

딥러닝 논문 리뷰 발표

트랜스포머 아키텍처를 최소화하기 (Reducing the Transformer Architecture to a Minimum)



소프트웨어융합학과 박사과정

김동우

목차

1. 서론
2. 배경 및 관련 연구
3. 트랜스포머 아키텍처 및 제안
4. 실험 방법
5. 결론
6. 수식 설명

1. 서론-초록

- 트랜스포머 : 자연어 처리(NLP)와 컴퓨터 비전(CV)에서 널리 사용되고 성공적인 모델 아키텍처
- 어텐션 메커니즘 : NLP에서 긴 시퀀스의 관련 맥락 정보를 추출하는 문제와 CV에서 현실적인 장면에서 관련 정보를 추출하는 문제를 해결
- 다층 퍼셉트론(MLP) : 일반적으로 전체 모델에서 가장 많은 학습 가능한 파라미터를 포함하며, 쿼리 및 키 행렬을 동일한 크기의 단일 행렬로 축소할 수 있다.
- 이 논문은 MNIST, CIFAR-10 및 제한된 조건에서 ImageNet과 같은 널리 사용되는 CV 벤치마크를 테스트하여 기초 작업을 수행하였다.
- 실험 결과, 단순화된 트랜스포머 아키텍처는 MLP 없이, 행렬 축소, 대칭 유사성 행렬을 사용한 경우 원래 아키텍처와 유사한 성능을 보이며, 분류 성능을 저하시키지 않고 최대 90%까지 파라미터를 절약할 수 있음을 보여주었다.

* MNIST (Modified National Institute of Standards and Technology)

- MNIST는 손글씨 숫자(0~9)의 그레이스케일 이미지로 이루어진 데이터셋입니다. 이 데이터셋은 머신 러닝 모델의 성능을 평가하는 데 널리 사용

* CIFAR-10 (Canadian Institute for Advanced Research)

- CIFAR-10은 10개의 서로 다른 객체 클래스를 포함하는 컬러 이미지 데이터셋

1. 서론

- 1) 대형 언어 모델(LLM)은 주어진 질문에 대해 복잡한 텍스트 답변을 생성하는 데 뛰어난 성능을 보였다.
- 2) 모델들의 뛰어난 특징은 파라미터 세트의 엄청난 크기(수십억 개에 달함)이다.
- 3) 탁월한 성과를 거둔 컴퓨팅 모델은 트랜스포머 아키텍처(Vaswani et al., 2017)를 기반이며, 트랜스포머의 성공은 문맥 정보를 포착하는 뛰어난 능력 덕분이다.
- 4) 계산적인 관점에서 트랜스포머는 다음으로 구성된 구조이다.
 - 토큰 문맥을 고려하는 알고리즘인 어텐션 메커니즘
 - 중간 데이터를 비선형적으로 변환하는 다층 퍼셉트론(MLP).

1. 서론

- 어텐션 메커니즘만으로도 충분히 복잡한 패턴을 학습할 수 있다면, MLP를 제거하여 파라미터 수와 계산 비용을 획기적으로 줄일 수 있다는 주장이 가능하다.
 - 모델 해석 가능성(interpretability) 증가
 - 오버피팅 방지
 - 모바일 및 임베디드 환경에서의 활용성 증가 등의 부가적인 장점을 가져올 수 있다.
- 이 논문은 이와 같은 트랜스포머 아키텍처의 구조적 단순화가 실제로도 가능한지를 이론적 분석과 실험적 검증을 통해 입증하고자 하고 있다.

* 오버 피팅 : 머신러닝에서 모델이 훈련 데이터에 너무 과하게 맞춰져서, 훈련 데이터에서는 높은 성능을 보이지만 새로운 데이터(검증 데이터 또는 테스트 데이터)에 대해서는 성능이 떨어지는 현상

2. 배경 및 관련 연구 (Background and Related Work)

◆ 트랜스포머 경량화 시도들

기존 연구 중 일부는 트랜스포머 구조의 특정 요소를 제거하거나 축소하여 경량화된 모델을 제안했습니다.

예를 들어:

- TinyBERT, MobileBERT와 같은 모델은 BERT 기반 구조를 경량화 하여 모바일 환경에서도 실행 가능하게 만들었습니다.
 - DistilBERT는 원래의 BERT 모델보다 더 적은 층(layer)을 사용하면서도, 비슷한 성능을 유지합니다.
- 이러한 모델들은 지식 증류(knowledge distillation)를 이용하여 성능을 유지한 채 모델 크기를 줄이는데 초점을 맞췄습니다.

2. 배경 및 관련 연구 (Background and Related Work)

- ◆ MLP 제거 및 어텐션 단순화 관련 연구

일부 연구는 트랜스포머 내의 MLP(Feed-forward network)를 제거하거나, 어텐션 연산 자체를 단순화하는 방향으로 접근했습니다.

- 예를 들어, Performer, Linformer, Reformer와 같은 모델은 어텐션의 시간 복잡도를 낮추는 데 초점을 맞췄습니다.

→ 이들은 어텐션의 핵심적인 계산 비용을 줄이기 위해 근사(attention approximation)를 사용했습니다.

- 대부분의 연구는 어텐션 효율화에만 집중했을 뿐, "트랜스포머 블록에서 MLP 자체가 필요하지 않은가?"에 대한 근본적인 질문은 거의 다루지 않았습니다.

2. 배경 및 관련 연구 (Background and Related Work)

◆ 본 논문의 차별점

본 논문은 기존 연구들과 다음과 같은 점에서 구별됩니다:

1. MLP가 아예 필요하지 않은가에 대한 실험적 탐색

→ 파라미터 수와 계산 비용을 줄이면서도 성능이 유지되는지를 직접 검증함

2. Q, K, V 행렬의 통합 및 대칭화

→ 어텐션 내부 구조를 간소화하여 구조 자체를 근본적으로 단순화함

3. 단순한 선형 연산으로 충분한 성능 유지 가능성 실험

→ 복잡한 트랜스포머 계층 없이도 모델이 충분히 학습 가능하다는 증거 제시

(Transformer Architecture and Our Modifications)

◆ 기본 트랜스포머 구조

트랜스포머는 일반적으로 두 가지 주요 구성 요소로 이루어진 블록을 여러 개 쌓아 구성됩니다:

1. 어텐션 계층 (Attention Layer)

→ 입력 간의 상호작용을 캡처하며, 쿼리(Q), 키(K), 값(V) 벡터를 사용하여 입력 토큰 간의 관계를 계산합니다.

