Attention Is All You Need

어텐션만 있으면 된다

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

초록

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best

주류 시퀀스 변환 모델은 인코더와 디코더를 포함하는 복잡한 순환 신경망 또는 합성곱 신경망을 기반으로 한다.

performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer,

최고 성능 모델은 어텐션 메커니즘을 통해 인코더와 디코더를 연결하기도 한다.

based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to

본 논문에서는 어텐션 메커니즘만을 기반으로 하고 순환과 합성곱을 완전히 배제한 새로운 단순한 네트워크 아키텍처인 Transformer를 제안한다.

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-

2개의 기계 번역 과제에 대한 실험을 통해 이 모델이 품질 면에서 우수하면서도 더 높은 병렬화가 가능하고 학습에 필요한 시간이 크게 줄어든다는 것을 보인다.

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,

우리 모델은 WMT 2014 영어-독일어 번역 과제에서 28.4 BLEU를 달성했으며, 앙상블을 포함한 기존 최고 성능 결과를 2 BLEU 이상 향상시켰다.

our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

WMT 2014 영어-프랑스어 번역 과제에서 우리 모델은 8개의 GPU로 3.5일 동안 학습한 후 41.0이라는 새로운 단일 모델 최고 수준 성능 BLEU 점수를 기록했으며, 이는 문헌에 보고된 최고 성능 모델의 학습 비용 중 극히 일부에 불과하다.

1 Introduction

1 서론

Recurrent neural networks, long short-term memory [12] 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 [29, 2, 5]. Numerous

특히 순환 신경망, 장단기 메모리 [12] 및 게이트 순환 [7] 신경망은 언어 모델링과 기계 번역 [29, 2, 5] 같은 시퀀스 모델링 및 변환 문제에서 최고 수준 성능 접근법으로 확고히 자리 잡았다.

efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [31, 21, 13].

그 후에도 수많은 연구가 순환 언어 모델과 인코더-디코더 아키텍처 [31, 21, 13]의 한계를 확장하기 위해 계속 진행되었다.

---

*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started

∗동등 기여. 저자는 RNN을 자기 어텐션으로 대체할 것을 제안했으며 Jakob은 RNNs를 자기 어텐션으로 대체하자고 제안하고

the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and

Ashish는 Illia와 함께 최초의 Transformer 모델을 설계하고 구현했으며 모든 작업 측면에 핵심적으로 참여했다.

has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head

Noam은 스케일 조정 내적 어텐션과 다중 헤드 Noam은 스케일 조정 내적 어텐션과 다중 헤드

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

어텐션과 매개변수가 없는 위치 표현을 제안했으며 거의 모든 세부 사항에 관여한 또 다른 인물이 되었다.

tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and

Niki는 기존 코드베이스와 tensor2tensor에서 셀 수 없이 많은 모델 변형을 설계하고 구현하며 조정하고 평가했다. 효율적인 추론과 시각화를 담당했다.

efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and

Lukasz와 Aidan은 tensor2tensor의 다양한 부분을 설계하고 구현하는 데 수없이 많은 긴 시간을 보냈으며, 이는 이전 코드베이스를 대체하고 결과를 크게 향상시키며 연구를 대폭 가속했다

implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating

Lukasz와 Aidan은 tensor2tensor의 여러 부분을 설계하고 구현하며 이전 코드베이스를 대체하고 결과를 크게 개선하며 연구를 대폭 가속하는 데 셀 수 없이 많은 날을 보냈다.

our research.

†Google Brain에서 수행한 작업.

†Work performed while at Google Brain.

‡Google Research에서 수행한 작업.

‡Work performed while at Google Research.

제31회 신경 정보 처리 시스템 학회(NIPS 2017), 미국 캘리포니아주 롱비치.

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

신경 정보 처리 시스템에 관한 제31회 학술대회(NIPS 2017), 미국 캘리포니아주 롱비치.

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
순환 모델은 일반적으로 입력 및 출력 시퀀스의 기호 위치를 따라 계산을 인수분해한다.
states h_t, as a function of the previous hidden state h_{t-1} and the input for position t. This inherently
계산 시간의 단계에 위치를 대응시켜, 이전 은닉 상태 ht−1과 위치 t의 입력을 함수로 하는 은닉 상태 ht의 시퀀스를 생성한다.
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
이러한 본질적으로 순차적인 특성은 학습 예제 내 병렬화를 불가능하게 하며, 메모리 제약으로 인해 예제 간 배치 크기가 제한되는 긴 시퀀스에서 이는 더욱 중요해진다.
significant improvements in computational efficiency through factorization tricks [18] and conditional computation [26], while also improving model performance in case of the latter. The fundamental
최근 연구에서는 인수분해 기법 [18]과 조건부 계산 [26]을 통해 계산 효율성을 크게 향상시켰으며, 후자의 경우에는 모델 성능도 향상시켰다.
constraint of sequential computation, however, remains.

그러나 순차 계산이라는 근본적인 제약은 여전히 남아 있다.

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, 16]. In all but a few cases [22], however, such attention mechanisms
어텐션 메커니즘은 다양한 과제에서 강력한 시퀀스 모델링 및 변환 모델의 필수적인 요소가 되었으며, 입력 또는 출력 시퀀스에서의 거리에 관계없이 종속성을 모델링할 수 있게 한다 [2, 16].
are used in conjunction with a recurrent network.

그러나 극히 일부 경우 [22]를 제외하면 이러한 어텐션 메커니즘은 순환 네트워크와 함께 사용된다.

