# generating long sequences

# with sparse transformers

최우진

2020/01/14

# 0. Abstract (요약)

- 문제: 트랜스포머는 좋은 성능을 가졌지만, $O(N^2)$으로 시간과 메모리가 필요하다.
- 해결: 이 논문은 이를 $O(n\sqrt{n})$로 감소시키는 어텐션 행렬의 희소 인수분해를 소개한다.
- 해결방안:

  a) 더 깊은 네트워크를 훈련시키기 위한 구조와 초기화 변형

  b) 메모리 절약을 위한 어텐션 행렬 재계산

  c) 학습을 위한 빠른 어텐션 커널

- 위와 같은 해결방안을 가진 모델을 "Sparse Transformers"라고 부른다.
- Sparse Transformers의 특징:

  수백 개의 레이어를 통해 수만 번의 시계열 데이터를 학습할 수 있다.

- 위에서 설명한 특징을 확인하기 위해 여러 종류의 시계열 데이터를 사용하였다.

# 1. Inrtoduction (도입)

- 본 논문의 주된 기여는 성능 저하없이 시퀀스 길이에 따라 $O(n\sqrt[p]{n})$로 확장되는 어텐션 행렬의 몇 가지 희소 인수분해를 도입한 것이다.

- 기존 트랜스포머와 다른 점

- 매우 깊은 네트워크의 학습을 개선하기 위해, 재구성된 잔차 블록 및 가중치 초기화
- 어텐션 행렬의 부분집합을 효율적으로 계산하는 희소 어텐션 커널 집합
- 메모리 사용량을 줄이기 위해 역전파 중 어텐션 행렬 재계산

# 2. Background (배경지식)

- Sequence의 확률: 이전 단어가 주어졌을 때, 다음 단어가 나올 확률들을 전부 곱한 것
- $p(x) = \prod_{i=1}^{n} p(x_i|x_1, \ldots, x_{i-1};\ \theta) = p(x_1)p(x_2|x_1)\cdots p(x_T|x_1, \ldots, x_{T-1})$
- 예) P("he is student") = P("he") x P("is"|"he") x P("student"|"he", "is")
- 데이터 특징:

- 이미지, 텍스트 및 오디오를 이산 토큰으로 다루며, 보통 데이터 값 자체를 사용한다.

- 학습 과정:

- 네트워크 $\theta$가 시계열 데이터를 입력으로 받고 소프트맥스를 사용한 후 다음 토큰으로 가능한 v(vocabulary 크기)개의 값을 categorical 분포로 출력한다.

- 학습목표:

