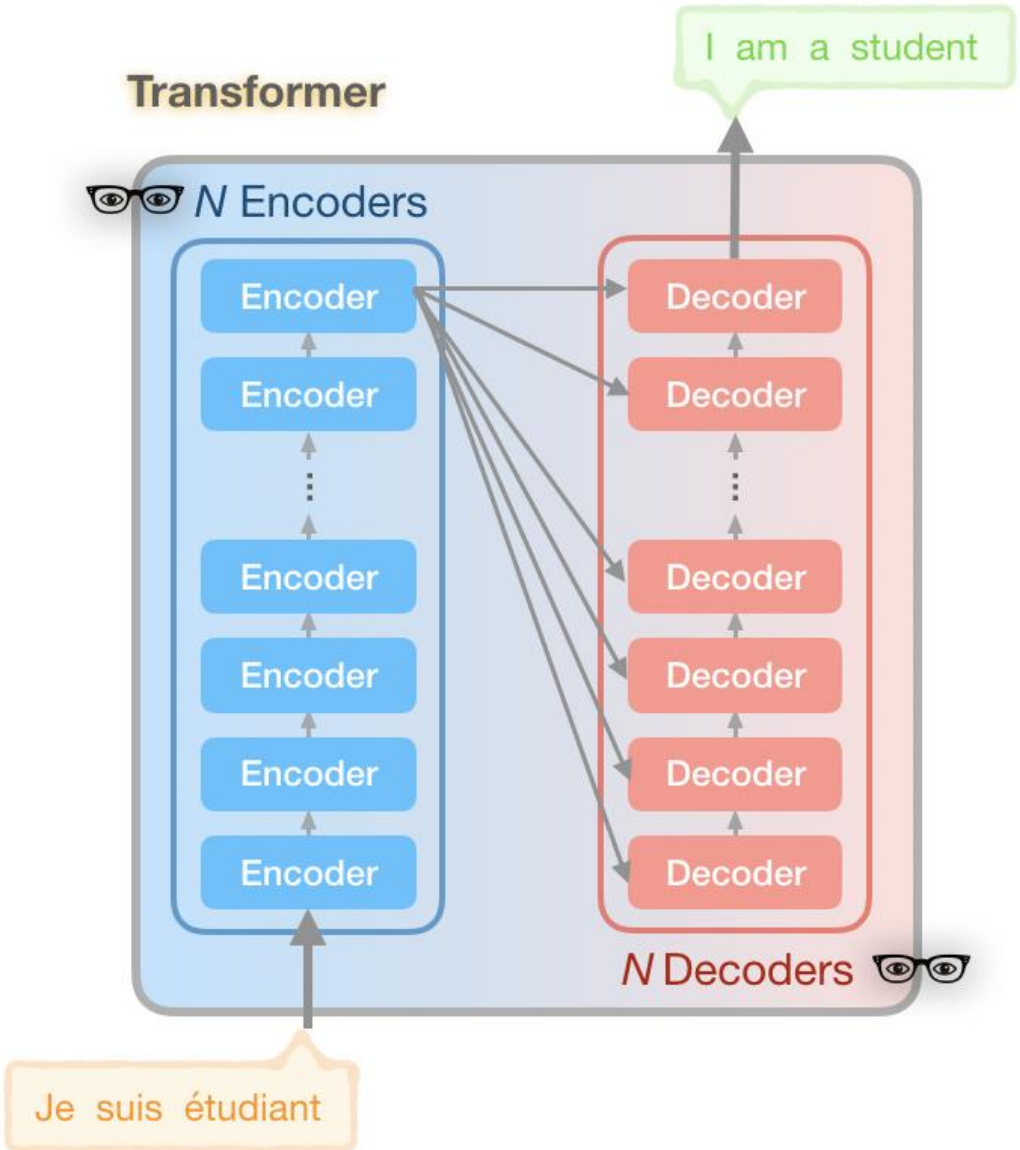
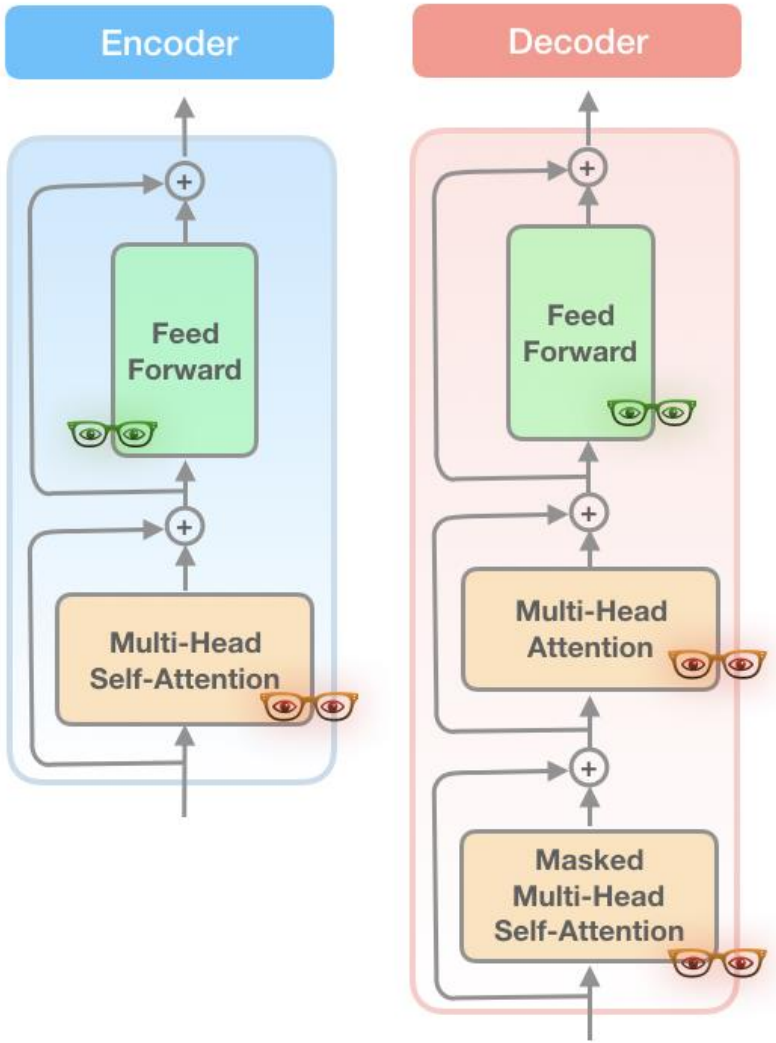
2010-10-26 세미나 발표

