

Unter der Voraussetzung, dass eine ordnungsgemäße Quellenangabe angegeben wird, erteilt Google hiermit die Erlaubnis, die Tabellen und Abbildungen in diesem Artikel ausschließlich für die Verwendung in journalistischen oder wissenschaftlichen

Arbeiten zu vervielfältigen.

Aufmerksamkeit ist alles, was Sie brauchen

Ashish Vaswani Google Gehirn avaswani@google.com	Noam Shazeer Google Gehirn noam@google.com	Niki Parmar Google-Recherche nikip@google.com	Jakob Uskoreit Google-Recherche usz@google.com
Llion Jones Google-Recherche llion@google.com	† Aidan N. Gomez Universität von Toronto aidan@cs.toronto.edu	Łukasz Kaiser Google Gehirn lukaszkaizer@google.com	
	Illia Polosukhin ‡ illia.polosukhin@gmail.com		

Abstrakt

Die dominanten Sequenztransduktionsmodelle basieren auf komplexen rekurrenten oder konvolutionalen neuronalen Netzen, die einen Encoder und einen Decoder umfassen. Die leistungsstärksten Modelle verbinden den Encoder und den Decoder auch über einen Aufmerksamkeitsmechanismus. Wir schlagen eine neue, einfache Netzwerkarchitektur, den Transformer, vor, die ausschließlich auf Aufmerksamkeitsmechanismen basiert und vollständig auf Wiederholungen und Faltungen verzichtet. Experimente mit zwei maschinellen Übersetzungsaufgaben zeigen, dass diese Modelle qualitativ überlegen sind, gleichzeitig parallelisierbarer sind und deutlich weniger Zeit für das Training benötigen. Unser Modell erreicht bei der Übersetzung WMT 2014 vom Englischen ins Deutsche eine BLEU von 28,4 und verbessert damit die bisherigen besten Ergebnisse, einschließlich Ensembles, um mehr als 2 BLEU. Bei der WMT 2014-Übersetzungsaufgabe vom Englischen ins Französische erreicht unser Modell nach 3,5-tägigem Training auf acht GPUs einen neuen BLEU-Wert von 41,8 für ein einzelnes Modell, was 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 Parsen englischer Wahlkreise sowohl mit großen als auch mit begrenzten Trainingsdaten anwenden.

Gleicher Beitrag. Die Reihenfolge der Auflistung ist zufällig. Jakob schlug vor, RNNs durch Selbstaufmerksamkeit zu ersetzen und begann mit den Bemühungen, diese Idee zu evaluieren. Ashish entwarf und implementierte zusammen mit Illia die ersten Transformer-Modelle und war maßgeblich an jedem Aspekt dieser Arbeit beteiligt. Noam schlug die skalierte Punktproduktaufmerksamkeit, die Mehrkopfaufmerksamkeit und die parameterfreie Positionsdarstellung vor und wurde zur anderen Person, die an fast jedem Detail beteiligt war. Niki entwarf, implementierte, stimmte und evaluierte unzählige Modellvarianten in unserer ursprünglichen Codebasis und tensor2tensor. Llion experimentierte auch mit neuartigen Modellvarianten, war verantwortlich für unsere anfängliche Codebasis und effiziente Inferenz und Visualisierungen. Lukasz und Aidan verbrachten unzählige lange Tage damit, verschiedene Teile von tensor2tensor zu entwerfen und zu implementieren, unsere frühere Codebasis zu ersetzen, die Ergebnisse erheblich zu verbessern und unsere Forschung massiv zu beschleunigen.

† Arbeit, die während meiner Zeit bei Google Brain durchgeführt wurde.

‡ Arbeit, die während der Zeit bei Google Research durchgeführt wurde.

1 Einleitung

Insbesondere rekurrente neuronale Netze, das Long Shortterm Memory [13] und Gated Recurrent [7] Neuronale Netze haben sich als State-of-the-Art-Ansätze in der Sequenzmodellierung und Transduktionsprobleme wie der Sprachmodellierung und maschinellen Übersetzung fest etabliert [35, 2, 5]. Zahlreiche Bemühungen haben seither die Grenzen rekurrenter Sprachmodelle und Encoder-Decoder-Architekturen weiter verschoben [38, 24, 15].

Wiederkehrende Modelle faktorisieren die Berechnung in der Regel entlang der Symbolpositionen der Eingabe- und Ausgabesequenzen. Indem sie die Positionen an Schritten in der Rechenzeit ausrichten, erzeugen sie eine Abfolge von verborgenen Zuständen h_t als Funktion des vorherigen verborgenen Zustands h_{t-1} und der Eingabe für die Position t . Diese inhärent sequenzielle Natur schließt die Parallelisierung innerhalb von Trainingsbeispielen aus, was bei längeren Sequenzlängen kritisch wird, da Speicherbeschränkungen die Batchverarbeitung über Beispiele hinweg einschränken. Neuere Arbeiten haben signifikante Verbesserungen der Recheneffizienz durch Faktorisierungstricks [21] und bedingte Berechnung [32] erzielt, während sie auch die Modelleleistung im letzteren Fall verbessert haben. Die grundlegende Einschränkung der sequentiellen Berechnung bleibt jedoch bestehen.

Aufmerksamkeitsmechanismen sind zu einem integralen Bestandteil zwingender Sequenzmodellierungs- und Transduktionsmodelle in verschiedenen Aufgaben geworden, die es ermöglichen, Abhängigkeiten ohne Rücksicht auf ihre Entfernung in den Ein- oder Ausgabesequenzen zu modellieren [2, 19]. Bis auf wenige Ausnahmen [27] werden solche Aufmerksamkeitsmechanismen jedoch in Verbindung mit einem rekurrenten Netzwerk verwendet.

In dieser Arbeit schlagen wir den Transformer vor, eine Modellarchitektur, die auf Wiederholungen verzichtet und sich stattdessen vollständig auf einen Aufmerksamkeitsmechanismus verlässt, um globale Abhängigkeiten zwischen Eingabe und Ausgabe zu zeichnen. Der Transformer ermöglicht eine deutlich höhere Parallelisierung und kann nach nur zwölfstündigem Training auf acht P100-GPUs einen neuen Stand der Technik in der Übersetzungsqualität erreichen.