- log 확률을 최대화

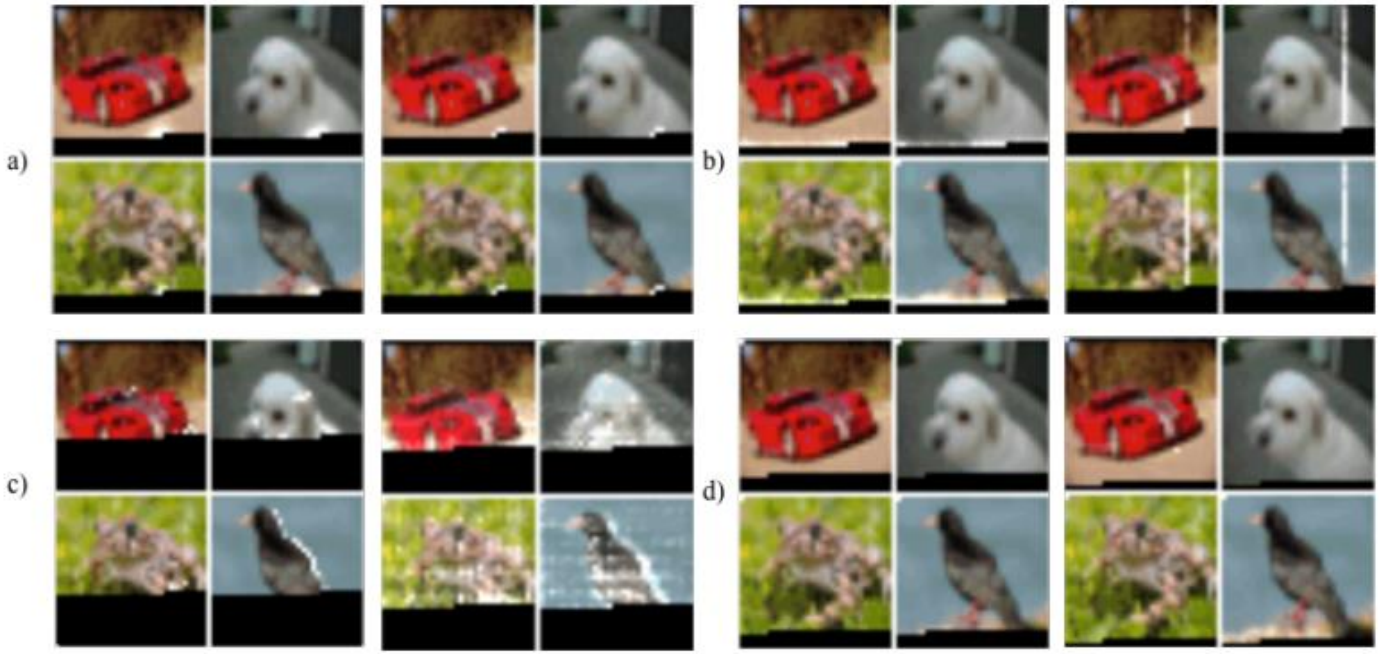
# 3. Factorized Self-Attention

## 3.1. Qualitative assessment of learned attention patterns

- 128개 레이어 self-attention 네트워크가 'CIFAR-10'을 학습한 어텐션 패턴 시각화

초기 레이어

19~20




중간 레이어

64~128

# 3. Factorized Self-Attention

## 3.1. Qualitative assessment of learned attention patterns

- 대부분 계층에서 희박한 어텐션 패턴을 가지고 있다는 것을 확인.
- 이로 인해, 전체적으로 어텐션하기 보다는 미리 결정된 희소 패턴을 도입하기로 결정.
- 이러한 희소 어텐션 패턴을 다양한 데이터셋을 통해 검증.

# 3. Factorized Self-Attention
## 3.2. Factorized self-attention

- $S_i$는 i번째 출력 벡터가 어텐션하는 입력 벡터의 인덱스들의 집합이다.
- 본 논문에서 기준 attention mechanism

$$\text{Attend}(X, S) = \left( a(\mathbf{x}_i, S_i) \right)_{i \in \{1,\ldots,n\}} \quad (2)$$

$$a(\mathbf{x}_i, S_i) = \text{softmax}\left( \frac{(W_q \mathbf{x}_i) K_{S_i}^T}{\sqrt{d}} \right) V_{S_i} \quad (3)$$

$$K_{S_i} = \left( W_k \mathbf{x}_j \right)_{j \in S_i} \quad V_{S_i} = \left( W_v \mathbf{x}_j \right)_{j \in S_i} \quad (4)$$

- Fully self-attention은 $S_i = \{j : j \leq i\}$으로 정의하고 모든 요소가 자신 위치와 이전 모든 위치를 어텐션한다.

# 3. Factorized Self-Attention
## 3.2. Factorized self-attention

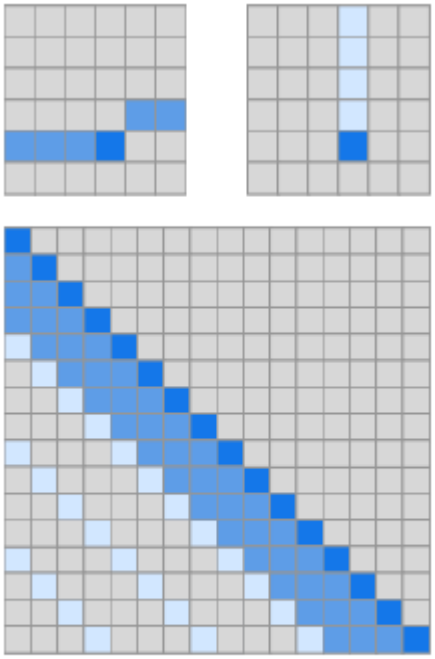
- Factorized self-attention은 p개의 self-attention head를 가지고 m번째 head가 잘 고른 indices에만 어텐션할 수 있게 하는 방식이다.