2. 피드포워드 계층 (Feed-Forward Network, 즉 MLP)

→ 어텐션으로부터 얻은 표현을 정교화하고 확장하는 역할을 합니다.

각 블록은 다음과 같은 순서로 처리됩니다:

• LayerNorm → Attention → LayerNorm → MLP

3. 트랜스포머 아키텍처 및 우리의 수정 제안

(Transformer Architecture and Our Modifications)

1. LayerNorm (레이어 정규화)

- LayerNorm은 입력을 정규화하여 학습을 안정화시키고, 각 층에서의 출력값이 일정한 분포를 갖도록 합니다.

2. Attention (어텐션)

- 어텐션 메커니즘은 모델이 입력 데이터에서 중요한 부분에 집중하도록 도와줍니다. 트랜스포머 모델에서 어텐션은 특히 Self-Attention이라고 불리며, 각 입력 요소가 다른 입력 요소들과 얼마나 관련이 있는지 계산하여, 중요한 정보에 더 많은 가중치를 부여합니다.

3. LayerNorm (레이어 정규화)

- 두 번째 LayerNorm은 어텐션 단계를 거친 후 결과를 다시 정규화하여 모델의 출력을 안정화시킵니다. 첫 번째 정규화 후 모델이 학습한 특성들이 잘 정규화된 상태에서 다음 단계를 진행할 수 있도록 돕습니다.

4. MLP (다층 퍼셉트론)

- MLP는 비선형 변환을 수행하는 신경망의 구성 요소입니다. MLP는 일반적으로 여러 개의 은닉층을 포함하며, 각 층을 통과하면서 입력을 비선형적으로 변환합니다.

3. 트랜스포머 아키텍처 및 우리의 수정 제안

(Transformer Architecture and Our Modifications)

◆ 어텐션 메커니즘의 수식

전형적인 Scaled Dot-Product Attention은 다음과 같은 수식으로 표현됩니다:

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

여기서,

- Q: 쿼리 행렬, K: 키 행렬, V: 값 행렬
- d_k : 키 벡터의 차원 수

이 수식에서 핵심은 소프트맥스를 거친 QK^T 의 유사도 행렬을 V에 곱해 중요한 정보를 취합한다는 것입니다.

3. 트랜스포머 아키텍처 및 우리의 수정 제안

(Transformer Architecture and Our Modifications)

1. LayerNorm (레이어 정규화)

- LayerNorm은 입력을 정규화하여 학습을 안정화시키고, 각 층에서의 출력값이 일정한 분포를 갖도록 한다.

2. Attention (어텐션)

- 어텐션 메커니즘은 모델이 입력 데이터에서 중요한 부분에 집중하도록 도와준다. 트랜스포머 모델에서 어텐션은 특히 Self-Attention이라고 불리며, 각 입력 요소가 다른 입력 요소들과 얼마나 관련이 있는지 계산하여, 중요한 정보에 더 많은 가중치 부여

3. LayerNorm (레이어 정규화)

- 두 번째 LayerNorm은 어텐션 단계를 거친 후 결과를 다시 정규화하여 모델의 출력을 안정화하며, 첫 번째 정규화 후 모델이 학습한 특성들이 정규화된 상태에서 다음 단계를 진행할 수 있도록 돕는다.

4. MLP (다층 퍼셉트론)

- MLP는 비선형 변환을 수행하는 신경망의 구성 요소이며, MLP는 일반적으로 여러 개의 은닉층을 포함하며, 각 층을 통과하면서 입력을 비선형적으로 변환하게 된다.

3. 트랜스포머 아키텍처 및 우리의 수정 제안

(Transformer Architecture and Our Modifications)

이 논문에서는 트랜스포머를 구성하는 요소들 중 일부를 제거하거나 단순화함으로써 구조를 최소화하려 합니다.

1. MLP 제거

- 대부분의 트랜스포머 블록에서 상당수의 파라미터는 MLP에서 발생합니다.
- 어텐션만으로도 비선형성을 충분히 학습할 수 있다고 가정하고, MLP를 완전히 제거합니다.
- 이로 인해 계산량과 메모리 사용량이 큰 폭으로 감소합니다.

2. $Q = K = V$ 통합

- 일반적으로 Q , K , V 는 서로 다른 가중치 행렬을 통해 생성됩니다.
- 하지만 이들을 하나의 공통 행렬 W 로 통합하여 : $Q=K=V=XW$
- → 이렇게 하면 매개변수 수가 세 배에서 한 배로 줄어듭니다.

3. 대칭적인 유사도 계산 도입

- 기존의 QK^T 는 비대칭적인 유사도를 생성할 수 있어, 입력 간 유사도 해석이 불안정할 수 있습니다.
- 이를 개선하기 위해 아래와 같은 대칭 유사도를 도입합니다:

3. 트랜스포머 아키텍처 및 우리의 수정 제안

(Transformer Architecture and Our Modifications)

- 이 방식은 수학적으로 더 안정적이며, 쌍방향 상호작용의 균형을 제공하게 된다.
- 연산량 또한 절반으로 줄어드는 효과가 있다.
- 예시 : 아키텍처 요약 (수정된 트랜스포머 블록)

$$\text{Sim}(x_i, x_j) = \frac{1}{2}(x_i^T x_j + x_j^T x_i)$$

구성 요소	기존 트랜스포머	우리의 간소화 버전
MLP	O	✗ 제거됨
Q, K, V 분리	O	✗ 하나의 행렬로 통합
유사도 계산 방식	비대칭 (QK^T)	✓ 대칭적 계산 적용
파라미터 수	많음	훨씬 적음

4. 실험(Experiments)

4.1 실험 환경 및 설정

* 데이터셋

- MNIST: 손글씨, 숫자(0~9)의 그레이스케일 이미지 분류 (28x28, 흑백, 10클래스)
- CIFAR-10: 10개의 서로 다른 객체 사물 이미지 분류 (32x32, 컬러, 10클래스)

두 데이터셋 모두 이미지 분류 작업으로, 트랜스포머가 자연어 외에 시각적 데이터에 적용될 수 있음을 평가하기 적합하다.

