딥러닝 논문 요약 및 구현 스터디

**발표자**

이기환

# Transformer

## - Attention Is All You Need

# 소개

**Transformer?**

: Transformer는 Self-Attention이라는 기법을 사용한 모델

Self-Attention은 기존의 RNN, CNN계열의 신경망 구조를 탈피하고자 하여 등장한 Attention기법

**기존의 한계점**

1) CNN(Convolutional Neural Network)

CNN은 이미지를 처리할 때 주로 사용되는 신경망으로, 설정해 놓은 윈도우들이 차례로 움직이며 특징들을 뽑아냄. 이로 인해 윈도우 내에 있는 언어(local context)에 집중을 할 수 있지만, 윈도우 밖에 존재하는(거리가 먼) 단어의 의미를 파악하는 것은 굉장히 어려움

2) RNN(Recurrent Neural Network)

RNN은 기계번역에 주로 사용되는 신경망으로, 연쇄적인 특징을 이용하여 이전 문장들의 특징을 반영할 수 있음. 하지만 문장의 길이가 길어질 수록 첫 단어의 의미가 끝 단어의 의미까지 반영되기 어려움

**+ 인코더 부분에서 입력 시퀀스를 고정된 크기의 vector로 만들기 때문에 정보를 압축하는 과정에서 손실이 발생 : 입력 시퀀스가 클수록 큰 성능 저하**

# 소개

**문제 상황**

하나의 고정된 문맥 벡터가 소스 문장의 모든 정보를 가지고 있어야 하기에 성능이 저하

**해결 방안**

인코더 부분에서 매번 Hidden state가 나올 때 마다 이 값들을 전부 출력값으로써 그냥 별도의 배열에다가 기록하면 어떨까?

즉, 이를 디코더는 매번 소스 문장에서의 출력 전부를 입력으로 받음

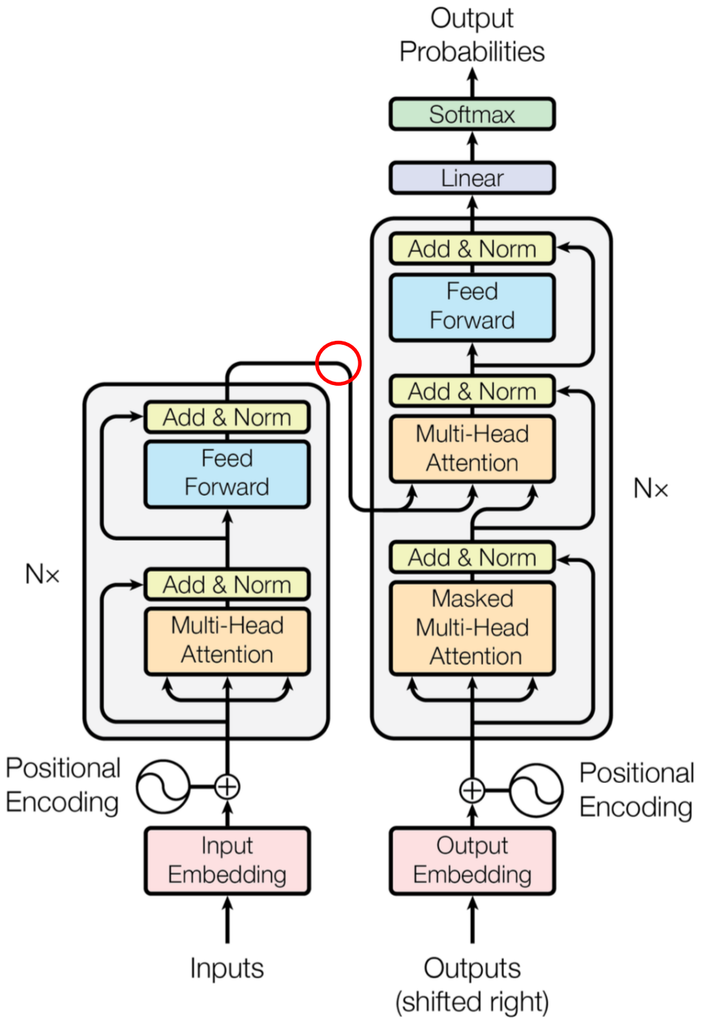



Figure 1: The Transformer - model architecture.