2 Hintergrund

Das Ziel, sequentielle Berechnungen zu reduzieren, bildet auch die Grundlage für die Extended Neural GPU [16], ByteNet [18] und ConvS2S [9], die alle Convolutional Neural Networks als Grundbaustein verwenden und versteckte Darstellungen parallel für alle Ein- und Ausgabepositionen berechnen. In diesen Modellen wächst die Anzahl der Operationen, die erforderlich sind, um Signale von zwei beliebigen Ein- oder Ausgangspositionen in Beziehung zu setzen, im Abstand zwischen den Positionen, linear für ConvS2S und logarithmisch für ByteNet. Dies erschwert das Erlernen von Abhängigkeiten zwischen entfernten Positionen [12]. Im Transformer reduziert sich dies auf eine konstante Anzahl von Operationen, allerdings auf Kosten einer reduzierten effektiven Auflösung durch die Mittelung aufmerksamkeitsgewichteter Positionen, einem Effekt, dem wir mit Multi-Head Attention entgegenwirken, wie in Abschnitt 3.2 beschrieben.

Selbstaufmerksamkeit, manchmal auch Intra-Aufmerksamkeit genannt, ist ein Aufmerksamkeitsmechanismus, der verschiedene Positionen einer einzelnen Sequenz miteinander in Beziehung setzt, um eine Repräsentation der Sequenz zu berechnen. Selbstaufmerksamkeit wurde erfolgreich in einer Vielzahl von Aufgaben eingesetzt, darunter Leseverständnis, abstrakte Zusammenfassung, textuelle Schlussfolgerung und das Erlernen aufgabenunabhängiger Satzdarstellungen [4, 27, 28, 22].

End-to-End-Speichernetzwerke basieren auf einem rekurrenten Aufmerksamkeitsmechanismus anstelle einer sequenzorientierten Wiederholung und haben sich bei einfachsprachigen Fragebeantwortungs- und Sprachmodellierungsaufgaben als gut erwiesen [34].

Nach unserem besten Wissen ist der Transformer jedoch das erste Transduktionsmodell, das sich ausschließlich auf Selbstaufmerksamkeit verlässt, um Repräsentationen seiner Ein- und Ausgabe zu berechnen, ohne sequenzorientierte RNNs oder Faltung zu verwenden. In den folgenden Abschnitten werden wir den Transformer beschreiben, zur Selbstaufmerksamkeit motivieren und seine Vorteile gegenüber Modellen wie [17, 18] und [9] diskutieren.

3 Modell-Architektur

Die meisten kompetitiven neuronalen Sequenztransduktionsmodelle haben eine Encoder-Decoder-Struktur [5, 2, 35]. Dabei bildet der Encoder eine Eingabesequenz von Symboldarstellungen (x^1, \dots, x^n) auf eine Folge von kontinuierlichen Darstellungen $z = (z^1, \dots, z^n)$ ab. Bei gegebenem z erzeugt der Decoder dann eine Ausgabesequenz (y^1, \dots, y_m) von Symbolen, Element für Element. Bei jedem Schritt ist das Modell autoregressiv [10] und verbraucht die zuvor generierten Symbole als zusätzliche Eingabe bei der Generierung des nächsten Symbols.

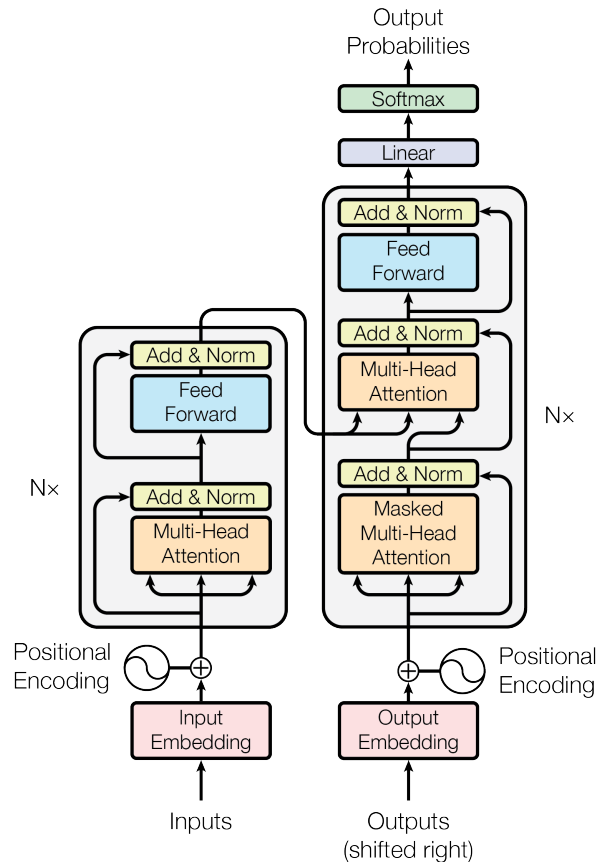


Abbildung 1: Der Transformer - Modellarchitektur.

Der Transformer folgt dieser Gesamtarchitektur mit gestapelter Selbstaufmerksamkeit und punktwisen, vollständig verbundenen 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-Stacks

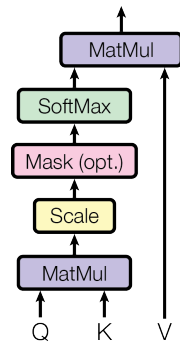
Encoder: Der Encoder besteht aus einem Stapel von $N = 6$ identischen Schichten. Jede Schicht hat zwei Unterschichten. Das erste ist ein Selbstbeobachtungsmechanismus mit mehreren Köpfen, und das zweite ist ein einfaches, positionsbezogenes, vollständig verbundenes Feed-Forward-Netzwerk. Wir verwenden eine Restverbindung [11] um jede der beiden Unterschichten, gefolgt von einer Schichtnormalisierung [1]. Das heißt, die Ausgabe jeder Unterschicht ist $\text{LayerNorm}(x + \text{Sublayer}(x))$, wobei $\text{Sublayer}(x)$ die Funktion ist, die von der Unterschicht selbst implementiert wird. Um diese Restverbindungen zu erleichtern, erzeugen alle Unterschichten im Modell sowie die Einbettungsschichten Ausgaben der Dimension $d_{\text{model}} = 512$.

Decoder: Der Decoder besteht ebenfalls aus einem Stapel von $N = 6$ identischen Lagen. Zusätzlich zu den beiden Unterlagen in jeder Geberschicht fügt der Decoder eine dritte Unterschicht ein, die eine Mehrkopfaufmerksamkeit über den Ausgang des Geberstapels ausführt. Ähnlich wie beim Encoder verwenden wir Restverbindungen um jede der Subschichten, gefolgt von einer Schichtnormalisierung. Wir modifizieren auch die Selbstaufmerksamkeits-Unterschicht im Decoder-Stack, um zu verhindern, dass Positionen auf nachfolgende Positionen reagieren. Diese Maskierung, kombiniert mit der Tatsache, dass die Ausgabeembeddings um eine Position versetzt sind, stellt sicher, dass die Vorhersagen für die Position i nur von den bekannten Ausgaben an Positionen kleiner als i abhängen können.

