Low-Complexity Attentions for Transformer

郝永昌 July 9, 2020

Problems of Scaled Dot-Product

- $\tilde{X} = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad \tilde{X}, Q \in \mathbb{R}^{N \times E}, \; K, V \in \mathbb{R}^{M \times E}$
- $X \in \mathbb{R}^{N \times E}$ refer to a sentence with N tokens and embedding size of E
- Complexity for each sublayer:
  - Attention: $O(N^2E) + O(NE^2)$
  - FFN: $O(NE^2)$
  - ReLU / LayerNorm / Dropout: $O(NE)$

Summary

Model Name	Paper Title	Venue	Affl.	Complexity
Sparse Transformer	Generating Long Sequences with Sparse Transformers	arXiv 2019	OpenAI	$O(EN\sqrt{N})$
Reformer	Reformer: The Efficient Transformer	ICLR 2020	Berkeley / Google	$O(EN \log N)$
Linformer	Linformer: Self-Attention with Linear Complexity	arXiv 2020	Facebook	$O(NEM)$
Linear-Transformer	Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention	ICML 2020	IDIAP / EPFL	$O(NE^2)$

Sparse Transformer

- ***Generating Long Sequences with Sparse Transformers,*** Rewon Child, Scott Gray, Alec Radford, Ilya Sutskever (arXiv 2019)

Sparse Transformer

Main idea

- For any position i (i.e. the i-th row in attention), construct 2 index sets A_i, B_i so that

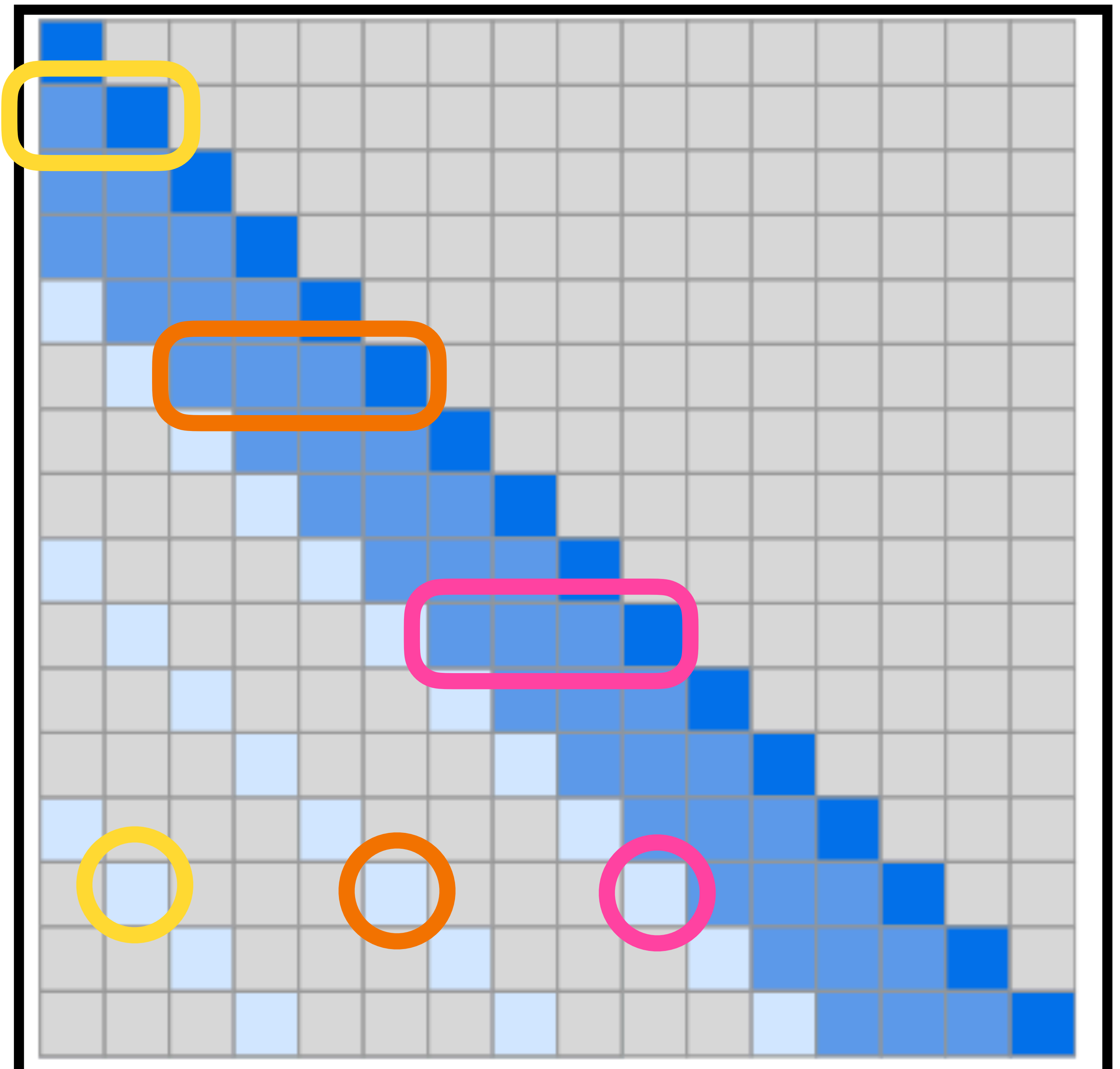
    1. $|A_i| \approx \sqrt{N} \quad |B_i| \approx \sqrt{N}$ sparsity of attention matrix