- $A_i^{(m)} \subset \{j : j \leq i\}$이고 $S_i = A_i^{(m)}$으로 설정하고 $\left|A_i^{(m)}\right| \propto \sqrt[p]{n}$ 이다.

# 3. Factorized Self-Attention
## 3.3. Two-dimensional factorized attention

- strided attention( p = 2 )

-> 한 헤드는 $l$ 이전 위치들을 어텐션, 다른 한 헤드는 모든 $l$ 위치에 어텐션



(b) Sparse Transformer (strided)

$$A_i^{(1)} = \{t, t+1, \ldots, i\}, t = \max(0, i - l)$$

$$A_i^{(2)} = \{j : (i - j) \bmod l = 0\}$$

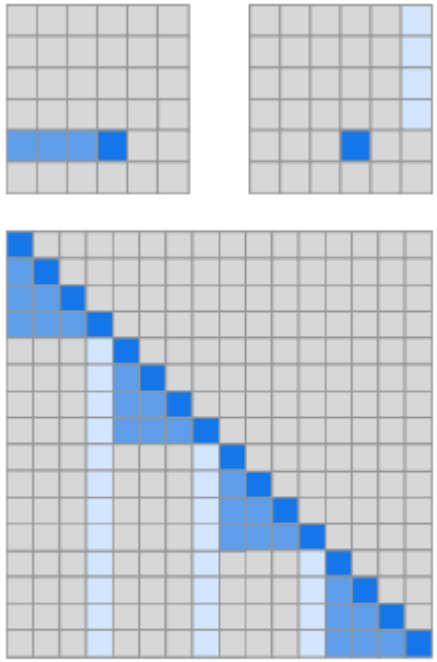
$l$: stride, $\sqrt{n}$와 가까운 값으로 설정

**데이터 구조가 일정한 경우에 사용**

# 3. Factorized Self-Attention
## 3.3. Two-dimensional factorized attention

- fixed attention( p = 2 )

-> 한 헤드는 $l$ 이전 위치들을 어텐션, 다른 한 헤드는 모든 $l$ 위치에 어텐션



(c) Sparse Transformer (fixed)

$$A_i^{(1)} = \{j: (\lfloor j/l \rfloor = \lfloor i/l \rfloor)\}$$

$$A_i^{(2)} = \{j : j \bmod l \in \{t, t+1, \ldots, l\}\}$$

$t = l - c$, $c$는 하이퍼파라미터

**데이터 구조가 일정하지 않은 경우**

# 4. Sparse Transformer
## 4.1. Factorized attention heads

- Standard dense attention 식

$$\text{attention}(X) = W_p \cdot \text{attend}(X, S) \qquad (5)$$

- $W_p$: post-attention 가중치 행렬

- Factorized self-attention을 결합하는 방식(1)

$$\text{attention}(X) = W_p \cdot \text{attend}(X, A^{(r \bmod p)}) \qquad (6)$$

-> r: 현재 residual 블록의 인덱스, p: factorized 어텐션 헤드 수

# 4. Sparse Transformer
## 4.1. Factorized attention heads

- Factorized self-attention을 결합하는 방식(2)

$$\text{attention}(X) = W_p \cdot \text{attend}(X, \bigcup_{m=1}^{p} A^{(m)}) \qquad (7)$$

-> merged 헤드

- Factorized self-attention을 결합하는 방식(3)

$$\text{attention}(X) = W_p \Big( \text{attend}(X, A)^{(i)} \Big)_{i \in \{1,\ldots,n_h\}} \qquad (8)$$

# 4. Sparse Transformer
## 4.2. Scaling to hundreds of layers

- Transformer는 많은 레이어로 학습하기 어렵다는 것을 발견했다.
- Pre-activation residual 블록을 사용하여 N개 레이어의 네트워크를 정의한다.