3.2 Achtung

Eine Aufmerksamkeitsfunktion kann so beschrieben werden, dass eine Abfrage und ein Satz von Schlüssel-Wert-Paaren einer Ausgabe zugeordnet werden, wobei die Abfrage, die Schlüssel, die Werte und die Ausgabe alle Vektoren sind. Die Ausgabe wird als gewichtete Summe berechnet: Skaliertes

Punkt-Produkt: Achtung



Mehrköpfige Aufmerksamkeit

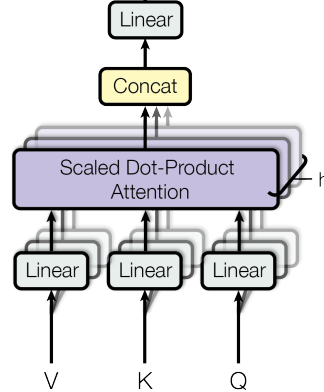


Abbildung 2: (links) Skalierte Punkt-Produkt-Aufmerksamkeit. (rechts) Multi-Head Attention besteht aus mehreren parallel laufenden Aufmerksamkeitsschichten.

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

3.2.1 Skalierte Punktproduktaufmerksamkeit

Unsere besondere Aufmerksamkeit nennen wir "Scaled Dot-Product Attention" (Abbildung 2). Die Eingabe besteht aus Abfragen und Schlüsseln der Dimension d_k und $\sqrt{d_v}$ Werten der Dimension d_v . Wir berechnen die Punktprodukte der Abfrage mit allen Schlüsseln, dividieren jeden durch d_k und wenden eine Softmax-Funktion an, um die Gewichtungen der Werte zu erhalten.

In der Praxis berechnen wir die Aufmerksamkeitsfunktion für eine Reihe von Abfragen gleichzeitig, die in einer Matrix Q zusammengefasst sind. Die Schlüssel und Werte werden ebenfalls zusammen in die Matrizen K und V gepackt. Wir berechnen die Matrix der Ausgaben wie folgt:

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

Die beiden am häufigsten verwendeten Aufmerksamkeitsfunktionen sind die additive Aufmerksamkeit [2] und die Punktprodukt-Aufmerksamkeit (multiplikativ). Die Punktproduktaufmerksamkeit ist identisch mit unserem Algorithmus, mit Ausnahme des Skalierungsfaktors von $\sqrt{d_k}$. Die additive Aufmerksamkeit berechnet die Kompatibilitätsfunktion unter Verwendung eines Feed-Forward-Netzwerks mit

D_k

Eine einzelne verborgene Schicht. Während sich die beiden in ihrer theoretischen Komplexität ähneln, ist die Punktproduktaufmerksamkeit in der Praxis viel schneller und platzsparender, da sie mit hochoptimiertem Matrixmultiplikationscode implementiert werden kann.

Während bei kleinen Werten von d_k die beiden Mechanismen ähnlich funktionieren, übertrifft die additive Aufmerksamkeit bei größeren Werten von d_k die Aufmerksamkeit des Punktprodukts ohne Skalierung [3]. Wir vermuten, dass bei großen Werten von d_k die Punktprodukte groß werden, wodurch die Softmax-Funktion in Bereiche verschoben wird, in denen sie extrem kleine Gradienten aufweist⁴. Um diesem Effekt entgegenzuwirken, skalieren wir die dot-Produkte um $\frac{1}{\sqrt{d_k}}$.

3.2.2 Mehrköpfige Aufmerksamkeit

Anstatt eine einzelne Aufmerksamkeitsfunktion mit d_{model} -dimensionalen Schlüsseln, Werten und Abfragen auszuführen, fanden wir es vorteilhaft, die Abfragen, Schlüssel und Werte h -mal linear mit unterschiedlichen, gelernten linearen Projektionen auf d_k -, d_k - bzw. d_v -Dimensionen zu projizieren. Für jede dieser projizierten Versionen von Abfragen, Schlüsseln und Werten führen wir dann die Aufmerksamkeitsfunktion parallel aus, was zu d_v -dimensionalen

⁴ Um zu veranschaulichen, warum die Punktprodukte groß werden, nehmen wir an, dass die Komponenten von q und k unabhängige Zufallsvariablen mit dem Mittelwert 0 und der Varianz 1 sind. Dann wird ihr Punktprodukt $q \cdot k = \sum_{i=1}^n q_i k_i$, hat den Mittelwert 0 und die Varianz d_k .

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

Die Mehrkopfaufmerksamkeit ermöglicht es dem Modell, sich gemeinsam um Informationen aus verschiedenen Darstellungsunterräumen an verschiedenen Positionen zu kümmern. Mit einem einzigen Aufmerksamkeitskopf wird dies durch die Mittelwertbildung verhindert.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{Kopf}^1, \dots, \text{Kopf}^h) W^O$$

$$Q, K, V \text{ wobei } \text{Kopf} = \text{Achtung}(QW_{i,i}^Q, KW_{i,i}^K, VW_{i,i}^V)$$

Wobei es sich bei den Projektionen um Parametermatrizen $W^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ und $W^O \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$ handelt.

In dieser Arbeit verwenden wir $h = 8$ parallele Aufmerksamkeitschichten oder Köpfe. Für jede dieser Methoden verwenden wir $d_k = d_v = d_{\text{model}} / h = 64$. Aufgrund der reduzierten Abmessung jedes Kopfes ist der Gesamtrechenaufwand ähnlich wie bei der Einzelkopfaufmerksamkeit bei voller Dimensionalität.

3.2.3 Anwendungen von Aufmerksamkeit in unserem Modell

Der Transformer nutzt die Multi-Head-Aufmerksamkeit auf drei verschiedene Arten:

- In "Encoder-Decoder-Achtung"-Schichten stammen die Abfragen aus der vorherigen Decoder-Schicht, und die Speicherschlüssel und -werte stammen aus der Ausgabe des Encoders. Dadurch kann jede Position im Decoder über alle Positionen in der Eingabesequenz betrachtet werden. Dies ahmt die typischen Encoder-Decoder-Aufmerksamkeitsmechanismen in Sequenz-zu-Sequenz-Modellen wie [38, 2, 9] nach.
- Der Encoder enthält Self-Attention-Layer. In einer Self-Attention-Schicht stammen alle Schlüssel, Werte und Abfragen von derselben Stelle, in diesem Fall von der Ausgabe der vorherigen Schicht im Encoder. Jede Position im Encoder kann alle Positionen in der vorherigen Schicht des Encoders berücksichtigen.
- In ähnlicher Weise ermöglichen Selbstwahrnehmungsschichten im Decoder, dass jede Position im Decoder alle Positionen im Decoder bis einschließlich dieser Position berücksichtigt. Wir müssen den Informationsfluss nach links im Decoder verhindern, um die autoregressive Eigenschaft zu erhalten. Wir setzen dies innerhalb der skalierten Punktproduktaufmerksamkeit um, indem wir alle Werte in der Eingabe des Softmax, die illegalen Verbindungen entsprechen, maskieren (auf $-\infty$ setzen). Siehe Abbildung 2.