* 모델 설정

- 입력 이미지는 패치 단위로 나누어 임베딩한 후 트랜스포머에 입력
- 트랜스포머 블록 수 : 실험에 따라 1~4개
- 하이퍼파라미터 : 학습률, 배치 사이즈 등은 동일하게 설정

4. 실험(Experiments)

4.2 실험 조건 및 비교 모델

우리는 다음 네 가지 아키텍처 구성을 비교 대상으로 실험하였다:

모델 번호	구성 설명
Full Transformer (Baseline)	전통적인 트랜스포머: 어텐션 + MLP
No-MLP	MLP 제거, 어텐션만 사용
Q=K=V 통합	쿼리, 키, 값의 하나의 공통 행렬로 생성
대칭 어텐션	QK^T 대신 대칭 유사도 계산 사용

이 외에도 모든 수정 사항을 동시에 적용한 최소 아키텍처도 별도로 실험하였다.

4. 실험(Experiments)

4.3 결과 및 분석

◆ 정확도 (Accuracy)

* MNIST

- Full Transformer: 약 99.3% Minimal Transformer: 약 99.2%

* CIFAR-10

- Full Transformer: 약 71.5% Minimal Transformer: 약 70.8%

→ 모든 수정 사항을 적용한 간소화 모델도 거의 동등한 성능을 유지함을 확인됨.

◆ 파라미터 수 (Parameter Count)

- 기존 트랜스포머 대비, 최대 90%까지 파라미터 수 절감
- 특히 MLP와 Q/K/V 통합에서 절감 효과가 큼

◆ 학습 속도 및 수렴 (Convergence)

- 간소화 모델은 더 빠르게 수렴함
- 오버피팅 경향도 적음, 더 일반화된 결과를 보임.

4. 실험(Experiments)

4.4 핵심 요약

- MLP는 반드시 필요하지 않다: 제거해도 성능 유지
- $Q=K=V$ 통합: 성능 유지되며 파라미터 절감 효과 큼
- 대칭 유사도 계산: 안정적인 수렴 및 계산량 절감
- 최소 구성의 트랜스포머도 실전에서 충분히 효과적

5. 결론(Conclusion)

본 연구는 트랜스포머 아키텍처에서 흔히 사용되는 구성 요소들 중 일부가 반드시 필요하지 않을 수도 있다는 가능성을 실험적으로 제시했다.

핵심 결론 요약

- MLP 제거

트랜스포머 블록에서 MLP를 완전히 제거해도, 다양한 데이터셋에서 성능 손실 없이 분류 정확도를 유지할 수 있음을 확인했다.

- 쿼리/키/값(Q, K, V)의 통합

세 개의 서로 다른 가중치 행렬을 사용하는 대신, 하나의 공통 행렬을 통해 생성함으로써 모델을 간단하게 만들 수 있고, 이 경우에도 성능 저하 없이 작동할 수 있다.

- 대칭적인 유사도 계산 방식

기존의 비대칭적인 어텐션 유사도 계산 대신, 대칭 구조를 적용하여 더 안정적인 수렴과 계산량 절감 효과를 확인했다.

5. 결론(Conclusion)

💡 연구의 의의

이러한 실험 결과들은 다음과 같은 중요한 시사점을 제공합니다:

1. 모델의 최소화 가능성

트랜스포머 구조는 본래 생각했던 것보다 훨씬 단순화 가능하며, 복잡한 구조를 유지하지 않더라도 고성능을 유지할 수 있습니다.

2. 해석 가능한 구조로의 전환

구성 요소가 줄어들수록 모델이 어떤 방식으로 학습하는지 해석이 용이해지며, 이는 신뢰성과 안정성이 중요한 분야에서 더 큰 장점이 됩니다.

3. 자원 제약 환경에서의 활용 가능성

파라미터 수와 연산량이 감소한 간소화된 트랜스포머는 엣지 디바이스, 모바일 환경, 저전력 시스템등에서도 실용적으로 활용될 수 있는 가능성을 보여줍니다.

5. 결론(Conclusion)

향후 연구 방향

- 더욱 복잡한 데이터셋(예: ImageNet, 자연어 처리 코퍼스)에서의 성능 검증
- 다중 레이어 구조에서도 이러한 최소화 전략이 효과적인지에 대한 탐구
- 어텐션 구조 자체를 더 간결하게 만드는 이론적 분석 및 최적화

이 논문은 트랜스포머를 성능 저하 없이 단순화할 수 있다는 실험적 증거를 제시하며, 향후 다양한 연구자들이 경량 딥러닝 모델 설계에 참고할 수 있는 기초를 마련합니다.

6. 수식 설명

1. 수식 (1):

$$z_{si} = \left(\sum_{j=1}^i a_{sij} x_{sj} \right) W_s^V W_s^O$$

- 이 수식은 입력 임베딩 x_{si} 의 가중 평균을 구하는 방식입니다. x_{si} 는 각 단어 임베딩을 의미하며, W_s^V 와 W_s^O 는 각각 벡터 공간 변환을 위한 행렬입니다. 각 입력의 가중치는 a_{si} 로 나타내며, 이 가중치는 뒤에서 설명할 Softmax 함수로 계산됩니다.

2. 수식 (2):

$$a_{si} = \text{Softmax}(s_{si})$$

- a_{si} 는 Softmax 함수를 통해 계산되는 가중치 벡터입니다. Softmax 함수는 각 단어의 중요도를 결정하며, s_{si} 는 특정 쿼리 토큰과 다른 토큰 간의 유사도를 나타내는 값입니다. 이를 통해 가중치를 계산합니다.