REFORMER: THE EFFICIENT TRANSFORMER

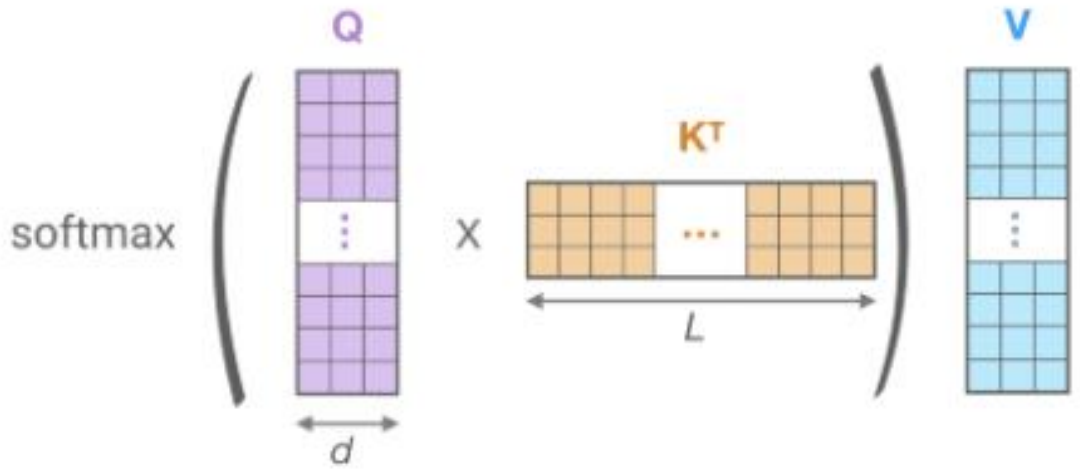
김남형

1. 기존 Transformer의 문제점





1. 기존 Transformer의 문제점: ① Attention 구조에 의한 메모리 문제



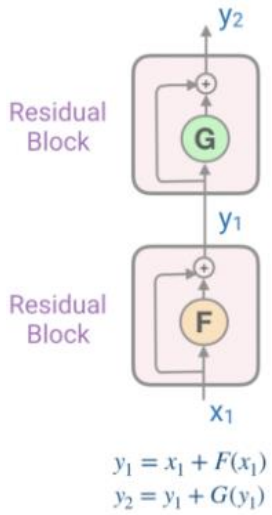
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Attention의 복잡도는 Q의 크기와 K의 크기의 곱에 비례
데이터 Sequence가 L배 길어지면 L^2의 큰 구조가 필요

▶문서 단위에 적용시키는 것에 한계

▶서로 비슷한 쌍만 Attend한다.