    2. For any position $j \leq i$, there exists a path (j, k, i) that $j \in A_k, k \in B_i$

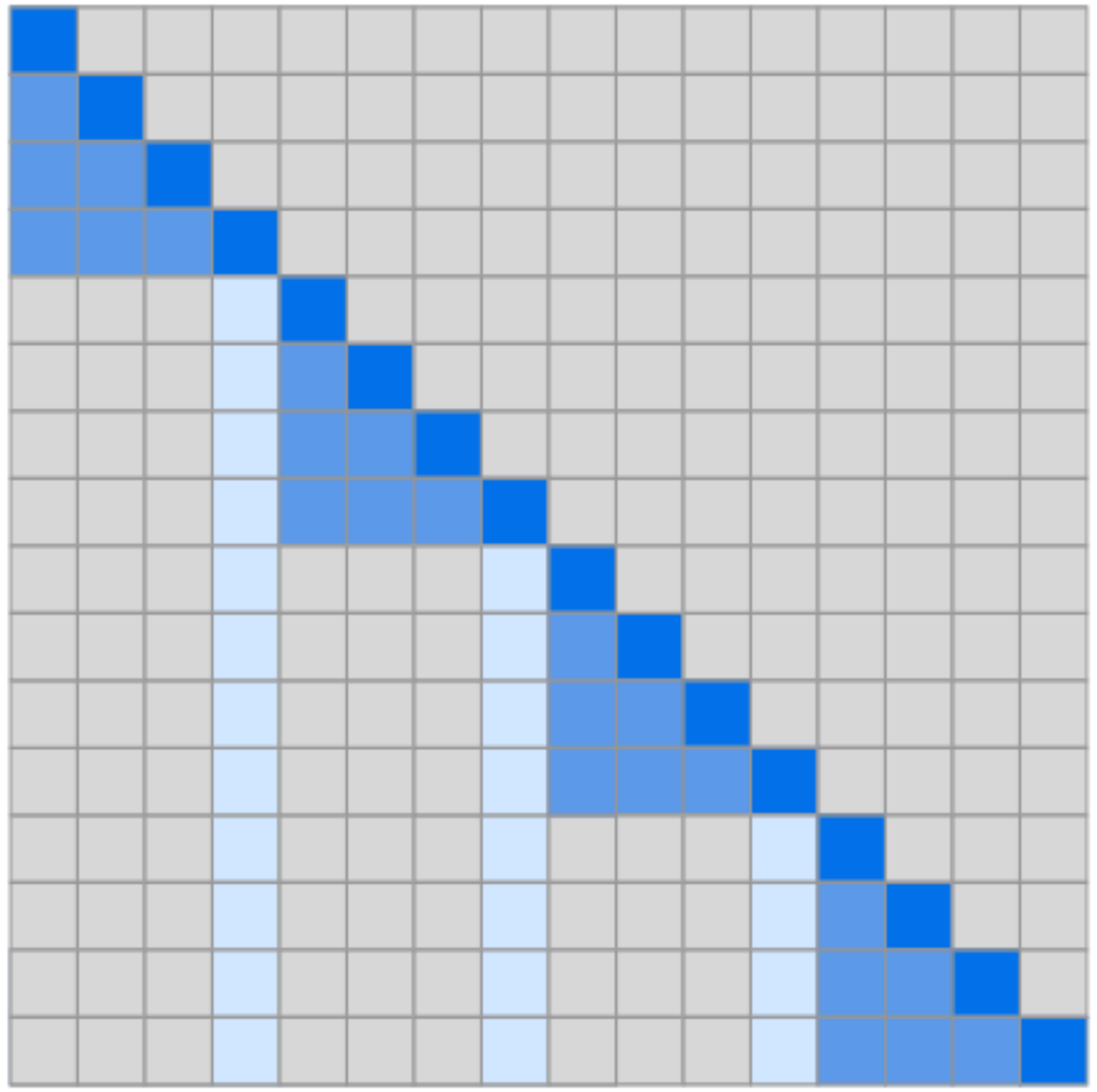
i could attend to any $j \leq i$
within 2 attention steps

Sparse Transformer

Two patterns manually designed by the authors



(b) Sparse Transformer (strided)



(c) Sparse Transformer (fixed)

Sparse Transformer

Contributions & Limitations

- Contribution:

    1. Speedup attention layers from a complexity of $O(EN^2)$ to $O(EN\sqrt{N})$

- Limitations：

    1. Can only be used as single direction attention: can’t directly apply to encoder self-attention and decoder cross-attention.

    2. Hard to implement sparse attention on GPU.

Reformer

- ***Reformer: The Efficient Transformer,*** Nikita Kitaev, Łukasz Kaiser, Anselm Levskaya (ICLR 2020)

