Long-Context Language Models

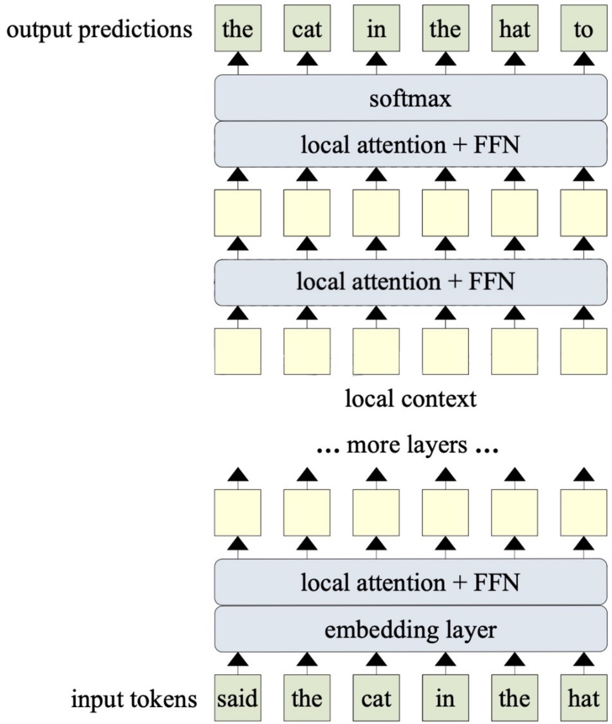
Group members: Christine Li, Qilin Chen, Yirong Xu

Washington University in St. Louis

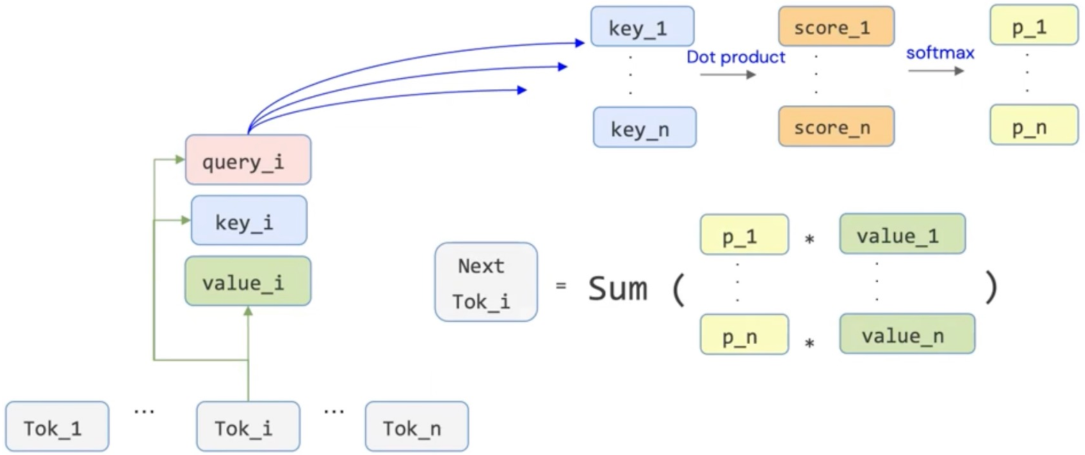
Outlines

- Review: Transformer Architectures
- Papers
  - Memorizing Transformers (Mar, 2022)
  - LongNet: Scaling Transformers to 1,000,000,000 Token (Jul, 2023)
  - LongLoRA: Efficient Fine-tuning of Long-Context Large Language Models (Dec,2023)
  - Lost in the Middle: How Language Models Use Long Contexts (Nov, 2023)
- Q&A

Transformer Architecture

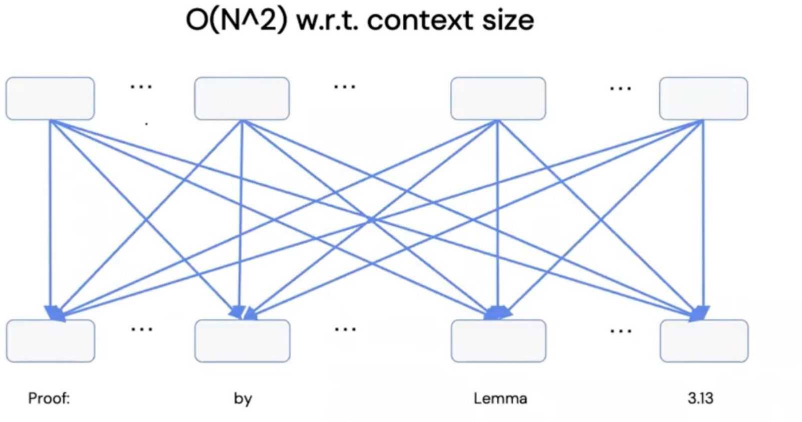


- Word Embedding
- Self-Attention Mechanism
  - Query, Key, Value
  - Dot product Query and Keys to find relevance between tokens: Attention score

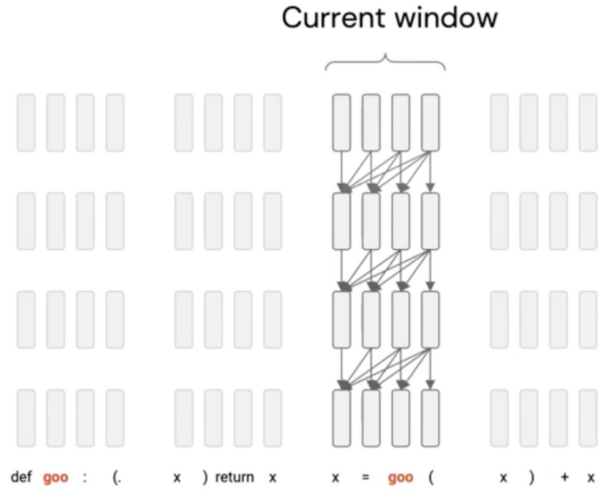


Limitation of Transformers

- Bottleneck
  Attend from **each** of the token to **every other** token



- Fixed window size due to quadratic complexity



How to achieve a larger context window?

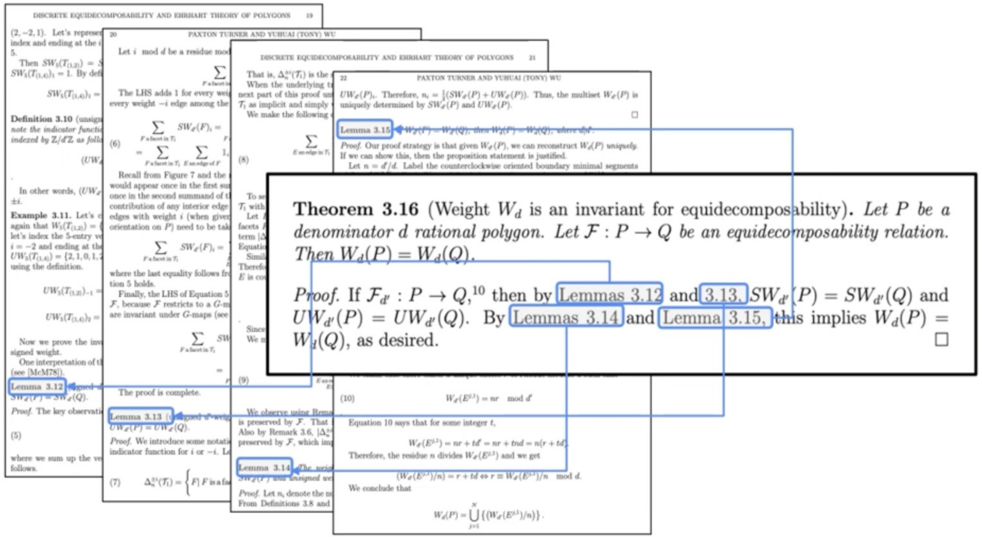
Memorizing Transformers

Yuhuai Wu, Markus N. Rabe, DeLesley Hutchins, Christian Szegedy

Reference: Wu, Y., Rabe, M. N., Hutchins, D., & Szegedy, C. (2022). Memorizing transformers. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2203.08913

Growing Knowledge Base

- Theorem database in mathematics



Theorem 3.16 (Weight W_d is an invariant for equidecomposability). *Let P be a denominator d rational polygon. Let $\mathcal{F} : P \to Q$ be an equidecomposability relation. Then $W_d(P) = W_d(Q)$.*

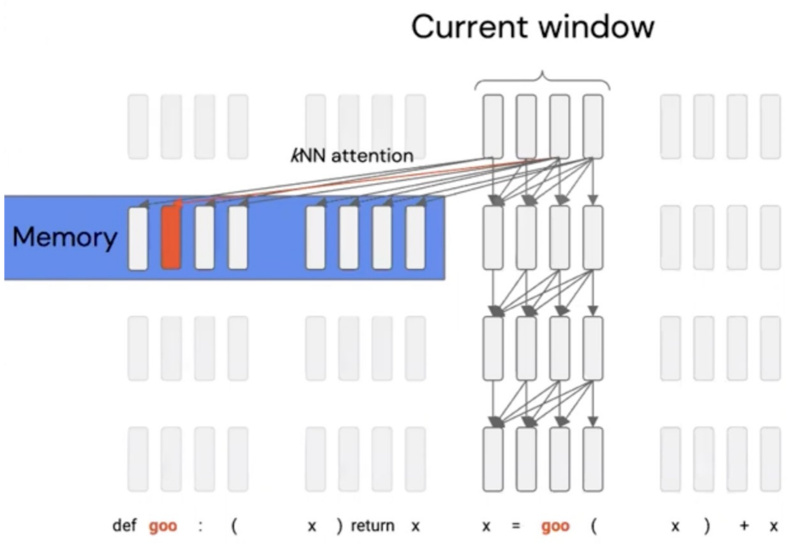
Proof. If $\mathcal{F}_{d'} : P \to Q$,[10] then by $\boxed{\text{Lemmas 3.12}}$ and $\boxed{3.13,}$ $SW_{d'}(P) = SW_{d'}(Q)$ and $UW_{d'}(P) = UW_{d'}(Q)$. By $\boxed{\text{Lemmas 3.14}}$ and $\boxed{\text{Lemma 3.15,}}$ this implies $W_d(P) = W_d(Q)$, as desired. $\square$

