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

適切な出典表示がなされた場合、Googleは本論文の表および図を、journalism 学術研究目的でのみの複製を許可する。

Attention はすべてである

Ashish Vaswani*
Google Brain
avasiani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

ヤコブ・ウスツコーライト*
グーグル・リサーチ
usz@google.com

リオン・ジョーンズ*
Google リサーチ
llion@google.com

エイドン・N・ゴメズ*
トロント大学
aidan@cs.toronto.edu

ウワシス・カイザー*
グーグル・ブレイン
lukaszkaiser@google.com

イリア・ポロスウキhin*
illia.polosukhin@gmail.com

要旨

支配的なシーケンス変換モデルは、エンコーダーとデコーダーを含む複雑な反復型または畳み込みニューラルネットワークに基づいている。最も性能の高いモデルは、エンコーダーとデコーダーをアテンションメカニズムを介して接続する。我々は、反復処理と畳み込みを完全に排除し、アテンションメカニズムのみに基づいた新しい単純なネットワークアーキテクチャである、Transformer を提案する。2つの機械翻訳タスクにおける実験により、これらのモデルが品質において優れており、かつ並列化が可能であり、訓練に要する時間が大幅に短縮されることが示された。我々のモデルはWMT 2014英語からドイツ語への翻訳タスクにおいて28.4のBLEUを達成し、アンサンブルを含む既存の最良結果を2 BLEU以上上回った。WMT 2014英語からフランス語への翻訳タスクでは、我々のモデルは文献中の最良モデルの訓練コストの僅かな部分で8台のGPUを用いて3.5日間訓練した後に、41.8という新しい単一モデルにおけるstate-of-the-art BLEUスコアを樹立した。我々は、Transformerが大きな訓練データおよび限られた訓練データの両方に対して英語の構成分解析に成功して適用されることにより、他のタスクにも良好に一般化することを示す。

*同等的寄与。記載順序はランダムである。ヤコブはRNNをセルフアテンションで置き換えることを提案し、このアイデアの評価に着手した。アシシュはイリアと共に、最初のトランスフォーマーモデルの設計と実装を行い、本作業のあらゆる側面において中心的な役割を果たしてきた。ノームはスケーリングされたドット積アテンション、マルチヘッドアテンション、パラメータフリーな位置表現を提案し、ほぼすべての詳細に関与するもう一人の人物となった。ニキは、多数のモデルバリエントを設計・実装・チューニング・評価し、我々のオリジナルコードベースおよびtensor2tensorに関与した。リオンも新たなモデルバリエントの実験を行ない、初期のコードベース、ならびに効率的な推論と可視化を担当した。ウカシとアイダンは、tensor2tensorの設計と実装に無数の長い日々を費やし、それまで使用していたコードベースを置き換え、結果を大幅に向上させ、研究を劇的に加速させた。

†Google Brain在籍時に行われた研究。

‡Google Research 在籍時に実施した研究。

第31回ニューラル情報処理システム会議（NIPS 2017）、アメリカ合衆国カリフォルニア州ロングビーチ

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 緒言

反復ニューラルネットワーク、特に長短期記憶 [13] およびゲート付き反復 [7] ニューラルネットワークは、言語モデルや機械翻訳のような系列モデリングおよびトランスダクション問題における最先端の手法として確立されている [35, 2, 5]。それ以来、反復型言語モデルおよびエンコーダー-デコーダーアーキテクチャの限界を拡大するための多数の試みが続けられている [38, 24, 15]。

再帰モデルは通常、入力および出力シーケンスの記号位置に沿って計算を因数分解する。これらの位置を計算時間のステップに対応させることにより、過去の隠れ状態 h_{t-1} と位置 t における入力の関数として、一連の隠れ状態 h_t を生成する。この本質的に順次的な性質は、 training examples 内の並列化を排除する。これは長鎖長において重要となり、メモリ制約がサンプル間のバッチングを制限するためである。最近の研究では、因数分解の工夫 [21] および条件付き計算 [32] を通じて計算効率を大幅に向上させ、後者の場合モデル性能も改善している。しかしながら、順次計算という根本的な制約は依然として残っている。

Attention 機構は、様々なタスクにおける説得力のある系列モデルおよび変換モデルの不可欠な構成要素となり、入力または出力系列におけるその距離を問わず依存関係をモデル化することを可能にした [2, 19]。しかしながら、わずかな例 [27] を除き、これらの Attention 機構は反復ネットワークと併用されて使用されている。

本稿では、再帰性を排し、代わりにAttentionメカニズムのみを用いて入力と出力の間のグローバルな依存関係を抽出するモデルアーキテクチャであるTransformerを提案する。Transformerは大幅な並列化を可能とし、8台のP100 GPU上でわずか12時間の学習結束后に、翻訳品質において新技術の最高水準に達することができる。

2 背景

逐次計算の削減という目的は、Extended Neural GPU [16]、ByteNet [18]、ConvS2S [9] の基礎をなしており、これらはいずれも基本構成要素として畳み込みニューラルネットワークを用い、すべての入力および出力位置について隠れ表現を並列に計算する。これらのモデルにおいて、2つの任意の入力または出力位置間の信号を関連づけるために必要な演算数は、位置間の距離とともに増加する。ConvS2Sでは線形に、ByteNetでは対数的に増加する。これは、遠隔位置間の依存関係を学習することを困難にする [12]。Transformerにおいては、これは定数の演算数にまで削減されるが、その代償として、attention重み付けされた位置の平均化により有効解像度が低下するという側面がある。この効果については、3.2節で記述するMulti-Head Attentionによって相殺される。

自己Attention（ sometimes called intra-attention ）とは、単一シーケンスの異なる位置間を関連付けて、そのシーケンスの表現を計算するためのAttentionメカニズムである。自己Attentionは、読解、要約、文脈推論、タスク非依存的文脈表現の学習といった多様なタスクにおいて成功裏に用いられてきた [4, 27, 28, 22]。

エンドツーエンド記憶ネットワークは、シーケンス整列型反復ではなく反復注意メカニズムに基づいており、単純な言語の質問応答および言語モデル化タスクにおいて良好な性能を示すことが示されている [34]。

しかしながら、著者の知る限り、Transformer は、シーケンス整列されたRNNや畳み込みを一切使用せず、入力と出力の表現を自己attentionのみを用いて計算する最初の伝達モデルである。以下の節では、Transformerについて記述し、自己attentionの動機づけを行い、[17, 18] および [9] などのモデルと比較した際のその利点について議論する。

