

適切な帰属表示が提供されることを条件に、Googleはここに、ジャーナリスティックまたは学術的な著作物での使用に限り、本論文の表および図の複製を許可する。

必要なのは注意力

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* † ‡
ロント大学
aidan@cs.toronto.edu

ウカシュ・カイザー*
グーグルブレイン
lukaszkaizer@google.com

イリア・ポロスーヒン* ‡
illia.polosukhin@gmail.com

要旨

優勢な配列伝達モデルは、エンコーダーとデコーダーを含む複雑なリカレント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。また、最も性能の良いモデルは、注意メカニズムを介してエンコーダーとデコーダーを接続している。我々は、リカレントや畳み込みを完全に排除し、注意メカニズムのみに基づく新しいシンプルなネットワークアーキテクチャ、トランスフォーマーを提案する。2つの機械翻訳タスクで実験を行った結果、これらのモデルは並列化可能で学習時間が大幅に短縮される一方で、品質が優れていることが示された。我々のモデルはWMT 2014英独翻訳タスクで28.4 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英仏翻訳タスクでは、我々のモデルは8GPUで3.5日間学習した後、41.8という新たな単一モデルの最新BLEUスコアを確立した。我々は、Transformerが他のタスクにうまく一般化することを、大規模な学習データと限られた学習データの両方で英語の小選挙区構文解析にうまく適用することで示す。

*平等な貢献。リスト順はランダム。JakobはRNNを自己アテンションに置き換えることを提案

し、このアイデアを評価する取り組みを開始した。AshishはIlliaと共に、最初のTransformerモデルを設計し、実装した。

は、この研究のあらゆる面で重要な役割を担ってきた。Noamはスケーリングされたドット積注意、多頭注意、パラメータフリーの位置表現を提案し、ほぼすべての細部に関与するもう一人の人物となった。ニキは、私たちのオリジナルのコードベースとtensor2tensorで、数え切れないほどのモデルのバリエーションを設計し、実装し、調整し、評価した。Llionもまた、新しいモデルのバリエーションを実験し、私たちの初期のコードベース、効率的な推論と可視化を担当しました。LukaszとAidanは、tensor2tensorの様々な部分の設計と実装に数え切れないほどの長い日々を費やし、私たちの初期のコードベースを置き換え、結果を大幅に改善し、私たちの研究を大幅に加速させました。

†グーグル・ブレイン在籍時の仕事。

‡グーグルリサーチ在籍時の仕事。

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

1 はじめに

リカレントニューラルネットワーク、特に長期短期記憶型[13]およびゲート型リカレント[7]ニューラルネットワークは、言語モデリングや機械翻訳などのシーケンスモデリングや伝達問題における最先端のアプローチとして確固たる地位を築いてきた[35, 2, 5]。それ以来、リカレント言語モデルやエンコーダ・デコーダ・アーキテクチャの限界を押し広げようとする数多くの努力が続けられている[38, 24, 15]。

リカレント・モデルは通常、入出力シーケンスのシンボル位置に沿って計算を行う。¹この本質的に逐次的な性質は、学習例内での並列化を妨げ、メモリ制約が例間でのバッチングを制限するため、シーケンス長が長くなると重要になる。最近の研究では、因数分解のトリック[21]や条件付き計算[32]によって計算効率の大幅な改善が達成されており、後者の場合はモデルの性能も向上している。しかし、逐次計算の基本的な制約は残っています。

注意メカニズムは、様々なタスクにおいて、説得力のあるシーケンスモデリングや変換モデルの不可欠な一部となっており、入力シーケンスや出力シーケンス内の距離に関係なく依存関係をモデル化することを可能にしている[2, 19]。しかし、一部のケース[27]を除いて、このような注意メカニズムはリカレントネットワークと組み合わせて使用される。

Transformerは、入力と出力の間の大域的な依存関係を描くために、再帰を避け、代わりに注意メカニズムに完全に依存するモデルアーキテクチャである。Transformerは、大幅に並列化することを可能にし、8台のP100 GPUで12時間という短い学習時間で、翻訳品質の新たな技術水準に到達することができる。

2 背景

逐次的な計算を減らすという目標は、Extended Neural GPU [16]、ByteNet [18]、ConvS2S [9]の基礎にもなっている。これらのモデルでは、2つの任意の入力または出力位置からの信号を関連付けるために必要な演算の数は、位置間の距離に応じて増加し、ConvS2Sでは線形に、ByteNetでは対数に増加する。このため、離れた位置間の依存関係を学習することが難しくなる[12]。Transformerでは、アテンションで重み付けされた位置の平均化によって有効解像度が低下する代償はあるものの、これは一定の演算回数に削減される。

自己注意は、時にはイントラ注意と呼ばれることもあるが、シーケンスの表現を計算するために、1つのシーケンスの異なる位置を関連付ける注意メカニズムである。自己注意は、読解、抽象的要約、テキストの含意、タスクに依存しない文表現の学習など、様々なタスクでうまく利用されている[4, 27, 28, 22]。

エンド・ツー・エンドの記憶ネットワークは、配列に沿った再帰ではなく、再帰的注意メカニズムに基づいており、単純な言語による質問応答や言語モデリング・タスクで優れた性能を発揮することが示されている[34]。

しかし、我々の知る限り、Transformerは、配列整列RNNや畳み込みを使用せずに、入力と出力の表現を計算するために完全に自己注意に依存する最初のトランスダクションモデルである。以下のセクションでは、Transformerについて説明し、自己注意を動機付け、[17, 18]や[9]のようなモデルに対する優位性について議論する。