3.3 Positionsweise Feed-Forward-Netzwerke

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

$$\text{FFN}(x) = \max(0, xW^1 + b^1)W^2 + b^2 \quad (2)$$

Während die linearen Transformationen an verschiedenen Positionen gleich sind, verwenden sie von Ebene zu Ebene unterschiedliche Parameter. Eine andere Möglichkeit, dies zu beschreiben, ist als zwei Faltungen mit der Korngröße 1. Die Dimensionalität der Ein- und Ausgabe beträgt $d_{\text{model}} = 512$, und die innere Schicht hat die Dimensionalität $d_{\text{ff}} = 2048$.

3.4 Einbettungen und Softmax

Ähnlich wie bei anderen Sequenztransduktionsmodellen verwenden wir gelernte Einbettungen, um die Eingabe- und Ausgabetoken in Vektoren der Dimension d_{model} umzuwandeln. Wir verwenden auch die übliche erlernte lineare Transformation und Softmax-Funktion, um den Decoderausgang in vorhergesagte Wahrscheinlichkeiten des nächsten Tokens umzuwandeln. In unserem Modell teilen wir uns die gleiche Gewichtungsmatrix zwischen den beiden Einbettungsschichten und der prä-softmax linearen Transformation, ähnlich wie bei [30]. In den Einbettungsschichten multiplizieren wir diese Gewichte mit d_{model} . Tabelle 1:

Maximale Pfadlängen, Komplexität pro Schicht und minimale Anzahl sequenzieller Operationen für verschiedene Layer-Typen. n ist die Sequenzlänge, d ist die Repräsentationsdimension, k ist die Kerngröße von Faltungen und r ist die Größe der Nachbarschaft in eingeschränkter Selbstaufmerksamkeit.

Layer-Typ	Komplexität pro Schicht	Sequenzielle Operationen	Maximale Pfadlänge
Selbstaufmerksamkeit	$O(n \cdot d)$	$O(1)$	$O(1)$
Wiederkehrende	$O(n \cdot d^2)$	$O(n)$	$O(n) O(\log_k n)$
Faltung	$O(k \cdot n \cdot d^2)$	$O(1)$	(n)
Selbstaufmerksamkeit (eingeschränkt)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

3.5 Positionskodierung

Da unser Modell keine Wiederholung und keine Faltung enthält, müssen wir, damit das Modell die Reihenfolge der Sequenz nutzen kann, einige Informationen über die relative oder absolute Position der Token in der Sequenz einfügen. Zu diesem Zweck fügen wir den Eingabeeinbettungen am unteren Rand der Encoder- und Decoder-Stacks "Positionskodierungen" hinzu. Die Positionskodierungen haben die gleiche Dimension d_{model} wie die Einbettungen, so dass beide summiert werden können. Es gibt viele Möglichkeiten für Positionskodierungen, gelernt und fixiert [9].

In dieser Arbeit verwenden wir Sinus- und Kosinusfunktionen unterschiedlicher Frequenzen:

$$PE = \sin(\text{pos}/10000^{2i/d_{\text{Modell}}})$$

$$PE = 2i/d_{\text{Modell}} (\text{pos}, 2i+1) \cos(\text{pos}/10000^{d_{\text{Modell}}})$$

Dabei ist POS die Position und i die Dimension. Das heißt, jede Dimension der Positionskodierung entspricht einer Sinuskurve. Die Wellenlängen bilden eine geometrische Abfolge von 2π bis $10000 \cdot 2\pi$. Wir haben diese Funktion gewählt, weil wir die Hypothese angenommen haben, dass sie es dem Modell ermöglichen würde, leicht zu lernen, durch relative Positionen zu bedienen, da für jeden festen Versatz k $PE_{\text{pos}+k}$ als lineare Funktion von PE_{pos} dargestellt werden kann.

Wir experimentierten auch mit der Verwendung von erlernten Positionseinbettungen [9] und stellten fest, dass die beiden Versionen nahezu identische Ergebnisse lieferten (siehe Tabelle 3 Zeile (E)). Wir haben uns für die sinusförmige Version entschieden, da sie es dem Modell ermöglicht, auf Sequenzlängen zu extrapolieren, die länger sind als die, die während des Trainings angetroffen werden.

4 Warum Selbstaufmerksamkeit

In diesem Abschnitt vergleichen wir verschiedene Aspekte von Self-Attention-Schichten mit den rekurrenten und konvolutiven Layern, die üblicherweise für die Abbildung einer Sequenz von Symboldarstellungen variabler Länge verwendet werden (x_1, \dots, x_n) zu einer anderen Sequenz gleicher Länge (z_1, \dots, z_n) mit $x_i, z_i \in \mathbb{R}^d$, wie z. B. einer verborgenen Schicht in einem typischen Sequenztransduktions-Encoder oder -Decoder. Um unsere Selbstaufmerksamkeit zu motivieren, betrachten wir drei Desiderate.

Eine davon ist die gesamte Rechenkomplexität pro Schicht. Ein weiterer ist die Menge an Berechnungen, die parallelisiert werden können, gemessen an der minimalen Anzahl der erforderlichen sequenziellen Operationen.

Die dritte ist die Pfadlänge zwischen weitreichenden Abhängigkeiten im Netzwerk. Das Erlernen von langreichweitigen Abhängigkeiten ist eine zentrale Herausforderung bei vielen Sequenztransduktionsaufgaben. Ein Schlüsselfaktor, der die Fähigkeit beeinflusst, solche Abhängigkeiten zu lernen, ist die Länge der Pfade, die Vorwärts- und Rückwärtssignale im Netzwerk durchlaufen müssen. Je kürzer diese Pfade zwischen einer beliebigen Kombination von Positionen in der Ein- und Ausgabesequenz sind, desto einfacher ist es, langfristige Abhängigkeiten zu erlernen [12]. Daher vergleichen wir auch die maximale Weglänge zwischen zwei beliebigen Ein- und Ausgangspositionen in Netzwerken, die aus den verschiedenen Schichttypen bestehen.