- Codebase in program synthesis

```
def cast_tuple(val, length = 1):
    return val if isinstance(val, tuple) else ((val,)

def l2norm(t):
    return F.normalize(t, dim = -1)

# helper classes

class PreNormResidual(nn.Module):
    def __init__(self, dim, fn):
        super().__init__()
        self.fn = fn
        self.norm = nn.LayerNorm(dim)

    def forward(self, x, **kwargs):
        out = self.fn(self.norm(x), **kwargs)

        if not isinstance(out, tuple):
            return out + x

        head, *tail = out
        return (head + x, *tail)
```

Memorizing Transformers

- Maintain an external memory
  Memorize the previously generated keys and values
- kNN Attention
  - An approximate K-Nearest-Neighbor (kNN) lookup into the memory
  - Find top-k most relevant (key, value) pairs in the broad context

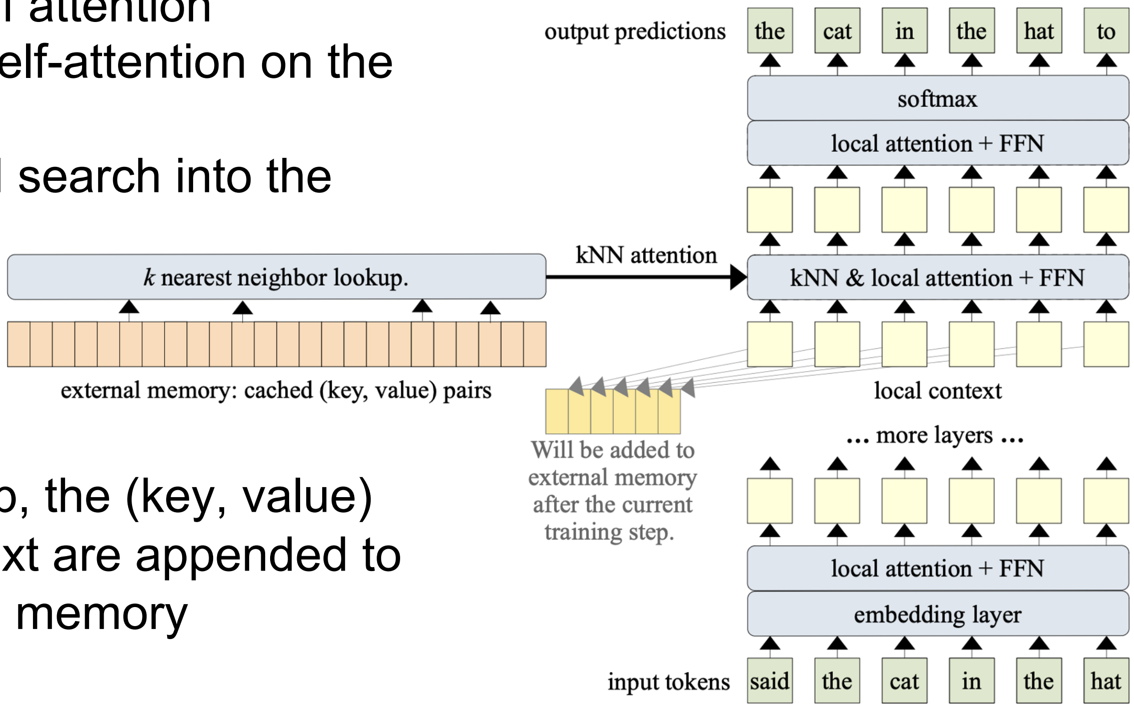


Innovations

- Precision
  - Other approaches average or summarize of tokens at long distances.
  - kNN lookup retrieves **exact values** even from the distant context.
- Scalability
  - In traditional transformer models, gradients are backpropagated through the entire model, updating weights of all the learned information .
  - In the non-differentiable external memory, key-value pairs **remain static** once they are stored and are not updated through the training process
  - The system focus **solely on retrieval** during inference without needing to re-learn or re-compute everything

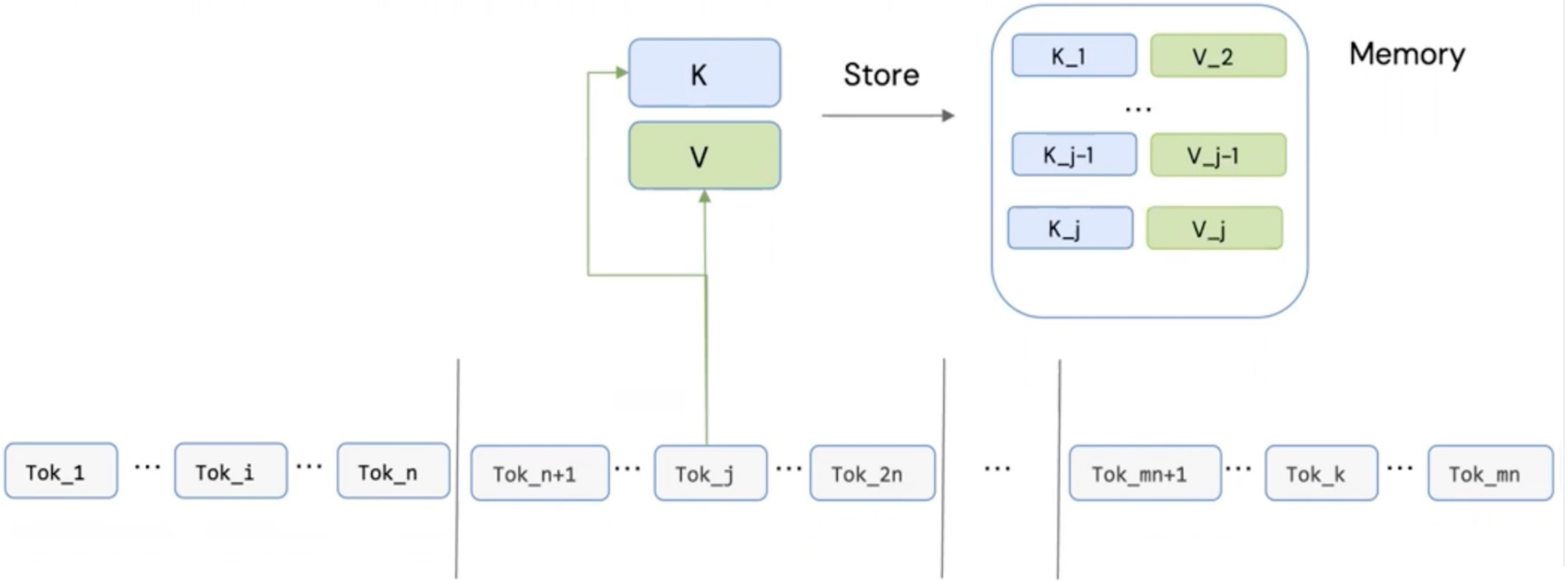
Memorizing Transformers Architecture

- Combines two forms of attention
  - Standard dense self-attention on the local context
  - Approximate KNN search into the external memory

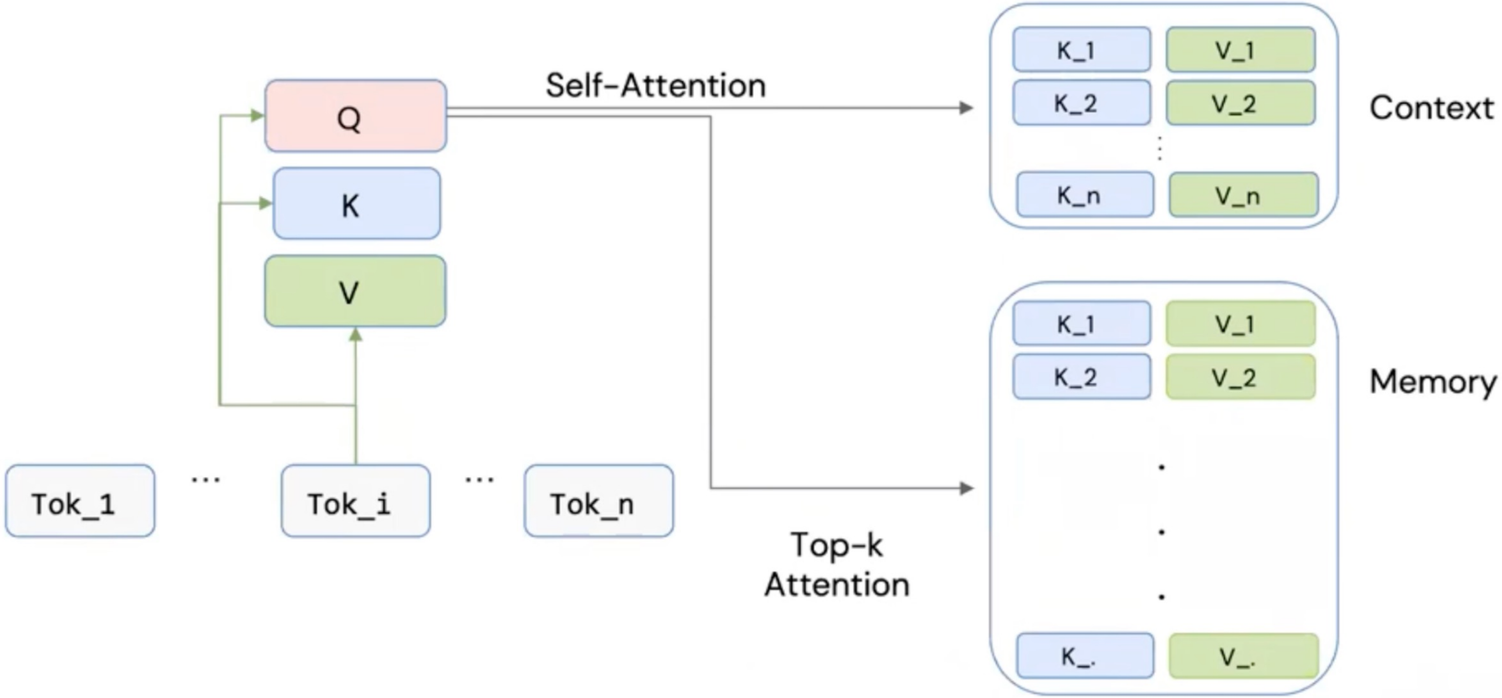


- After each training step, the (key, value) pairs in the local context are appended to the end of the external memory