3. 수식 (3):

$$s_{sij} = x_{si} W^Q W^K^T x_{sj}$$

- s_{sij} 는 쿼리 토큰 x_{si} 와 다른 토큰 x_{sj} 간의 유사도를 나타냅니다. 여기서 W_s^Q 와 W_s^K 는 각각 쿼리와 키 변환 행렬이며, 이 행렬들의 내적을 통해 유사도를 계산합니다.

6. 수식 설명

4. 수식 (4):

$$z_{si} = \sum_{h=1}^H \left(\sum_{j=1}^i a_{shij} x_{sj} \right) W_{sh}^V W_{sh}^O$$

- 이 수식은 "멀티 헤드 어텐션"에서 사용됩니다. 여러 헤드를 사용하여 각각 독립적으로 계산한 후 결과를 합산합니다. 각 헤드는 $W_h^Q, W_h^K, W_h^V, W_h^O$ 라는 별도의 변환 행렬을 사용하여 계산됩니다.

5. 수식 (5):

$$a_{shi} = \text{Softmax}(s_{shi})$$

- 각 헤드에서의 가중치 a_{shi} 는 Softmax 함수로 계산됩니다. s_{shi} 는 각 헤드에 대해 계산된 유사도 값으로, 이를 통해 해당 헤드의 중요도를 결정합니다.

6. 수식 설명

수식 (6):

$$s_{shij} = x_{si} W_{sh}^Q W_{sh}^K x_{sj}^T$$

- 이 수식은 특정 쿼리 토큰 x_{si} 와 다른 토큰 x_{sj} 간의 유사도를 계산하는 수식입니다. 여기서 W_{sh}^Q 와 W_{sh}^K 는 각각 쿼리와 키 변환 행렬로, 이들을 사용하여 두 토큰 간의 유사도를 계산합니다.

수식 (7):

$$h_{si} = f(z_{si} W_s^{(1)} + b_s^{(1)})$$

- 이 수식은 다층 퍼셉트론(Multi-Layer Perceptron, MLP)의 첫 번째 층을 나타냅니다. z_{si} 는 중간 임베딩 벡터로, 이 벡터에 대해 가중치 행렬 $W_s^{(1)}$ 과 편향 벡터 $b_s^{(1)}$ 를 적용하고, 그 결과에 비선형 함수 $f()$ 를 적용하여 h_{si} 를 계산합니다. 이때 비선형 함수 $f()$ 는 일반적으로 Gaussian Error Linear Unit (GELU)가 사용됩니다.

수식 (8):

$$y_{si} = h_{si} W_s^{(2)} + b_s^{(2)}$$

- 이 수식은 다층 퍼셉트론의 두 번째 층을 나타냅니다. 첫 번째 층에서 계산된 h_{si} 에 대해 다시 가중치 행렬 $W_s^{(2)}$ 과 편향 벡터 $b_s^{(2)}$ 를 적용하여 최종 출력 벡터 y_{si} 를 계산합니다.

이 수식들은 다층 퍼셉트론(MLP) 구조를 설명하며, 주로 신경망에서 사용되는 기본적인 층 연산을 보여줍니다. $f()$ 는 비선형 활성화 함수이며, 보통 GELU 함수가 사용됩니다.

6. 수식 설명

■ 멀티 헤드 어텐션(Multi-Head Attention).

- 스택의 모든 트랜스포머에 대해 다음과 같은 처리는 어텐션 메커니즘(멀티 헤드 어텐션 또는 MHA)에 의해 수행됩니다.
- 스택의 s 번째 트랜스포머에서 훈련 샘플의 입력(전체 개수 S 중)은 입력 벡터 x_{si} 의 시퀀스입니다.
- 이 시퀀스는 동일하게 긴 출력 임베딩 z_{si} 시퀀스로 변환됩니다.

6. 수식 설명

수식은 트랜스포머 인코더 레이어의 연결을 나타내며, 아래와 같은 방식으로 구성됩니다:

1. 수식 (8):

$$y_{1i} = \left(\sum_{j=1}^i a_{1ij} x_{1j} \right) W_1^{VO}$$

- y_{1i} 는 첫 번째 출력 값입니다. 이 값은 각 입력 값에 대해 가중치 a_{1ij} 를 곱하고, 이를 W_1^{VO} 행렬로 변환한 결과입니다.

2. 수식 (9):

$$y_{2i} = \left(\sum_{j=1}^i a_{2ij} y_{1j} \right) W_2^{VO}$$

- y_{2i} 는 두 번째 출력 값입니다. 첫 번째 출력인 y_{1i} 에 대해 다시 가중치 a_{2ij} 를 곱한 후, 이를 W_2^{VO} 행렬로 변환한 결과입니다.

3. 수식 (10):

$$y_{3i} = \left(\sum_{k=1}^i a_{2ik} \left(\sum_{j=1}^k a_{1kj} x_{1j} \right) \right) W_1^{VO} W_2^{VO}$$

- y_{3i} 는 세 번째 출력 값입니다. 두 개의 합성 과정이 포함되어 있으며, 첫 번째 단계에서 x_{1j} 에 대해 가중치 a_{1kj} 를 곱하고, 두 번째 단계에서 이 값에 대해 a_{2ik} 가중치를 곱한 후, $W_1^{VO} W_2^{VO}$ 행렬로 변환합니다.

4. 결과:

- 이 수식은 여러 레이어가 순차적으로 연결된 형태입니다. 최종 출력은 여러 번의 연산을 통해 가중치를 적용하고, 마지막으로 $W_1^{VO} W_2^{VO}$ 행렬을 통해 변환된 값입니다.

6. 수식 설명

3. 싱글 헤드 구성

When stacking the attention modules, the matrices W_s^{VO} concatenate to their product over $s = 1, \dots, S$. Then, they collapse into a single matrix

$$W^{VO} = \prod_{s=1}^S W_s^{VO} \quad (9)$$

Since every sum $\sum_{j=1}^i a_{sij}$ is equal to unity (as a result of the softmax operation), every successive transformer layer performs a weighted mean of stacked inputs x_{1j} .

The total number of parameters with S matrices W_s^{OK} and a single matrix W^{VO} is $(S+1)N^2$, only slightly more than 25% of the original size without MLP. So far, all this is possible without losing any expressive power of the single-head transformer without MLP — only obsolete parameters are deleted.