3 モデル建築

ほとんどの競合的なニューラル・シーケンス伝達モデルは、エンコーダー・デコーダー構造を持つ[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, \dots, x_n)を連続表現のシーケンス $\mathbf{z} = (z_1, \dots, z_n)$ にマッピングする。 \mathbf{z} が与えられると、デコーダはシンボルの出力シーケンス (y_1, \dots, y_m) を1要素ずつ生成する。各ステップにおいて、モデルは自動回帰[10]し、次のシンボルを生成する際に、以前に生成されたシンボルを追加入力として消費する。

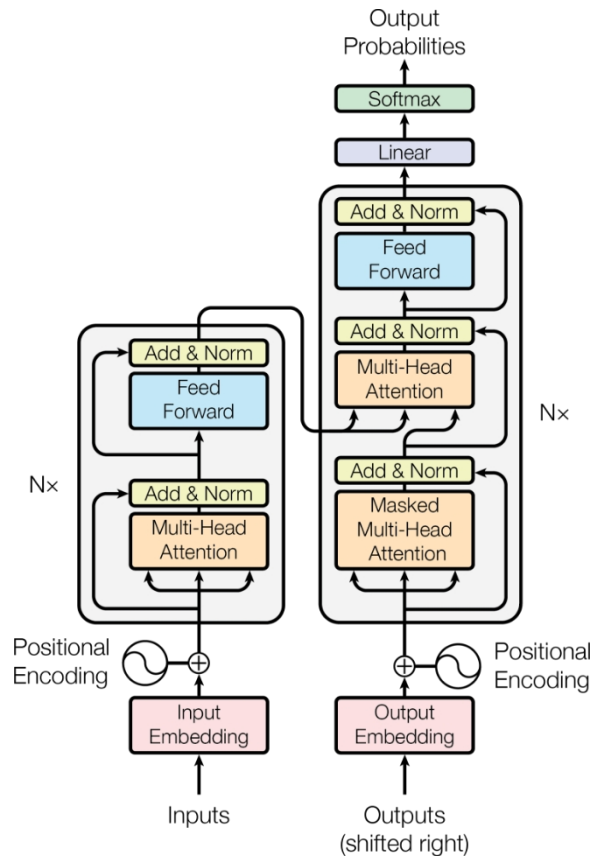


図1: トランスフォーマー - モデル・アーキテクチャ。

Transformerは、図1の左半分と右半分にそれぞれ示されているように、エンコーダーとデコーダーの両方に、スタックド・セルフ・アテンションとポイント・ワイズ・フル・コネクテッド・レイヤーを使用し、この全体的なアーキテクチャを踏襲している。

3.1 エンコーダーとデコーダーのスタック

エンコーダー： エンコーダーは $N=6$ 個の同じレイヤーのスタックで構成される。各レイヤーには2つのサブレイヤーがある。1つ目はマルチヘッド自己アテンションメカニズムであり、2つ目は単純な位置ワイズ完全連結フィードフォワードネットワークである。このサブレイヤーそれぞれに残差接続 [11]を採用し、レイヤーの正規化 [1]を行う。つまり、各サブレイヤーの出力は $\text{LayerNorm}(x + \text{Sublayer}(x))$ であり、 $\text{Sublayer}(x)$ はサブレイヤー自身が実装する関数である。これらの残差接続を容易にするために、埋め込み層と同様にモデル内のすべてのサブ層は次元 $d_{\text{model}} = 512$ の出力を生成する。

デコーダー： デコーダーも $N=6$ 個の同じレイヤーのスタックで構成される。各エンコーダー層の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、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行う。エンコーダーと同

様に、各サブレイヤーの周囲に残差接続を採用し、レイヤーの正規化を行う。また、デコーダスタックの自己アテンションサブレイヤーを変更し、ポジションが後続のポジションにアテンションしないようにする。このマスキングは、出力の埋め込みが1位置オフセットされることと組み合わせられ、位置 i の予測は i より小さい位置の既知の出力にのみ依存することを保証する。

3.2 注意

アテンション関数は、クエリとキーと値のペアのセットを出力にマッピングするものとして記述することができ、クエリ、キー、値、出力はすべてベクトルである。出力は重み付き和として計算される。

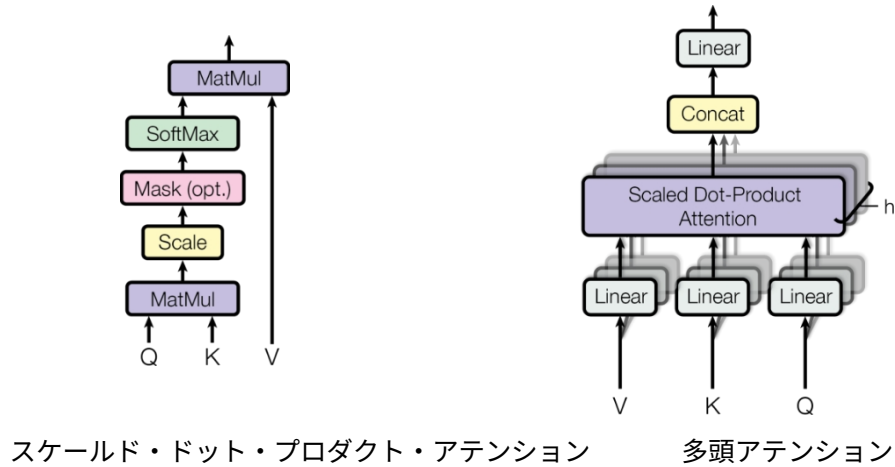


図2: (左)スケールドット・プロダクト・アテンション。(右) マルチヘッド注意は、並行して実行される複数の注意層で構成される。

各値に割り当てられる重みは、対応するキーとクエリ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

私たちはこの特殊な注意を「スケールド・ドット・プロダクト・アテンション」と呼んでいる(図2)。入力はクエリーと次元 d_k のキー、および次元 d_v の値のドット積を計算する。クエリにすべてのキーを指定 d_k 、ソフトマックス関数を適用して、以下の重みを求め、それぞれを値で割る。