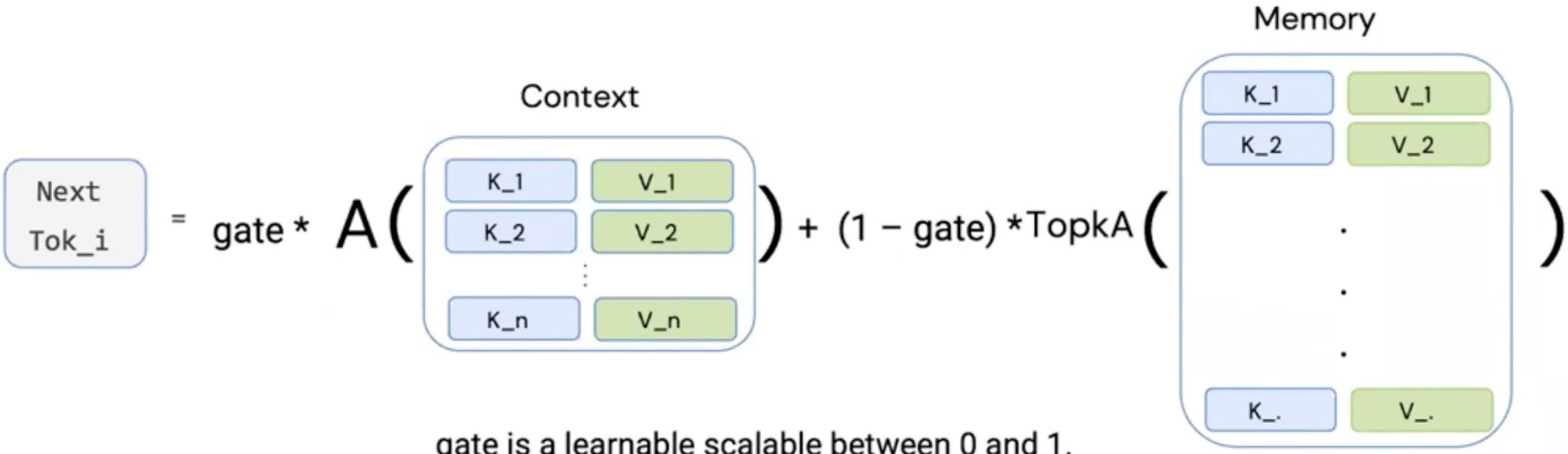
Memorizing Transformer Layers



Memorizing Transformer Layers



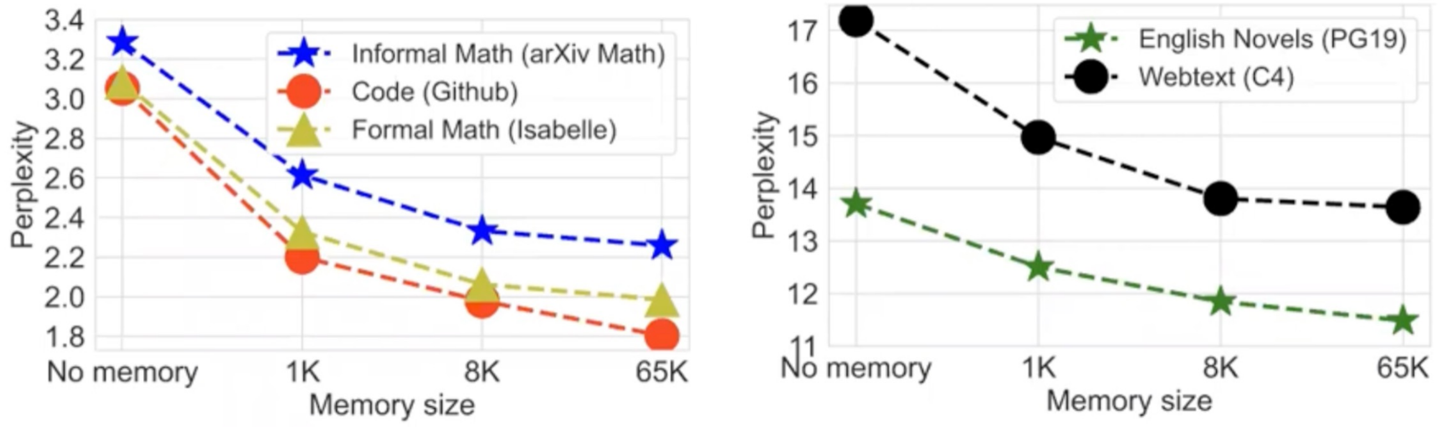
Memorizing Transformer Layers



gate is a learnable scalable between 0 and 1.

Improvements with External Memory

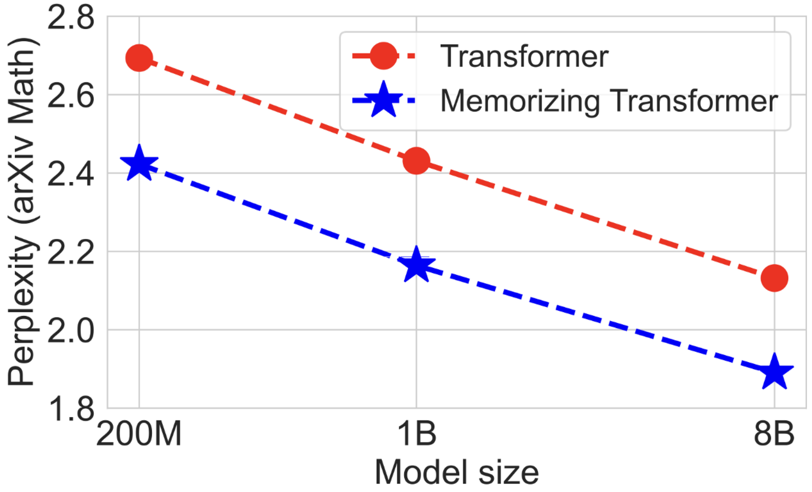
- Test on a variety of language modelling tasks involving long-form text
- Evaluate perplexity: The uncertainty of a model to predict the next word
- Lower perplexity values = better (more confident) predictions by model



- Model perplexity steadily improves with the size of external memory
- Diminishing marginal decreasing from an increasing memory size

Improvements by Memory on Large Models

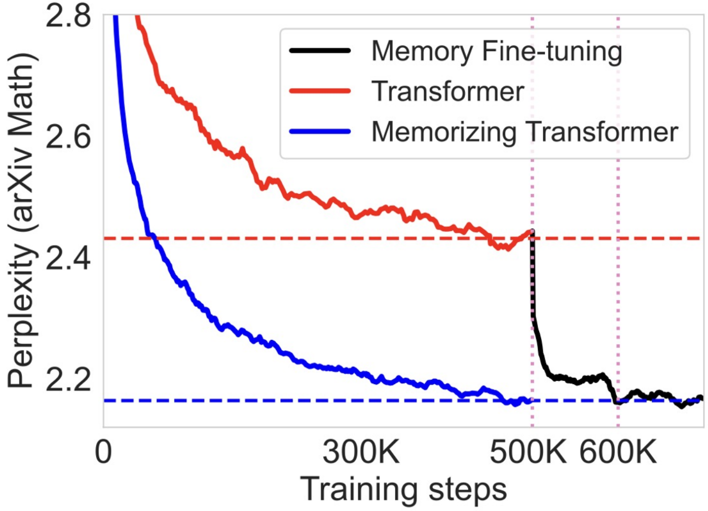
- Compare to normal transformers on arXiv math dataset
- Add a memory of size 8K to normal transformer models in different sizes
- The memory mechanism helps consistently when scaling model size up to 8B.
- 8K memory attained results comparable to the larger model which has 5-8X more trainable parameters



Fine-Tuning transformer to use memory

- Train memory from scratch v.s. Fine tunes the model to use memory

- Finetuning a 1B vanilla Transformer model to use external memory of size 65K.
- Within 20K steps (**4% of the pre-training time**), the fine-tuned model has already closed **85% of the gap** between it and the 1B Memorizing Transformer.
- After 100k steps it has closed the gap entirely.

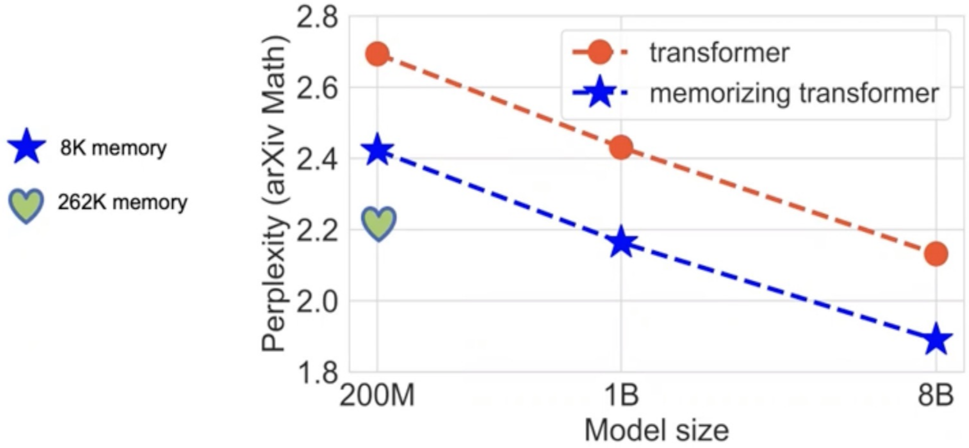


Fine-Tuning for a Larger Memory

- Firstly pretrain the model with a small memory and fine tunes it to make use of a larger memory (on the arXiv dataset)
- Increasing the size of external memory provided consistent gains up to a size of **262K**, which achieved results comparable to a **40X** larger model

Perplexity: The lower the better.