3 モデルアーキテクチャ

最も競争力のあるニューラルシーケンス転送モデルは、エンコーダー-デコーダー構造を有する [5, 2, 35]。ここで、エンコーダーは記号表現の入力シーケンス $(x_1, \ldots, x_n)$ を連続的な表現 $\mathbf{z}=(z_1, \ldots, z_n)$ のシーケンスに写像する。$\mathbf{z}$ が与えられた後、デコーダーは記号の出力シーケンス $(y_1, \ldots, y_m)$ を要素ごとに順次生成する。各ステップにおいて、モデルは自己回帰的 [10] であり、次の記号を生成する際に、までに生成された記号を追加入力を消費する。

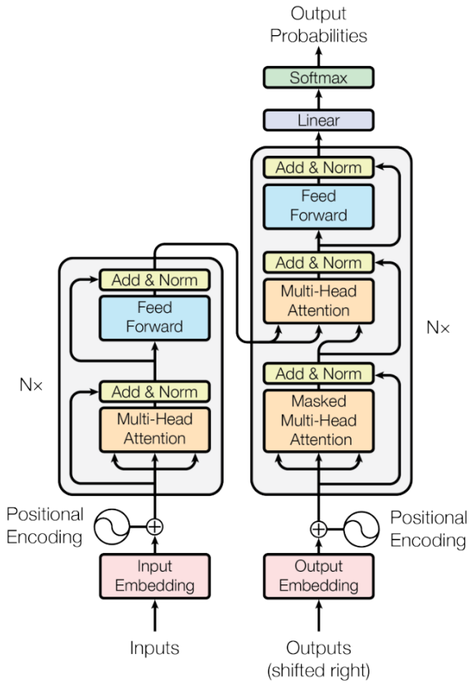


Figure 1: The Transformer - model architecture.

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

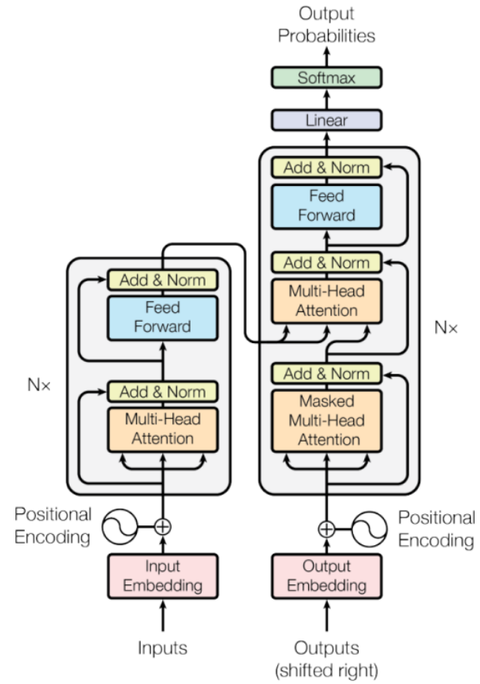


図1: Transformerモデルのアーキテクチャ。

Transformerは、この全体的なアーキテクチャを、エンコーダーとデコーダーの両方において、積層された自己アテンションおよび点別全結合層を用いて実装する。これらは、図1の左側および右側にそれぞれ示されている。

3.1 エンコーダおよびデコーダ・スタック

エンコーダー：エンコーダーは、$N=6$ の同一なレイヤーのスタックから構成される。各レイヤーには2つのサブレイヤーが存在する。1つ目はマルチヘッド・セルフ・アテンションメカニズムであり、2つ目は単純なポジションワイズ全結合フィードフォワードネットワークである。我々は、2つのサブレイヤーのそれぞれ周囲に残差接続 [11] を採用し、その後にレイヤ normalization [1] を行う。すなわち、各サブレイヤーの出力は $\mathrm{LayerNorm}(x+\mathrm{Sublayer}(x))$ であり、ここで $\mathrm{Sublayer}(x)$ はサブレイヤー自体が実装する関数を指す。これらの残差接続を容易にするため、モデル内のすべてのサブレイヤーおよび埋め込み層は、次元 $d_{\text{model}}=512$ の出力を生成する。

デコーダー：デコーダーもまた、$N=6$ の同一な層のスタックで構成される。各エンコーダー層に含まれる2つのサブレイヤーに加え、デコーダーは第3のサブレイヤーを挿入し、これはエンコーダースタックの出力に対してマルチヘッドアテンションを実行する。エンコーダーと同様に、我们は各サブレイヤーの周囲に残差接続を採用し、その後にレイヤー正規化を行う。また、デコーダースタック内のセルフアテンションサブレイヤーを修正し、ある位置が後の位置にアテンションすることを防ぐ。このマスキングと、出力埋め込みが1つ位置ずれているという事実は組み合わさり、位置 i に対する予測が、位置 i よりも小さい既知の出力のみ依頼于すること的保障する。

3.2 注意力

attention関数は、クエリと一連のキー・バリュー対を出力にマッピングすることとして記述できる。ここで、クエリ、キー、値、および出力はすべてベクトルである。出力は重み付け和として計算される。

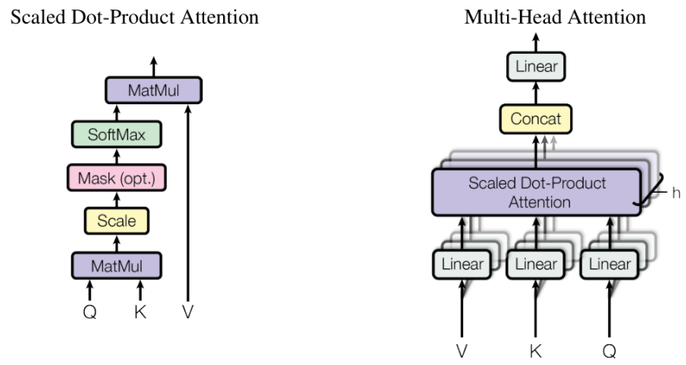


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

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:

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

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.