Wie in Tabelle 1 erwähnt, verbindet eine Self-Attention-Schicht alle Positionen mit einer konstanten Anzahl sequenziell ausgeführter Operationen, während eine rekurrente Schicht sequenzielle $O(n)$ -Operationen erfordert. In Bezug auf die Rechenkomplexität sind Self-Attention-Schichten schneller als rekurrente Layer, wenn die Sequenzlänge n kleiner ist als

die Repräsentationsdimensionalität d , was am häufigsten bei Satzrepräsentationen der Fall ist, die von modernen Modellen in maschinellen Übersetzungen verwendet werden, wie z. B. Wortstück- [38] und Byte-Paar- [31]-Darstellungen. Um die Rechenleistung für Aufgaben mit sehr langen Sequenzen zu verbessern, könnte die Selbstaufmerksamkeit darauf beschränkt werden, nur eine Nachbarschaft der Größe r in der Eingabesequenz zu berücksichtigen, die um die jeweilige Ausgabe positioniert ist. Dadurch würde sich die maximale Pfadlänge auf $O(n/r)$ erhöhen. Wir planen, diesen Ansatz in zukünftigen Arbeiten weiter zu untersuchen.

Eine einzelne Faltungsschicht mit der Kernbreite $k < n$ verbindet nicht alle Paare von Ein- und Ausgangspositionen. Dies erfordert einen Stapel von $O(n/k)$ - Faltungsschichten im Falle von zusammenhängenden Kernen oder $O(\log_k(n))$ im Falle von dilatierten Faltungen [18], wodurch die Länge der längsten Pfade zwischen zwei beliebigen Positionen im Netzwerk erhöht wird. Faltungsschichten sind in der Regel um den Faktor k teurer als wiederkehrende Schichten. Trennbare Faltungen [6] verringern jedoch die Komplexität erheblich, auf $O(k \cdot n \cdot d + n \cdot d^2)$. Aber selbst bei $k = n$ ist die Komplexität einer trennbaren Faltung gleich der Kombination aus einer Selbstaufmerksamkeitsschicht und einer punktwisen Feed-Forward-Schicht, dem Ansatz, den wir in unserem Modell verfolgen.

Als Nebeneffekt könnte die Selbstaufmerksamkeit zu besser interpretierbaren Modellen führen. Wir untersuchen Aufmerksamkeitsverteilungen aus unseren Modellen und präsentieren und diskutieren Beispiele im Anhang. Nicht nur, dass individuelle Aufmerksamkeitsköpfe eindeutig lernen, verschiedene Aufgaben auszuführen, viele scheinen auch ein Verhalten zu zeigen, das mit der syntaktischen und semantischen Struktur der Sätze zusammenhängt.

5 Ausbildung

In diesem Abschnitt wird das Trainingsprogramm für unsere Modelle beschrieben.

5.1 Trainingsdaten und Batchverarbeitung

Wir trainierten mit dem Standarddatensatz WMT 2014 Englisch-Deutsch, der aus etwa 4,5 Millionen Satzpaaren besteht. Die Sätze wurden mit Hilfe der Byte-Paar-Kodierung [3] kodiert, die ein gemeinsames Quell-Ziel-Vokabular von etwa 37000 Token hat. Für Englisch-Französisch haben wir den deutlich größeren WMT 2014 Englisch-Französisch-Datensatz verwendet, der aus 36 Millionen Sätzen besteht und Token in ein Vokabular von 32000 Wörtern aufteilt [38]. Die Satzpaare wurden nach ungefähre Sequenzlänge gestapelt. Jeder Trainingsbatch enthielt einen Satz von Satzpaaren, die ungefähr 25000 Quelltoken und 25000 Zieltoken enthielten.

5.2 Hardware und Zeitplan

Wir haben unsere Modelle auf einem Rechner mit 8 NVIDIA P100 GPUs trainiert. Für unsere Basismodelle, die in der Arbeit beschriebenen Hyperparameter verwendeten, dauerte jeder Trainingsschritt etwa 0,4 Sekunden. Wir trainierten die Basismodelle für insgesamt 100.000 Schritte oder 12 Stunden. Bei unseren großen Modellen (beschrieben in der unteren Zeile von Tabelle 3) betrug die Schrittzeit 1,0 Sekunden. Die großen Modelle wurden für 300.000 Schritte trainiert (3,5 Tage).

5.3 Optimierer

Wir haben den Adam-Optimierer [20] mit $\beta^1 = 0,9$, $\beta^2 = 0,98$ und $\epsilon = 10$ verwendet. Wir haben das Lerntempo im Laufe des Trainings nach folgender Formel variiert:

$$lrate = d \cdot \text{meine}(\text{step_num}, \text{step_num} \cdot \text{warmup_steps})^{(3)} \text{Modell}$$

Dies entspricht einer linearen Erhöhung der Lernrate für die ersten warmup_steps Trainingsschritten und einer anschließenden Verringerung proportional zur inversen Quadratwurzel der Schrittzahl. Wir haben $\text{warmup_steps} = 4000$ verwendet.

5.4 Regularisierung

Wir verwenden drei Arten der Regularisierung während des

Trainings: Tabelle 2: Der Transformer erzielt bessere BLEU-Ergebnisse als frühere State-of-the-Art-Modelle bei den Englisch-Deutsch- und Englisch-Französisch-newstest2014-Tests zu einem Bruchteil der Schulungskosten.

Modell	BLEU		Trainingskosten (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18] Deep-Att + PosUnk [39]	23.75	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 Set [39] GNMT + RL Set [38] ConvS2S Set [9]	26.30	40.4	$1.8 \cdot 10^{20}$	$8.0 \cdot 10^{20}$
	26.36	41.16	$1.9 \cdot 10^{21}$	$1.1 \cdot 10^{21}$ kg
		41.29	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Transformator (Basismodell)	27.3	38.1	$3.3 \cdot 10^{18}$	
Transformator (groß)	28.4	41.8	$2.3 \cdot 10^{19}$	

Residualer Dropout Wir wenden Dropout [33] auf die Ausgabe jeder Unterschicht an, bevor sie zur Sublayer-Eingabe hinzugefügt und normalisiert wird. Darüber hinaus wenden wir Dropout auf die Summen der Einbettungen und die Positionscodierungen sowohl im Encoder- als auch im Decoder-Stack an. Für das Basismodell verwenden wir einen Satz von

$P_{drop} = 0,1$.

Label-Glättung Während des Trainings wurde eine Label-Glättung des Wertes ϵ $\epsilon = 0,1$ verwendet [36]. Dies führt zu Verwirrung, da das Modell lernt, unsicherer zu sein, verbessert aber die Genauigkeit und den BLEU-Score.

6 Ergebnisse

6.1 Maschinelle Übersetzung

Bei der Übersetzungsaufgabe WMT 2014 vom Englischen ins Deutsche übertrifft das Big-Transformer-Modell (Transformer (big) in Tabelle 2) die besten zuvor berichteten Modelle (einschließlich Ensembles) um mehr als 2,0 BLEU und stellt einen neuen BLEU-Wert von 28,4 auf. Die Konfiguration dieses Modells ist in der unteren 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, und das zu einem Bruchteil der Schulungskosten aller Konkurrenzmodelle.