Pretrain	Fine-tune	Perplexity
8192	None	2.33
65K	None	2.26
65K	131k	2.23
65K	262k	**2.21**

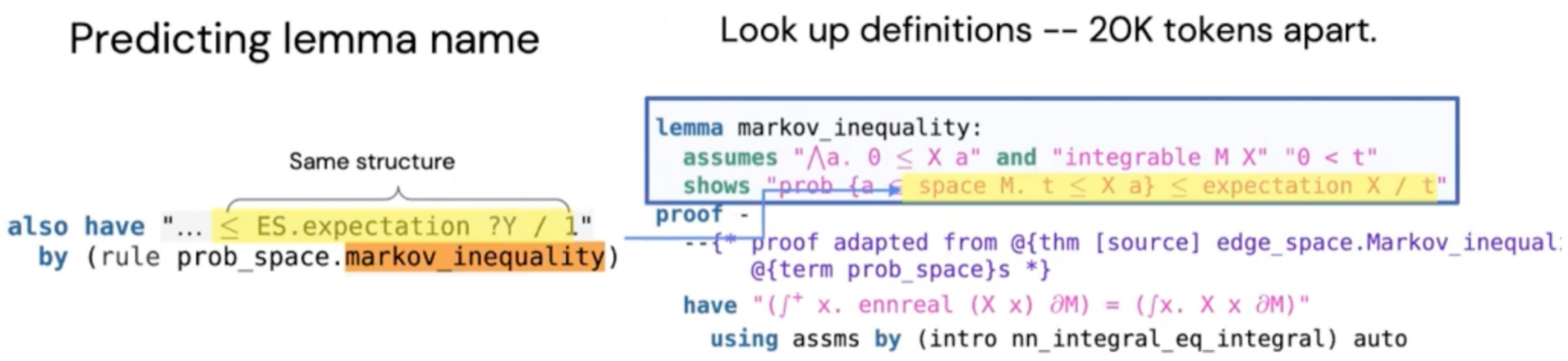


Information Retrieval Patterns

- A qualitative study of what the model was actually retrieving from external memory
- Find tokens which showed the biggest improvements in cross-entropy loss when the size of the memory was increased, and then examining the **top-k retrieved memories** for those tokens.
- The model gained the most when looking up **rare words**: proper names, references, citations, and function names, where the first use of a name is too far away from subsequent uses to fit in the local context.

Information Retrieval Patterns

- Examples of memory retrieval
- The retrieved surrounding context (highlighted) is the definition body of the mathematical object highlighted in the querying context.



Takeaways

- K-Nearest-Neighbor lookup into a large external memory
- Dramatically increases the length of the context that a language model can attend to
- Genericness: A large improvement across variety of long-document tasks
- Scalability: Perplexity continues to improve with increasing memory size
- A Memorizing Transformer does not need to be pre-trained from scratch
- Immediate utilization of newly acquired knowledge

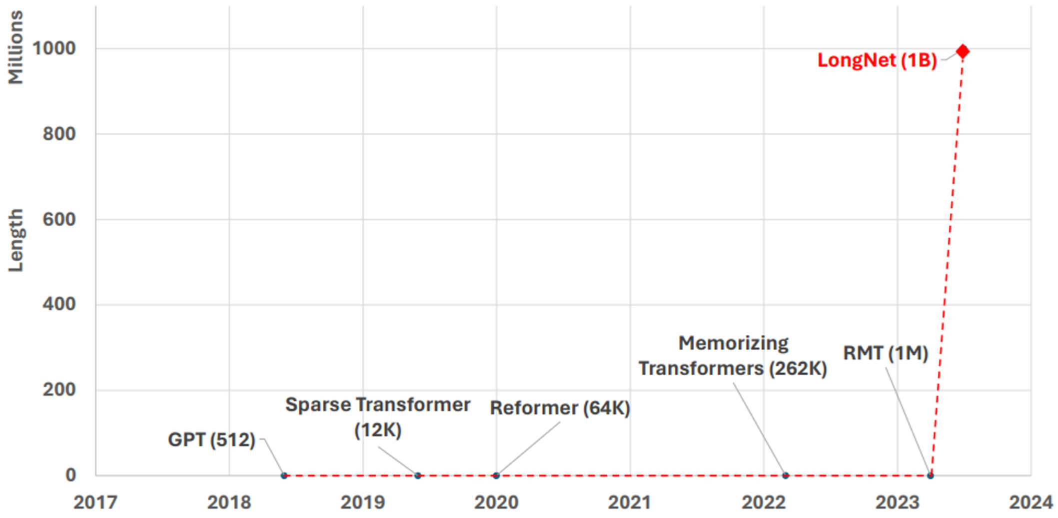
LONGNET: Scaling Transformers to 1,000,000,000 Tokens

Jiayu Ding Shuming Ma Li Dong Xingxing Zhang
Shaohan Huang Wenhui Wang Nanning Zheng Furu Wei

https://arxiv.org/pdf/2307.02486

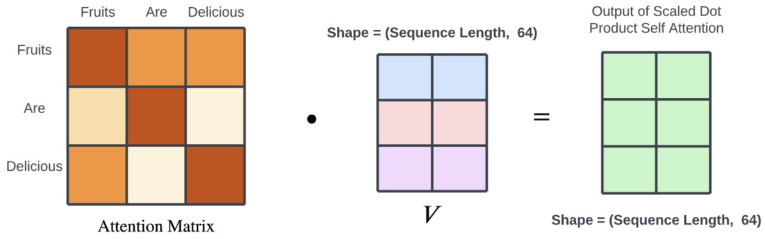
Background

- Conflicts between the demanding need to scale up LLMs and degrades on performances.
- Degrades originate in the computational complexity, which is quadratic.



Attention Recap

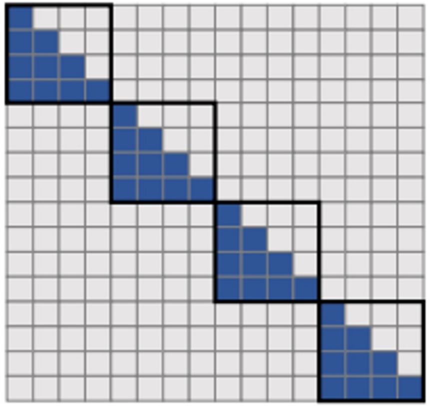
- Why is it quadratic?
- Turn quadratic into linear or near linear



Method	Computation Complexity
Recurrent	$\mathcal{O}(Nd^2)$
Vanilla Attention	$\mathcal{O}(N^2d)$
Sparse Attention	$\mathcal{O}(N\sqrt{N}d)$
Dilated Attention (This Work)	$\mathcal{O}(Nd)$

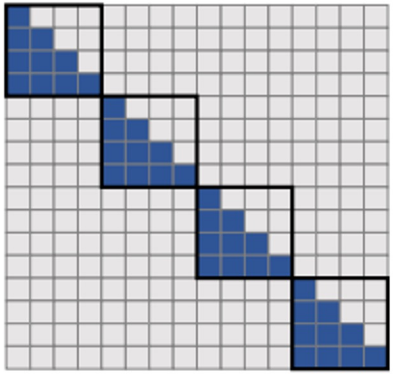
Dilated Attention - Key innovation

- Sparse attention did dramatically reduce the computation, but they are LOCAL!!!
- Dilated Attention with dilation rate = 1 is just the same as sparse attention.
- How to handle information flow

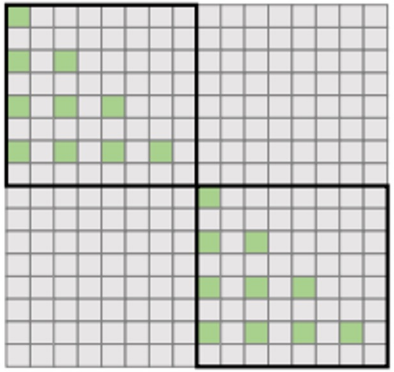


Dilated Attention

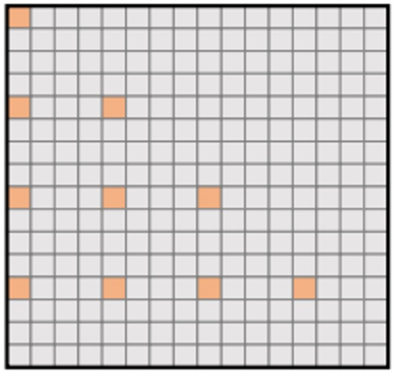
- Multiple dilation rates and stack the layers



