Provided proper attribution is provided, Google hereby grants permission to reproduce the tables and figures in this paper solely for use in journalistic or scholarly works.

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

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.

*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

†Work performed while at Google Brain.

‡Work performed while at Google Research.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

arXiv:1706.03762v7 [cs.CL] 2 Aug 2023

적절한 출전을 명시하는 경우, 구글은 이 논문 내의 표와 그림을 저널리즘 또는 학술적 목적에 한해 복제할 수 있는 권한을ここに 부여합니다.

주의(Attention)만 있으면 충분하다

아시什 바스와니* 구글 브레인 avaswani@google.com

노암 샤지어* 구글 브레인 noam@google.com

니키 파마* 구글 리서치 nikip@google.com

야코부스 우스크로이트* 구글 리서치usz@google.com

Llion Jones* Google Research llion@google.com

에이던 N. 고메즈* 토론토대학교 aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* illia.polosukhin@gmail.com

초록

우세한 순서 변환 모델은 인코더와 디코더를 포함하는 복잡한 재귀적 또는 컨볼루션 신경망에 기반한다. 가장 성능이 높은 모델들은 또한 인코더와 디코더를 주의 메커니즘을 통해 연결한다. 우리는 재귀와 컨볼루션을 완전히 배제하고 오직 주의 메커니즘에만 기반한 새로운 간단한 네트워크 구조인 Transformer를 제안한다. 두 기계 번역 과제에 대한 실험 결과, 이 모델들은 품질 면에서 우위에 있으며 병렬화가 더 용이하며 훈련에 필요한 시간이 현저히 적게 소요됨을 보여준다. 우리 모델은 WMT 2014 영어-독일어 번역 과제로 28.4의 BLEU 점수를 달성하여, 엔블블(집합)을 포함한 기존 최상 결과보다 2 BLEU 이상 향상시켰다. WMT 2014 영어-프랑스어 번역 과제에서는, 문헌에 reported된 최고 수준의 모델들의 훈련 비용에 비해 매우 작은 부분인 8개의 GPU에서 3.5일간 훈련한 후, 우리 모델은 41.8의 새로운 단일 모델 최첨단 BLEU 점수를 기록하였다. 우리는 Transformer가 큰 데이터와 제한된 훈련 데이터를 모두 사용하여 영어 구문 구성 분석에 성공적으로 적용됨으로써 다른 과제에도 잘 generalizes(적응/확장)함을 보여준다.

*동등 기여. 나열 순서는 무작위임. 야코브(R Jakob)는 RNN을 self-attention으로 대체할 것을 제안하고 이 아이디어를 평가하기 위한 노력을 시작함. 아시시(Ashish)는 일리아(Illia)와 함께 첫 번째 Transformer 모델을 설계 및 구현했으며, 이 작업의 모든 측면에 핵심적으로 관여함. 노암(Noam)은 스케일된 점곱 주의机制(scaled dot-product attention), 멀티 헤드 주의机制(multi-head attention), 매개변수 없는 위치 표현을 제안하며 거의 모든 세부 사항에 관여한 또 다른 주요 인물이 됨. 니키(Niki)는 우리 원본 코드베이스와 tensor2tensor에서 수없이 많은 모델 변형을 설계, 구현, 튜닝 및 평가함. 리온(Llion) 또한 novel model variants로 실험했으며, 초기 코드베이스와 효율적인 추론 및 시각화를 담당함. 루카시(Lukasz)와 에 (Aidan)은 수많은 긴 날들을 보내 여러 부분을 설계하고 our earlier codebase를 대체하는 tensor2tensor를 구현하여 연구 결과를 크게 향상시키고 연구 속도를 획기적으로 가속화함.

†구글 브레인 재직 시 수행된 연구.
구글 리서치 재직 중 수행된 연구.

제31회 신경 정보 처리 시스템Conference (NIPS 2017), 미국 캘리포니아주 롱비치

arXiv:1706.03762v7 [cs.CL] 2 Aug 2023

1 Introduction

Recurrent neural networks, long short-term memory [13] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation [35, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [38, 24, 15].

Recurrent models typically factor computation along the symbol positions of the input and output sequences. Aligning the positions to steps in computation time, they generate a sequence of hidden states h_t, as a function of the previous hidden state h_{t-1} and the input for position t. This inherently sequential nature precludes parallelization within training examples, which becomes critical at longer sequence lengths, as memory constraints limit batching across examples. Recent work has achieved significant improvements in computational efficiency through factorization tricks [21] and conditional computation [32], while also improving model performance in case of the latter. The fundamental constraint of sequential computation, however, remains.

Attention mechanisms have become an integral part of compelling sequence modeling and transduction models in various tasks, allowing modeling of dependencies without regard to their distance in the input or output sequences [2, 19]. In all but a few cases [27], however, such attention mechanisms are used in conjunction with a recurrent network.

In this work we propose the Transformer, a model architecture eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output. The Transformer allows for significantly more parallelization and can reach a new state of the art in translation quality after being trained for as little as twelve hours on eight P100 GPUs.

2 Background

The goal of reducing sequential computation also forms the foundation of the Extended Neural GPU [16], ByteNet [18] and ConvS2S [9], all of which use convolutional neural networks as basic building block, computing hidden representations in parallel for all input and output positions. In these models, the number of operations required to relate signals from two arbitrary input or output positions grows in the distance between positions, linearly for ConvS2S and logarithmically for ByteNet. This makes it more difficult to learn dependencies between distant positions [12]. In the Transformer this is reduced to a constant number of operations, albeit at the cost of reduced effective resolution due to averaging attention-weighted positions, an effect we counteract with Multi-Head Attention as described in section 3.2.

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].

End-to-end memory networks are based on a recurrent attention mechanism instead of sequence-aligned recurrence and have been shown to perform well on simple-language question answering and language modeling tasks [34].

To the best of our knowledge, however, the Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or convolution. In the following sections, we will describe the Transformer, motivate self-attention and discuss its advantages over models such as [17, 18] and [9].

3 Model Architecture

Most competitive neural sequence transduction models have an encoder-decoder structure [5, 2, 35]. Here, the encoder maps an input sequence of symbol representations $(x_1, ..., x_n)$ to a sequence of continuous representations $\mathbf{z} = (z_1, ..., z_n)$. Given $\mathbf{z}$, the decoder then generates an output sequence $(y_1, ..., y_m)$ of symbols one element at a time. At each step the model is auto-regressive [10], consuming the previously generated symbols as additional input when generating the next.

1 서론

재귀 신경망, 특히 장단기 메모리 [13] 및 게이트드 재귀 [7] 신경망은 언어 모델링 및 기계 번역 [35, 2, 5]과 같은 시퀀스 모델링 및 전도 문제에서 최첨단 기법으로 확고히 자리 잡았습니다. 이후 수많은 연구가 재귀 언어 모델 및 인코더-디코더 아키텍처의 한계를 넓혀 나가는 데 기여해 왔습니다 [38, 24, 15].

재귀 모델은 일반적으로 입력 및 출력 시퀀스의 기호 위치를 따라 연산을 분해한다. 이러한 위치를 계산 시간의 단계에 정렬하여, 그들은 이전 숨겨진 상태 h_{t-1}과 위치 t에 대한 입력을 함수로 하여 숨겨진 상태 h_t의 시퀀스를 생성한다. 이러한 본질적인 순차적 성질은 훈련 예제 내 병렬화를 불가능하게 하며, 이는 메모리 제약이 예제 간 배치(batching)를 제한하기 때문에 더 긴 시퀀스 길이에서 핵심적인 문제가 된다. 최근 연구는 [21]의 분해 기법과 조건부 연산 [32]을 통해 계산 효율성에서 상당한 개선을 이루었으며, 후자의 경우에는 모델 성능 또한 향상시켰다. 그러나 순차적 연산에 대한 근본적인 제약은 여전히 남아 있다.

주의 메커니즘은 다양한 작업에서 설득력 있는 시퀀스 모델링 및 변환 모델의 필수적인 부분이 되었으며, 입력 또는 출력 시퀀스 내 거리와 무관하게 종속성을 모델링할 수 있게 합니다 [2, 19]. 그러나 몇몇 예외 [27]를 제외하고는, 이러한 주의 메커니즘은 대부분 순환 네트워크와 결합되어 사용됩니다.

이 연구에서 우리는 순환을 배제하고 대신 입력과 출력 간의 전역 종속성을 포착하기 위해注意力 메커니즘에만 의존하는 모델 아키텍처인 Transformer를 제안한다. Transformer는 병렬化处理을 크게 증대시킬 수 있으며, 8대의 P100 GPU에서 최소 12시간만 훈련하여 번역 품질 분야에서 새로운 최상수준(new state of the art)에 도달할 수 있다.

2 배경

순차 연산의 감축 목표는 Extended Neural GPU [16], ByteNet [18], ConvS2S [9]의 기초를 이루며, 이 모든 모델은 기본 구성 요소로卷积神经网络를 사용하며, 모든 입력 및 출력 위치의 숨겨진 표현을 병렬로 계산한다. 이러한 모델에서 두 임의의 입력 또는 출력 위치 간의 신호를 연결하는 데 필요한 연산 횟수는 위치 간 거리에 따라 증가하는데, ConvS2S인 경우 선형적으로, ByteNet인 경우 로그함수적으로 증가한다. 이는远距离 위치 사이의 종속성을 학습하는 것을 더 어렵게 만든다 [12]. Transformer에서는 이를 일정한 연산 횟수로 줄였으나, 이는attention-weighted positions를 평균화하여 유효한 해상도가 감소한다는 대가를 치르는 것이다. 우리는 3.2절에서 설명하듯 Multi-Head Attention를 통해 이러한 효과를 상쇄한다.

셀프-attention(때로는 인트라-attention이라 불림)은 시퀀스의 다른 위치들 간의 관계를 설정하여 해당 시퀀스의 표현을 계산하기 위한 attention 메커니즘이다. 셀프-attention은 독해 이해, 추상적 요약, 텍스트 함의 및 태스크 독립적 문장 표현 학습 등 다양한 작업에서 성공적으로 활용되어 왔다 [4, 27, 28, 22].

엔드 투 엔드 메모리 네트워크는 시퀀스 정렬 반복 대신 순환 주의 메커니즘에 기반하며, 간단한 언어 질문 응답 및 언어 모델링 작업에서 우수한 성능을 보이는 것으로 나타났다 [34].

저자의 지식에 의하면, Transformer는 서열 정렬 RNN이나 합성곱을 사용하지 않고 입력 및 출력의 표현을 계산하기 위해 자기 주의 Mechanism(self-attention)에만 의존하는 최초의 전도 모델입니다. 다음 섹션에서 우리는 Transformer를 설명하고, 자기 주의 Mechanism(self-attention)의 필요성을 논하며, 이를 [17, 18] 및 [9]와 같은 모델에 비해優位性(advantages)을 논의할 것입니다.