"어텐션 모듈을 쌓을 때, 행렬 $W_s^V O_s$ 는 $s = 1, \dots, S$ 에 대해 그들의 곱으로 이어집니다. 그런 다음 이들은 단일 행렬로 축소됩니다.

$$W^V O = \prod_{s=1}^S W_s^V O_s \quad (9)$$

각각의 합 $\sum_{j=1}^i a_{sij}$ 가 소프트맥스 연산의 결과로 1이므로, 각 후속 변환 층은 쌓인 입력 x_{1j} 의 가중 평균을 수행합니다.

$W^Q K_s$ 와 단일 행렬 $W^V O$ 로 이루어진 파라미터의 총 수는 $(S+1)N^2$ 로, MLP 없이 원래 크기의 25%보다 조금 더 많습니다. 지금까지 모든 것이 단일 헤드 트랜스포머에서 MLP 없이도 가능한 것으로, MLP는 단지 불필요한 파라미터들만 삭제되었습니다."

수식 설명:

- 수식 (9):

$$W^V O = \prod_{s=1}^S W_s^V O_s$$

- 이 수식은 S 개의 어텐션 모듈을 쌓은 후, 이들 행렬 $W_s^V O_s$ 가 하나의 행렬로 축소되는 과정을 나타냅니다. 각 어텐션 모듈이 가질 수 있는 출력 행렬 $W^V O_s$ 는 최종적으로 전체 트랜스포머 모델에 통합되어 $W^V O$ 라는 단일 행렬을 형성합니다. 이 과정을 통해 트랜스포머의 여러 레이어에서 발생하는 파라미터가 축소되고, 모델의 복잡성을 줄이면서도 효과적인 표현을 유지할 수 있습니다.

추가 설명:

- **소프트맥스 연산:** 각 어텐션 가중치 a_{sij} 는 소프트맥스 연산에 의해 결정됩니다. 이 연산의 결과로 가중치의 합은 항상 1이 되며, 이는 각 입력 토큰에 대한 가중 평균을 계산하는 데 사용됩니다.
- **파라미터 수:** 트랜스포머에서의 파라미터 수는 $W^Q K_s$ 와 $W^V O$ 를 포함하여 계산됩니다. MLP를 제외한 트랜스포머에서는 파라미터 수가 원래 크기의 약 25%로 축소됩니다. 이는 모델의 효율성을 높이며, 더 적은 파라미터로도 동일한 성능을 유지할 수 있음을 의미합니다.

6. 수식 설명

$$z_{si} = \sum_{j=1}^i a_{sij} x_{sj} \quad (10)$$

The output embedding z_{si} is a convex combination of input embeddings x_{1i} . In other words, it is a member of the convex set spanned by x_{1i} .

This concept has been implemented in the Keras framework by setting the matrices W_s^V and W_s^O to unit matrices. Collapsing $W_s^Q W_s^{KT}$ to W_s^{QK} has been reached by setting the matrix W^K to a unit matrix. The newly defined matrix W_s^{QK} replaces matrix W_s^Q .

"출력 임베딩 z_{si} 는 입력 임베딩 x_{1i} 들의 볼록 조합입니다. 다시 말해, z_{si} 는 x_{1i} 로 펼쳐진 볼록 집합의 구성 원입니다.

이 개념은 Keras 프레임워크에서 행렬 W_s^V 와 W_s^O 를 단위 행렬로 설정하여 구현되었습니다. $W^S W_s^{KT}$ 를 W^S 로 축소하는 것은 행렬 W^K 를 단위 행렬로 설정하여 달성되었습니다. 새롭게 정의된 행렬 $W^S QK$ 는 행렬 W_s^Q 를 대체합니다."

수식 설명:

- 수식 (10):

$$z_{si} = \sum_{j=1}^i a_{sij} x_{sj}$$

- 이 수식은 출력 임베딩 z_{si} 가 입력 임베딩들 x_{sj} 의 가중 평균으로 계산된다는 것을 나타냅니다. 가중치 a_{sij} 는 각 입력 임베딩에 대해 계산된 가중치를 의미합니다. 이는 어텐션 메커니즘에서 입력 임베딩의 중요한 특징을 반영하는 값입니다.

추가 설명:

- **볼록 조합:** z_{si} 는 입력 임베딩들의 가중 평균이며, 이들은 특정 벡터 x_{1i} 로 펼쳐지는 볼록 집합에 속합니다. 이는 트랜스포머에서 입력 정보의 결합 방법을 설명하는 중요한 개념입니다.
- **Keras 구현:** 이 아이디어는 Keras 프레임워크에서 구현되어 있으며, 행렬을 단위 행렬로 설정하는 방식으로 최적화됩니다. $W^S W_s^{KT}$ 는 단순화되어 W^S 로 축소되며, 새롭게 정의된 행렬 $W^S QK$ 는 이전의 W_s^Q 를 대체합니다.

6. 수식 설명

4. MULTI-HEAD CONFIGURATION

The relationships of Section 3 are valid wherever the matrices W_{sh}^V , W_{sh}^O , W_{sh}^Q , and W_{sh}^K are square. This may also apply to multiple heads. However, it is usual to commit to a reduced dimension per head. With $H > 1$ heads, it is common to map the embedding vector to a narrower vector of width N/H , assumed to be integer.

In such cases, the matrices W_{sh}^V , W_{sh}^O , W_{sh}^Q , and W_{sh}^K are not square but of dimension $(N, N/H)$. Collapsing $W_{sh}^Q W_{sh}^{KT}$ to W_{sh}^{QK} is then no longer efficient since W_{sh}^{QK} is of dimension (N, N) and has thus N^2 parameters while W_{sh}^Q and W_{sh}^K together have $2N^2/H$, which is a smaller or equal number for $H > 1$.

Moreover, it is impossible to equivalently concatenate the value/projection matrices W_{sh}^{VO} to a unique product because of varying index h along various paths through the heads.

Nevertheless, omitting the W_{sh}^{VO} at all would have the same justification as for single-head configuration: the output embedding z_{Si} would become a convex combination of input embeddings x_{1i} , which can be expected to correspond to a meaningful word or token.

