Unter der Voraussetzung einer ordnungsgemäßen Quellenangabe räumt Google hiermit die Erlaubnis zur Vervielfältigung der Tabellen und Abbildungen in diesem Artikel ausschließlich für die Verwendung in journalistischen oder wissenschaftlichen Werken ein.

Aufmerksamkeit ist alles, was Sie brauchen

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

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

Zusammenfass
ung

Die dominierenden Sequenz-Transduktionsmodelle basieren auf komplexen rekurrenten oder konvolutionellen neuronalen Netzen, die einen Encoder und einen Decoder enthalten. Die am besten abschneidenden Modelle verbinden den Encoder und den Decoder zudem über einen Aufmerksamkeitsmechanismus (Attention Mechanism). Wir schlagen eine neue, einfache Netzwerkarchitektur vor, den Transformer, der ausschließlich auf Aufmerksamkeitsmechanismen basiert und vollständig auf Rekurrenz und Faltung verzichtet. Experimente zu zwei Maschinelle-Übersetzungsaufgaben zeigen, dass diese Modelle in der Qualität überlegen sind, gleichzeitig besser parallelisierbar sind und deutlich weniger Trainingszeit erfordern. Unser Modell erreicht 28,4 BLEU bei der WMT 2014 Englisch-zu-Deutsch-Übersetzungsaufgabe und verbessert damit die bisherigen Bestwerte, einschließlich Ensembles, um mehr als 2 BLEU. Bei der WMT 2014 Englisch-zu-Französisch-Übersetzungsaufgabe erzielt unser Modell nach 3,5 tägiger Training auf acht GPUs einen neuen State-of-the-Art-BLEU-Score von 41,8 für ein einzelnes Modell, was nur einem kleinen Bruchteil der Trainingskosten der besten Modelle aus der Literatur entspricht. Wir zeigen, dass sich der Transformer gut auf andere Aufgaben verallgemeinern lässt, indem wir ihn erfolgreich auf das englische Konstituenz-Parsing sowohl mit großen als auch mit begrenzten Trainingsdaten anwenden.

*Gleicher Beitrag. Die Reihenfolge der Auflistung ist zufällig. Jakob schlug vor, RNNs durch Self-Attention zu ersetzen, und begann mit der Bewertung dieser Idee. Ashish entwarf und implementierte mit Illia die ersten Transformer-Modelle und war in jeder Hinsicht entscheidend an dieser Arbeit beteiligt. Noam schlug skalierte Dot-Produkt-Attention, Multi-Head-Attention und die parameterfreie Positionsrepräsentation vor und war an fast jedem Detail beteiligt. Niki entwarf, implementierte, justierte und bewertete unzählige Modellvarianten in unserem ursprünglichen Codebase und tensor2tensor. Llion experimentierte ebenfalls mit neuen Modellvarianten, war verantwortlich für unsere erste Codebase sowie für effiziente Inferenz und Visualisierungen. Lukasz und Aidan verbrachten unzählige lange Tage damit, verschiedene Teile zu entwerfen und tensor2tensor zu implementieren, unsere frühere Codebase zu ersetzen, die Ergebnisse erheblich zu verbessern und unsere Forschung massiv zu beschleunigen.

Arbeit durchgeführt während einer Tätigkeit bei Google Brain.

Arbeit durchgeführt während der Tätigkeit bei Google Research.

31. Konferenz über neuronale Informationverarbeitungs-systeme (NIPS 2017), Long Beach, CA, USA

1 Einleitung

Rekurrente neuronale Netze, Long Short-Term Memory [13] und insbesondere gated recurrent [7] neuronale Netze haben sich als State-of-the-Art-Ansätze bei Sequenzmodellierungs- und Transduktionsproblemen wie Sprachmodellierung und maschineller Übersetzung [35, 2, 5] fest etabliert. Zahlreiche Bemühungen setzten sich seither fort, die Grenzen rekurrenter Sprachmodelle und Encoder-Decoder-Architekturen zu erweitern [38, 24, 15].

Rekurrente Modelle faktorisieren die Berechnung typischerweise entlang der Symbolpositionen der Eingabe- und Ausgabesequenzen. Durch die Zuordnung der Positionen zu Schritten in der Rechenzeit erzeugen sie eine Folge von versteckten Zuständen h_t als Funktion des vorherigen versteckten Zustands h_{t-1} und der Eingabe für die Position t. Diese inhärent sequenzielle Natur schließt eine Parallelisierung innerhalb von Trainingsbeispielen aus, was bei längeren Sequenzlängen kritisch wird, da Speicherbeschränkungen das Batching über Beispiele hinweg begrenzen. Jüngste Arbeiten haben erhebliche Verbesserungen der rechnerischen Effizienz durch Faktorisierungstricks [21] und bedingte Berechnung [32] erzielt, wobei auch die Modellleistung im Falle des letzteren verbessert wurde. Die fundamentale Beschränkung der sequenziellen Berechnung bleibt jedoch bestehen.

Aufmerksamkeitsmechanismen sind zu einem integralen Bestandteil überzeugender Sequenzmodellierungs- und Transduktionsmodelle in verschiedenen Aufgaben geworden, wodurch die Modellierung von Abhängigkeiten ohne Berücksichtigung ihrer Distanz in den Eingangs- oder Ausgangssequenzen ermöglicht wird [2, 19]. In allen, aber wenigen Fällen [27] werden solche Aufmerksamkeitsmechanismen jedoch in Kombination mit einem rekurrenten Netzwerk eingesetzt.

In dieser Arbeit schlagen wir den Transformer vor, eine Modellarchitektur, die auf Rekurrenz verzichtet und stattdessen vollständig auf einen Aufmerksamkeitsmechanismus (Attention-Mechanismus) zurückgreift, um globale Abhängigkeiten zwischen Eingabe und Ausgabe zu erfassen. Der Transformer ermöglicht eine deutlich stärkere Parallelisierung und kann nach einer Trainingsdauer von lediglich zwölf Stunden auf acht P100-GPUs einen neuen Stand der Technik in der Übersetzungsqualität erreichen.

2 Hintergrund

Das Ziel der Reduzierung sequenzieller Berechnungen bildet auch die Grundlage für den Extended Neural GPU [16], ByteNet [18] und ConvS2S [9], bei denen alle convolutionale neuronale Netze als grundlegender Baustein verwendet werden, um versteckte Repräsentationen parallel für alle Eingangs- und Ausgangspositionen zu berechnen. In diesen Modellen nimmt die Anzahl der Operationen, die erforderlich ist, um Signale von zwei beliebigen Eingangs- oder Ausgangspositionen in Beziehung zu setzen, mit der Distanz zwischen den Positionen zu, linear für ConvS2S und logarithmisch für ByteNet. Dies erschwert das Erlernen von Abhängigkeiten zwischen entfernten Positionen [12]. Beim Transformer wird dies auf eine konstante Anzahl von Operationen reduziert, freilich zu Lasten einer verringerten effektiven Auflösung aufgrund der Durchschnittsbildung über aufmerksamkeitsgewichtete Positionen, einem Effekt, dem wir mit Multi-Head Attention begegnen, wie in Abschnitt 3.2 beschrieben.

Self-Attention, manchmal auch als Intra-Attention bezeichnet, ist ein Aufmerksamkeitsmechanismus, der verschiedene Positionen einer einzelnen Sequenz verknüpft, um eine Repräsentation der Sequenz zu berechnen. Self-Attention wurde erfolgreich in einer Vielzahl von Aufgaben eingesetzt, darunter Leseverstehen, abstraktive Zusammenfassung, textuelle Implikation und das Lernen sequenzunabhängiger Satzrepräsentationen [4, 27, 28, 22].

