Landmark Attention

Random-Access Infinite Context Length for Transformers

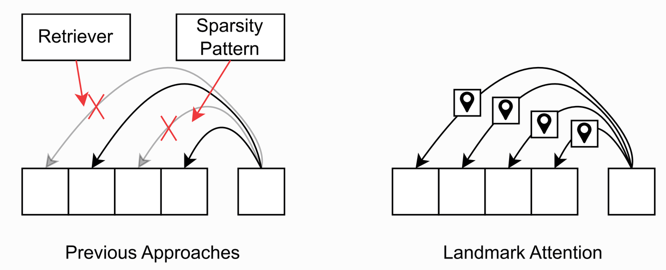
Amirkeivan Mohtashami, Martin Jaggi

Machine Learning and Optimization Laboratory

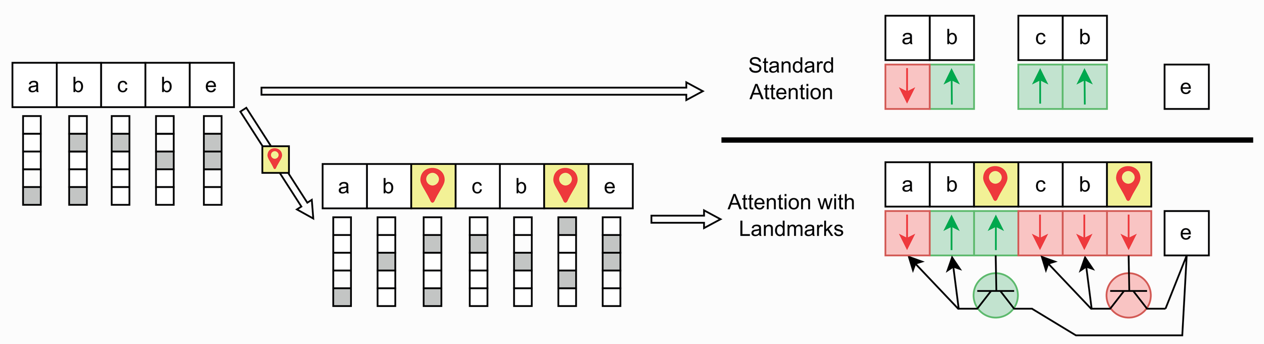
EPFL

Overview

- Attention is a key component in the highly successful Transformer architecture.
- However, it has a quadratic computational cost, limiting the input (context) length.
- Previous approaches to alleviate this cost sacrifice Attention's random-access flexibility.
- We propose Landmark Attention to allow the attention itself to be used for retrieval, maintaining its random access flexibility.

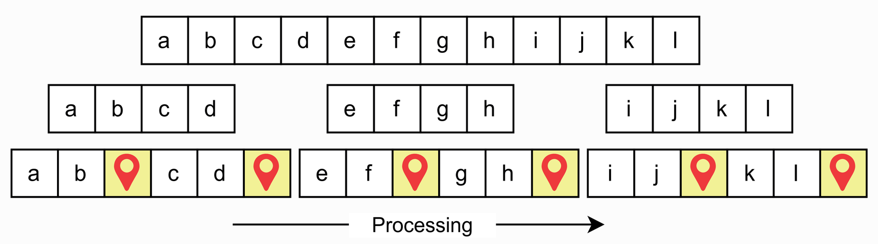


Landmark Attention



- We break the input into **blocks of fixed length** and introduce a special token for each block, called a **landmark**, which acts as a gate for attending to its corresponding block.
- The gating mechanism is controlled by the attention score to the landmark token.
- At inference time, we compute the attention scores of the landmarks and retrieve the blocks corresponding to the highest scoring landmarks (active gates), integrating them into the attention.
- Our proposed approach maintains the random-access flexibility of attention and offers an alternative solution to the recurrent memory approaches.