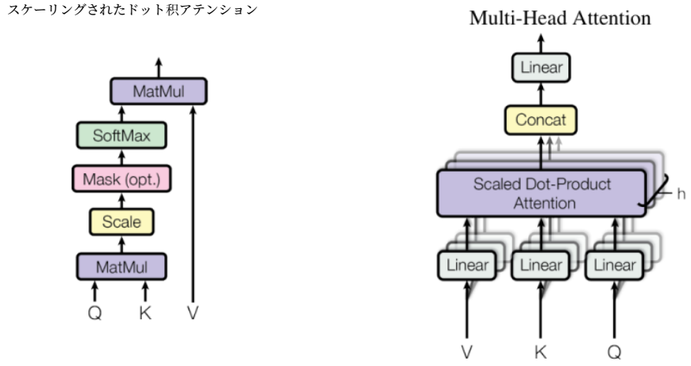


図2:（左）スケーリングされたドット積アテンション。（右）マルチヘッド・アテンションは、並列に動作する複数のアテンション層から構成される。

値の其中、各値に割り当てられた重みは、クエリと対応するキーとの互換性関数によって計算される。

3.2.1 スケールド・ドット・プロダクト・アテンション

我々の特筆すべき注意は「スケーラード・ドット・プロダクト・アテンション」（図2）である。入力は、次元 d_k のクエリおよびキー、および次元 d_v のバリューから構成される。クエリとすべてのキーとの間でのドット積を計算し、それぞれを $\sqrt{d_k}$ で除した後にソフトマックス関数を適用して、バリューに対する重みを得る。

実際には、一連のクエリに対してAttention関数を同時に計算し、それらを行列 Q に詰める。キーとバリューも同様に行列 K および V に詰める。出力の行列を次のように計算する：

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

最も一般的に用いられる2つのattention関数は、additive attention [2] および dot-product (multiplicative) attention である。dot-product attention は、スケーリング因子 $\frac{1}{\sqrt{d_k}}$ を除けば、本アルゴリズムと同一である。additive attention は、単一隠れ層を持つフィードフォワードネットワークを用いて互換性関数を計算する。理論的な複雑さは両者に類似しているものの、実際には dot-product attention の方がはるかに高速であり、空間効率も優れている。なぜなら、高度に最適化された行列乗算コードを用いて実装できるからである。

d_k の値が小さい場合、2つのメカニズムは類似した性能を示すが、 d_k が大きい場合、スケーリングを施さない加算型attentionは内積型attentionよりも優位に立つ [3]。我々は、 d_k が大きい場合、内積の絶対値が大きくなり、softmax関数を極めて勾配が小さい領域へ押し込むためではないかと推測する 4 。この影響を相殺するため、我々は内積を $\frac{1}{\sqrt{d_k}}$ でスケールする。

3.2.2 マルチヘッド・アテンション

単一のattention関数を用いて d_{model} 次元のkey、value、queryを実行する代わりに、query、key、valueをそれぞれ異なる学習済みの線形射影によって d_k, d_k および d_v 次元へ h 回線形に射影することが有効であることが判明した。これらの射影された各版本的なquery、key、valueに対しては、attention関数を並列に行うことにより、d_v 次元の

[4]内積の値が大きくなる理由を示すため、q と k の成分が平均 0、分散 1 の独立な確率変数であると仮定する。このとき、それらの内積 $q \cdot k = \sum_{i=1} q_i k_i$ は平均 0、分散 d_k を持つ。

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 $\sqrt{d_\text{model}}$.

出力値。これらは連結され、再度射影されることにより最終値が得られ、図2に示す通りである。

マルチヘッドAttentionは、モデルが異なる位置における複数の表現部分空間からの情報を統合的に注意深く処理することを可能にする。単一のAttentionヘッドのみを用いる場合、平均化はこれを阻害する。

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

ここで、投影行列は $W_i^Q \in \mathbb{R}^{d_\text{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_\text{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_\text{model} \times d_v}$ および $W^O \in \mathbb{R}^{hd_v \times d_\text{model}}$ である。

本稿では、$h=8$ の並列attention層、すなわちヘッドを採用する。各ヘッドに対しては、$d_k = d_v = d_\text{model}/h = 64$ を用いる。各ヘッドの次元が縮小されているため、全体的な計算コストは完全な次元を持つ単一ヘッドattentionと同等となる。

3.2.3 本モデルにおけるアテンションの応用

Transformer は、3つの異なる方法でマルチヘッドアテンションを使用する：

「エンコーダー-デコーダー・アテンション」層において、クエリは直前のデコーダー層から得られ、メモリー・キーおよびバリューはエンコーダーの出力から得られる。これにより、デコーダーの各位置が入力シーケンスのすべての位置にアテンションを行うことが可能となる。これは、[38, 2, 9]に示されるシーケンス・トゥ・シーケンス・モデルにおける典型的なエンコーダー-デコーダー・アテンション・メカニズムを模倣するものである。

エンコーダーは自己注意層を含む。自己注意層において、すべてのキー、バリュー、クエリは同一の場所から得られ、この場合、エンコーダー内の前層の出力である。エンコーダーの各位置は、エンコーダーの前層のすべての位置に注意力を向けることができる。

同様に、デコーダーの自己注意層は、各位置がその位置を含むデコーダー内のすべての位置に注視することを可能にする。自動回帰性を維持するため、デコーダー内での左方向の情報流れを防止する必要がある。この目的のために、スケーリングされたドット積Attentionの実装において、ソフトマックスの入力に対応する非法な接続の値すべてを$-\infty$に設定することでマスク処理を行う。図2を参照されたい。

3.3 位置ごとの順伝ネットワーク

attention サブレイヤーに加え、エンコーダーおよびデコーダーの各レイヤーには、全結合フィードフォワードネットワークが含まれており、これは各位置に対して個別かつ同一に適用される。このネットワークは、その間に ReLU 活性化関数を挟んだ2つの線形変換から構成される。

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

線形変換は異なる位置において同一であるが、層ごとに異なるパラメータを用いる。これを別の表現で言えば、カーネルサイズが1の2つの畳み込み操作に相当する。入力および出力の次元数は $d_\text{model}=512$ であり、内部層の次元数は $d_{ff}=2048$ である。

3.4 エンベッディングとソフトマックス

他のシーケンス変換モデルと同様に、我们は学習された埋め込みを用いて、入力トークンおよび出力トークンを次元 d_model のベクトルに変換する。また、デコーダーの出力を予測される次のトークンの確率に変換するために、通常の学習済み線形変換とsoftmax関数も使用する。当モデルでは、2つの埋め込み層とpre-softmax線形変換の間で同一の重み行列を共有する（[30]参照）。埋め込み層においては、それらの重みに $\sqrt{d_\text{model}}$ を乗算する。

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

表1: 異なるレイヤー種別における最大パス長、レイヤーごとの計算量、および最小順次操作数。n はシーケンス長、d は表現次元、k は畳み込みのカーネルサイズ、r は制限付き自己注意の近傍サイズである。

レイヤータイプ	レイヤーあたりの計算量	逐次処理型演算操作	最大パス長
自己注意機構	O(n² · d)	O(1)	O(1)
再帰型	O(n · d²)	O()	O(n)
畳み込み型	O(k · n · d²)	O(1)	O(log_k(n))
自己注意機構（制限付き）	O(r · n · d)	O(1)	O(n/r)