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.
본 연구에서는 순환을 배제하고 대신 어텐션 메커니즘만을 사용하여 입력과 출력 간의 전역 종속성을 포착하는 모델 아키텍처인 Transformer를 제안한다.
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.

Transformer는 훨씬 더 높은 병렬화를 가능하게 하며, 8개의 P100 GPU에서 불과 12시간 동안 학습한 후 번역 품질에서 새로운 최고 수준 성능을 달성할 수 있다.

2 Background

2 배경

The goal of reducing sequential computation also forms the foundation of the Extended Neural GPU [20], ByteNet [15] and ConvS2S [8], 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,
순차 계산을 줄이려는 목표는 Extended Neural GPU [20], ByteNet [15] 및 ConvS2S [8]의 기반이기도 하다. 이 모델들은 모두 합성곱 신경망을 기본 구성 요소로 사용하며, 모든 입력 및 출력 위치의 은닉 표현을 병렬로 계산한다.
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
이러한 모델에서 임의의 두 입력 또는 출력 위치의 신호를 연결하는 데 필요한 연산 수는 위치 간 거리에 따라 증가하며, ConvS2S에서는 선형적으로, ByteNet에서는 로그적으로 증가한다.
it more difficult to learn dependencies between distant positions [11]. In the Transformer this is
이로 인해 서로 먼 위치 간의 종속성을 학습하기가 더 어려워진다 [11].
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.

Transformer에서는 이를 일정한 연산 수로 줄인다. 다만 어텐션 가중치가 적용된 위치를 평균화하기 때문에 유효 해상도가 감소하는 대가를 치러야 하며, 3.2절에서 설명하는 다중 헤드 어텐션을 사용하여 이러한 효과를 상쇄한다.

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
자기 어텐션은 인트라 어텐션이라고도 하며, 시퀀스의 표현을 계산하기 위해 단일 시퀀스의 서로 다른 위치를 연결하는 어텐션 메커니즘이다.
used successfully in a variety of tasks including reading comprehension, abstractive summarization, textual entailment and learning task-independent sentence representations [4, 22, 23, 19].

자기 어텐션은 독해, 추상적 요약, 텍스트 함의 및 과제와 무관한 문장 표현 학습을 비롯한 다양한 과제에서 성공적으로 사용되었다 [4, 22, 23, 19].

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 [28].

종단 간 메모리 네트워크는 시퀀스 정렬 순환 대신 순환 어텐션 메커니즘을 기반으로 하며, 간단한 언어 질의응답 및 언어 모델링 과제에서 우수한 성능을 보이는 것으로 나타났다 [28].

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
그러나 우리가 아는 한 Transformer는 시퀀스 정렬 RNN이나 합성곱을 사용하지 않고 전적으로 자기 어텐션에 의존하여 입력과 출력의 표현을 계산하는 최초의 변환 모델이다.
self-attention and discuss its advantages over models such as [14, 15] and [8].

다음 절에서는 Transformer를 설명하고, 자기 어텐션을 사용하는 근거를 제시하며, [14, 15] 및 [8]과 같은 모델에 비해 자기 어텐션이 갖는 장점을 논의한다 [14, 15] [8].

3 Model Architecture

3 모델 아키텍처

Most competitive neural sequence transduction models have an encoder-decoder structure [5, 2, 29].
대부분의 경쟁력 있는 신경 시퀀스 변환 모델은 인코더-디코더 구조를 갖는다 [5, 2, 29].
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
여기서 인코더는 기호 표현의 입력 시퀀스 (x1, ..., xn)를 연속 표현 z = (z1, ..., zn)의 시퀀스로 매핑한다.
sequence $(y_1, ..., y_m)$ of symbols one element at a time. At each step the model is auto-regressive
z가 주어지면 디코더는 출력 기호 시퀀스 (y1, ..., ym)를 한 번에 하나의 원소씩 생성한다.
[9], consuming the previously generated symbols as additional input when generating the next.

각 단계에서 모델은 자기회귀적이며 [9], 다음 기호를 생성할 때 이전에 생성된 기호를 추가 입력으로 사용한다.

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.

Transformer는 인코더와 디코더 모두에 적층된 자기 어텐션 및 위치별 완전 연결 계층을 사용하여 이러한 전체 아키텍처를 따르며, 이는 각각 그림 1의 왼쪽과 오른쪽 절반에 제시되어 있다.

3.1 Encoder and Decoder Stacks

3.1 인코더 및 디코더 스택

Encoder: The encoder is composed of a stack of $N = 6$ identical layers. Each layer has two
인코더: 인코더는 N = 6개의 동일한 계층을 쌓은 구조로 구성된다.
sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-

각 계층에는 2개의 하위 계층이 있다.

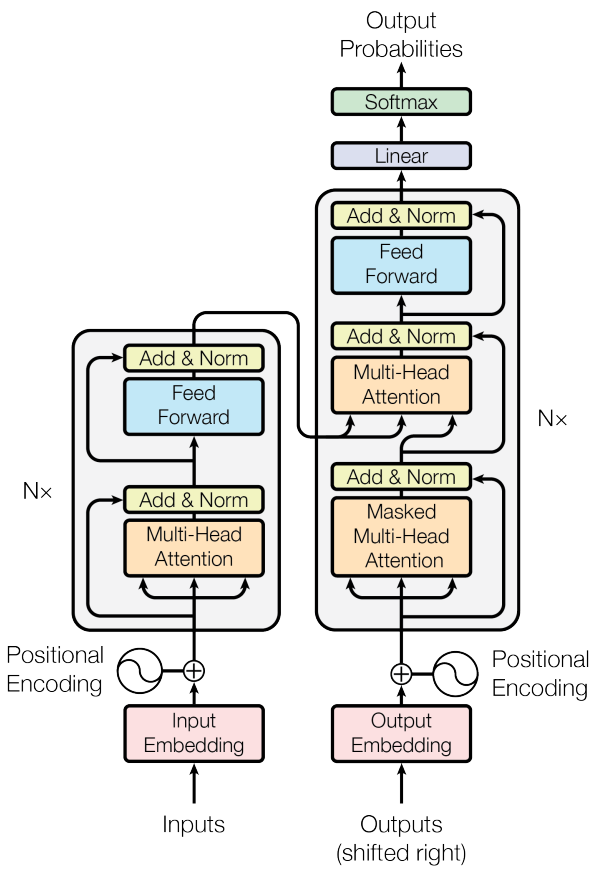


Figure 1: The Transformer - model architecture.