"섹션 3의 관계는 행렬 $W_{sh}^V, W_{sh}^O, W_{sh}^Q, W_{sh}^K$ 가 정사각형일 때 유효합니다. 이는 멀티-헤드에도 적용됩니다. 그러나 각 헤드마다 축소된 차원에 맞추는 것이 일반적입니다. $H > 1$ 일 경우, 임베딩 벡터를 N/H 너비를 가진 더 좁은 벡터로 매핑하는 것이 일반적이며, 이는 정수로 가정됩니다.

이러한 경우, 행렬 $W_{sh}^V, W_{sh}^O, W_{sh}^Q, W_{sh}^K$ 는 정사각형이 아니고, 차원은 $(N, N/H)$ 입니다. $W_{sh}^Q W_{sh}^{KT} h$ 를 W_{sh}^{QK} 로 축소하는 것은 더 이상 효율적이지 않습니다. 왜냐하면 W_{sh}^{QK} 의 차원이 (N, N) 이고, 따라서 N^2 개의 파라미터를 가지게 됩니다. 반면에 W_{sh}^Q 와 W_{sh}^K 는 $2N^2/H$ 개의 파라미터를 가지고, 이는 $H > 1$ 일 때 더 적은 숫자입니다.

또한, 헤드마다 다양한 경로를 통해 인덱스 h 가 달라지기 때문에, 값/프로젝션 행렬 $W^V O_{sh}$ 를 하나의 곱으로 동등하게 연결하는 것은 불가능합니다.

그럼에도 불구하고, W_{sh}^V 를 전혀 생략하면 단일 헤드 구성과 동일한 정당성을 가집니다. 출력 임베딩 z_{Si} 는 입력 임베딩 x_{1i} 의 볼록 조합이 되며, 이는 의미 있는 단어 또는 토큰에 해당하는 것으로 예상될 수 있습니다."

6. 수식 설명

5. SYMMETRY OF SIMILARITY

The expression Eq. (3) measures the similarity between queries and keys. The general concept of characterizing similarity between vectors by their product is symmetric: a is equally similar to b as is b to a .

However, the similarity between a key and a query evaluated with the help of $x_{si}W_{sh}^QW_{sh}^{KT}x_{sj}^T$ is asymmetric. This is because the matrices W_{sh}^Q and W_{sh}^K are potentially different.

This asymmetry leads to different similarities between x_{si} and x_{sj} in the roles of key and query: x_{si} is not as similar to x_{sj} as is x_{sj} to x_{si} . The vector x_{si} is also not the most similar to itself. The matrix product $W_{sh}^QW_{sh}^{KT}$ is generally not positive definite, so it is not even guaranteed that the similarity of x_{si} to itself is positive.

The asymmetry can be deliberate and justified from some viewpoints. It is not a matter of course that the roles of queries and keys are symmetric. However, some of the mentioned properties can make its use harmful.

"식 (3)은 쿼리와 키 사이의 유사도를 측정합니다. 벡터 간 유사성을 그들의 내적으로 특징짓는 일반적인 개념은 대칭적입니다: a 는 b 와 마찬가지로 b 는 a 와 동일하게 유사합니다.

그러나 쿼리와 키 간의 유사도는 $x_{si}W_{sh}^QW_{sh}^{KT}x_{sj}^T$ 를 통해 평가되며, 이는 비대칭적입니다. 그 이유는 행렬 W_{sh}^Q 와 W_{sh}^K 가 본질적으로 다르기 때문입니다.

이 비대칭성은 키와 쿼리 역할을 하는 x_{si} 와 x_{sj} 간에 다른 유사도를 초래합니다: x_{si} 는 x_{sj} 와 비교할 때 x_{sj} 만큼 유사하지 않습니다. 벡터 x_{si} 는 자신에게 가장 유사하지도 않습니다. 행렬 곱 $W_{sh}^QW_{sh}^{KT}$ 는 일반적으로 양의 정부호가 아니므로, x_{si} 와 자신 간의 유사도가 양의 값이 될 것이라고 보장되지 않습니다.

이 비대칭성은 일부 관점에서 의도적으로 정당화될 수 있습니다. 쿼리와 키 역할이 대칭적이어야 한다는 것은 당연한 문제가 아닙니다. 그러나 앞서 언급된 몇 가지 특성은 그 사용을 해롭게 만들 수 있습니다."

설명:

이 텍스트는 트랜스포머 모델에서 쿼리와 키 간의 유사도를 계산하는 방식에 대한 설명입니다. 일반적으로 벡터 간의 유사도는 대칭적이어야 하지만, 트랜스포머의 경우 쿼리와 키 간의 유사도는 비대칭적이라는 점을 강조합니다. 행렬 W_{sh}^Q 와 W_{sh}^K 의 비대칭성으로 인해 벡터 간 유사도가 다르게 계산되며, 이로 인해 유사도가 자기 자신과 비교할 때 반드시 양수가 되지 않는 문제도 발생할 수 있습니다.

6. 수식 설명

The symmetry can be guaranteed by simply setting $W_{sh}^Q = W_{sh}^K$. Then, half of the parameters dedicated to the query and key matrices can be economized. In the single-head case, the same effect is reached by a symmetric matrix W_s^{QK} , with identical parameters mirrored over the diagonal, i.e., $w_{sij}^{QK} = w_{sji}^{QK}$. Another possibility is to parameterize a lower triangular matrix T_s^{QK} and to multiply it by its transpose, getting

$$W_s^{QK} = T_s^{QK} T_s^{QKT} \quad (11)$$

This amounts to the well-known *Cholesky decomposition* (Cholesky, 1924) of a symmetric matrix.

With both methods, the number of parameters is $\frac{N(N+1)}{2}$ instead of N^2 , or even $2N^2$ of the original version without collapsing W^Q and W^K .

The symmetry is implemented by reusing W_{sh}^Q as W_{sh}^K , omitting the use of W_{sh}^K at all.