1. 기존 Transformer의 문제점: ② N-stacked Residual Connection에 의한 메모리 문제



ResNet 에서의 메모리는 Bottleneck 현상을 보임
Back-prop.을 하기 위해서 개별 layer의 activation을 메모리에 저장해야함

▶Transformer는 N층으로 Attention-Layer와 FFL가 결합되어 있으므로, 중간 결과를 저장하는 데 N배 달하는 메모리가 필요.

▶Reversible layer를 이용해 메모리를 줄인다.

1. 기존 Transformer의 문제점: ③ Feed Forward Layer에 의한 메모리 문제

$$\mathrm{FFN}(x) = max(0, x \cdot W_1 + b_1) \cdot W_2 + b_2$$

Transformer 논문에서 제안된 토큰의 최대 개수: 512
내부 Feed Forward Layer의 입출력은 그 4배인 2048차원

▶입력 차원이 크기 때문에 데이터 길이가 길면 FF 구조가 차지하는 메모리도 상당함.

▶Chunking을 이용해 메모리를 줄인다

2. Reformer의 Idea: ① Locality-Sensitive Hashing Attention

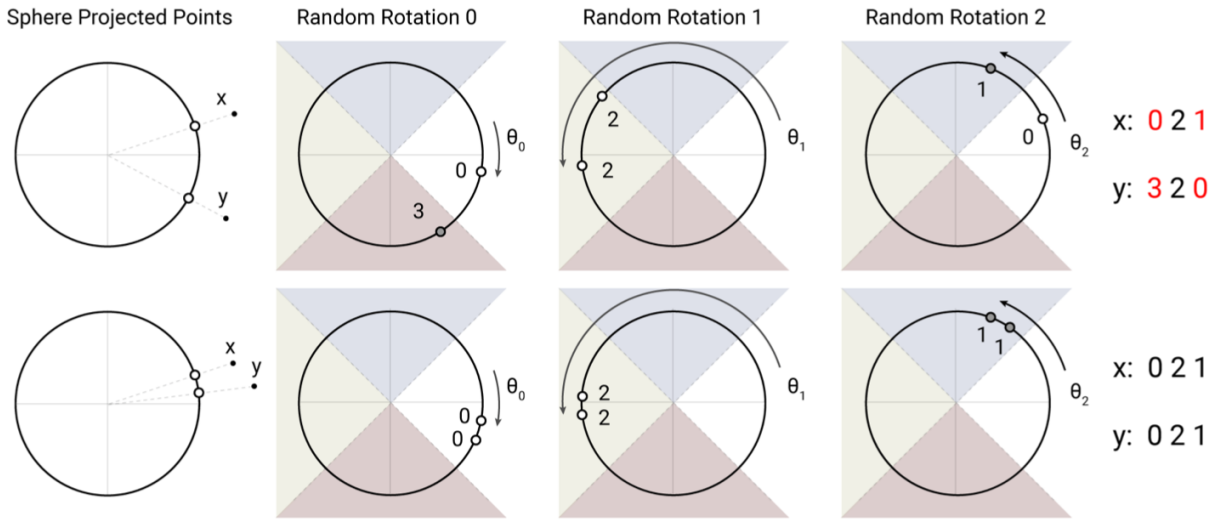
$$\text{softmax}(\frac{q_i K^T}{\sqrt{d_k}})V$$

Softmax는 가장 큰 값들에 의해 결정되어 지므로,
각 qi별로 가장 가까운 key들만 포커스 해도 됨.

예를 들어, length가 64K 일 때, 각 query 별로 가까운 32 or 64개 정도의 키만 봐도 충분함.