キーと値は行列KとVにまとめられる。出力の行列は次のように計算される：

$$\text{注意}(Q, K, V) = \frac{QK^T}{\sqrt{d_k}} \text{ソフトマックス} V \quad (1)$$

最も一般的に使用される2つの注意関数は、加法的注意[2]とドット積(multi-plicative)注意である。ドット積注意は、 $\sqrt{d_k}$ のスケーリングファクターを除けば、我々のアルゴリズムと同じである。加法的注意は、以下のようなフィード・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。

単一の隠れ層。この2つは理論的な複雑さでは似ているが、ドット積注意は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ため、実際にはより高速でスペース効率も高い。

d_k の値が小さい場合、この2つのメカニズムは同じような性能を示すが、 d_k [3]の値が大きくなると、加法的注意はスケーリングすることなくドット積注意を上回る。我々は、

d_k の値が大きいと、ドット積の大きさが大きくなり、ソフトマックス関数を勾配が極端に小さくなる領域に押しやるのではないかと考えている。⁴ この効果を打ち消すために、ドット積を $\sqrt{d_k}$ だけスケールリングする。

3.2.2 マルチヘッド・アテンション

d_{model} -次元のキー、値、およびクエリーを用いて単一の注意関数を実行する代わりに、クエリー、キー、および値を、それぞれ d_k 、 d_k 、および d_v の次元に、異なる学習された線形投影を用いて h 回線形投影することが有益であることがわかった。^v これらの投影されたクエリー、キー、値のそれぞれに対して、注意関数を並列に実行する。

⁴ ドット積が大きくなる理由を説明するために、 q と k の成分が独立したランダムであると仮定する。変数は平均 0、分散 1 である。そして、それらのドット積、 $q \cdot k = \sum_{i=1}^{d_k} q_i k_i$ は、平均 0 と分散 d_k を持つ。

出力値。これらを連結し、もう一度投影すると、図2に示すような最終的な値が得られる。

マルチヘッド注意は、モデルが異なる位置にある異なる表現部分空間からの情報に共同して注意することを可能にする。単一の注意ヘッドでは、平均化がこれを阻害する。

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) W^O$$

ここでヘッド $_i = \text{Attention}(Q W_i^Q, K W_i^K, V W_i^V)$

ここで、投影はパラメータ行列 $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$
and $W^O \in \mathbb{R}^{h d_v \times d_{\text{model}}}$.

この作品では、 $h=8$ の並列注意層（ヘッド）を採用している。これらのそれぞれについて、 $d_k = d_v = d_{\text{model}} / h = 64$ を使用する。各ヘッドの次元が小さくなるため、総計算コストは全次元の単一ヘッド注意のそれと同程度になる。

3.2.3 我々のモデルにおける注意の応用

トランスフォーマーは、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・デコーダ注目」層では、クエリは前のデコーダ層から、メモリーキーと値はエンコーダの出力から来る。これにより、デコーダーの各ポジションは入力シーケンスの全ポジションに注意を向けることができる。これは[38, 2, 9]のようなsequence-to-sequenceモデルにおける典型的なエンコーダとデコーダの注意メカニズムを模倣している。
- ・エンコーダーには自己アテンション層がある。自己アテンション層では、すべてのキー、値、クエリーは同じ場所、この場合はエンコーダーの前の層の出力から来る。エンコーダーの各ポジションは、エンコーダーの前のレイヤーのすべてのポジションにアテンションすることができます。
- ・同様に、デコーダーの自己アテンション・レイヤーは、デコーダーの各ポジションが、そのポジションまでのデコーダーの全てのポジションにアテンションすることを可能にする。自己回帰特性を維持するためには、デコーダ内の左向きの情報フローを防ぐ必要がある。ソフトマックスの入力のうち、不正な接続に対応するすべての値をマスキングする（に設定する）ことで、スケールド・ドット・プロダクト・アテンションの内部でこれを実装する。図2を参照。

3.3 位置ワイズフィードフォワードネットワーク

アテンション・サブレイヤに加えて、我々のエンコーダーとデコーダーの各レイヤには、各ポジションに別々に同じように適用される完全連結フィードフォワードネットワークが含まれている。これは2つの線形変換とその間のReLU活性化で構成される。

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

線形変換は異なる位置で同じであるが、レイヤーごとに異なるパラメーターを使用する。別の言い方をすれば、カーネル・サイズ1の2つの畳み込みである。入力と出力の次元は $d_{\text{model}} = 512$ であり、内層の次元は $d_{\text{ff}} = 2048$ である。