$$H_0 = \text{embed}(X, W_e) \tag{9}$$

$$H_k = H_{k-1} + \text{resblock}(H_{k-1}) \tag{10}$$

$$y = \text{softmax}(\text{norm}(H_N)W_{out}) \tag{11}$$

$$a(H) = \text{dropout}(\text{attention}(\text{norm}(H))) \tag{12}$$

$$b(H) = \text{dropout}(\text{ff}(\text{norm}(H + a(H)))) \tag{13}$$

$$\text{resblock}(H) = a(H) + b(H) \tag{14}$$

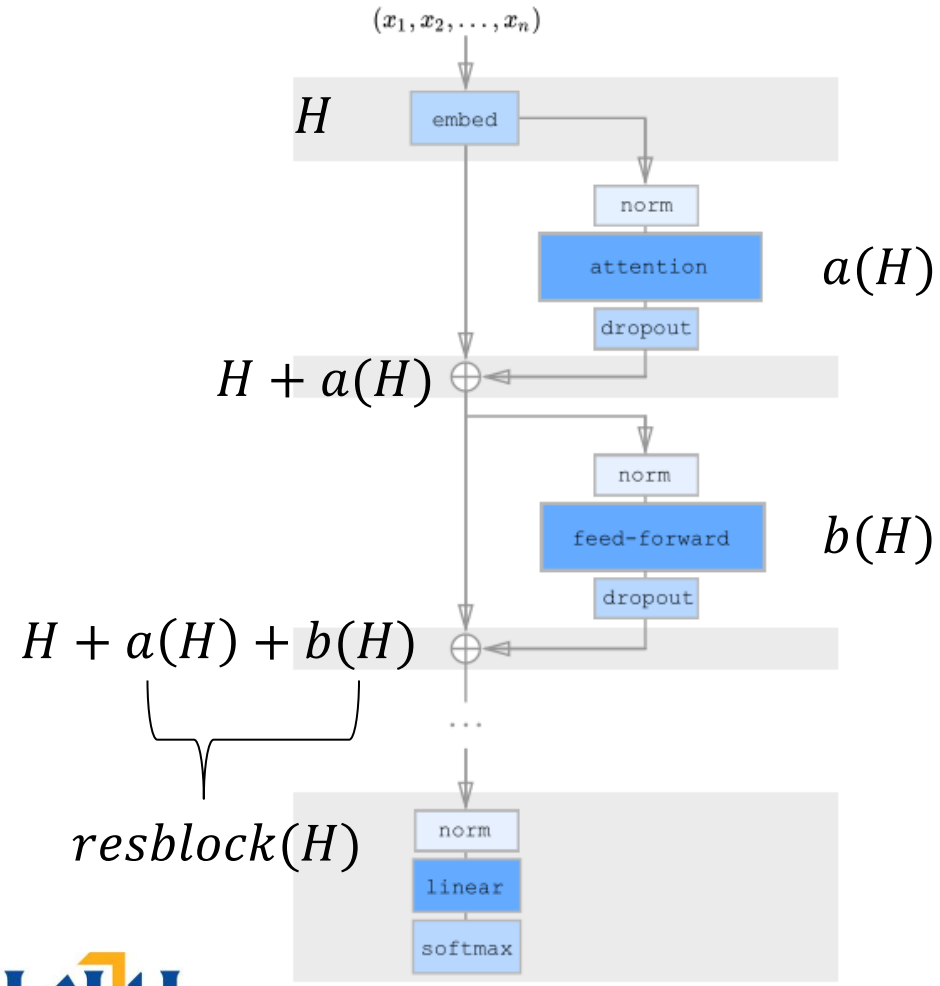
$W_{\text{out}}$: 가중치 행렬

$$ff(x) = W_2 f(W_1 x + b_1) + b_2$$

$$f(x) = x \odot sigmoid(1.702 \cdot x)$$

# 4. Sparse Transformer

## 4.2. Scaling to hundreds of layers



# 4. Sparse Transformer
## 4.3. Modeling diverse data types

- Transformer와 같이 위치에 영향 받지 않는 구조에서 보통 공간적인 임베딩을 한다.
- 따라서 데이터의 구조나 factorized 어텐션 패턴을 인코딩하여 학습된 임베딩을 사용하는 것이 모델 성능에 중요하다.
- $n_{emb} = d_{data}$ 혹은 $n_{emb} = d_{attn}$ 을 입력 부분에 추가한다.

