Reformer : The efficient transformer

Author : Nikita Kitaev, Lukaz Kaiser, Anselm Levskaya
Institue : Google Research
Paper : https://arxiv.org/abs/2001.04451
github : https://github.com/lucidrains/reformer-pytorch
Arix uploaded date : 18 Feb 2020 / ICLR 2020

Contents

Abstract

Reformer : The efficient transformer

-problem

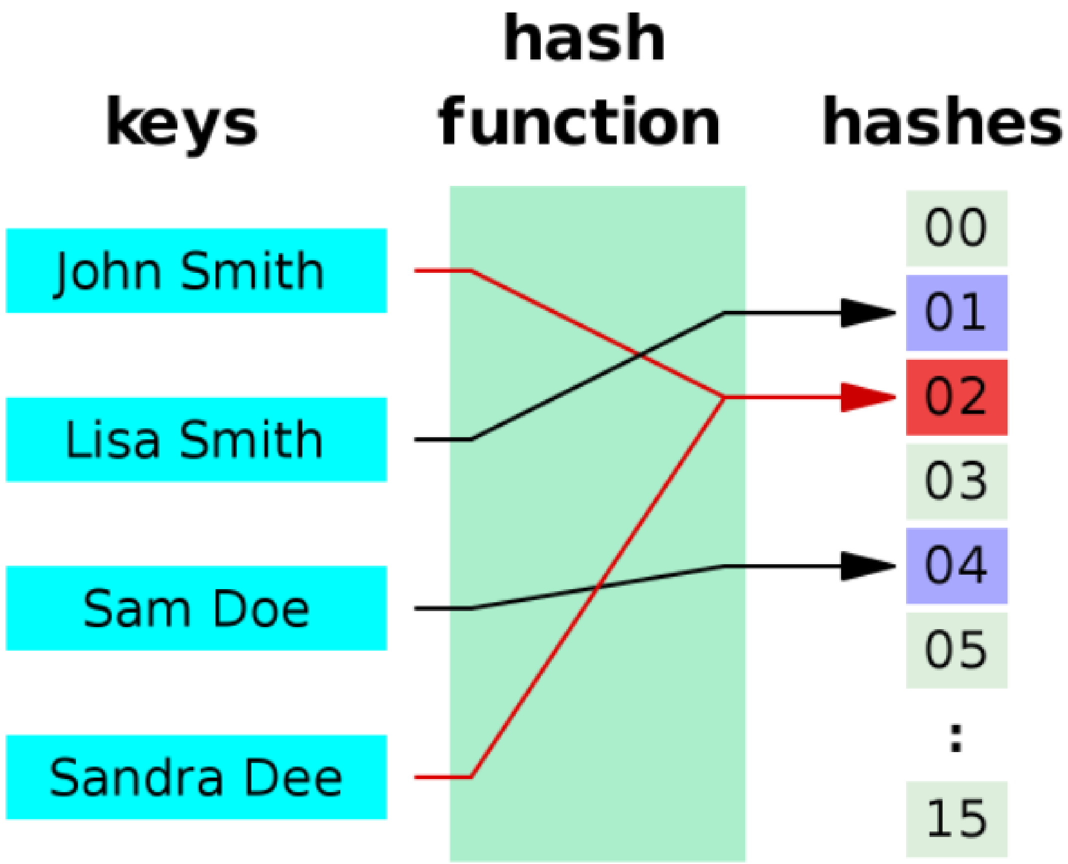
- Transformer attention structure $O(n^2)$ complexity problem
- Feed Foward Layer memory problem -> all outputs of attention layer applied (512 inside -2048)
- N-stacked Residual connection memory problem -> gradient check point

-Contribution

- Attend on similar pair
- Point out that 1) high-impact word pairs are similar to each other in the embedding space, and 2) these pairs can be quickly found using Locality-Sensitive Hashing (LSH).
- Chunking data point
- Data point chunking can reduce memory regardless feed-forward layer position
- Reversible layer
- Residual connection can be converted as reversible (Back propagation, recover weight)

Teoretical background

- Locality-Sensitive Hahsing



가까운 값들끼리 가까운 Hash값을 가지도록 Hashing하는 방법
-> Locality-Sensitive Hashing

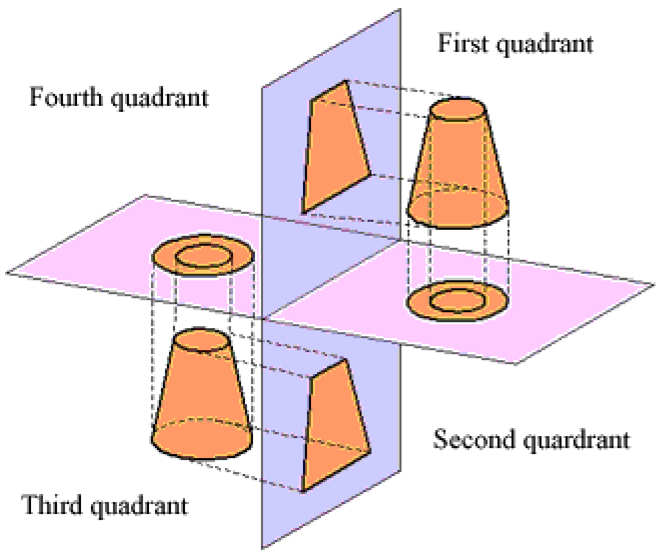
- An operation that compares actual data values is absolutely necessary.
- The comparison operation can be approximated as an operation on the hash value.

Teoretical background

- Locality-Sensitive Hahsing

  - How to machine can make locality-sensitive hash?