3 모델 아키텍처

대부분의 경쟁력 있는 신경 시퀀스 전도 모델은 인코더-디코더 구조를 가진다 [5, 2, 35]. 여기서 인코더는 기호 표현의 입력 시퀀스 $(x_1, \ldots, x_n)$을 연속적 표현 $\mathbf{z}=(z_1, \ldots, z_n)$의 시퀀스로 매핑한다. $\mathbf{z}$가 주어지면, 디코더는 기호의 출력 시퀀스 $(y_1, \ldots, y_m)$을 한 요소씩 생성한다. 각 단계에서 모델은 자동 회귀적(auto-regressive)이며 [10], 다음을 생성할 때 이전에 생성된 기호를 추가 입력으로 사용한다.

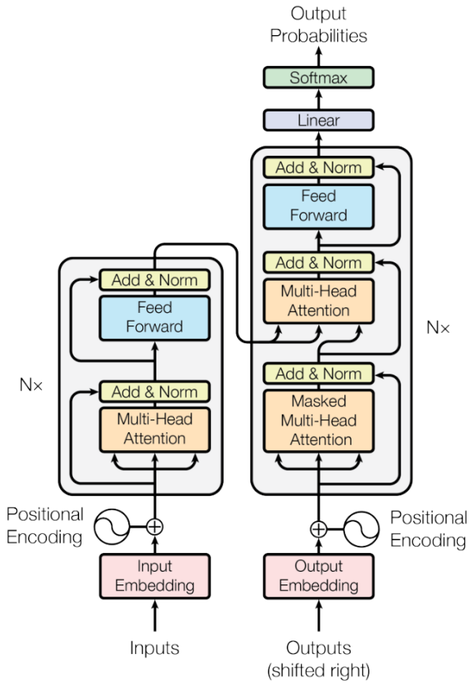


Figure 1: The Transformer - model architecture.

The Transformer follows this overall architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder, shown in the left and right halves of Figure 1, respectively.

3.1 Encoder and Decoder Stacks

Encoder: The encoder is composed of a stack of $N = 6$ identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. We employ a residual connection [11] around each of the two sub-layers, followed by layer normalization [1]. That is, the output of each sub-layer is $\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x))$, where $\mathrm{Sublayer}(x)$ is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension $d_{\text{model}} = 512$.

Decoder: The decoder is also composed of a stack of $N = 6$ identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, we employ residual connections around each of the sub-layers, followed by layer normalization. We also modify the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with fact that the output embeddings are offset by one position, ensures that the predictions for position i can depend only on the known outputs at positions less than i.

3.2 Attention

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum

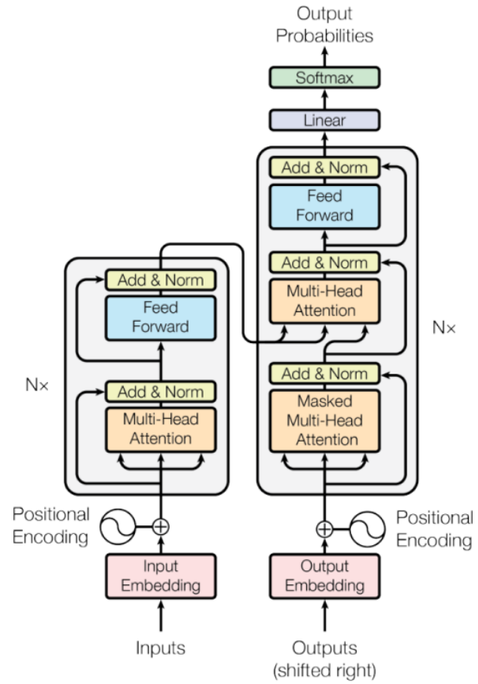


그림 1: 트랜스포머 모델 아키텍처.

Transformer는 인코더와 디코더 모두에 대해 쌓인 자기 주의력(self-attention)과 점별 전 연결층(point-wise, fully connected layers)을 사용하는 이 전체 아키텍처를 따르며, 이는 각각 1번 도서의 좌측 및 우측 부분에 나타납니다.

3.1 인코더 및 디코더 스택

인코더: 인코더는 $N=6$개의 동일한 레이어로 구성된 스택으로 이루어져 있다. 각 레이어는 두 개의 서브레이어를 갖는다. 첫 번째 서브레이어는 멀티 헤드 셀프 어텐션 메커니즘이며, 두 번째는 위치별 전 연결 피드포워드 네트워크이다. 우리는 두 서브레이어 주변에 잔차 연결(residual connection) [11]을 적용하고, 그 다음 계층 정규화(layer normalization) [1]을 수행한다. 즉, 각 서브레이어의 출력은 $\mathrm{LayerNorm}(x+\mathrm{Sublayer}(x))$이며, 여기서 $\mathrm{Sublayer}(x)$는 서브레이어 자체가 구현한 함수를 의미한다. 이러한 잔차 연결을 위해 모델 내의 모든 서브레이어 및 임베딩 레이어는 $d_{\text{model}}=512$ 차원의 출력을 산출한다.

디코더: 디코더도 $N=6$개의 동일한 레이어로 구성된 스택으로 이루어져 있습니다. 인코더 레이어의 두 서브레이어 외에도, 디코더는 인코더 스택의 출력에 대해 멀티 헤드 어텐션을 수행하는 세 번째 서브레이어를 삽입합니다. 인코더와 마찬가지로, 우리는 각 서브레이어 주변에 잔차 연결(residual connections)을 적용하고, 그 다음 레이어 정규화(layer normalization)를 수행합니다. 또한 디코더 스택의 셀프 어텐트(sub-layer)를 수정하여 특정 위치가 이후 위치들(subsequent positions)에 어텐트(attention)를 할 수 없도록 합니다. 이 마스킹(masking)과 출력 임베딩(output embeddings)이 한 위치만큼 오프셋(offset)되어 있다는 사실은, 위치 i에 대한 예측이 오직 i보다 작은 위치들의已知된 outputs에만 의존하도록 보장합니다.

3.2 주의력

주의 함수는 쿼리와 키-값 쌍의 집합을 출력으로 매핑하는 것으로 설명할 수 있으며, 여기서 쿼리, 키, 값, 그리고 출력은 모두 벡터이다. 출력은 가중 합으로 계산된다.

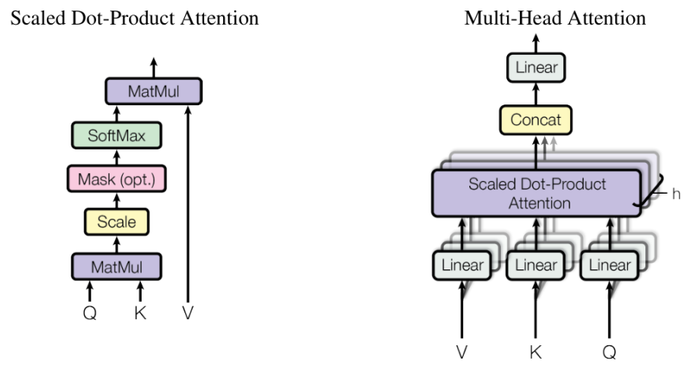


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

3.2.1 Scaled Dot-Product Attention

We call our particular attention "Scaled Dot-Product Attention" (Figure 2). The input consists of queries and keys of dimension d_k, and values of dimension d_v. We compute the dot products of the query with all keys, divide each by $\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values.

In practice, we compute the attention function on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V. We compute the matrix of outputs as:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

The two most commonly used attention functions are additive attention [2], and dot-product (multiplicative) attention. Dot-product attention is identical to our algorithm, except for the scaling factor of $\frac{1}{\sqrt{d_k}}$. Additive attention computes the compatibility function using a feed-forward network with a single hidden layer. While the two are similar in theoretical complexity, dot-product attention is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code.

While for small values of d_k the two mechanisms perform similarly, additive attention outperforms dot product attention without scaling for larger values of d_k [3]. We suspect that for large values of d_k, the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients [4]. To counteract this effect, we scale the dot products by $\frac{1}{\sqrt{d_k}}$.

3.2.2 Multi-Head Attention

Instead of performing a single attention function with d_{model}-dimensional keys, values and queries, we found it beneficial to linearly project the queries, keys and values h times with different, learned linear projections to d_k, d_k and d_v dimensions, respectively. On each of these projected versions of queries, keys and values we then perform the attention function in parallel, yielding d_v-dimensional

[4]To illustrate why the dot products get large, assume that the components of q and k are independent random variables with mean 0 and variance 1. Then their dot product, $q \cdot k = \sum_{i=1}^{d_k} q_i k_i$, has mean 0 and variance d_k.

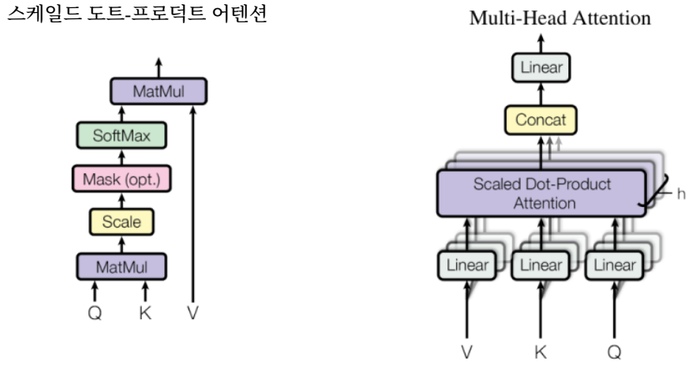


그림 2: (왼쪽) 스케일된 점곱attention. (오른쪽) 멀티헤드attention은 병렬로 작동하는 여러 attention 레이어로 구성됩니다.

값들 중 일부이며, 각 값에 할당된 가중치는 해당 키와 쿼리 간의 호환성 함수에 의해 계산됩니다.

3.2.1 스케일드 도트-프로덕트 어텐션

우리의 특정.attention을 "스케일된 점积.attention" (Figure 2)라고 부릅니다. 입력은 차원 d_k의 쿼리 및 키와 차원 d_v의 값으로 구성됩니다. 우리는 쿼리와 모든 키의 점积을 계산한 뒤, 각값을 $\sqrt{d_k}$로 나누고 softmax 함수를 적용하여 값에 대한 가중치를 얻습니다.