그림 1: Transformer - 모델 아키텍처.

wise fully connected feed-forward network. We employ a residual connection [10] around each of
첫 번째는 다중 헤드 자기 어텐션 메커니즘이고, 두 번째는 단순한 위치별 완전 연결 피드포워드 네트워크이다.
the two sub-layers, followed by layer normalization [1]. That is, the output of each sub-layer is
두 하위 계층 각각에 잔차 연결 [10]을 적용한 후 레이어 정규화 [1]를 적용한다.
$\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x))$, 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
즉, 각 하위 계층의 출력은 LayerNorm(x + Sublayer(x))이며, 여기서 Sublayer(x)는 하위 계층 자체가 구현하는
함수이다.
layers, produce outputs of dimension $d_{\text{model}} = 512$.

이러한 잔차 연결을 용이하게 하기 위해 모델의 모든 하위 계층과 임베딩 계층은 dmodel = 512 차원의 출력을 생성한다.

Decoder: The decoder is also composed of a stack of $N = 6$ identical layers. In addition to the two
디코더: 디코더 역시 N = 6개의 동일한 계층을 쌓아 구성한다.
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
각 인코더 계층의 두 하위 계층에 더해, 디코더는 인코더 스택의 출력에 대해 다중 헤드 어텐션을 수행하는 세 번째 하위 계층을 추가한다.
around each of the sub-layers, followed by layer normalization. We also modify the self-attention
인코더와 마찬가지로 각 하위 계층 주위에 잔차 연결을 적용한 후 레이어 정규화를 적용한다.
sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This
또한 디코더 스택의 자기 어텐션 하위 계층을 수정하여 각 위치가 이후 위치를 참조하지 못하도록 한다.
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.

이러한 마스킹은 출력 임베딩을 1개 위치만큼 오프셋하는 것과 결합되어 위치 i의 예측이 i보다 작은 위치의 알려진 출력에만 의존하도록 보장한다.

3.2 Attention

3.2 어텐션

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
어텐션 함수는 쿼리와 키-값 쌍 집합을 출력에 매핑하는 함수로 설명할 수 있으며, 쿼리, 키, 값 및 출력은 모두 벡터이다.
of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

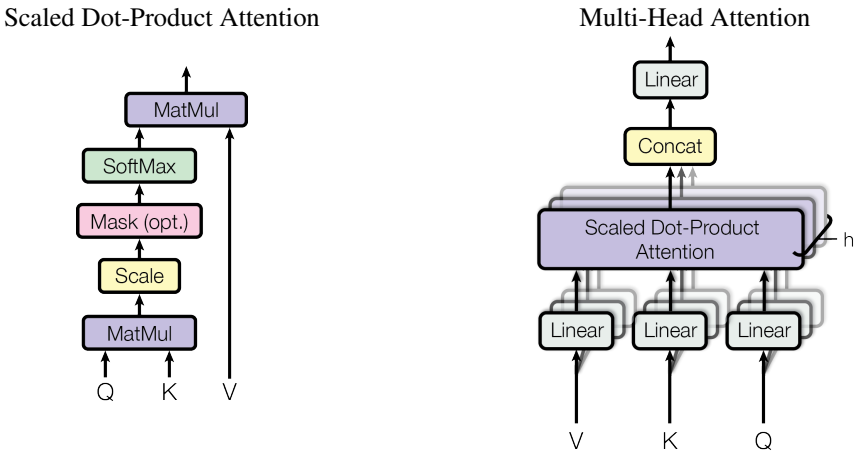
출력은 값들의 가중합으로 계산되며, 각 값에 할당되는 가중치는 쿼리와 해당 키의 호환성 함수로 계산한다.

3.2.1 Scaled Dot-Product Attention

3.2.1 스케일 조정 내적 어텐션

We call our particular attention "Scaled Dot-Product Attention" (Figure 2). The input consists of
우리는 우리의 특정 어텐션을 "스케일 조정 내적 어텐션"이라고 부른다(그림 2).
queries and keys of dimension d_k, and values of dimension d_v. We compute the dot products of the

입력은 dk 차원의 쿼리와 키, 그리고 dv 차원의 값으로 구성된다. 다음의 내적을 계산한다



스케일 조정 내적 어텐션 다중 헤드 어텐션

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

그림 2: (왼쪽) 스케일 조정 내적 어텐션. (오른쪽) 다중 헤드 어텐션은 여러 어텐션 계층이 병렬로 실행되는 구조이다.

query with all keys, divide each by $\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values.

모든 키와 쿼리의 내적을 계산하고, 각각을 √dk로 나눈 다음 소프트맥스 함수를 적용하여 값에 대한 가중치를 얻는다.