Reformer

Main idea

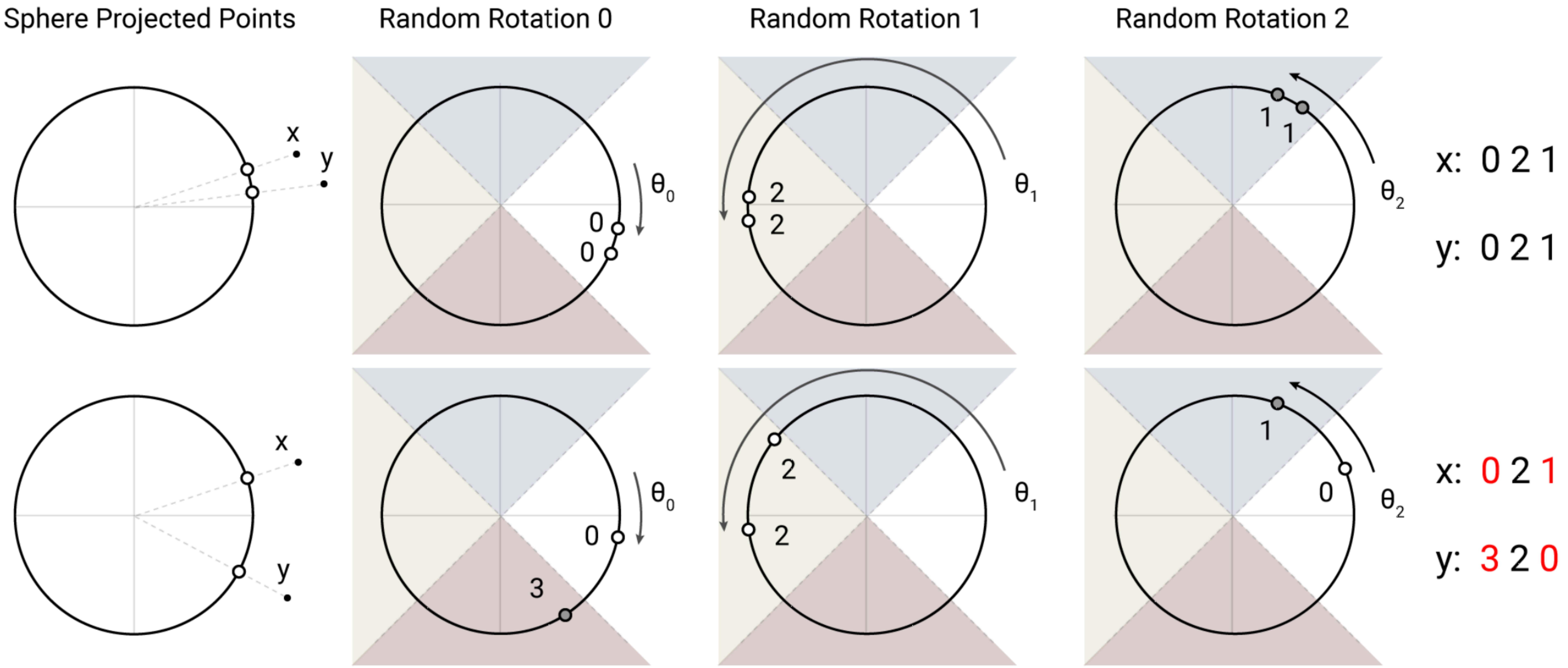
- Large elements in $\mathrm{softmax}(QK^T/d)$ come from similar Q_i, K_j
- Find nearest top-α K_j for any Q_i, ignore other K s.
    1. Sort by Locality-Sensitive Hashing (LSH)
    2. Cut into N/α chunks
    3. For Q_i in chunk c, do dot-product with Ks in chunk c and $c-1$.
- Time complexity： $O\left(EN(\log N+\alpha)\right)$