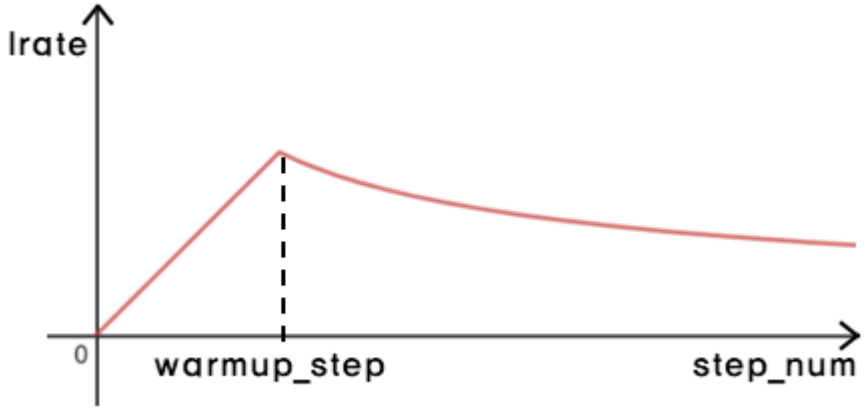
→ $d_{data}$ : 데이터의 차원 수, $d_{attn}$ : factorized 어텐션의 차원 수

$$\text{embed}(X, W_e) = \left( \mathbf{x}_i W_e + \sum_{j=1}^{n_{emb}} \mathbf{o}_i^{(j)} W_j \right)_{\mathbf{x}_i \in X} \qquad (15)$$

→ $x_i$ : 시퀀스에서 i번째 요소를 원-핫 인코딩, $o_i^{(j)}$ : j차원에서 $x_i$ 의 위치를 원-핫 인코딩

# 5. Training

- Adam optimizer, learning rate는 고정하지 않고 진행한다.



# 6. Experiments

| Model | Bits per byte |
|---|---|
| **CIFAR-10** | |
| PixelCNN (Oord et al., 2016) | 3.03 |
| PixelCNN++ (Salimans et al., 2017) | 2.92 |
| Image Transformer (Parmar et al., 2018) | 2.90 |
| PixelSNAIL (Chen et al., 2017) | 2.85 |
| **Sparse Transformer 59M (strided)** | **2.80** |
| **Enwik8** | |
| Deeper Self-Attention (Al-Rfou et al., 2018) | 1.06 |
| Transformer-XL 88M (Dai et al., 2018) | 1.03 |
| Transformer-XL 277M (Dai et al., 2018) | **0.99** |
| **Sparse Transformer 95M (fixed)** | **0.99** |
| **ImageNet 64x64** | |
| PixelCNN (Oord et al., 2016) | 3.57 |
| Parallel Multiscale (Reed et al., 2017) | 3.7 |
| Glow (Kingma & Dhariwal, 2018) | 3.81 |
| SPN 150M (Menick & Kalchbrenner, 2018) | 3.52 |
| **Sparse Transformer 152M (strided)** | **3.44** |
| **Classical music, 5 seconds at 12 kHz** | |
| Sparse Transformer 152M (strided) | **1.97** |

https://openai.com/blog/sparse-transformer/

# 5. Reference (참조)


- Child, R., Gray, S., Radford, A., & Sutskever, I. (2019). Generating long sequences with sparse transformers. arXiv preprint arXiv:1904.10509.
- https://github.com/YoonjinXD/Yoonjin/blob/master/posts/Transformer.md
- https://medium.com/platfarm/%EC%96%B4%ED%85%90%EC%85%98-%EB%A9%94%EC%BB%A4%EB%8B%88%EC%A6%98%EA%B3%BC-transfomer-self-attention-842498fd3225

# generating long sequences

# with sparse transformers

감사합니다