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

실제로는 쿼리 집합에 대한 어텐션 함수를 행렬 Q에 함께 묶어 동시에 계산한다. 키와 값도 각각 행렬 K 및 V에 함께 묶는다.

the matrix of outputs as:

출력 행렬은 다음과 같이 계산한다.

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

어텐션(Q, K, V) = 소프트맥스 함수(QKT √dk)V (1)

The two most commonly used attention functions are additive attention [2], and dot-product (multi-plicative) 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

가장 일반적으로 사용되는 2가지 어텐션 함수는 가산 어텐션 [2]과 내적 어텐션(곱셈 어텐션)이다. 내적 어텐션은 스케일링 인자를 제외하면 우리의 알고리즘과 동일하다. √dk이다.

a single hidden layer. While the two are similar in theoretical complexity, dot-product attention is

가산 어텐션은 단일 은닉 계층을 갖는 피드포워드 네트워크를 사용하여 호환성 함수를 계산한다.

much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code.

두 방법은 이론적 복잡도가 유사하지만, 내적 어텐션은 고도로 최적화된 행렬 곱셈 코드를 사용하여 구현할 수 있으므로 실제로는 훨씬 빠르고 메모리 효율적이다.

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

dk의 값이 작을 때는 2가지 메커니즘의 성능이 유사하지만, dk의 값이 클 때는 가산 어텐션이 스케일링하지 않은 내적 어텐션보다 우수하다 [3].

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 $\frac{1}{\sqrt{d_k}}$.

dk의 값이 클수록 내적의 크기가 커져 소프트맥스 함수가 극도로 작은 그래디언트를 갖는 영역으로 들어간다고 추측한다 4. 이 효과를 상쇄하기 위해 내적을 √dk로 스케일 조정한다.

3.2.2 Multi-Head Attention

3.2.2 다중 헤드 어텐션

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

dmodel차원의 키, 값 및 쿼리를 사용하는 단일 어텐션 함수를 수행하는 대신, 서로 다른 학습된 선형 사상을 사용하여 쿼리, 키 및 값을 각각 dk, dk 및 dv 차원으로 h회 선형 사상하는 것이 유리하다는 사실을 발견했다.

queries, keys and values we then perform the attention function in parallel, yielding d_v-dimensional output values. These are concatenated and once again projected, resulting in the final values, as

그런 다음 이렇게 사상된 쿼리, 키 및 값 각각에 대해 어텐션 함수를 병렬로 수행하여 dv차원의 출력 값을 얻는다.

depicted in Figure 2.

이들을 연결한 후 다시 선형 사상하여 최종 값을 얻으며, 그 구조는 그림 2에 제시되어 있다.

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.

다중 헤드 어텐션을 사용하면 모델이 서로 다른 위치에서 서로 다른 표현 부분공간의 정보에 공동으로 주의를 기울일 수 있다. 단일 어텐션 헤드를 사용하면 평균화로 인해 이러한 기능이 제한된다.

[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.

4내적이 커지는 이유를 설명하기 위해 q와 k의 성분이 평균 0, 분산 1인 독립 확률 변수라고 가정하자. 그렇다면 이들의 내적, q · k = Pdk i=1 qiki는 평균이 0이고 분산이 dk이다.

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

여기서 headi = Attention(QW Q i , KW K i , V W V이다.

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 $W^O \in \mathbb{R}^{hd_v \times d_\text{model}}$.

여기서 투영은 매개변수 행렬 W Q이다.

In this work we employ $h = 8$ parallel attention layers, or heads. For each of these we use

이 연구에서는 h = 8개의 병렬 어텐션 계층, 즉 헤드를 사용한다.

$d_k = d_v = d_\text{model}/h = 64$. Due to the reduced dimension of each head, the total computational cost

각 헤드에 대해 dk = dv = dmodel/h = 64를 사용한다.

is similar to that of single-head attention with full dimensionality.

각 헤드의 차원이 줄어들기 때문에 전체 계산 비용은 전체 차원을 사용하는 단일 헤드 어텐션의 경우와 유사하다.

3.2.3 Applications of Attention in our Model

3.2.3 우리 모델에서의 어텐션 적용

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

Transformer는 다중 헤드 어텐션을 3가지 방식으로 사용한다.

- 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 [31, 2, 8].

"인코더-디코더 어텐션" 계층에서는 쿼리가 이전 디코더 계층에서 나오고, 메모리 키와 값은 인코더의 출력에서 나온다. 이를 통해 디코더의 모든 위치가 입력 시퀀스의 모든 위치에 어텐션할 수 있다. 이는 [31, 2, 8]과 같은 시퀀스-투-시퀀스 모델에서 사용하는 전형적인 인코더-디코더 어텐션 메커니즘을 모방한다.

- 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.

인코더에는 자기 어텐션 계층이 포함된다. 자기 어텐션 계층에서는 모든 키, 값, 쿼리가 동일한 곳에서 나오며, 이 경우에는 인코더의 이전 계층 출력에서 나온다. 인코더의 각 위치는 인코더 이전 계층의 모든 위치에 어텐션할 수 있다.

- 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.

마찬가지로 디코더의 자기 어텐션 계층에서는 디코더의 각 위치가 해당 위치를 포함하여 그 이전까지의 디코더 내 모든 위치에 어텐션할 수 있다. 자기회귀 속성을 유지하려면 디코더에서 왼쪽 방향으로 정보가 흐르는 것을 방지해야 한다. 이를 위해 스케일 조정 내적 어텐션 내부에서 허용되지 않는 연결에 해당하는 소프트맥스 함수 입력의 모든 값을 마스킹하여 −∞으로 설정한다. 그림 2를 참조하라.