End-to-End-Gedächtnisnetzwerke basieren auf einem rekurrenten Aufmerksamkeitsmechanismus anstelle von sequenzalignierter Rekurrenz und haben sich als leistungsfähig bei Aufgaben zur Beantwortung einfacher Sprachfragen und zur Sprachmodellierung erwiesen [34].

So weit unsere Kenntnisse reichen, ist der Transformer jedoch das erste Transduktionsmodell, das ausschließlich auf Selbst-Aufmerksamkeit zurückgreift, um Darstellungen von Eingabe und Ausgabe zu berechnen, ohne sequenzausgerichtete RNNs oder Faltung zu verwenden. In den folgenden Abschnitten werden wir den Transformer beschreiben, die Motivation für die Selbst-Aufmerksamkeit darlegen und seine Vorteile im Vergleich zu Modellen wie [17, 18] und [9] diskutieren.

3 Modellarchitektur

Die meisten wettbewerbsfähigen neuronalen Sequenztransduktionsmodelle verfügen über eine Encoder-Decoder-Struktur [5, 2, 35]. Hierbei abbildet der Encoder eine Eingabesequenz von Symbolrepräsentationen $(x_1, \ldots, x_n)$ auf eine Sequenz kontinuierlicher Repräsentationen $\mathbf{z} = (z_1, \ldots, z_n)$. Unter Zuhilfenahme von $\mathbf{z}$ generiert der Decoder anschließend eine Ausgabesequenz $(y_1, \ldots, y_m)$ von Symbolen ein Element nach dem anderen. In jedem Schritt ist das Modell auto-regressiv [10] und verarbeitet die zuvor generierten Symbole als zusätzlichen Input bei der Erzeugung des nächsten.

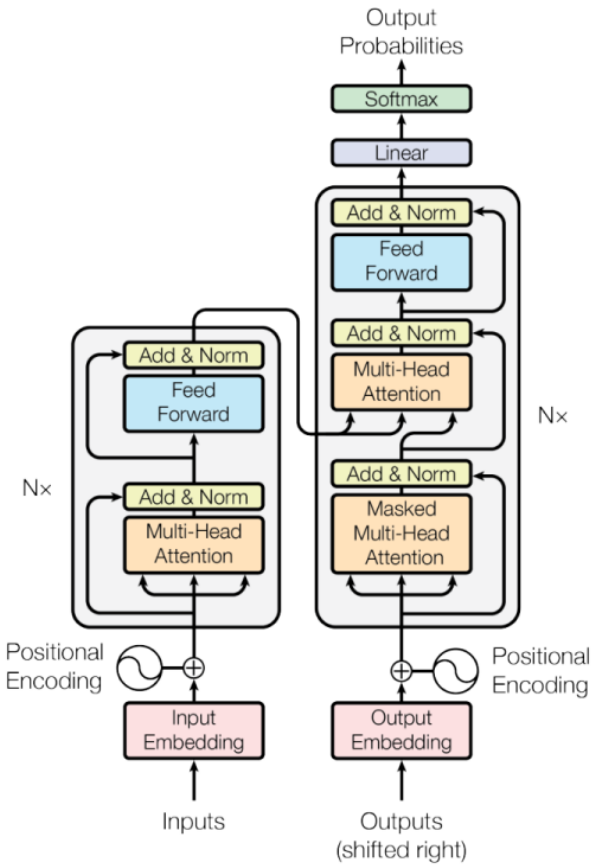


Abbildung 1: Die Transformer-Modellarchitektur.

Der Transformer folgt dieser Gesamtarchitektur unter Verwendung von gestapelten Self-Attention- und punktweisen, vollverknüpften Schichten sowohl für den Encoder als auch für den Decoder, die in der linken bzw. rechten Hälfte von Abbildung 1 dargestellt sind.

3.1 Encoder- und Decoder-Stapel

Encoder: Der Encoder besteht aus einem Stack von $N=6$ identischen Schichten. Jede Schicht verfügt über zwei Sub-Schichten. Die erste ist ein Multi-Head-Selbst-Aufmerksamkeitsmechanismus, und die zweite ist ein einfaches, positionweise vollverbundenes Feed-Forward-Netzwerk. Wir verwenden eine Residualverbindung [11] um jede der beiden Sub-Schichten, gefolgt von einer Layer-Normalisierung [1]. Das bedeutet, die Ausgabe jeder Sub-Schicht ist LayerNorm(x + Sublayer(x)), wobei Sublayer(x) die Funktion ist, die durch die Sub-Schicht selbst implementiert wird. Um diese Residualverbindungen zu erleichtern, erzeugen alle Sub-Schichten im Modell sowie die Einbettungsschichten Ausgaben der Dimension $d_{\text{model}}=512$.

Decoder: Der Decoder besteht ebenfalls aus einem Stack von $N=6$ identischen Schichten. Zusätzlich zu den beiden Sub-Schichten in jeder Encoder-Schicht fügt der Decoder eine dritte Sub-Schicht hinzu, die eine Multi-Head-Aufmerksamkeit über dem Output des Encoder-Stacks durchführt. Ähnlich wie beim Encoder verwenden wir Residualverbindungen um jede der Sub-Schichten, gefolgt von einer Layer-Normalisierung. Wir modifizieren auch die Self-Attention-Sub-Schicht im Decoder-Stack, um zu verhindern, dass Positionen auf nachfolgende Positionen aufmerksam werden. Diese Maskierung, kombiniert mit der Tatsache, dass die Output-Embeddings um eine Position versetzt sind, stellt sicher, dass die Vorhersagen für Position i nur von den bekannten Outputs an Positionen abhängen können, die kleiner als i sind.

3.2 Aufmerksamkeit

Eine Attention-Funktion lässt sich als Abbildung einer Abfrage und einer Menge von Schlüssel-Wert-Paaren auf eine Ausgabe beschreiben, wobei die Abfrage, die Schlüssel, die Werte und die Ausgabe jeweils Vektoren sind. Die Ausgabe wird als gewichtete Summe berechnet.

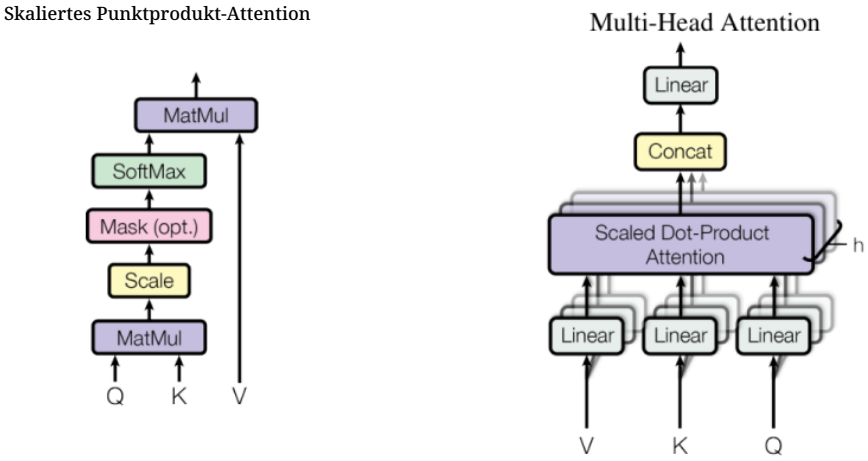


Abbildung 2: (links) Skaliertes Dot-Produkt-Attention. (rechts) Multi-Head-Attention besteht aus mehreren Attention-Schichten, die parallel ablaufen.

der Werte, wobei das jedem Wert zugewiesene Gewicht durch eine Kompatibilitätsfunktion der Abfrage mit dem entsprechenden Schlüssel berechnet wird.