실제로는, 일련의 쿼리를 동시에 처리하기 위해 행렬 Q에-Pack하여注意力 함수를 계산합니다. 키와 값 또한 행렬 K 및 V에-Pack하여 함께 묶습니다. 우리는 출력의 행렬을 다음과 같이 계산합니다:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

가장 널리 사용되는 두 가지attention 함수는 가산 attention [2]과 점곱(multiplicative) attention입니다. 점곱 attention은 $\frac{1}{\sqrt{d_k}}$의 스케일링 인자를 제외하면 우리 알고리즘과 동일합니다. 가산 attention은 단일 숨겨진 계층을 가진 전방 연결 네트워크를 사용하여 호환성 함수를 계산합니다. 두 방법은 이론적 복잡성이 유사하지만, 점곱 attention은 고도로 최적화된 행렬 곱셈 코드를 통해 구현할 수 있어 실제로는 훨씬 빠르고 공간 효율적입니다.

d_k의 값이 작을 때 두 메커니즘은 유사하게 작동하지만, 가산 주의Mechanism(additive attention)은 d_k가 클 때 스케일링 없는 내적 곱 주의Mechanism(dot product attention)보다 우월하다 [3]. 우리는 d_k의 값이 클 때 내적 곱의 크기가 커져 softmax 함수를 그 기울기가 극도로 작은 영역으로 밀어 넣는다고 추측한다 4. 이 효과를 상쇄하기 위해, 우리는 내적 곱을 $\frac{1}{\sqrt{d_k}}$으로 스케일링한다.

3.2.2 멀티 헤드 어텐션

d_{model} 차원의 키, 값, 쿼리를 사용하는 단일 주의 함수를 수행하는 대신, 쿼리, 키 및 값을 h回, 서로 다른 학습된 선형 투영을 통해 각각 d_k, d_k 및 d_v 차원으로 선형 투영하는 것이 유리함을 발견했습니다. 쿼리, 키 및 값의 이러한 투영된 버전에 대해 병렬로 주의 함수를 수행하여 d_v 차원의 결과를 얻습니다.

[4]점积의 크기가 커지는 이유를 설명하기 위해, q와 k_i의 성분이 평균 0, 분산 1인 독립적인 확률변수라고 가정하자. 그러면 그들의 점积 $q \cdot k = \sum_{i=1} q_i k_i$는 평균 0과 분산 d_k를 가진다.

output values. These are concatenated and once again projected, resulting in the final values, as depicted in Figure 2.

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.

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

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_\text{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_\text{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_\text{model} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_\text{model}}$.

In this work we employ $h = 8$ parallel attention layers, or heads. For each of these we use $d_k = d_v = d_\text{model}/h = 64$. Due to the reduced dimension of each head, the total computational cost is similar to that of single-head attention with full dimensionality.

3.2.3 Applications of Attention in our Model

The Transformer uses multi-head attention in three different ways:

- In "encoder-decoder attention" layers, the queries come from the previous decoder layer, and the memory keys and values come from the output of the encoder. This allows every position in the decoder to attend over all positions in the input sequence. This mimics the typical encoder-decoder attention mechanisms in sequence-to-sequence models such as [38, 2, 9].
- The encoder contains self-attention layers. In a self-attention layer all of the keys, values and queries come from the same place, in this case, the output of the previous layer in the encoder. Each position in the encoder can attend to all positions in the previous layer of the encoder.
- Similarly, self-attention layers in the decoder allow each position in the decoder to attend to all positions in the decoder up to and including that position. We need to prevent leftward information flow in the decoder to preserve the auto-regressive property. We implement this inside of scaled dot-product attention by masking out (setting to $-\infty$) all values in the input of the softmax which correspond to illegal connections. See Figure 2.

3.3 Position-wise Feed-Forward Networks

In addition to attention sub-layers, each of the layers in our encoder and decoder contains a fully connected feed-forward network, which is applied to each position separately and identically. This consists of two linear transformations with a ReLU activation in between.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{2}$$

While the linear transformations are the same across different positions, they use different parameters from layer to layer. Another way of describing this is as two convolutions with kernel size 1. The dimensionality of input and output is $d_\text{model} = 512$, and the inner-layer has dimensionality $d_{ff} = 2048$.

3.4 Embeddings and Softmax

Similarly to other sequence transduction models, we use learned embeddings to convert the input tokens and output tokens to vectors of dimension d_model. We also use the usual learned linear transformation and softmax function to convert the decoder output to predicted next-token probabilities. In our model, we share the same weight matrix between the two embedding layers and the pre-softmax linear transformation, similar to [30]. In the embedding layers, we multiply those weights by $\sqrt{d_\text{model}}$.

출력 값들. 이들은 연결된 후 다시 투사되어, Figure 2에 묘사된 바와 같이 최종 값들을 산출한다.

멀티 헤드 어텐션은 모델이 서로 다른 위치에서 다양한 표현 서브스페이스의 정보를 동시에 주시할 수 있게 합니다. 단일 어텐션 헤드를 사용할 경우, 평균화 과정이 이를 방해합니다.

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

여기서 투사 행렬 $W_i^Q \in \mathbb{R}^{d_\text{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_\text{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_\text{model} \times d_v}$ 및 $W^O \in \mathbb{R}^{hd_v \times d_\text{model}}$는 매개변수 행렬이다.

이 연구에서 우리는 $h = 8$개의 병렬attention 레이어, 즉 헤드(heads)를 사용한다. 각 헤드에 대해 $d_k = d_v = d_\text{model}/h = 64$를 적용한다. 각 헤드의 차원이 축소되므로, 총 계산 비용은 완전한 차원을 가진 단일 헤드attention과 유사하다.

3.2.3 우리 모델에서의 어텐션 적용

Transformer는 세 가지 다른 방식으로 멀티 헤드 어텐션을 사용합니다:

"엔코더-디코더 어텐션" 계층에서 쿼리는 이전 디코더 계층에서 오고, 메모리 키와 값은 엔코더의 출력에서 온다. 이를 통해 디코더의 모든 위치가 입력 시퀀스의 모든位置上에 어텐션을 할 수 있다. 이는 [38, 2, 9]와 같은 시퀀스-투-시퀀스 모델의 전형적인 엔코더-디코더 어텐션 메커니즘을 모방한다.

인코더는 셀프 어텐션 레이어를 포함한다. 셀프 어텐션 레이어에서 키, 값, 쿼리는 모두 같은 위치에서 오는데, 이 경우 인코더의 이전 레이어 출력에서 온다. 인코더의 각 위치는 인코더의 이전 레이어의 모든 위치에 어텐트할 수 있다.

similarly, decoder의 self-attention 레이어는 각 위치가 그 위치와 그 이전의 모든 위치까지 주의를 기울일 수 있게 합니다. 자동 회귀 성질을 보존하기 위해 Decoder 내의 좌측 정보 흐름을 차단해야 합니다. 우리는 이를scaled dot-product attention 내부에서 구현하는데, softmax의 입력에 해당하는 불법 연결에 대응하는 모든 값($-\infty$)을 마스킹(설정)함으로써 수행합니다. Figure 2를 참고하십시오.

3.3 위치별 전방 피드-forward 네트워크

주의 하위 계층 외에, 엔코더와 디코더의 각 레이어는 Fully Connected 피드포워드 네트워크를 포함하며, 이는 각 위치별로 개별적이고 동일하게 적용됩니다. 이 네트워크는 그 사이에 ReLU 활성화 함수가 있는 두 개의 선형 변환으로 구성됩니다.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{2}$$

선형 변환은 서로 다른 위치에서 동일하게 적용되지만, 레이어마다 매개변수는異합니다. 이를 두 가지 커널 크기가 1인 컨볼루션으로 설명할 수도 있습니다. 입력 및 출력의 차원은 $d_\text{model} = 512$이며, 내부 레이어의 차원은 $d_{ff} = 2048$입니다.

3.4 임베딩과 소프트맥스

다른 시퀀스 전도 모델과 마찬가지로, 우리는 입력 토큰과 출력 토큰을 d_model 차원의 벡터로 변환하기 위해 학습된 임베딩을 사용한다. 또한 디코더 출력을 예측된 다음 토큰 확률로 변환하기 위해 일반적인 학습된 선형 변환 및 softmax 함수를 사용한다. 본 모델에서는 두 임베딩 계층과 pre-softmax 선형 변환 사이에서 동일한 가중치 행렬을 공유하며, 이는 [30]과 유사하다. 임베딩 계층에서 우리는 해당 가중치에 $\sqrt{d_\text{model}}$를 곱한다.

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
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)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

3.5 Positional Encoding

Since our model contains no recurrence and no convolution, in order for the model to make use of the order of the sequence, we must inject some information about the relative or absolute position of the tokens in the sequence. To this end, we add "positional encodings" to the input embeddings at the bottoms of the encoder and decoder stacks. The positional encodings have the same dimension d_{model} as the embeddings, so that the two can be summed. There are many choices of positional encodings, learned and fixed [9].

In this work, we use sine and cosine functions of different frequencies:

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

where pos is the position and i is the dimension. That is, each dimension of the positional encoding corresponds to a sinusoid. The wavelengths form a geometric progression from 2π to $10000 \cdot 2\pi$. We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset k, PE_{pos+k} can be represented as a linear function of PE_{pos}.

We also experimented with using learned positional embeddings [9] instead, and found that the two versions produced nearly identical results (see Table 3 row (E)). We chose the sinusoidal version because it may allow the model to extrapolate to sequence lengths longer than the ones encountered during training.

4 Why Self-Attention

In this section we compare various aspects of self-attention layers to the recurrent and convolutional layers commonly used for mapping one variable-length sequence of symbol representations $(x_1, ..., x_n)$ to another sequence of equal length $(z_1, ..., z_n)$, with $x_i, z_i \in \mathbb{R}^d$, such as a hidden layer in a typical sequence transduction encoder or decoder. Motivating our use of self-attention we consider three desiderata.

One is the total computational complexity per layer. Another is the amount of computation that can be parallelized, as measured by the minimum number of sequential operations required.

The third is the path length between long-range dependencies in the network. Learning long-range dependencies is a key challenge in many sequence transduction tasks. One key factor affecting the ability to learn such dependencies is the length of the paths forward and backward signals have to traverse in the network. The shorter these paths between any combination of positions in the input and output sequences, the easier it is to learn long-range dependencies [12]. Hence we also compare the maximum path length between any two input and output positions in networks composed of the different layer types.

As noted in Table 1, a self-attention layer connects all positions with a constant number of sequentially executed operations, whereas a recurrent layer requires $O(n)$ sequential operations. In terms of computational complexity, self-attention layers are faster than recurrent layers when the sequence

표 1: 서로 다른 레이어 유형에 대한 최대 경로 길이, 레이어별 복잡도 및 순차 연산의 최소 개수. n은 시퀀스 길이, d는 표현 차원, k는 컨볼루션의 커널 크기, r은 제한된 자기 주의机制(attention)에서의 근접 영역 크기를 나타낸다.