3.3 Position-wise Feed-Forward Networks

3.3 위치별 피드포워드 네트워크

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

어텐션 하위 계층과 더불어 인코더와 디코더의 각 계층에는 완전 연결 피드포워드 네트워크가 포함되며, 이 네트워크는 각 위치에 개별적으로 동일하게 적용된다.

consists of two linear transformations with a ReLU activation in between.

이는 중간에 ReLU 활성화가 있는 2개의 선형 변환으로 구성된다.

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

FFN(x) = max(0, xW1 + b1)W2 + b2 (2)로 정의한다.

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.

선형 변환은 서로 다른 위치에서 동일하지만, 계층마다 서로 다른 매개변수를 사용한다. 이를 설명하는 또 다른 방법은 커널 크기가 1인 2개의 합성곱으로 보는 것이다.

The dimensionality of input and output is $d_\text{model} = 512$, and the inner-layer has dimensionality $d_{ff} = 2048$.

입력과 출력의 차원은 dmodel = 512이고, 내부 계층의 차원은 dff = 2048이다.

3.4 Embeddings and Softmax

3.4 임베딩과 소프트맥스

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 transfor-

다른 시퀀스 변환 모델과 마찬가지로, 학습된 임베딩을 사용하여 입력 토큰과 출력 토큰을 dmodel 차원의 벡터로 변환한다.

mation and softmax function to convert the decoder output to predicted next-token probabilities. In

또한 일반적인 학습된 선형 변환과 소프트맥스 함수를 사용하여 디코더 출력을 예측된 다음 토큰 확률로 변환한다.

our model, we share the same weight matrix between the two embedding layers and the pre-softmax linear transformation, similar to [24]. In the embedding layers, we multiply those weights by $\sqrt{d_\text{model}}$.

우리 모델에서는 [24]과 유사하게 2개의 임베딩 계층과 소프트맥스 이전 선형 변환 사이에서 동일한 가중치 행렬을 공유한다. 임베딩 계층에서는 해당 가중치에 √dmodel을 곱한다.

3.5 Positional Encoding

3.5 위치 인코딩

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

우리 모델에는 순환도 합성곱도 없으므로 모델이 시퀀스의 순서를 활용하려면 시퀀스 내 토큰의 상대적 또는 절대적 위치에 관한 정보를 주입해야 한다. 이를 위해 입력 임베딩에 "위치 인코딩"을 추가한다

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.

표 1: 서로 다른 계층 유형의 최대 경로 길이, 계층당 복잡도 및 순차 연산의 최소 횟수. n은 시퀀스 길이이고, d은 표현 차원이며, k는 합성곱의 커널 크기이고, r은 제한된 자기 어텐션에서 이웃의 크기이다.

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)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

계층 유형 계층당 복잡도 순차 연산 최대 경로 길이 자기 어텐션 O(n2 · d) O(1) O(1) 순환 O(n · d2) O(n) O(n) 합성곱 O(k · n · d2) O(1) O(logk(n)) 제한된 자기 어텐션 O(r · n · d) O(1) O(n/r)

bottoms of the encoder and decoder stacks. The positional encodings have the same dimension d_{model}
인코더와 디코더 스택의 하단이다.
as the embeddings, so that the two can be summed. There are many choices of positional encodings,
위치 인코딩은 임베딩과 동일한 차원 dmodel을 가지므로 두 벡터를 더할 수 있다.
learned and fixed [8].
학습되는 위치 인코딩과 고정된 위치 인코딩을 비롯해 다양한 위치 인코딩이 존재한다 [8].

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

본 연구에서는 서로 다른 주파수의 사인 함수와 코사인 함수를 사용한다.

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

where pos is the position and i is the dimension. That is, each dimension of the positional encoding
여기서 pos은 위치이고 i은 차원이다.
corresponds to a sinusoid. The wavelengths form a geometric progression from 2π to $10000 \cdot 2\pi$. We
즉, 위치 인코딩의 각 차원은 하나의 사인파에 대응한다. 파장은 2π에서 10000 · 2π까지 기하급수적으로 증가한다.
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}.
이 함수를 선택한 이유는 고정된 오프셋 k에 대해 PEpos+k를 의 선형 함수로 표현할 수 있으므로, 모델이 상대적 위치에 따라 쉽게 어텐션하는 방법을 학습할 수 있을 것이라고 가정했기 때문이다.

We also experimented with using learned positional embeddings [8] instead, and found that the two versions produced nearly identical results (see Table 3 row (E)). We chose the sinusoidal version
대신 학습되는 위치 임베딩 [8]을 사용하는 실험도 수행했으며, 두 버전이 거의 동일한 결과를 산출한다는 것을 확인했다(표 3의 행 (E) 참조).
because it may allow the model to extrapolate to sequence lengths longer than the ones encountered during training.
사인파 버전을 선택한 이유는 모델이 학습 중 접한 것보다 긴 시퀀스 길이로 외삽할 수 있게 해줄 가능성이 있기 때문이다.

4 Why Self-Attention

4 자기 어텐션

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
이 절에서는 하나의 가변 길이 기호 표현 시퀀스 (x1, ..., xn)를 동일한 길이의 다른 시퀀스 (z1, ..., zn)로 매핑하는 데 일반적으로 사용되는 자기 어텐션 계층과 순환 계층 및 합성곱 계층의 여러 측면을 비교한다. 여기서 xi, zi ∈Rd이며, 이는 일반적인 시퀀스 변환 인코더 또는 디코더의 은닉 계층과 같은 것이다.
consider three desiderata.
자기 어텐션을 사용하는 근거로 3가지 요구 사항을 고려한다.