# Abstract

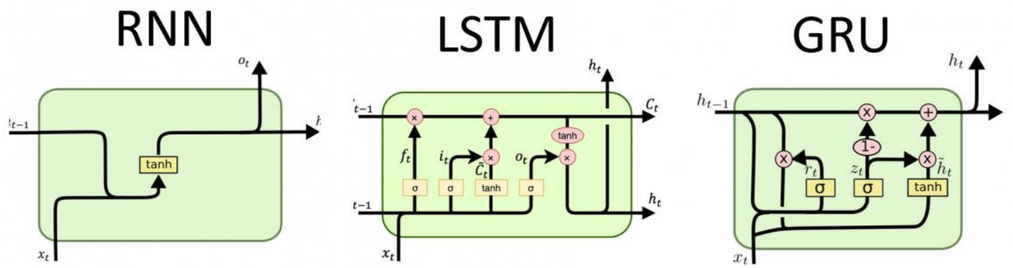

**Abstract**

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.


- **Transformer**
  - 기존에는 인코더 디코더를 포함한 복잡한 RNN CNN 사용
  - 이러한 RNN, CNN을 전혀 쓰지 않고 오직 Attention 매커니즘에만 기반한 모델을 제시
  - Recurrent한 각각의 sequence를 처리할 필요가 없어지기에 단순 행렬곱을 통해 병렬적으로 sequence 데이터 활용

- **모델 성능 :WMT-14**
  - WMT-14년도 데이터를 이용하여 영어를 독일어로, 영어를 불어로 번역하는 task에서 훨씬 성능한 보인 것을 확인할 수 있음
  - 8개의 GPU를 이용해서 상대적으로 적은 시간을 들여 학습을 마칠 수 있음
    + 기계번역 뿐 아니라, sequence 데이터를 처리하는 다양한 데이터에 대해서도 일반화가 가능하고 좋은 성능을 보임

# I. Introduction




이전에 sequence데이터 모델링을 위해 RNN, LSTM, GRU 사용

한번에 한 단어씩 넣는 것처럼 토큰들을 정렬시킨 뒤 이를 반복적으로 넣어서 hiddden state값을 갱신 시키는 방법으로 동작을 함 : 이는 sequence의 길이(토큰의 개수)만큼 입력을 넣어야 하기에 병렬적인 처리를 하기에 어려움 => 메모리 및 속도 측면에서 비효율성을 야기시킴

Reccurence한 특성 자체를 없애기 위해 오직 Attention 매커니즘만을 사용

순차적으로 입력을 넣지 않고, 한 번에 위치 정보가 포함된 전체 sequence를 한번에 처리

=> 병렬적 처리 가능 / 속도 및 성능 향상

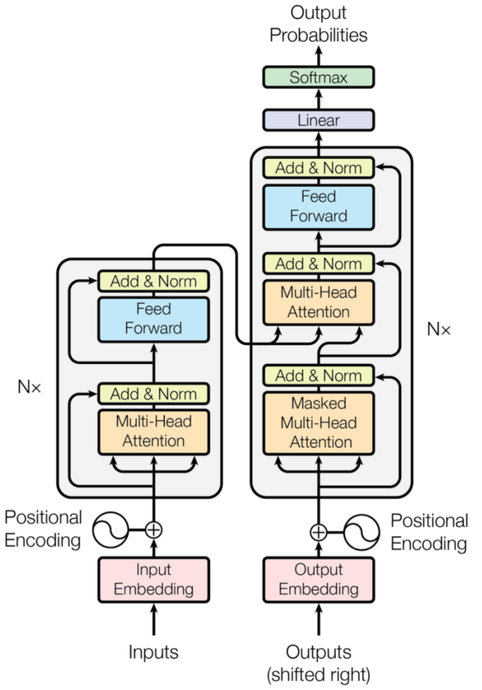



Figure 1: The Transformer - model architecture.