Bei der Übersetzungsaufgabe WMT 2014 vom Englischen ins Französische erreicht unser großes Modell einen BLEU-Wert von 41,0 und übertrifft damit alle zuvor veröffentlichten Einzelmodelle, und das bei weniger als 1/4 der Schulungskosten des vorherigen hochmodernen Modells. Das Transformer-Modell (big), das für das Englisch-Französisch-Modell trainiert wurde, verwendete die Dropout-Rate $P_{drop} = 0,1$ anstelle von 0,3.

Für die Basismodelle haben wir ein einzelnes Modell verwendet, das durch Mittelung der letzten 5 Checkpoints erhalten wurde, die in 10-Minuten-Intervallen geschrieben wurden. Für die großen Modelle haben wir den Durchschnitt der letzten 20 Checkpoints ermittelt. Wir verwendeten die Strahlsuche mit einer Strahlgröße von 4 und einer Längenstrafe von $\alpha = 0,6$ [38]. Diese Hyperparameter wurden nach Experimenten mit dem Entwicklungssatz ausgewählt. Wir setzen die maximale Ausgabelänge während der Inferenz auf Eingabelänge + 50, beenden sie aber vorzeitig, 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 Gleitkommaoperationen, die zum Trainieren eines Modells verwendet werden, indem wir die Trainingszeit, die Anzahl der verwendeten GPUs und eine Schätzung der anhaltenden

⁵ Gleitkommakapazität jeder GPU mit einfacher Genauigkeit .

6.2 Modellvarianten

Um die Bedeutung der verschiedenen Komponenten des Transformers zu bewerten, haben wir unser Basismodell auf unterschiedliche Weise variiert, indem wir die Leistungsänderung bei der Übersetzung vom Englischen ins Deutsche auf dem

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

Tabelle 3: Variationen der Transformer-Architektur. Nicht aufgelistete Werte sind identisch mit denen des Basismodells. Alle Metriken beziehen sich auf das Englisch-Deutsch-Übersetzungsentwicklungsset newstest2013. Die aufgelisteten Verwirrungen beziehen sich auf ein Wortstück, entsprechend unserer Byte-Paar-Codierung, und sollten nicht mit Verwirrungen pro Wort verglichen werden.

	Auf der anderen Seite ist das Modell des T-Shirts	Zug Schritte	PPL BLEU params 6 (dev) (dev) ×10
Basis	6 512 2048 8 64 64 0,1 0,1 100 Tsd.		4.92 25.8 65
(ein)	1 512 512 4 128 128 16 32 32 32 16 16		5.29 24.9 5.00 25.5 4.91 25.8 5.01 25.4
(B)	16 32		5.16 25.1 58 5.01 25.4 60
(C)	2 4 8 256 32 32 1024 128 128 1024 4096		6.11 23.7 36 5.19 25.3 50 4.88 25.5 80 5.75 24.5 28 4.66 26.0 168 5.12 25.4 53 4.75 26.2 90
(D)	0.0 0.2 0.0 0.2		5.77 24.6 4.95 25.5 4.67 25.3 5.47 25.7
(E)	Positionseinbettung statt Sinuskurven		4.92 25.7
groß	6 1024 4096 16 0,3 300K		4,33 26,4 213 kg

Entwicklungsset, NewsTest2013. Wir haben die Beam-Suche verwendet, wie im vorherigen Abschnitt beschrieben, aber keine Prüfpunkt-Mittelwertbildung. Diese Ergebnisse stellen wir in Tabelle 3 vor.

In den Zeilen (A) von Tabelle 3 variieren wir die Anzahl der Aufmerksamkeitsköpfe und die Aufmerksamkeits Schlüssel- und Wertdimensionen, wobei der Rechenaufwand konstant bleibt, wie in Abschnitt 3.2.2 beschrieben. Während die Aufmerksamkeit mit einem Kopf 0,9 BLEU schlechter ist als die beste Einstellung, sinkt auch die Qualität, wenn zu viele Köpfe vorhanden sind.

In Tabelle 3 Zeilen (B) stellen wir fest, dass die Reduzierung der Größe des Aufmerksamkeits Schlüssels die Modellqualität beeinträchtigt. Dies deutet darauf hin, dass die Bestimmung der Kompatibilität nicht einfach ist und dass eine ausgefeiltere Kompatibilitätsfunktion als das Punktprodukt von Vorteil sein kann. Des Weiteren beobachten wir in den Reihen (C) und (D), dass erwartungsgemäß größere Modelle besser sind und Dropout sehr hilfreich ist, um eine Überanpassung zu vermeiden. In Zeile (E) ersetzen wir unsere sinusförmige Positionskodierung durch gelernte Positionseinbettungen [9] und beobachten nahezu identische Ergebnisse mit dem Basismodell.

6.3 Englische Wahlkreisanalyse

Um zu evaluieren, ob der Transformer auf andere Aufgaben verallgemeinert werden kann, führten wir Experimente zum Parsen englischer Wahlkreise durch. Diese Aufgabe stellt besondere Herausforderungen dar: Der Output unterliegt starken strukturellen Zwängen und ist deutlich länger als der Input. Darüber hinaus waren RNN-Sequenz-zu-Sequenz-Modelle nicht in der Lage, in Small-Data-Regimen Ergebnisse auf dem neuesten Stand der Technik zu erzielen [37].

Wir haben einen 4-Schicht-Transformator mit $d_{model} = 1024$ im Wall Street Journal (WSJ) Teil der Penn Treebank [25] trainiert, etwa 40K Trainingssätze. Wir trainierten es auch in einer semi-überwachten Umgebung, wobei wir die größeren High-Confidence- und BerkleyParser-Korpora mit etwa 17 Mio. Sätzen verwendeten [37]. Wir haben ein Vokabular von 16.000 Token für die reine WSJ-Einstellung und ein Vokabular von 32.000 Token für die halbüberwachte Einstellung verwendet.