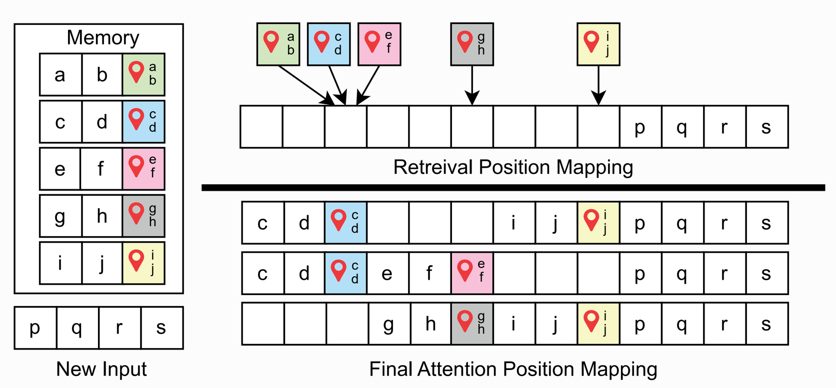
Inference



- First, break the input into **chunks** and augment the sequence by landmark tokens. Each chunk contains multiple blocks.
- The chunks are iteratively fed to the model from the beginning to the end.
- When processing each chunk at each layer, for each token we first compute the attention score to the landmark tokens currently in the cache.
- We only compute the attention score to tokens in the blocks of the top k high scoring landmarks.
- This is a close approximation to training as tokens in blocks with low scoring landmarks will not receive a high weight anyway.
- The performance can be further improved by using nearest neighbor data structures.
- Computation cost is immediately improved by a factor of the block size (**50x speedup** in this work).
- The chunk size can be chosen smaller than the training context size to **decouple training and inference context lengths.**

Stingy Position Mapping

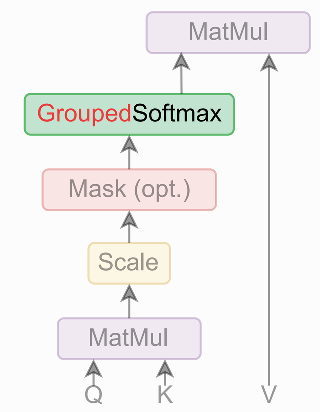
- **Challenge:** Transformers are unable to extrapolate to position indices not observed during training.
- Previous methods proposed to alleviate this flaw usually penalize or prevent attention to long distance tokens.
- Instead we propose a special approximate position mapping scheme called stingy position mapping.
- The position mapping maintains the position of the last k blocks but maps all earlier blocks to the same position.
- The retrieved blocks are prepended in order to the current chunk with an empty block separating retrieved blocks in the last k blocks from earlier ones.



Paper: Github:

Training

- Mostly standard training procedure is used except for the following changes:
  - Landmark token is added to the vocabulary, increasing the size by one.
  - Landmark tokens are inserted every after every ℓ_{block} tokens.
  - GroupedSoftmax is used to compute Softmax in groups.
  - A special grouping scheme imposes a hierarchy leading to the gating mechanism that can be used for retrieval.



Landmark Attention

- Given the vector of dot products v and grouping g, the grouped softmax is computed as:

$$\sigma_G(\mathbf{v},\mathbf{g})_i := \text{GroupedSoftmax}(\mathbf{v},\mathbf{g})_i := \frac{e^{\mathbf{v}_i}}{\sum_{j:\mathbf{g}_j=\mathbf{g}_i} e^{\mathbf{v}_j}}.$$

- For each block, we create a separate group and put its regular (non-landmark) tokens in that group.