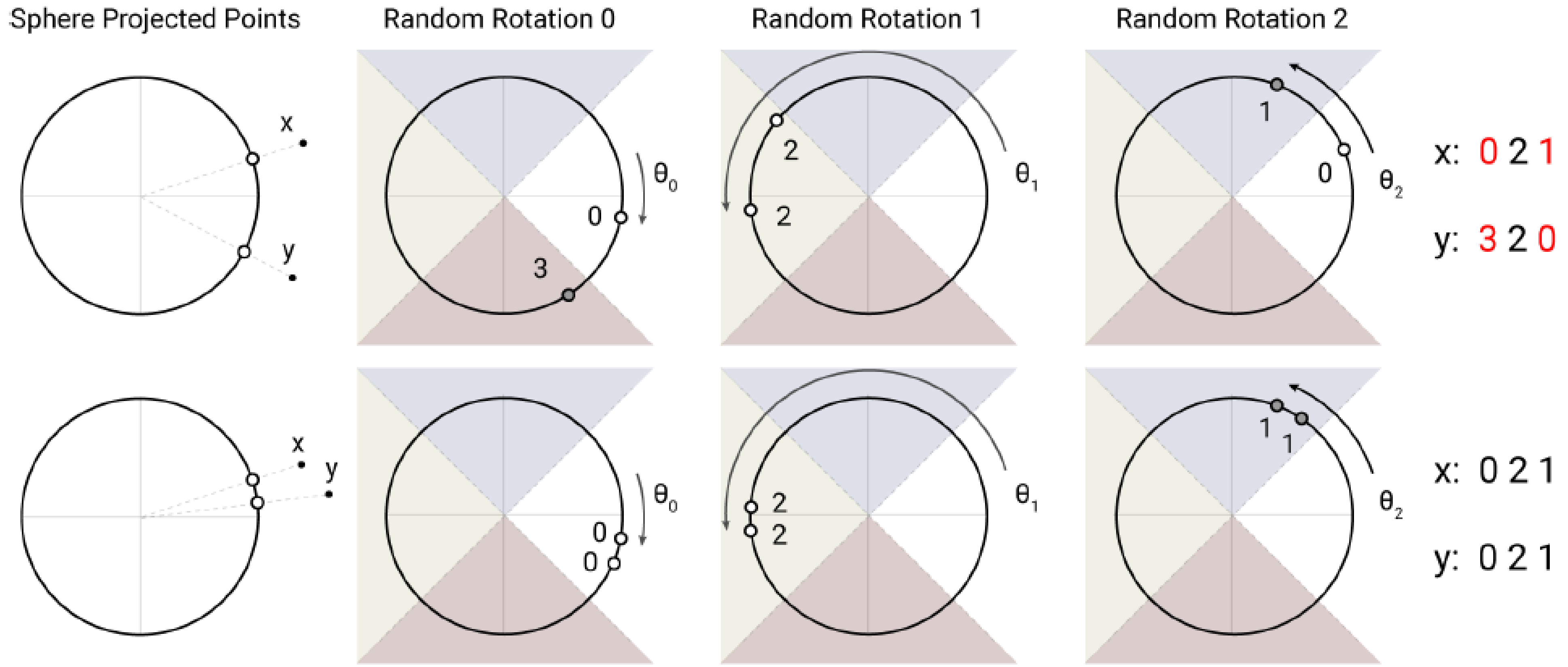
  - Projection method



- Shared plane -> close value   (+ - sign)

