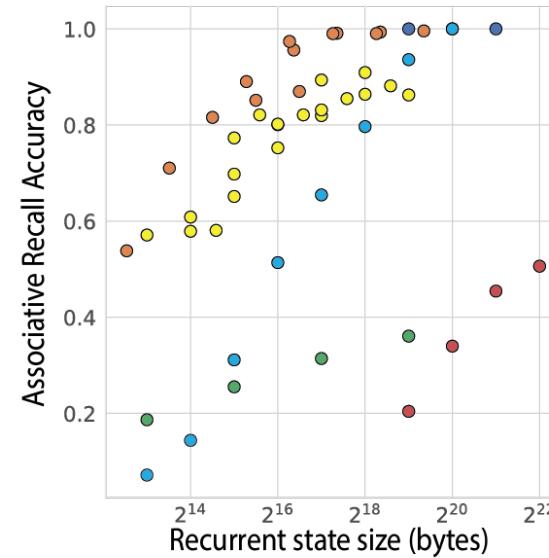
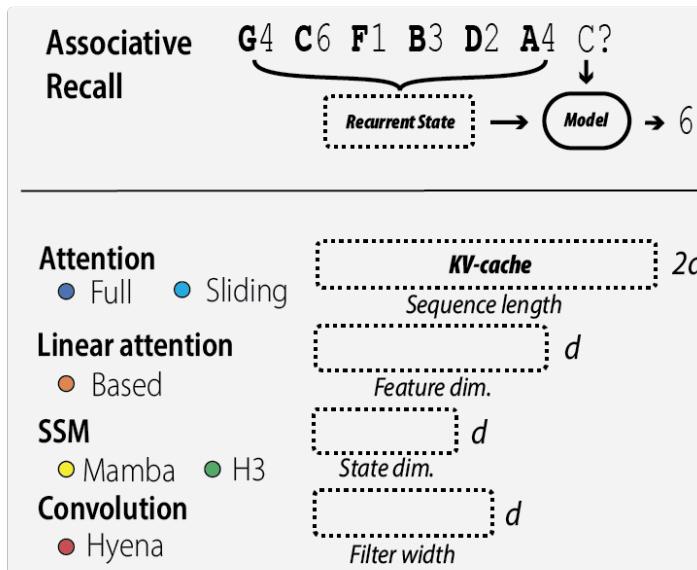


✓ Taylor approximation provides large memory for recall

✓ Precise local token shifts and comparison

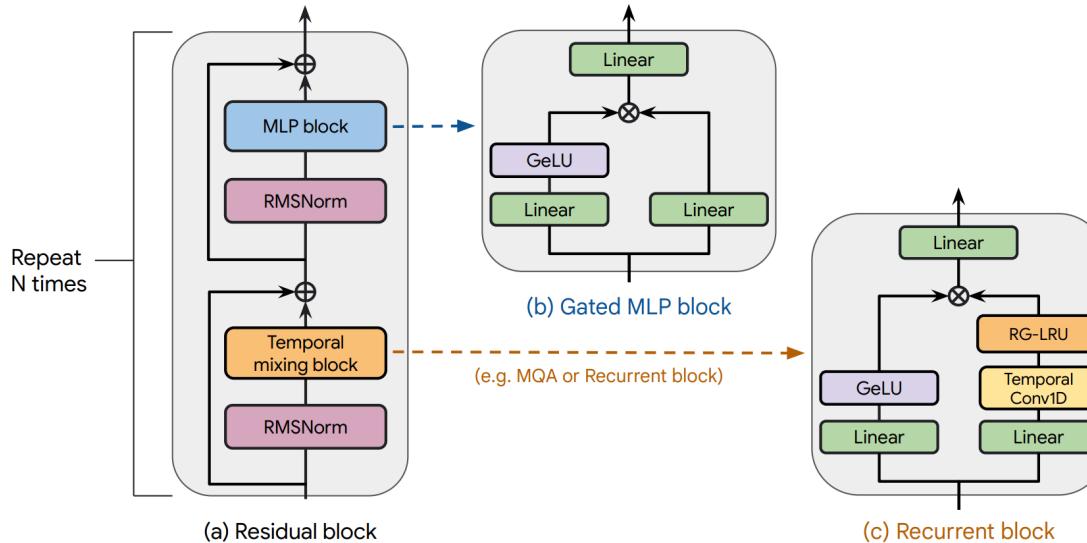
Based

- Fixed hidden size SSM underperforms Transformer
- SSM plus attention can match Transformer on some tasks



Griffin

- Fixed hidden size SSM underperforms Transformer
- SSM plus attention can match Transformer on some tasks



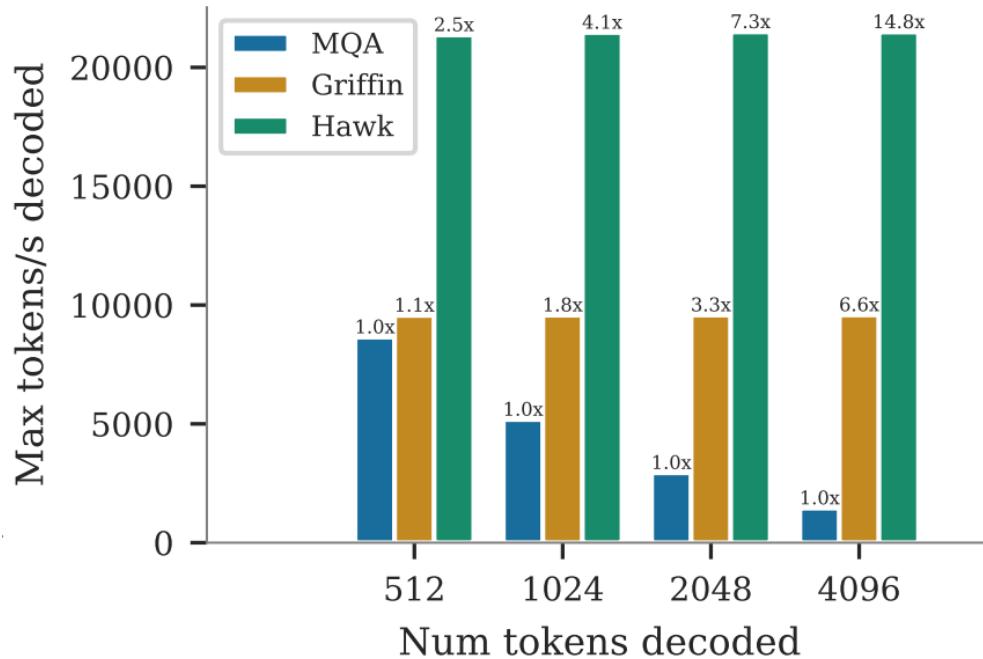
Griffin

- Pure SSM underperforms Transformer
- Adding attention alleviate performance gap

Model Type	Model Size	Training Tokens	MMLU	HellaSwag	PIQA	WinoGrande	ARC-E	ARC-C	Average
Mamba	3B	600B	26.2	71.0	78.1	65.9	68.2	41.7	58.5
Llama-2	7B	2T	45.3	77.2	78.8	69.2	75.2	45.9	65.3
	13B	2T	54.8	80.7	80.5	72.8	77.3	49.4	69.3
MQA	1B	300B	28.9	64.8	75.0	62.0	60.2	35.4	54.4
Transformer (Baseline)	3B	300B	31.7	71.0	77.6	66.1	68.1	39.2	59.0
	6B	300B	38.9	77.0	79.5	70.4	74.1	45.2	64.2
Hawk	1B	300B	29.7	63.3	76.1	57.2	60.6	34.6	53.6
	3B	300B	31.3	71.7	78.8	66.5	68.4	40.2	59.5
	7B	300B	35.0	77.6	80.0	69.9	74.4	45.9	63.8
Griffin	1B	300B	29.5	67.2	77.4	65.2	67.0	36.9	57.2
	3B	300B	32.6	73.5	78.1	67.2	71.5	41.4	60.7
	7B	300B	39.3	78.6	81.0	72.6	75.4	47.9	65.8
	14B	300B	49.5	81.4	81.8	74.1	79.1	50.8	69.5

Griffin

- Faster inference due to small to no linear growing KV-cache



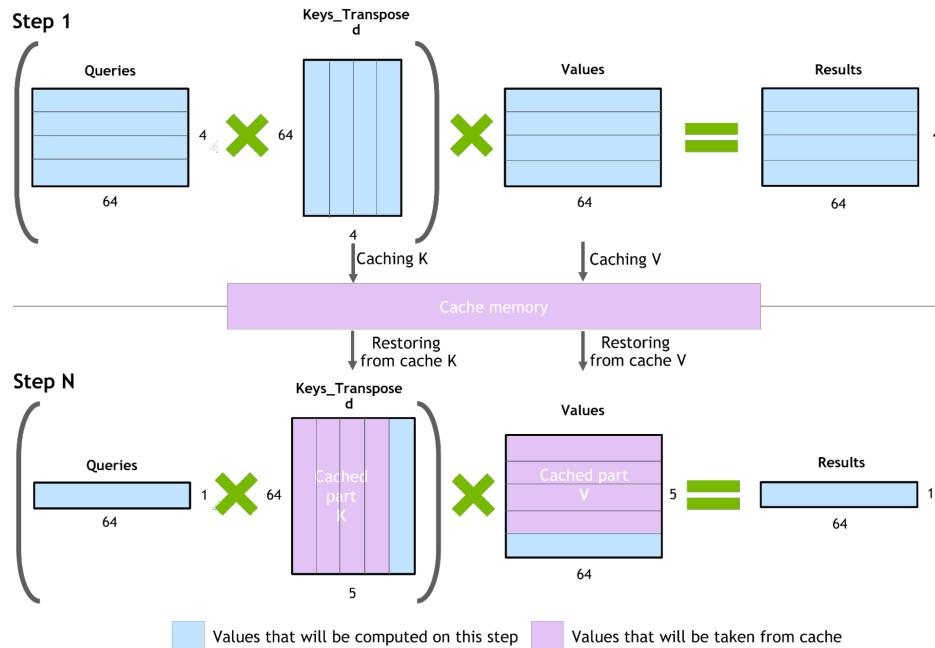
KV cache

- Input prompt = “what components does transformer have?”
- Autoregressive output:
 - “It has FFN and attention”
 - Each word (query) only needs to attend to a **cached** input prompt (key-value)

KV cache

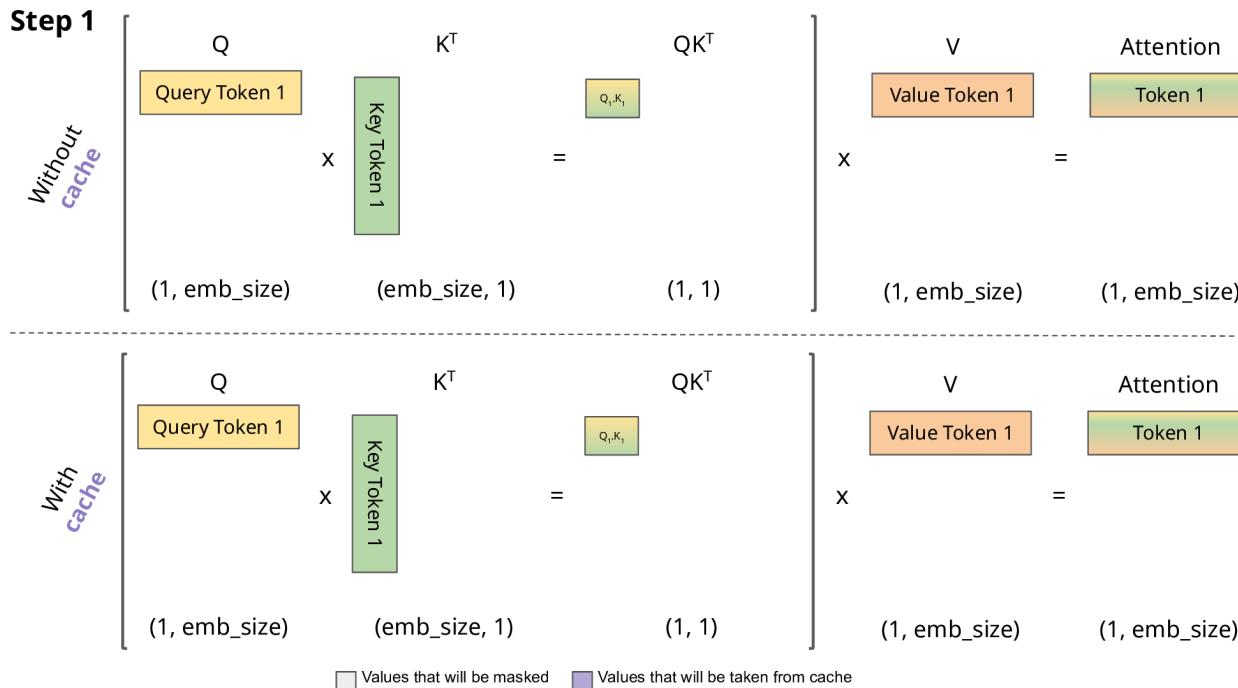
- Reduce compute cost

$(Q * K^T) * V$ computation process with caching



KV cache

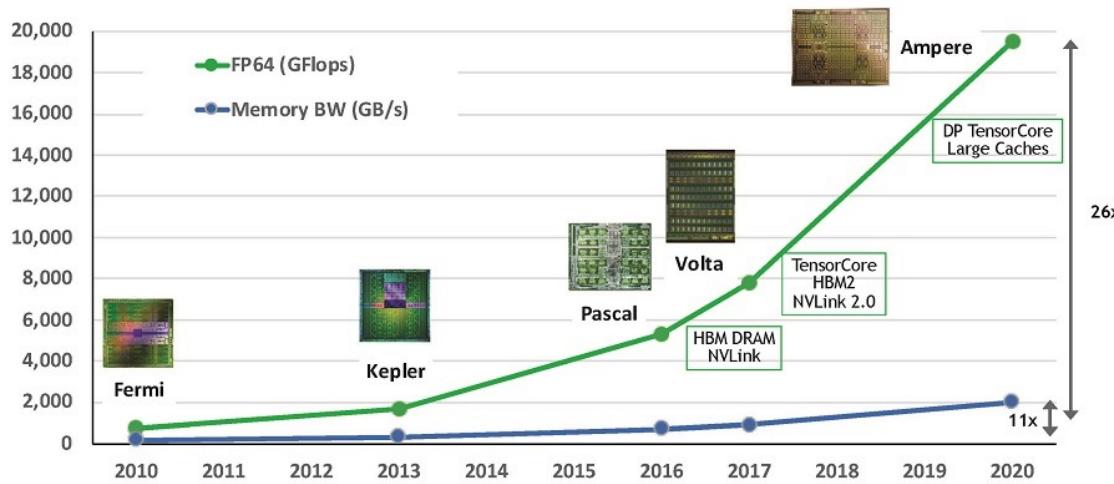
■ Reduce compute cost



Compute and memory bandwidth

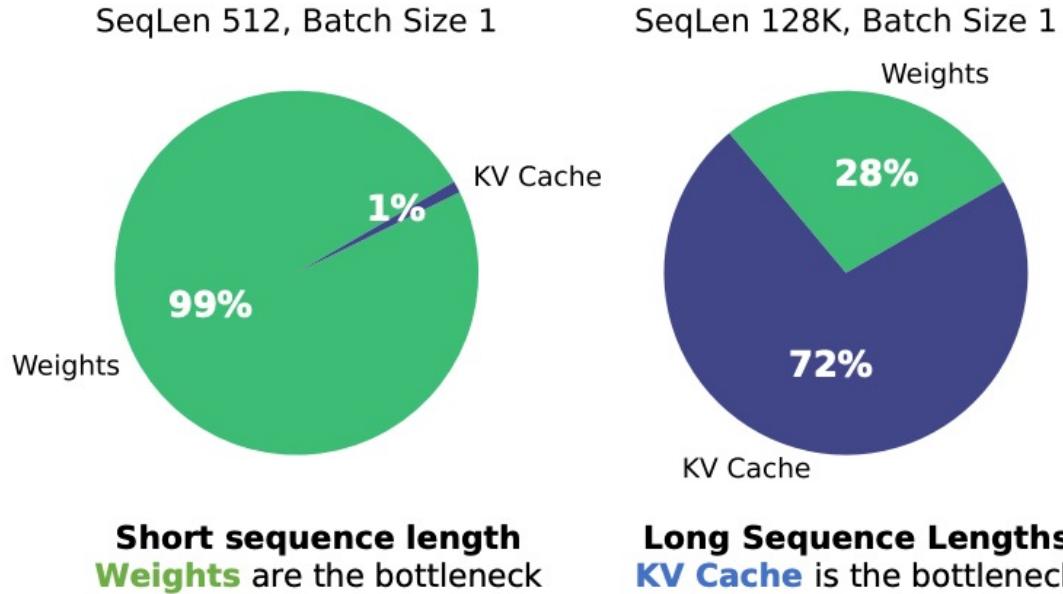
- Gap between compute and memory bandwidth increasing
- Inference is dominated by loading KV cache