Reformer

A toy visualization of LSH

LSH function: $h(X) = \underset{i}{\operatorname{argmax}}(XR)^{(i)}$

$$h(X) = h(Y) \iff X \simeq Y$$



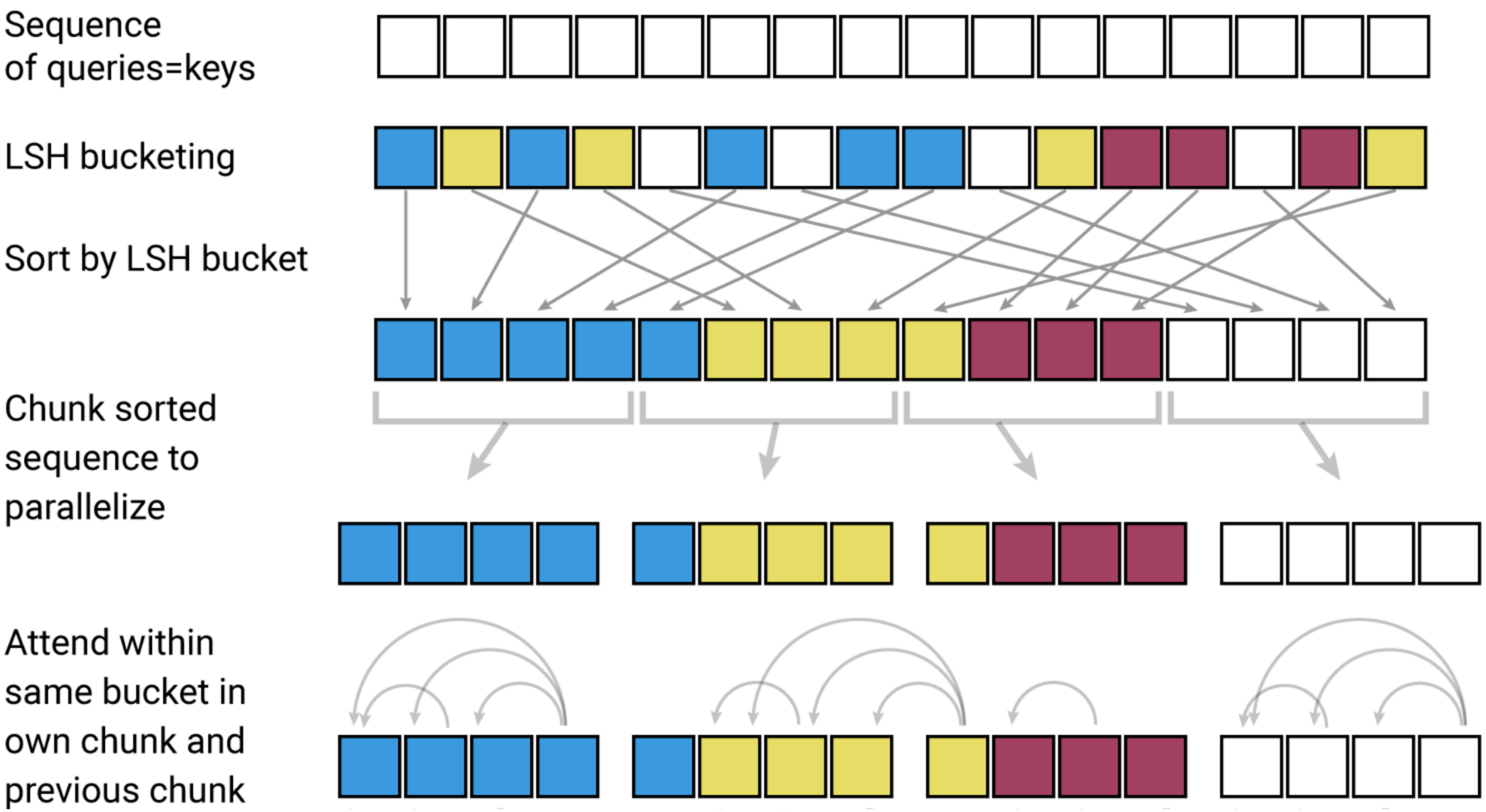
Reformer

Main idea

- Large elements in $\mathrm{softmax}(QK^T/d)$ come from similar Q_i, K_j
- Find nearest top-α K_j for any Q_i, ignore other K s.
    1. Sort by Locality-Sensitive Hashing (LSH)
    2. Cut into N/α chunks
    3. For Q_i in chunk c, do dot-product with Ks in chunk c and $c-1$.
- Time complexity：$O\left(EN\log N\right)$

Reformer

Main idea



Reformer

Details (free to skip)

1. Trick: $h(X) = \mathrm{sign}(XR) \in \{0,1\}^{\log \frac{N}{\alpha}}$

2. Use $h(X)$ to project X into an integer from 0 to $\frac{N}{\alpha} - 1$ in $O(EN \log N)$ time.

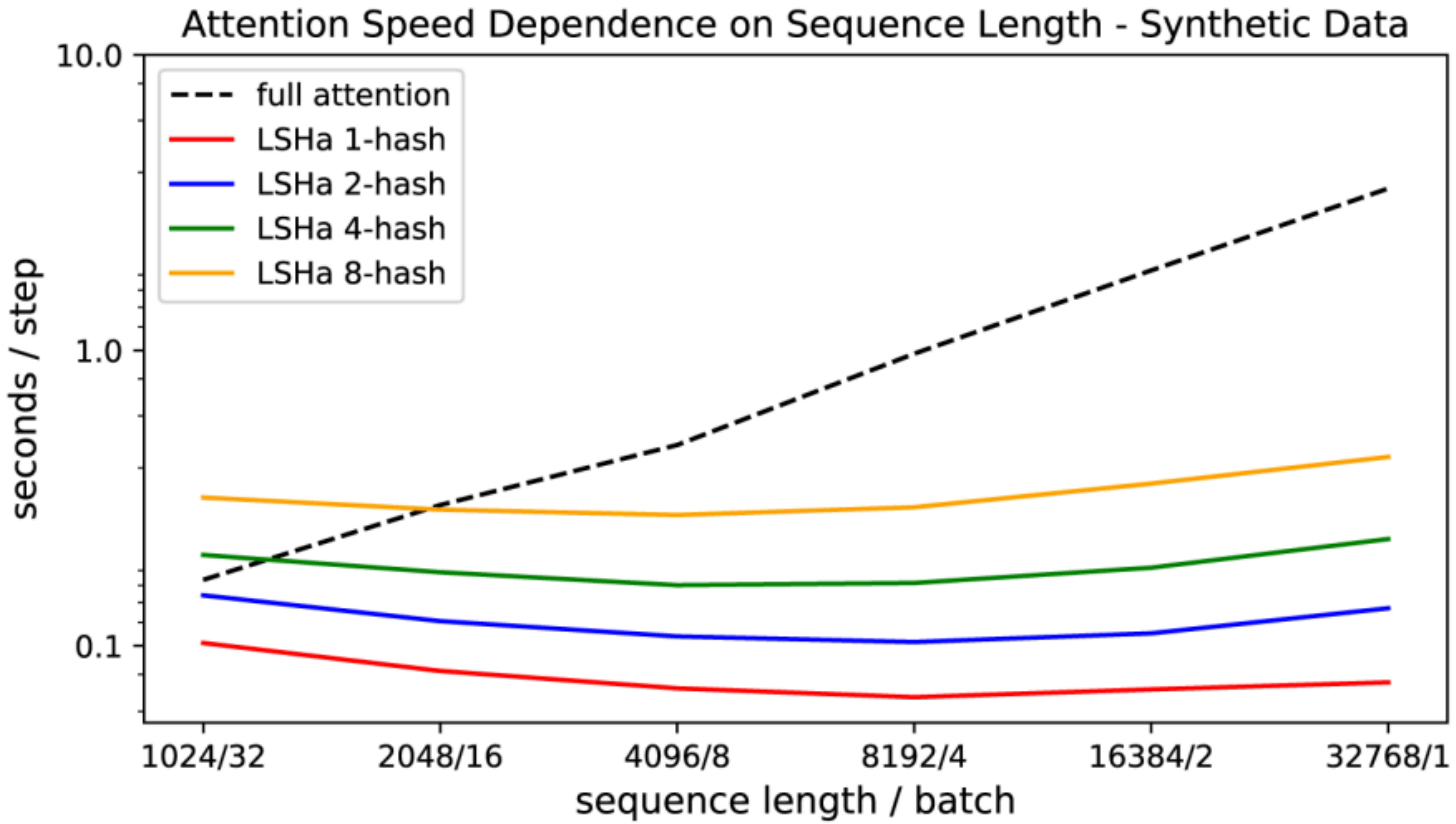
3. Sort tokens by $h(X)$ in $O(N \log N)$ time.

4. Do dot-product attention for each Q_i with 2α Ks in the previous and the current chunk. The time complexity is $O(EN\alpha)$.