# II.Background

**- Self Attention**

타켓문장을 만들기 위해 소스문장에서의 히든 정보를 참고하는 것이 아니라

자기 자신 문장 스스로에게 어텐션을 수행해서 가중치를 부여 : 단어간 관계의 정도를 확인할 수 있음

Self-attention, sometimes called intra-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence. Self-attention has been used successfully in a variety of tasks including reading comprehension, abstractive summarization, textual entailment and learning task-independent sentence representations [4, 27, 28, 22].


**- Transformer**

Transformer는 시퀀스 정렬 RNN, CNN을 이용하지 않은 전적으로 Attention매커니즘에 의존하기에

시퀀스간의 변형이 가능하게 해줌

# III. Model Architecture

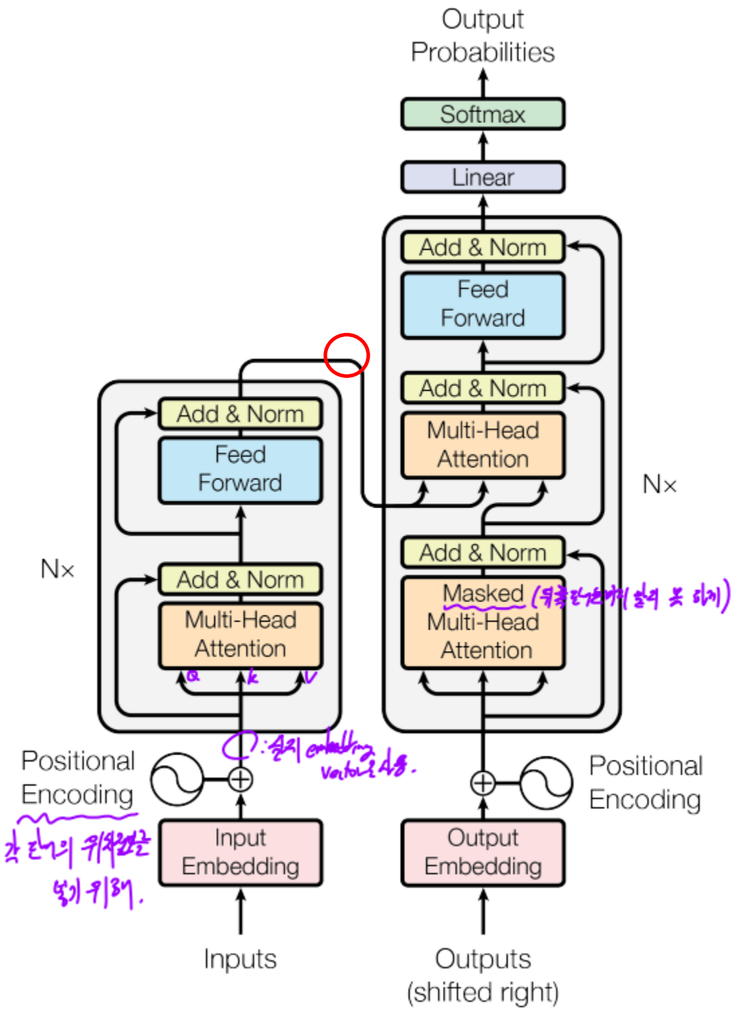



Figure 1: The Transformer - model architecture.