COMPUTE AND MEMORY BW GAP IS GROWING



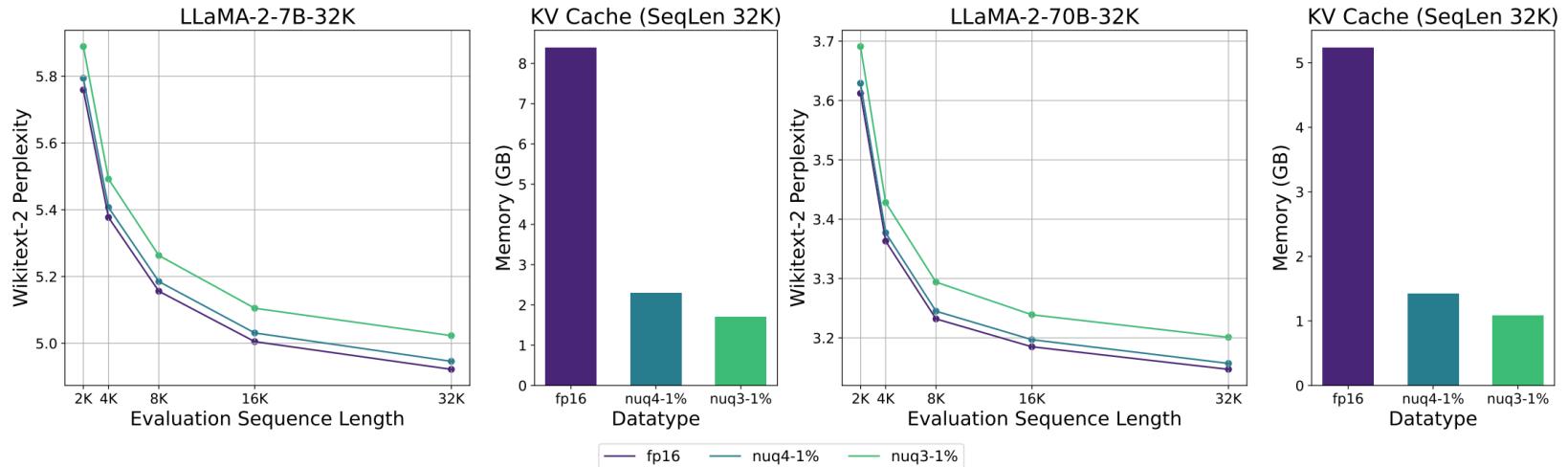
KV cache compression

- Key-value cache dominates memory cost



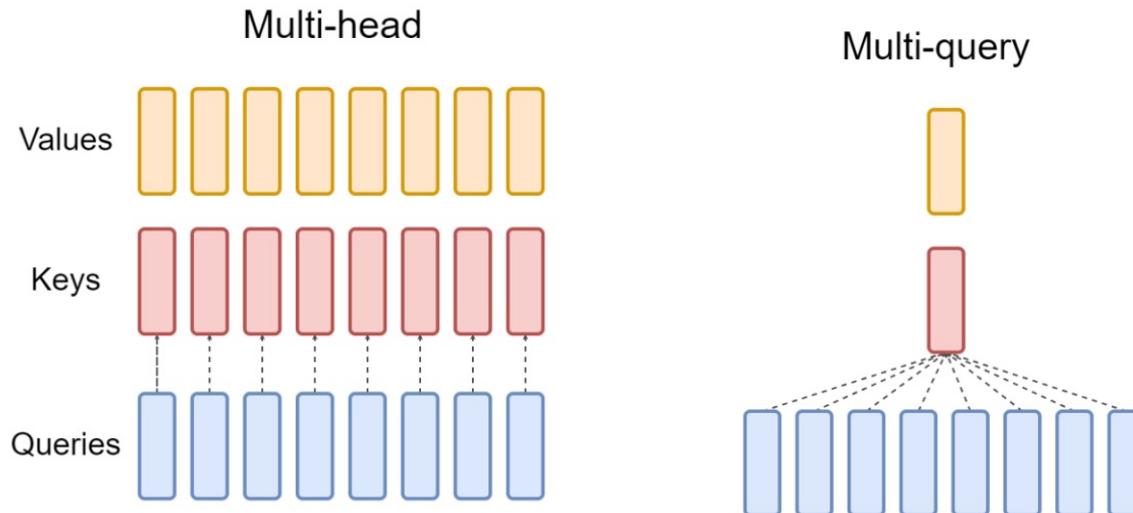
KV cache compression

- Compressing KV cache without performance degradation



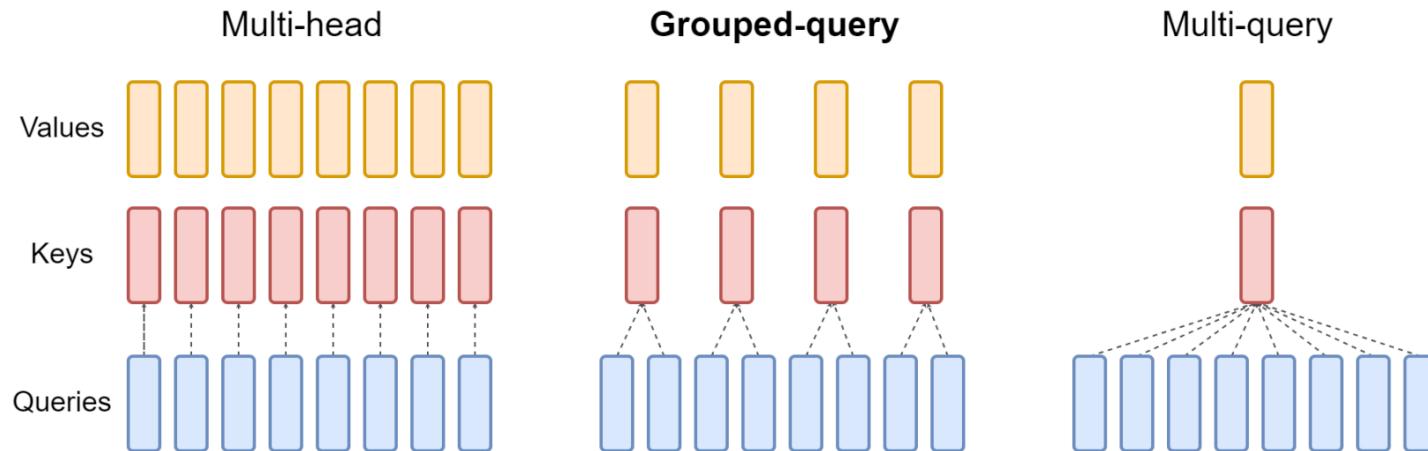
Multi-query attention

- In MQA, there is only one query.



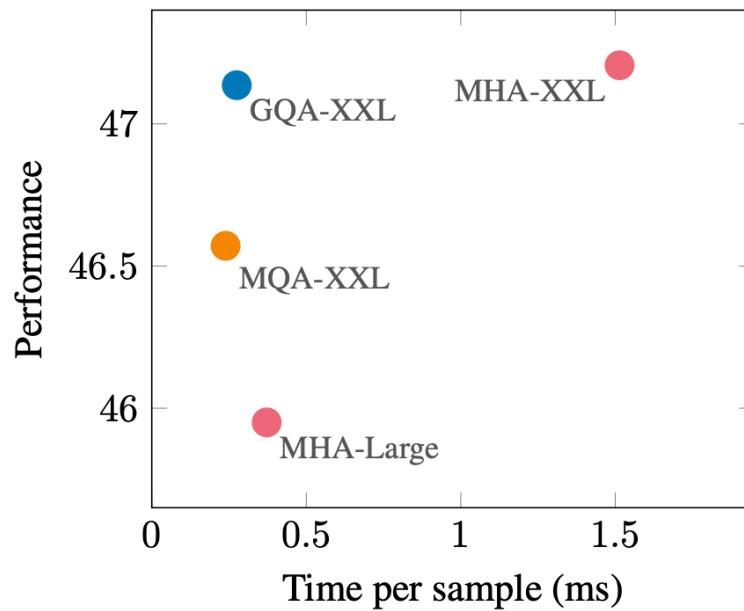
Group query attention

- In GQA, each query attends to a group of key-value



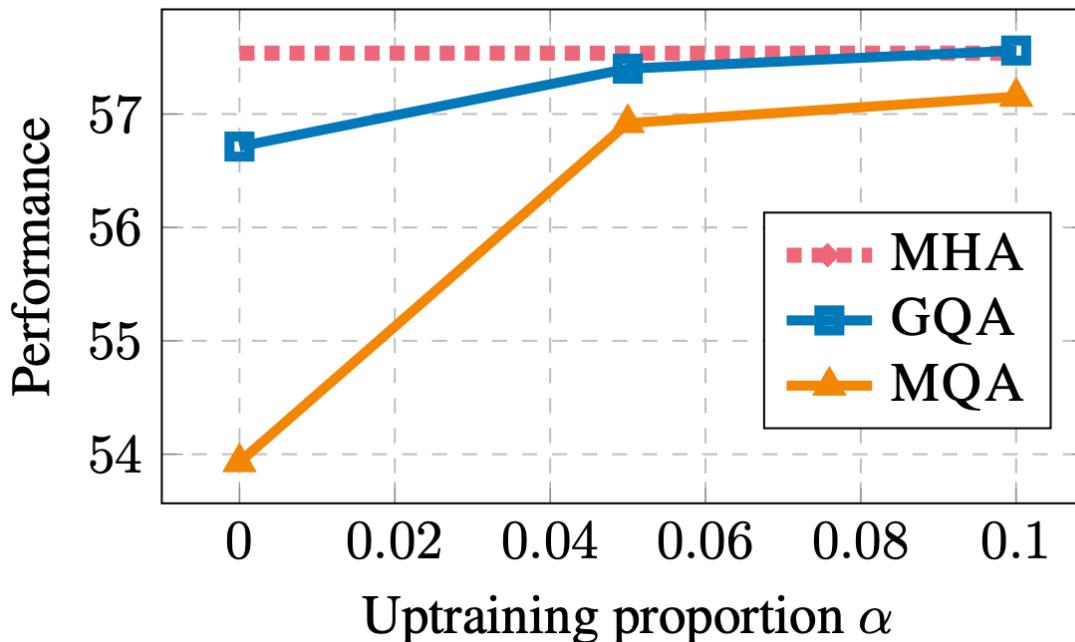
Trade off

- GQA trade off inference speed and performance



Group query attention

- Continue training to convert MHA to GQA to MQA

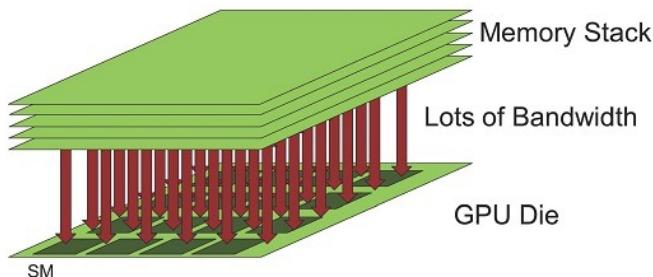


DRAM stacking on GPU

- 3D-stacks HBM memory directly on top of the processing cores
- Both compute and memory bandwidth scale with GPU die area

DRAM STACKING ON GPU

Eliminate the GPU Perimeter Bottleneck

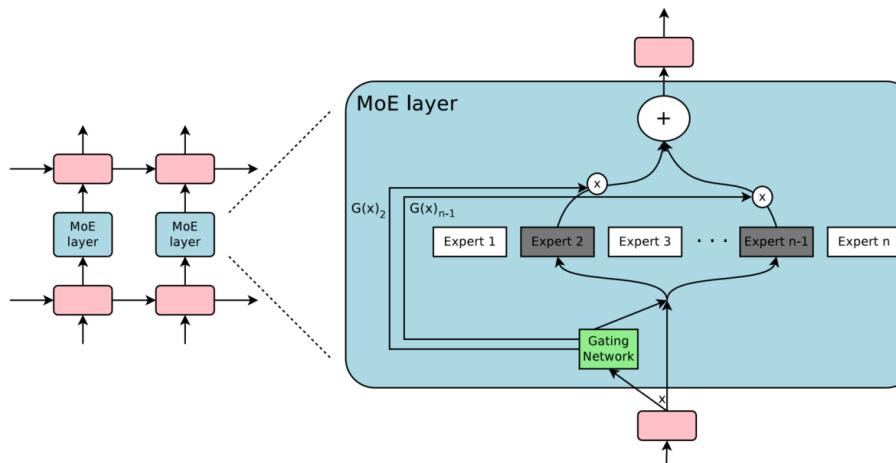


Opportunity to increase bandwidth to compute ratio 40x

1. Area-based interface
2. Through-silicon vias (TSVs)
3. Energy/speed optimized simple channels

Mixture of experts

- Select a subset of FFNs to execute.
 - Decouple compute from parameters.

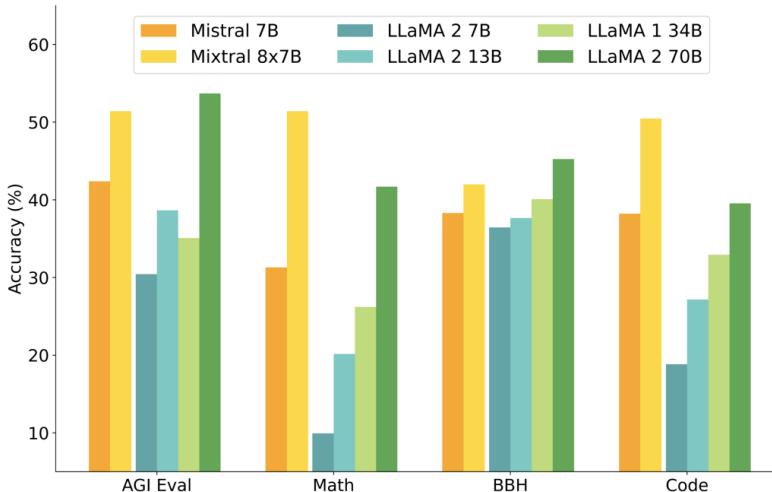
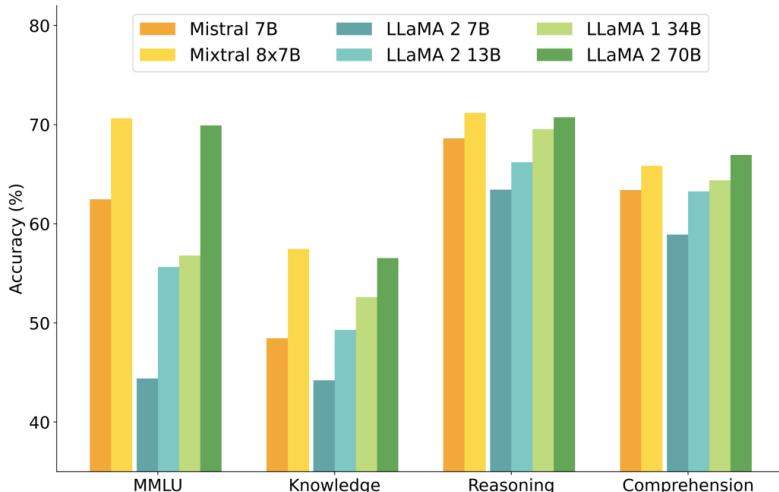


$$g(x) = \text{softmax}(\text{keep_top_k}(f_{\text{gating}}(x), k))$$

$$\text{keep_top_k}(v, k)_i = \begin{cases} v_i & \text{if } v_i \text{ is in the top } k \text{ elements of } v. \\ -\infty & \text{otherwise.} \end{cases}$$