One is the total computational complexity per layer. Another is the amount of computation that can
첫 번째는 계층당 전체 계산 복잡도이다.
be parallelized, as measured by the minimum number of sequential operations required.
두 번째는 병렬화할 수 있는 계산량이며, 이는 필요한 순차 연산의 최소 횟수로 측정한다.

The third is the path length between long-range dependencies in the network. Learning long-range
세 번째는 네트워크에서 장거리 의존성 사이의 경로 길이이다.
dependencies is a key challenge in many sequence transduction tasks. One key factor affecting the
장거리 의존성을 학습하는 것은 많은 시퀀스 변환 과제에서 핵심적인 난제이다.
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
이러한 의존성을 학습하는 능력에 영향을 미치는 핵심 요인 중 하나는 순방향 신호와 역방향 신호가 네트워크에서 통과해야 하는 경로의 길이이다.
and output sequences, the easier it is to learn long-range dependencies [11]. Hence we also compare
입력 시퀀스와 출력 시퀀스의 임의의 위치 조합 사이의 이러한 경로가 짧을수록 장거리 의존성을 더 쉽게 학습할 수 있다 [11].
the maximum path length between any two input and output positions in networks composed of the different layer types.
따라서 서로 다른 계층 유형으로 구성된 네트워크에서 입력 위치와 출력 위치의 임의의 두 지점 사이의 최대 경로 길이도 비교한다.

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
표 1에 제시된 것처럼 자기 어텐션 계층은 일정한 횟수의 순차 실행 연산으로 모든 위치를 연결하는 반면, 순환 계층은 O(n)회의 순차 연산을 필요로 한다.
computational complexity, self-attention layers are faster than recurrent layers when the sequence 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 [31] and byte-pair [25] representations. To improve computational performance for tasks involving
계산 복잡도 측면에서 자기 어텐션 계층은 시퀀스 길이 n이 표현 차원 d보다 작을 때 순환 계층보다 빠르다. 이는 word-piece [31] 및 byte-pair [25] 표현과 같이 최신 기계 번역 모델에서 사용하는 문장 표현의 경우에 대부분 성립한다.
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
매우 긴 시퀀스를 포함하는 작업의 계산 성능을 향상시키기 위해, 자기 어텐션이 입력 시퀀스에서 해당 출력 위치를 중심으로 크기가 r인 이웃 영역만 고려하도록 제한할 수 있다.
path length to $O(n/r)$. We plan to investigate this approach further in future work.

이렇게 하면 최대 경로 길이가 O(n/r)로 증가한다. 향후 연구에서 이 접근법을 더 자세히 조사할 계획이다.

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,
커널 폭이 k < n인 단일 합성곱 계층은 모든 입력 위치와 출력 위치의 쌍을 연결하지 않는다.
or $O(log_k(n))$ in the case of dilated convolutions [15], increasing the length of the longest paths between any two positions in the network. Convolutional layers are generally more expensive than
이를 위해 연속 커널의 경우 O(n/k)개의 합성곱 계층을 쌓아야 하며, 확장 합성곱의 경우에는 O(logk(n))개의 계층을 쌓아야 한다 [15]. 이에 따라 네트워크에서 임의의 두 위치 사이의 가장 긴 경로 길이가 증가한다.
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
합성곱 계층은 일반적으로 재current 계층보다 k배 더 많은 비용이 든다. 그러나 분리 가능한 합성곱 [6]은 복잡도를 크게 낮춰 O(k · n · d + n · d2)로 만든다.
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.

그러나 k = n인 경우에도 분리 가능한 합성곱의 복잡도는 자기 어텐션 계층과 위치별 피드포워드 계층의 조합과 같으며, 이것이 본 모델에서 사용하는 접근법이다.

As side benefit, self-attention could yield more interpretable models. We inspect attention distributions
부수적인 이점으로 자기 어텐션은 더 해석 가능한 모델을 산출할 수 있다.
from our models and present and discuss examples in the appendix. Not only do individual attention
우리 모델의 어텐션 분포를 조사하고, 부록에서 예를 제시하고 논의한다.
heads clearly learn to perform different tasks, many appear to exhibit behavior related to the syntactic and semantic structure of the sentences.

개별 어텐션 헤드는 서로 다른 작업을 명확히 학습할 뿐만 아니라, 많은 헤드가 문장의 통사적·의미적 구조와 관련된 행동을 보이는 것으로 나타난다.

5 Training

5 학습

This section describes the training regime for our models.

이 절에서는 우리 모델의 학습 방식을 설명한다.

5.1 Training Data and Batching

5.1 학습 데이터와 배칭

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-
약 4.5 million개의 문장 쌍으로 구성된 표준 WMT 2014 영어-독일어 데이터셋으로 학습했다.
target vocabulary of about 37000 tokens. For English-French, we used the significantly larger WMT
문장은 바이트 쌍 인코딩 [3]을 사용해 인코딩했으며, 이 인코딩은 약 37000개의 토큰으로 구성된 공유 소스-타깃 어휘를 사용한다.
2014 English-French dataset consisting of 36M sentences and split tokens into a 32000 word-piece vocabulary [31]. Sentence pairs were batched together by approximate sequence length. Each training
영어-프랑스어의 경우, 36M개의 문장으로 구성된 훨씬 더 큰 WMT 2014 영어-프랑스어 데이터셋을 사용하고 토큰을 32000개의 워드피스 어휘 [31]로 분할했다. 문장 쌍은 대략적인 시퀀스 길이에 따라 함께 배치했다.
batch contained a set of sentence pairs containing approximately 25000 source tokens and 25000 target tokens.