"대칭성은 $W_s^Q = W_s^K$ 로 설정함으로써 보장할 수 있습니다. 그러면 쿼리와 키 행렬에 할당된 파라미터의 절반을 절약할 수 있습니다. 단일 헤드의 경우, 동일한 효과는 대칭 행렬 W_s^{QK} 로 달성되며, 대각선을 따라 동일한 파라미터 $w_{sij}^Q = w_{sji}^Q$ 가 있습니다. 또 다른 가능성은 하삼각 행렬 T_s^Q 를 파라미터화하고 그것을 전치 행렬과 곱하는 것입니다. 이렇게 하면 다음과 같습니다:

$$W_s^Q = T_s^Q T_s^{QT}$$

이것은 대칭 행렬의 잘 알려진 쉐레스키 분해(Cholesky, 1924)에 해당합니다.

두 방법 모두에서 파라미터 수는 $\frac{N(N+1)}{2}$ 로, 원래 버전의 N^2 또는 심지어 $2N^2$ 보다 적습니다. 이는 W^Q 와 W^K 를 축소하지 않고도 가능합니다.

대칭성은 W_s^Q 와 W_s^K 를 재사용하고, W_s^K 의 사용을 생략하여 구현됩니다."

설명:

이 텍스트는 트랜스포머 모델에서 쿼리와 키 행렬의 대칭성을 구현하는 방법을 설명하고 있습니다. W_s^Q 와 W_s^K 를 동일하게 설정하여 파라미터 수를 절감할 수 있는 방법을 제시하고, 이를 통해 효율적인 모델을 설계하는 방법을 다룹니다. 쉐레스키 분해를 사용하여 행렬을 파라미터화하고, 이로 인해 파라미터 수를 줄일 수 있는 방법을 설명합니다.

6. 수식 설명

6.1 Results for MNIST

Following the arguments of Sections 2 to 5, the following reduced transformer variants have been tested:

- with and without an MLP in each transformer-encoder,
- with 1 and 4 heads,
- with the original matrix configuration as well matrix pair W^Q and W^K collapsed into one matrix, W^V and W^O omitted (one head variants only), and

"6.1 MNIST에 대한 결과

섹션 2부터 5까지의 주장을 따라, 다음과 같은 축소된 트랜스포머 변형들이 테스트되었습니다:

- 각 트랜스포머 인코더에서 MLP가 있는 경우와 없는 경우,
- 1개 및 4개의 헤드를 사용하는 경우,
- 원래의 행렬 구성과 W^Q 및 W^K 행렬 쌍을 하나의 행렬로 축소하고, W^V 와 W^O 를 생략한 경우(단일 헤드 변형만 해당).

6. 수식 설명

표 1: 6개 또는 12개의 연속된 트랜스포머 인코더와 각 인코더 레이어당 1개 또는 4개의 어텐션 헤드를 사용한 두 데이터셋(MNIST 및 CIFAR-10)에서의 16개 실험 결과. 각 인코더 레이어 내에서 기본 MLP를 사용하거나 MLP를 완전히 생략한 경우의 실험. 각 모델은 정확히 500 에폭 동안 훈련된 후 훈련 세트와 검증 세트에 대한 손실과 정확도가 보고됩니다.

Dataset	#Encs-#Heads	MLP?	Q	Train loss	Val. loss	Train. acc. [%]	Val. acc. [%]
MNIST	6-1	yes	2.15	0.0067	0.0747	99.78	98.38
	6-1	no	6.46	0.0277	0.1023	99.07	97.49
	6-4	yes	2.09	0.0018	0.0739	99.95	98.26
	6-4	no	5.91	0.0021	0.0912	99.92	98.29
	12-1	yes	1.08	0.0052	0.0652	99.81	98.71
	12-1	no	3.29	0.0117	0.0970	99.62	97.94
	12-4	yes	1.08	0.0025	0.0656	99.92	98.70
	12-4	no	3.29	0.0026	0.1002	99.93	98.10
CIFAR-10	6-1	yes	1.74	0.1533	2.2418	94.63	60.24
	6-1	no	4.93	0.9341	1.3590	66.16	55.30
	6-4	yes	1.74	0.1109	2.4033	96.01	60.46
	6-4	no	4.92	0.5621	1.6984	80.82	52.37
	12-1	yes	0.89	2.3026	2.3026	9.82	10.00
	12-1	no	2.52	0.5604	1.7219	79.48	54.06
	12-4	yes	0.89	0.0632	2.6379	97.92	58.02
	12-4	no	2.52	0.1787	2.3200	93.59	55.60

6. 수식 설명

그림 1: MNIST에서 6개의 인코더 레이어를 사용하는 다양한 축소된 트랜스포머 인코더로 얻은 훈련 및 검증 손실.

- 가로축: 다양한 트랜스포머 인코더 변형을 나타내며, 각 레이블은 해당 모델 설정을 나타냅니다. 예를 들어, "1H/MLP/unchanged"는 1개의 헤드와 MLP가 있는 변경되지 않은 설정을 의미합니다.
- "1H/NoMLP/PWqk+noWvVo"는 1개의 헤드와 MLP가 없으며, 일부 행렬이 변경된 설정을 의미합니다.
- 세로축: 손실(Loss) 값이며, 훈련 손실(녹색 바)과 검증 손실(파란색 바)로 나누어 표시됩니다.
- 훈련 손실과 검증 손실: 각 설정에 대해 훈련 손실과 검증 손실이 비교됩니다. 훈련 손실은 일반적으로 모델이 학습하는 동안 점차적으로 감소하는 경향이 있으며, 검증 손실은 모델의 일반화 성능을 평가하는 데 중요한 지표입니다.