3.2.1 Skaliertes Dot-Produkt-Attention

Wir bezeichnen unsere spezifische Aufmerksamkeitsmechanismus als "Skaliertes Dot-Produkt-Aufmerksamkeit" (Abbildung 2). Die Eingabe besteht aus Abfragen und Schlüsseln der Dimension d_k sowie Werten der Dimension d_v. Wir berechnen die Dot-Produkte der Abfrage mit allen Schlüsseln, teilen jedes durch $\sqrt{d_k}$ und wenden eine Softmax-Funktion an, um die Gewichte auf den Werten zu erhalten.

In der Praxis berechnen wir die Aufmerksamkeitsfunktion für eine Menge von Abfragen gleichzeitig, die gemeinsam in einer Matrix Q zusammengefasst sind. Die Schlüssel und Werte sind ebenfalls zu den Matrizen K und V zusammengefasst. Wir berechnen die Ausgabe-Matrix wie folgt:

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

Die beiden am häufigsten verwendeten Attention-Funktionen sind additive Attention [2] und Dot-Product-(multiplikative) Attention. Die Dot-Product-Attention ist unserem Algorithmus identisch, mit Ausnahme des Skalierungsfaktors $\frac{1}{\sqrt{d_k}}$. Additive Attention berechnet die Kompatibilitätsfunktion mithilfe eines Feed-Forward-Netzwerks mit einer einzelnen versteckten Schicht. Obwohl sich beide in der theoretischen Komplexität ähneln, ist die Dot-Product-Attention in der Praxis deutlich schneller und speichereffizienter, da sie unter Verwendung hochoptimierten Matrixmultiplikationscodes implementiert werden kann.

Während die beiden Mechanismen für kleine Werte von d_k ähnlich abschneiden, übertrifft die additive Aufmerksamkeitsfunktion (Attention) bei größeren Werten von d_k das skalierte Skalarprodukt ohne Skalierung [3]. Wir vermuten, dass die Skalarprodukte für große Werte von d_k betragsmäßig groß werden und die Softmax-Funktion in Bereiche drängen, in denen sie extrem kleine Gradienten aufweist 4. Um diesem Effekt entgegenzuwirken, skalieren wir die Skalarprodukte mit $\frac{1}{\sqrt{d_k}}$.

3.2.2 Multi-Head-Attention

Anstatt eine einzelne Attention-Funktion mit d_{model}-dimensionalen Schlüsseln, Werten und Abfragen durchzuführen, erwies es sich als vorteilhaft, die Abfragen, Schlüssel und Werte h-mal mit unterschiedlichen, gelernten linearen Projektionen auf die Dimensionen d_k, d_k und d_v linear zu projizieren. Für jede dieser projizierten Versionen von Abfragen, Schlüsseln und Werten führen wir dann die Attention-Funktion parallel durch, was d_v-dimensionale

[4] Um zu veranschaulichen, warum die Skalarprodukte groß werden, nehme man an, dass die Komponenten von q und k unabhängige Zufallsvariablen mit Mittelwert 0 und Varianz 1 sind. Dann haben ihr Skalarprodukt, $q \cdot k = \sum_{i=1}^{d_k} q_i k_i$, Mittelwert 0 und Varianz d_k.

Ausgabewerte. Diese werden verkettet und erneut projiziert, was zu den Endwerten führt, wie in Abbildung 2 dargestellt.

Multi-Head-Attention ermöglicht es dem Modell, Informationen aus verschiedenen Repräsentationsunterräumen an unterschiedlichen Positionen gleichzeitig zu berücksichtigen. Bei einer einzigen Attention-Head unterdrückt das Mitteln diesen Prozess.

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

Wobei die Projektionen Parametermatrizen $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ und $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$ sind.

In dieser Arbeit verwenden wir $h = 8$ parallele Aufmerksamkeits-Schichten, auch Heads genannt. Für jedes dieser Heads setzen wir $d_k = d_v = d_{\text{model}}/h = 64$. Aufgrund der reduzierten Dimension jedes Heads sind die gesamten Rechenkosten ähnlich denen der Single-Head-Aufmerksamkeit mit voller Dimensionalität.

3.2.3 Anwendungen der Aufmerksamkeitsmechanismen in unserem Modell

Der Transformer verwendet Multi-Head-Attention auf drei verschiedene Weisen:

In Encoder-Decoder-Aufmerksamkeit "-Schichten stammen die Abfragen von der vorherigen Decoder-Schicht, während die Speicher Schlüssel und Werte vom Ausgang des Encoders kommen. Dies ermöglicht es jeder Position im Decoder, auf alle Positionen der Eingabesequenz zuzugreifen. Dies nachahmt die typischen Encoder-Decoder-Aufmerksamkeitsmechanismen in Sequenz-zu-Sequenz-Modellen wie [38, 2, 9].

Der Encoder enthält Self-Attention-Schichten. In einer Self-Attention-Schicht stammen alle Schlüssel, Werte und Abfragen vom selben Ort, in diesem Fall von der Ausgabe der vorherigen Schicht im Encoder. Jede Position im Encoder kann auf alle Positionen der vorherigen Schicht des Encoders zugreifen.

Ähnlich wie bei den Encoder-Schichten erlauben die Self-Attention-Schichten im Decoder jeder Position, auf alle Positionen im Decoder bis einschließlich dieser Position zuzugreifen. Wir müssen einen informationellen Fluss nach links im Decoder verhindern, um die autoregressive Eigenschaft zu wahren. Dies implementieren wir innerhalb der skalierten Dot-Produkt-Attention, indem wir (durch Setzen auf $-\infty$) alle Werte in der Eingabe der Softmax-Funktion maskieren, die illegalen Verbindungen entsprechen. Siehe Abbildung 2.

3.3 Positionsweise Feed-Forward-Netzwerke

Zusätzlich zu den Attention-Unterschichten enthält jede der Schichten in unserem Encoder und Decoder ein vollverknüpftes Feed-Forward-Netzwerk, das separat und identisch auf jede Position angewendet wird. Dieses besteht aus zwei linearen Transformationen mit einer ReLU-Aktivierung dazwischen.

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

Obwohl die linearen Transformationen über verschiedene Positionen hinweg identisch sind, verwenden sie von Schicht zu Schicht unterschiedliche Parameter. Eine weitere Beschreibung dieses Vorgangs erfolgt als zwei Faltungen mit Kernelgröße 1. Die Dimensionalität von Eingabe und Ausgabe beträgt $d_{\text{model}} = 512$, und die innere Schicht hat eine Dimensionalität von $d_{ff} = 2048$.

3.4 Embeddings und Softmax

Ähnlich wie bei anderen Sequenztransduktionsmodellen verwenden wir gelernte Embeddings, um die Eingabe- und Ausgabe-Tokens in Vektoren der Dimension d_{model} umzuwandeln. Wir nutzen auch die übliche gelernte lineare Transformation und Softmax-Funktion, um die Decoder-Ausgabe in vorhergesagte Next-Token-Wahrscheinlichkeiten zu überführen. In unserem Modell teilen wir dieselbe Gewichtsmatrix zwischen den beiden Embedding-Schichten und der Pre-Softmax-linearen Transformation, ähnlich wie in [30]. In den Embedding-Schichten multiplizieren wir diese Gewichte mit $\sqrt{d_{\text{model}}}$.