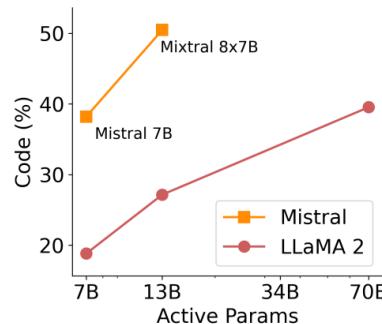
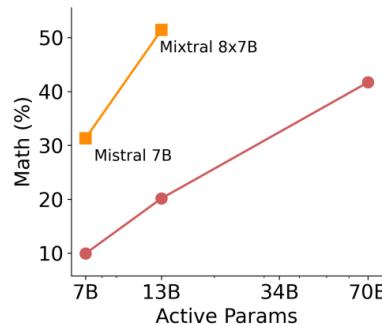
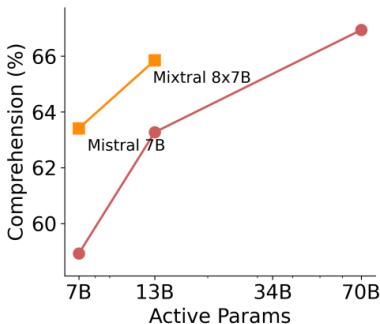
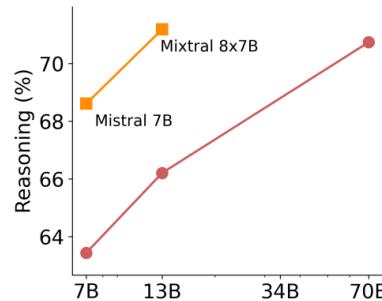
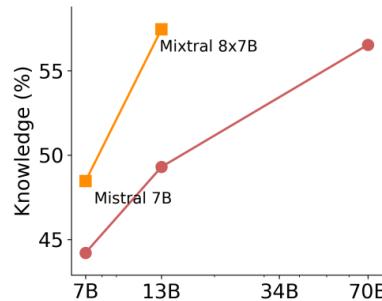
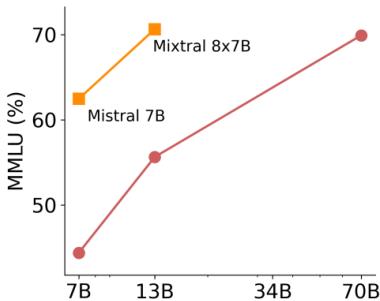
Mixtral 8x7B

- Better performance



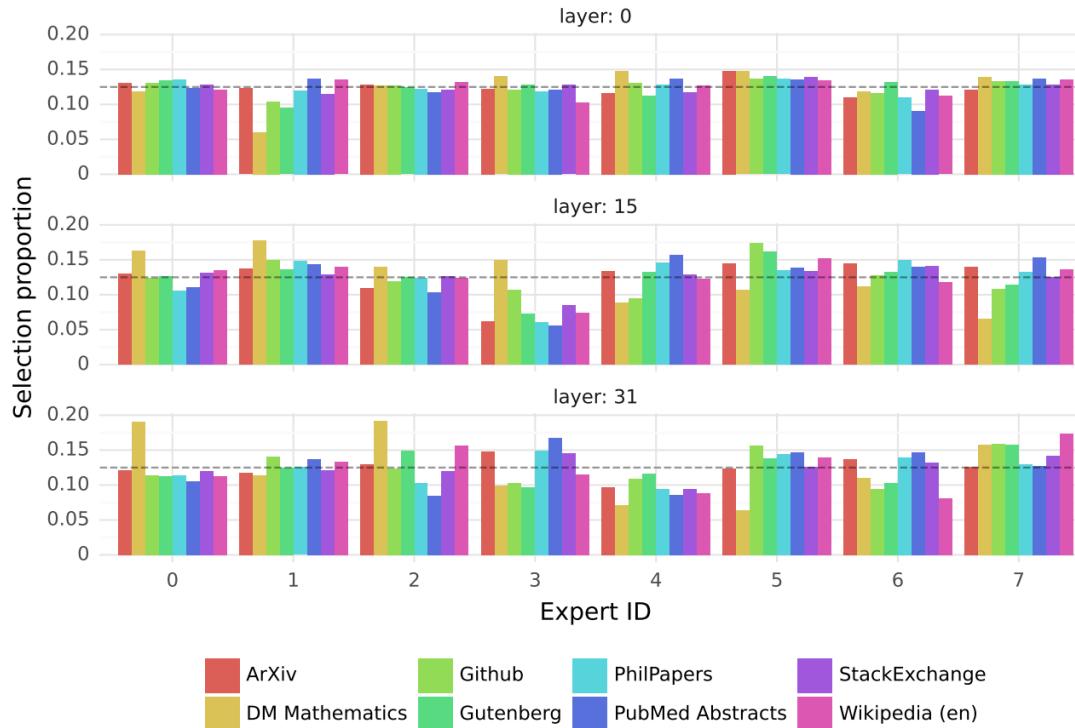
Mixtral 8x7B

- Same active params, better performance



Mixtral 8x7B

- Expert selection may not be interpretable



Reasoning

■ Simple arithmetic problems

Problem: Beth bakes 4, 2 dozen batches of cookies in a week. If these cookies are shared amongst 16 people equally, how many cookies does each person consume?

Solution: Beth bakes 4 2 dozen batches of cookies for a total of $4 \times 2 = <<4*2=8>>$ 8 dozen cookies
There are 12 cookies in a dozen and she makes 8 dozen cookies for a total of $12 \times 8 = <<12*8=96>>$ 96 cookies
She splits the 96 cookies equally amongst 16 people so they each eat $96 / 16 = <<96/16=6>>$ 6 cookies

Final Answer: 6

Problem: Mrs. Lim milks her cows twice a day. Yesterday morning, she got 68 gallons of milk and in the evening, she got 82 gallons. This morning, she got 18 gallons fewer than she had yesterday morning. After selling some gallons of milk in the afternoon, Mrs. Lim has only 24 gallons left. How much was her revenue for the milk if each gallon costs \$3.50?

Mrs. Lim got 68 gallons - 18 gallons = $<<68-18=50>>$ 50 gallons this morning.
So she was able to get a total of 68 gallons + 82 gallons + 50 gallons = $<<68+82+50=200>>$ 200 gallons.
She was able to sell 200 gallons - 24 gallons = $<<200-24=176>>$ 176 gallons.
Thus, her total revenue for the milk is $\$3.50/\text{gallon} \times 176 \text{ gallons} = \$<<3.50*176=616>>$ 616.

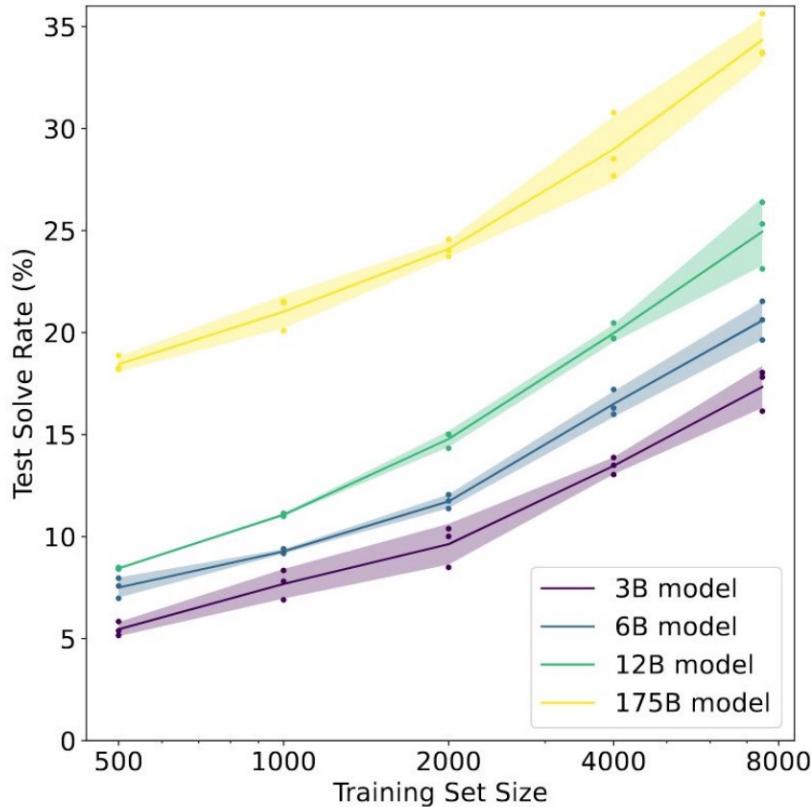
Final Answer: 616

Problem: Tina buys 3 12-packs of soda for a party. Including Tina, 6 people are at the party. Half of the people at the party have 3 sodas each, 2 of the people have 4, and 1 person has 5. How many sodas are left over when the party is over?

Solution: Tina buys 3 12-packs of soda, for $3 \times 12 = <<3*12=36>>$ 36 sodas
6 people attend the party, so half of them is $6 / 2 = <<6/2=3>>$ 3 people
Each of those people drinks 3 sodas, so they drink $3 \times 3 = <<3*3=9>>$ 9 sodas
Two people drink 4 sodas, which means they drink $2 \times 4 = <<4*2=8>>$ 8 sodas
With one person drinking 5, that brings the total drank to $5 + 9 + 8 + 3 = <<5+9+8+3=25>>$ 25 sodas
As Tina started off with 36 sodas, that means there are $36 - 25 = <<36-25=11>>$ 11 sodas left

Final Answer: 11

Finetuning requires lots of data



- Not scalable to rely on humans to curate reasoning data
- to achieve > 80%, needs 100 times more fine-tuning data for 175B model

Scratchpad

- Scratchpad for finetuning on step by step reasoning

Input:

2 9 + 5 7

Target:

<scratch>

2 9 + 5 7 , C: 0

2 + 5 , 6 C: 1 # added 9 + 7 = 6 carry 1

, 8 6 C: 0 # added 2 + 5 + 1 = 8 carry 0

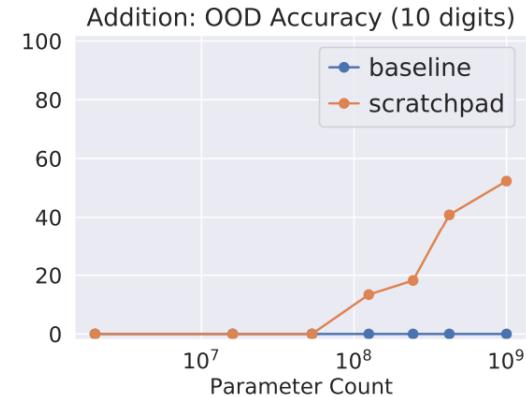
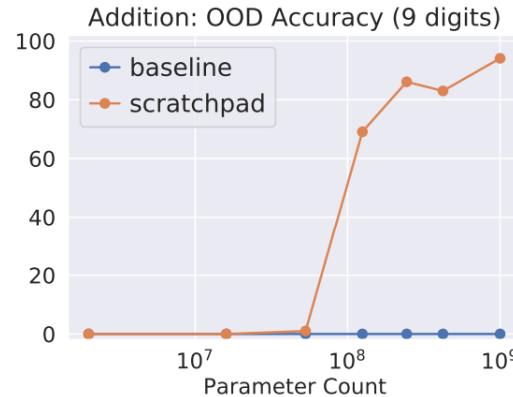
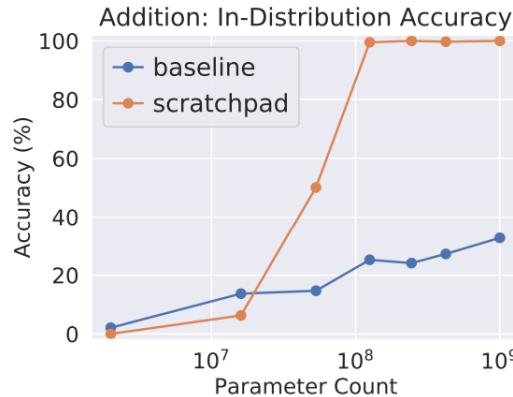
0 8 6

</scratch>

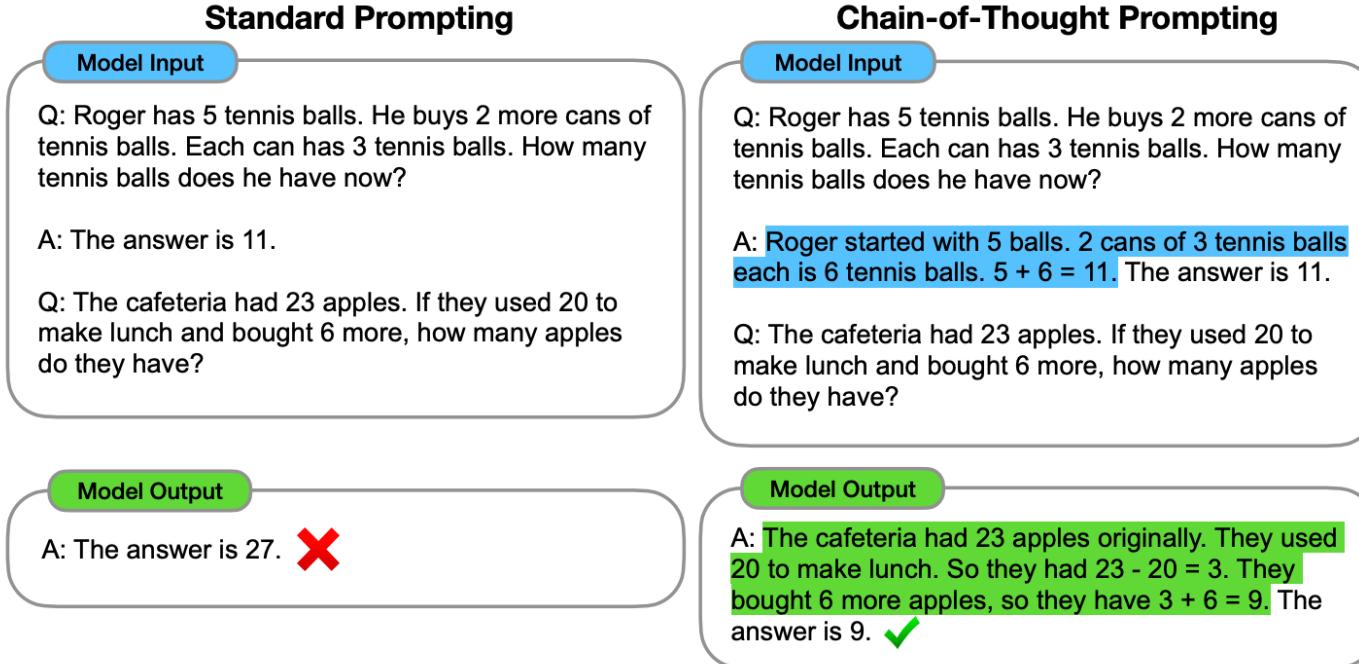
8 6

Scratchpad

- Scratchpad for finetuning on step by step reasoning
 - Improved reasoning

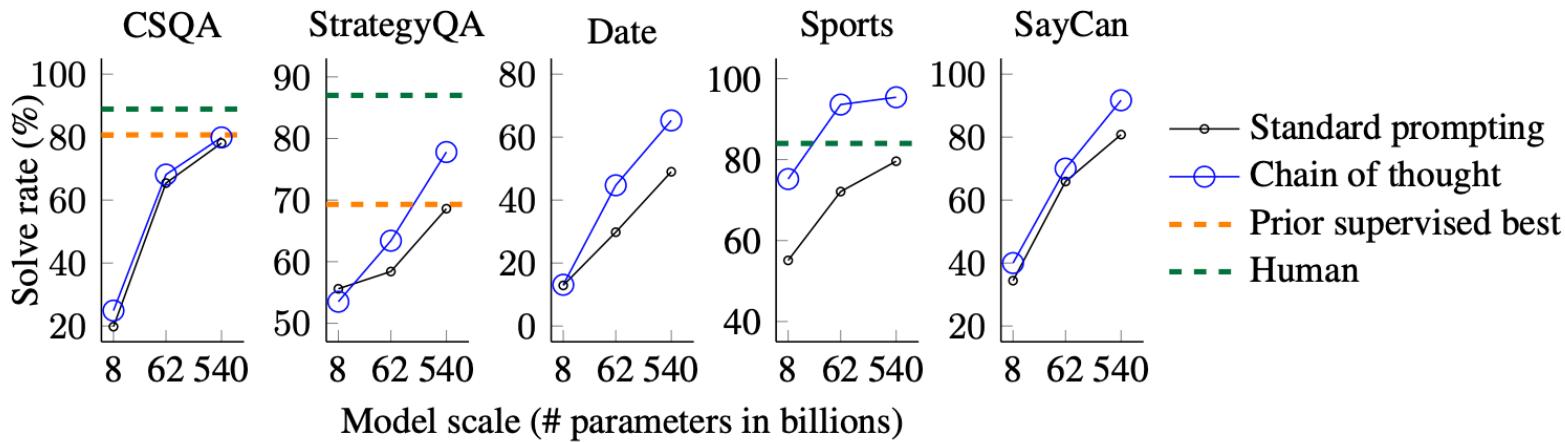


- Chain-of-thought prompting



- Chain-of-thought prompting

- Improved reasoning



Zero-shot CoT

- Chain-of-thought prompting without manual examples

(a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The answer is 8. X

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) *The juggler can juggle 16 balls. Half of the balls are golf balls. So there are $16 / 2 = 8$ golf balls. Half of the golf balls are blue. So there are $8 / 2 = 4$ blue golf balls. The answer is 4. ✓*