3.5 位置エンコーディング

我々のモデルは反復と畳み込みを含まないため、モデルがシーケンスの順序を利用するためには、シーケンス内のトークンの相対的または絶対的な位置に関する情報を何らかの形で付与しなければならない。この目的のため、エンコーダーおよびデコーダーのスタックの最下部にある入力埋め込みに「位置エンコーディング」を追加する。位置エンコーディングは埋め込みと同じ次元 d_{model} を持つため、両者は加算可能である。位置エンコーディングの選定方法は多様であり、学習型と固定型がある [9]。

本稿では、異なる周波数の正弦関数および余弦関数を用いる。

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

ここで pos は位置、i は次元を表す。すなわち、位置エンコーディングの各次元は正弦波に対応する。波長は 2π から $10000 \cdot 2\pi$ までの等比数列をなす。この関数を選択したのは、固定されたオフセット k に対して、PE_{pos+k} を PE_{pos} の線形関数として表せるため、モデルが相対位置による注意力の配分を容易に学習できると仮定したからである。

また、学習された位置エンコーディング [9] を使用する場合も実験したところ、両者の結果はほぼ同一であることが判明した（表3の行(E)を参照されたい）。我們選擇了正弦波形式のバージョンは、モデルが訓練中に遭遇したシーケンス長さよりも長い長さへの外挿を可能にする可能性があるためである。

4 自己注意メカニズムの必要性

本節では、記号表現の可変長シーケンス $(x_1, \ldots, x_n)$ を同じ長さの別のシーケンス $(z_1, \ldots, z_n)$ へマッピングするために一般的に用いられる再帰層および畳み込み層と、自己注意層の様々な側面を比較する。ここで $x_i, z_i \in \mathbb{R}^d$ であり、典型的なシーケンス転送エンコーダーまたはデコーダーの隠れ層などが該当する。自己注意の採用を正当化するため、我らは3つの望ましい特性を検討する。

1つ目は層ごとの総計算量である。もう1つ目は並列化可能な計算の量であり、それは必要な最小限の順次演算の数によって測定される。

3つ目は、ネットワーク内の長距離依存関係間のパス長さである。長距離依存関係の学習は、多くのシーケンス変換タスクにおける主要な課題である。このような依存関係を学習する能力に影響を与える主要な要因の一つは、順方向および逆方向信号がネットワーク内を通過しなければならないパスの長さである。入力および出力シーケンス内の任意の位置の組み合わせ間でのこれらのパスが短いほど、長距離依存関係を学習することは容易になる [12]。したがって、異なる層タイプで構成されるネットワークにおいて、任意の2つの入力および出力位置間の最大パス長さも比較する。

表1に示した通り、セルフ・アテンション層は順次実行される操作の数が一定であるためすべての位置を接続するが、反復層は $O(n)$ 個の順次操作を必要とする。計算複雑性の観点から、序列の長さが十分大きい場合、セルフ・アテンション層は反復層よりも高速である。

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:

長さ n は表現の次元 d より小さいことが多く、これは機械翻訳における最新モデルで用いられる文表現、すなわち word-piece [38] および byte-pair [31] 表現において最も一般的なケースである。非常に長いシーケンスを扱うタスクにおける計算性能を向上させるため、自己attentionは respective 出力位置を中心としたサイズ r の近傍のみを考慮するように制限され得る。これにより最大パス長は $O(n/r)$ に増加する。今後の研究においてこの手法をさらに調査する計画である。

カーネル幅 $k<n$ の単一畳み込み層では、入力と出力の位置のすべてのペアを接続できない。連続するカーネルの場合、これを行うには $O(n/k)$ 個の畳み込み層のスタックが必要であり、拡張畳み込みの場合は $O(\log_k(n))$ となる [18]。これは、ネットワーク内の任意の2つの位置間の最長パスの長さが増加することを意味する。畳み込み層は一般に、反復層に比べて k 倍だけ計算コストが高い。しかし、可分畳み込み [6] は複雑度を大幅に低減し、$O(k \cdot n \cdot d + n \cdot d^2)$ とする。さらに $k=n$ の場合でも、可分畳み込みの複雑度は、自己注意層とポイントワイズ前向き層を併用したものと等しく、これが我々のモデルで採用している手法である。

補助的な利点として、自己attentionはより解釈可能なモデルを生み出すことができる。我々はモデルからのattention分布を検証し、付録に例を提示して議論する。個々のattentionヘッドが明確に異なるタスクを実行することを学習するだけでなく、多くのヘッドが文の構文および意味構造に関連する振る舞いを示すように見える。

5 学習

本節では、当社のモデルに対する学習手法について記述する。

5.1 学習データとバッチ処理

私たちは、約450万の文対からなる標準的なWMT 2014英語-ドイツ語データセットを用いて学習を行った。文は byte-pair encoding [3] を用いてエンコードされ、これは約37,000トークンの共有されたソース-ターゲット・ボカビュラリを有している。英語-フランス語については、より大規模なWMT 2014英語-フランス語データセット（3600万文からなる）を使用し、トークンを32,000の word-piece ボカビュラリ [38] に分割した。文対は概ね同じ列長になるようバッチ化された。各学習バッチには、おおよそ25,000のソース・トークンおよび25,000のターゲット・トークンを含む文対のセットが含まれていた。

5.2 ハードウェアおよびスケジューリング

我々のモデルは、8 NVIDIA P100 GPU を搭載した1台の機械上で訓練された。本紙全体で記述したハイパーパラメータを用いたベースモデルについて、各訓練ステップは約0.4秒を要した。ベースモデルの総訓練ステップ数は100,000ステップ（12時間）であった。一方、大型モデル（表3の最終行に記述）については、ステップあたりの時間が1.0秒であった。大型モデルは300,000ステップ（3.5日）にわたって訓練された。

5.3 最適化アルゴリズム

我們使用 Adam 最適化器 [20]，其中 $\beta_1 = 0.9$、$\beta_2 = 0.98$、$\varepsilon = 10^{-9}$。根據下式，在訓練過程中變化學習率：

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

これは、最初の *warmup_steps* 分の学習ステップにおいて学習率を線形に増加させ、その後はステップ数の逆平方根に比例して減少させることに対応する。我們使用した *warmup_steps* = 4000 である。

5.4 正則化

我々は、学習中に3種類の正則化手法を採用する。

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.