레이어 유형	레이어당 복잡도	순차적 연산	최대 경로 길이
자기 주의 (Self-Attention)	O(n² · d)	O(1)	O(1)
순환적	O(n · d²)	O()	O(n)
합성곱 (Convolutional)	O(k · n · d²)	O(1)	O(log□(n))
자기 주의 (제한적)	O(r · n · d)	O(1)	O(n/r)

3.5 위치 인코딩

우리의 모델에는 재귀와卷积이 포함되어 있지 않으므로, 모델이 서열의 순서를 활용할 수 있도록 서열 내 토큰의 상대적 또는 절대적 위치에 대한 정보를 주입해야 합니다. 이를 위해 엔코더 및 디코더 스택 하단의 입력 임베딩에 "위치 인코딩"을 추가합니다. 위치 인코딩은 임베딩과 동일한 차원 d_{model}을 가지므로 두 가지를 더할 수 있습니다. 위치 인코딩의 선택 방법은 다양하며, 학습되는 방식과 고정되는 방식으로 나뉩니다 [9].

이 연구에서는 서로 다른 주파수의 사인 및 코사인 함수를 사용한다:

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

pos는 위치를, i는 차원을 나타낸다. 즉, 위치 인코딩의 각 차원은 사인파에 대응한다. 파장은 2π에서 $10000 \cdot 2\pi$까지 기하급수적 배열을 이룬다. 우리는 이 함수를 선택한 이유는 모델이 상대적 위치에 대한 주의를 쉽게 학습할 수 있을 것으로 가설했기 때문이며, 임의의 고정된 오프셋 k에 대해 PE_{pos+k}는 PE_{pos}의 선형 함수로 표현될 수 있기 때문이다.

또한 대안으로 학습된 위치 인코딩[9]을 적용해 보았으나, 두 버전이 거의 동일한 결과를 내는 것을 확인했습니다(표 3 행 (E) 참조). 우리는 모델이 훈련 중 접한 시퀀스 길이보다 더 긴 길이에 대해 외삽(extrapolate)할 수 있을 것으로 기대되어 사인파(sinusoidal) 버전을 선택하였습니다.

4 자주 Attention(주의)이 필요한 이유

이 섹션에서 우리는 기호 표현의 변수 길이 서열 $(x_1, \ldots, x_n)$을 동일한 길이의 다른 서열 $(z_1, \ldots, z_n)$로 매핑하는 데 일반적으로 사용되는 순환 및 합성층과 자기 주의(self-attention) 계층의 다양한 측면을 비교한다. 여기서 $x_i, z_i \in \mathbb{R}^d$이며, 이는 일반적인 서열 전도 인코더 또는 디코더의 숨겨진 계층에 해당한다.我们自己注意를 사용하는 동기를 부여하기 위해, 우리는 세 가지 요구 사항을 고려한다.

하나는 계층별 총 계산 복잡도이다. 다른 하나는 필요한 최소 순차 연산 수로 측정되는 병렬화 가능한 계산량이다.

세 번째는 네트워크 내 장거리 의존성 사이의 경로 길이이다. 장거리 의존성을 학습하는 것은 많은 시퀀스 전도 작업에서 핵심적인 과제로 남아 있다. 이러한 의존성을 학습하는 능력에 영향을 미치는 주요 요인 중 하나는 프로그래밍 및 백워드 신호가 네트워크를 통과해야 하는 경로의 길이이다. 입력 및 출력 시퀀스의 위치 조합 간 이러한 경로가 짧을수록 장거리 의존성을 학습하기가 쉬워진다 [12]. 따라서 우리는 서로 다른 레이어 유형으로 구성된 네트워크의 임의의 두 입력 및 출력 위치 사이의 최대 경로 길이를 비교한다.

Table 1에서 언급한 바와 같이, 셀프-어텐션 레이어는 순차적으로 실행되는 연산의 수를 일정하게 유지하며 모든 위치를 연결하지만, 재귀 레이어는 $O(n)$개의 순차적 연산을 필요로 한다. 계산 복잡도 측면에서 시퀀스 길이가 일정 수준에 도달하면 셀프-어텐션 레이어가 재귀 레이어보다 빠르다.

length n is smaller than the representation dimensionality d, which is most often the case with sentence representations used by state-of-the-art models in machine translations, such as word-piece [38] and byte-pair [31] representations. To improve computational performance for tasks involving very long sequences, self-attention could be restricted to considering only a neighborhood of size r in the input sequence centered around the respective output position. This would increase the maximum path length to $O(n/r)$. We plan to investigate this approach further in future work.

A single convolutional layer with kernel width $k < n$ does not connect all pairs of input and output positions. Doing so requires a stack of $O(n/k)$ convolutional layers in the case of contiguous kernels, or $O(log_k(n))$ in the case of dilated convolutions [18], increasing the length of the longest paths between any two positions in the network. Convolutional layers are generally more expensive than recurrent layers, by a factor of k. Separable convolutions [6], however, decrease the complexity considerably, to $O(k \cdot n \cdot d + n \cdot d^2)$. Even with $k = n$, however, the complexity of a separable convolution is equal to the combination of a self-attention layer and a point-wise feed-forward layer, the approach we take in our model.

As side benefit, self-attention could yield more interpretable models. We inspect attention distributions from our models and present and discuss examples in the appendix. Not only do individual attention heads clearly learn to perform different tasks, many appear to exhibit behavior related to the syntactic and semantic structure of the sentences.

5 Training

This section describes the training regime for our models.

5.1 Training Data and Batching

We trained on the standard WMT 2014 English-German dataset consisting of about 4.5 million sentence pairs. Sentences were encoded using byte-pair encoding [3], which has a shared source-target vocabulary of about 37000 tokens. For English-French, we used the significantly larger WMT 2014 English-French dataset consisting of 36M sentences and split tokens into a 32000 word-piece vocabulary [38]. Sentence pairs were batched together by approximate sequence length. Each training batch contained a set of sentence pairs containing approximately 25000 source tokens and 25000 target tokens.

5.2 Hardware and Schedule

We trained our models on one machine with 8 NVIDIA P100 GPUs. For our base models using the hyperparameters described throughout the paper, each training step took about 0.4 seconds. We trained the base models for a total of 100,000 steps or 12 hours. For our big models,(described on the bottom line of table 3), step time was 1.0 seconds. The big models were trained for 300,000 steps (3.5 days).

5.3 Optimizer

We used the Adam optimizer [20] with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. We varied the learning rate over the course of training, according to the formula:

$$lrate = d_{\text{model}}^{-0.5} \cdot \min(step_num^{-0.5}, step_num \cdot warmup_steps^{-1.5}) \quad (3)$$

This corresponds to increasing the learning rate linearly for the first $warmup_steps$ training steps, and decreasing it thereafter proportionally to the inverse square root of the step number. We used $warmup_steps = 4000$.

5.4 Regularization

We employ three types of regularization during training:

길이 n이 표현 차원 d보다 작은 경우가 가장 흔한데, 이는 기계 번역의 최첨단 모델에서 사용되는 문장 표현, 예를 들어 word-piece [38] 및 byte-pair [31] 표현에서 그렇다. 매우 긴 서열을 다루는 작업에 대한 계산 성능을 향상시키기 위해, self-attention은 respective 출력 위치를 중심으로 하는 크기 r의 인접 영역만 고려하도록 제한할 수 있다. 这样的话 maximum path length는 $O(n/r)$로 증가한다. 우리는 향후 연구에서 이 접근법을 더 깊이 조사할 계획이다.

커널幅 $k < n$ 인 단일 합성층은 모든 입력 및 출력 위치 쌍을 연결하지 못한다. 연속 커널의 경우 이를 위해 $O(n/k)$ 개의 합성층 스택이 필요하며, 확장 합성의 경우 $O(\log_k(n))$ 개가 필요하다 [18]. 이는 네트워크 내 임의의 두 위치 사이의 최장 경로 길이를 증가시킨다. 합성층은 일반적으로 재귀층보다 k 배 정도 비용이 많이 든다. 그러나 가분 합성(separable convolutions) [6]은 복잡도를 크게 줄여 $O(k \cdot n \cdot d + n \cdot d^2)$ 로 만든다.即便如此, $k = n$ 인 경우에도 가분 합성의 복잡도는 자기 주의(self-attention) 층과 지점별 피드포워드(layer)의 결합과 동일하며, 이것이 우리 모델에서 채택하는 방법이다.

부수적인 이점으로, 셀프 어텐션은 더 해석 가능한 모델을 생성할 수 있다. 우리는 모델의 어텐션 분포를 조사하며 부록에서 예시를 제시하고 논의한다. 개별 어텐션 헤드가 서로 다른 작업을 수행하도록 명확하게 학습하는 것뿐만 아니라, 많은 헤드가 문장의 구문 및 의미 구조와 관련된 행동을 보이는 것으로 보인다.

5 학습

이 섹션은 우리 모델의 훈련 체계를 설명한다.

5.1 학습 데이터 및 배치

우리는 약 450만 문장 쌍으로 구성된 표준 WMT 2014 영어-독일어 데이터셋을 사용하여 학습하였다. 문장은 공유된 소스-타겟 어휘가 약 37,000개의 토큰인 바이트-페어 인코딩(byte-pair encoding) [3]을 사용하여 인코딩되었다. 영어-프랑스어의 경우, 약 3,600만 문장으로 구성된 훨씬 더 큰 WMT 2014 영어-프랑스어 데이터셋을 사용하였으며, 토큰을 32,000개 단어 조각(word-piece) 어휘로 분할하였다 [38]. 문장 쌍은 근사한 시퀀스 길이에 따라 배치(batch)되었다. 각 학습 배치는 약 25,000개의 소스 토큰과 25,000개의 타겟 토큰을 포함하는 문장 쌍 세트를 포함하였다.

5.2 하드웨어 및 스케줄

우리는 8개의 NVIDIA P100 GPU가 장착된 단일 기계에서 모델들을 훈련시켰다. 본 논문 전반에 서술된 하이퍼파라미터를 사용한 기본 모델의 경우, 각 훈련 단계당 약 0.4초가 소요되었다. 기본 모델들은 총 100,000단계 또는 12시간 동안 훈련되었다. 대형 모델들(표 3의 하단 줄에 서술됨)의 경우, 단계당 소요 시간은 1.0초였다. 대형 모델들은 300,000단계(3.5일) 동안 훈련되었다.