Total: $O\left(EN \log N\right)$

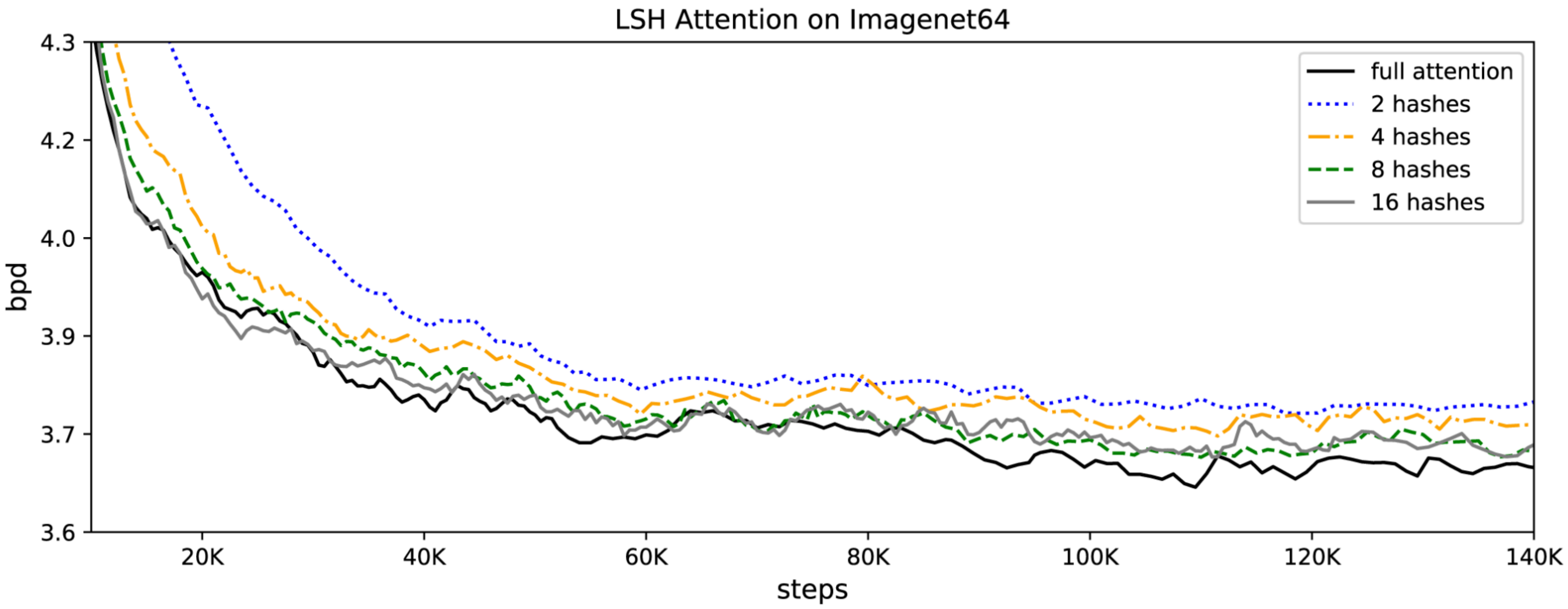
Reformer

Experiments - Speed



Reformer

Experiments - Image generating



Reformer

Contributions & Limitations

- Contributions:
    1. Attention in $O(EN \log N)$ time.
    2. Reversible Transformer (not related to our topic though)
- Limitations：
    1. Hard to implement
    2. Large constant in the time complexity: multi-round hashing, sorting, unordered causal masking,, etc…

Linformer

- ***Linformer: Self-Attention with Linear Complexity,*** Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, Hao Ma (arXiv 2020)

Linformer

Main idea

- Replace N Keys and Values with constant M Keys and Values.
- Directly apply linear transformations $E, F \in \mathbb{R}^{M \times N}$ to $K, V \in \mathbb{R}^{N \times E}$ to get $\tilde{K} = EK \in \mathbb{R}^{M \times E}, \tilde{V} = FV \in \mathbb{R}^{M \times E}$.
- Then $\mathrm{softmax}\left(Q\tilde{K}^T/\sqrt{d_k}\right)\tilde{V}$
- Time Complexity： $O(ENM + EM^2)$

Linformer

Experiments - NLU tasks

n	Model	SST-2	IMDB	QNLI	QQP	Average
512	Liu et al. (2019), RoBERTa-base	93.1	94.1	90.9	**90.9**	92.25
	Linformer, 128	92.4	94.0	90.4	90.2	91.75
	Linformer, 128, shared kv	**93.4**	93.4	90.3	90.3	91.85
	Linformer, 128, shared kv, layer	93.2	93.8	90.1	90.2	91.83
	Linformer, 256	93.2	94.0	90.6	90.5	92.08
	Linformer, 256, shared kv	93.3	93.6	90.6	90.6	92.03
	Linformer, 256, shared kv, layer	93.1	94.1	**91.2**	90.8	**92.30**
512	Devlin et al. (2019), BERT-base	92.7	93.5	91.8	89.6	91.90
	Sanh et al. (2019), Distilled BERT	91.3	92.8	89.2	88.5	90.45
1024	Linformer, 256	93.0	93.8	90.4	90.4	91.90
	Linformer, 256, shared kv	93.0	93.6	90.3	90.4	91.83
	Linformer, 256, shared kv, layer	93.2	**94.2**	90.8	90.5	92.18

Linformer

Experiments - Speed

NLU tasks reported

length n	projected dimensions k 128	256	512	1024	2048
512	1.5x	1.3x	-	-	-
1024	1.7x	1.6x	1.3x	-	-
2048	2.6x	2.4x	2.1x	1.3x	-
4096	3.4x	3.2x	2.8x	2.2x	1.3x
8192	5.5x	5.0x	4.4x	3.5x	2.1x
16384	8.6x	7.8x	7.0x	5.6x	3.3x
32768	13x	12x	11x	8.8x	5.0x
65536	20x	18x	16x	14x	7.9x