6. 수식 설명

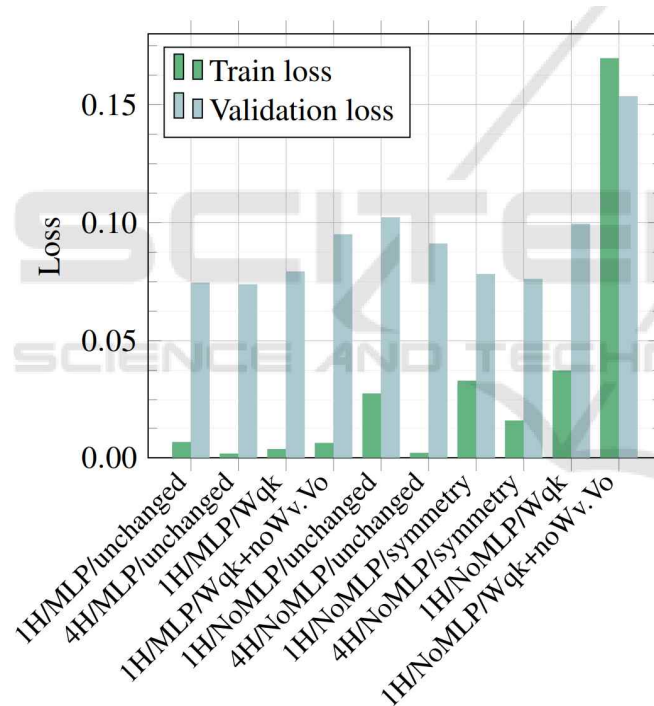


Figure 1: Training and validation losses attained by various reduced transformer-encoders with six encoder layers on MNIST.

- 비대칭적 및 대칭적 유사도 측정을 사용하는 경우.
- 나타낸 변형들은 행렬 옵션에 해당합니다:
 - **unchanged**는 원래의 어텐션 모듈 행렬 변형에 해당합니다.
 - **Wqk** 변형은 곱셈에 대해 단일 행렬을 사용합니다: $W^Q W^{KT}$; 이 변형은 단일 어텐션 헤드에만 사용 가능하며, 그들의 유사도 측정은 원래 버전처럼 비대칭적입니다.

6. 수식 설명

Table 2: Loss and accuracy for different variants of transformer-encoder modifications on MNIST: 1 or 4 heads, with or without the MLP, with a single W_{qk} matrix, no value and projection matrices, or a symmetric similarity measurement.

# Heads	MLP?	Modification	# Parameters	Q	Train loss	Val. loss	Train. acc. [%]	Val. acc. [%]
1	yes	unchanged	279,106	2.15	0.0067	0.0747	99.78	98.38
4	yes	unchanged	287,746	2.09	0.0018	0.0739	99.95	98.26
1	yes	Wqk	257,506	2.33	0.0037	0.0794	99.89	98.43
1	yes	Wqk+noWv,Vo	212,866	2.82	0.0063	0.0951	99.78	98.27
1	no	unchanged	92,890	6.46	0.0277	0.1023	99.07	97.49
4	no	unchanged	101,530	5.91	0.0021	0.0912	99.92	98.29
1	no	symmetry	69,910	8.58	0.0331	0.0783	98.85	97.80
4	no	symmetry	69,910	8.58	0.0158	0.0762	99.46	98.24
1	no	Wqk	70,570	8.50	0.0374	0.0996	98.70	97.60
1	no	Wqk+noWv,Vo	26,650	22.51	0.1697	0.1536	94.82	95.32

6. 수식 설명

설명:

- 헤드 수 (# Heads): 모델이 사용하는 어텐션 헤드의 수입니다. 실험에서는 1개 또는 4개의 헤드가 사용되었다.
- MLP? : 각 트랜스포머 인코더 레이어에서 MLP(다층 퍼셉트론) 사용 여부를 나타냅니다. "예"는 MLP를 사용한 모델을, "아니오"는 MLP를 생략한 모델을 의미합니다.
- 변형: 모델에서 사용된 특정 변형을 나타냅니다. 예를 들어, "Wqk"는 쿼리와 키 행렬을 단일 행렬로 사용하는 변형을 의미하고, "대칭성"은 대칭적 유사도 측정을 사용한 변형을 의미합니다.
- 파라미터 수: 모델에서 사용된 총 파라미터 수입니다.
- Q: 과다 결정 비율을 나타내며, 모델이 얼마나 충분히 결정되었는지의 척도를 제공합니다.
- 훈련 손실 (Train loss): 훈련 데이터에서 계산된 손실 값입니다. 모델이 훈련 세트에서 얼마나 잘 학습되었는지 나타냅니다.
- 검증 손실 (Val. loss): 검증 데이터에서 계산된 손실 값입니다. 모델의 일반화 성능 이다.
- 훈련 정확도 (Train acc.): 훈련 세트에서 모델의 정확도입니다.
- 검증 정확도 (Val. acc.): 검증 세트에서 모델의 정확도입니다.

6. 수식 설명

# 헤드	MLP?	변형	파라미터 수	Q	훈련 손실	검증 손실	훈련 정확도 [%]	검증 정확도 [%]
1	예	변경되지 않음	279,106	2.15	0.0067	0.0747	99.78	98.38
4	예	변경되지 않음	287,746	2.09	0.0018	0.0739	99.95	98.26
1	예	Wqk	257,506	2.33	0.0037	0.0794	99.89	98.43
1	예	Wqk + noWv, Vo	212,866	2.82	0.0063	0.0951	99.78	98.27
1	아니오	변경되지 않음	92,890	6.46	0.0277	0.1023	99.07	97.49
4	아니오	변경되지 않음	101,530	5.91	0.0021	0.0912	99.92	98.29
1	아니오	대칭성	69,910	8.58	0.0331	0.0783	98.85	97.80
4	아니오	대칭성	69,910	8.58	0.0158	0.0762	99.46	98.24
1	아니오	Wqk	70,570	8.50	0.0374	0.0996	98.70	97.60
1	아니오	Wqk + noWv, Vo	26,650	22.55	0.1697	0.1536	94.82	95.32

이 표는 다양한 트랜스포머 변형이 MNIST 데이터셋에서 어떻게 성능을 발휘하는지 비교하고 있습니다. MLP가 있는 모델과 없는 모델의 성능 차이, 그리고 각 변형의 파라미터 수와 성능에 대한 영향을 살펴볼 수 있습니다. MLP가 있는 모델은 일반적으로 더 좋은 성능을 보이지만, MLP가 없을 경우 더 적은 파라미터 수로도 괜찮은 성능을 유지하는 경우도 있음을 알 수 있습니다.

Q & A