3.4 埋め込みとソフトマックス

他のシーケンス変換モデルと同様に、学習された埋め込みを用いて、入力トークンと出力トークンを次元 d_{model} のベクトルに変換する。また、通常の学習された線形変換とソフトマックス関数を用いて、デコーダ出力を予測されるネクストトークン確率に変換する。次の

このモデルでは、2つの埋め込み層で同じウェイト行列を共有し、 $\sqrt{\text{softmax}}$ の前処理を行う。^{30]}と同様に線形変換を行う。埋め込み層では、これらの重みを d_{model} 。

表1: 異なるレイヤタイプの最大パス長、レイヤごとの複雑さ、最小逐次演算数。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における近傍のサイズ。

レイヤータイプ レイヤごとの長		複雑さ	シーケンシャル最大パス長
		オペレーション	
セルフ・アテンションズ	$O(n^2 - d)$	$O(1)$	$O(1)$
リカレント	$O(n - d)^2$	$O(n)$	$O(n)$
畳み込み	$O(k - n - d)^2$	$O(1)$	$O(\log_k(n))$
セルフ・アテンションズ (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

このモデルには再帰も畳み込みも含まれていないので、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注入しなければならない。そのために、エンコーダスタックとデコーダスタックの一番下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エンベディングと同じ次元 d_{model} を持つので、両者を合計することができる。位置エンコーディングには多くの選択肢があり、学習されたものと固定されたものがあります[9]。

この作品では、異なる周波数のサイン関数とコサイン関数を使用する：

$$PE_{(pos, 2i)} = \sin(pos/10000^{2i/d_{\text{モデル}}})$$

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

ここでposは位置、iは次元である。つまり、位置エンコードの各次元は正弦波に対応する。波長は 2π から $10000 \cdot 2\pi$ までの幾何学的な進行を形成する。私たちがこの関数を選んだのは、この関数を使えば、モデルが相対的な位置によるアテンダントを容易に学習できると考えたからである。というのも、任意の固定オフセット k に対して、 PE_{pos+k} は PE_{pos} の一次関数として表現できるからである。

また、代わりに学習された位置埋め込み[9]を使って実験したところ、2つのバージョンはほぼ同じ結果をもたらすことがわかった（表3の行(E)参照）。正弦波バージョンを選択した理由は、トレーニング中に遭遇したものより長いシーケンス長までモデルを外挿できる可能性があるからである。

4 なぜセルフ・アテンションなのか

このセクションでは、典型的なシーケンス・トランスダクション・エンコーダやデコーダの隠れ層のように、シンボル表現の可変長シーケンス (x_1, \dots, x_n) を同じ長さの別

のシーケンス (z_1, \dots, z_n) 、 x_i 、 z_i 、 R^d) にマッピングするために一般的に使われるリカレント層や畳み込み層と、自己注意層の様々な側面を比較する。自己注意を使用する動機として、我々は3つの望みを考える。

ひとつは、レイヤーごとの総計算量である。もうひとつは、並列化可能な計算量であり、必要な逐次演算の最小数で測られる。

3つ目は、ネットワーク内の長距離依存関係間のパスの長さである。長距離依存関係を学習することは、多くのシーケンス伝達タスクにおける重要な課題である。このような依存関係を学習する能力に影響を与える重要な要因の1つは、前方信号と後方信号がネットワーク内で通過しなければならないパスの長さである。入力配列と出力配列のどの位置の組み合わせでも、これらのパスが短ければ短いほど、長距離依存性の学習が容易になる[12]。したがって、異なる層のタイプで構成されるネットワークにおいて、任意の2つの入出力位置間の最大経路長も比較する。

表1にあるように、リカレント・レイヤーが $O(n)$ の逐次演算を必要とするのに対し、自己アテンション・レイヤーは逐次実行される演算の数が一定で、すべてのポジションを接続する。計算量の点では、自己注意層はリカレント層よりも高速である。

長さ n は、表現の次元数 d よりも小さい。これは、機械翻訳の最先端モデルで使用される文の表現、例えば、単語-部分文のような表現でよく見られる。

[38]やバイトペア[31]表現がある。非常に長いシーケンスを含むタスクの計算性能を向上させるために、自己注意は、それぞれの出力位置を中心とする入力シーケンスのサイズ r の近傍のみを考慮するように制限することができる。これにより、最大パス長は $O(n/r)$ に増加する。このアプローチについては今後の研究でさらに検討する予定である。

カーネル幅 $k < n$ の単一の畳み込み層は、入力と出力の位置のすべてのペアを接続しない。そのためには、連続カーネルの場合は $O(n/k)$ の畳み込み層のスタックが必要であり、拡張畳み込み[18]の場合は $O(\log_k(n))$ となり、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長くなる。しかし、分離可能な畳み込み[6]は、複雑さを $O(knd + nd^2)$ まで大幅に減少させる。しかし、 $k = n$ であっても、分離可能な畳み込みの複雑さは、自己アテンション層とポイントワイズ・フィードフォワード層の組み合わせに等しい。

副次的な利点として、自己アテンションはより解釈しやすいモデルをもたらす可能性がある。我々のモデルから注意の分布を調べ、付録で例を示し、議論する。個々のアテンション・ヘッドは明らかに異なるタスクの実行を学習するだけでなく、多くは文の構文構造や意味構造に関連した振る舞いを示すようである。