Linformer

Opinions

- Potential problems:

  1. E, F projections limit the max-length of the model.

  2. If K is a constant, then Linformer would lose information when N keep growing.

  3. On the other hand, if K is proportional to N, then the complexity will be deteriorated from $O(KNE + K^2E)$ to $O(N^2E)$,

Linear-Transformer

- ***Transformer Dissection: A Unified Understanding of Transformer's Attention via the Lens of Kernel,*** Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, Ruslan Salakhutdinov (EMNLP 2019)
- ***Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention,*** Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, Franc¸ois Fleuret (ICML 2020)

Linear-Transformer

Main idea

- Size of QK^TV: $(N \times E)(E \times N)(N \times E)$
- Different multiplication orders have different complexity:
  1. $\big((N \times E)(E \times N)\big) \cdot (N \times E)$: $O(N^2E)$
  2. $(N \times E) \cdot \big((E \times N)(N \times E)\big)$: $O(NE^2)$
- Sadly: $\mathrm{softmax}\left(\frac{QK^T}{d_k}\right)V$ determines the order.

Linear-Transformer

Main idea

- Goal: to find a $\mathrm{magic}(\,\cdot\,)$ to perform:

$$\mathrm{magic}\left(\frac{QK^T}{d_k}\right)V = f(Q)\big(g(K)\big)^T V$$

$$= f(Q)\Big(\big(g(K)\big)^T V\Big)$$

Linear-Transformer

Introducing Kernel

- **Kernel** is a function $\kappa : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ that there implicitly exists a corresponding feature map function $\phi : \mathscr{X} \to \mathbb{R}^d$ satisfying $\kappa(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{y})$.

- Intuitive property: Kernel can measure similarity between two high-dimensional vectors.

- **Theorem: dot-product attention can be viewed as a kernel smoother.**

$$\mathrm{y} = \frac{\sum_j \kappa\left(\mathrm{x}, X^{(j)}\right) Y^{(j)}}{\sum_j \kappa\left(\mathrm{x}, X^{(j)}\right)} \qquad X \in \mathbb{R}^{n \times d} \quad \mathrm{x}, \mathrm{y} \in \mathbb{R}^d$$

Linear-Transformer

Reformation of attention (free to skip)

- $f_{attn}(X_q, X_k, M) = \text{softmax}\left(\frac{X_q W_q (X_k W_k)^T}{\sqrt{d_k}} \oplus M\right) X_k W_v$
- encoder self-attention: $X_q = X_k,\ M_{i,j} = 0$
- decoder self-attention: $X_q = X_k,\ M_{i,j} = \begin{cases} 0 & j \leq i \\ -\infty & j > i \end{cases}$
- decoder cross-attention: $X_q \neq X_k,\ M_{i,j} = 0$

Linear-Transformer

Reformation of attention (free to skip)

- $f_{attn}(X_q, X_k, M) = \text{softmax}\left(\frac{X_q W_q (X_k W_k)^T}{\sqrt{d_k}} \oplus M\right) X_k W_v$

Raw format

- $\left[f_{attn}(X_q^{(i)}, X_k, \tilde{M}_i)\right]_i = \sum_{j \in \tilde{M}_i} \frac{\exp\left(\frac{(X_q^{(i)} W_q) \cdot (X_k^{(j)} W_k)}{\sqrt{d_k}}\right)}{\sum_{r \in \tilde{M}_i} \exp\left(\frac{(X_q^{(i)} W_q) \cdot (X_k^{(r)} W_k)}{\sqrt{d_k}}\right)} (X_k^{(j)} W_v)$

consider the i-th row,

M_i is the index set that i can attend to.

Linear-Transformer

Reformation of attention (free to skip)

- $$\left[f_{attn}(X_q^{(i)}, X_k, \tilde{M}_i)\right]_i = \frac{\sum_{j \in \tilde{M}_i} \exp\left(\frac{(X_q^{(i)}W_q) \cdot (X_k^{(j)}W_k)}{\sqrt{d_k}}\right)(X_k^{(j)}W_v)}{\sum_{j \in \tilde{M}_i} \exp\left(\frac{(X_q^{(i)}W_q) \cdot (X_k^{(j)}W_k)}{\sqrt{d_k}}\right)}$$

Rewrite

$$v(x) \triangleq xW_v$$

$$\kappa(x, y) \triangleq \exp\left(\langle xW_q, yW_k \rangle / \sqrt{d_k}\right)$$

- $$\left[f_{attn}(X_q^{(i)}, X_k, \tilde{M}_i)\right]_i = \frac{\sum_{j \in \tilde{M}_i} \kappa(X_q^{(i)}, X_k^{(j)}) v(X_k^{(j)})}{\sum_{j \in \tilde{M}_i} \kappa(X_q^{(i)}, X_k^{(j)})}$$

- $$\tilde{X}_i = \frac{\sum_j \kappa(X_i, X_j)\ v(X_j)}{\sum_j \kappa(X_i, X_j)}$$

Simplify

Linear-Transformer

FYI

- $\tilde{X}_i = \dfrac{\sum_{j \le M(i)} \kappa(X_i, X_j)\ v(X_j)}{\sum_{j \le M(i)} \kappa(X_i, X_j)}$

- Changing kernel $\kappa = \kappa_F(f_q, f_k) \cdot \kappa_T(t_q, t_k)$ here may improve BLEU:

Approach	PE Incorporation	Kernel Form	NMT (BLEU↑)
Vaswani et al. (2017) (Eq. (4))	Direct-Sum	$k_{\exp}\left(f_q + t_q, f_k + t_k\right)$	33.98
Shaw et al. (2018) (Eq. (6))	Look-up Table	$L_{t_q - t_k, f_q} \cdot k_{\exp}\left(f_q, f_k\right)$	34.12
Dai et al. (2019) (Eq. (5))	Product Kernel	$k_{\exp}\left(f_q, f_k\right) \cdot k_{f_q}\left(t_q, t_k\right)$	33.62
Ours (Eq. (9))	Product Kernel	$k_F\left(f_q, f_k\right) \cdot k_T\left(t_q, t_k\right)$	**34.71**