Segment Length: 4
Dilated Rate: 1

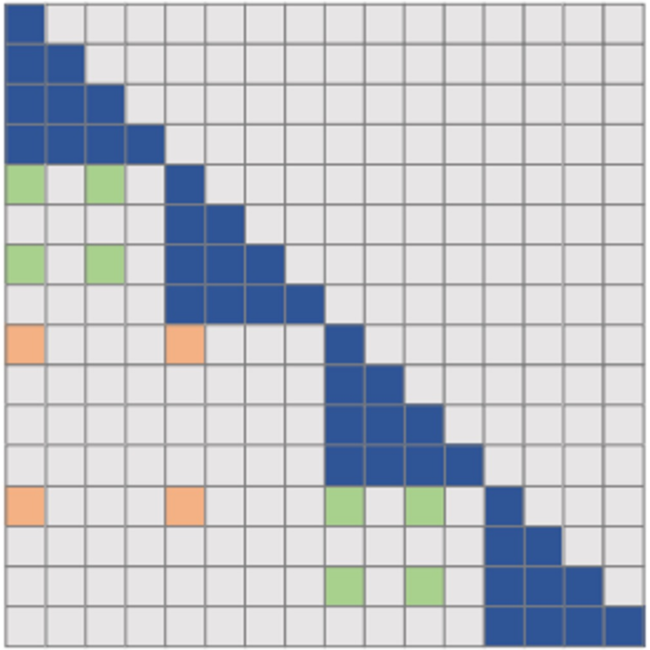


Segment Length: 8
Dilated Rate: 2



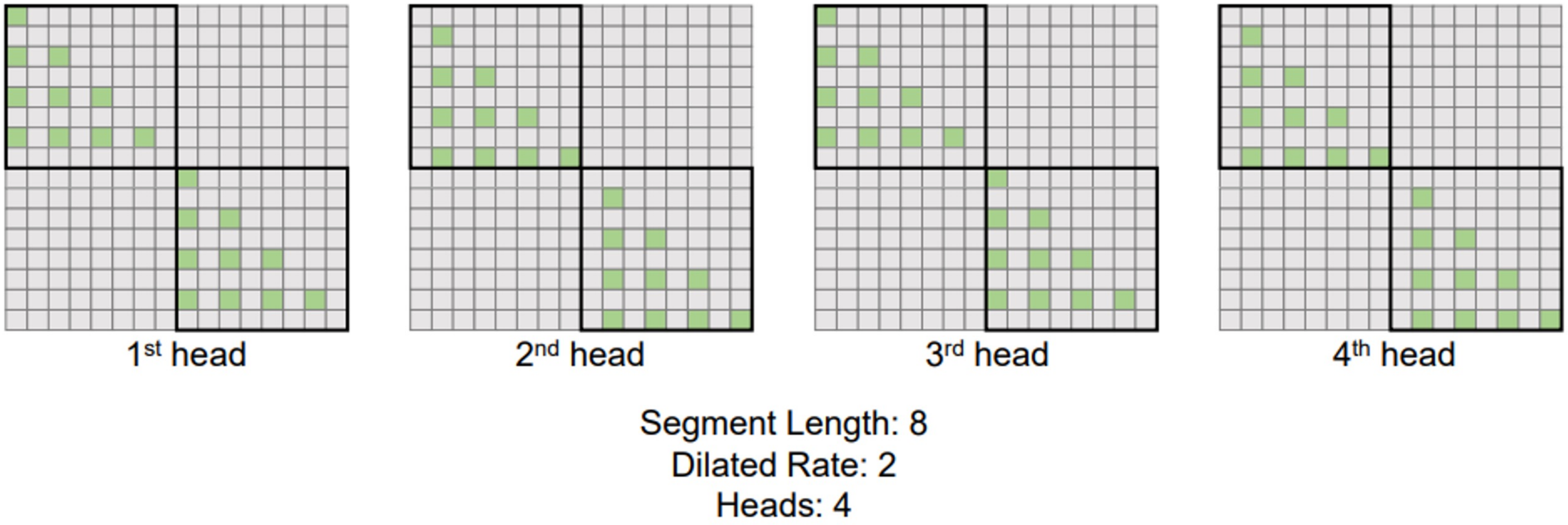
Segment Length: 16
Dilated Rate: 4

LongNet: Dilated Attention



Multihead Dilated Attention

- To make it converge even faster, we can have different patterns under the same dilation rate for each head.

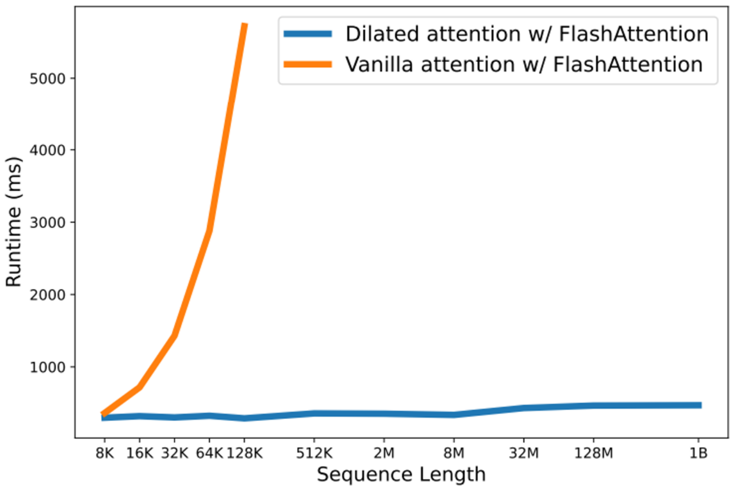


Computational Complexity

$$FLOPs = \frac{2N}{w}\left(\frac{w}{r}\right)^2 d = \frac{2Nwd}{r^2}$$

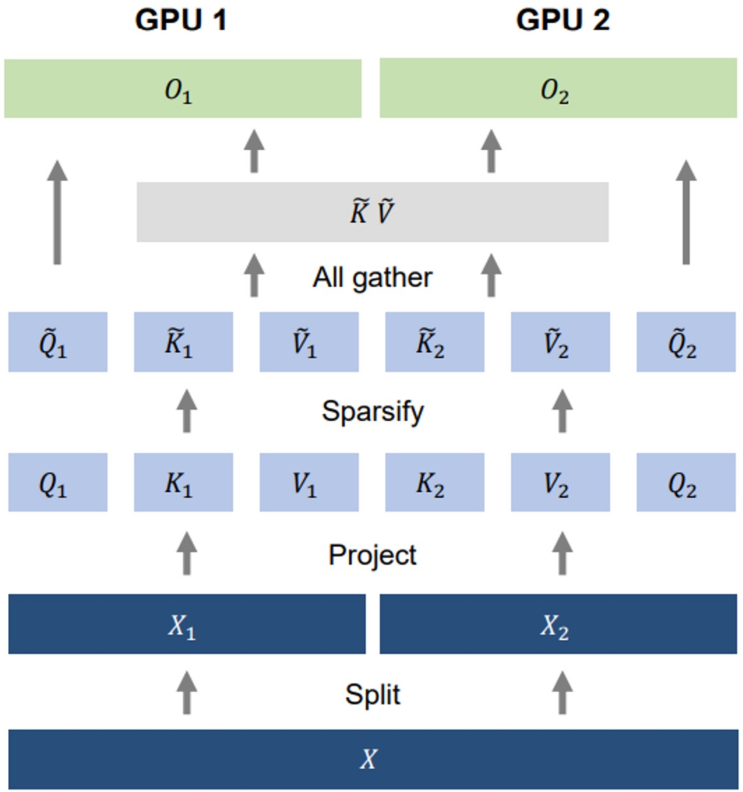
$$FLOPs = 2Nd\sum_{i=1}^{k}\frac{w_i}{r_i^2}$$

$$FLOPs = 2w_0Nd\sum_{i=0}^{k-1}\frac{1}{\alpha^i} \leq \frac{2\alpha}{\alpha-1}w_0Nd \quad (\alpha > 1)$$



The complexity is now *O(Nd)*. LINEAR!

Parallelizing computation on GPUs



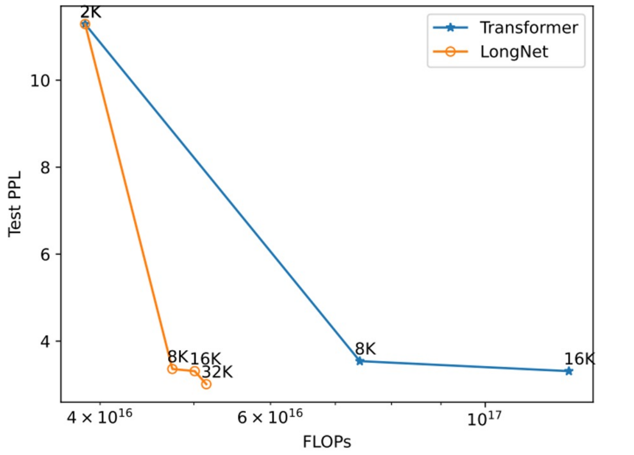
Results

Perplexity:

- LongNet consistently outperform the benchmark models with different context lengths.
- LongNet achieved similar performance level with significantly less computational cost.

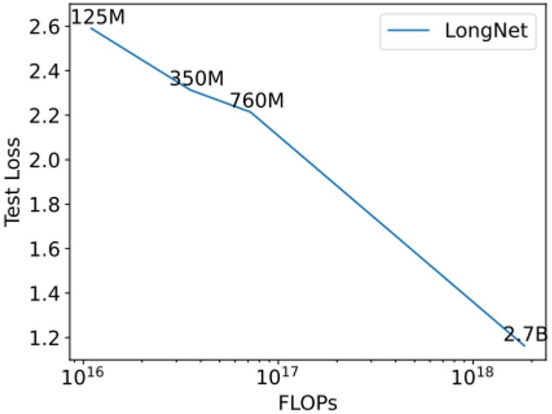
Model	Length	Batch	Github 2K	Github 8K	Github 32K
Transformer [VSP+17]	2K	256	4.24	5.07	11.29
Sparse Transformer [CGRS19]	8K	64	4.39	3.35	8.79
LONGNET (ours)	8K	64	4.23	3.24	3.36
Sparse Transformer [CGRS19]	16K	32	4.85	3.73	19.77
LONGNET (ours)	16K	32	4.27	3.26	3.31
Sparse Transformer [CGRS19]	32K	16	5.15	4.00	3.64
LONGNET (ours)	32K	16	4.37	3.33	3.01

Table 2: Perplexity of language models for LONGNET and the baselines.