5.3 최적화기

우리는 $\beta_1 = 0.9$, $\beta_2 = 0.98$ 및 $\epsilon = 10^{-9}$인 Adam 최적화기 [20]를 사용하였다. 학습률은 다음 공식에 따라 훈련 과정 전체에 걸쳐 변경하였다:

$$lrate = d_{\text{model}}^{-0.5} \cdot \min(step_num^{-0.5}, step_num \cdot warmup_steps^{-1.5}) \quad (3)$$

이는 첫 $warmup_{steps}$ 학습 단계 동안 학습률을 선형적으로 증가시키고, 이후에는 단계 수의 제곱근 역수에 비례하여 감소시키는 것에 해당합니다. 우리는 $warmup_{steps} = 4000$을 사용했습니다.

5.4 정칙화

우리는 학습 과정에서 세 가지 유형의 정칙화를 적용합니다:

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

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	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Residual Dropout We apply dropout [33] to the output of each sub-layer, before it is added to the sub-layer input and normalized. In addition, we apply dropout to the sums of the embeddings and the positional encodings in both the encoder and decoder stacks. For the base model, we use a rate of $P_{drop} = 0.1$.

Label Smoothing During training, we employed label smoothing of value $\epsilon_{ls} = 0.1$ [36]. This hurts perplexity, as the model learns to be more unsure, but improves accuracy and BLEU score.

6 Results

6.1 Machine Translation

On the WMT 2014 English-to-German translation task, the big transformer model (Transformer (big) in Table 2) outperforms the best previously reported models (including ensembles) by more than 2.0 BLEU, establishing a new state-of-the-art BLEU score of 28.4. The configuration of this model is listed in the bottom line of Table 3. Training took 3.5 days on 8 P100 GPUs. Even our base model surpasses all previously published models and ensembles, at a fraction of the training cost of any of the competitive models.

On the WMT 2014 English-to-French translation task, our big model achieves a BLEU score of 41.0, outperforming all of the previously published single models, at less than $1/4$ the training cost of the previous state-of-the-art model. The Transformer (big) model trained for English-to-French used dropout rate $P_{drop} = 0.1$, instead of 0.3.

For the base models, we used a single model obtained by averaging the last 5 checkpoints, which were written at 10-minute intervals. For the big models, we averaged the last 20 checkpoints. We used beam search with a beam size of 4 and length penalty $\alpha = 0.6$ [38]. These hyperparameters were chosen after experimentation on the development set. We set the maximum output length during inference to input length + 50, but terminate early when possible [38].

Table 2 summarizes our results and compares our translation quality and training costs to other model architectures from the literature. We estimate the number of floating point operations used to train a model by multiplying the training time, the number of GPUs used, and an estimate of the sustained single-precision floating-point capacity of each GPU [5].

6.2 Model Variations

To evaluate the importance of different components of the Transformer, we varied our base model in different ways, measuring the change in performance on English-to-German translation on the

[5]We used values of 2.8, 3.7, 6.0 and 9.5 TFLOPS for K80, K40, M40 and P100, respectively.

표 2: Transformer은 훈련 비용의 일부로만 영어-독어 및 영어-프랑스어 newstest2014 테스트에서 이전 최첨단 모델보다 높은 BLEU 점수를 달성합니다.

모델	BLEU		학습 비용 (FLOPs)	
	영어-독일어	영어-프랑스어	영어-독일어	영어-프랑스어
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 앙상블 [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL 앙상블 [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S 앙상블 [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (기본 모델)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (대형)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

잔차 드롭아웃 각 서브레이어의 출력에 드롭아웃 [33]을 적용하여, 이를 서브레이어 입력에 더하고 정규화하기 전에 수행합니다. 또한 인코더 및 디코더 스택 모두에서 임베딩과 위치 인코딩의 합에 대해 드롭아웃을 적용합니다. 기본 모델의 경우, $P_{drop} = 0.1$의 비율을 사용합니다.

라벨 스무딩 훈련 중, 우리는 값 $\varepsilon_{ls} = 0.1$인 라벨 스무딩을 적용하였다 [36]. 이는 모델이 더 불확실하게 학습되도록 하여 퍼플렉스(perplexity)를 높이지만, 정확도와 BLEU 점수는 향상시킨다.

6. 결과

6.1 기계 번역

WMT 2014 영어-독어 번역 과제에서, Table 2의 대형 트랜스포머 모델(Transformer(big))은 엔블블을 포함한 Previously 보고된 모든 모델보다 2.0 BLEU를 초과하여 우위이며, 새로운 최첨단 BLEU 점수인 28.4를 달성하였다. 이 모델의 구성은 Table 3의 하단 행에 나열되어 있다. 훈련은 8대의 P100 GPU에서 3.5일이 소요되었다.是我们的 기본 모델조차도 경쟁 모델들의的训练 비용의 일부분으로, previously published된 모든 모델과 엔블블을 초월한다.

WMT 2014 영어-프랑스어 번역 과제를 대상으로, 본 대형 모델은 BLEU 점수 41.0을 달성하여 기존에 발표된 모든 단일 모델보다 우수하며, 이전 최첨단 모델의 학습 비용의 1/4 미만으로 achievement하였다. 영어-프랑스어 번역을 위해 훈련된 Transformer(대형) 모델은 0.3 대신 드롭아웃율 $P_{drop} = 0.1$을 사용하였다.

베이스 모델의 경우, 10분 간격으로 작성된 마지막 5개의 체크포인트를 평균화하여 얻은 단일 모델을 사용했습니다. 대형 모델의 경우, 마지막 20개의 체크포인트를 평균화했습니다. 비암搜索(beam search)를 사용했으며, 비암 크기(beam size)는 4, 길이 페널티 α는 0.6 [38]로 설정했습니다. 이러한 하이퍼파라미터들은 개발셋(development set)에 대한 실험을 통해 선정되었습니다. 추론 시 최대 출력 길이는 입력 길이 + 50으로 설정하되, 가능하면 조종 종료(early termination)를 적용했습니다 [38].

표 2는 우리 결과를 요약하고, 문헌에 제시된 다른 모델 아키텍처와 비교하여 우리 번역의 품질과 훈련 비용을 보여줍니다. 우리는 각 GPU의 지속 가능한 단일 정밀도 부동소수점 성능 추정을 사용된 GPU의 수 및 훈련 시간에 곱하여 모델을 훈련하는 데 사용된 부동소수점 연산의 개수를 추정합니다 [5].

6.2 모델 변형

Transformer의 구성 요소별 중요도를 평가하기 위해, 우리는 기본 모델을 다양한 방식으로 변형하여 영어-독일어 번역에서의 성능 변화를 측정하였다.

[5]K80, K40, M40, P100에 대해서는 각각 2.8, 3.7, 6.0, 9.5 TFLOPS의 값을 사용하였다.

Table 3: Variations on the Transformer architecture. Unlisted values are identical to those of the base model. All metrics are on the English-to-German translation development set, newstest2013. Listed perplexities are per-wordpiece, according to our byte-pair encoding, and should not be compared to per-word perplexities.

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{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

development set, newstest2013. We used beam search as described in the previous section, but no checkpoint averaging. We present these results in Table 3.

In Table 3 rows (A), we vary the number of attention heads and the attention key and value dimensions, keeping the amount of computation constant, as described in Section 3.2.2. While single-head attention is 0.9 BLEU worse than the best setting, quality also drops off with too many heads.

In Table 3 rows (B), we observe that reducing the attention key size d_k hurts model quality. This suggests that determining compatibility is not easy and that a more sophisticated compatibility function than dot product may be beneficial. We further observe in rows (C) and (D) that, as expected, bigger models are better, and dropout is very helpful in avoiding over-fitting. In row (E) we replace our sinusoidal positional encoding with learned positional embeddings [9], and observe nearly identical results to the base model.

6.3 English Constituency Parsing

To evaluate if the Transformer can generalize to other tasks we performed experiments on English constituency parsing. This task presents specific challenges: the output is subject to strong structural constraints and is significantly longer than the input. Furthermore, RNN sequence-to-sequence models have not been able to attain state-of-the-art results in small-data regimes [37].

We trained a 4-layer transformer with $d_{model} = 1024$ on the Wall Street Journal (WSJ) portion of the Penn Treebank [25], about 40K training sentences. We also trained it in a semi-supervised setting, using the larger high-confidence and BerkleyParser corpora from with approximately 17M sentences [37]. We used a vocabulary of 16K tokens for the WSJ only setting and a vocabulary of 32K tokens for the semi-supervised setting.

We performed only a small number of experiments to select the dropout, both attention and residual (section 5.4), learning rates and beam size on the Section 22 development set, all other parameters remained unchanged from the English-to-German base translation model. During inference, we

표 3: Transformer 아키텍처의 변형. 나열되지 않은 값은 기본 모델과 동일하다. 모든 지표는 영어-독어 번역 개발セット(newstest2013) 상에서 측정되었다. 나열된 퍼플렉시티(perplexities)는 바이트-페어 인코딩(byte-pair encoding)에 따른 단단어piece당 값이며, 단단어(per-word) 퍼플렉시티와 비교해서는 안 된다.

	N	dmodel	dff	h	dk	dv	Pdrop	ϵs	학습 단계	PPL (개발)	BLEU (개발)	파라미터 수 ×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	09
(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	06
(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)	사인 함수 대신 위치 임베딩 사용									4.92	25.7	
big	6	1024	4096	16			0.3		300K	4.33	26.4	213

개발 세트, newstest2013. 이전 섹션에서 설명한 바와 같이 빔 서치를 사용했으나 체크포인트 평균화는 적용하지 않았다. 이러한 결과는 Table 3에 제시한다.

표 3의 행 (A)에서, 3.2.2절에 설명된 바와 같이 연산량을 일정하게 유지한 채 주의력 헤드 수 및 주의력 키 및 값 차원을 변경하였다. 단일 헤드 주의력은 최상 설정 대비 BLEU 점수가 0.9 낮지만, 헤드가 너무 많을 경우 품질도 저하된다.

표 3의 행 (B)에서 우리는 attention 키 크기 d_k를 줄이면 모델 품질이 저하됨을 관찰합니다. 이는 호환성을 결정하기 어렵다는 것을 시사하며, 내적 곱(dot product)보다 더 정교한 호환성 함수가 유익할 수 있음을 의미합니다.我们又行 (C)와 (D)에서 예상대로 더 큰 모델일수록 성능이 우수하며, dropout은 과적합(over-fitting)을 방지하는 데 매우 도움이 된다는 것을 추가로 관찰합니다. 행 (E)에서는 기사(sinusoidal) 위치 인코딩을 학습된 위치 임베딩[9]으로 교체하여, 기본 모델(base model)과 거의 동일한 결과를 얻었습니다.