LSH

1) 영향력이 높은 단어 쌍은 임베딩 공간에서 서로 비슷한 쌍
2) LSH를 이용해서 빠르게 찾을 수 있다

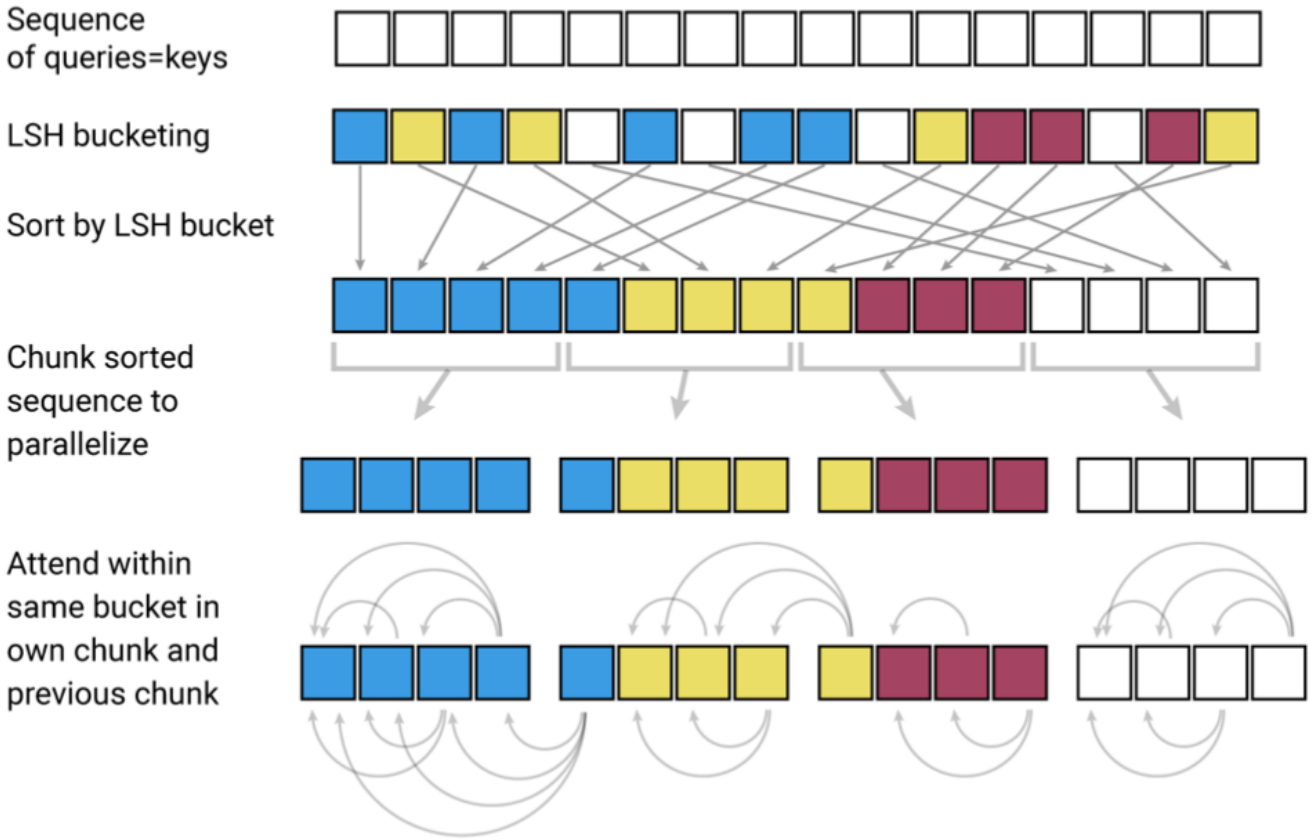


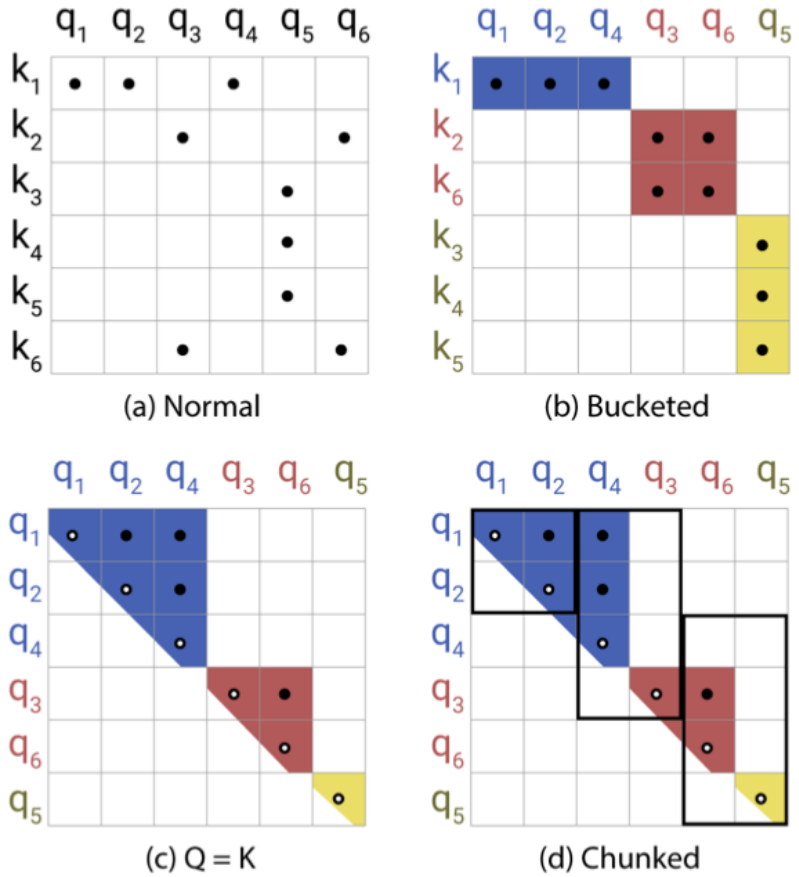
Random matrix R을 설정 (size: d_k, b/2)
h(x) = argmax([xR; -xR])