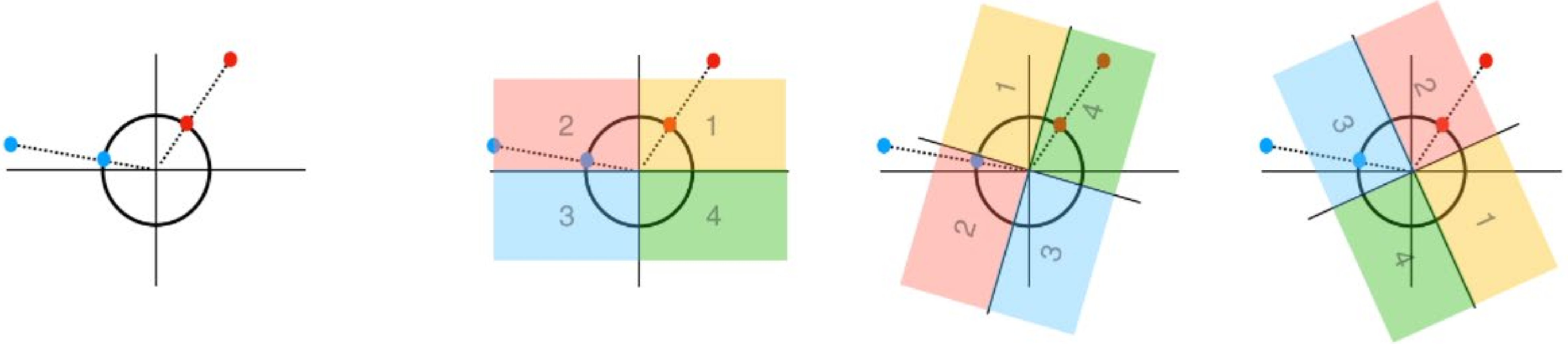
Teoretical background

- Locality-Sensitive Hahsing
  - Angular LSH

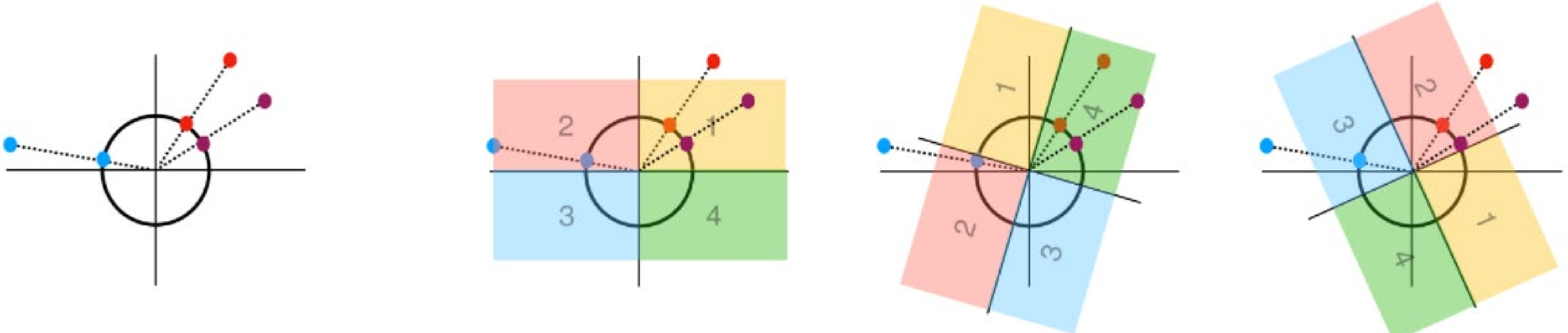


Teoretical background

- Locality-Sensitive Hahsing



- X1 = (3,4) X2= (-12,5) X1’ = (3/5, 4/5) X2’ = (-12/13, 5/13)   Hash X1 = (1, 4, 2) Hash X2 = (2, 2, 3)



- Y1 = (4,3)   Y’ = (4/5, 3/5) Hash Y1 = (1,4,2)   Hash X1 = Hash Y1

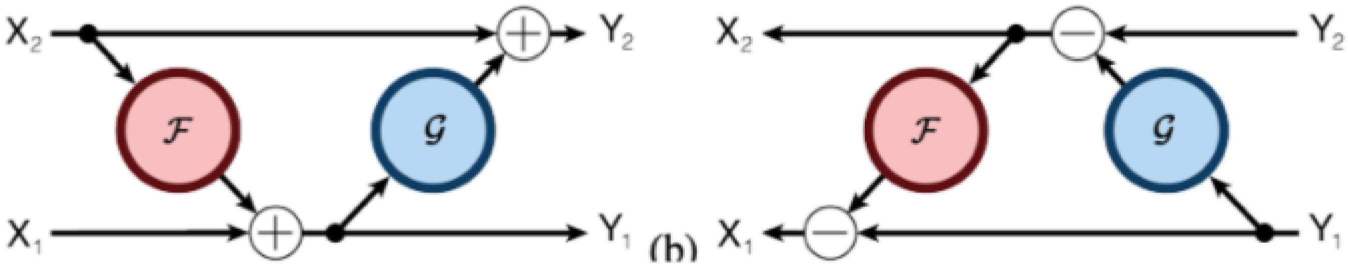
Teoretical background

- Reversible Network
  - Residual connection (Resnet or Transformer)

$y = x + F(x)$ x -> y possible but y -> x

x = (x1,x2) y = (y1,y2) pair representation

$$y_1 = x_1 + F(x_2), y_2 = x_2 + G(y_1)$$



$$x_2 = y_2 - G(y_1)$$
$$x_1 = y_1 - F(x_2)$$

-> reversible calculation !

Model Architecture

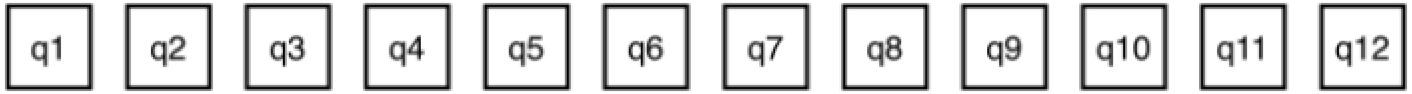
- Assumption

  Query (Q) = Key(K)

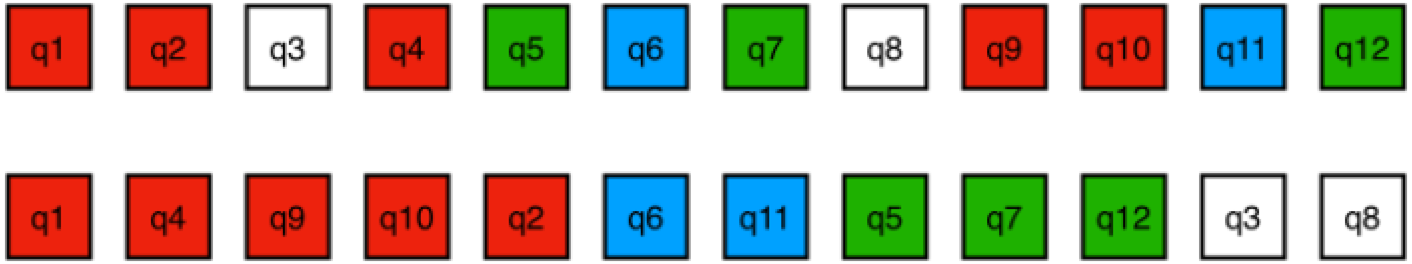
  -If dataset is enough, not decrease performance of transformer

- LSH Attentin to Transformer

  - Q = K (each data query)