Linear-Transformer

Reformation of attention (free to skip)

- $\tilde{X}_i = \dfrac{\sum_{j \le M(i)} \kappa(X_i, X_j)\ v(X_j)}{\sum_{j \le M(i)} \kappa(X_i, X_j)}$
- According the definition of kernel, we have

$$\tilde{X}_i = \frac{\sum_{j \le M(i)} \phi(X_i)\left(\phi(X_j)\right)^T v(X_j)}{\sum_{j \le M(i)} \phi(X_i)\left(\phi(X_j)\right)^T} = \frac{\phi(X_i)\sum_{j \le M(i)}\left(\phi(X_j)\right)^T v(X_j)}{\phi(X_i)\sum_{j \le M(i)}\left(\phi(X_j)\right)^T}$$

Linear-Transformer

Complexity Analysis - w/o causal mask

- $$\tilde{X}_i = \frac{\phi(X_i) \sum_{j \le N} \left(\phi(X_j)\right)^T v(X_j)}{\phi(X_i) \sum_{j \le N} \left(\phi(X_j)\right)^T}$$
- Preprocess $S = \sum_{j \le N} \left(\phi(X_j)\right)^T v(X_j)$ and $Z = \sum_{j \le N} \left(\phi(X_j)\right)^T$ in $O(NE^2)$.
- For every i, do $\tilde{X}_i = \frac{\phi(X_i) \cdot S}{Z}$ in $O(E^2)$, and repeat for N times.