Tabelle 1: Maximale Pfadlängen, Komplexität pro Schicht und minimale Anzahl sequentieller Operationen für verschiedene Schichttypen. n ist die Sequenzlänge, d die Repräsentationsdimension, k die Kernelgröße der Faltungen und r die Größe der Nachbarschaft bei beschränkter Selbst-Aufmerksamkeit.

Schichttyp	Komplexität pro Schicht	Sequenziell Operationen	Maximale Pfadlänge
Selbst-Aufmerksamkeit	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Rekurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Konvolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(\log_k(n))$
Selbst-Aufmerksamkeit (eingeschränkt)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

3.5 Positionale Kodierung

Da unser Modell weder Rekursion noch Faltung enthält, müssen wir Informationen über die relative oder absolute Position der Tokens in der Sequenz einfließen lassen, damit das Modell die Reihenfolge der Sequenz nutzen kann. Dazu fügen wir den Input-Embeddings an den unteren Enden der Encoder- und Decoder-Stacks "positionale Kodierungen" hinzu. Die positionale Kodierungen haben dieselbe Dimension d_{model} wie die Embeddings, sodass diese addiert werden können. Es gibt viele Möglichkeiten zur Wahl von positionalen Kodierungen, sowohl gelernte als auch feste [9].

In dieser Arbeit verwenden wir Sinus- und Cosinus-Funktionen unterschiedlicher Frequenzen:

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

wobei pos die Position und i die Dimension ist. Das bedeutet, jede Dimension der Positionsencodierung entspricht einer Sinusfunktion. Die Wellenlängen bilden eine geometrische Progression von 2π bis $10000 \cdot 2\pi$. Wir wählten diese Funktion, weil wir annahmen, dass sie es dem Modell ermöglichen würde, leicht zu lernen, auf relative Positionen zu achten, da sich PE_{pos+k} für jeden festen Offset k als lineare Funktion von PE_{pos} darstellen lässt.

Wir experimentierten auch mit der Verwendung gelernter positionaler Einbettungen [9] und stellten fest, dass beide Versionen nahezu identische Ergebnisse lieferten (siehe Tabelle 3, Zeile (E)). Wir wählten die sinusförmige Version, da sie das Modell möglicherweise in die Lage versetzt, auf Sequenzlängen zu extrapolieren, die länger sind als diejenigen, die während des Trainings encountered wurden.

4 Warum Self-Attention

In diesem Abschnitt vergleichen wir verschiedene Aspekte von Self-Attention-Schichten mit den rekurrenten und Faltungsschichten, die üblicherweise zur Abbildung einer Variabellängensequenz von Symbolrepräsentationen $(x_1, \ldots, x_n)$ auf eine weitere Sequenz gleicher Länge $(z_1, \ldots, z_n)$ verwendet werden, wobei $x_i, z_i \in \mathbb{R}^d$, wie beispielsweise eine versteckte Schicht in einem typischen Sequenztransduktions-Encoder oder -Decoder. Wir motivieren unseren Einsatz von Self-Attention durch drei desiderata.

Eines ist die gesamte rechnerische Komplexität pro Schicht. Das andere ist der Umfang der Berechnungen, die parallelisiert werden können, gemessen an der minimalen Anzahl erforderlicher sequentieller Operationen.

Die dritte ist die Pfadlänge zwischen langreichweitigen Abhängigkeiten im Netzwerk. Das Lernen langreichweitiger Abhängigkeiten stellt eine zentrale Herausforderung in vielen Sequenztransduktionsaufgaben dar. Ein entscheidender Faktor, der die Fähigkeit zum Erlernen solcher Abhängigkeiten beeinflusst, ist die Länge der Pfade, die Vorwärts- und Rückwärtssignale im Netzwerk zurücklegen müssen. Je kürzer diese Pfade zwischen beliebigen Kombinationen von Positionen in den Eingangs- und Ausgangssequenzen sind, desto einfacher ist es, langreichweitige Abhängigkeiten zu lernen [12]. Daher vergleichen wir auch die maximale Pfadlänge zwischen beliebigen Eingangs- und Ausgangspositionen in Netzwerken, die aus den verschiedenen Schichttypen bestehen.

Wie in Tabelle 1 ausgeführt, verbindet eine Self-Attention-Schicht alle Positionen mit einer konstanten Anzahl sequentiell ausgeführter Operationen, während eine rekurrente Schicht $O(n)$ sequentielle Operationen erfordert. Bezüglich der Rechenkomplexität sind Self-Attention-Schichten schneller als rekurrente Schichten, wenn die Sequenz

Länge n ist kleiner als die Repräsentationsdimensionalität d, was bei Satzrepräsentationen, die von state-of-the-art-Modellen in maschinellen Übersetzungen verwendet werden, am häufigsten der Fall ist, wie z. B. Word-Piece-[38]- und Byte-Pair-[31]-Repräsentationen. Zur Verbesserung der rechnerischen Leistung für Aufgaben mit sehr langen Sequenzen könnte die Selbstaufmerksamkeit auf die Berücksichtigung nur einer Nachbarschaft der Größe r in der Eingabesequenz beschränkt werden, die sich um die jeweilige Ausgabeposition zentriert. Dies würde die maximale Pfadlänge auf $O(n/r)$ erhöhen. Wir planen, diesen Ansatz in zukünftigen Arbeiten weiter zu untersuchen.

Eine einzelne Faltungsschicht mit Kernelbreite $k < n$ verbindet nicht alle Paare von Eingangs- und Ausgangspositionen. Dies erfordert im Fall zusammenhängender Kernel einen Stapel von $O(n/k)$ Faltungsschichten bzw. $O(log_k(n))$ im Fall dilatierter Faltungen [18], was die Länge der längsten Pfade zwischen beliebigen zwei Positionen im Netzwerk erhöht. Faltungsschichten sind im Allgemeinen um den Faktor k aufwendiger als rekurrente Schichten. Separierbare Faltungen [6] verringern die Komplexität jedoch erheblich auf $O(k \cdot n \cdot d + n \cdot d^2)$. Selbst bei $k = n$ entspricht die Komplexität einer separierbaren Faltung jedoch der Kombination aus einer Self-Attention-Schicht und einer punktweisen Feed-Forward-Schicht, dem Ansatz, den wir in unserem Modell verfolgen.

Als Nebeneffekt könnten Self-Attention-Modelle interpretierbarer sein. Wir untersuchen die Aufmerksamkeitsverteilungen unserer Modelle und stellen sowie diskutieren Beispiele im Anhang. Nicht nur lernen einzelne Attention-Heads offensichtlich, unterschiedliche Aufgaben auszuführen, viele zeigen auch Verhalten, das mit der syntaktischen und semantischen Struktur der Sätze zusammenhängt.

5 Training

Dieser Abschnitt beschreibt das Trainingsregime für unsere Modelle.

5.1 Trainingsdaten und Batching

Wir trainierten auf dem Standard-WMT-2014-Datensatz Englisch-Deutsch, der aus etwa 4,5 Millionen Satzpaaaren besteht. Die Sätze wurden mittels Byte-Pair-Encoding [3] kodiert, das ein gemeinsames Source-Target-Vokabular von etwa 37.000 Token aufweist. Für Englisch-Französisch verwendeten wir den deutlich größeren WMT-2014-Datensatz Englisch-Französisch, der aus 36 Millionen Sätzen besteht, und teilten die Token in ein Word-Piece-Vokabular mit 32.000 Einträgen auf [38]. Die Satzpaaare wurden basierend auf der ungefähren Sequenzlänge gebündelt. Jeder Trainingsbatch enthielt eine Menge von Satzpaaern mit ungefähr 25.000 Source-Tokens und 25.000 Target-Tokens.