Wir führten nur eine kleine Anzahl von Experimenten durch, um den Dropout, sowohl Aufmerksamkeit als auch Residuum (Abschnitt 5.4), Lernraten und Strahlgröße auf dem Entwicklungsset von Sektion 22 auszuwählen, alle anderen Parameter blieben gegenüber dem Englisch-Deutsch-Basistranslationsmodell unverändert. Während der

Inferenz verallgemeinern wir Tabelle 4: Der Transformer lässt sich gut auf das englische Constituency Parsing verallgemeinern (die Ergebnisse sind in Abschnitt 23 des WSJ)

Parser	Ausbildung	WSJ 23 F1
Vinyals & Kaiser et al. (2014) [37]	Nur WSJ, Nur diskriminierendes	88.3
Petrov et al. (2006) [29]	WSJ, Nur diskriminierendes	90.4
Zhu et al. (2013) [40]	WSJ, Nur diskriminierendes	90.4
Dyer et al. (2016) [8]	WSJ, diskriminierend	91.7
Transformator (4 Schichten)	Nur WSJ, diskriminierend	91.3
Zhu et al. (2013) [40]	halbbeaufsichtigt	91.3
Huang & Harper (2009) [14]	halb beaufsichtigt	91.3
McClosky et al. (2006) [26]	halb beaufsichtigt	92.1
Vinyals & Kaiser et al. (2014) [37]	halb beaufsichtigt	92.1
Transformator (4 Schichten)	halb betreut	92.7
Luong et al. (2015) [23]	Multitasking-Generativ	93.0
Dyer et al. (2016) [8]		93.3

Die maximale Ausgabelänge wurde auf Eingabelänge + 300 erhöht. Wir haben eine Strahlgröße von 21 und $\alpha = 0,3$ sowohl für WSJ als auch für die semi-überwachte Einstellung verwendet.

Unsere Ergebnisse in Tabelle 4 zeigen, dass unser Modell trotz des Fehlens einer aufgabenspezifischen Abstimmung überraschend gut abschneidet und bessere Ergebnisse liefert als alle zuvor berichteten Modelle mit Ausnahme der Recurrent Neural Network Grammar [8].

Im Gegensatz zu RNN-Sequenz-zu-Sequenz-Modellen [37] übertrifft der Transformer den Berkeley-Parser [29], selbst wenn nur mit dem WSJ-Trainingssatz von 40K Sätzen trainiert wird.

7 Fazit

In dieser Arbeit haben wir den Transformer vorgestellt, das erste Sequenztransduktionsmodell, das vollständig auf Aufmerksamkeit basiert und die in Encoder-Decoder-Architekturen am häufigsten verwendeten rekurrenten Schichten durch mehrköpfige Selbstaufmerksamkeit ersetzt.

Für Übersetzungsaufgaben kann der Transformer deutlich schneller trainiert werden als Architekturen, die auf Rekurrent- oder Convolutional Layer basieren. Sowohl bei den Übersetzungsaufgaben WMT 2014 Englisch-Deutsch als auch bei WMT 2014 Englisch-Französisch erreichen wir einen neuen Stand der Technik. In der ersten Aufgabe übertrifft unser bestes Modell sogar alle zuvor berichteten Ensembles.

Wir sind gespannt auf die Zukunft der aufmerksamkeitsbasierten Modelle und planen, sie auf andere Aufgaben anzuwenden. Wir planen, den Transformer auf Probleme auszuweiten, die andere Eingabe- und Ausgabemodalitäten als Text betreffen, und lokale, eingeschränkte Aufmerksamkeitsmechanismen zu untersuchen, um große Ein- und Ausgänge wie Bilder, Audio und Video effizient zu verarbeiten. Die Generierung weniger sequentiell zu gestalten, ist ein weiteres Forschungsziel von uns.

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

Danksagung Wir danken Nal Kalchbrenner und Stephan Gouws für ihre fruchtbaren Kommentare, Korrekturen und Inspirationen.

Referenzen

- [1] Jimmy Lei Ba, Jamie Ryan Kiros und Geoffrey E Hinton. Normalisierung der Schicht. arXiv-Vorabdruck arXiv:1607.06450, 2016.
- [2] Dzmitry Bahdanau, Kyunghyun Cho und Yoshua Bengio. Neuronale maschinelle Übersetzung durch gemeinsames Erlernen des Ausrichtens und Übersetzens. CoRR, abs/1409.0473, 2014.
- [3] Denny Britz, Anna Goldie, Minh-Thang Luong und Quoc V. Le. Massive Erforschung neuronaler Architekturen für maschinelle Übersetzung. CoRR, abs/1703.03906, 2017.
- [4] Jianpeng Cheng, Li Dong und Mirella Lapata. Long Short-Term Memory-Netzwerke für maschinelles Lesen. arXiv-Vorabdruck arXiv:1601.06733, 2016.

- [5] Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülçehre, Fethi Bougares, Holger Schwenk und Yoshua Bengio. Erlernen von Phrasendarstellungen mit rnn Encoder-Decoder für statistische maschinelle Übersetzung. CoRR, abs/1406.1078, 2014.
- [6] François Chollet. Xception: Deep Learning mit tiefentrennbaren Faltungen. arXiv-Vorabdruck arXiv:1610.02357, 2016.
- [7] Junyoung Chung, Çağlar Gülçehre, Kyunghyun Cho und Yoshua Bengio. Empirische Evaluierung von gated rekurrenten neuronalen Netzen bei der Sequenzmodellierung. CoRR, abs/1412.3555, 2014.
- [8] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros und Noah A. Smith. Wiederkehrende Grammatiken neuronaler Netzwerke. In Proc. von NAACL, 2016.
- [9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats und Yann N. Dauphin. Faltungslernen von Sequenz zu Sequenz. arXiv-Vorabdruck arXiv:1705.03122v2, 2017.
- [10] Alex Graves. Generierung von Sequenzen mit rekurrenten neuronalen Netzen. arXiv Vorabdruck arXiv:1308.0850, 2013.
- [11] Kaiming He, Xiangyu Zhang, Shaoqing Ren und Jian Sun. Tiefes Restlernen zur Erkennung von Bildern. 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. Gradientenströmung in rekurrenten Netzen: die Schwierigkeit, langfristige Abhängigkeiten zu lernen, 2001.
- [13] Sepp Hochreiter und Jürgen Schmidhuber. Langes Kurzzeitgedächtnis. Neuronale Berechnung, 9(8):1735– 1780, 1997.
- [14] Zhongqiang Huang und Mary Harper. Selbstlernende PCFG-Grammatiken mit latenten Annotationen in verschiedenen Sprachen. 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. Die Grenzen der Sprachmodellierung ausloten. arXiv-Vorabdruck arXiv:1602.02410, 2016.
- [16] Łukasz Kaiser und Samy Bengio. Kann das aktive Gedächtnis die Aufmerksamkeit ersetzen? In Advances in Neural Information Processing Systems, (NIPS), 2016.
- [17] Łukasz Kaiser und Ilya Sutskever. Neuronale GPUs lernen Algorithmen. In: Internationale Konferenz über Lernrepräsentationen (ICLR), 2016.
- [18] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves und Koray Kavukcuoglu. Neuronale maschinelle Übersetzung in linearer Zeit. arXiv-Vorabdruck arXiv:1610.10099v2, 2017.
- [19] Yoon Kim, Carl Denton, Luong Hoang und Alexander M. Rush. Strukturierte Aufmerksamkeitsnetzwerke. In Internationale Konferenz über Lernrepräsentationen, 2017.
- [20] Diederik Kingma und Jimmy Ba. Adam: Eine Methode zur stochastischen Optimierung. In ICLR, 2015.
- [21] Oleksii Kuchaiev und Boris Ginsburg. Faktorisierungstricks für LSTM-Netzwerke. arXiv-Vorabdruck 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-Vorabdruck arXiv:1703.03130, 2017.
- [23] Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals und Łukasz Kaiser. Multitasking-Lernen von Sequenz zu Sequenz. arXiv-Vorabdruck arXiv:1511.06114, 2015.
- [24] Minh-Thang Luong, Hieu Pham und Christopher D. Manning. Effektive Ansätze für aufmerksamkeitsbasierte neuronale maschinelle Übersetzung. arXiv-Vorabdruck arXiv:1508.04025, 2015.