$$Query(Q) = Key(K)$$

Query, Key를 비교하기 위해 Shared-QK
: query들과 key가 같아지게 설정. 성능에 영향을 끼치지 않음

2. Reformer의 Idea: ① Locality-Sensitive Hashing Attention





2. Reformer의 Idea: ① Locality-Sensitive Hashing Attention

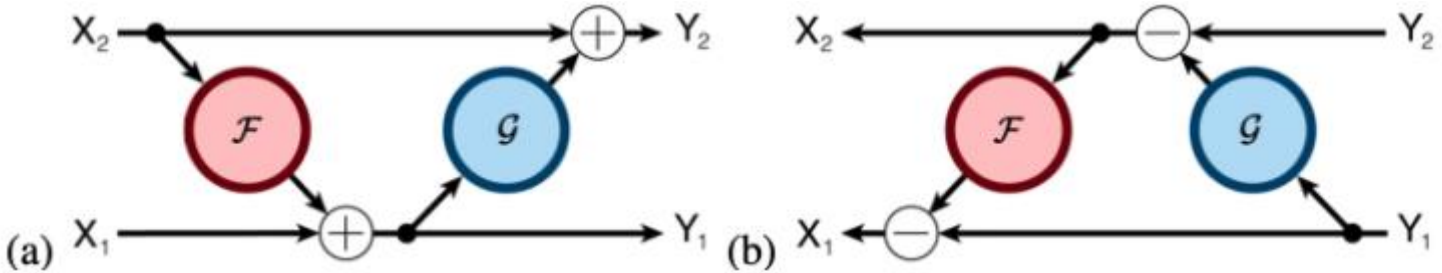
Multi-round LSH attention

해시를 쓰면 비슷한 토큰이 다른 버킷에 들어갈 확률이 낮음
그럼에도 이 낮은 확률 또한 없애기 위해, 여러 번 LSH attention을 사용함

Causal masking for shared-QK attention

일반적으로 Transformer는 자기자신을 포함해서 어텐션
그러나 Q==K인 상황에서는 자기 자신이 무조건 큰 값을 가질 것이므로
자기 자신을 어텐션 하지 않도록 함. (K가 한 개인 경우 제외)

2. Reformer의 Idea: ② Reversible Transformer



$$y_1 = x_1 + F(x_2) \qquad y_2 = x_2 + G(y_1)$$

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

$y = x + F(x)$ 기존 Residual connection: x에서 y를 계산할 수 있어도, y에서 x를 역으로 계산할 수 없음

RevNet: The reversible residual network: Backpropagation without storing activations

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

y1, y2가 주어졌을 때, x2=y2−G(y1)로 역산할 수 있고 x1=y1−F(x2)로 역산할 수 있음

$$Y_1 = X_1 + \text{Attention}(X_2) \qquad Y_2 = X_2 + \text{FeedForward}(Y_1)$$

출력으로 입력을 복원할 수 있음 → 각 중간 단계의 입출력을 저장할 필요 없이 출력에서 gradient 계산
Reversible residual layers를 써서 activation을 N번 저장하지 않고, 오직 한번만 저장한다

2. Reformer의 Idea: ③ Chunking

Feed-Forward Layer는 Attention Layer와 다르게 데이터 포인트의 위치에 무관하게 계산됨

- ▶ 데이터 포인트들을 묶어 줄 수 있다면 계산하는 단위를 나눌 수 있음
- ▶ 전체 데이터 포인트에 대한 FFL의 가중치를 한번에 메모리에 올리지 않아도 됨.

$$Y_2 = \left[Y_2^{(1)}; \ldots; Y_2^{(c)}\right] = \left[X_2^{(1)} + \text{FeedForward}(Y_1^{(1)}); \ldots; X_2^{(c)} + \text{FeedForward}(Y_1^{(c)})\right]$$