5 トレーニング

このセクションでは、我々のモデルのトレーニング方法について説明する。

5.1 トレーニングデータとバッチング

約450万文対からなる標準的なWMT 2014英独データセットで学習した。文はバイトペアエンコーディング[3]を使用してエンコードされ、約37000個のトークンでソースとターゲットの語彙を共有している。英語-フランス語については、3600万文からなる非常に大規模なWMT 2014英語-フランス語データセットを使用し、トークンを32000ワードピースの語彙[38]に分割した。文のペアは、おおよその配列の長さによってバッチ化された。各トレーニングバッチには、約25000のソーストークンと約25000のターゲットトークンを含む文ペアのセットが含まれる。

5.2 ハードウェアとスケジュール

我々は、8個のNVIDIA P100 GPUを搭載した1台のマシンでモデルを訓練した。本稿で説明したハイパーパラメータを使用したベースモデルでは、各トレーニングステップに約0.4秒を要した。合計100,000ステップ、12時間ベースモデルを訓練した。ビッグモデル（表3の一番下の行に記述）では、ステップ時間は1.0秒であった。ビッグモデルは300,000

ステップ (3.5日間) トレーニングしました。

5.3 オプティマイザー

Adam optimizer [20] を使用し、 $\theta_1 = 0.9$, $\theta_2 = 0.98$, $\hookrightarrow LI_3F5 = 10^{-9}$ とした。学習率は訓練期間中、以下の式に従って変化させた：

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

これは、最初の $warmup_steps$ 学習ステップでは学習率を直線的に増加させ、それ以降はステップ数の逆平方根に比例して減少させることに相当する。ここでは $warmup_steps = 4000$ とした。

5.4 正則化

トレーニングでは3種類の正則化を採用：

表2: Transformerは、英独間および英仏間のnewstest2014テストにおいて、わずかな学習コストで、これまでの最新モデルよりも優れたBLEUスコアを達成した。

モデル	BLEUTトレーニングコスト (FLOPs)			
	エンデ	EN-FR	EN-DE	EN-FR
バイトネット [18]	23.75			
ディープアット+ポゼッション [39]		39.2		$1.0 \cdot 10^{10}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^9$	$101.4 \cdot 10^9$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^9$	$101.5 \cdot 10^9$
MoE [32]	26.03	40.56	$2.0 \cdot 10^9$	$101.2 \cdot 10^9$
Deep-Att+PosUnkアンサンブル [39]		40.4		$8.0 \cdot 10^{10}$
GNMT + RLアンサンブル [38]	26.30	41.16	$1.8 \cdot 10^9$	$101.1 \cdot 10^9$
ConvS2S アンサンブル [9]	26.36	41.29	$7.7 \cdot 10^9$	$101.2 \cdot 10^9$
トランスフォーマー (ベースモデル)	27.3	38.1	$3.3 \cdot 10^{18}$	
トランスフォーマー (大)	28.4	41.8	$2.3 \cdot 10^{10}$	

残留ドロップアウト 各サブレイヤの出力にドロップアウト[33]を適用する。さらに、エンコーダスタックとデコーダスタックの両方において、エンベディングと位置エンコーディングの合計にドロップアウトを適用する。ベースモデルでは、 $P_{drop} = 0.1$ のレートを使用する。

ラベルスムージング 学習中、ラベルスムージング値 $\epsilon_{ls} = 0.1$ [36]を採用した。これは、モデルがより不確実であることを学習するため、複雑さを悪化させるが、精度とBLEUスコアを向上させる。

6 結果

6.1 機械翻訳

WMT 2014の英語からドイツ語への翻訳タスクにおいて、大きな変換モデル（表2のTransformer (big)）は、こ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以上上回り、28.4という新しい最先端のBLEUスコアを確立した。このモデルの構成を表3の最下行に示す。トレーニングには8台のP100 GPUで3.5日かかりました。私たちのベースモデルでさえ、競合モデルの何分の一かのトレーニングコストで、これまでに発表されたすべてのモデルやアンサンブルを凌駕しています。

WMT 2014の英語からフランス語への翻訳タスクにおいて、我々のビッグモデルはBLEUスコア41.0を達成し、以前に発表されたすべての単一モデルを凌駕し、以前の最先端モデルの1/4以下の学習コストで達成した。Transformer(big)モデルは、英語からフランス語へ

の翻訳において、0.3の代わりに $p_{drop} = 0.1$ の脱落率を使用しました。

ベースモデルについては、10分間隔で書き込まれた直近の5つのチェックポイントを平均して得られた単一のモデルを使用した。ビッグモデルについては、直近の20個のチェックポイントを平均した。ビームサイズ4、長さペナルティ $\alpha=0.6$ [38]のビームサーチを使用した。これらのハイパーパラメータは、開発セットで実験した後に選択した。推論中の最大出力長を入力長+50に設定したが、可能な限り早期に終了させた[38]。

表2は、結果をまとめたもので、翻訳品質と学習コストを文献にある他のモデル・アーキテクチャと比較しています。トレーニング時間、使用したGPUの数、および各GPUの持続的な単精度浮動小数点演算能力の推定値を乗じることで、モデルのトレーニングに使用した浮動小数点演算の数を推定します。⁵

6.2 モデルバリエーション

Transformerのさまざまなコンポーネントの重要性を評価するために、ベースモデルをさまざまな方法で変化させ、英語からドイツ語への翻訳のパフォーマンスの変化を測定した。