6.3 영어 구성소 구문 분석

Transformer가 다른 과업에도 적용 가능한지 평가하기 위해 영어 구성소 구문 분석에 대한 실험을 수행하였습니다. 이 과업은 출력이 강력한 구조적 제약을 받으며 입력보다 훨씬 길다는 점이라는 특유의 난이점을 가지고 있습니다. 또한, RNN 순서-순서 모델은 소량 데이터 환경에서 최첨단 성능을 달성하지 못해 왔습니다 [37].

우리는 Penn Treebank [25]의 Wall Street Journal (WSJ) 부분(약 40K 학습 문장)에서 모델 차원 $d_{model} = 1024$인 4층 트랜스포머를 훈련시켰다. 또한 약 17M 문장을 포함하는 더 큰 고품질 신뢰도(high-confidence) 및 BerkleyParser 코퍼스를 사용하여 반-supervised 설정에서도 훈련하였다. WSJ 전용 설정에는 16K 토큰 어휘를, 반-supervised 설정에는 32K 토큰 어휘를 사용하였다.

드롭아웃, 어텐션 및 잔차(5.4절), 학습률 및 빔 크기를 선택하기 위해 소수의 실험만 수행하였으며, 모든 다른 매개변수는 영어-독어 기본 번역 모델에서 그대로 유지되었습니다. 추론 중에는

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

increased the maximum output length to input length + 300. We used a beam size of 21 and $\alpha = 0.3$ for both WSJ only and the semi-supervised setting.

Our results in Table 4 show that despite the lack of task-specific tuning our model performs surprisingly well, yielding better results than all previously reported models with the exception of the Recurrent Neural Network Grammar [8].

In contrast to RNN sequence-to-sequence models [37], the Transformer outperforms the Berkeley-Parser [29] even when training only on the WSJ training set of 40K sentences.

7 Conclusion

In this work, we presented the Transformer, the first sequence transduction model based entirely on attention, replacing the recurrent layers most commonly used in encoder-decoder architectures with multi-headed self-attention.

For translation tasks, the Transformer can be trained significantly faster than architectures based on recurrent or convolutional layers. On both WMT 2014 English-to-German and WMT 2014 English-to-French translation tasks, we achieve a new state of the art. In the former task our best model outperforms even all previously reported ensembles.

We are excited about the future of attention-based models and plan to apply them to other tasks. We plan to extend the Transformer to problems involving input and output modalities other than text and to investigate local, restricted attention mechanisms to efficiently handle large inputs and outputs such as images, audio and video. Making generation less sequential is another research goals of ours.

The code we used to train and evaluate our models is available at `https://github.com/tensorflow/tensor2tensor`.

Acknowledgements We are grateful to Nal Kalchbrenner and Stephan Gouws for their fruitful comments, corrections and inspiration.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[3] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc V. Le. Massive exploration of neural machine translation architectures. *CoRR*, abs/1703.03906, 2017.

[4] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016.

표 4: Transformer는 영어 구성소 분석에 잘 일반화된다 (결과WSJ의 섹션 23 기준)

파서	학습	WSJ 23 F1
Vinyals 및 Kaiser 외 (2014) [37]	WSJ 전용, 판별적	88.3
Petrov 외 (2006) [29]	WSJ 전용, 판별적	90.4
Zhu 외 (2013) [40]	WSJ 전용, 판별적	90.4
Dyer 외 (2016) [8]	WSJ 전용, 판별적	91.7
Transformer (4층)	WSJ 전용, 판별적	91.3
Zhu 외 (2013) [40]	반지도 학습	91.3
Huang & Harper (2009) [14]	반지도 학습	91.3
McClosky 외 (2006) [26]	반지도 학습	92.1
Vinyals 및 Kaiser 외 (2014) [37]	반지도 학습	92.1
Transformer (4층)	반지도 학습	92.7
Luong 외 (2015) [23]	다중 작업	93.0
Dyer 외 (2016) [8]	생성적	93.3

최대 출력 길이를 입력 길이 + 300로 늘렸습니다. WSJ 전용 및 반 Supervised 설정 모두에서 빔 크기를 21, α를 0.3으로 사용했습니다.

표 4의结果表明, 비록 작업 특화 튜닝이 이루어지지 않았음에도 불구하고 본 모델은 놀라울 정도로 우수한 성능을 보이며, 재귀 신경망 문법[8]을 제외하고는 이전 보고된 모든 모델보다 더 나은 결과를 산출한다.

RNN 순서-순서 모델 [37]과 달리, Transformer는 40K 문장의 WSJ 학습 세트에서만 훈련된 경우에도 Berkeley-Parser [29]를 초월한다.

7 결론

본 연구에서 우리는 전 внимание에 기반하여 완전히 구축된 최초의 시퀀스 전도 모델인 트랜스포머를 제시하였으며, 엔코더-디코더 아키텍처에서 가장 일반적으로 사용되는 순환 계층을 멀티 헤드 셀프 어텐션으로 대체하였다.

번역 작업에서 트랜스포머는 순환 또는 컨볼루션 레이어 기반 아키텍처에 비해 훨씬 빠르게 훈련할 수 있다. WMT 2014 영어-독일어 번역 및 WMT 2014 영어-프랑스어 번역 작업 모두에서 최첨단 기술의 새로운 기록을 달성했다. 전자의 경우, 우리 모델이 Previously 보고된 모든 엔semble(집합 학습)보다 더 높은 성능을 보인다.

우리는attention 기반 모델의 미래에 대해 기대를 걸고 있으며, 이를 다른 작업에 적용할 계획이다. 우리는 Transformer를 텍스트가 아닌 입력 및 출력 모달리티를 포함하는 문제로 확장하고, 이미지, 오디오, 비디오와 같은 대규모 입력과 출력을 효율적으로 처리하기 위해 국소적이고 제한된 attention 메커니즘을 조사할 계획이다. 생성 과정을 더 순차적이지 않게 만드는 것은 우리의 또 다른 연구 목표이다.

우리가 모델의 학습 및 평가를 위해 사용된 코드는 https://github.com/tensorflow/tensor2tensor에서 확인할 수 있습니다.

감사 인사 Nal Kalchbrenner와 Stephan Gouws분께 유익한 의견, 수정 및 영감을 주신 점에 감사드립니다.

참고문헌

[1] 지미 레이 바, 제이미 라이언 키로스, 조피리 E 힌튼. 레이어 정규화. arXiv 사전 인인 arXiv:1607.06450, 2016.

[2] Dzmitry Bahdanau, Kyunghyun Cho, 및 Yoshua Bengio. 정렬과 번역을 공동으로 학습하는 신경 기계 번역. CoRR, abs/1409.0473, 2014.

[3] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc V. Le. 신경 기계 번역 아키텍처의 대규모 탐색. CoRR, abs/1703.03906, 2017.

[4] Jianpeng Cheng, Li Dong, 및 Mirella Lapata. 기계 독해를 위한 긴 단기 메모리 네트워크. arXiv 사전 인쇄물 arXiv:1601.06733, 2016.

[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[6] Francois Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357*, 2016.

[7] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[8] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. Recurrent neural network grammars. In *Proc. of NAACL*, 2016.

[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122v2*, 2017.

[10] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[12] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[14] Zhongqiang Huang and Mary Harper. Self-training PCFG grammars with latent annotations across languages. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 832–841. ACL, August 2009.

[15] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

[16] Łukasz Kaiser and Samy Bengio. Can active memory replace attention? In *Advances in Neural Information Processing Systems, (NIPS)*, 2016.

[17] Łukasz Kaiser and Ilya Sutskever. Neural GPUs learn algorithms. In *International Conference on Learning Representations (ICLR)*, 2016.

[18] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099v2*, 2017.

[19] Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. Structured attention networks. In *International Conference on Learning Representations*, 2017.

[20] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[21] Oleksii Kuchaiev and Boris Ginsburg. Factorization tricks for LSTM networks. *arXiv preprint arXiv:1703.10722*, 2017.

[22] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

[23] Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*, 2015.

[24] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, 및 Yoshua Bengio. 통계 기계 번역을 위한 RNN 인코더-디코더를 이용한 구절 표현 학습. CoRR, abs/1406.1078, 2014.

[6] Francois Chollet. Xception: 심층 학습을 위한 depthwise separable convolutions를 이용한deeplearning. arXiv 사전 인쇄본 arXiv:1610.02357, 2016.

[7] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, 및 Yoshua Bengio. 시퀀스 모델링에서 게이트드 순환 신경망에 대한 경험적 평가. CoRR, abs/1412.3555, 2014.

[8] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, 및 Noah A. Smith. 재귀 신경망 문법. NAACLProceedings에 수록, 2016.

[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, Yann N. Dauphin.卷积형 순서에서 순서로 학습. arXiv 사전 인출 arXiv:1705.03122v2, 2017.

[10] Alex Graves. 순환 신경망을 이용한 시퀀스 생성. arXiv 사전 인쇄본, arXiv:1308.0850, 2013.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, 및 Jian Sun. 이미지 인식을 위한 깊은 잔차 학습. IEEE计算机视觉 및 패턴 인식 회의록(Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition), 770 – 778페이지, 2016년.

[12] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, 및 Jürgen Schmidhuber. 순환 네트워크에서의 기울기 흐름: 장기 의존성 학습의 어려움, 2001.

[13] 세프 호크라이터와 융어른 슈미트후버. "롱 숏 트임 메모리." 니얼 컴퓨테이션, 9(8):1735 – 1780, 1997.

[14] Zhongqiang Huang 및 Mary Harper. 언어 간 잠재 주석과 함께 self-training PCFG 문법. In Natural Language Processing의 경험적 방법에 관한 2009ConferenceProceedings, 832 – 841쪽. ACL, 2009년 8월.

[15] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 언어 모델의 한계 탐구. arXiv 사전 인쇄본 arXiv:1602.02410, 2016.

[16] Łukasz Kaiser and Samy Bengio. 액티브 메모리가 어텐션을 대체할 수 있는가? , (NIPS), 2016.

[17] Łukasz Kaiser 및 Ilya Sutskever. 신경 GPU는 알고리즘을 학습한다. In International Conference on Learning Representations (ICLR), 2016.

[18] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 선형 시간 내 신경 기계 번역. arXiv 사전 인쇄물 arXiv:1610.10099v2, 2017.

[19] Yoon Kim, Carl Denton, Luong Hoang, 및 Alexander M. Rush. 구조화된 주의 네트워크. In 국제 학습 표현Conference, 2017.

[20] Diederik Kingma 및 Jimmy Ba. Adam: 확률적 최적화를 위한 한 방법. ICLR에, 2015.