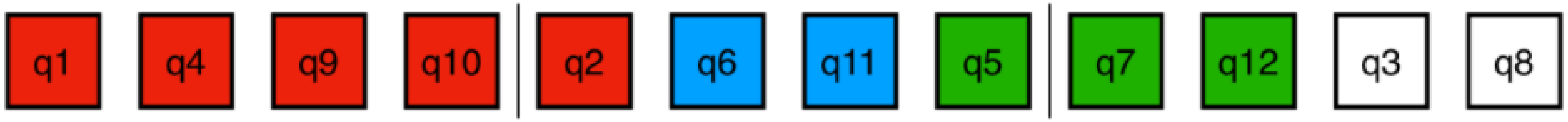


  - LSH applied on each data point & Bucketing



Model Architecture

- Chunking

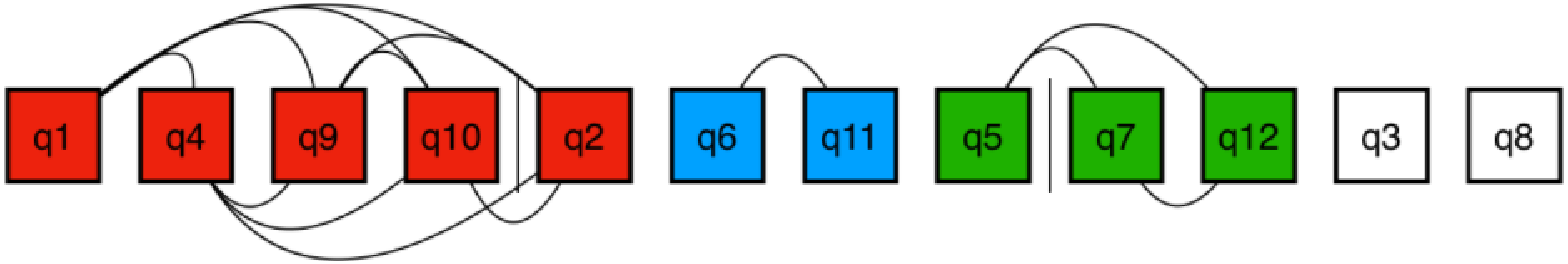


- Attention weight calculation

1. Two data point should be in same bucket (same color)

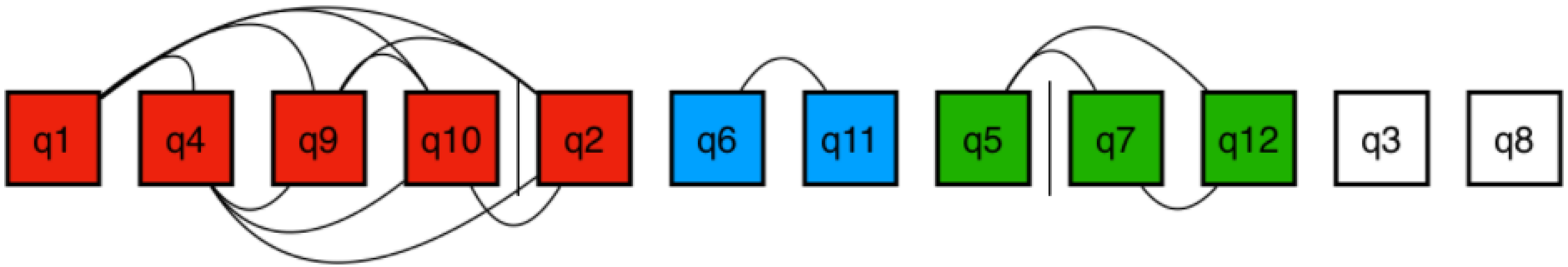
2. Two data point should be in same chunck or end point attention should be the next chuck of start point attention

3. Q=K -> Self-attention always large weight -> not allow self-attention



Model Architecture

- Linear complexity



data point length : l
chunk number : c
chunked part length : l / c
Attentio number is portion of l x $(2l / c)^2$