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: The answer (arabic numerals) is

(Output) 8 X

(d) Zero-shot-CoT (Ours)

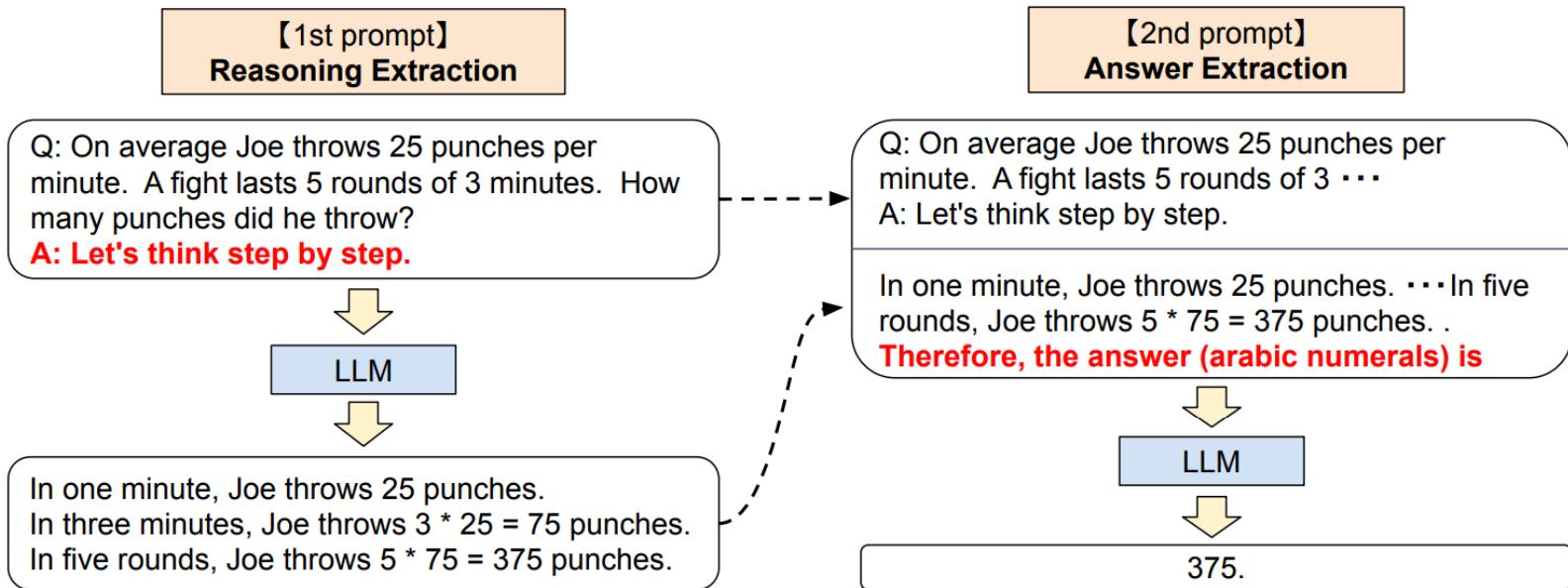
Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: **Let's think step by step.**

(Output) *There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓*

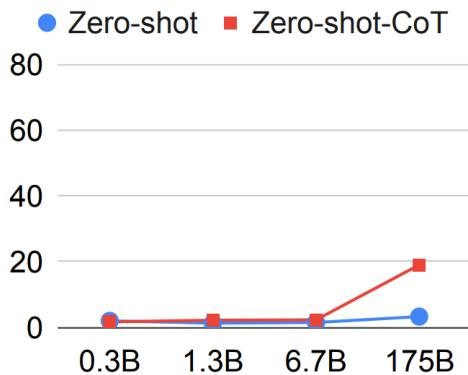
Zero-shot CoT

- Chain-of-thought prompting without manual examples

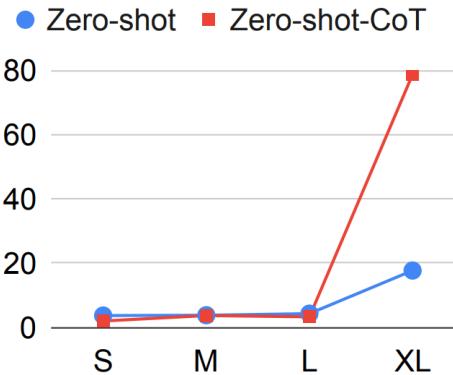


Zero-shot CoT

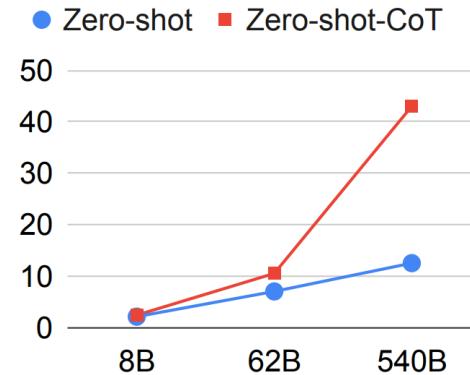
- Chain-of-thought prompting without manual examples



(a) MultiArith on Original GPT-3



(b) MultiArith on Instruct GPT-3



(c) GMS8K on PaLM

Zero-shot CoT

■ Chain-of-thought prompting without manual examples

The screenshot shows a comparison between two AI models, Model A (claude-3-sonnet-20240229) and Model B (mistral-7b-instruct-v0.2), using a chain-of-thought prompting technique.

Model A:

- Initial prompt: "Sure, let's break it down step by step."
- Given:
 - Janet's ducks lay 16 eggs per day.
 - Janet eats 3 eggs for breakfast every morning.
 - Janet uses 4 eggs to bake muffins for her friends every day.
 - Janet sells the remaining eggs at the farmers' market for \$2 per fresh duck egg.
- Step 1: Calculate the number of eggs Janet consumes or uses for baking each day.
Number of eggs consumed or used for baking = 3 (for breakfast) + 4 (for muffins) = 7 eggs
- Step 2: Calculate the number of eggs remaining for sale at the farmers' market.
Number of eggs remaining for sale = Total eggs laid - Number of eggs consumed or used for baking
Number of eggs remaining for sale = 16 - 7 = 9 eggs
- Step 3: Calculate the amount of money Janet makes by selling the remaining eggs at the farmers' market.
Amount of money made = Number of eggs remaining for sale × Price per egg
Amount of money made = 9 eggs × \$2 per egg = \$18
- Conclusion: Therefore, Janet makes \$18 every day by selling the remaining fresh duck eggs at the farmers' market.

Model B:

- Initial prompt: "Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for \$2 per fresh duck egg. How much in dollars does she make every day at the farmers' market? Let's think step by step."
- Intermediate step: "Let's first find out how many eggs Janet has left to sell after using some for her own consumption."
 - Janet uses 3 eggs for breakfast, so she has $16 - 3 = 13$ eggs left.
 - She uses 4 eggs to make muffins, so she has $13 - 4 = 9$ eggs left.
- Final step: "Now let's calculate how much she makes by selling these eggs at the farmers' market."
 - She sells the remaining eggs for \$2 each, so her daily earnings are $9 * \$2 = \18 .

Answer: `\boxed{18}`

At the bottom, there are four buttons for user feedback:

- 👉 A is better
- 👉 B is better
- 👉 Tie
- 👉 Both are bad

Process feedback

Let's verify step by step

The denominator of a fraction is 7 less than 3 times the numerator. If the fraction is equivalent to $\frac{2}{5}$, what is the numerator of the fraction? (Answer: 14)

   Let's call the numerator x .

   So the denominator is $3x-7$.

   We know that $x/(3x-7) = \frac{2}{5}$.

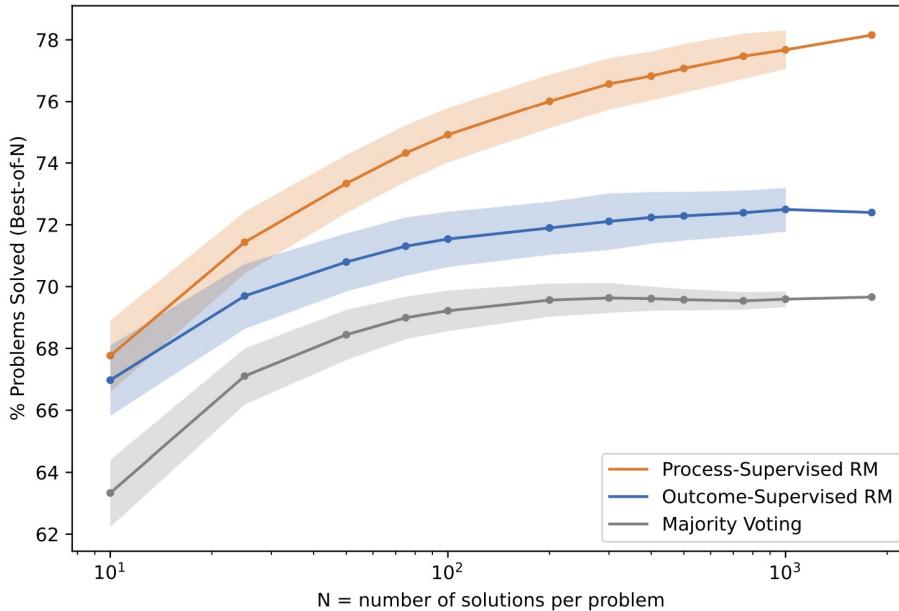
   So $5x = 2(3x-7)$.

   $5x = 6x - 14$.

   So $x = 7$.

Process feedback

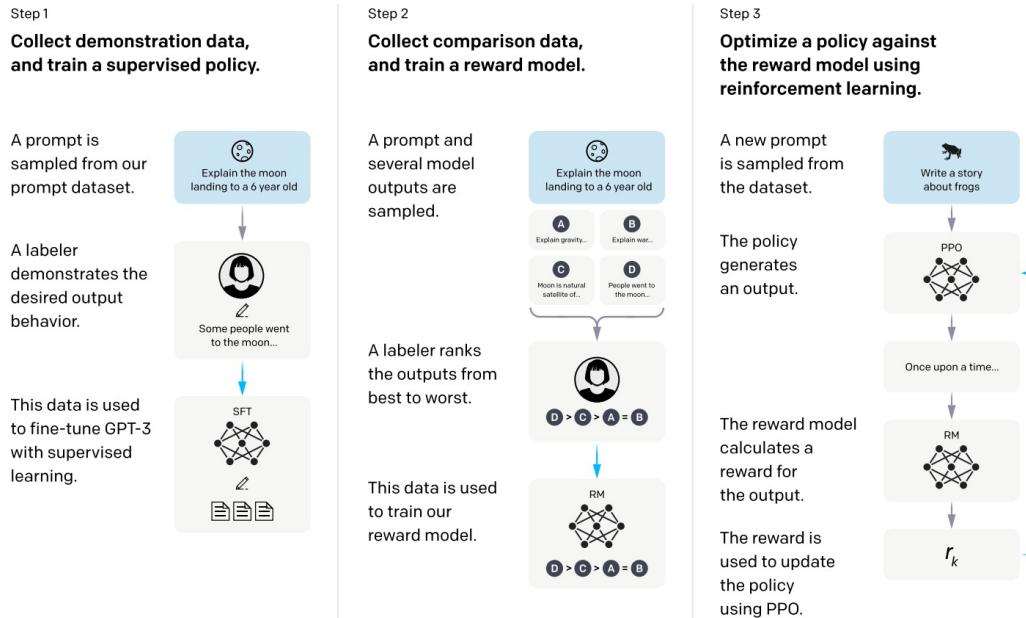
	ORM	PRM	Majority Voting
% Solved (Best-of-1860)	72.4	78.2	69.6



- Process supervision does not saturate early
- However, process supervision is very expensive

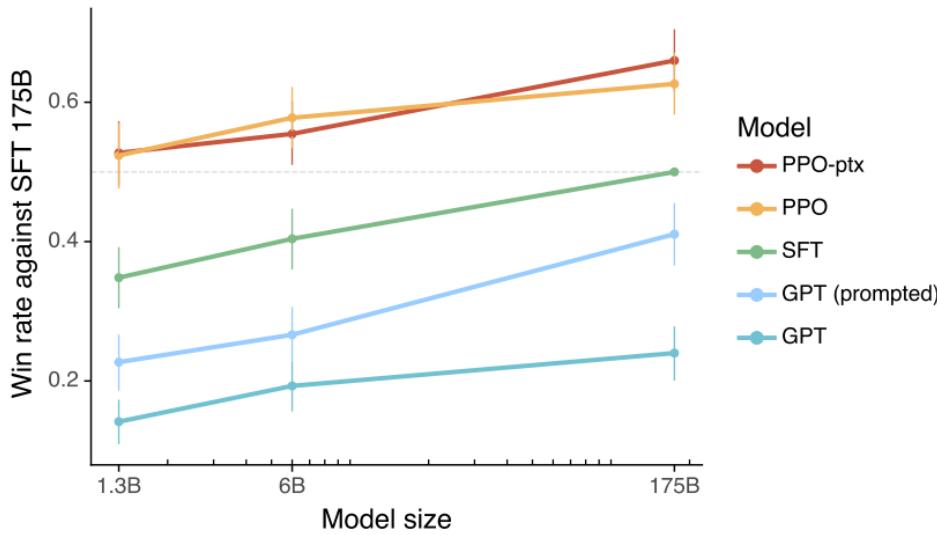
RLHF

- Collecting demonstration for supervised finetuning
- Train a reward model for reinforcement learning