각 학습 배치에는 약 25000개의 소스 토큰과 25000개의 타깃 토큰을 포함하는 문장 쌍 집합이 들어 있었다.

5.2 Hardware and Schedule

5.2 하드웨어와 일정

We trained our models on one machine with 8 NVIDIA P100 GPUs. For our base models using
8개의 NVIDIA P100 GPU가 장착된 한 대의 머신에서 모델을 학습했다.
the hyperparameters described throughout the paper, each training step took about 0.4 seconds. We
논문 전체에서 설명한 하이퍼파라미터를 사용하는 base 모델의 경우, 각 학습 단계에 약 0.4초가 걸렸다.
trained the base models for a total of 100,000 steps or 12 hours. For our big models,(described on the
base 모델을 총 100000단계 또는 12시간 동안 학습했다.
bottom line of table 3), step time was 1.0 seconds. The big models were trained for 300,000 steps
big 모델(표 3의 마지막 행에 설명된 모델)의 경우 단계 시간은 1.0초였다.
(3.5 days).

big 모델은 300000단계(3.5일) 동안 학습했다.

5.3 Optimizer

5.3 옵티마이저

We used the Adam optimizer [17] with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. We varied the learning
Adam 옵티마이저 [17]를 β1 = 0.9, β2 = 0.98 및 ε = 10−9와 함께 사용했다.
rate over the course of training, according to the formula:

학습 과정에서 다음 공식에 따라 학습률을 변화시켰다:

$$lrate = d_{\text{model}}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5}) \tag{3}$$

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
이는 처음 warmup_steps개의 학습 단계 steps 동안 학습률을 선형적으로 증가시키고, 그 이후에는 단계 수의 역제곱근에 비례하여 학습률을 감소시키는 것에 해당한다. 다음을 사용했다
$warmup\_steps = 4000$.

5.4 Regularization

5.4 정규화

We employ three types of regularization during training:

학습 중에 3가지 유형의 정규화를 사용한다:

Residual Dropout We apply dropout [27] 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
잔차 드롭아웃 각 하위 계층의 출력에 드롭아웃 [27]을 적용한 후, 이를 하위 계층의 입력에 더하고 정규화한다.
positional encodings in both the encoder and decoder stacks. For the base model, we use a rate of
또한 인코더와 디코더 스택 모두에서 임베딩과 위치 인코딩의 합에 드롭아웃을 적용한다. base 모델에서는 다음 비율을 사용한다
$P_{drop} = 0.1$.

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.

표 2: Transformer는 영어-독일어 및 영어-프랑스어 newstest2014 테스트에서 기존 최첨단 모델보다 더 높은 BLEU 점수를 기록하며, 학습 비용의 일부만 사용한다.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	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**	**41.0**	$2.3 \cdot 10^{19}$	

BLEU 학습 비용 (FLOPs) 모델 EN-DE EN-FR EN-DE EN-FR ByteNet [15] 23.75 Deep-Att + PosUnk [32] 39.2 1.0 · 1020 GNMT + RL [31] 24.6 39.92 2.3 · 1019 1.4 · 1020 ConvS2S [8] 25.16 40.46 9.6 · 1018 1.5 · 1020 MoE [26] 26.03 40.56 2.0 · 1019 1.2 · 1020 Deep-Att + PosUnk Ensemble [32] 40.4 8.0 · 1020 GNMT + RL Ensemble [31] 26.30 41.16 1.8 · 1020 1.1 · 1021 ConvS2S Ensemble [8] 26.36 41.29 7.7 · 1019 1.2 · 1021 Transformer (base model) 27.3 38.1 3.3 · 1018 Transformer (big) 28.4 41.0 2.3 · 1019

Label Smoothing During training, we employed label smoothing of value $\epsilon_{ls} = 0.1$ [30]. This
레이블 평활화 학습 중에는 εls = 0.1의 레이블 평활화를 사용했다 [30].
hurts perplexity, as the model learns to be more unsure, but improves accuracy and BLEU score.

모델이 더 불확실해지도록 학습되므로 퍼플렉서티는 악화되지만, 정확도와 BLEU 점수는 향상된다.

6 Results

6 결과

6.1 Machine Translation

6.1 기계 번역

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
WMT 2014 영어-독일어 번역 과제에서 대형 Transformer 모델(표 2의 Transformer (big))은 앙상블을 포함한 기존에 보고된 최상의 모델보다 2.0 BLEU 이상 높은 성능을 내며, 28.4라는 새로운 최첨단 BLEU 점수를 달성한다.
listed in the bottom line of Table 3. Training took 3.5 days on 8 P100 GPUs. Even our base model
이 모델의 구성은 표 3의 마지막 행에 제시되어 있다. 학습에는 8개의 P100 GPU에서 3.5일이 걸렸다.
surpasses all previously published models and ensembles, at a fraction of the training cost of any of the competitive models.

우리의 기본 모델은 경쟁 모델 중 어느 모델보다도 훨씬 적은 학습 비용으로 기존에 발표된 모든 모델과 앙상블을 능가한다.

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
WMT 2014 영어-프랑스어 번역 과제에서 우리의 대형 모델은 41.0의 BLEU 점수를 달성하며, 기존 최첨단 모델의 학습 비용의 1/4 미만으로 기존에 발표된 모든 단일 모델보다 높은 성능을 낸다.
dropout rate $P_{drop} = 0.1$, instead of 0.3.

영어-프랑스어용으로 학습한 Transformer (big) 모델은 0.3 대신 드롭아웃 비율 Pdrop = 0.1을 사용했다.