if c is large then, approximation to linear complexity

Feed Foward Network with Reversible Network

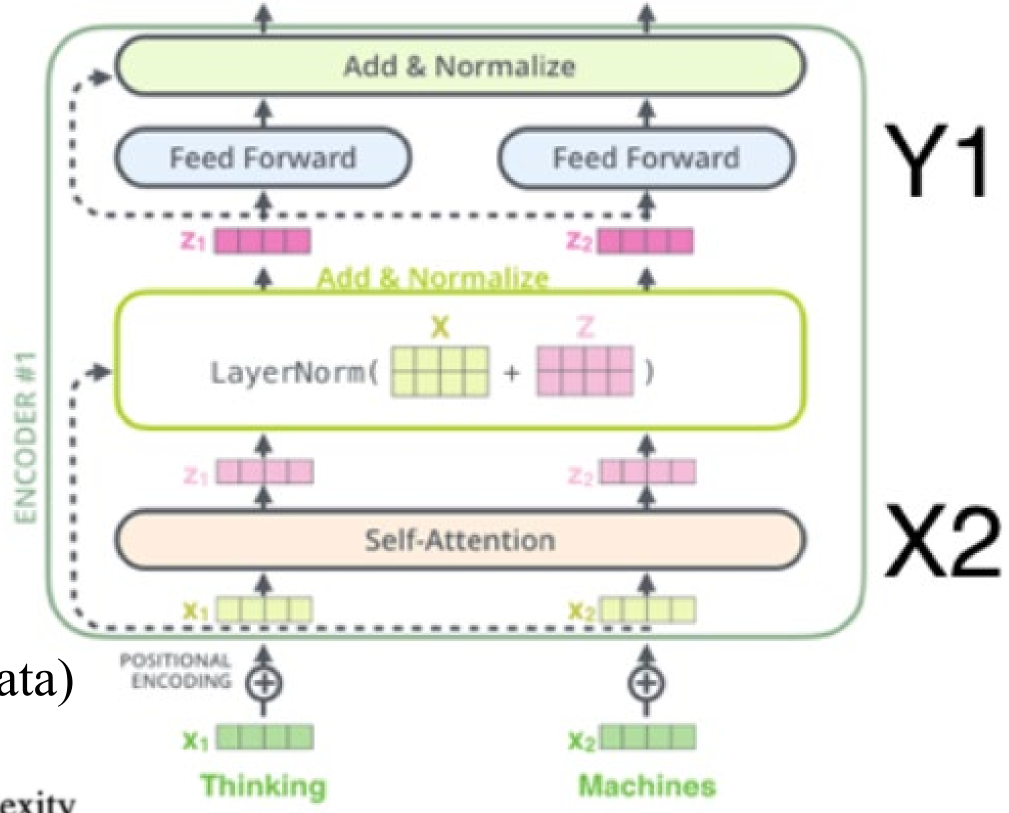
- Memory reduce from reversible network

$$y_1 = x_1 + F(x_2), y_2 = x_2 + G(y_1)$$

$$y_1 = x_1 + Attention(x_2)$$

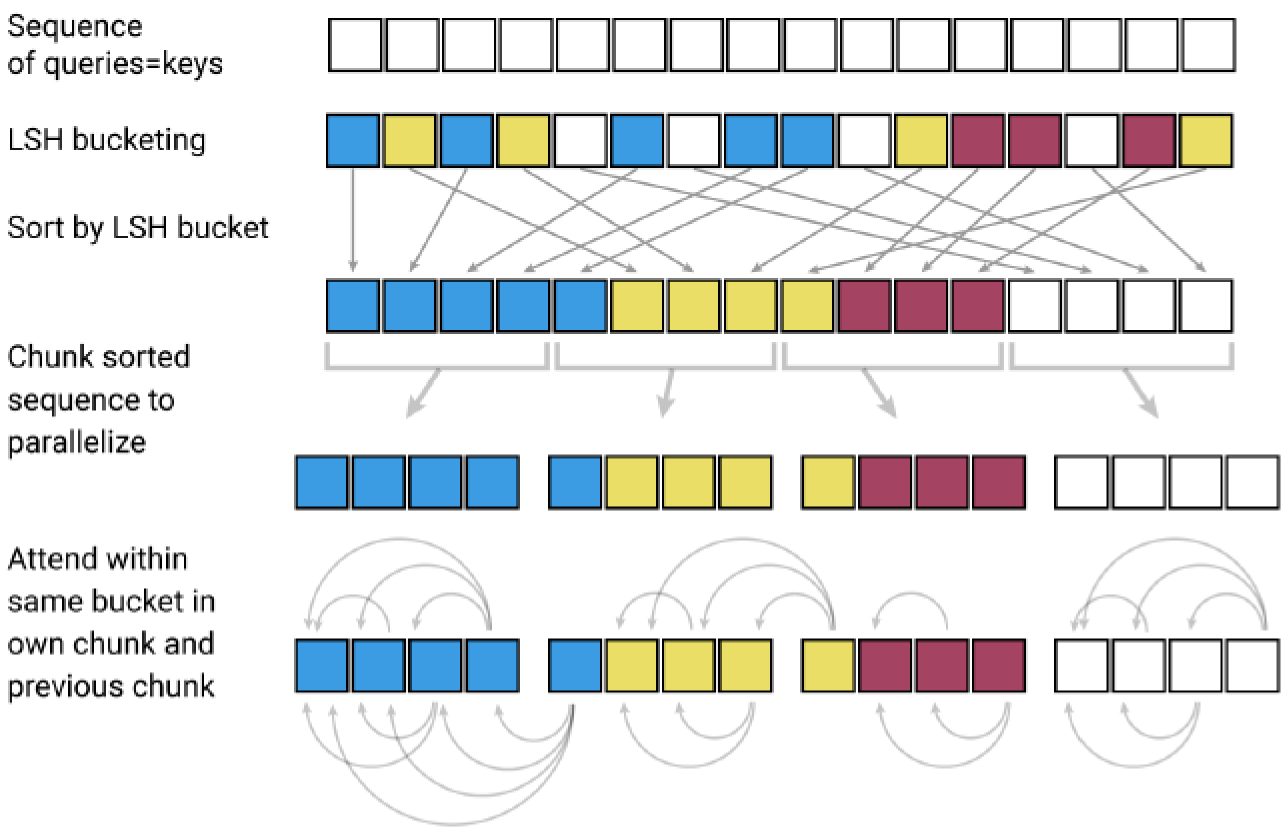
$$y_2 = x_2 + FeedForward(y_1)$$

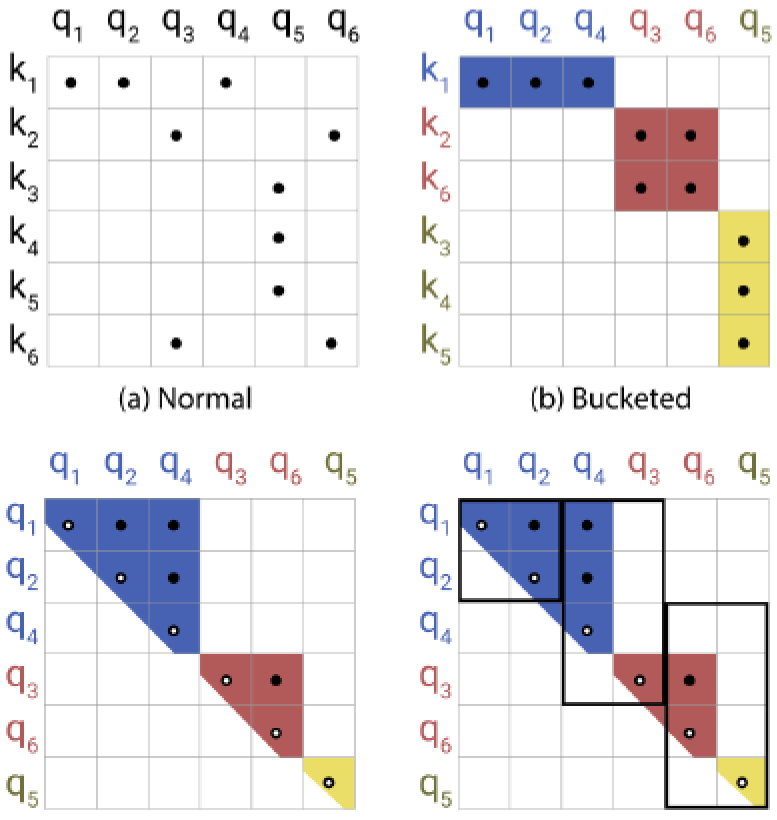
- 64K sequence -> 16K chunk ! (each chunck has 4 data)