$p_i \longrightarrow$ index of landmark token corresponding to i-th token's block

- When computing the attention weights for the i-th token, landmark tokens for other blocks are placed in the same group as the i-th token.
- The landmark token for the i-th token's block is ignored when computing the attention weights for the i-th token.
- Using the above grouping, the attention weight for each token can be computed as the product of the token's attention weight and it's corresponding landmark token's attention weight.
- Under this scheme, the attention weights sum to one same as in the standard Softmax function.

$$\mathbf{G}_{i,j} := \begin{cases} p_j & p_j \neq j \\ -1 & p_i = j \\ p_i & \text{otherwise} \end{cases}$$

- Since tokens in the same block and the landmark tokens share the Softmax group, the model has to choose between attending to other blocks and current tokens.
- Intuitively, the grouping forces the model to only attend to relevant blocks because of this trade-off.

$$\begin{aligned}\mathbf{S}_{i,j} &:= \text{SoftmaxScore}(\mathbf{Q},\mathbf{K})_i \\ &:= \text{GroupedSoftmax}\left(\frac{\mathbf{Q}_i^\top \times \mathbf{K}}{\sqrt{d_{\text{head}}}}, \mathbf{G}_i\right)\end{aligned}$$

$$\text{Att}(\mathbf{Q},\mathbf{K})_{i,j} := \begin{cases} 0 & p_j = j \\ \mathbf{S}_{i,j} & \mathbf{G}_{i,j} = \mathbf{G}_{i,i} \wedge p_j \neq j \\ \mathbf{S}_{i,j}\cdot\mathbf{S}_{i,p_j} & \mathbf{G}_{i,j} \neq \mathbf{G}_{i,i} \wedge p_j \neq j \end{cases}$$

i	0	1	2	3	4	5	6	7	8
$\text{mask} + \frac{Q_6^T K}{\sqrt{d_{\text{head}}}}$	1	1	1	1	1	1	1	mask	mask
p_i	2	2	2	5	5	5	8	8	8
$G_{6,j}$	2	2	8	3	3	8	8	8	-1
$S_{6,j}$	0.5	0.5	0.33	0.5	0.5	0.33	0.33	0	ign
$W_{6,j}$	0.167	0.167	0	0.167	0.167	0	0.33	0	0

Language Modeling

Eval. Length	ℓ_{local}	XL cache	Blocks	k	Attention Size	PG19	arXiv	
512	512	None	None	-	512	16.12	4.01	Baseline
	360	None	None	-	360	16.76	4.31	
	250	None	10	2	360	16.23	4.01	Ours
2048	256	256	None	-	512	14.72	-	[9]
	250	None	40	2	360	15.14	3.43	Ours
	350	None	40	2	460	15.07	3.41	
	300	None	40	3	460	14.97	3.36	
	250	None	20	4	460	15.02	3.37	
	250	None	40	4	460	14.92	3.35	
4096	256	256	None	-	512	14.55	-	[9]
	250	None	40	4	460	14.79	3.19	Ours
	250	None	80	2	370	15.00	3.29	
	250	None	80	4	470	14.72	3.18	

- We train a 12-layer decoder-only Transformer with 8 heads and 128 hidden dimension on language modeling tasks.
- In particular, we consider perplexity on PG-19 dataset and math papers from arXiv.
- Results demonstrate that Landmark Attention allows inference at much larger context lengths than training context length.
- The performance at the larger context length is comparable to a Transformer-XL trained directly at the larger context lengths on PG-19.

LLaMA 7B 32k

- LLaMA 7B is originally trained at 2048 context length.
- We fine-tune LLaMA 7B using our method and develop the pass key retrieval task to benchmark its performance on much larger context lengths.