表2：Transformerは、訓練コストの僅か几分の一で、英語からドイツ語および英語からフランス語へのnewstest2014テストにおいて、過去の最先進モデルよりも高いBLEUスコアを達成する。

モデル	BLEU		トレーニングコスト（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 アンサンブル [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL アンサンブル [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S アンサンブル [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer（ベースモデル）	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer（ビッグ）	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

残留ドロップアウト 各サブレイヤーの出力に対して、それがサブレイヤー入力に加算され正規化される前に、ドロップアウト [33] を適用する。さらに、エンコーダーおよびデコーダスタックの両方において、埋め込みと位置符号化の和に対してもドロップアウトを適用する。ベースモデルについては、$P_{drop}=0.1$ の率を使用する。

ラベルスムージング 学習中，値 $\varepsilon_{ls}=0.1$ のラベルスムージングを採用した [36]．これはモデルがより不確実となることを学ばせることでパプлексリティを損なうものの，精度および BLEU スコアを向上させる効果がある．

6 結果

6.1 機械翻訳

WMT 2014 英語からドイツ語への翻訳タスクにおいて、Table 2 に示す大規模トランスフォーマモデル (Transformer (big)) は、それまでに報告された最良のモデル（アンサンブルを含む）を 2.0 BLEU を超えて上回り、28.4 という新たな最高性能 BLEU スコアを達成した。このモデルの構成は Table 3 の最終行に示されている。 training には 8 台の P100 GPU を用いて 3.5 日要した。ベースモデルでさえも、競合モデルのいずれかの training コストの一部で、それまでに公開されたすべてのモデルやアンサンブルを上回っている。

WMT 2014 英語からフランス語への翻訳タスクにおいて、本大規模モデルは BLEU スコア 41.0 を達成し、 previously published single models（以前に公開された単一モデル）すべてを上回った。これは、 previous state-of-the-art model（以前の最先進モデル）の training cost（学習コスト）の 1/4 未満で達成されたものである。英語からフランス語への学習に用いた Transformer (big) モデルでは、 dropout rate P_{drop} を 0.3 ではなく 0.1 とした。

ベースモデルについては、10分間隔で作成された最後の5つのチェックポイントの平均値を用いた単一のモデルを使用した。大規模モデルについては、最後の20つのチェックポイントを平均化した。ビームサイズ4、長さペナルティ $\alpha=0.6$ [38] を用いたビームサーチ実施了。これらの超パラメータは開発セットでの実験の結果に基づいて選択された。推論時の最大出力長は入力長 + 50に設定したが、可能であれば早期終了を行う [38]。

表2は、本研究の結果を要約し、文献に報告されている他のモデルアーキテクチャとの比較において、翻訳品質およびトレーニングコストを示す。浮動小数点演算の総数は、トレーニング時間、使用されたGPUの数、各GPUの持続的な単精度浮動小数点処理能力の推定値を乗算することによって算出される 5。

6.2 モデルのバリエーション

Transformerの異なる構成要素の重要性を評価するために、基底モデルを多様な方法で変異させ、その結果における英語からドイツ語への翻訳タスクでの性能変化を測定した。

[5] K80, K40, M40, P100 に対しては、それぞれ2.8, 3.7, 6.0, 9.5 TFLOPSの値を用いた。

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

表3: Transformerアーキテクチャの変異。未記載の値はベースモデルの値と同一である。すべての指標は、英語からドイツ語への翻訳開発セットであるnewstest2013に基づいている。記載されたパプルゼクシティスは、当社のバイトペアエンコーディングに基づくワードピースあたりの値であり、単語あたりのパプルゼクシティスと比較してはならない。

	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)	正弦波の代わりに位置埋め込み									4.92	25.7	
big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

開発セットには newstest2013 を用いた。前節で述べた通りビームサーチを実施したが、チェックポイント平均化は行わなかった。これらの結果を表3に示す。

表3の行(A)では、計算量を一定に保ちながら（3.2.2節で記述されている通り）、attention headsの数およびattention keyとvalueの次元を変化させた。単一headのattentionは最良の設定に対して0.9 BLEU低いものの、heads数が多すぎても品質は低下する。

表3の行(B)において、attention keyのサイズ d_k を小さくするとモデルの品質が低下することが観察された。これは、互換性の判定が容易ではないことを示唆しており、内積よりも高度な互換性関数を用いることが有益である可能性がある。さらに、行(C)および(D)で確認できた通り、より大きなモデルほど性能が高く、dropoutは過学習の回避に極めて有効である。行(E)では、正弦波型位置エンコーディングに代わって学習可能な位置埋め込み[9]を適用したが、ベースモデルとほぼ同等の結果が得られた。

6.3 英語構成分解

Transformerが他のタスクにも適用可能か評価するため、英語構成分解解析の実験を行った。本タスクには特定の課題が存在する：出力は強い構造的制約を受け、入力よりも著しく長くなる。さらに、RNNのシーケンス・トゥ・シーケンスモデルは、小規模データ条件下において最良の結果を達成できていない [37]。

私たちは、Penn Treebank [25] の Wall Street Journal (WSJ) コーパス（約4万文の学習文）を用いて、モデル次元 d_{model}＝1024 の4層Transformerを学習させた。また、約1700万文を含む、より大規模で信頼性の高いBerkleyParserコーパスを使用した半教師あり設定でも学習を行った。WSJのみを対象とする設定では16Kトークンの語彙を、半教師あり設定では32Kトークンの語彙を使用した。

ドロップアウト、アテンションおよびリジューアル（5.4節）、学習率、ビームサイズについては、22節の開発セットにおいて少数の実験を行って選定したのみであり、他のすべてのパラメータは英語からドイツ語へのベース翻訳モデルと同一に保った。推論中、

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.

表4: Transformerは英語の構成分解析に良好に一般化する（結果はWSJセクション23より）

パーサー	トレーニング	WSJ 23 F1値
Vinyals & Kaiser他（2014）[37]	WSJのみ、判別モデル	88.3
Petrov他（2006）[29]	WSJのみ、判別モデル	90.4
Zhu他（2013）[40]	WSJのみ、判別モデル	90.4
Dyer他（2016）[8]	WSJのみ、判別モデル	91.7
Transformer（4層）	WSJのみ、判別モデル	91.3
Zhu他（2013）[40]	半教師あり学習	91.3
Huang & Harper（2009）[14]	半教師あり学習	91.3
McClosky他（2006）[26]	半教師あり学習	92.1
Vinyals & Kaiser他（2014）[37]	半教師あり学習	92.1
Transformer（4層）	半教師あり学習	92.7
Luong他（2015）[23]	マルチタスク学習	93.0
Dyer他（2016）[8]	生成モデル	93.3

最大出力長を入力長＋300に増加させた。WSJのみおよび半教師付き設定の両方において、ビームサイズを21、αを0.3とした。

表4に示す結果によれば、タスク固有のチューニングを行っていないにもかかわらず、本モデルは驚くほど高い性能を示し、再帰ニューラルネットワーク文法 [8] を除くすべての従来報告モデルよりも優れた結果を導き出した。