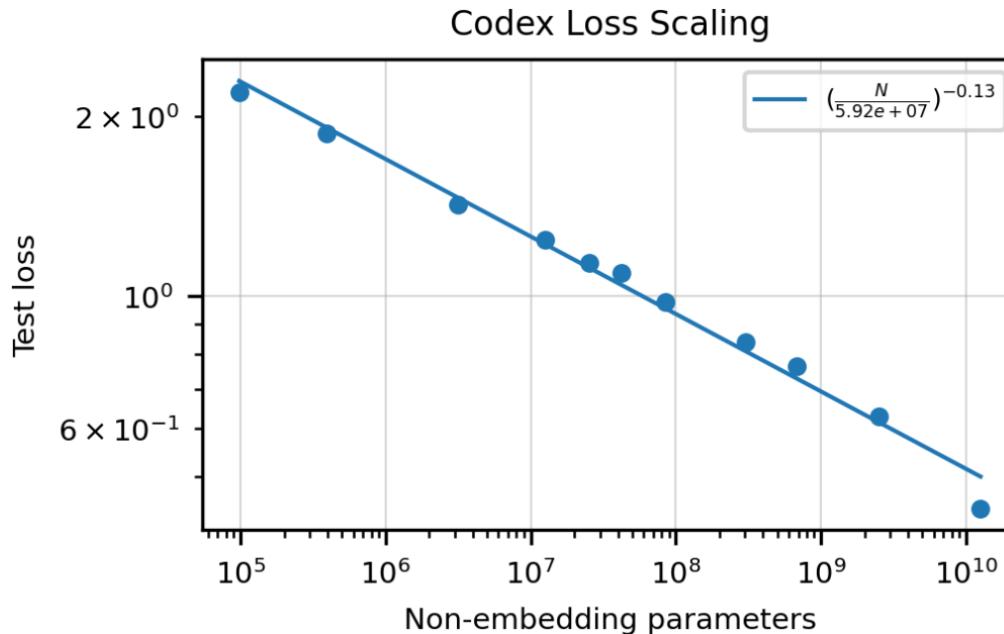
RLHF

- RLHF outperforms SFT and scales well to larger model size



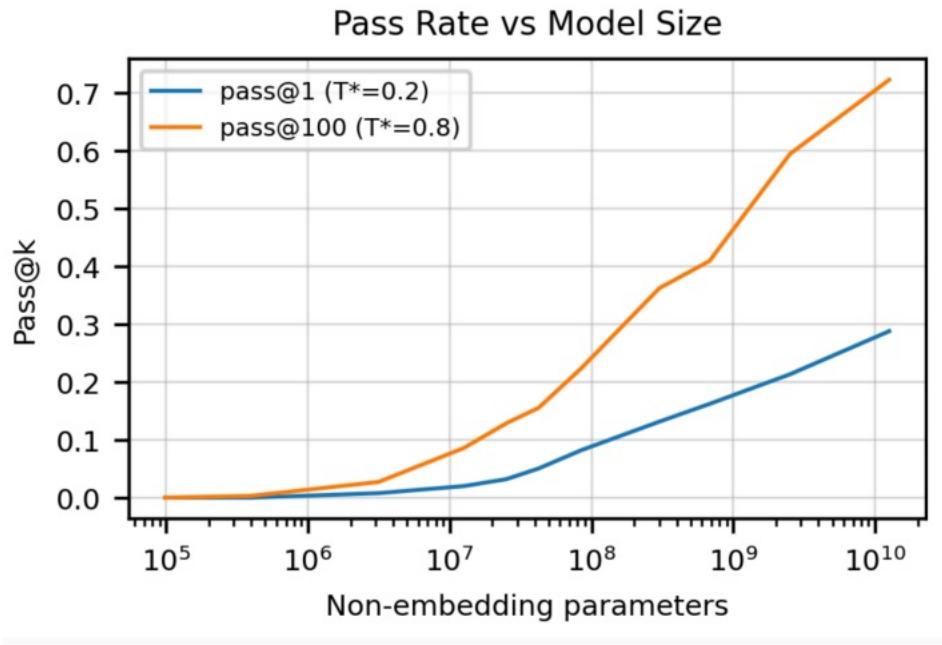
Code generation loss scaling

- Scaling works for code synthesis too
- Test loss shows power-law w.r.t model parameters



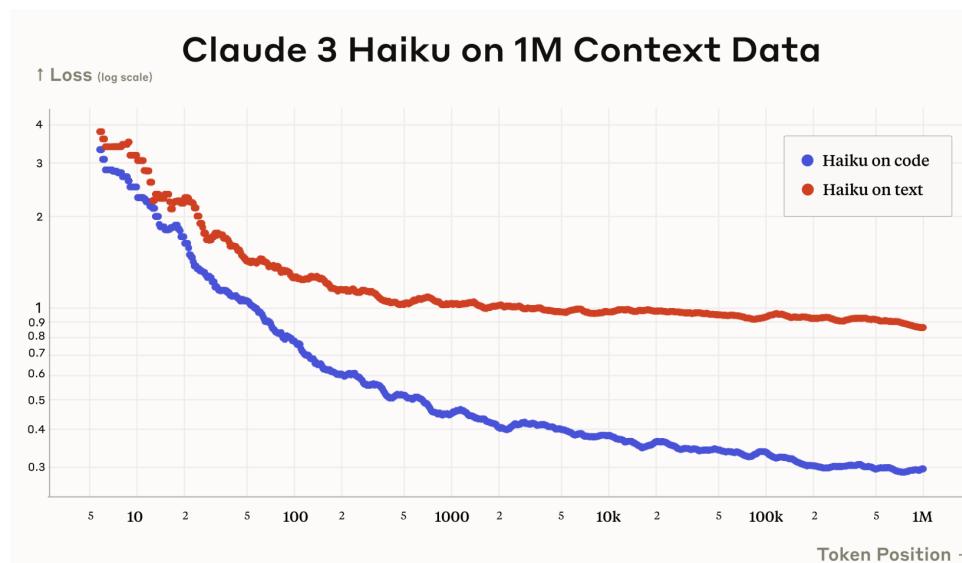
Code generation accuracy scaling

- Pass@1 and pass@100 both increase with larger model



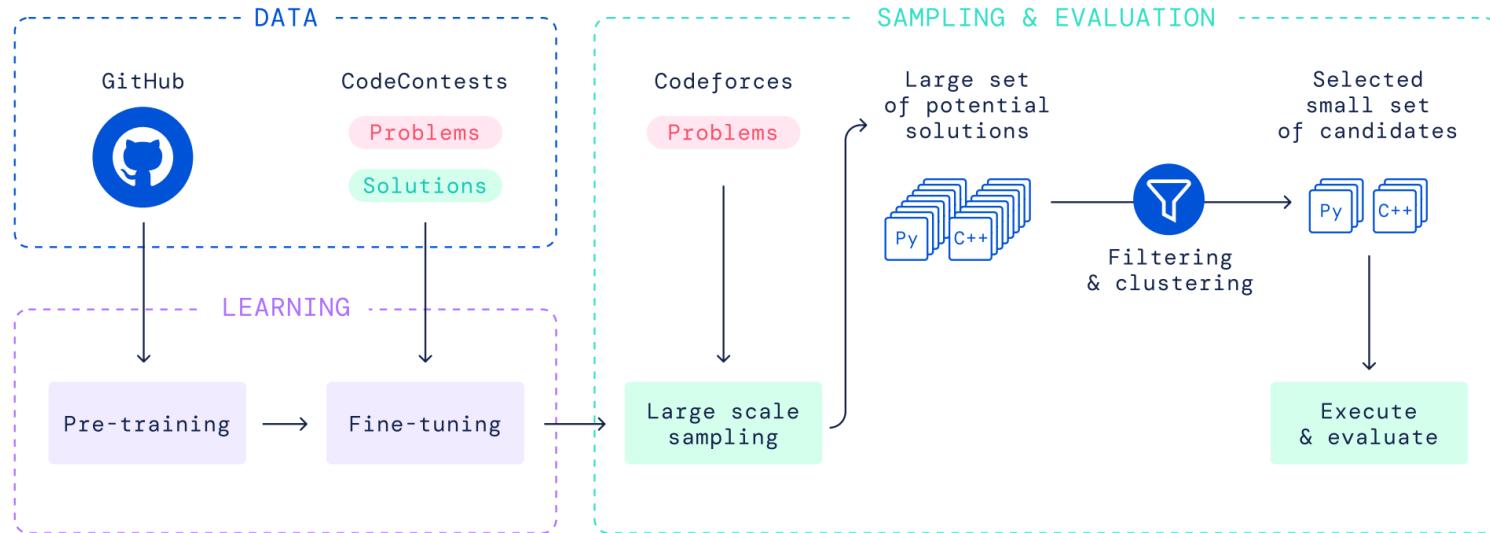
Large context code loss

- Power-law on the context size axis too
- Loss on code are generally lower compared with on text



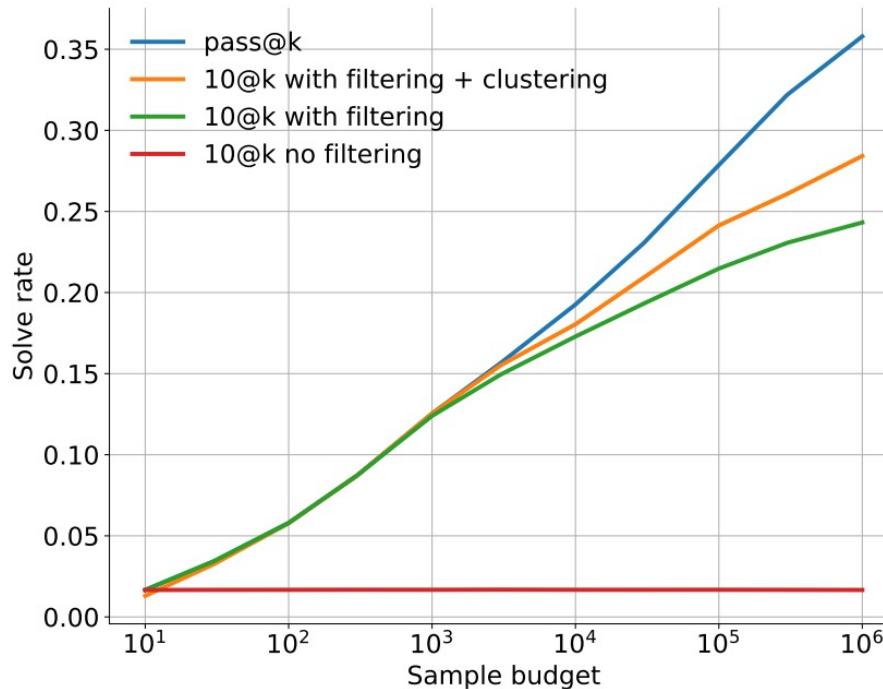
AlphaCode

- Inference optimization by large scale sampling and filtering



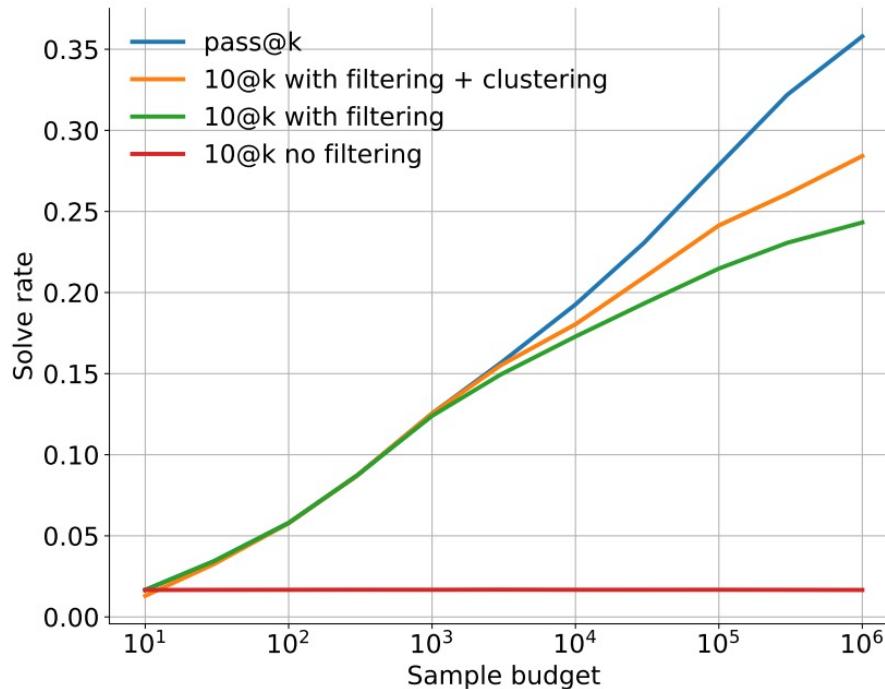
AlphaCode

- Inference optimization by large scale sampling and filtering



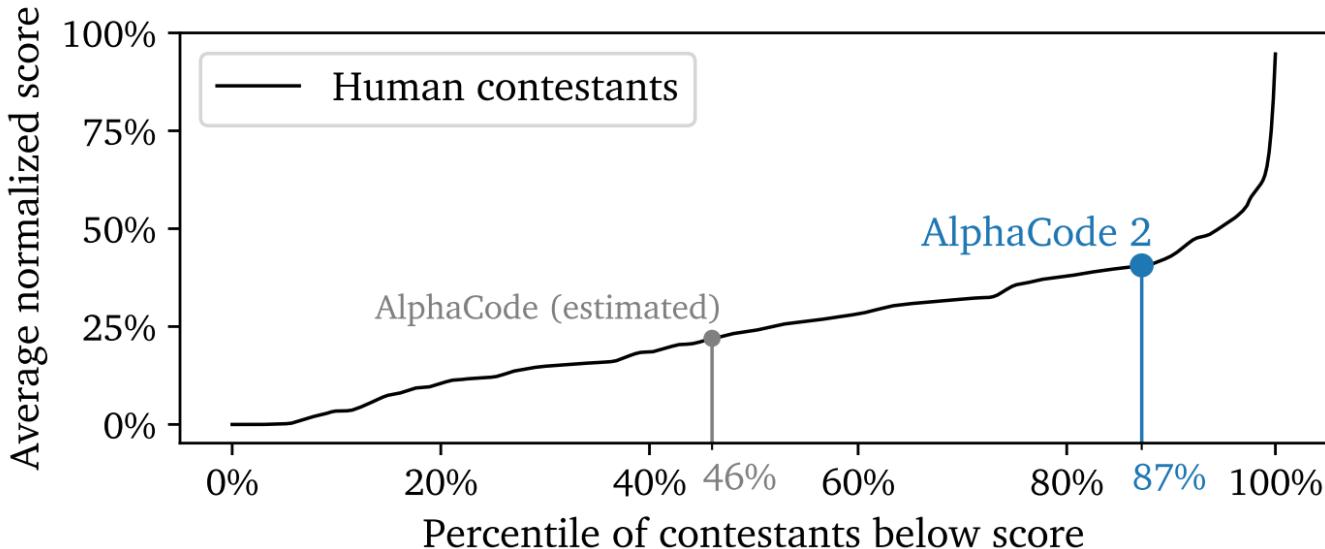
AlphaCode

- Inference optimization by large scale sampling and filtering



AlphaCode2

- AlphaCode2 replaces base model with Gemini pro
- A good pretrained model plus inference time sampling



Pretraining data

- Most data is from common crawl

Component	Raw Size	Weight	Epochs	Effective Size	Mean Document Size
Pile-CC	227.12 GiB	18.11%	1.0	227.12 GiB	4.33 KiB
PubMed Central	90.27 GiB	14.40%	2.0	180.55 GiB	30.55 KiB
Books ^{3†}	100.96 GiB	12.07%	1.5	151.44 GiB	538.36 KiB
OpenWebText2	62.77 GiB	10.01%	2.0	125.54 GiB	3.85 KiB
ArXiv	56.21 GiB	8.96%	2.0	112.42 GiB	46.61 KiB
Github	95.16 GiB	7.59%	1.0	95.16 GiB	5.25 KiB
FreeLaw	51.15 GiB	6.12%	1.5	76.73 GiB	15.06 KiB
Stack Exchange	32.20 GiB	5.13%	2.0	64.39 GiB	2.16 KiB
USPTO Backgrounds	22.90 GiB	3.65%	2.0	45.81 GiB	4.08 KiB
PubMed Abstracts	19.26 GiB	3.07%	2.0	38.53 GiB	1.30 KiB
Gutenberg (PG-19) [†]	10.88 GiB	2.17%	2.5	27.19 GiB	398.73 KiB
OpenSubtitles [†]	12.98 GiB	1.55%	1.5	19.47 GiB	30.48 KiB
Wikipedia (en) [†]	6.38 GiB	1.53%	3.0	19.13 GiB	1.11 KiB
DM Mathematics [†]	7.75 GiB	1.24%	2.0	15.49 GiB	8.00 KiB
Ubuntu IRC	5.52 GiB	0.88%	2.0	11.03 GiB	545.48 KiB
BookCorpus2	6.30 GiB	0.75%	1.5	9.45 GiB	369.87 KiB
EuroParl [†]	4.59 GiB	0.73%	2.0	9.17 GiB	68.87 KiB
HackerNews	3.90 GiB	0.62%	2.0	7.80 GiB	4.92 KiB
YoutubeSubtitles	3.73 GiB	0.60%	2.0	7.47 GiB	22.55 KiB
PhilPapers	2.38 GiB	0.38%	2.0	4.76 GiB	73.37 KiB
NIH ExPorter	1.89 GiB	0.30%	2.0	3.79 GiB	2.11 KiB
Enron Emails [†]	0.88 GiB	0.14%	2.0	1.76 GiB	1.78 KiB
The Pile	825.18 GiB			1254.20 GiB	5.91 KiB

Filtering data

- Llama's dataset is calibrated toward Wikipedia
- Outperforming GPT3 and other models

Dataset	Sampling prop.	Epochs	Disk size
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
Github	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
StackExchange	2.0%	1.03	78 GB

		0-shot	1-shot	5-shot	64-shot
GPT-3	175B	14.6	23.0	-	29.9
Gopher	280B	10.1	-	24.5	28.2
Chinchilla	70B	16.6	-	31.5	35.5
PaLM	8B	8.4	10.6	-	14.6
	62B	18.1	26.5	-	27.6
	540B	21.2	29.3	-	39.6
LLaMA	7B	16.8	18.7	22.0	26.1
	13B	20.1	23.4	28.1	31.9
	33B	24.9	28.3	32.9	36.0
	65B	23.8	31.0	35.0	39.9

Table 4: **NaturalQuestions**. Exact match performance.