5.2 Hardware und Zeitplan

Wir haben unsere Modelle auf einer Maschine mit 8 NVIDIA P100-GPUs trainiert. Für unsere Basismodelle unter Verwendung der im gesamten Papier beschriebenen Hyperparameter dauerte jeder Trainingsschritt etwa 0,4 Sekunden. Wir haben die Basismodelle insgesamt 100.000 Schritte oder 12 Stunden lang trainiert. Bei unseren großen Modellen (beschrieben in der letzten Zeile von Tabelle 3) betrug die Schrittzeit 1,0 Sekunde. Die großen Modelle wurden 300.000 Schritte lang trainiert (3,5 Tage).

5.3 Optimierer

Wir verwendeten den Adam-Optimierer [20] mit $\beta_1 = 0.9$, $\beta_2 = 0.98$ und $\varepsilon = 10^{-9}$. Wir variierten die Lernrate während des Trainings gemäß der Formel:

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

Dies entspricht einer linearen Erhöhung der Lernrate für die ersten warmup_steps Trainingsschritte und einer anschließenden proportionalen Verringerung entsprechend dem Kehrwert der Quadratwurzel der Schrittzahl. Wir verwendeten $warmup_steps = 4000$.

5.4 Regularisierung

Wir verwenden während des Trainings drei Arten der Regularisierung:

Tabelle 2: Der Transformer erzielt bessere BLEU-Werte als vorherige State-of-the-Art-Modelle bei den English-to-German- und English-to-French-newstest2014-Tests zu einem Bruchteil der Trainingskosten.

Modell	BLEU		Trainingskosten (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 (Basismodell)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (groß)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Residual Dropout Wir wenden Dropout [33] auf die Ausgabe jeder Sub-Ebene an, bevor diese zur Eingabe der Sub-Ebene addiert und normalisiert wird. Zusätzlich wenden wir Dropout auf die Summen der Embeddings und der Positionsencodings in beiden Encoder- und Decoder-Stapeln an. Für das Basismodell verwenden wir eine Rate von $P_{drop} = 0.1$.

Label Smoothing Während des Trainings verwendeten wir Label Smoothing mit dem Wert $\varepsilon_{ls} = 0.1$ [36]. Dies verschlechtert die Perplexität, da das Modell lernt, unsicherer zu sein, verbessert jedoch die Genauigkeit und den BLEU-Score.

6 Ergebnisse

6.1 Maschinelle Übersetzung

Auf der WMT 2014 Übersetzungsaufgabe Englisch-Deutsch übertrifft das große Transformer-Modell (Transformer (big) in Tabelle 2) die zuvor berichteten besten Modelle (einschließlich Ensembles) um mehr als 2,0 BLEU und erzielt einen neuen State-of-the-Art-BLEU-Wert von 28,4. Die Konfiguration dieses Modells ist in der untersten Zeile von Tabelle 3 aufgeführt. Das Training dauerte 3,5 Tage auf 8 P100-GPUs. Selbst unser Basismodell übertrifft alle zuvor veröffentlichten Modelle und Ensembles bei einem Bruchteil der Trainingskosten jeder konkurrierenden Modelle.

Auf der WMT 2014 Englisch-zu-Französisch-Übersetzungsaufgabe erzielt unser großes Modell einen BLEU-Score von 41,0 und übertrifft damit alle zuvor veröffentlichten Einzelmodelle bei weniger als 1/4 der Trainingskosten des vorherigen State-of-the-Art-Modells. Das für Englisch-zu-Französisch trainierte Transformer-(Groß)-Modell verwendete eine Dropout-Rate $P_{drop} = 0,1$ anstelle von 0,3.

Für die Basismodelle verwendeten wir ein einzelnes Modell, das durch Mittelung der letzten 5 Checkpoints erzeugt wurde, welche in 10-Minuten-Intervallen erstellt wurden. Für die großen Modelle mittelten wir die letzten 20 Checkpoints. Wir setzten eine Beam-Suche mit einer Beam-Größe von 4 und einer Längenstrafe $\alpha = 0,6$ [38] ein. Diese Hyperparameter wurden nach Experimenten auf dem Entwicklungssatz ausgewählt. Die maximale Ausgabelänge während der Inferenz haben wir auf Eingabelänge + 50 gesetzt, beenden den Prozess jedoch frühzeitig, wenn möglich [38].

Tabelle 2 fasst unsere Ergebnisse zusammen und vergleicht unsere Übersetzungsqualität und Trainingskosten mit anderen Modellarchitekturen aus der Literatur. Wir schätzen die Anzahl der Fließkommoperationen, die zum Training eines Modells verwendet wurden, indem wir die Trainingszeit, die Anzahl der verwendeten GPUs und eine Schätzung der anhaltenden Single-Precision-Fließkomma-Kapazität jeder GPU multiplizieren 5.

6.2 Modellvariationen

Um die Bedeutung verschiedener Komponenten des Transformers zu bewerten, variierten wir unser Basismodul auf unterschiedliche Weise und maßten die Änderungen in der Leistung bei der Englisch-Deutsch-Übersetzung an Hand

Wir verwendeten Werte von 2,8, 3,7, 6,0 und 9,5 TFLOPS für K80, K40, M40 bzw. P100.

Tabelle 3: Variationen der Transformer-Architektur. Nicht aufgeführte Werte entsprechen denen des Basismodells. Alle Metriken beziehen sich auf den englisch-deutschen Übersetzungsentwicklungssatz newstest2013. Die angegebenen Perplexitäten sind pro Wordpiece gemäß unserer Byte-Pair-Encoding und dürfen nicht mit per-Wort-Perplexitäten verglichen werden.

	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)	positionale Einbettung anstelle von Sinusoiden									4.92	25.7	
big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

Entwicklungssatz, newstest2013. Wir verwendeten Beam Search, wie in der vorherigen Abschnitt beschrieben, jedoch kein Checkpoint-Averaging. Diese Ergebnisse stellen wir in Tabelle 3 vor.

In Tabelle 3, Zeile (A), variieren wir die Anzahl der Aufmerksamkeitsköpfe sowie die Dimensionen der Aufmerksamkeits Schlüssel und Werte, wobei die Rechenmenge konstant gehalten wird, wie in Abschnitt 3.2.2 beschrieben. Während die Single-Head-Aufmerksamkeit im Vergleich zur besten Einstellung um 0.9 BLEU schlechter abschneidet, nimmt die Qualität auch bei zu vielen Köpfen ab.

In Tabelle 3, Zeile (B), stellen wir fest, dass eine Verringerung der Größe des Attention-Schlüssels d_k die Modellqualität beeinträchtigt. Dies deutet darauf hin, dass die Bestimmung der Kompatibilität nicht einfach ist und eine aufwendigere Kompatibilitätsfunktion als das Skalarprodukt vorteilhaft sein könnte. In den Zeilen (C) und (D) beobachten wir, wie erwartet, dass größere Modelle bessere Ergebnisse liefern; Dropout ist zudem sehr hilfreich, um Überanpassung zu vermeiden. In Zeile (E) ersetzen wir unsere sinusförmige Positionsencodierung durch gelernte Positions-Embeddings [9] und erhalten nahezu identische Ergebnisse im Vergleich zum Basismodell.

6.3 Englische Konstituentenanalyse

Um zu bewerten, ob der Transformer auf andere Aufgaben verallgemeinert werden kann, führten wir Experimente zur englischen Konstituentenparsung durch. Diese Aufgabe stellt spezifische Herausforderungen dar: Die Ausgabe unterliegt starken strukturellen Einschränkungen und ist deutlich länger als die Eingabe. Darüber hinaus konnten RNN-Sequenz-zu-Sequenz-Modelle in Kleindatenregimen keine state-of-the-art-Ergebnisse erzielen [37].