**Encoder Decoder**

N개의 토큰으로 구성된 입력 시퀀스가 있을 때 continuous한 임베딩 벡터로 바꿔주고 이를 디코더는 이전 단계 심볼을 이용해 다음을 예측 후 출력

트랜스포머도 인코더 디코더를 이용하지만 모델을 reccurent하게 이용하지 않고, 시퀀스에 대한 정보를 한번에 입력으로 전달

# III. Model Architecture

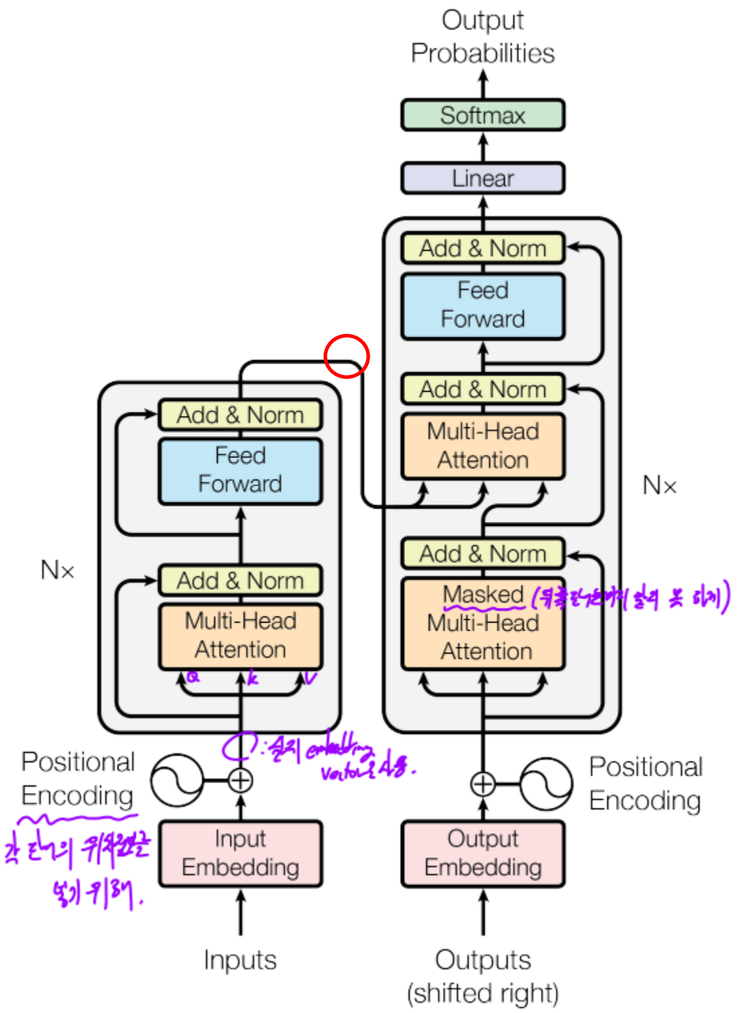



Figure 1: The Transformer - model architecture.

## Encoder Part

rnn을 사용하지 않는 대신 positional encoding을 통한 위치정보 주입 ->이 임베딩 벡터는 Q K V로 각각 복제가 되어 입력 -> Self attetion수행 , residual connection수행 이후 정규화를 시행하고 FeedForward layer를 거친 후 정규화

< 여기서 입력과 출력 모두 dimension은 같음 >

## Decoder Part

Self attention 수행할 때 마스크를 씌워서 뒤쪽 단어를 알 수 없게 만듬 -> 디코더에서도 마찬가지로 self attetion 수행 후 K Q V 사용

-> 디코더의 Q를 이용하여 각각의 출력 단어를 만들기 위해 인코더의 K V를 이용해 어떠한 정보를 참고하면 좋을 지 attetion을 수행(인코더 마지막에 나온 출력 값을 매 인코더 디코더 Attention에서 사용)

-> FF를 거치고 정규화

-> Linear를 거치고 sofmax를 거쳐서 어떤 단어에 해당하는 최종 출력

# III-i. Encoder and Decoder Stacks

**Encoder**

6개의 인코더 레이어 쌓음

정규화 이전 residual connection 이용

임베딩 벡터 차원 : 512

**Decoder**

6개의 디코더 레이어 쌓음 (주로 인코더와 같게 사용)