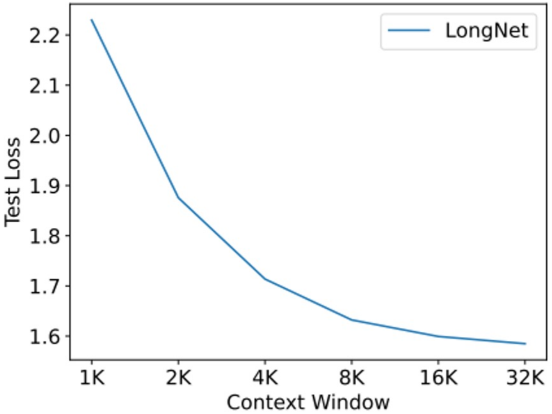


Results

- Larger model size → lower test loss
- Larger context window → lower test loss



(a)



(b)

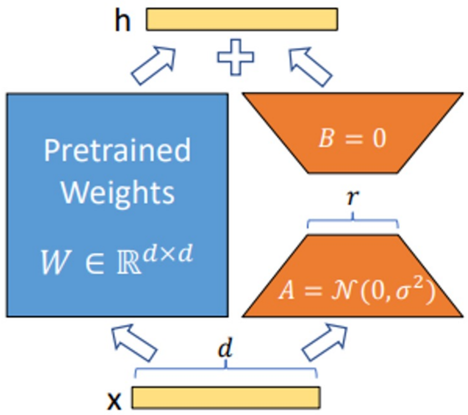
LONGLORA: EFFICIENT FINE-TUNING OF LONGCONTEXT LARGE LANGUAGE MODELS

Yukang Chen Shengju Qian Haotian Tang Xin Lai Zhijian Liu Song Han Jiaya Jia

https://arxiv.org/pdf/2309.12307

LoRA Recap

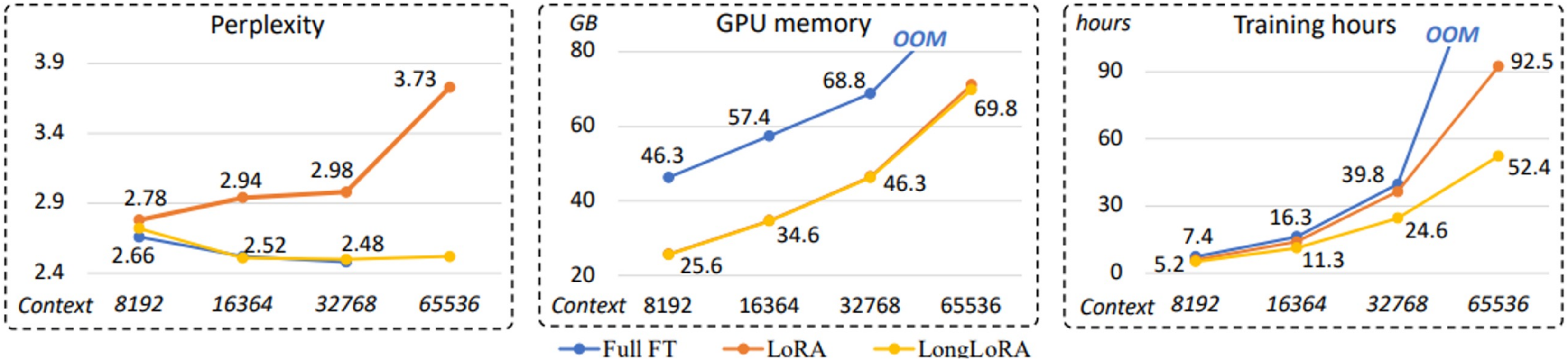
- Observations: Weights learned after training contains redundancies.
- Using low-rank approximation instead of tuning the entire weights in the model.



$$W_0 + \Delta W = W_0 + BA,$$

Problems with LoRA

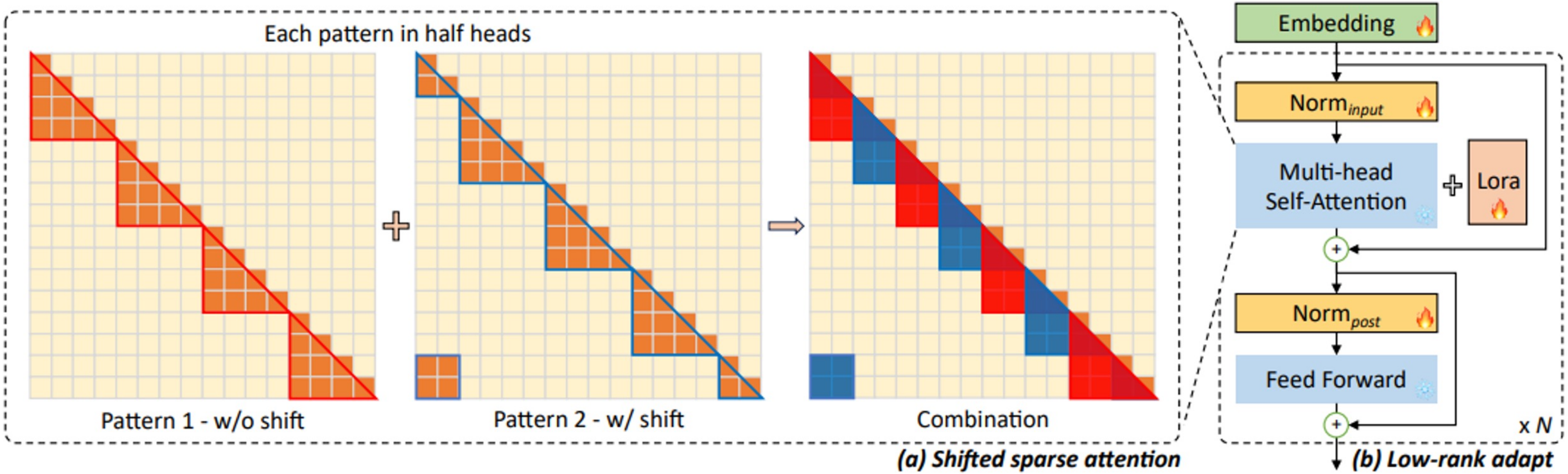
- LoRA is neither sufficiently effective nor efficient when the context length increases to more than 8K tokens.



*A perplexity of N can be interpreted as the model being as confused as if it had to choose uniformly among N options for each word. The lower, the better.

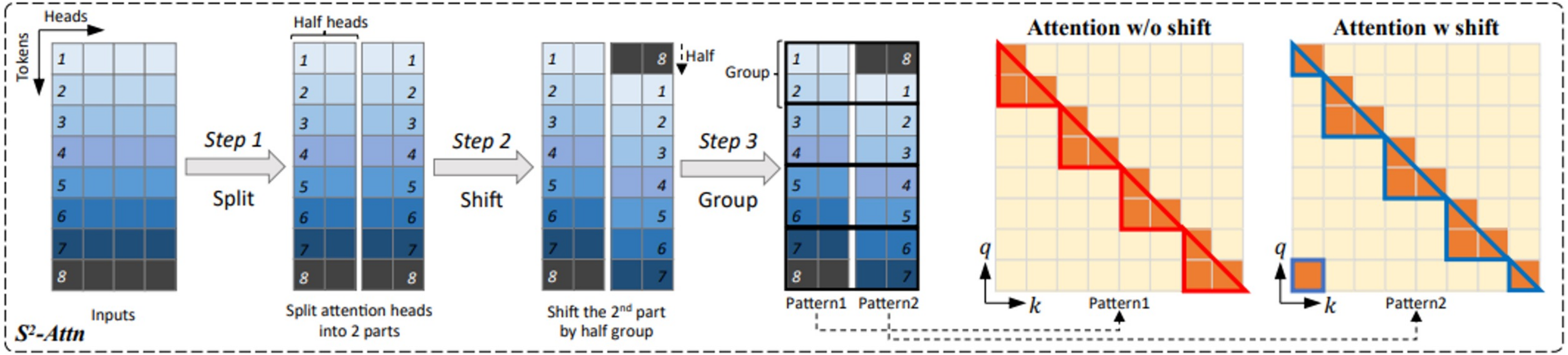
What is LongLoRA

- Shifted Sparse Attention (S^2 attention)
- Parameter efficient tuning



S^2 Attention

- Split attention heads into two partitions, shift one of the partition half the group size.
- Reduce computation by local sparse attention.
- Ensure information flow by shifting.



S^2 Attention

Algorithm 1: Pseudocode of S^2-Attn in PyTorch-like style.

```
# B: batch size; S: sequence length or number of tokens; G: group size;
# H: number of attention heads; D: dimension of each attention head

# qkv in shape (B, N, 3, H, D), projected queries, keys, and values
# Key line 1: split qkv on H into 2 chunks, and shift G/2 on N
qkv = cat((qkv.chunk(2, 3)[0], qkv.chunk(2, 3)[1].roll(-G/2, 1)), 3).view(B*N/G,G,3,H,D)

# standard self-attention function
out = self_attn(qkv)

# out in shape (B, N, H, D)
# Key line 2: split out on H into 2 chunks, and then roll back G/2 on N
out = cat((out.chunk(2, 2)[0], out.chunk(2, 2)[1].roll(G/2, 1)), 2)
```

`cat`: concatenation; `chunk`: split into the specified number of chunks; `roll`: roll the tensor along the given dimension.

S^2 Attention

Design process:

- Sparse attention to reduce computational cost
- How to handle information flow → Shifting

Pros:

- Consistent to Full attention: same architecture & full attention while inferencing
- Easy implementation

Parameter Efficient Tuning

- Lora only works with attention layers → open normalization and embedding layers for training
- These layers only occupy limited parameters in the whole model and thus will not introduce new computational cost.

Table 2: **Finetuning normalization and embedding layers is crucial for low-rank long-context adaptation**. Llama2 7B (Touvron et al., 2023b) models with the proposed S^2-Attn are trained on the RedPajama (Computer, 2023) dataset. The target context length is 32768. '+ Normal / Embed' means normalization or embedding layers are trainable. Perplexity results are evaluated on PG19 (Rae et al., 2020) validation set. For long context adaptation, there is a large performance gap between standard LoRA (Hu et al., 2022) and full fine-tuning. Without trainable normalization or embeddings, larger ranks in LoRA can not close this gap.