[21] Oleksii Kuchaiev와 Boris Ginsburg. LSTM 네트워크를 위한 인수화 기법. arXiv预印本, arXiv:1703.10722, 2017.

[22] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, 및 Yoshua Bengio. 구조화된 자기 주의 문장 임베딩. arXiv 사전 인쇄본, arXiv:1703.03130, 2017.

[23] Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, 및 Lukasz Kaiser. 멀티태스크 시퀀스 투 시퀀스 학습. arXiv 프리프린트 arXiv:1511.06114, 2015.

[24] Minh-Thang Luong, Hieu Pham, 및 Christopher D Manning. 어텐션 기반 신경 기계 번역을 위한 효과적인 접근법. arXiv 프리프린트 arXiv:1508.04025, 2015.

[25] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[26] David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159. ACL, June 2006.

[27] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model. In *Empirical Methods in Natural Language Processing*, 2016.

[28] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.

[29] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 433–440. ACL, July 2006.

[30] Ofir Press and Lior Wolf. Using the output embedding to improve language models. *arXiv preprint arXiv:1608.05859*, 2016.

[31] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[32] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[33] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[34] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2440–2448. Curran Associates, Inc., 2015.

[35] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

[36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[37] Vinyals & Kaiser, Koo, Petrov, Sutskever, and Hinton. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, 2015.

[38] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[39] Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. Deep recurrent models with fast-forward connections for neural machine translation. *CoRR*, abs/1606.04199, 2016.

[40] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the ACL (Volume 1: Long Papers)*, pages 434–443. ACL, August 2013.

[25] Mitchell P Marcus, Mary Ann Marcinkiewicz, 및 Beatrice Santorini. 영어의 대규모 주석 부여 코퍼러스 구축: 펜 트리뱅크. 계산 언어학, 19(2):313 – 330, 1993.

[26] David McClosky, Eugene Charniak, 및 Mark Johnson. 구문 분석을 위한 효과적인 자생 학습. In NAACL 인간 언어 기술conferenceProceedings, 메인 컨퍼런스, 152 – 159페이지. ACL, 2006년 6월.

[27] Ankur Parikh, Oscar Täckström, Dipanjan Das, Jakob Uszkoreit. 분해 가능한 주의력 모델. Empirical Methods in Natural Language Processing, 2016.

[28] Romain Paulus, Caiming Xiong, 및 Richard Socher. 추상적 요약에 대한 딥 강화 모델. arXiv 사전 인쇄물 arXiv:1705.04304, 2017.

[29] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 정밀하고 압축 가능하며 해석 가능한 트리 주석 학습. 제21회 국제 계산 언어학 회의 및 ACL 제44회 연례 회의록, 433 – 440페이지. ACL, 2006년 7월.

[30] Ofir Press and Lior Wolf. 언어 모델을 개선하기 위한 출력 임베딩 활용. arXiv 전간 arXiv:1608.05859, 2016.

[31] 리코 센리치, 바리 해도우, 알렉산드라 버치. 서브워드 유닛을 이용한 드문 단어의 신경 기계 번역. arXiv 사전 인쇄본 arXiv:1508.07909, 2015.

[32] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, 및 Jeff Dean. 극도로 큰 신경 네트워크: 희소 게이트형 전문가의 혼합 계층. arXiv 프리프린트 arXiv:1701.06538, 2017.

[33] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, 및 Ruslan Salakhutdinov. Dropout: 신경망의 과적합을 방지하는 간단한 방법. Journal of Machine Learning Research, 15(1):1929 – 1958, 2014.

[34] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, Rob Fergus. 엔드투엔드 메모리 네트워크. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, R. Garnett, 편집자, Neural Information Processing Systems 28에 대한 연구 진전, 2440 – 2448쪽. Curran Associates, Inc., 2015.

[35]伊利亞·蘇茨克維爾, 오리오르 비날스, 그리고 퀵 VV 레. 신경망을 이용한 순서에서 순서로 학습. Neural Information Processing Systems 진전에서, 3104 – 3112페이지, 2014년.

[36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, 및 Zbigniew Wojna. 컴퓨터 비전을 위한 인셉션 아키텍처에 대한 재고찰. CoRR, abs/1512.00567, 2015.

[37] Vinyals & Kaiser, Koo, Petrov, Sutskever, 및 Hinton. 'Grammat 외국어로서(Grammar as a foreign language)'. 《Neural Information Processing Systems 진전(Advances in Neural Information Processing Systems)》, 2015.

[38] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, 외. 구글의 인공 신경 기계 번역 시스템: 인간과 기계 번역 사이의 격차 해소. arXiv 사전 인쇄본, arXiv:1609.08144, 2016.

[39] Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, 및 Wei Xu. 신경 기계 번역을 위한 고속 전진 연결이 있는 깊은 순환 모델. CoRR, abs/1606.04199, 2016.

[40] 무화 주, 요장 장, 원량 첸, 민 장, 징보 주. 빠르고 정확한 시프트-감소 구성 요소 구문 분석. ACL 제51차 연례 회의 논문집 (제1권: 롱 페이퍼), 434~443페이지. ACL, 2013년 8월.

Attention Visualizations

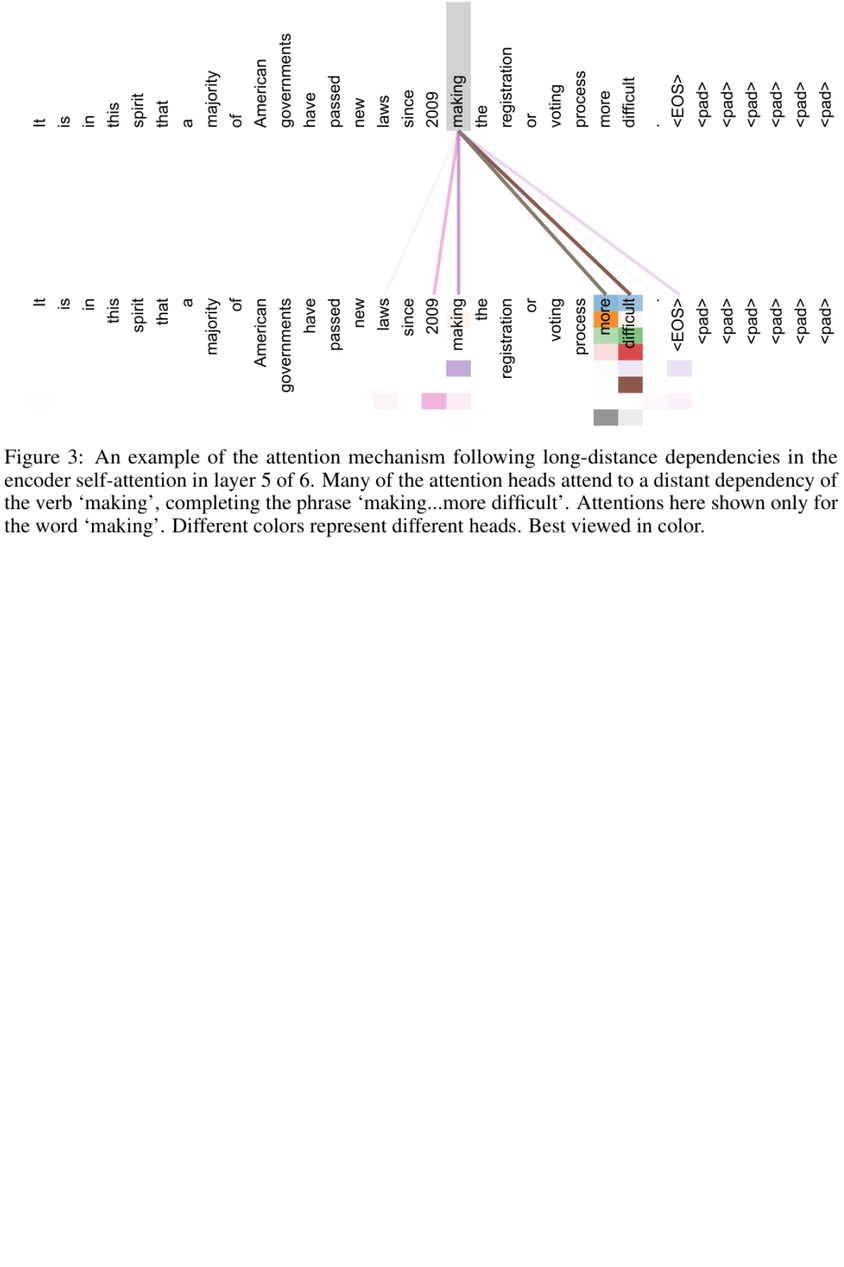


Figure 3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb ‘making’, completing the phrase ‘making...more difficult’. Attentions here shown only for the word ‘making’. Different colors represent different heads. Best viewed in color.