RNNのシーケンス・トゥ・シーケンスモデル [37] と対照的に、Transformer は WSJ 訓練セットの 40K 文のみで学習した場合でも、Berkeley-Parser [29] を凌駕する。

7 結論

本研究では、完全にアテンションに基づく最初のシーケンス変換モデルであるTransformerを紹介した。これは、エンコーダー-デコーダー構造で最も一般的に用いられている再帰層を、マルチヘッド・セルフ・アテンションに置き換えるものである。

翻訳タスクにおいて、Transformer は反復層または畳み込み層に基づくアーキテクチャに比べて著しく高速に訓練できる。WMT 2014 英独翻訳および WMT 2014 英仏翻訳の両方のタスクにおいて、当手法は新的な最高性能を達成した。前者のタスクにおいて、当グループが提案する最良モデルは、それまでに報告されたすべてのアンサンブルモデルをも凌駕する性能を示した。

私たちは、attentionに基づくモデルの将来に期待を寄せ、他のタスクへの適用を計画している。Transformerを、テキスト以外の入力および出力モダリティを伴う問題へ拡張し、画像、音声、動画のような大規模な入出力を効率的に処理するための局所的かつ制約付きのattentionメカニズムを検証することを企図している。生成プロセスをより非順序的（非逐次的）とすることは、私たちの別の研究目標である。

モデルの学習および評価に使用したコードは、https://github.com/tensorflow/tensor2tensor で入手可能である。

謝辞 Nal Kalchbrenner および Stephan Gouws の有益なご指摘、訂正、および刺激に深く感謝する。

参考文献

[1] Jimmy Lei Ba, Jamie Ryan Kiros, および Geoffrey E Hinton. レイヤー正規化. arXivプレプリント, arXiv:1607.06450, 2016.

[2] Dzmitry Bahdanau, Kyunghyun Cho,および Yoshua Bengio. 対斉と翻訳を統合的に学習するによるニューラル機械翻訳. CoRR, abs/1409.0473, 2014.

[3] Denny Britz, Anna Goldie, Minh-Thang Luong, および Quoc V. Le. ニュラル機械翻訳アーキテクチャの大規模探索. CoRR, abs/1703.03906, 2017.

[4] Jianpeng Cheng, Li Dong, および Mirella Lapata. マシン・リーディングのための長短期メモリー・ネットワークs. 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] 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, Yoshua Bengio. 統計的機械翻訳におけるRNNエンコーダー-デコーダーを用いたフレーズ表現の学習. CoRR, abs/1406.1078, 2014.

[6] フランソワ・ショレ. Xception: 深層学習におけるディプスワイズ・セパラブル・コンボリューション. arXiv preprint arXiv:1610.02357, 2016.

[7] Junyoung Chung, Ç¸aglar G¨ulc¸ehre, Kyunghyun Cho, および Yoshua Bengio. シーケンスモデルにおけるゲート付き再帰ニューラルネットワークの実証的評価. CoRR, abs/1412.3555, 2014.

[8] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, および Noah A. Smith. 再帰型ニューラルネットワーク文法. In Proc. of NAACL, 2016.

[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, Yann N. Dauphin. 畳み込み列から列への学習. arXiv preprint arXiv:1705.03122v2, 2017.

[10] Alex Graves. リカレントニューラルネットワークによる系列生成. arXivプレプリント arXiv:1308.0850, 2013.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, および Jian Sun. 画像認識のためのディープ残差学習.『IEEE計算机视觉およびパターンRecognition会議報告』, 770-778頁, 2016.

[12] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, および Jürgen Schmidhuber. 再帰ネットにおける勾配フロー：長期依存性の習得の困難さ, 2001.

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

[14] Zhongqiang Huang および Mary Harper. 言語間潜在アノテーションを有する自己学習 PCFG 文法. In 2009年自然言語処理の経験的手法に関する会議論文集, 832－841頁. ACL, 2009年8月.

[15] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, および Yonghui Wu. 言語モデル化の限界の探査. arXivプレプリント arXiv:1602.02410, 2016.

[16] Łukasz Kaiser and Samy Bengio. アクティブ・メモリーはアテンションを置き換えるか? In《Neural Information Processing Systems (NIPS) における進歩》, 2016.

[17] Łukasz Kaiser および 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, および Koray Kavukcuoglu. 線形時間で実行するニューラル機械翻訳. arXiv preprint arXiv:1610.10099v2, 2017.

[19] Yoon Kim, Carl Denton, Luong Hoang, および Alexander M. Rush. 構造化アテンションネットワーク. In International Conference on Learning Representations, 2017.

[20] Diederik Kingma および Jimmy Ba. Adam: A method for stochastic optimization. ICLR, 2015.

[21] オレクシイ・クチャイエフとボリス・ギンズブルグ。LSTMネットワークのための因数分解の技術。arXivプレプリント, arXiv:1703.10722, 2017年。

[22] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, Yoshua Bengio. 構造化自己注意文埋め込み. arXivプレプリント arXiv:1703.03130, 2017.

[23] ミン・タン・ルオン、クオック・V.・レ、イリヤ・スツスケバー、オリ奥尔・ヴィンヤルズ、ルカシュ・カイザー。マルチタスクシーケンス・トゥ・シーケンス・ラーニング。arXivプレプリント arXiv:1511.06114, 2015.

[24] Minh-Thang Luong, Hieu Pham, および Christopher D Manning. アテンションベースの機械翻訳への効果的なアプローチ. arXiv preprint 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] ミッチェル P マーカス, メアリー・アン マルチンキエヴィチ, および ビアトリス・サン托リーニ. エングリッシュの大型注釈付きコーパスの構築：ペントリーバンク. 計算言語学, 19(2):313 – 330, 1993.

[26] デビッド・マクロスキー、ユージェーン・チャルニアク、マーク・ジョンソン。構文解析のための効果的な自己学習法。『NAACL人間言語技術会議議事録：本会議』、152 – 159頁。ACL、2006年6月。

[27] Ankur Parikh, Oscar Täckström, Dipanjan Das, Jakob Uszkoreit. 分解可能なattentionモデル. In Empirical Methods in Natural Language Processing, 2016.

[28] Romain Paulus, Caiming Xiong, および Richard Socher. 要約生成のための深層強化学習モデル. arXiv preprint arXiv:1705.04304, 2017.

[29] Slav Petrov, Leon Barrett, Romain Thibaux, および Dan Klein. 正確でコンパクト、かつ解釈可能な木アノテーションの学習. In 第21回国際計算言語学会議および第44回ACL年次総会議事録, pages 433 – 440. ACL, July 2006.