Method	Full FT	LoRA (rank)						LoRA (rank = 8)		
		8	16	32	64	128	256	+ Norm	+ Embed	+ Norm & Embed
PPL	8.08	11.44	11.82	11.92	11.96	11.97	11.98	10.49	8.29	8.12

Evaluations

Experiment settings:

- 7B ,13B, 20B Llama2 pretrained;
- Position indices all rescaled based on *positional encoding*
- Trained on a single 8× A100 GPUs machine
- Fine tune objectives: Next token prediction
- Two tasks:
  - Long Sequence Language Modeling
  - Topic Retrieval

Evaluations - Long Sequence Language Modeling

Perplexity evaluation on PG19 dataset

Size	Training Context Length	LongLoRA S^2-Attn	LongLoRA LoRA$^+$	2048	4096	8192	16384	32768
7B	8192			3.14	2.85	2.66	-	-
		✓		3.15	2.86	2.68	-	-
		✓	✓	3.20	2.91	2.72	-	-
	16384	✓		3.17	2.87	2.68	2.55	-
		✓	✓	3.17	2.87	2.66	2.51	-
	32768	✓		3.20	2.90	2.69	2.54	2.49
		✓	✓	3.35	3.01	2.78	2.61	2.50
13B	8192			2.96	2.69	2.53	-	-
		✓		3.01	2.74	2.57	-	-
		✓	✓	3.04	2.77	2.60	-	-
	16384	✓		2.99	2.72	2.53	2.40	-
		✓	✓	3.03	2.74	2.55	2.41	-
	32768	✓		3.04	2.75	2.56	2.42	2.33
		✓	✓	3.05	2.76	2.57	2.42	2.32

Evaluations

Maximum context length can be tuned

Size	Training Context Length	Evaluation Context Length						
		2048	4096	8192	16384	32768	65536	100,000
7B	100,000	3.36	3.01	2.78	2.60	2.58	2.57	2.52
13B	65536	3.20	2.88	2.66	2.50	2.39	2.38	-
70B	32768	2.84	2.57	2.39	2.26	2.17	-	-

Topic Retrieval

Evaluation Context	3k	6k	10k	13k	16k
ChatGLM2-6B (Du et al., 2022)	0.88	0.46	0.02	0.02	0.02
MPT-30B-chat (Team, 2023a)	0.96	**1.0**	0.76	-	-
MPT-7B-storywriter (Team, 2023b)	0.46	0.46	0.28	0.34	0.36
LongChat-13B (Li et al., 2023)	**1.0**	**1.0**	**1.0**	**0.98**	0.9
Ours-13B	**1.0**	0.98	0.98	**0.98**	**0.94**

Evaluations

Efficiency Evaluation

Substantially decreases FLOPs, particularly with longer context lengths.

Group size

Set group size as ¼ in experiments based on the results.

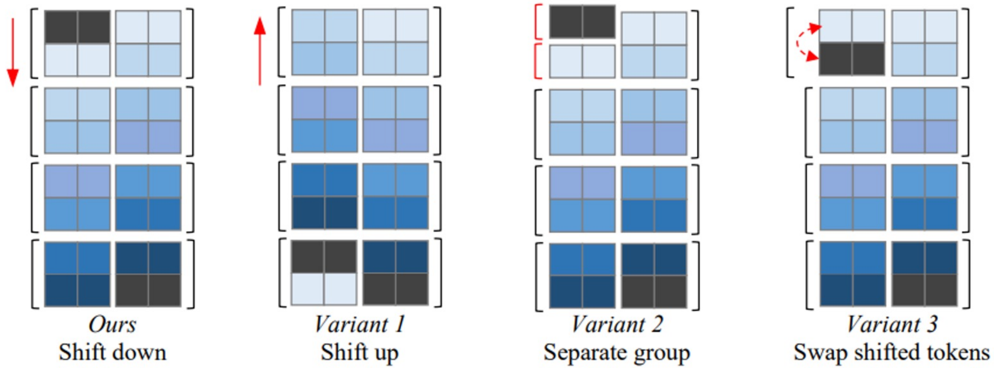
Context Length	S^2-Attn	FLOPs (T) Attn	Proj	FFN	Others	Total
8192	✗	35.2	35.2	70.9	2.2	143.5
	✓	8.8				117.1
16384	✗	140.7	70.4	141.8	4.3	357.2
	✓	35.2				251.7
32768	✗	562.9	140.7	283.7	8.7	996.0
	✓	140.7				573.8
65536	✗	2251.8	281.5	567.4	17.3	3118.0
	✓	562.9				1429.1

Context Length	Full	1/2	1/4	1/6	1/8
8192	8.02	8.04	8.04	8.10	8.16
16384	7.82	7.84	7.86	7.94	7.98

Ablation Studies

Variants of S^2 Attention

Shifting direction has no effect on the perplexity; performances are similar.



Attn	Full	Ours	Variant 1	Variant 2	Variant 3
PPL	8.02	8.04	8.04	8.03	8.05

Lost in the Middle: How Language Models Use Long Contest

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape,
Michele Bevilacqua, Fabio Petroni, Percy Liang

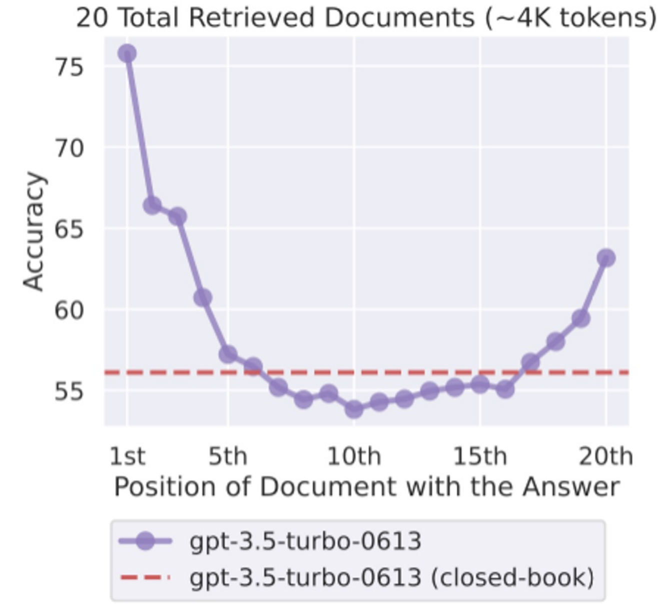
https://arxiv.org/abs/2307.03172

Background

Language models have significantly improved, enabling them to handle longer text inputs.

Despite these advancements, efficiently utilizing long text contexts remains a challenge.

How effectively do modern language models actually utilize long text contexts? Does the performance of these models significantly deteriorate when the relevant information is positioned in the middle of the text?



Multi-document question answering

Input: (1). A question to answer; (2). k documents

Dataset Utilization: NaturalQuestions-Open dataset featuring historical Google search queries and human-annotated answers from Wikipedia.

Open models: MPT-30B-Instruct, LongChat-13B

Closed models: GPT-3.5-Turbo, GPT-3.5-Turbo (16K), Claude-1.3, Claude-1.3 (100K)

Input Context

```
Write a high-quality answer for the given question using only the provided search
results (some of which might be irrelevant).

Document [1](Title: Asian Americans in science and technology) Prize in physics for
discovery of the subatomic particle J/ψ. Subrahmanyan Chandrasekhar shared...
Document [2](Title: List of Nobel laureates in Physics) The first Nobel Prize in
Physics was awarded in 1901 to Wilhelm Conrad Röntgen, of Germany, who received...
Document [3](Title: Scientist) and pursued through a unique method, was essentially
in place. Ramón y Cajal won the Nobel Prize in 1906 for his remarkable...

Question: who got the first nobel prize in physics
Answer:
```

Desired Answer

```
Wilhelm Conrad Röntgen
```

Input Context

```
Write a high-quality answer for the given question
using only the provided search results (some of
which might be irrelevant).

Document [1](Title: List of Nobel laureates in
Physics) ...
Document [2](Title: Asian Americans in science and
technology) ...
Document [3](Title: Scientist) ...

Question: who got the first nobel prize in physics
Answer:
```

Desired Answer

```
Wilhelm Conrad Röntgen
```

Input Context

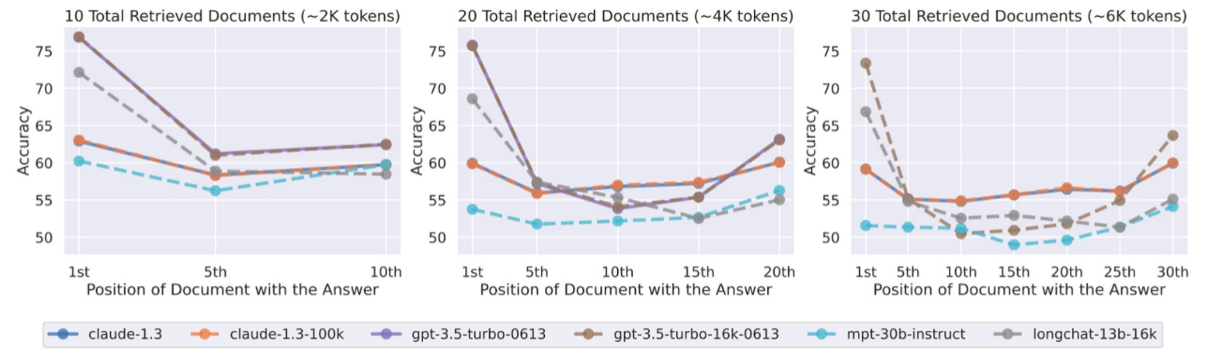
```
Write a high-quality answer for the given question
using only the provided search results (some of
which might be irrelevant).

Document [1](Title: Asian Americans in science and
technology) ...
Document [2](Title: List of Nobel laureates in
Physics) ...
Document [3](Title: Scientist) ...
Document [4](Title: Norwegian Americans) ...
Document [5](Title: Maria Goeppert Mayer) ...

Question: who got the first nobel prize in physics
Answer:
```

Desired Answer

```
Wilhelm Conrad Röntgen
```

Multi-document question answering

- Model performance is highest when relevant information occurs at the beginning or end of its input context.
- Extend-context models are not necessarily better at using input context.