주의력 시각화

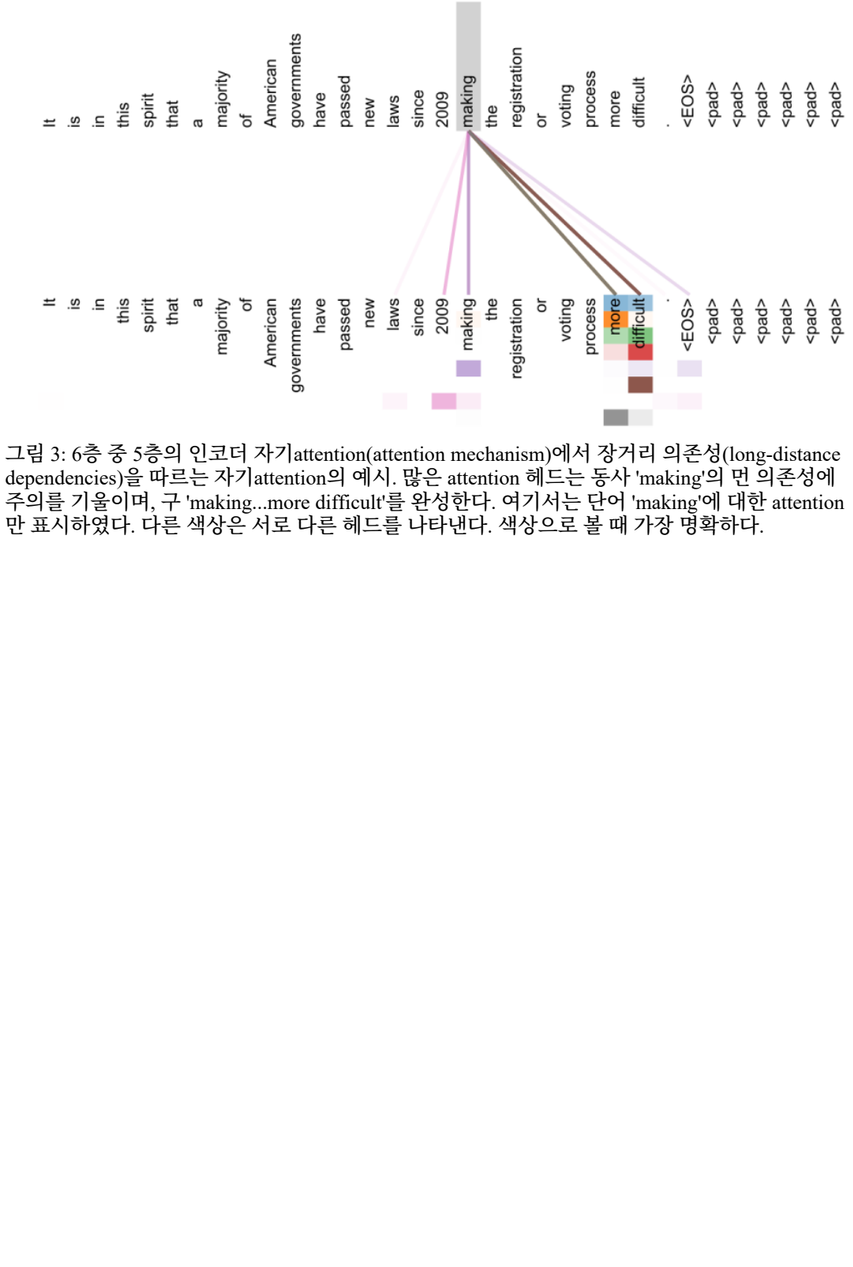


그림 3: 6층 중 5층의 인코더 자기attention(attention mechanism)에서 장거리 의존성(long-distance dependencies)을 따르는 자기attention의 예시. 많은 attention 헤드는 동사 'making'의 먼 의존성에 주의를 기울이며, 구 'making...more difficult'를 완성한다. 여기서는 단어 'making'에 대한 attention만 표시하였다. 다른 색상은 서로 다른 헤드를 나타낸다. 색상으로 볼 때 가장 명확하다.

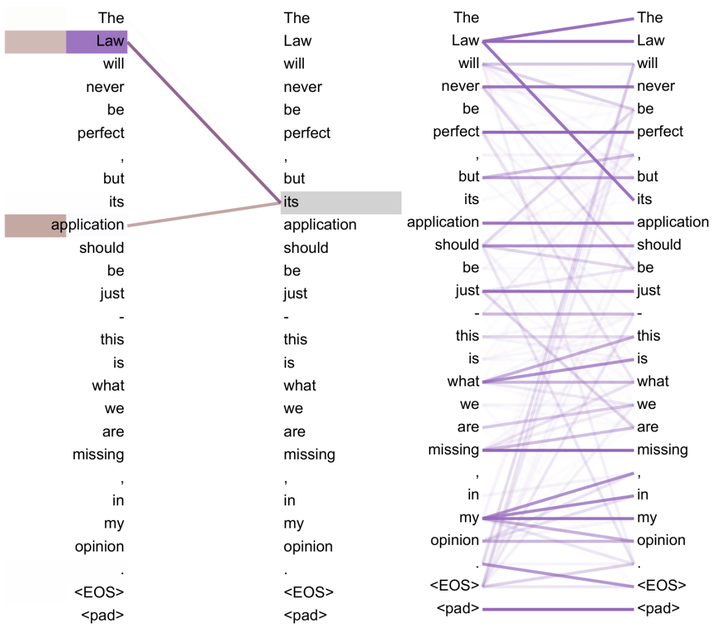
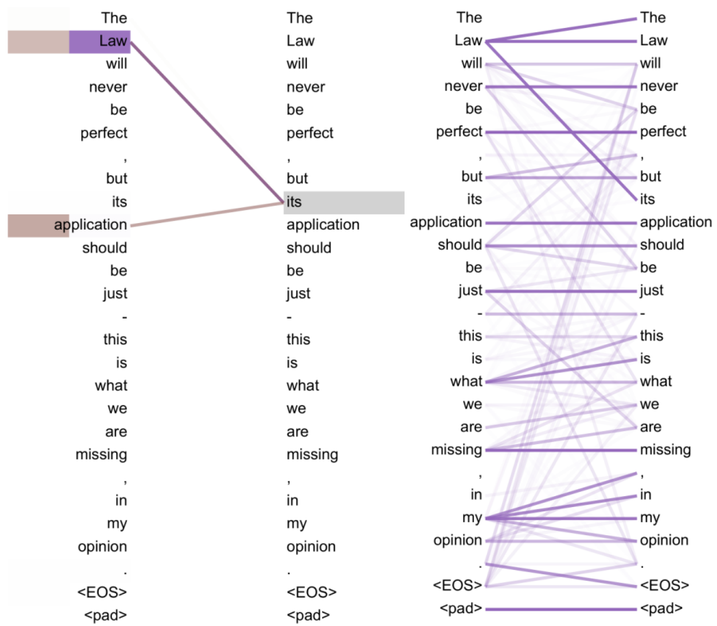


Figure 4: Two attention heads, also in layer 5 of 6, apparently involved in anaphora resolution. Top: Full attentions for head 5. Bottom: Isolated attentions from just the word 'its' for attention heads 5 and 6. Note that the attentions are very sharp for this word.

그림 4: 두 개의attention 헤드, 5/6 레이어에서도 안aphora 해결에 관여하는 것으로 보임. 위: 5번 헤드의 모든 attention. 아래: 단어 'its'에 대한 5번 및 6번 헤드의 isolated attention. 이 단어에 대한 attention이 매우 날카로움을 유의하십시오.



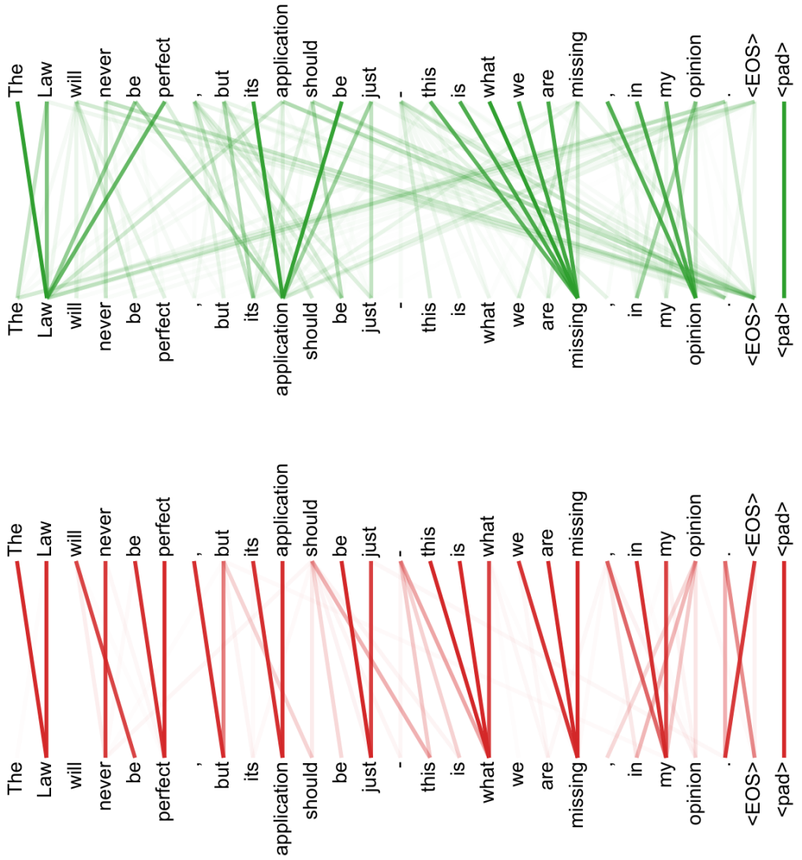


Figure 5: Many of the attention heads exhibit behaviour that seems related to the structure of the sentence. We give two such examples above, from two different heads from the encoder self-attention at layer 5 of 6. The heads clearly learned to perform different tasks.

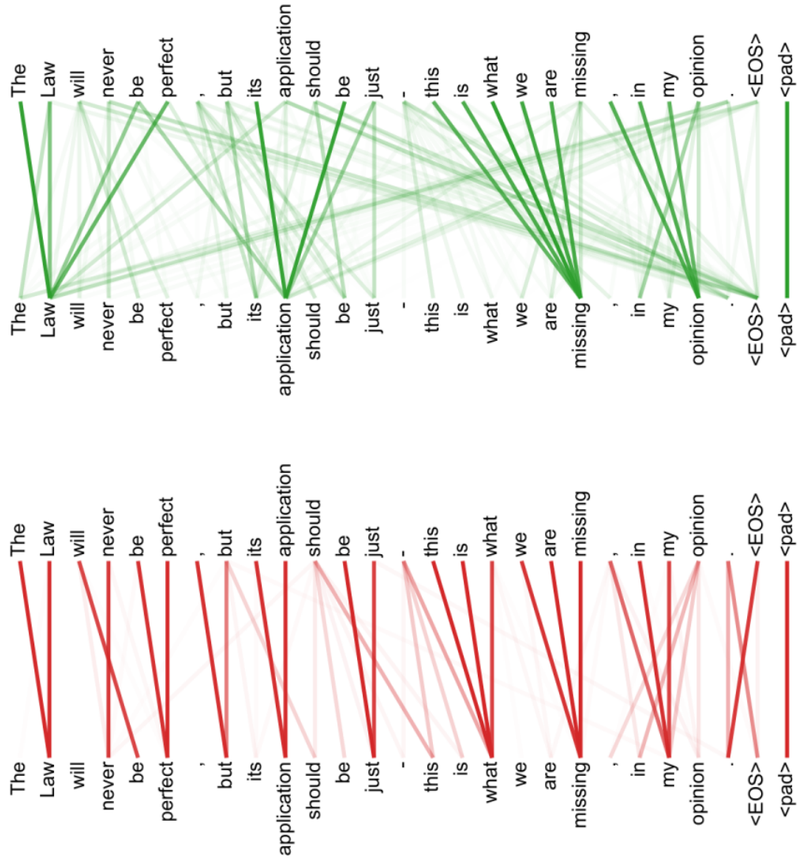


그림 5: 많은 어텐션 헤드가 문장의 구조와 관련된 행동을 보입니다. 위에는 인코더 셀프-어텐션의 6층 중 5층에서 나온 두 다른 헤드의 예시를 제시하였습니다. 이 헤드들은 명확하게 서로 다른 작업을 수행하도록 학습되었습니다.