디코더에서 self attention은 마스크를 씌워 이후 단어를 알 수 없게 만듬 - -무한대를 통해 없앰

마찬가지로 정규화 이전 residual connection 이용

(residual connection을 이용해 학습 난이도를 낮추고 보다 좋은 global optimal을 찾을 수 있음)

# III.ii Attention

**Query :** 질문을 하는 주체

각 Key에 대해 attention score를 구함

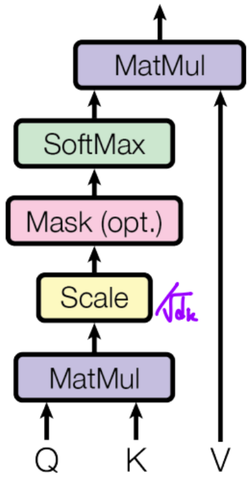
**Key :** 물어보는 대상

Attention을 수행하는 대상

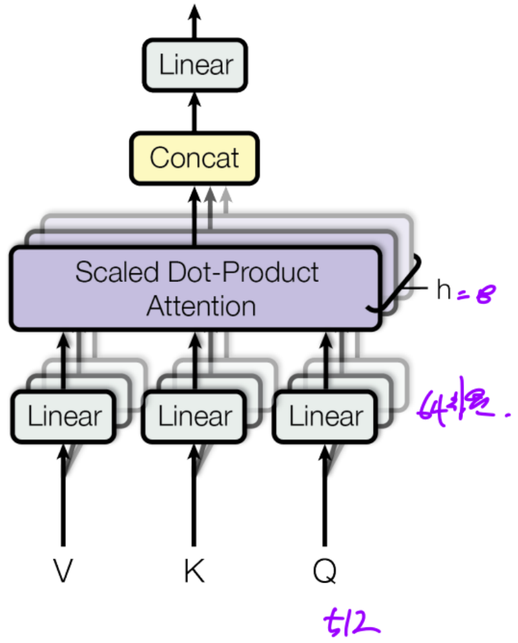
**Value**

Score를 구한 뒤 value값과 곱해서
최종 attention value 값을 도출

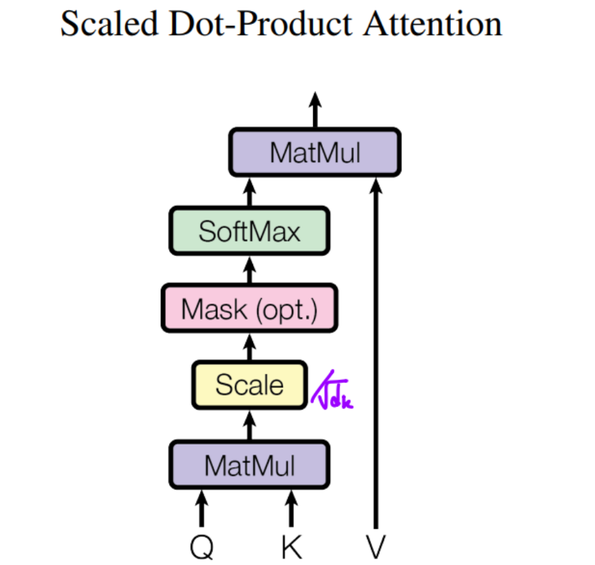
Scaled Dot-Product Attention




Multi-Head Attention



# III.ii.i Scaled Dot-Product Attention




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

**Scaled Dot-Product Attention**

dot( Query, Key )