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
기본 모델에는 10분 간격으로 저장된 마지막 5개 체크포인트의 평균으로 얻은 단일 모델을 사용했다. 대형 모델에는 마지막 20개 체크포인트의 평균을 사용했다.
used beam search with a beam size of 4 and length penalty $\alpha = 0.6$ [31]. These hyperparameters
빔 크기가 4인 빔 탐색과 길이 패널티 α = 0.6 [31]을 사용했다.
were chosen after experimentation on the development set. We set the maximum output length during
이러한 하이퍼파라미터는 개발 세트에서 실험한 후 선택했다.
inference to input length + 50, but terminate early when possible [31].

추론 중 최대 출력 길이는 입력 길이 + 50으로 설정했지만, 가능한 경우 조기에 종료했다 [31].

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
표 2에는 우리의 결과를 요약하고, 문헌에 보고된 다른 모델 아키텍처와 우리의 번역 품질 및 학습 비용을 비교한 결과가 제시되어 있다.
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].

모델 학습에 사용된 부동소수점 연산 수는 학습 시간, 사용한 GPU 수, 각 GPU의 지속 가능한 단정밀도 부동소수점 연산 능력 추정치를 곱하여 추정한다 5.

6.2 Model Variations

6.2 모델 변형

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 development set, newstest2013. We used beam search as described in the previous section, but no
Transformer의 다양한 구성 요소의 중요성을 평가하기 위해 기본 모델을 여러 방식으로 변경하고, 개발 세트인 newstest2013의 영어-독일어 번역에서 성능 변화를 측정했다.
checkpoint averaging. We present these results in Table 3.

이전 절에서 설명한 대로 빔 탐색을 사용했지만, 체크포인트 평균화는 사용하지 않았다. 이 결과를 표 3에 제시한다.

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
표 3의 (A)행에서는 3.2.2절에서 설명한 대로 연산량을 일정하게 유지하면서 어텐션 헤드 수와 어텐션 키 및 값 차원을 변경한다.
attention is 0.9 BLEU worse than the best setting, quality also drops off with too many heads.

단일 헤드 어텐션은 최상의 설정보다 0.9 BLEU 낮으며, 헤드가 너무 많아도 품질이 저하된다.

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

5K80, K40, M40 및 P100에는 각각 2.8, 3.7, 6.0 및 9.5 TFLOPS의 값을 사용했다.

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.

표 3: Transformer 아키텍처의 변형. 명시되지 않은 값은 기본 모델의 값과 동일하다. 모든 지표는 영어-독일어 번역 개발 세트인 newstest2013에서 측정했다. 명시된 퍼플렉서티는 우리의 바이트 쌍 인코딩에 따른 워드피스당 값이며, 단어당 퍼플렉서티와 비교해서는 안 된다.

	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

In Table 3 rows (B), we observe that reducing the attention key size d_k hurts model quality. This
표 3의 행 (B)에서 어텐션 키 크기 dk를 줄이면 모델 품질이 저하되는 것을 관찰했다.
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,
이는 호환성을 결정하는 일이 쉽지 않으며, 내적보다 더 정교한 호환성 함수가 유용할 수 있음을 시사한다.
bigger models are better, and dropout is very helpful in avoiding over-fitting. In row (E) we replace our
또한 행 (C)와 (D)에서 예상한 대로 더 큰 모델이 더 우수하며, 드롭아웃이 과적합을 방지하는 데 매우 유용하다는 것을 관찰했다.
sinusoidal positional encoding with learned positional embeddings [8], and observe nearly identical results to the base model.

행 (E)에서는 사인파 위치 인코딩을 학습된 위치 임베딩 [8]으로 대체했으며, 기본 모델과 거의 동일한 결과를 관찰했다.

7 Conclusion

7 결론

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.

이 연구에서는 전적으로 어텐션에 기반한 최초의 시퀀스 변환 모델인 Transformer를 제시하고, 인코더-디코더 아키텍처에서 가장 일반적으로 사용되는 순환 레이어를 다중 헤드 자기 어텐션으로 대체했다.

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
번역 과제에서 Transformer는 순환 레이어 또는 합성곱 레이어에 기반한 아키텍처보다 훨씬 빠르게 학습할 수 있다.
English-to-French translation tasks, we achieve a new state of the art. In the former task our best
WMT 2014 영어-독일어 및 WMT 2014 영어-프랑스어 번역 과제 모두에서 새로운 최고 수준 성능을 달성했다.
model outperforms even all previously reported ensembles.

전자의 과제에서는 우리의 최상위 모델이 이전에 보고된 모든 앙상블보다도 우수한 성능을 보였다.

We are excited about the future of attention-based models and plan to apply them to other tasks. We
어텐션 기반 모델의 미래에 큰 기대를 걸고 있으며, 이를 다른 과제에도 적용할 계획이다.
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.

Transformer를 텍스트 이외의 입력 및 출력 양식이 관련된 문제로 확장하고, 이미지, 오디오, 비디오와 같은 대규모 입력과 출력을 효율적으로 처리하기 위한 국소적이고 제한된 어텐션 메커니즘을 연구할 계획이다. 생성을 덜 순차적으로 만드는 것도 우리의 또 다른 연구 목표 중 하나이다.

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

모델을 학습하고 평가하는 데 사용한 코드는 https://github.com/에서 제공한다.

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

감사의 글 Nal Kalchbrenner와 Stephan Gouws의 유익한 의견, 수정 및 영감에 감사드린다.

References

참고문헌

[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.

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

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

[10] 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.

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

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

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

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

[15] 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.

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

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

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

[19] 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.

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

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

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

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

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

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

[26] 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.

[27] 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.

[28] 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.

[29] 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.

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

[31] 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.

[32] 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.