Model Type	Memory Complexity	Time Complexity
Transformer	$\max(bld_{ff}, bn_h l^2)n_l$	$(bld_{ff} + bn_h l^2)n_l$
Reversible Transformer	$\max(bld_{ff}, bn_h l^2)$	$(bn_h ld_{ff} + bn_h l^2)n_l$
Chunked Reversible Transformer	$\max(bld_{model}, bn_h l^2)$	$(bn_h ld_{ff} + bn_h l^2)n_l$
LSH Transformer	$\max(bld_{ff}, bn_h ln_r c)n_l$	$(bld_{ff} + bn_h n_r lc)n_l$
Reformer	$\max(bld_{model}, bn_h ln_r c)$	$(bld_{ff} + bn_h n_r lc)n_l$

Model Architecture





(a) Normal (b) Bucketed (c) Q = K (d) Chunked

Assumption prove

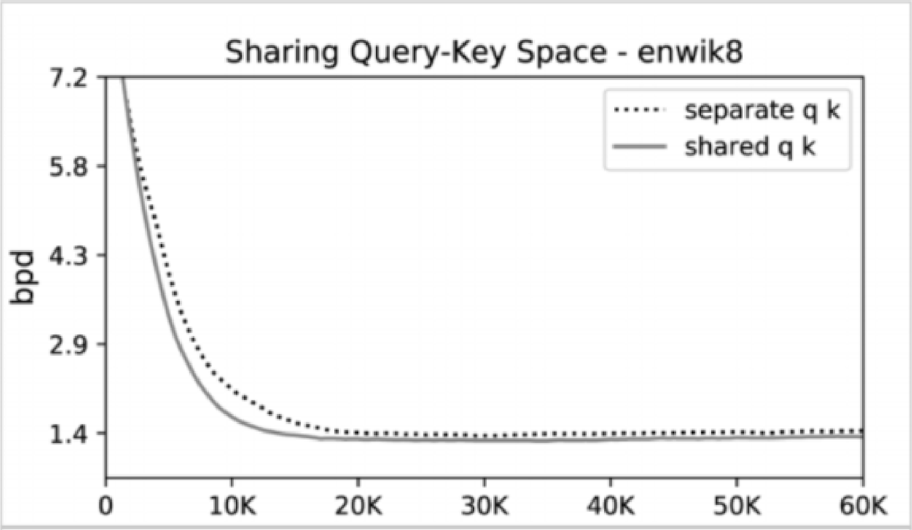
- Three main assumption