Wir trainierten einen 4-Schicht-Transformer mit $d_{model} = 1024$ auf dem Wall Street Journal (WSJ)-Teil des Penn Treebank [25], etwa 40.000 Trainingssätze. Wir trainierten ihn auch in einem halbüberwachten Setting, wobei wir die größeren High-Confidence- und BerkleyParser-Korpora mit etwa 17 Millionen Sätzen [37] verwendeten. Für das WSJ-nur-Setting nutzten wir ein Vokabular von 16.000 Tokens und für das halbüberwachte Setting ein Vokabular von 32.000 Tokens.

Wir führten nur eine kleine Anzahl von Experimenten durch, um das Dropout, sowohl Attention- als auch Residual-Dropout (Abschnitt 5.4), die Lernraten und die Strahlgröße auf dem Entwicklungsdatensatz aus Abschnitt 22 auszuwählen; alle anderen Parameter blieben unverändert gegenüber dem englisch-deutschen Basis-Übersetzungsmodell. Während der Inferenz haben wir

Tabelle 4: Der Transformer generalisiert gut auf die englische Konstituentenanalyse (Ergebnisse aus Abschnitt 23 des WSJ)

Parser	Training	WSJ 23 F1
Vinyals & Kaiser et al. (2014) [37]	Nur WSJ, diskriminativ	88.3
Petrov et al. (2006) [29]	Nur WSJ, diskriminativ	90.4
Zhu et al. (2013) [40]	Nur WSJ, diskriminativ	90.4
Dyer et al. (2016) [8]	Nur WSJ, diskriminativ	91.7
Transformer (4 Schichten)	Nur WSJ, diskriminativ	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 et al. (2014) [37]	semi-supervised	92.1
Transformer (4 Schichten)	semi-supervised	92.7
Luong et al. (2015) [23]	Multi-Task	93.0
Dyer et al. (2016) [8]	generativ	93.3

erhöhten die maximale Ausgabelänge auf Eingabelänge + 300. Wir verwendeten eine Strahlbreite von 21 und $\alpha = 0.3$ für beide Einstellungen, WSJ nur und semi-supervisiert.

Unsere Ergebnisse in Tabelle 4 zeigen, dass unser Modell trotz fehlender aufgaben spezifischer Anpassung überraschend gut abschneidet und bessere Ergebnisse liefert als alle zuvor berichteten Modelle mit Ausnahme des Rekurrenten Neuronalen Netzwerkgrammatik [8].

Im Gegensatz zu RNN-Sequenz-zu-Sequenz-Modellen [37] übertrifft der Transformer den Berkeley-Parser [29], selbst wenn nur auf dem WSJ-Trainingsdatensatz mit 40.000 Sätzen trainiert wird.

7 Schlussfolgerung

In dieser Arbeit stellen wir den Transformer vor, das erste Sequenztransduktionsmodell, das ausschließlich auf Aufmerksamkeitsmechanismen (Attention) basiert und die rekurrenten Schichten, die am häufigsten in Encoder-Decoder-Architekturen verwendet werden, durch mehrköpfige Selbst-Aufmerksamkeit ersetzt.

Für Übersetzungsaufgaben lässt sich der Transformer deutlich schneller trainieren als Architekturen, die auf rekurrenten oder konvolutionellen Schichten basieren. Bei beiden WMT-2014-Aufgaben (Englisch-Deutsch und Englisch-Französisch) erreichen wir einen neuen State of the Art. Im ersten Fall übertrifft unser bestes Modell sogar alle zuvor berichteten Ensembles.

Wir sind begeistert von der Zukunft auf Attention-basierten Modellen und planen, sie auf weitere Aufgaben anzuwenden. Wir beabsichtigen, den Transformer auf Probleme mit Eingangs- und Ausgangsmodalitäten jenseits von Text auszudehnen sowie lokale, eingeschränkte Attention-Mechanismen zu erforschen, um große Eingaben und Ausgaben wie Bilder, Audio und Video effizient zu verarbeiten. Die Reduktion der sequentiellen Natur der Generierung ist eines unserer weiteren Forschungsziele.

Der Code, den wir zum Trainieren und Evaluieren unserer Modelle verwendet haben, ist unter https://github.com/tensorflow/tensor2tensor verfügbar.

Dankesbekundung Wir danken Nal Kalchbrenner und Stephan Gouws für ihre fruchtbaren Kommentare, Korrekturen und Anregungen.

Literaturverzeichnis

[1] Jimmy Lei Ba, Jamie Ryan Kiros und Geoffrey E. Hinton. Layer-Normalisierung. arXiv-Preprint arXiv:1607.06450, 2016.

[2] Dzmitry Bahdanau, Kyunghyun Cho, und Yoshua Bengio. Neural maschinelle Übersetzung durch gemeinsames Lernen der Ausrichtung und Übersetzung. CoRR, abs/1409.0473, 2014.

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

[4] Jianpeng Cheng, Li Dong und Mirella Lapata. Long-Short-Term-Memory-Netzwerke für maschinelles Lesen. arXiv Preprint arXiv:1601.06733, 2016.

[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk und Yoshua Bengio. Lernen von Phrasenrepräsentationen unter Verwendung eines RNN-Encoder-Decoder für statistische maschinelle Übersetzung. CoRR, abs/1406.1078, 2014.

[6] Francois Chollet. Xception: Deep Learning mit depthwise separierbaren Faltungen. arXiv preprint arXiv:1610.02357, 2016.

[7] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, und Yoshua Bengio. Empirische Bewertung von gateden rekurrenten neuronalen Netzen für die Sequenzmodellierung. CoRR, abs/1412.3555, 2014.

[8] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros und Noah A. Smith. Rekurrente Neuralnetz-Grammatiken. In Proc. of NAACL, 2016.

[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, und Yann N. Dauphin. Konvolutionelles Sequenz-zu-Sequenz-Lernen. arXiv preprint arXiv:1705.03122v2, 2017.

[10] Alex Graves. Erzeugung von Sequenzen mit rekurrenten neuronalen Netzen. arXiv preprint arXiv:1308.0850, 2013.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, und Jian Sun. Tiefes residuelles Lernen für die Bilderkennung. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Seiten 770 – 778, 2016.

[12] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi und Jürgen Schmidhuber. Gradientenfluss in rekurrenten Netzen: die Schwierigkeit des Lernens langfristiger Abhängigkeiten, 2001.

[13] Sepp Hochreiter und Jürgen Schmidhuber. Long Short Term Memory. Neural computation, 9(8):1735 – 1780, 1997.

[14] Zhongqiang Huang und Mary Harper. Selbsttrainierende PCFG-Grammatiken mit latenten Annotationen über Sprachen hinweg. In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, Seiten 832 – 841. ACL, August 2009.

[15] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer und Yonghui Wu. Erforschung der Grenzen der Sprachmodellierung. arXiv Preprint arXiv:1602.02410, 2016.

[16] Łukasz Kaiser und Samy Bengio. Kann aktives Gedächtnis Aufmerksamkeit ersetzen? In Advances in Neural Information Processing Systems, (NIPS), 2016.

[17] Łukasz Kaiser und Ilya Sutskever. Neuronale GPUs lernen Algorithmen. In International Conference on Learning Representations (ICLR), 2016.

[18] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, und Koray Kavukcuoglu. Neuralmaschinelle Übersetzung in linearer Zeit. arXiv Preprint arXiv:1610.10099v2, 2017.