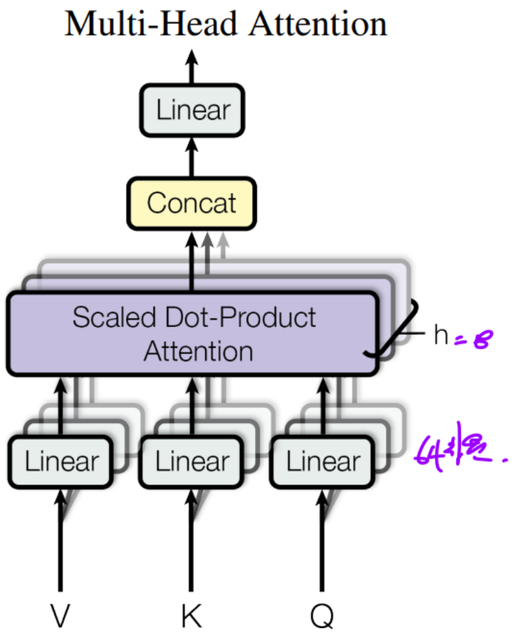
-> /(d_k)**(0.5) 를 통한 정규화

why? 차후 softmax에 대입해야 하는데 값이 너무 커지면 효율적이지 않음

-> 선택에 따라 Mask를 씌우고 softmax에 대입

-> Value값과 곱을 통해 최종 attention value값 도출

# III.ii.ii Muti-Head Attention




$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

**Muti-Head Attention**

각 Attention을 수행한 head들을 concat 후 output을 내보내기 위한 행렬 곱 수행

* d_k = 임베딩벡터 차원 / head개수 = 512 / 8 = 64

# III.v Positional Encoding

**Positional Encoding**

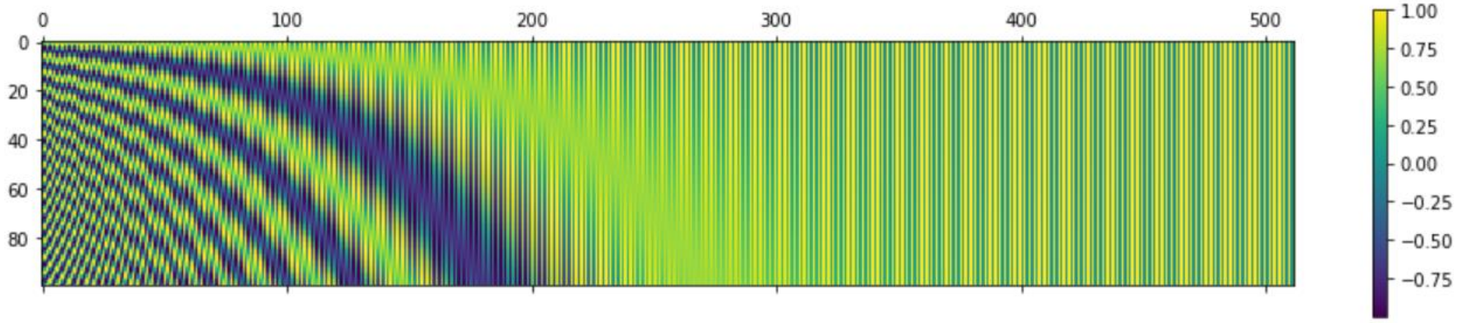
RNN CNN을 모두 사용하지 않기에 위치에 대한 정보를 추출하기 위해 사용

본 논문에서는 sin과 cos을 이용하여 구성

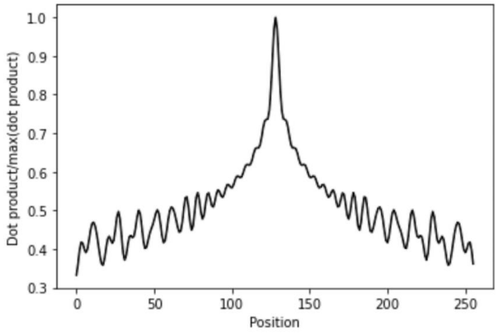
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

* 학습이 가능한 임베딩 레이어도 사용 가능

* Sin cos을 이용한 Positional encoding




* Positional encoding의 위치 유사성



# IV. Why Self-Attention?

1. **Computational complexity per layer**
2. **Amount computation that can be parallelized**
3. **Path length between long-range dependencies**