[30] Ofir Press および Lior Wolf. 出力埋め込みを用いた言語モデルの改善. arXiv プレプリント arXiv:1608.05859, 2016.

[31] リコ・センリッチ、バリー・ハドゥ、アレクサンドラ・バーチ。サブワード単位を用いた希少語のニューラル機械翻訳。arXivプレプリント arXiv:1508.07909, 2015.

[32] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, および Jeff Dean. 激しく巨大なニューラルネットワーク：スパース・ゲートド・マixture・オブ・エキスパート層。arXivプレプリント arXiv:1701.06538, 2017.

[33] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, および 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, および Rob Fergus. エンド・トゥ・エンド・メモリー・ネットワークス. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, および R. Garnett, 編, Neural Information Processing Systems 28 における進歩, pages 2440 – 2448. Curran Associates, Inc., 2015.

[35] Ilya Sutskever, Oriol Vinyals, および Quoc VV Le. ニューラルネットワークを用いたシーケンスからシーケンスへの学習. Neural Information Processing Systems における進歩（ Advances in Neural Information Processing Systems ）, pages 3104 – 3112, 2014.

[36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, および Zbigniew Wojna. コンピュータビジョンのためのインセプション・アーキテクチャの再考. CoRR, abs/1512.00567, 2015.

[37] Vinyals & Kaiser, Koo, Petrov, Sutskever, および 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, 他. グーグルのニューラル機械翻訳システム：人間と機械翻訳のギャップを埋めること. arXivプレプリント, arXiv:1609.08144, 2016.

[39] Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, および Wei Xu. 機械翻訳のための高速順伝接続を有する深層再帰モデル. CoRR, abs/1606.04199, 2016.

[40] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, および Jingbo Zhu. 高速かつ高精度なシフト・リダクション成分解析. In ACL 2013第51回年次会議報告書（ 第1巻：長文 ）, ページ 434 – 443. ACL, 2013年8月.

Attention Visualizations

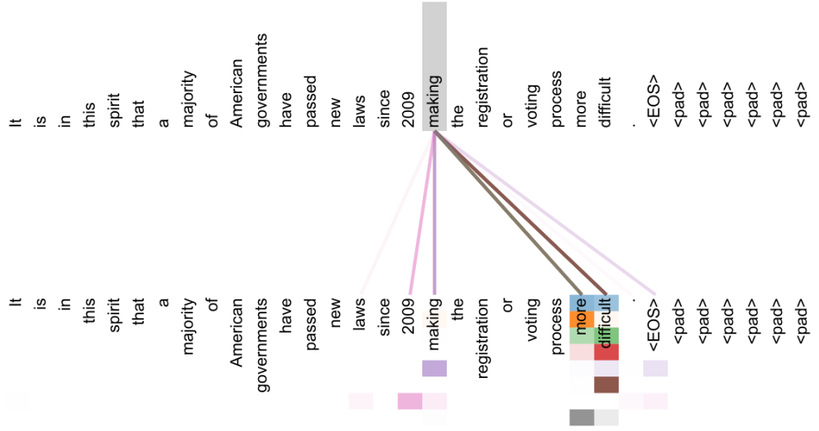


Figure 3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb ‘making’, completing the phrase ‘making...more difficult’. Attentions here shown only for the word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

アテンション可視化

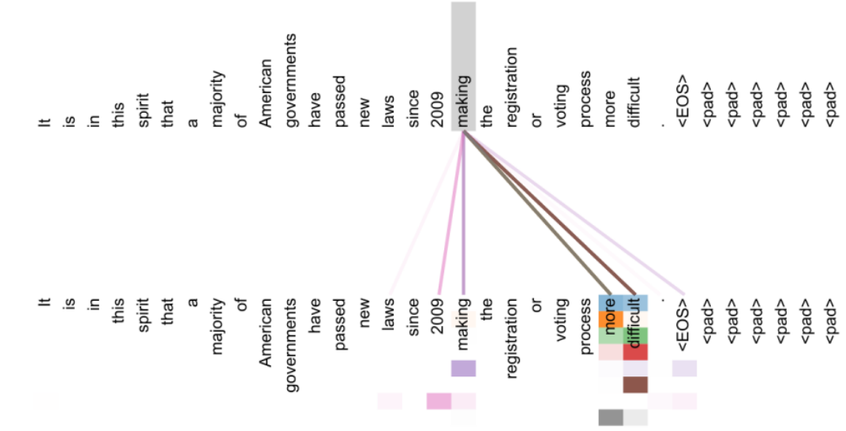


図3: エンコーダー自己注意メカニズムにおける長距離依存関係の例（6層中5層）。多くのattention headは動詞 'making' の遠隔依存関係に注視し、フレーズ 'making...more difficult' を完了させる。ここに示されたattentionは単語 'making' についてのみである。異なる色は異なるheadを表す。カラー版で視聴することをお勧めする。

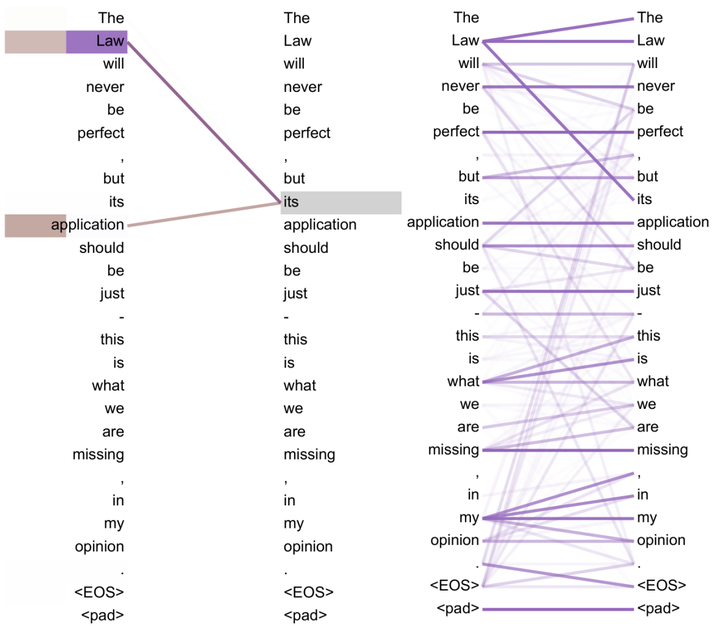


Figure 4: Two attention heads, also in layer 5 of 6, apparently involved in anaphora resolution. Top: Full attentions for head 5. Bottom: Isolated attentions from just the word 'its' for attention heads 5 and 6. Note that the attentions are very sharp for this word.