[19] Yoon Kim, Carl Denton, Luong Hoang und Alexander M. Rush. Strukturierte Aufmerksamkeitsnetzwerke. In International Conference on Learning Representations, 2017.

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

[21] Oleksii Kuchaiev und Boris Ginsburg. Faktorisierungstricks für LSTM-Netzwerke. arXiv Preprint arXiv:1703.10722, 2017.

[22] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, und Yoshua Bengio. Eine strukturierte selbstaufmerksame Satzeinbettung. arXiv Preprint arXiv:1703.03130, 2017.

[23] Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals und Lukasz Kaiser. Multi-Task-Sequenz-zu-Sequenz-Lernen. arXiv Preprint arXiv:1511.06114, 2015.

[24] Minh-Thang Luong, Hieu Pham, und Christopher D. Manning. Effektive Ansätze für die auf Aufmerksamkeit basierende neuronale maschinelle Übersetzung. arXiv-Preprint arXiv:1508.04025, 2015.

[25] Mitchell P. Marcus, Mary Ann Marcinkiewicz und Beatrice Santorini. Aufbau eines großen annotierten Korpus des Englischen: Der Penn Treebank. Computational Linguistics, 19(2):313 – 330, 1993.

[26] David McClosky, Eugene Charniak und Mark Johnson. Effektives Selbsttraining für die Parserung. In: Proceedings of the Human Language Technology Conference of the NAACL, Main Conference, Seiten 152 – 159. ACL, Juni 2006.

[27] Ankur Parikh, Oscar Täckström, Dipanjan Das und Jakob Uszkoreit. Ein zerlegbares Aufmerksamkeitsmodell. In Empirical Methods in Natural Language Processing, 2016.

[28] Romain Paulus, Caiming Xiong und Richard Socher. Ein tiefenverstärkendes Modell für abstraktive Zusammenfassung. arXiv preprint arXiv:1705.04304, 2017.

[29] Slav Petrov, Leon Barrett, Romain Thibaux und Dan Klein. Lernen präziser, kompakter und interpretierbarer Baumannotationen. In Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, Seiten 433 – 440. ACL, Juli 2006.

[30] Ofir Press und Lior Wolf. Nutzung der Ausgabe-Embedding zur Verbesserung von Sprachmodellen. arXiv preprint arXiv:1608.05859, 2016.

[31] Rico Sennrich, Barry Haddow und Alexandra Birch. Neuronale maschinelle Übersetzung seltener Wörter mit Subwort-Einheiten. arXiv Preprint arXiv:1508.07909, 2015.

[32] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton und Jeff Dean. Ausgezeichnet große neuronale Netze: Die sparsam gategesteuerte Mixture-of-Experts-Schicht. arXiv preprint arXiv:1701.06538, 2017.

[33] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever und Ruslan Salakhutdinov. Dropout: ein einfacher Weg, um neuronale Netze vor Überanpassung zu schützen. Journal of Machine Learning Research, 15(1):1929 – 1958, 2014.

[34] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston und Rob Fergus. End-zu-End-Gedächtnisnetzwerke. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama und R. Garnett, Herausgeber, Advances in Neural Information Processing Systems 28, Seiten 2440 – 2448. Curran Associates, Inc., 2015.

[35] Ilya Sutskever, Oriol Vinyals, und Quoc VV Le. Sequenzielles Lernen mit neuronalen Netzen. In Advances in Neural Information Processing Systems, Seiten 3104 – 3112, 2014.

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

[37] Vinyals & Kaiser, Koo, Petrov, Sutskever und Hinton. Grammatik als Fremdsprache. 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. Googles neuronales maschinelle Übersetzungssystem: Die Lücke zwischen menschlicher und maschineller Übersetzung schließen. arXiv Preprint arXiv:1609.08144, 2016.

[39] Jie Zhou, Ying Cao, Xuguang Wang, Peng Li und Wei Xu. Tiefe rekurrente Modelle mit Vorwärtsschleifen für die neuronale maschinelle Übersetzung. CoRR, abs/1606.04199, 2016.

[40] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang und Jingbo Zhu. Schnelle und genaue shift-reduce-Konstituentenparsung. In Proceedings of the 51st Annual Meeting of the ACL (Volume 1: Long Papers), Seiten 434 – 443. ACL, August 2013.

Aufmerksamkeitsvisualisierungen

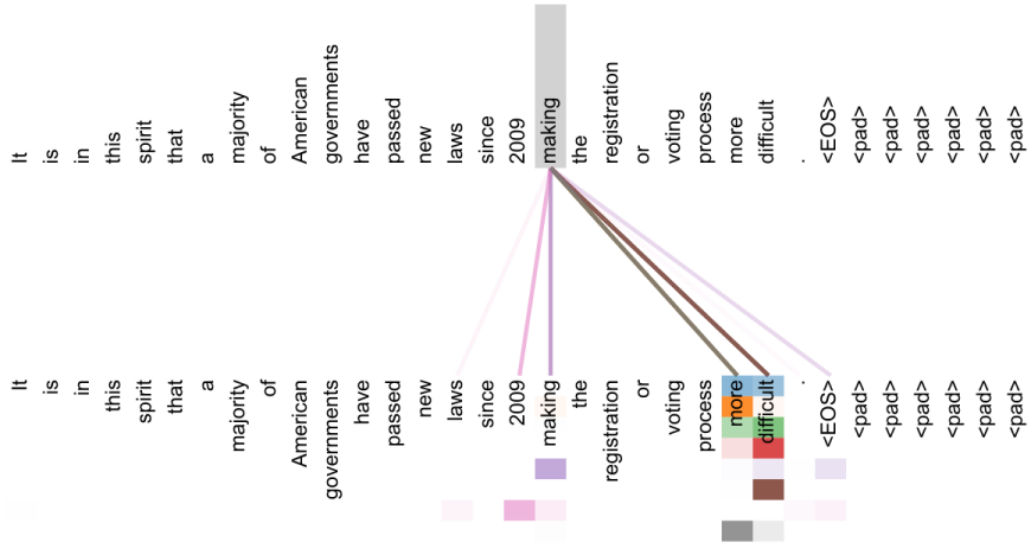


Abbildung 3: Ein Beispiel für den Aufmerksamkeitsmechanismus, der langstreckige Abhängigkeiten in der Encoder-Selbstaufmerksamkeit in Schicht 5 von 6 abbildet. Viele der Aufmerksamkeitsköpfe beziehen sich auf eine entfernte Abhängigkeit des Verbs making ‘ und vervollständigen damit den Ausdruck making...more difficult ‘. Die hier gezeigten Aufmerksamkeitszuordnungen betreffen ausschließlich das Wort making ‘. Unterschiedliche Farben repräsentieren unterschiedliche Köpfe. Beste Darstellung in Farbe.