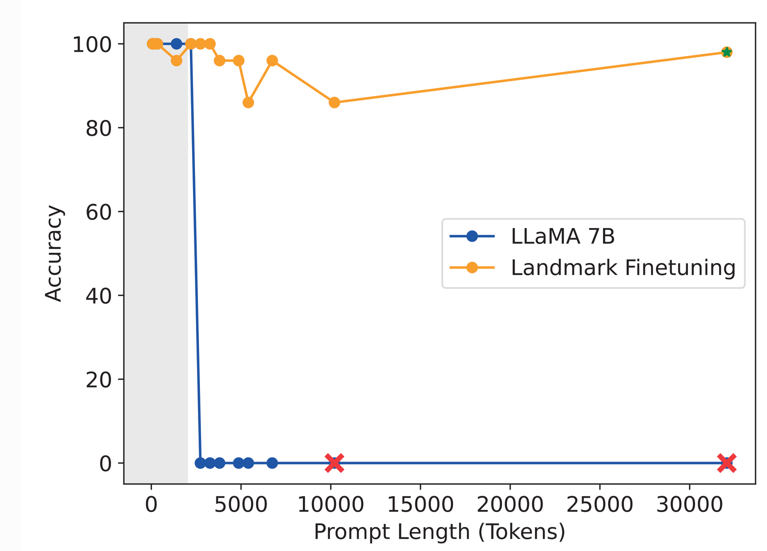
Pass Key Retrieval Task

- The task is to find a pass phrase hidden in a long piece of text.
- Instances of this task are generated by randomly choosing the pass key (a number between 1 and 50000) as well as its location.
- Prefix and suffix strings filled with a repetitive text are added to the pass key to ensure the desired length.
- We use the success rate as the evaluation metric.

```
There is an important info hidden inside
a lot of irrelevant text.  Find it and
memorize them.  I will quiz you about the
important information there.
<prefix filler by continuously repeating:
The grass is green.  The sky is blue.  The
sun is yellow.  Here we go.  There and back
again.>
The pass key is <PASS KEY>.  Remember it.
<PASS KEY> is the pass key.
<suffix filler>
What is the pass key?  The pass key is
```

Evaluation

- The original model fails to retrieve the pass key and even runs out of memory as the prompt gets longer (marked with a red cross)
- In contrast, the fine-tuned model using landmarks can successfully identify the pass key with **a high success rate for contexts with over 32k tokens.**
- The results demonstrate that our method can be successfully used even during fine-tuning to extend the context length limit to arbitrary large values.
- For very large context lengths, e.g. 32k, to avoid running out of memory, we use the capability offered by landmark attention allowing us to **offload the majority of the key-value cache to CPU** (a green star marks this mechanism's deployment).



Extensions

Retrieval Granularity

- Block retrieval can be performed on different levels of granularity.
- At the most granular level the set of retrieved blocks can be different for each head and each token.
- It is possible to further limit this granularity at inference, for increased system throughput.
- Experiments on PG19 show that reducing granularity hurts performance but most of the lost performance can be re-gained by retrieving more number of blocks.
- We use per-head retrieval to reduce the communication load when off-loading the key-value cache to CPU.

Per Head	Per Token	Eval. Length	k	Blocks	Perplexity
✓	✓	2048	2	$250\cdot 8\cdot 2$	15.14
		2048	4	$250\cdot 8\cdot 4$	14.92
		4096	4	$250\cdot 8\cdot 4$	14.72
✓	✗	2048	2	$8\cdot 2$	15.48
		2048	4	$8\cdot 4$	15.10
		4096	4	$8\cdot 4$	14.95
✗	✓	2048	2	$250\cdot 2$	15.44
		2048	4	$250\cdot 4$	15.04
		4096	4	$250\cdot 4$	14.89

Combination with Flash Attention

- Using Flash Attention's block size to that of Landmark Attention, the two can be naturally combined.
- We provide an open source implementation of this combination in Triton.

Context Miss Token

- Landmark Attention's grouping scheme can be adapted to introduce additional functionalities.
- For example, we show a Context Miss Token can be trained to signal a need for accessing memory.
- Experiments show that using this token around 50% of the retrievals can be dropped with minor effect on perplexity.

Cutoff	Perplexity	Drop Rate
Baseline	16.28	0%
0.0	16.38	0%
0.1	16.38	23%
0.3	16.43	57%
0.5	16.86	84%
1.0	19.49	100%

Extrapolating Positional Encoding

- Experiments on PG19 (without landmarks) show adding random index increases after each landmark token help generalization to unseen position indices but a fully extrapolating positional encoding is yet to be built.
- Such encoding removes the need for stingy position mapping but Landmark Attention is still needed to reduce the computation cost.