⁵K80、K40、M40、P100については、それぞれ2.8、3.7、6.0、9.5TFLOPSの値を使用した。

表3: Transformerアーキテクチャのバリエーション。記載されていない値は、基本モデルの値と同じである。すべてのメトリクスは英独翻訳開発セットnewstest2013のものである。記載されている当惑度は、我々のバイトペアエンコーディングに従った単語単位であり、単語単位の当惑度と比較すべきではない。

	N	ドモデル	ダッフ	h	d_k	d_v	ドロップ	ϵ	ト	有形 固定	ブルー (デブ)	パラメ ータ ₆
ベー	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	
大き	6	1024	4096	16			0.3		300K	4.33	26.4	213

開発セット、newstest2013。前節で説明したようにビームサーチを使用した、チェックポイントの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、セクション3.2.2で説明したように、計算量を一定に保ちながら、アテンションヘッドの数、アテンションキーと値の次元を変化させている。シングルヘッドアテンションは最良の設定よりも0.9BLEU悪いが、ヘッド数が多すぎると品質も低下する。

表3の(B)の行を見ると、注目キーのサイズ d_k を小さくすると、モデルの品質が低下することがわかる。このことは、互換性を決定することは容易ではなく、ドット積よりも洗練された互換性関数が有益であることを示唆している。さらに(C)行と(D)行では、予想されるように、より大きなモデルの方が優れており、ドロップアウトはオーバーフィッティングを避けるのに非常に有効であることがわかる。(E)の行では、正弦波位置エンコーディングを学習された位置埋め込み[9]に置き換えて、ベースモデルとほぼ同じ結果を観察する。

6.3 英語選挙区解析

Transformerが他のタスクに一般化できるかどうかを評価するために、英語の構文解析の

実験を行った。このタスクには特有の課題がある：出力は強い構造的制約を受け、入力よりもかなり長い。さらに、RNNのsequence-to-sequenceモデルは、小さなデータ領域で最先端の結果を達成することができなかった[37]。

Penn Treebank [25]のWall Street Journal (WSJ)部分、約40K文に対して、 $d_{model}=1024$ の4層変換器を学習した。また、約1,700万文[37]の大規模なhigh-confidenceとBerkleyParserコーパスを使用して、半教師あり設定で訓練した。WSJのみの設定では16Kの語彙を使用し、半教師付き設定では32Kの語彙を使用した。

我々は、セクション22の開発セットで、ドロップアウト、注意と残差（セクション5.4）、学習率、ビームサイズを選択するために少数の実験を行っただけで、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変更しませんでした。推論中

表4: The Transformerは英語の構文解析によく一般化する（結果はWSJのセクション23にある）

パーサー	トレーニング	WSJ 23 F1
ウィンヤルス&カイザー他 (2014) 【37】	WSJのみ、差別的	88.3
ペトロフ他 (2006) [29]。	WSJのみ、差別的	90.4
Zhuら (2013) 【40】	WSJのみ、差別的	90.4
ダイアーら (2016) [8]。	WSJのみ、差別的	91.7
トランスフォーマー (4層)	WSJのみ、差別的	91.3
Zhuら (2013) 【40】	半監視下	91.3
黄&ハーバー (2009) 【14】	半監視下	91.3
McCloskyら (2006) [26]。	半監視下	92.1
ウィンヤルス&カイザー他 (2014) 【37】	半監視下	92.1
トランスフォーマー (4層)	半監視下	92.7
ルオンら (2015) 【23】	マルチタスク	93.0
ダイアーら (2016) [8]。	ジェネレイティブ	93.3

は、最大出力長を入力長+300に増加させた。ビームサイズは21、 $\alpha = 0.3$ 。
WSJのみと半教師付き設定の両方で。

表4の結果から、タスクに特化したチューニングがないにもかかわらず、我々のモデルは驚くほど良好な結果を示し、リカレント・ニューラル・ネットワーク・グラマー[8]を除く、これまでに報告されたすべてのモデルよりも良い結果をもたらした。

RNNのsequence-to-sequenceモデル[37]とは対照的に、Transformerは、40K文のWSJ訓練セットだけで訓練した場合でも、Berkeley-Parser[29]を上回る。

7 結論

この研究では、エンコーダー・デコーダー・アーキテクチャーで最もよく使われるリカレント層をマルチヘッド自己注意に置き換えた、完全に注意に基づく最初のシーケンス変換モデルであるトランスフォーマーを発表した。

翻訳タスクにおいて、Transformerはリカレント層や畳み込み層に基づくアーキテクチャよりも大幅に高速に学習できる。WMT 2014英独翻訳タスクとWMT 2014英仏翻訳タスクの両方において、我々は新たな最先端技術を達成した。前者のタスクでは、我々の最良のモデルは、以前に報告された全てのアンサンブルを凌駕した。

我々は注意に基づくモデルの将来性に期待しており、他のタスクにも適用する予定である。テキスト以外の入出力モダリティを含む問題にTransformerを拡張し、画像、音声、動画などの大規模な入出力を効率的に扱うための局所的で制限された注意メカニズムを研究する予定です。世代を逐次的でなくすることも我々の研究目標である。

モデルの訓練と評価に使用したコードは、<https://github.com/tensorflow/tensor2tensor>で入手できる。

謝辞 有益なコメント、訂正、インスピレーションを与えてくれたナル・カルチブレナー

氏とステファン・グース氏に感謝する。

参考文献

- [1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- [2] Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio. アライメントと翻訳の共同学習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, and Quoc V. Le. ニュー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, and Mirella Lapata. 機械読解のための長期短期記憶ネットワーク. *arXiv preprint arXiv:1601.06733*, 2016.