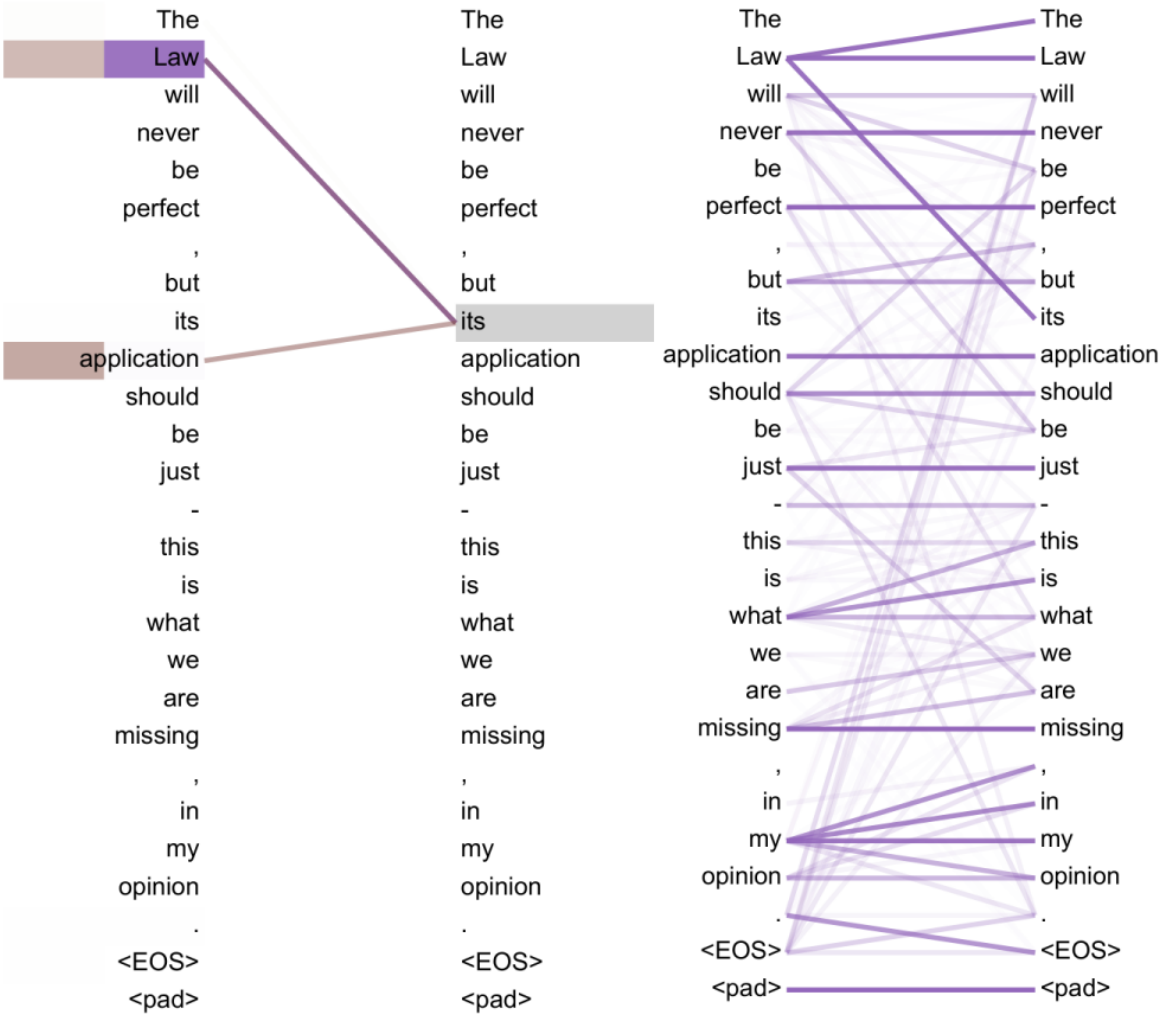


Abbildung 4: Zwei Aufmerksamkeitsköpfe, ebenfalls in Schicht 5 von 6, scheinbar an der Auflösung einer Anapher beteiligt. Oben: Full Attentions für Kopf 5. Unten: Isolierte Aufmerksamkeiten nur für das Wort 'its' für die Aufmerksamkeitsköpfe 5 und 6. Beachten Sie, dass die Aufmerksamkeiten für dieses Wort sehr scharf ausgeprägt sind.

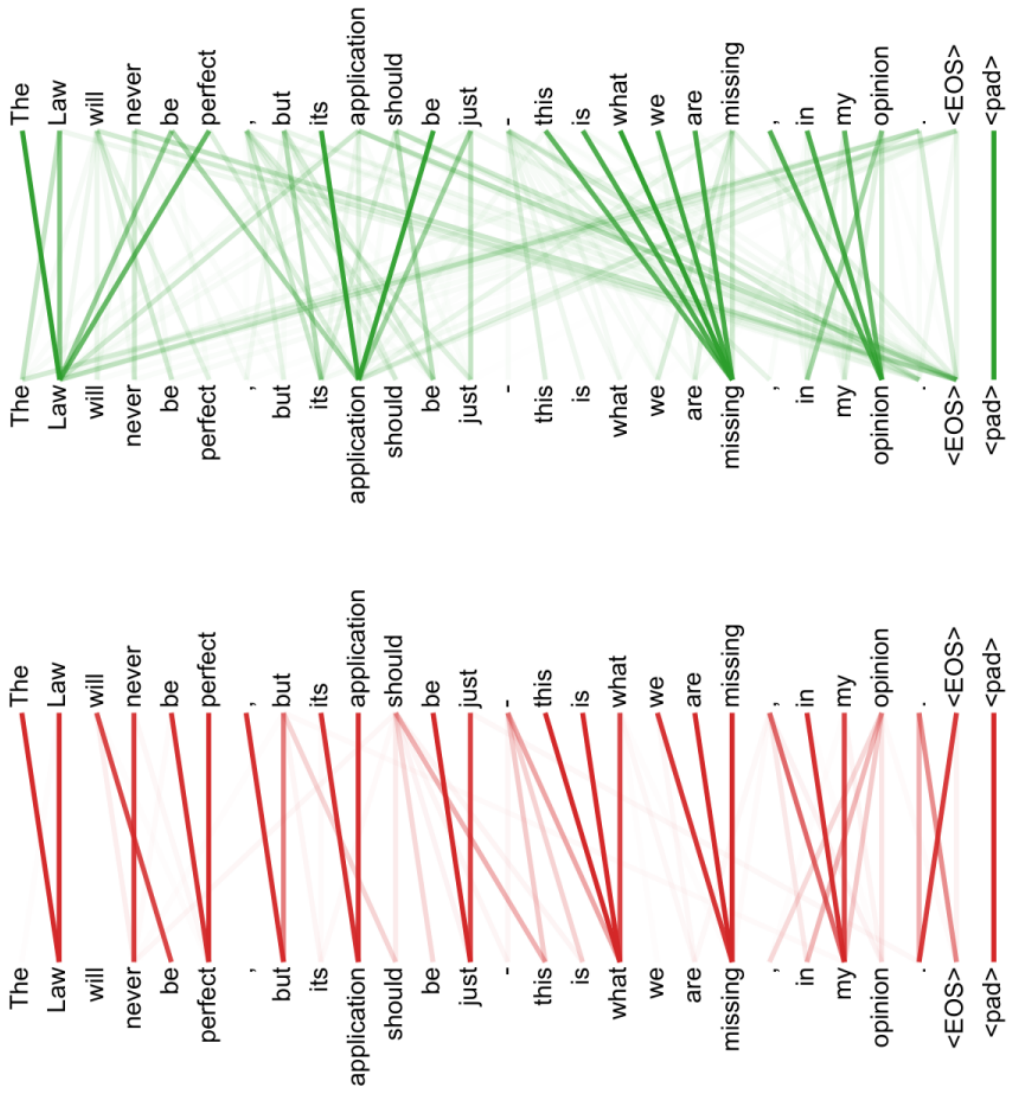


Abbildung 5: Viele der Attention-Head zeigen ein Verhalten, das mit der Struktur des Satzes zusammenzuhängen scheint. Wir haben zwei solche Beispiele oben gezeigt, und zwar zwei verschiedene Head aus der Encoder-Selbst-Aufmerksamkeit in Schicht 5 von 6. Die Head haben offensichtlich gelernt, unterschiedliche Aufgaben auszuführen.