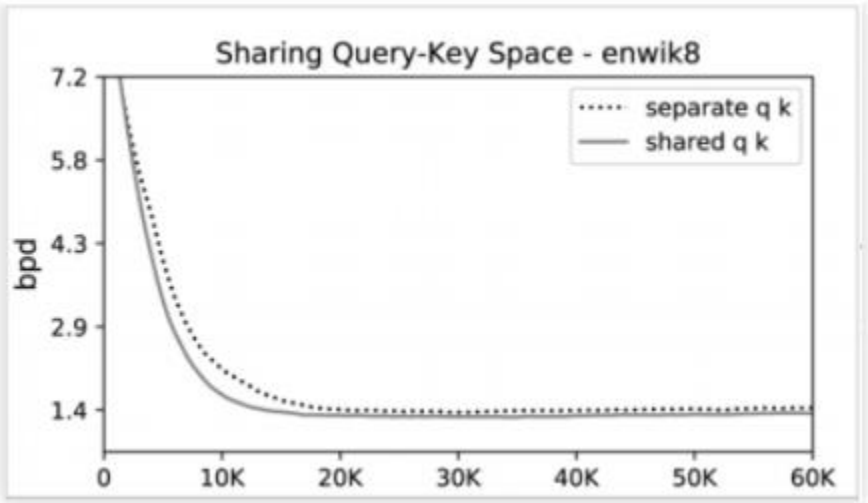
Model Type	Memory Complexity	Time Complexity
Transformer	$\max(bld_{ff}, bn_h l^2)n_l$	$(bld_{ff} + bn_h l^2)n_l$
Reversible Transformer	$\max(bld_{ff}, bn_h l^2)$	$(bn_h l d_{ff} + bn_h l^2)n_l$
Chunked Reversible Transformer	$\max(bld_{model}, bn_h l^2)$	$(bn_h l d_{ff} + bn_h l^2)n_l$
LSH Transformer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

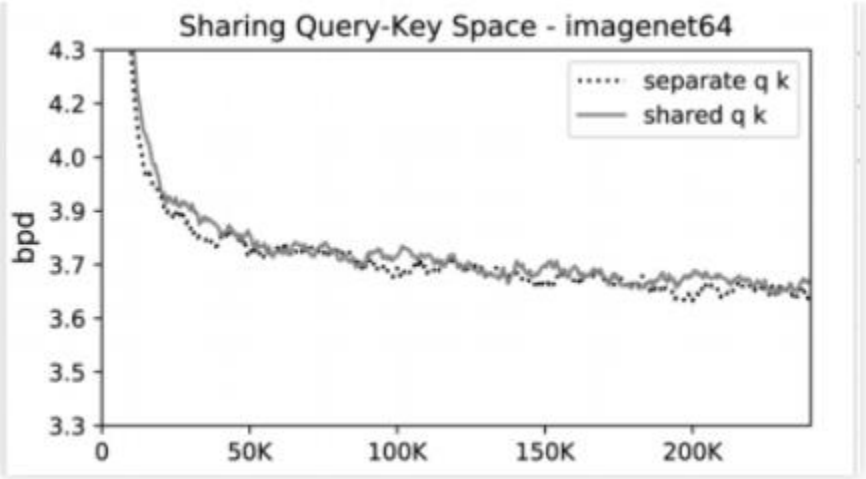
b = 배치수, l=입력길이, nl = 레이어 수, nc = 청크 개수, nr = 해쉬 반복 수, nh = 헤드수

3. Experiments

데이터셋
- Imagenet64: 20000개 이상의 종류, 14,000,000개 이상의 이미지
- Enwiki8-64K: 각 부분이 64K 토큰으로 구성된 데이터