- [5] 趙京鉉、Bart van Merriënboer、Caglar Gulcehre、Fethi Bougares、Holger Schwenk、Yoshua Bengio。統計的機械翻訳のためのrnnエンコーダデコーダ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: *ArXiv preprint arXiv:1610.02357*, 2016.
- [7] チョン・ジュニョン、チャグラル・ギュルチェレ、チョ・キョンヒョン、ベンジオ・ヨシュア。シーケンスモデリングにおけるゲート型リカレントニューラルネットワークの経験的評価。*CoRR*, abs/1412.3555, 2014.
- [8] クリス・ダイアー、アドヒグナ・クンコロ、ミゲル・バレストロス、ノア・A・スミス。リカレント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, and Yann N. Dauphin. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122v2*, 2017.
- [10] アレックス・グレイブスリカレントニューラルネットワークによるシーケンスの生成. *arXiv preprint arXiv:1308.0850*, 2013.
- [11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *IEEE Conference of the Computer Vision and Pattern Recognition*, pages 770-778, 2016.
- [12] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. リカレントネットにおける勾配フロー: 長期依存関係の学習の難しさ, 2001.
- [13] ゼップ・ホッホライター、ユルゲン・シュミットフーバー。長期短期記憶。 *Neural computation*, 9(8):1735-1780, 1997.
- [14] 黄中強、メアリー・ハーパー。言語間の潜在的注釈を用いたPCFG文法の自己学習。 *自然言語処理における経験的方法に関する2009年会議議事録*、ページ832-841。ACL, August 2009.
- [15] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 言語モデリングの限界を探索. *arXiv preprint arXiv:1602.02410*, 2016.
- [16] ウカシュ・カイザーとサミー・ベングイ能動記憶は注意に取って代われるか? In *Advances in Neural Information Processing Systems, (NIPS)*, 2016.
- [17] Łukasz Kaiser and Ilya Sutskever. ニューラルGPUはアルゴリズムを学習する。In *International Conference on Learning Representations (ICLR)*, 2016.
- [18] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Kory Kavukcuoglu. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099v2*, 2017.
- [19] ユン・キム、カール・デントン、ルオン・ホアン、アレクサンダー・M・ラッシュ。構造化注意ネットワーク。In *International Conference on Learning Representations*, 2017.

- [20] ディエデリク・キングマ、ジミー・バAdam: 確率的最適化のための手法。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. *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.

- [25] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 大規模な英語注釈付きコーパスの構築: The penn treebank. *計算言語学*, 19(2):313-330, 1993.
- [26] David McClosky, Eugene Charniak, and Mark Johnson. 構文解析のための効果的な自己訓練。 *NAACL Human Language Technology Conference, Main Conference*, pages 152-159. ACL, June 2006.
- [27] アンカー・パリク、オスカー・テックストローム、ディパンジャン・ダス、ヤコブ・ウスコレイト。 分解可能な注意モデル。 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] スラヴ・ペトロフ、レオン・バレット、ロマン・ティボー、ダン・クライン。 正確でコンパクト、かつ解釈可能な木注釈の学習。 *第21回計算言語学国際会議および第44回ACL年次総会予稿集*, ページ433-440. ACL, July 2006.
- [30] Ofir Press and Lior Wolf. 言語モデルを改善するための出力埋め込みの利用. *arXiv preprint arXiv:1608.05859*, 2016.
- [31] Rico Sennrich, Barry Haddow, and Alexandra Birch. *ArXiv preprint arXiv:1508.07909*, 2015.
- [32] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarczyk, Andy Davis, Quoc Le, Geoffrey Hinton, Jeff Dean. とんでもなく大きなニューラルネットワーク: 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. (1)ニューラルネットワークのオーバーフィッティングを防ぐ簡単な方法。 *Journal of Machine Learning Research*, 15(1):1929-1958, 2014.
- [34] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, Rob Fergus. エンドツーエンドのメモリネットワーク。 C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2440-2448. カランアソシエーツ、2015年
- [35] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. ニューラルネットを用いたシーケンス間学習。 In *Advances in Neural Information Processing Systems*, pages 3104-3112, 2014.
- [36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and 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] ヴィンヤルス&カイザー、クー、ペトロフ、スツツケバー、ヒント。 外国語としての文法。 で

神経情報処理システムの進歩, 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のニューラル機械翻訳システム: *arXiv preprint arXiv:1609.08144*, 2016.
- [39] 李鵬、周杰倫、曹英、王旭光、徐韋。ニューラル機械翻訳のためのファストフォワード接続を持つディープリカレントモデル。 *CoRR*, abs/1606.04199, 2016.
- [40] 朱夢華、張岳、陳文良、張敏、朱景博。高速で正確なシフト還元構文解析.このような場合、「Accounting」(訳注:日本語訳)を利用することで、「Accounting」(訳注:日本語訳)を利用することができる。ACL, August 2013.