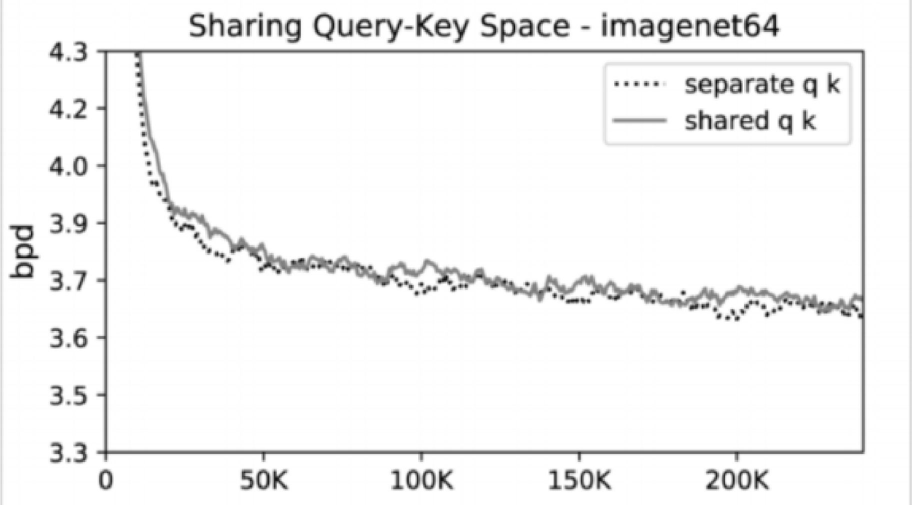
1. Reformer structure Q = K

2. Reformer attention block -> reversible layer

3. LSH attention -> not decrease performance of transformer & Linear complexity

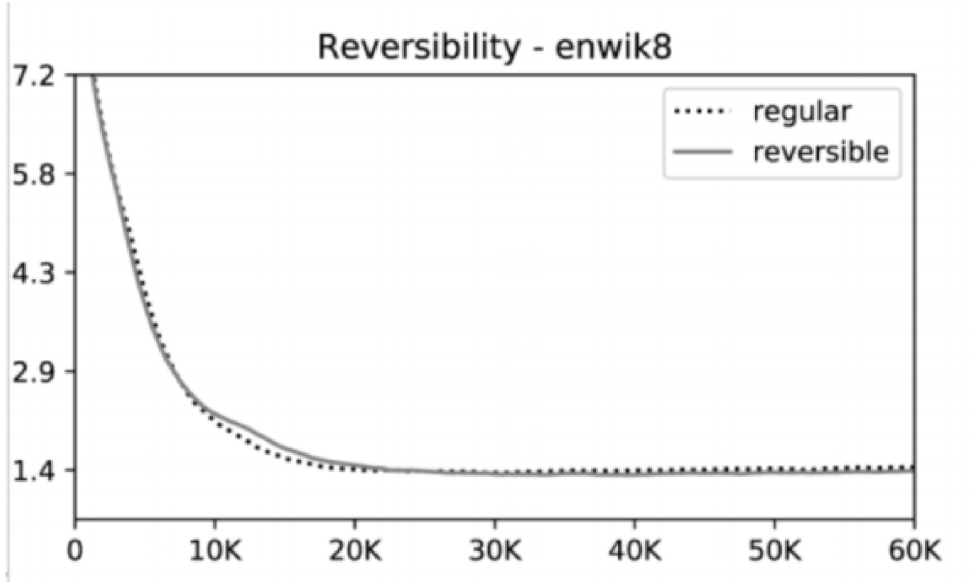
- Assumption 1 : Q= K

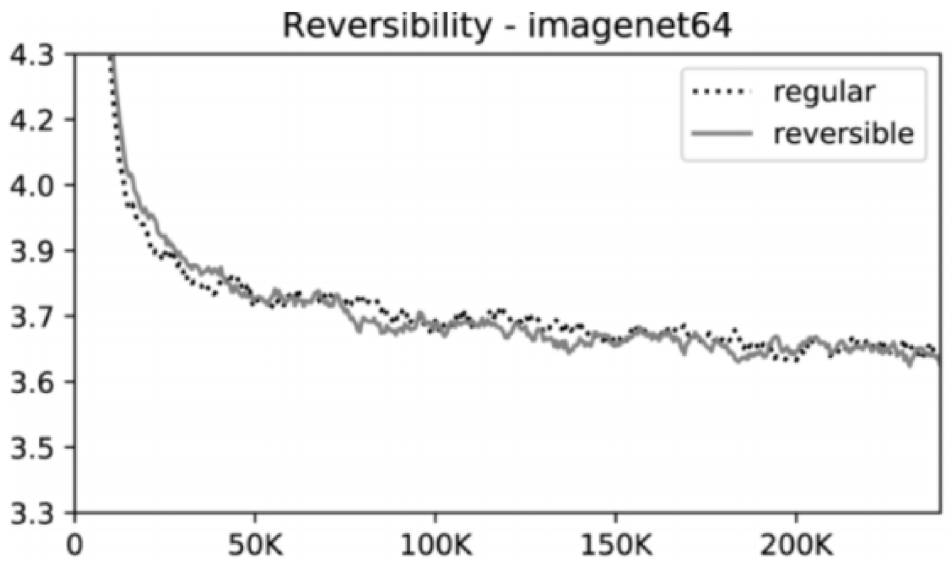




Assumption prove

- Assumption 2: Reversible layer

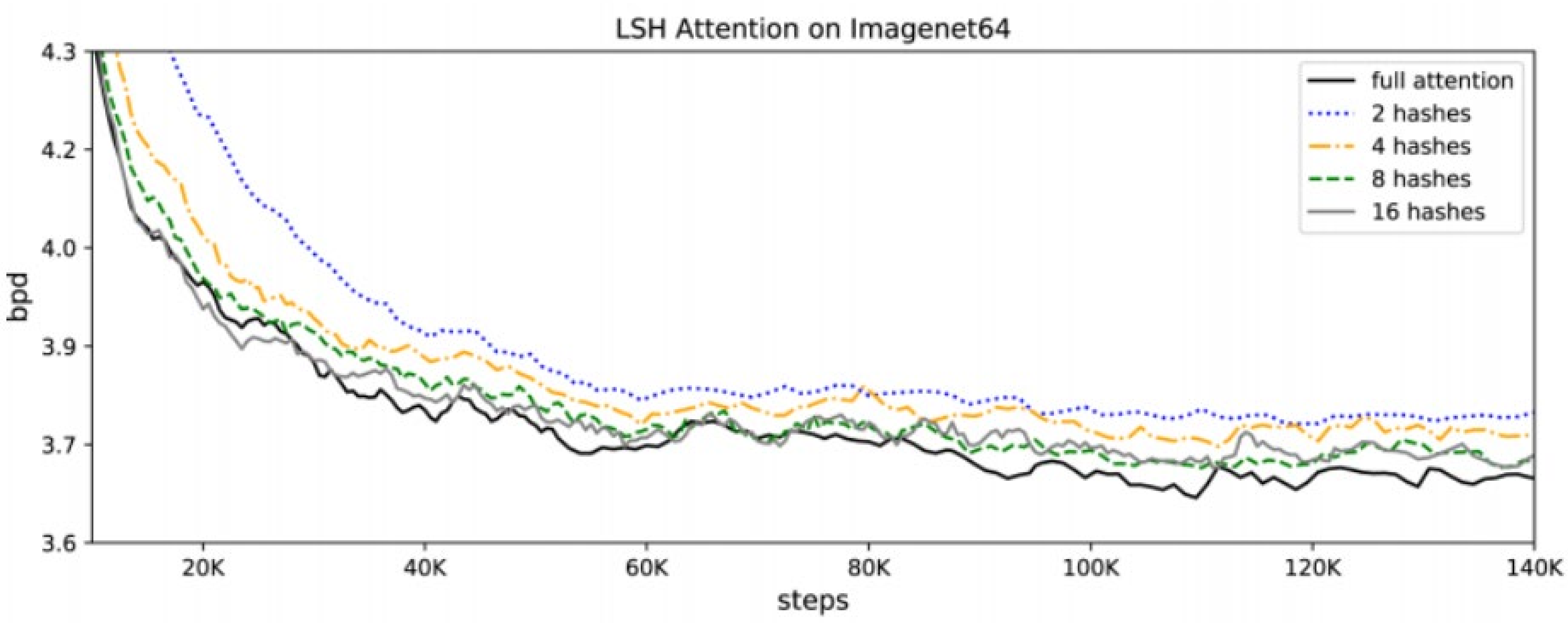




- No significant difference

Assumption prove

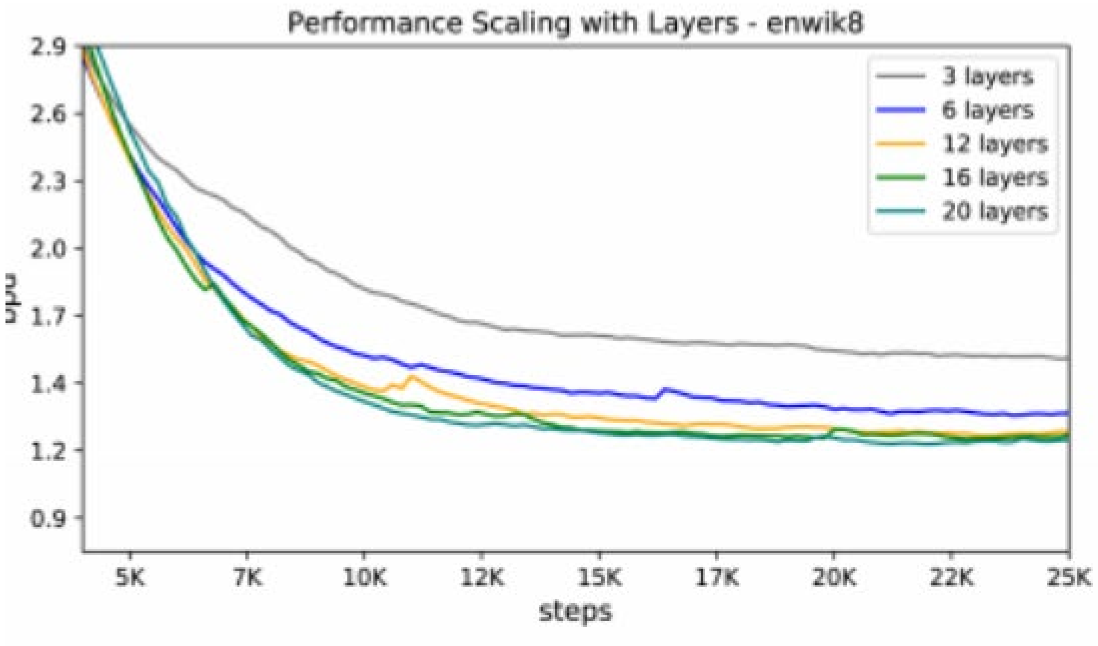
- Assumption 3: LSH attention

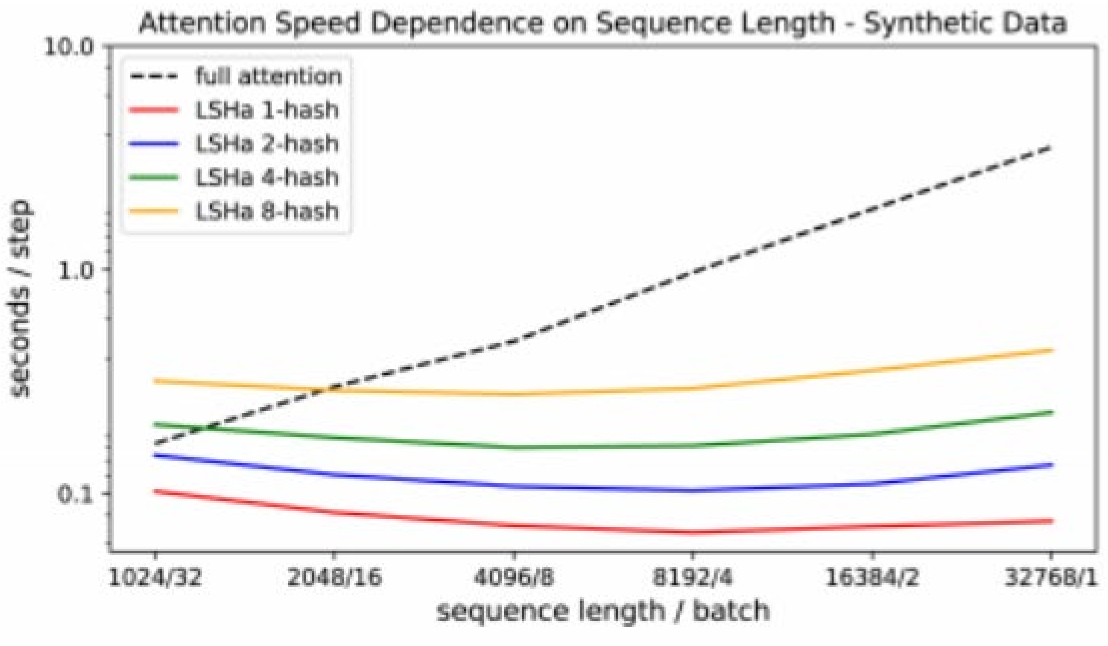


More parallel hashes -> lower gap from full attention

Assumption prove

- Assumption 3: LSH attention





  - Performance (bits per dimension)
  - Consistent speed on length

Appendix

- Angular LSH

- Magnitude / norm
- Angular LSH -> Transformer Layer normalization (scale is not important)
- cosine similiarty -> Comparison between Euclidean space and Spherical space

- LSH validty

Train \ Eval	Full Attention	LSH-8	LSH-4	LSH-2	LSH-1
Full Attention	100%	94.8%	92.5%	76.9%	52.5%
LSH-4	0.8%	100%	99.9%	99.4%	91.9%
LSH-2	0.8%	100%	99.9%	98.1%	86.8%
LSH-1	0.8%	99.9%	99.6%	94.8%	77.9%

End of presentation