Total: $O\left(NE^2\right)$

Linear-Transformer

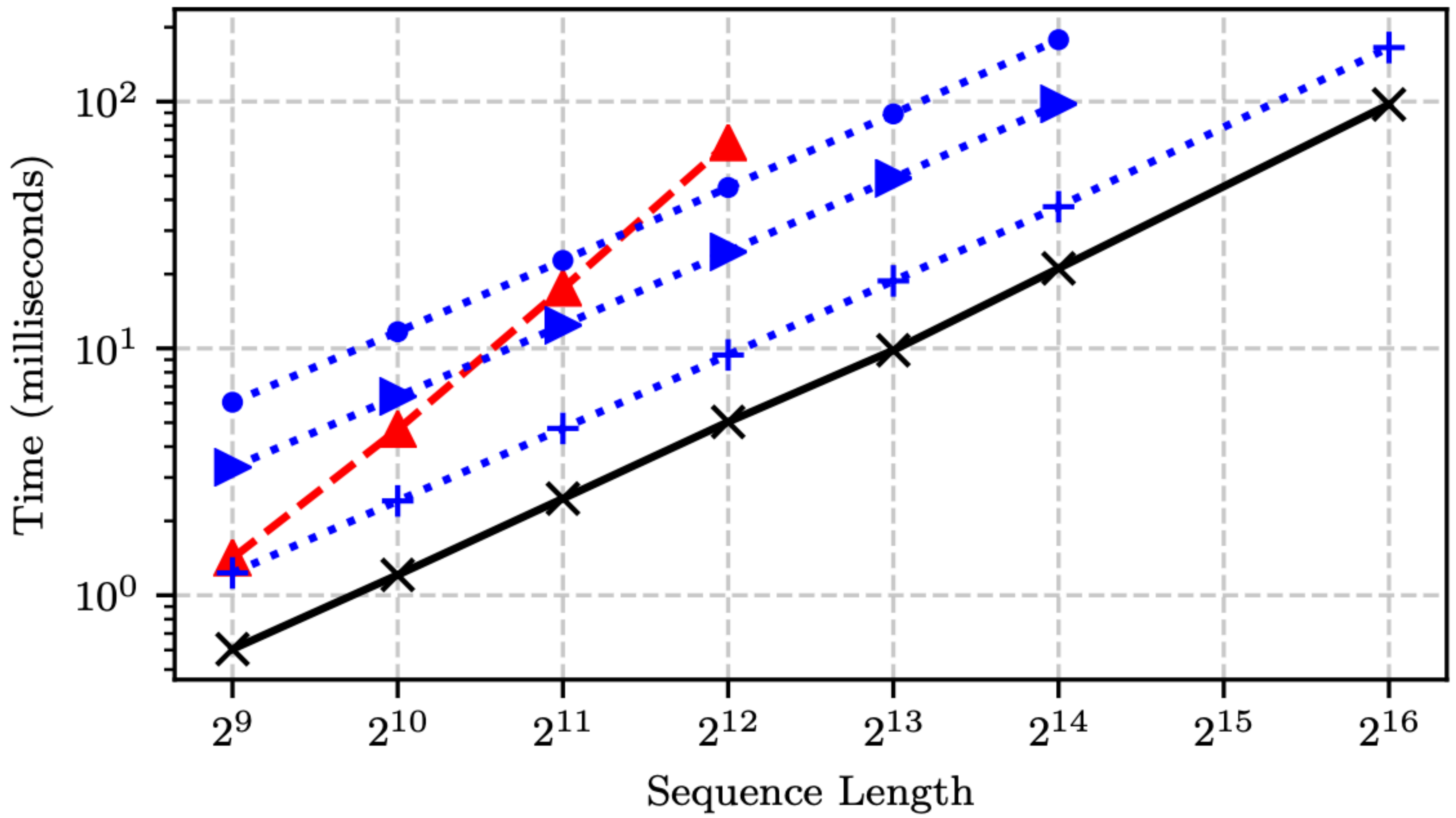
Complexity Analysis - w/ causal mask

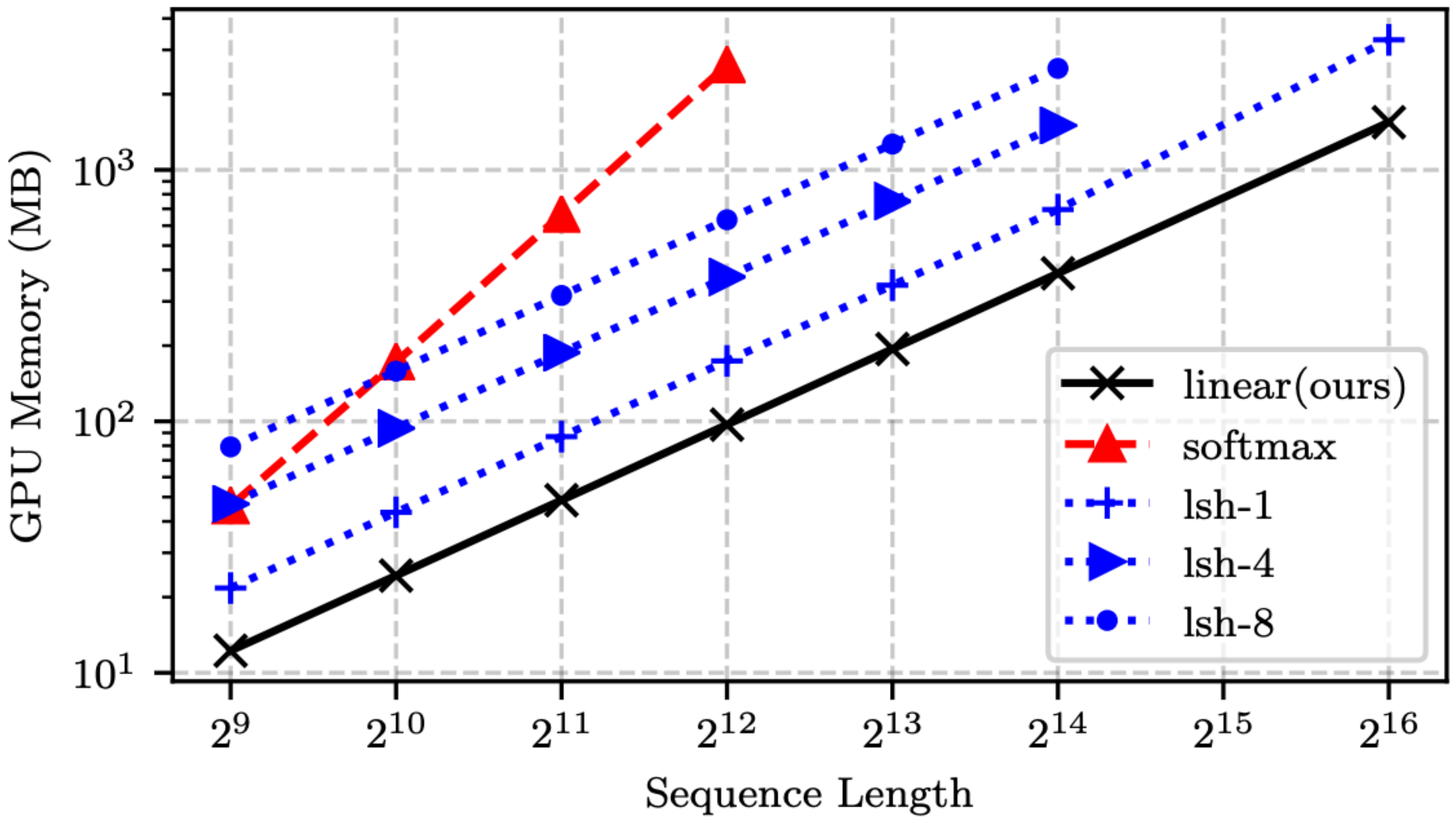
- $$\tilde{X}_i = \frac{\phi(X_i) \sum_{j \le M(i)} \left(\phi(X_j)\right)^T v(X_j)}{\phi(X_i) \sum_{j \le M(i)} \left(\phi(X_j)\right)^T}$$
- Initialize with $S_0 = Z_0 = \mathbf{0}$
- For every i do $S_i = S_{i-1} + \left(\phi(X_i)\right)^T v(X_i),\ \ Z_i = Z_{i-1} + \left(\phi(X_i)\right)^T,\ \ \tilde{X}_i = \frac{\phi(X_i) \cdot S_i}{Z_i}$ in $O(E^2)$, and repeat for N times.

Total: $O\left(NE^2\right)$

Linear-Transformer

Experiments - Speed and Memory use





Linear-Transformer

Experiments - Accuracy and Speed

Method	Bits/dim	Images/sec	
Softmax	0.621	0.45	(1×)
LSH-1	0.745	0.68	(1.5×)
LSH-4	0.676	0.27	(0.6×)
Linear (ours)	0.644	**142.8**	**(317×)**

Generating MNIST images

Method	Bits/dim	Images/sec	
Softmax	3.47	0.004	(1×)
LSH-1	3.39	0.015	(3.75×)
LSH-4	3.51	0.005	(1.25×)
Linear (ours)	3.40	**17.85**	**(4,462×)**

Generating CIFAR-10 images

Summary

Model Name	Paper Title	Venue	Affl.	Complexity
Sparse Transformer	Generating Long Sequences with Sparse Transformers	arXiv 2019	OpenAI	$O(EN\sqrt{N})$
Reformer	Reformer: The Efficient Transformer	ICLR 2020	Berkeley / Google	$O(EN \log N)$
Linformer	Linformer: Self-Attention with Linear Complexity	arXiv 2020	Facebook	$O(NEM)$
Linear-Transformer	Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention	ICML 2020	IDIAP / EPFL	$O(NE^2)$