アテンション・ビジュアライゼーション

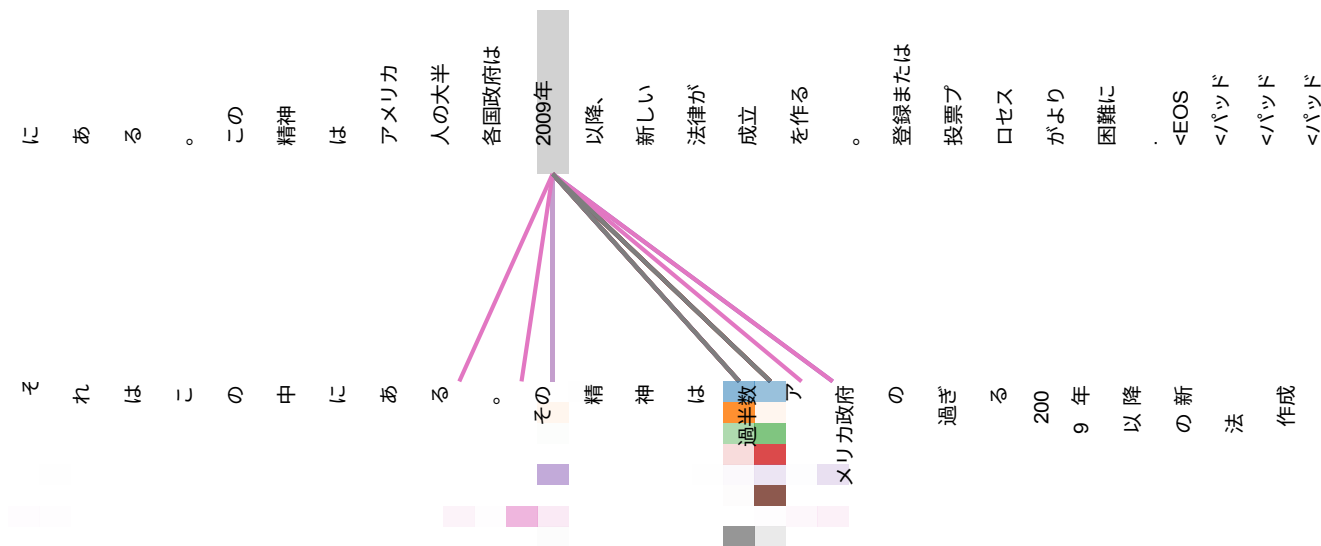


図3: 6層中5層目のエンコーダーの自己注意における、遠距離の依存関係に従う注意メカニズムの例。注意の頭の多くは、動詞「making」の遠距離の依存関係に注意を向け、「making...more difficult」というフレーズを完成させる。ここでは「making」という単語に対する注意のみを示す。異なる色は異なる頭部を表す。カラーで見るのがベスト。

ヨンは
ただ
-
これが私たちです
行方不明
,
私の
意見
.
<EOS
<パッド

法
は
決
し
て
完
全
で
は
な
い
,
し
か
し
ア
プ
リ
ケ
ー
シ

うに
<パッド

につい
ては
法律
して
完
壁であ
る
,
しかし
その
アプリ
ライセ
ンシー
た
だ
-
こ
れが
私た
ちで
す
行方不明
,
思

その

図4. *the head of the sentence*にも2つの注意ヘッドがあり、*the head of the sentence*がアナフォラの解決に関与している。
。上：頭部5の完全な注意。下：注意ヘッド5と6の*its*という単語だけの注意。この単語の注意は非常に鋭い。

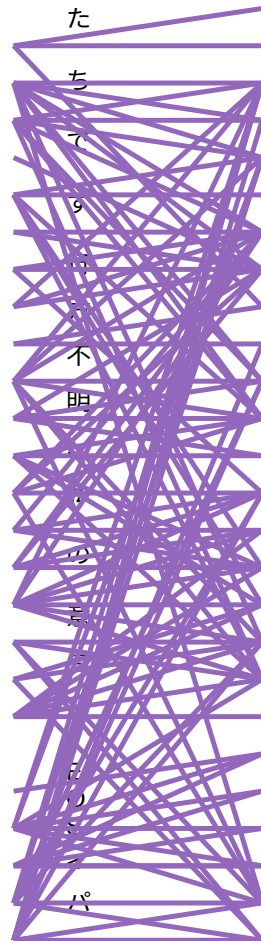
法は決して
完璧である
,
しかし、その適用
べきである
ただ
-
これが私たちです
行方不明
,
思うに
.
<EOS
<パッド

法
は
決
し
て
完
全
で
は
な
い
,
し
か
し
ア
プ
リ
ケ
ー
シ
ヨ
ン

は ッド

た

だ
-
こ
れ
が
私
た
ち
で
す
が
不
明
な
点
の
ま
ま
に
お
ま
か
し
て
い
ま
す
パ



ョンは
 ただ
 -
 これが私たちです
 行方不明
 ,
 私の
 意見
 .
 <EOS
 <パッド

法
 は
 決
 し
 て
 完
 全
 で
 は
 な
 い
 ,
 し
 か
 し
 ア
 プ
 リ
 ケ
 ー
 シ

うに
 .
 <EOS
 <パッド

法
 は決
 して
 完
 全
 であ
 る
 ,
 し
 そ
 の適用
 がある
 た
 だ
 -
 こ
 れが
 私た
 ちで
 す

図5: アテンション・ヘッドの多くが、文の構造に関連していると思われる行動を示している。このような例を2つ挙げるが、これは6層中5層目のエンコーダーの自己注意から得られた2つの異なるヘッドによるものである。ヘッドは明らかに異なるタスクを実行することを学習した。

法は決して
完璧である
、
しかし、その適
用
べきである
ただ
-
これが私たちです
行方不明
、
思うに
.
<EOS
<パッド

法は決して完全ではない，しかしアプリケーション

「**ア**、**私**の意見・**EOS**・**パ**」