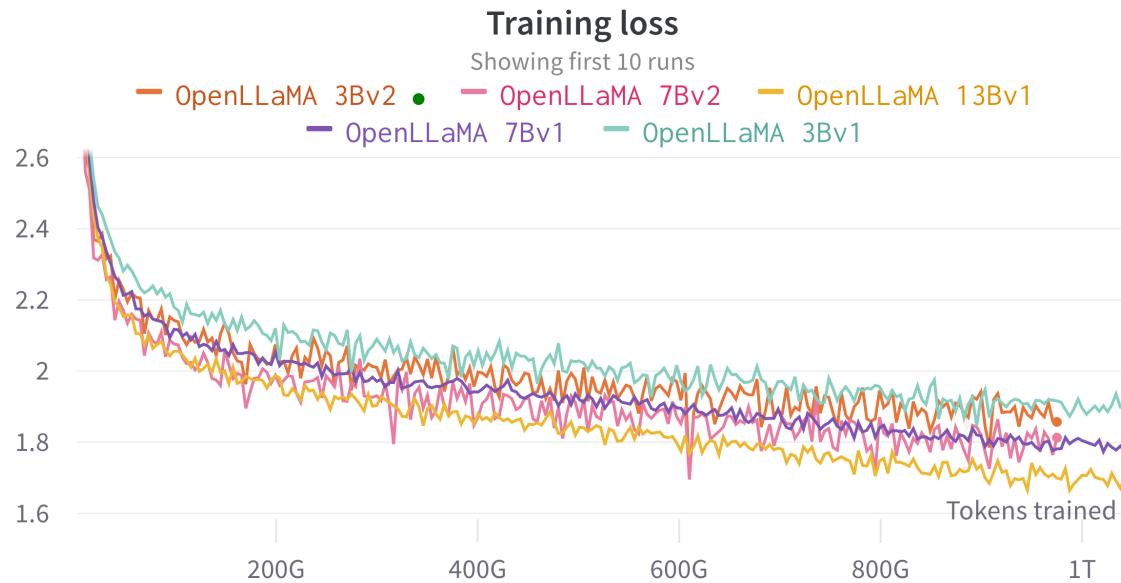
RedPajam

- RedPajam: an open-source dataset

	RedPajama	LLaMA*
CommonCrawl	878 billion	852 billion
C4	175 billion	190 billion
Github	59 billion	100 billion
Books	26 billion	25 billion
ArXiv	28 billion	33 billion
Wikipedia	24 billion	25 billion
StackExchange	20 billion	27 billion
Total	1.2 trillion	1.25 trillion

OpenLLaMA

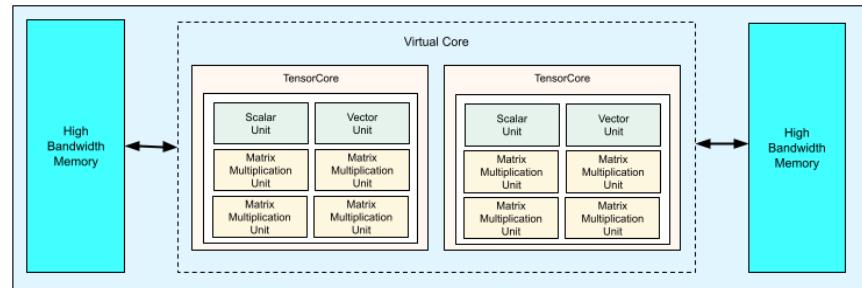
- OpenLLaMA matches LLaMA performance



TPU / GPU

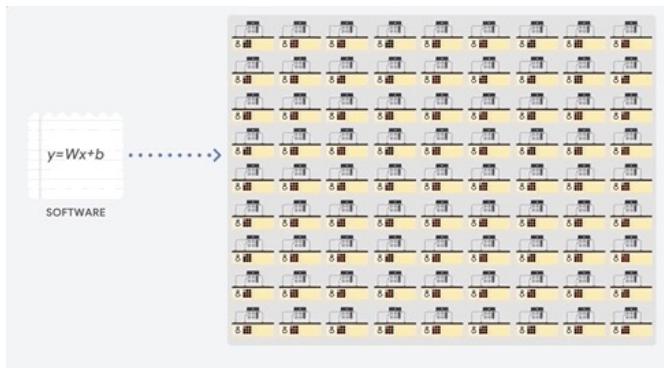
- Flops, HBM size and bandwidth, Interconnect

Key specifications	v4 Pod values
Peak compute per chip	275 teraflops (bf16 or int8)
HBM2 capacity and bandwidth	32 GB, 1200 GBps
Measured min/mean/max power	90/170/192 W
TPU Pod size	4096 chips
Interconnect topology	3D mesh
Peak compute per Pod	1.1 exaflops (bf16 or int8)
All-reduce bandwidth per Pod	1.1 PB/s
Bisection bandwidth per Pod	24 TB/s

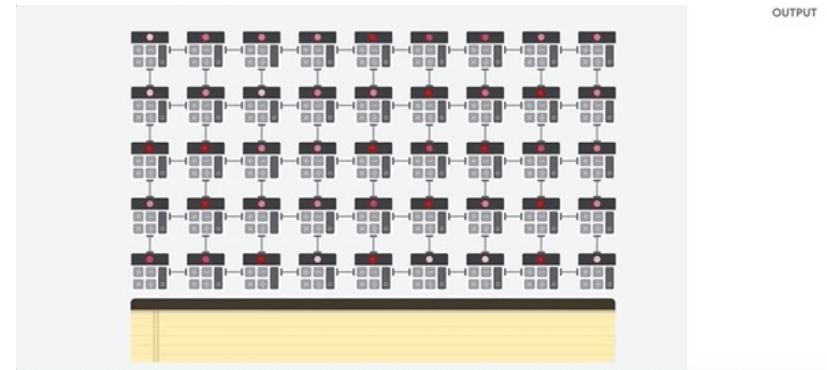


GEMM

- High-computational throughput on neural network calculations
- GPU compute unit is smaller but more, TPU has large compute units



GPU, many 8x4x8 ALU



TPU, systolic array 8x128x128

Matmul sharding

- Simple case: sharded matmul $Y = XA$
- Row sharding on A, column sharding on X



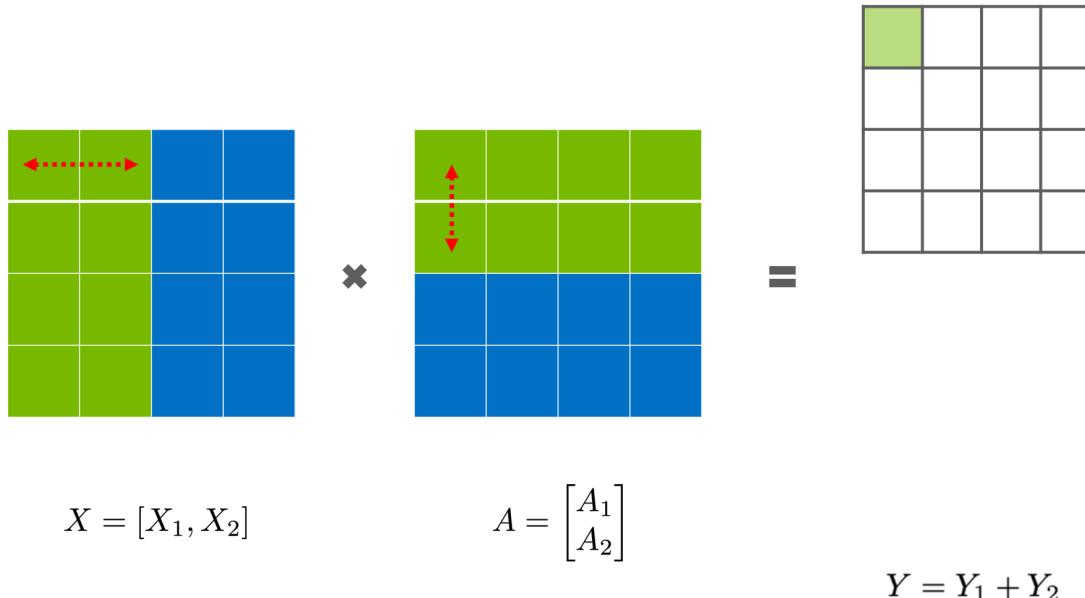
$$X = [X_1, X_2]$$

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$$

$$Y = Y_1 + Y_2$$

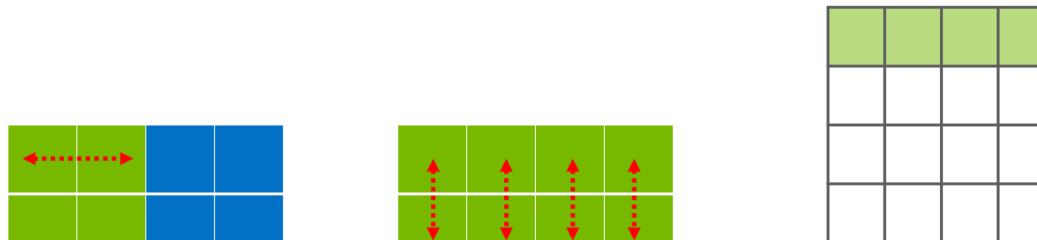
Matmul sharding

- Row sharding on A, column sharding on X



Matmul sharding

- Row sharding on A, column sharding on X



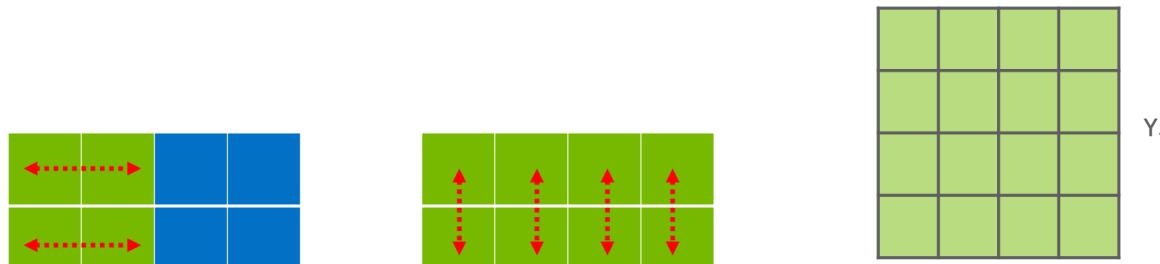
$$X = [X_1, X_2]$$

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$$

$$Y = Y_1 + Y_2$$

Matmul sharding

- Row sharding on A, column sharding on X



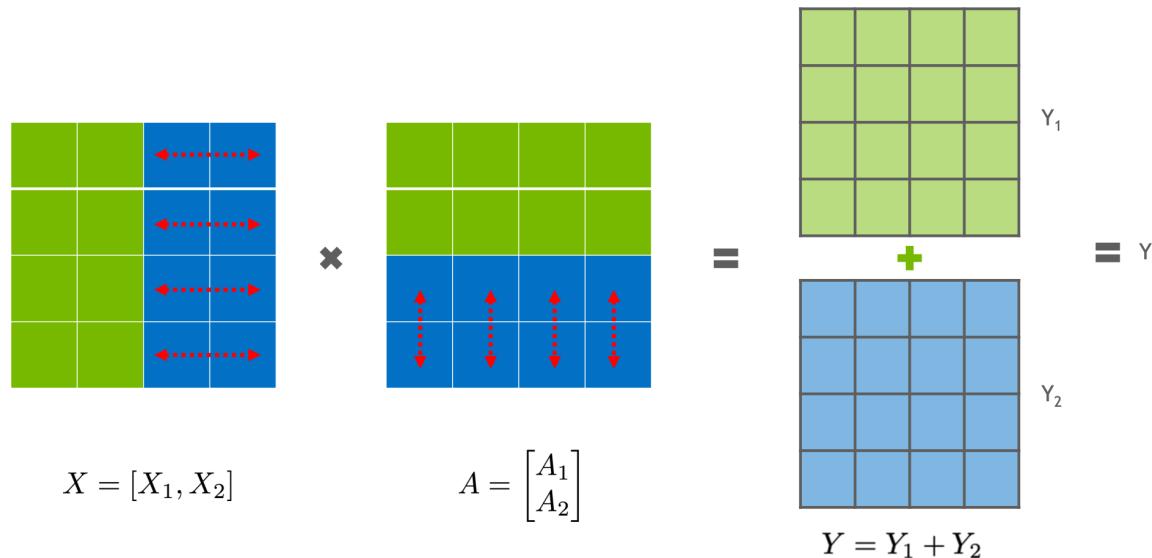
$$X = [X_1, X_2]$$

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$$

$$Y = Y_1 + Y_2$$

Matmul sharding

- Row sharding on A, column sharding on X



Matmul sharding

- Column sharding on A
- Replicate X

$$\begin{array}{c} \text{X} \\ \times \\ \text{A} = [A_1, A_2] \\ = \\ \left[\begin{array}{c|c} Y_1 & Y_2 \end{array} \right] = Y \end{array}$$

The diagram illustrates the matrix multiplication process. On the left, matrix X is shown as a 4x4 grid of green squares. Red dotted arrows indicate column sharding: each column contains a horizontal sequence of four red arrows pointing from left to right. In the middle, matrix A is shown as a 4x4 grid with the first two columns colored green and the last two columns colored blue. Red dotted arrows point from the first two columns of A to the corresponding columns of X . To the right of the multiplication symbol, the equation $A = [A_1, A_2]$ is given, where A_1 and A_2 represent the 4x2 submatrices formed by the first and second columns of A respectively. The result of the multiplication is shown as a bracketed expression $[Y_1, Y_2]$, where Y_1 and Y_2 are 4x2 matrices. The bracket is labeled with an equals sign, indicating that the resulting matrix Y is composed of these two submatrices.