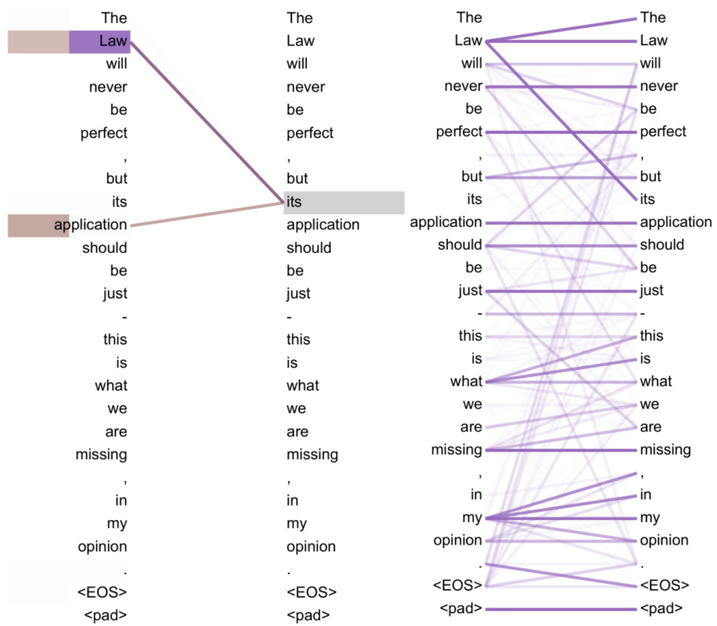


図4: 2つのattentionヘッド。層5と6の両方に関与しているようである。上段: ヘッド5における全attention。下段: 単語「its」からのみ生じるisolated attention。ヘッド5および6に注目されたい。この語に対するattentionが非常に鋭敏であることがわかる。

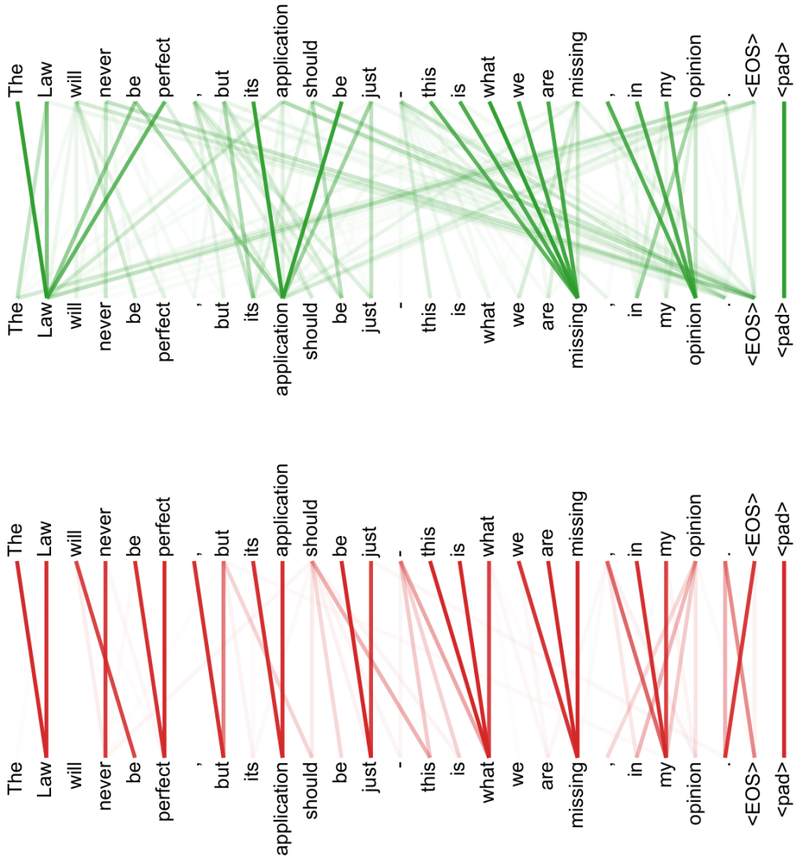


Figure 5: Many of the attention heads exhibit behaviour that seems related to the structure of the sentence. We give two such examples above, from two different heads from the encoder self-attention at layer 5 of 6. The heads clearly learned to perform different tasks.

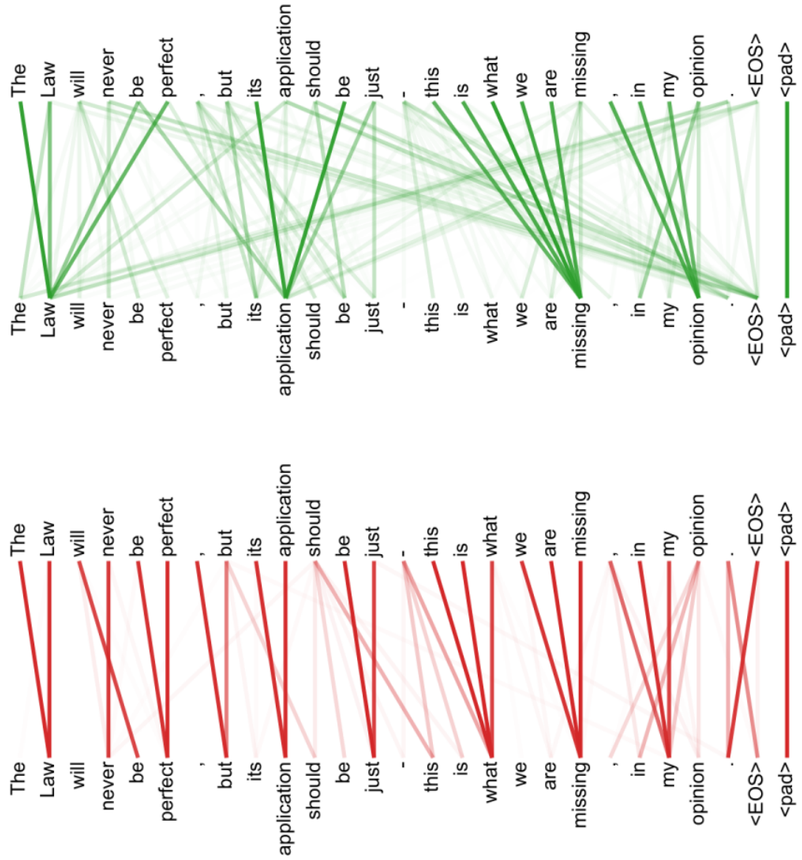


図5: 多くのアテンションヘッドは、文の構造に関連する振る舞いを示す。上記にはエンコーダー自己アテンションの6層中の5層における2つの異なるヘッドからの例を2つ示す。これらのヘッドは明確に異なるタスクを学習したことがわかる。