+ 설명가능하게 해줌 : 어떤 단어를 가장 많이 참고하여 출력하였는지
시각적으로 확인 가능 (self attention에서의 softmax값을 통해)

| Layer Type | Complexity per Layer | Sequential Operations | Maximum Path Length |
|---|---|---|---|
| Self-Attention | $O(n^2 \cdot d)$ (일반적으로 $n<d$) | $O(1)$ : 병렬적. | $O(1)$ |
| Recurrent | $O(n \cdot d^2)$ | $O(n)$ | $O(n)$ |
| Convolutional | $O(k \cdot n \cdot d^2)$ | $O(1)$ | $O(log_k(n))$ |
| Self-Attention (restricted) | $O(r \cdot n \cdot d)$ | $O(1)$ | $O(n/r)$ |

# V. Traing

- WMT-2014

: 영어 -> 독일어 450만개
/ 영어 -> 프랑스어 3600만개

- 8개 P100 GPU사용
- Adam optimizer
- Residual learning 수행 시 Dropout
- Label Smoothing을 통한
특정 출력값에 확신을 가지지 않게 함으로
일반화 효과를 더함

| Model | BLEU | | Training Cost (FLOPs) | |
|---|---|---|---|---|
| | EN-DE | EN-FR | EN-DE | EN-FR |
| ByteNet [18] | 23.75 | | | |
| Deep-Att + PosUnk [39] | | 39.2 | | $1.0 \cdot 10^{20}$ |
| GNMT + RL [38] | 24.6 | 39.92 | $2.3 \cdot 10^{19}$ | $1.4 \cdot 10^{20}$ |
| ConvS2S [9] | 25.16 | 40.46 | $9.6 \cdot 10^{18}$ | $1.5 \cdot 10^{20}$ |
| MoE [32] | 26.03 | 40.56 | $2.0 \cdot 10^{19}$ | $1.2 \cdot 10^{20}$ |
| Deep-Att + PosUnk Ensemble [39] | | 40.4 | | $8.0 \cdot 10^{20}$ |
| GNMT + RL Ensemble [38] | 26.30 | 41.16 | $1.8 \cdot 10^{20}$ | $1.1 \cdot 10^{21}$ |
| ConvS2S Ensemble [9] | 26.36 | **41.29** | $7.7 \cdot 10^{19}$ | $1.2 \cdot 10^{21}$ |
| Transformer (base model) | 27.3 | 38.1 | **$3.3 \cdot 10^{18}$** | |
| Transformer (big) | **28.4** | **41.8** | $2.3 \cdot 10^{19}$ | |

# VI. Results

**- Model Variations 실험**

어떤 component가 중요했는가 : model variation실행 (head의 수를 줄이거나, 특정 param의 수 증가 등)

A) Head의 dim 바꾸기

B) Head와 상관없이 key_dim 줄여보기

C) 모델의 크기를 변형

D) Dropout을 통한 overfitting 방지

E) Sin cos 대신
별도의 embedding layer 사용

| | $N$ | $d_{\text{model}}$ | $d_{\text{ff}}$ | $h$ | $d_k$ | $d_v$ | $P_{drop}$ | $\epsilon_{ls}$ | train steps | PPL (dev) | BLEU (dev) | params $\times 10^6$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| base | 6 | 512 | 2048 | 8 | 64 | 64 | 0.1 | 0.1 | 100K | 4.92 | 25.8 | 65 |
| (A) | | | | 1 | 512 | 512 | | | | 5.29 | 24.9 | |
| | | | | 4 | 128 | 128 | | | | 5.00 | 25.5 | |
| | | | | 16 | 32 | 32 | | | | 4.91 | 25.8 | |
| | | | | 32 | 16 | 16 | | | | 5.01 | 25.4 | |
| (B) | | | | | 16 | | | | | 5.16 | 25.1 | 58 |
| | | | | | 32 | | | | | 5.01 | 25.4 | 60 |
| (C) | 2 | | | | | | | | | 6.11 | 23.7 | 36 |
| | 4 | | | | | | | | | 5.19 | 25.3 | 50 |
| | 8 | | | | | | | | | 4.88 | 25.5 | 80 |
| | | 256 | | | 32 | 32 | | | | 5.75 | 24.5 | 28 |
| | | 1024 | | | 128 | 128 | | | | 4.66 | 26.0 | 168 |
| | | | 1024 | | | | | | | 5.12 | 25.4 | 53 |
| | | | 4096 | | | | | | | 4.75 | 26.2 | 90 |
| (D) | | | | | | | 0.0 | | | 5.77 | 24.6 | |
| | | | | | | | 0.2 | | | 4.95 | 25.5 | |
| | | | | | | | | 0.0 | | 4.67 | 25.3 | |
| | | | | | | | | 0.2 | | 5.47 | 25.7 | |
| (E) | positional embedding instead of sinusoids | | | | | | | | | 4.92 | 25.7 | |
| big | 6 | 1024 | 4096 | 16 | | | 0.3 | | 300K | **4.33** | **26.4** | 213 |