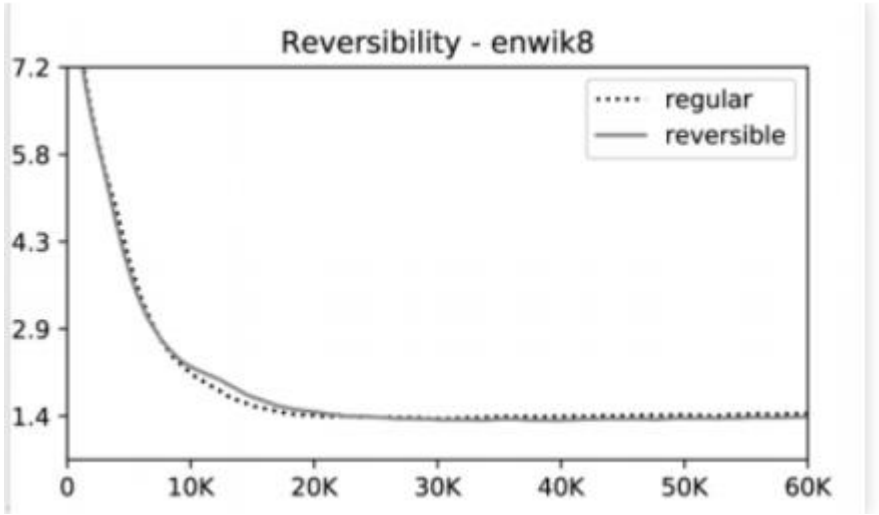
1. Query는 Key와 같아도 무방하다

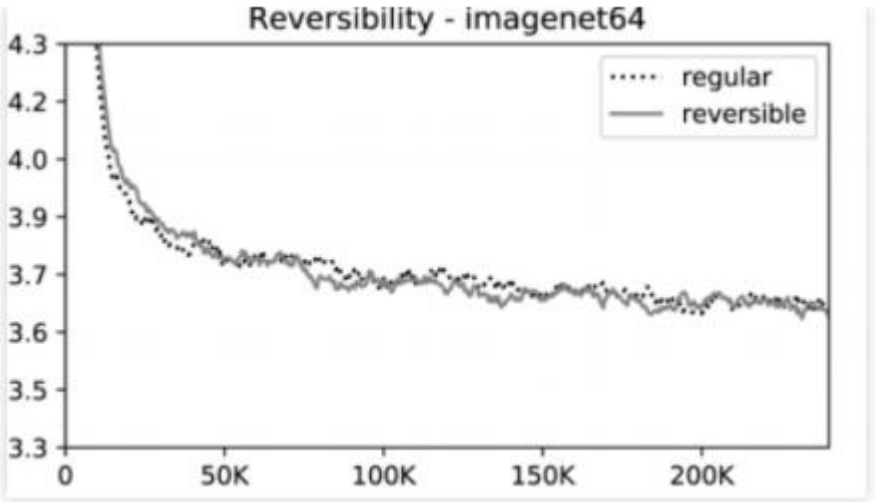




3. Experiments

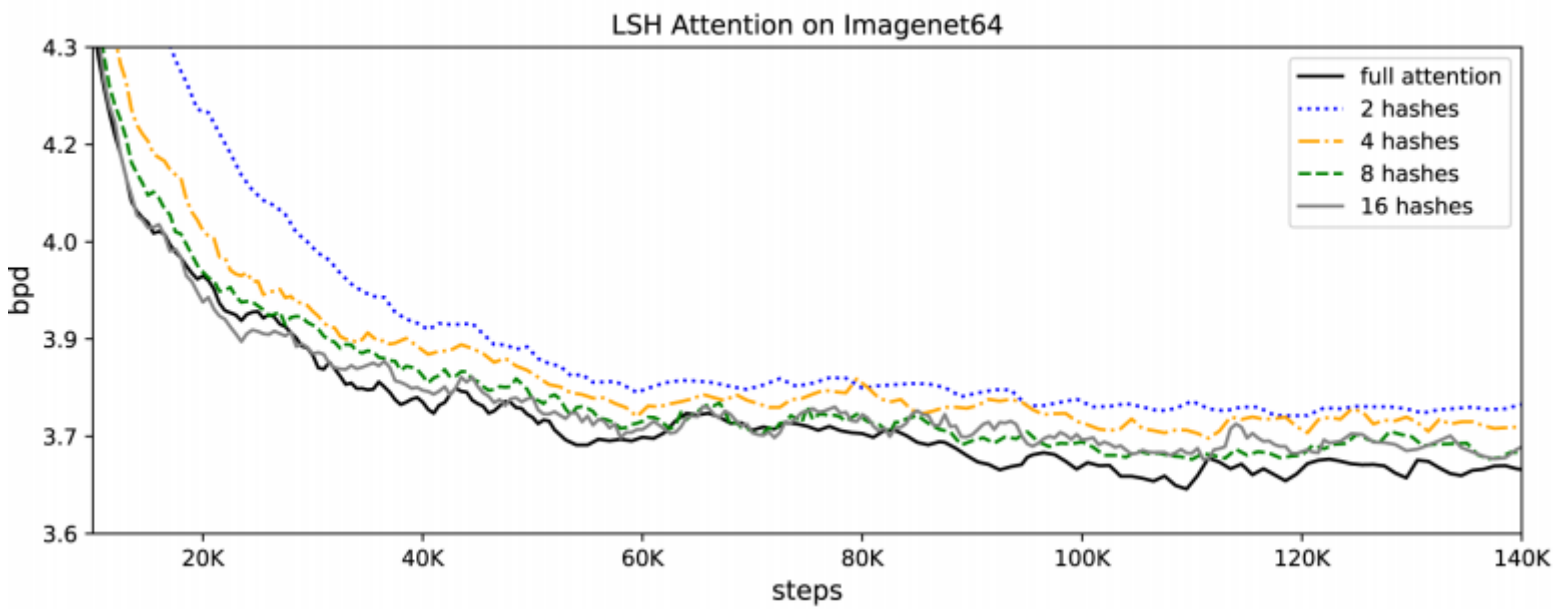
2. Attention Block은 Reversible Layer 형태로 중첩할 수 있다.