Model	Closed-Book	Oracle
LongChat-13B (16K)	35.0%	83.4%
MPT-30B-Instruct	31.5%	81.9%
GPT-3.5-Turbo	56.1%	88.3%
GPT-3.5-Turbo (16K)	56.0%	88.6%
Claude-1.3	48.3%	76.1%
Claude-1.3 (100K)	48.2%	76.4%

Table 1: Closed-book and oracle accuracy of language models on the multi-document question answering task.

How well can language models retrieve from input context

- Objective: Assess model adaptability to input changes and complex scenarios.
- Input: Serialized JSON with key-value pairs.
- Task: Synthetic key-value retrieval to find specific values.
- Evaluation: Focuses on model performance amid input context and structural changes.

Input Context

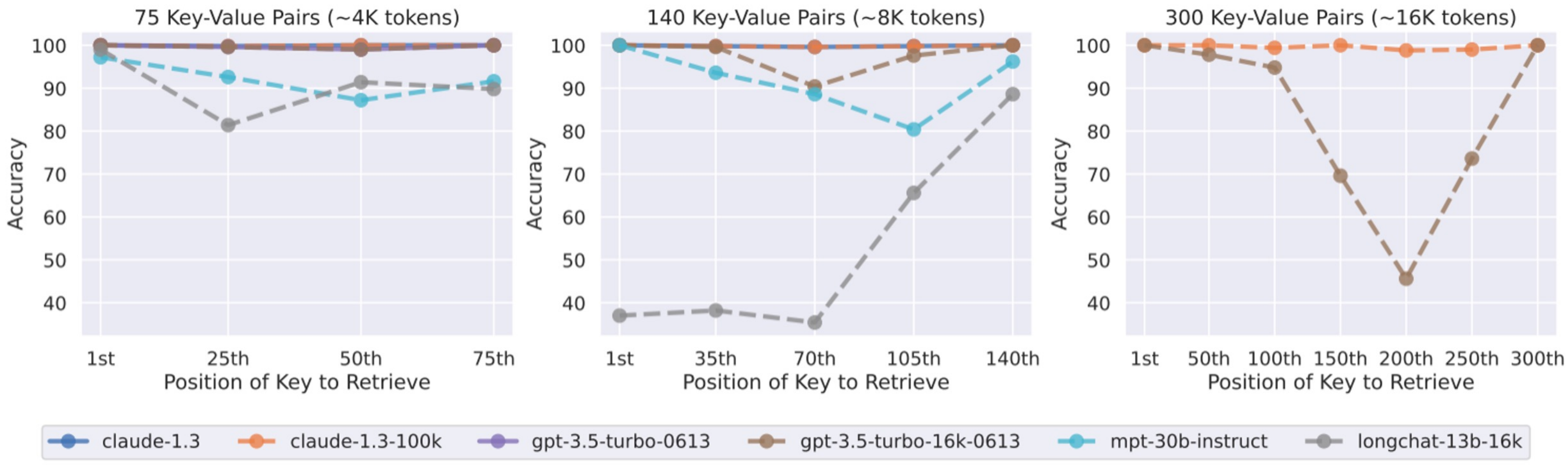
```
Extract the value corresponding to the specified key in the JSON object below.

JSON data:
{"2a8d601d-1d69-4e64-9f90-8ad825a74195": "bb3ba2a5-7de8-434b-a86e-a88bb9fa7289",
 "a54e2eed-e625-4570-9f74-3624e77d6684": "d1ff29be-4e2a-4208-a182-0cea716be3d4",
 "9f4a92b9-5f69-4725-ba1e-403f08dea695": "703a7ce5-f17f-4e6d-b895-5836ba5ec71c",
 "52a9c80c-da51-4fc9-bf70-4a4901bc2ac3": "b2f8ea3d-4b1b-49e0-a141-b9823991ebeb",
 "f4eb1c53-af0a-4dc4-a3a5-c2d50851a178": "d733b0d2-6af3-44e1-8592-e5637fdb76fb"}

Key: "9f4a92b9-5f69-4725-ba1e-403f08dea695"
Corresponding value:
```

Desired Output

```
703a7ce5-f17f-4e6d-b895-5836ba5ec71c
```

How well can language models retrieve from input context

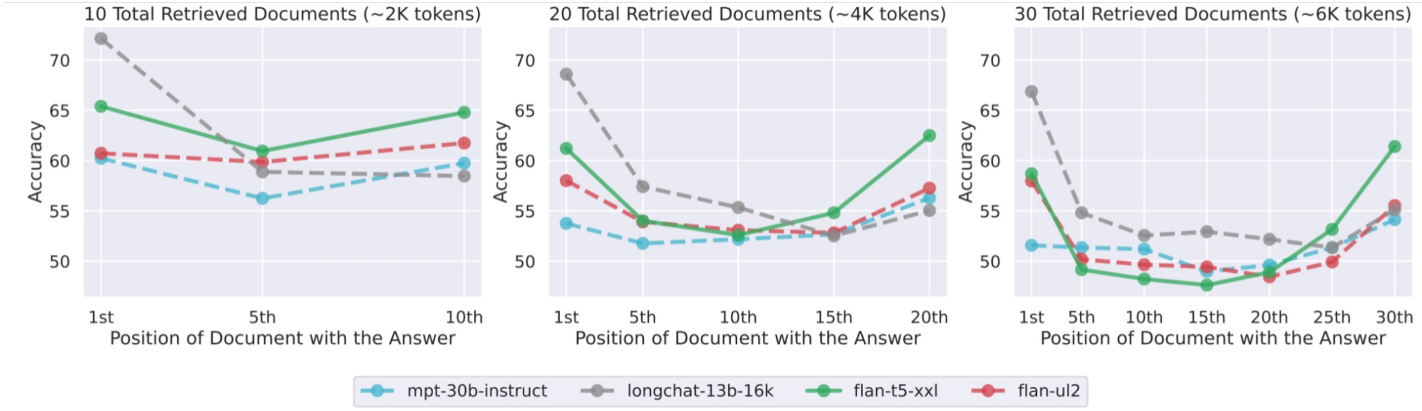
The models like Claude-1.3 perform almost perfectly in retrieving values, regardless of the number of distractors.
Models such as GPT-3.5-Turbo and LongChat-13B exhibit difficulties when key-value pairs are positioned in the middle of the input, with LongChat-13B generating code to retrieve keys instead of directly outputting values.



Why Are Language Models Not Robust to Changes in the Position of Relevant Information?

Effect of Model Architecture

- Decoder-only models struggle with long input contexts, especially when the relevant information shifts within the input.
- Encoder-decoder models like Flan-T5-XXL and Flan-UL2 show better resilience and performance due to their bi-directional context processing capabilities.



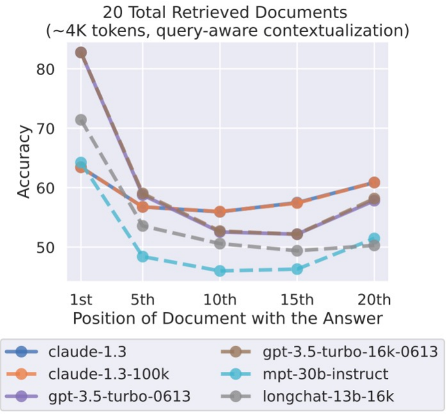
Why Are Language Models Not Robust to Changes in the Position of Relevant Information?

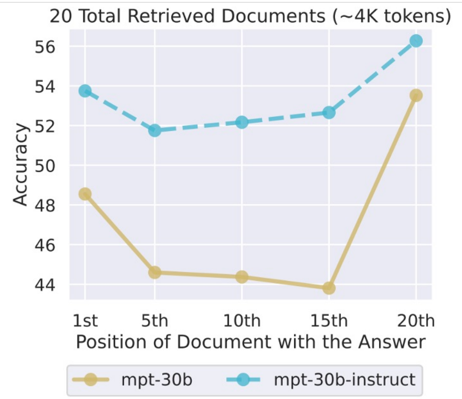
Effect of Query-Aware Contextualization

- Placing the query before and after documents
- Minimal improvement in question answering tasks; notable only when information is at the very beginning or end of the input.

Effect of Instruction Fine-Tuning

- Models are fine-tuned on instruction-specific datasets to enhance their response quality.
- Fine-tuning helps reduce performance disparity in models, especially in worst-case scenarios, but overall trends remain similar.





Is more context is always better? A case study with open-domain QA

Experiment Setup:

- Retriever-reader model with a retrieval system fine-tuned on MS-MARCO.
- Recall and accuracy based on retrieved documents containing correct answers.

Findings:

- Retrieval performance peaks with just 20 documents.
- Slight accuracy improvement (~1-1.5%) with more context but at a high computational cost.
- Suggests better document reranking or truncating retrieved lists over simply increasing context.

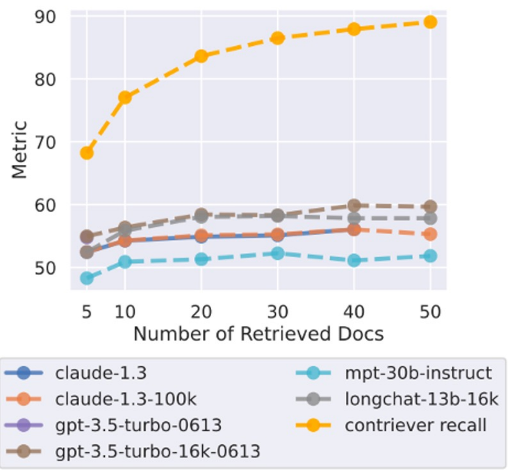


Figure 11: Retriever recall and model performance as a function of the number of retrieved documents. Model performance saturates long before retriever recall, indicating that the models have difficulty making use of the extra retrieved documents.

Conclusion

- Performance Degradation with Changing Information Position
  - Models struggle to robustly access and utilize information in long input contexts.
  - Performance is often lowest when the relevant information is located in the middle of long input contexts.

- Contributions and Future Directions
  - Provide a better understanding of how language models utilize their input context.
  - Propose new evaluation protocols for future long-context models and highlight areas for improvement.

Q&A