Matmul sharding

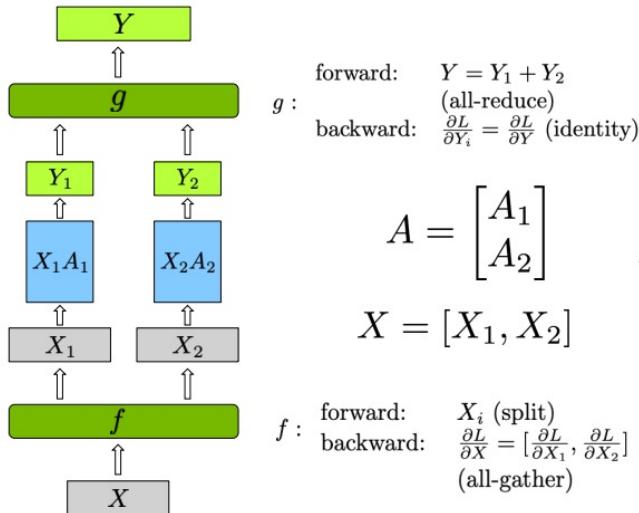
- Column sharding on A
- Replicate X

$$\begin{matrix} \text{X} & \times & \begin{matrix} A = [A_1, A_2] \\ A_1 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, A_2 = \begin{bmatrix} 10 & 11 & 12 \\ 13 & 14 & 15 \\ 16 & 17 & 18 \end{bmatrix} \end{matrix} & = & \begin{matrix} Y = [Y_1, Y_2] \\ Y_1 = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, Y_2 = \begin{bmatrix} 10 & 11 & 12 \\ 13 & 14 & 15 \\ 16 & 17 & 18 \end{bmatrix} \end{matrix} \end{matrix}$$

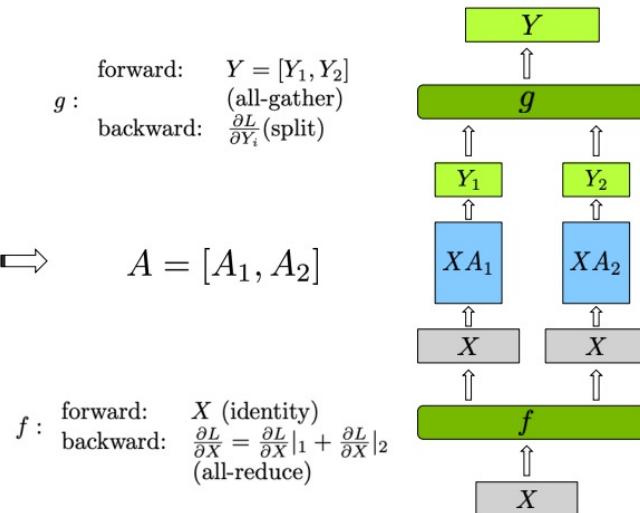
The diagram illustrates the computation of a matrix multiplication $X \times A = Y$. Matrix X is a 4x3 matrix of green blocks. Matrix A is a 3x3 matrix split into two columns: A_1 (green) and A_2 (blue). The result Y is a 4x3 matrix split into two columns: Y_1 (green) and Y_2 (blue). Red arrows indicate the data flow: horizontal arrows within X and A_1 indicate column sharding, while vertical arrows between A_1 and Y_1 , and between A_2 and Y_2 indicate the replication of X across columns of A .

Matmul sharding

Row Parallel Linear Layer



Column Parallel Linear Layer

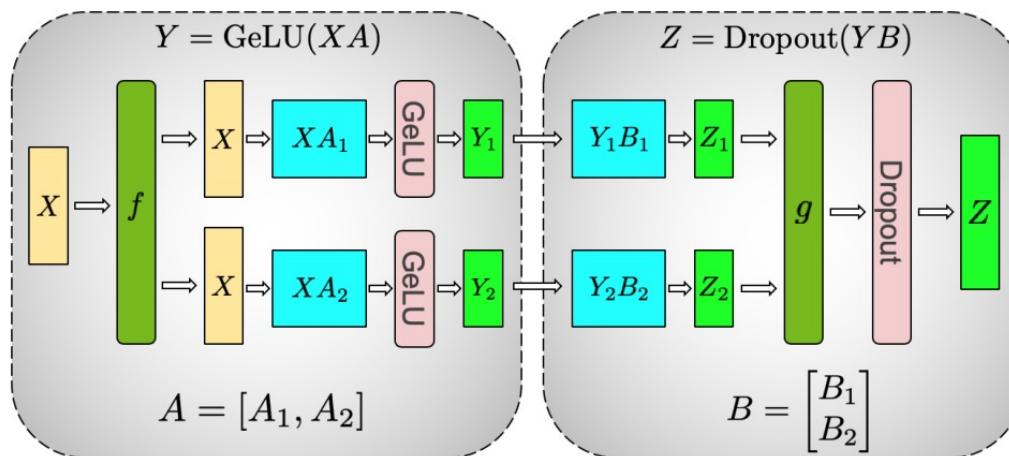


MLP sharding

- $Y = \text{GeLU}(XA)B$
- Approach 1: split X column-wise and A row-wise
 - GeLU of sums != sum of GeLUs $X = [X_1, X_2]$ $A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$ $Y = \text{GeLU}(X_1A_1 + X_2A_2)$
 - Requires synchronization before GeLU
- Approach 2: split A column-wise
 - No sharding on contraction dimension of X
 - $$A = [A_1, A_2] \quad [Y_1, Y_2] = [\text{GeLU}(XA_1), \text{GeLU}(XA_2)]$$
 - No synchronization necessary

MLP sharding

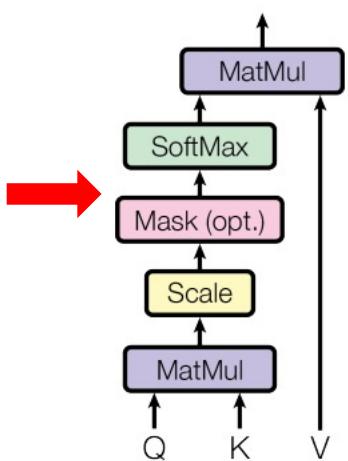
- **f** and **g** are conjugate, **f** is **identity** operator in the forward pass and **all-reduce** in the backward pass while **g** is **all-reduce** in forward and **identity** in backward.



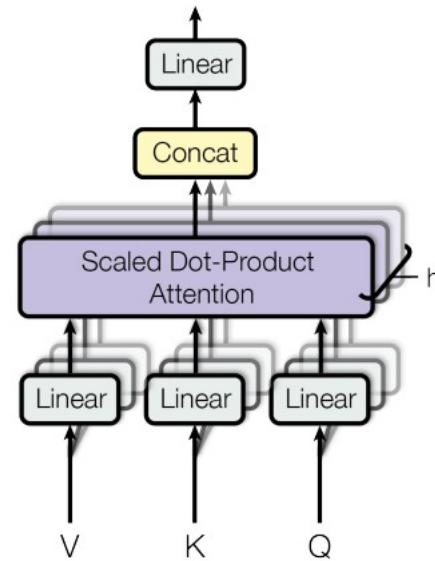
Attention sharding

- Nonlinearity in attention, so column sharding.

Scaled Dot-Product Attention

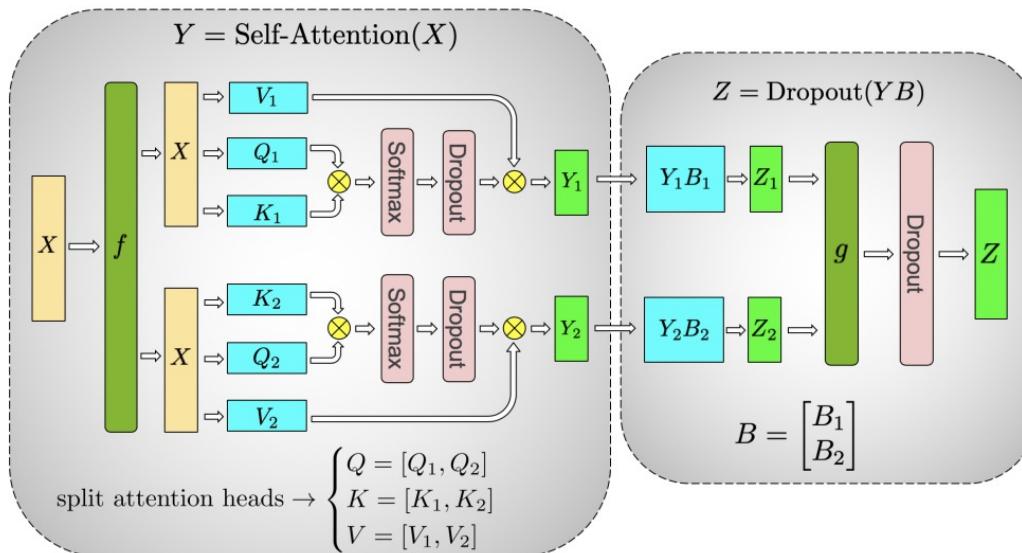


Multi-Head Attention



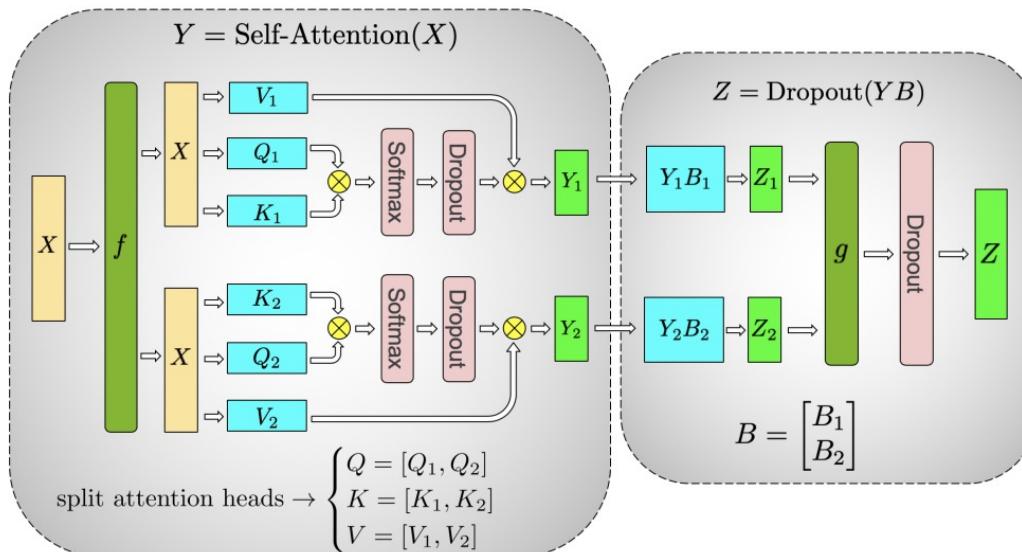
Attention sharding

- **f** and **g** are conjugate, **f** is **identity** operator in the forward pass and **all-reduce** in the backward pass while **g** is **all-reduce** in forward and **identity** in backward.



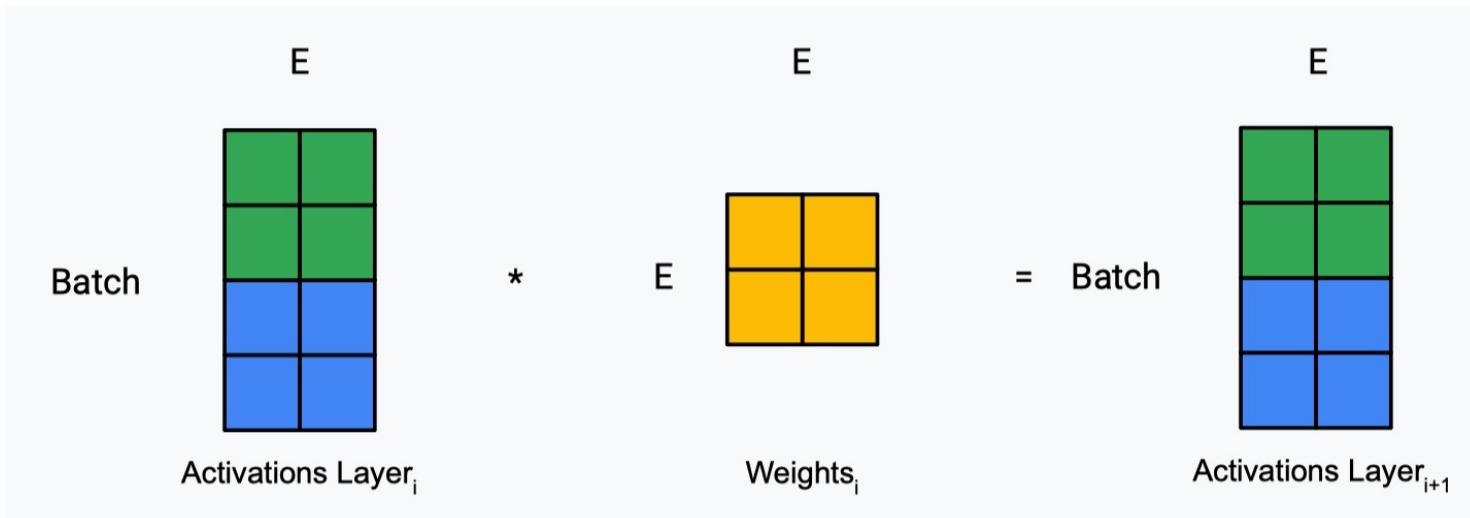
Attention sharding

- **f** and **g** are conjugate, **f** is **identity** operator in the forward pass and **all-reduce** in the backward pass while **g** is **all-reduce** in forward and **identity** in backward.



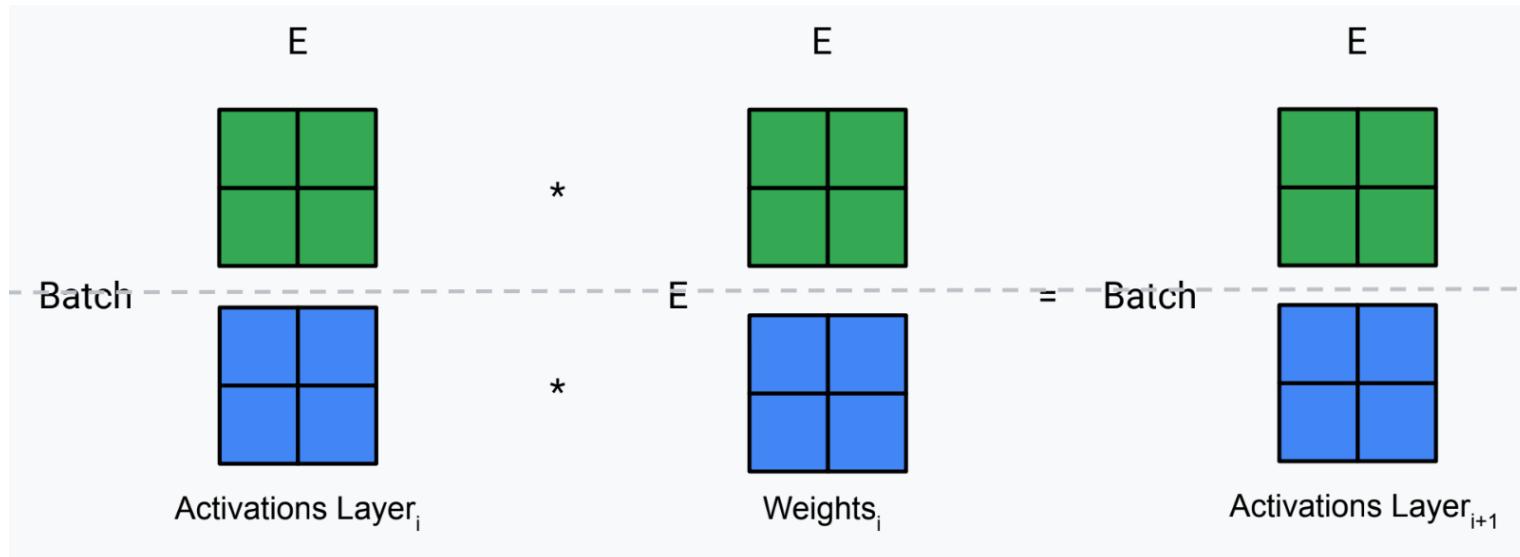
Data parallelism

- Activations need to be sharded – they're much bigger than weights. We can either shard them by rows or columns.
- It keeps working with a linear speedup as long as you scale batch linearly with number of TPUs



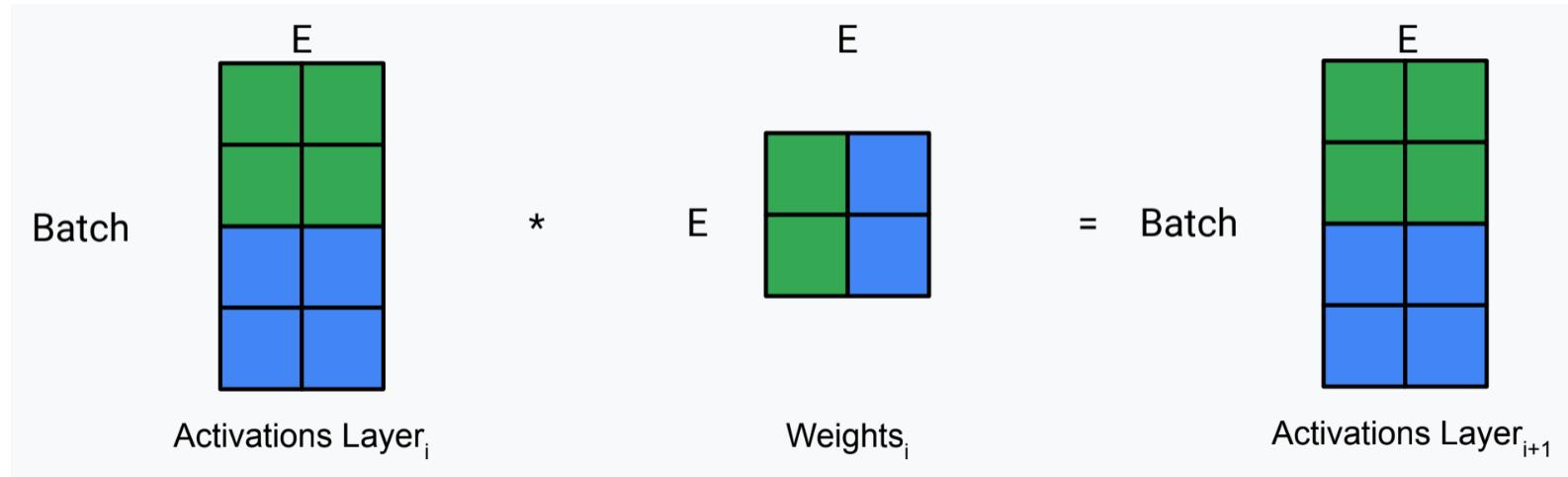
Data parallelism

- At 4 bytes per parameter we need 700GB for 175B GPT3
- But A100 has only 80GB.