3. Experiments

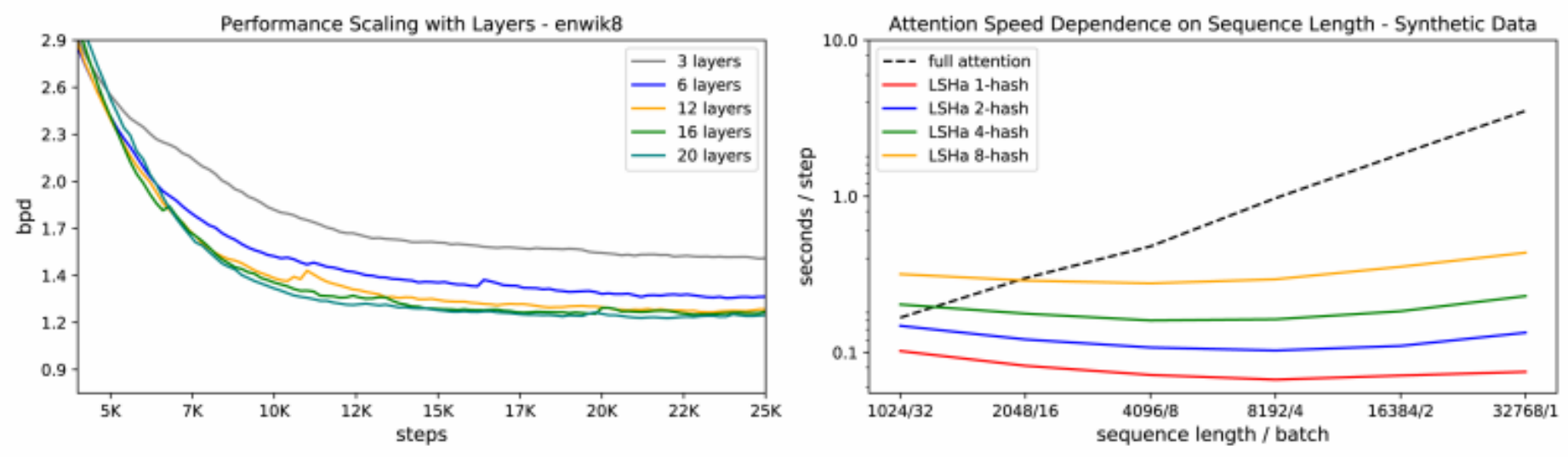
3. LSH Attention을 활용하면 기존 구조의 성능을 크게 저하시키지 않으면서,
입력 길이에 선형인 시간 복잡도를 개선할 수 있다.



Imagenet64 데이터셋으로 실험 결과
Hash를 많이 둘 수록 Full Attention을 한 경우와 성능 차이가 줄어드는 것을 확인할 수 있음.
Hash 8이상부터 거의 비등

3. Experiments

3. LSH Attention을 활용하면 기존 구조의 성능을 크게 저하시키지 않으면서,
입력 길이에 선형인 시간 복잡도를 개선할 수 있다.



en-wiki8: LSH Attention 사용시 데이터 시퀀스 길이가 길어져도 매 단계마다 소요시간이 일정
Full Attention의 기존 모델: 선형으로 증가함을 확인할 수 있음

LSH Attention: Layer를 많이 둘수록 더 높은 성능, 12개 넘어가면 향상폭은 미비

4. Conclusion

1. 실험을 통해 기존 Transformer와 비교하여 학습과정에서 무시할 수 있는 정도의 영향만 있음.

2. LSH 사용시 버킷 수에 따라 학습에 영향을 줄 수 있음.

3. 64K 시퀀스의 텍스트 처리와 12K 시퀀스 이미지 생성 작업 실험하여
   트랜스포머와 같은 결과를 얻지만, 속도와 메모리 효율성이 높은 것을 증명함.