# VI. Results

- **English Consistency Parsing**

: 기계번역 뿐 아니라 다양한 자연어 처리 분야도 좋은 성능을 보임
Ex) 구문 분석 분야Table 4

Table 4: The Transformer generalizes well to English constituency parsing (Results are on Section 23 of WSJ)

| Parser | Training | WSJ 23 F1 |
|---|---|---|
| Vinyals & Kaiser el al. (2014) [37] | WSJ only, discriminative | 88.3 |
| Petrov et al. (2006) [29] | WSJ only, discriminative | 90.4 |
| Zhu et al. (2013) [40] | WSJ only, discriminative | 90.4 |
| Dyer et al. (2016) [8] | WSJ only, discriminative | 91.7 |
| Transformer (4 layers) | WSJ only, discriminative | 91.3 |
| Zhu et al. (2013) [40] | semi-supervised | 91.3 |
| Huang & Harper (2009) [14] | semi-supervised | 91.3 |
| McClosky et al. (2006) [26] | semi-supervised | 92.1 |
| Vinyals & Kaiser el al. (2014) [37] | semi-supervised | 92.1 |
| Transformer (4 layers) | semi-supervised | 92.7 |
| Luong et al. (2015) [23] | multi-task | 93.0 |
| Dyer et al. (2016) [8] | generative | 93.3 |

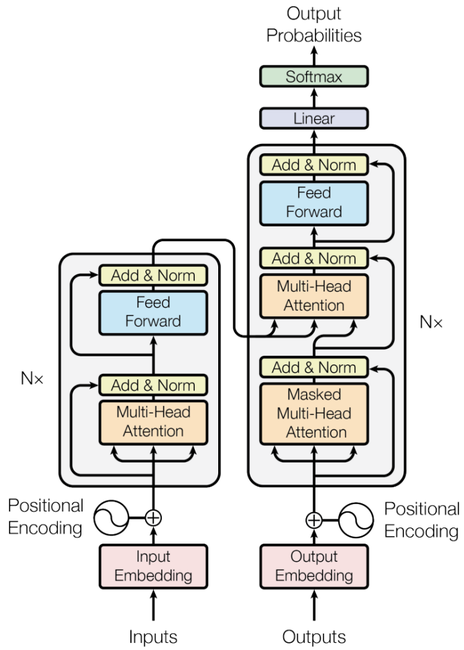
# VII. Conclusion

- **Conclusion : Transformer**

1) 본 논문은 transformer architecture를 제안
2) 이전과 다르게 전적으로 attention 매커니즘만을 이용하여 recurrent한 네트워크 자체를 아키텍쳐에서 뺌
3) 이를 통해 보다 높은 병렬성 및 성능 개선을 확인할 수 있음
4) 또한 기계번역 태스크 뿐 아니라 자연어 처리의 다양한 분야에서도 우수한 성능을 보이며, 높은 적용가능성을 확인해 볼 수 있었음

BERT

Encoder




GPT

Decoder

- GPT : Transformer의 디코더(Decoder) 아키텍처를 활용
- BERT : Transformer의 인코더(Encoder) 아키텍처를 활용