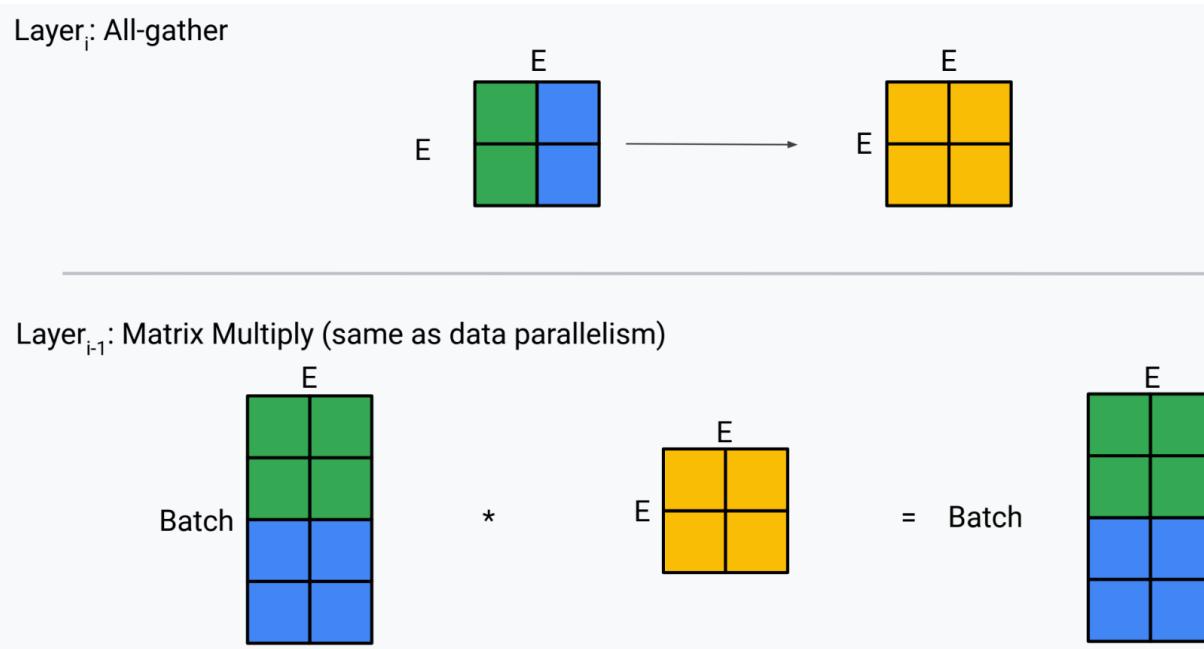
Fully sharded data parallelism

- All-gathering the next layer while doing the arithmetic on this layer
- Requires storing only 1 layer at a time (next slide)



Fully sharded data parallelism

- All-gathering the next layer while doing the arithmetic on this layer
- Requires storing only 1 layer at a time

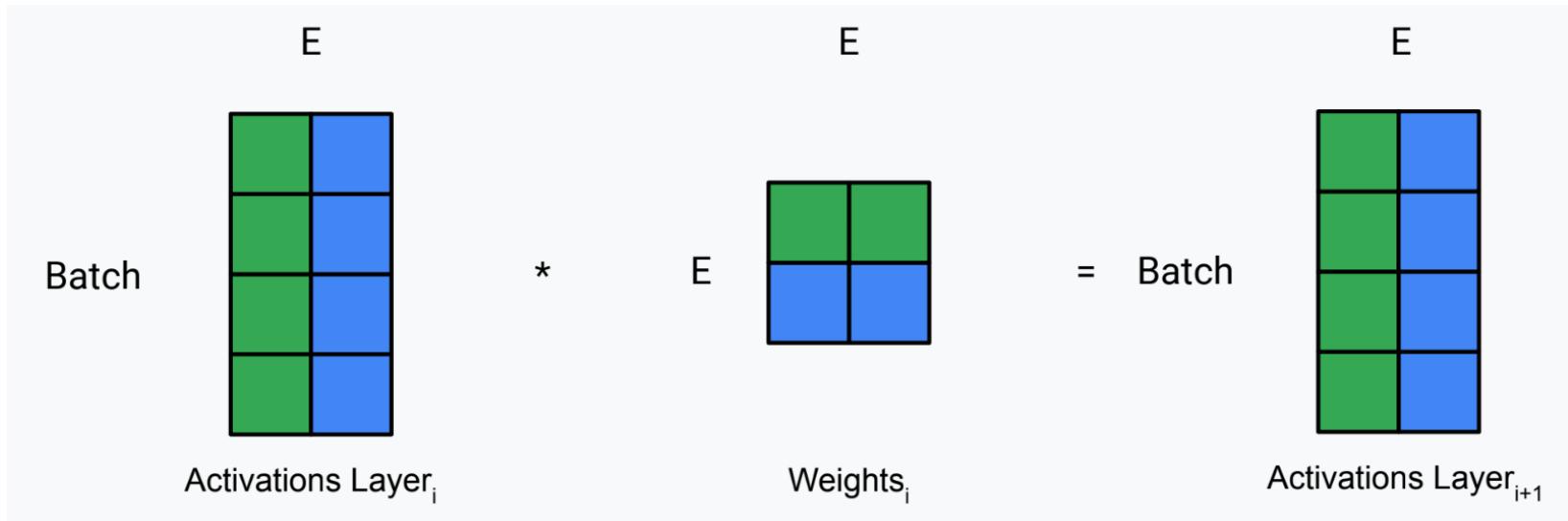


Fully sharded data parallelism

- The matrix multiply takes: FLOPs = $2 * \text{BATCH_PER_CHIP} * E2$
- The all gather requires reading $\sim 2 * E2$ bytes (assuming bfloat16)
- Arithmetic intensity (FLOPs/bytes) = $2 * \text{BATCH_PER_CHIP} * E2 / (2 * E2 \text{ bytes}) = \text{BATCH_PER_CHIP FLOPs / byte.}$
- On TPU, ICI arithmetic intensity is 1018 FLOPs/byte, so we need our BATCH_PER_CHIP to be comfortably larger than 1018.

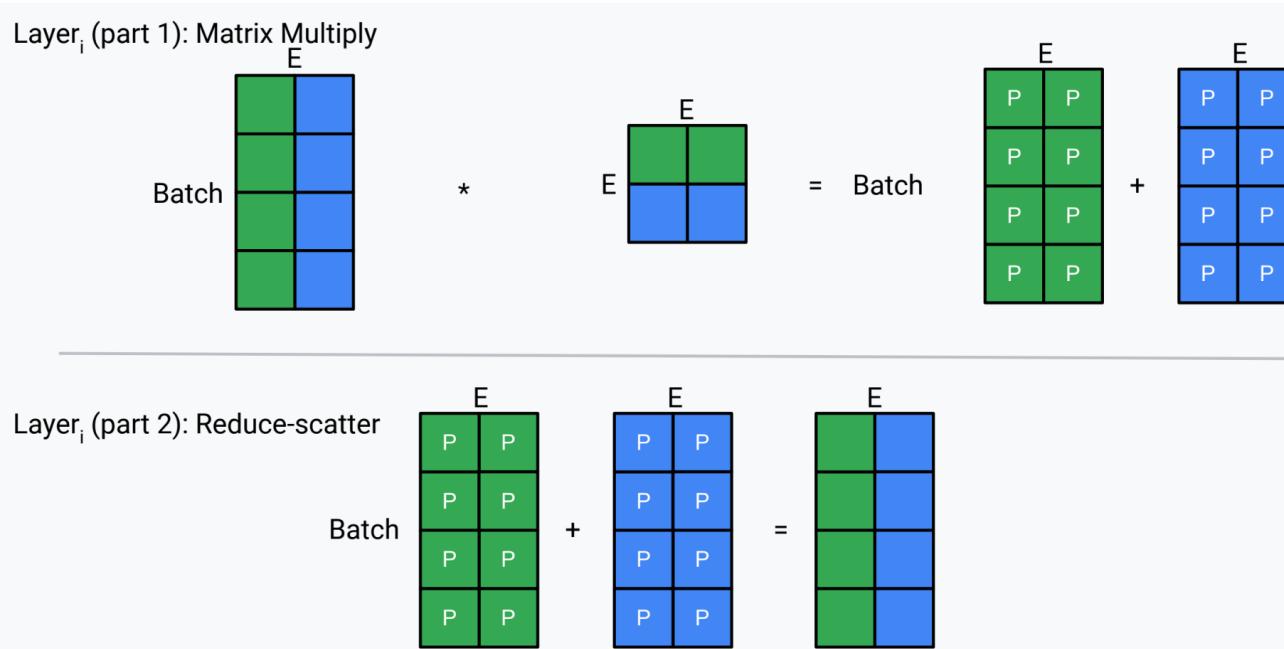
Tensor parallelism

- Shard the activation dimension



Tensor parallelism

- Shard the activation dimension

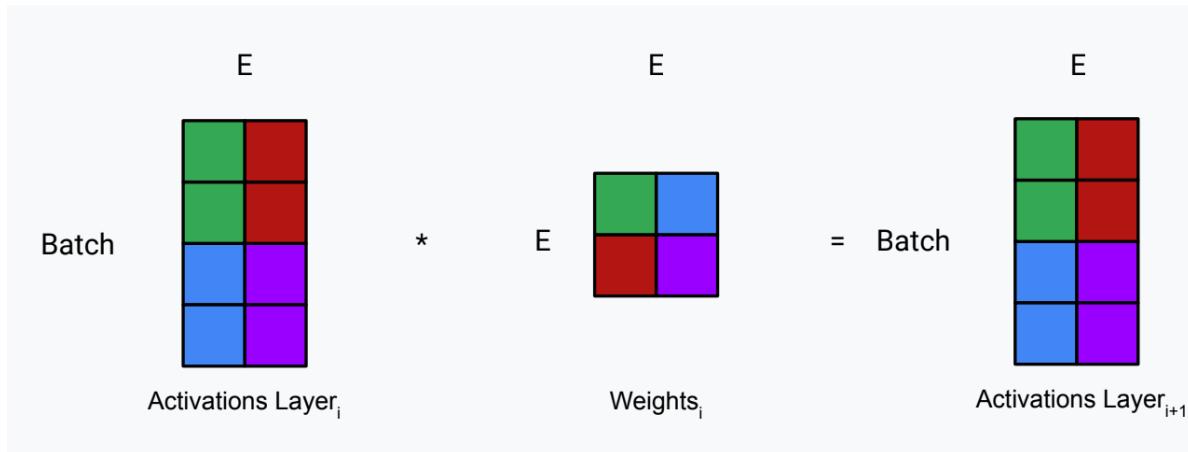


Tensor parallelism

- The matrix multiply takes (per chip): $\text{FLOPs} = 2 * \text{BATCH} * E^2 / (\text{NUM_TP_SHARDS})$
- The reduce scatter requires $\sim 2 * \text{BATCH} * E$ bytes (assuming bfloat16)
- Arithmetic intensity (FLOPs/bytes) = $E / \text{NUM_TP_SHARDS}$.
- TPU ICI arithmetic intensity is 1018 FLOPs/byte, so we need E to be comfortably larger than $1018 * \text{NUM_TP_SHARDS}$.

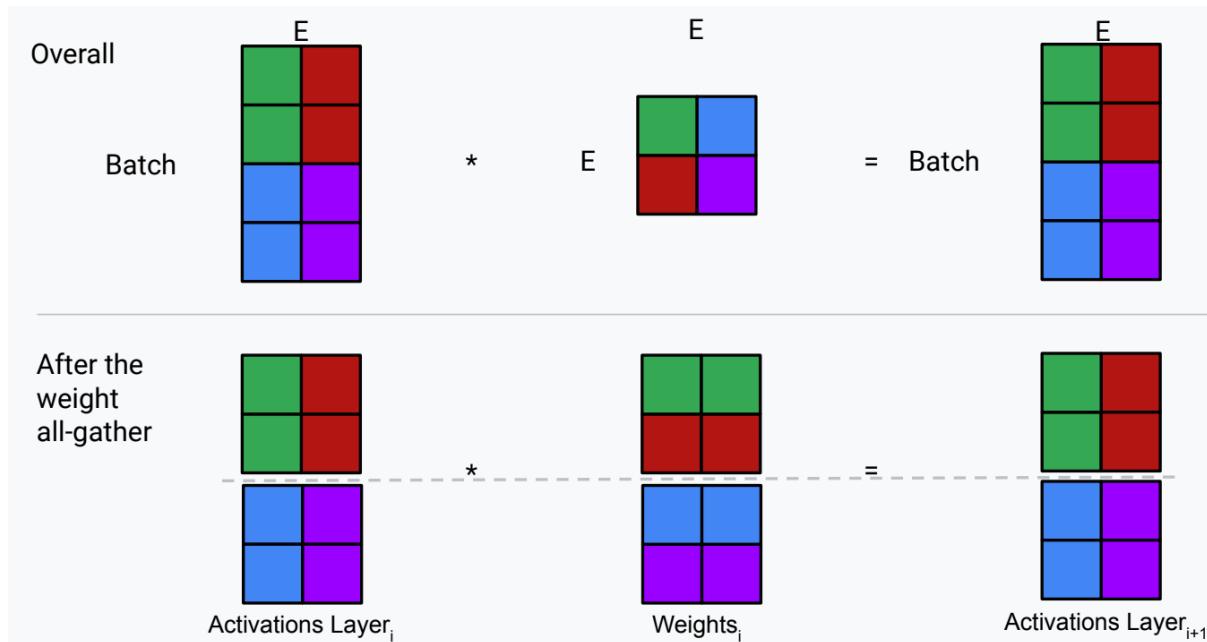
FSDP with TP

- While processing layer i , all-gather layer weights for $layer_{i+1}$ for FSDP
- Then it just becomes TP



FSDP with TP

- While processing layer i , all-gather layer weights for $layer_{i+1}$ for FSDP
- Then it just becomes TP



FSDP with TP

- Given NUM_TP_SHARDS and NUM_FSDP_SHARD
- When both:
 - $E > 2 * 1018 * \text{NUM_TP_SHARDS}$
 - $\text{NUM_TP_SHARD} * \text{BATCH_PER_CHIP} > 2 * 1018$
- 256 TPU: 16 for TP and 16 for FSDP
 - $E > \sim 32k$
 - $\text{BATCH_PER_CHIP} > 125$

Practical guideline

- Scaling model size and data size
 - Use FSDP as much as possible
 - Use TP/DP/SP to reduce batch size
- Scaling context size
 - Use Blockwise Transformers to reduce memory cost
 - Use RingAttention to increase context arbitrarily

Open problems

- Large language model → large world model
 - Predict world realistically, including text, vision, action.
 - Video conversation
- Consistent reasoning
 - With million-length “tape” but models stop step-by-step thinking after one hundred steps.
- Training and inference efficiency
 - Exponentially more efficient architecture or training paradigm