- [25] Mitchell P. Marcus, Mary Ann Marcinkiewicz und Beatrice Santorini. Aufbau eines großen kommentierten Korpus des Englischen: Die Penn Treebank. *Computerlinguistik*, 19(2):313– 330, 1993.
- [26] David McClosky, Eugene Charniak und Mark Johnson. Effektives Selbsttraining für das Parsen. In Tagungsband der Human Language Technology Conference der NAACL, Hauptkonferenz, Seiten 152– 159. ACL, Juni 2006.
- [27] Ankur Parikh, Oscar Täckström, Dipanjan Das und Jakob Uszkoreit. Ein zerlegbares Aufmerksamkeitsmodell. In *empirischen Methoden der Verarbeitung natürlicher Sprache*, 2016.
- [28] Romain Paulus, Caiming Xiong und Richard Socher. Ein tiefgreifendes verstärktes Modell für die abstrakte Zusammenfassung. *arXiv-Vorabdruck arXiv:1705.04304*, 2017.
- [29] Slav Petrov, Leon Barrett, Romain Thibaux und Dan Klein. Erlernen einer genauen, kompakten und interpretierbaren Baumannotation. 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. Verwenden der Ausgabe-einbettung zum Verbessern von Sprachmodellen. *arXiv-Vorabdruck arXiv:1608.05859*, 2016.
- [31] Rico Sennrich, Barry Haddow und Alexandra Birch. Neuronale maschinelle Übersetzung von seltenen Wörtern mit Teilworteinheiten. *arXiv-Vorabdruck arXiv:1508.07909*, 2015.
- [32] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton und Jeff Dean. Unverschämt große neuronale Netze: Die spärlich abgeschirmte Experten-Schicht. *arXiv-Vorabdruck arXiv:1701.06538*, 2017.
- [33] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever und Ruslan Salakhutdinov. Dropout: eine einfache Möglichkeit, um eine Überanpassung neuronaler Netze zu verhindern. *Zeitschrift für Forschung im Bereich des maschinellen Lernens*, 15(1):1929– 1958, 2014.
- [34] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston und Rob Fergus. End-to-End-Speichernetzwerke. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama und R. Garnett, Herausgeber, *Fortschritte in neuronalen Informationsverarbeitungssystemen* 28, Seiten 2440– 2448. Curran Associates, Inc., 2015.
- [35] Ilya Sutskever, Oriol Vinyals und Quoc VV Le. Lernen von Sequenz zu Sequenz 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. Überdenken der Anfangsarchitektur für Computer Vision. *CoRR*, abs/1512.00567, 2015.
- [37] Vinyals & Kaiser, Koo, Petrov, Sutskever und Hinton. Grammatik als Fremdsprache. In *Fortschritte bei neuronalen Informationsverarbeitungssystemen*, 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 maschinelles Übersetzungssystem: Überbrückung der Lücke zwischen menschlicher und maschineller Übersetzung. *arXiv-Vorabdruck arXiv:1609.08144*, 2016.
- [39] Jie Zhou, Ying Cao, Xuguang Wang, Peng Li und Wei Xu. Tiefgreifende rekurrente Modelle mit Fast-Forward-Verbindungen für die neuronale maschinelle Übersetzung. *CoRR*, abs/1606.04199, 2016.
- [40] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang und Jingbo Zhu. Schnelles und genaues Shift-Reduce-Parsen von Bestandteilen. In : *Proceedings of the 51st Annual Meeting of the ACL (Band 1: Long Papers)*, Seiten 434– 443. ACL, August 2013.

Eingang-Eingangs-Schicht 5

Achtung Visualisierungen

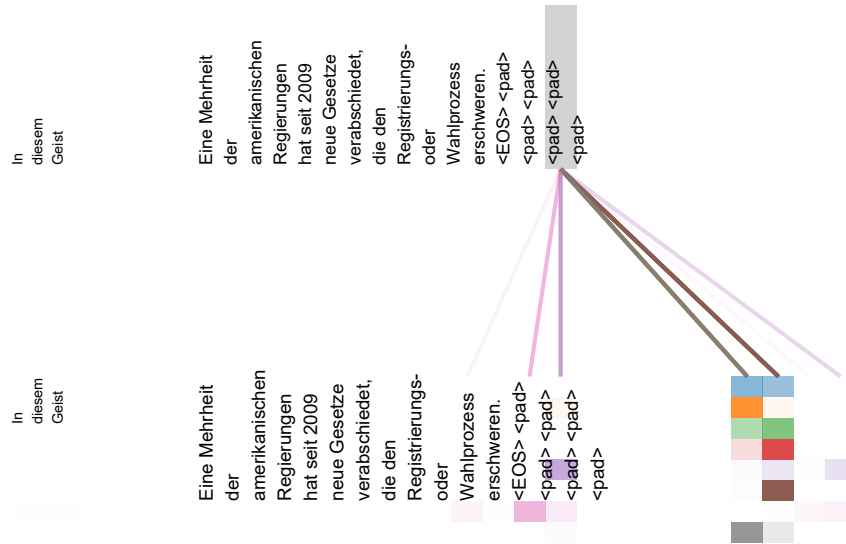


Abbildung 3: Ein Beispiel für den Aufmerksamkeitsmechanismus, der Langstreckenabhängigkeiten in der Selbstaufmerksamkeit des Encoders in Schicht 5 von 6 folgt. Viele der Aufmerksamkeitsköpfe widmen sich einer entfernten Abhängigkeit des Verbs "machen" und vervollständigen die Phrase "machen... schwieriger zu machen". Aufmerksamkeiten, die hier nur für das Wort "Herstellung" angezeigt werden. Unterschiedliche Farben stehen für unterschiedliche Köpfe. Am besten in Farbe betrachten.

Eingabe-Eingabe-Schicht5

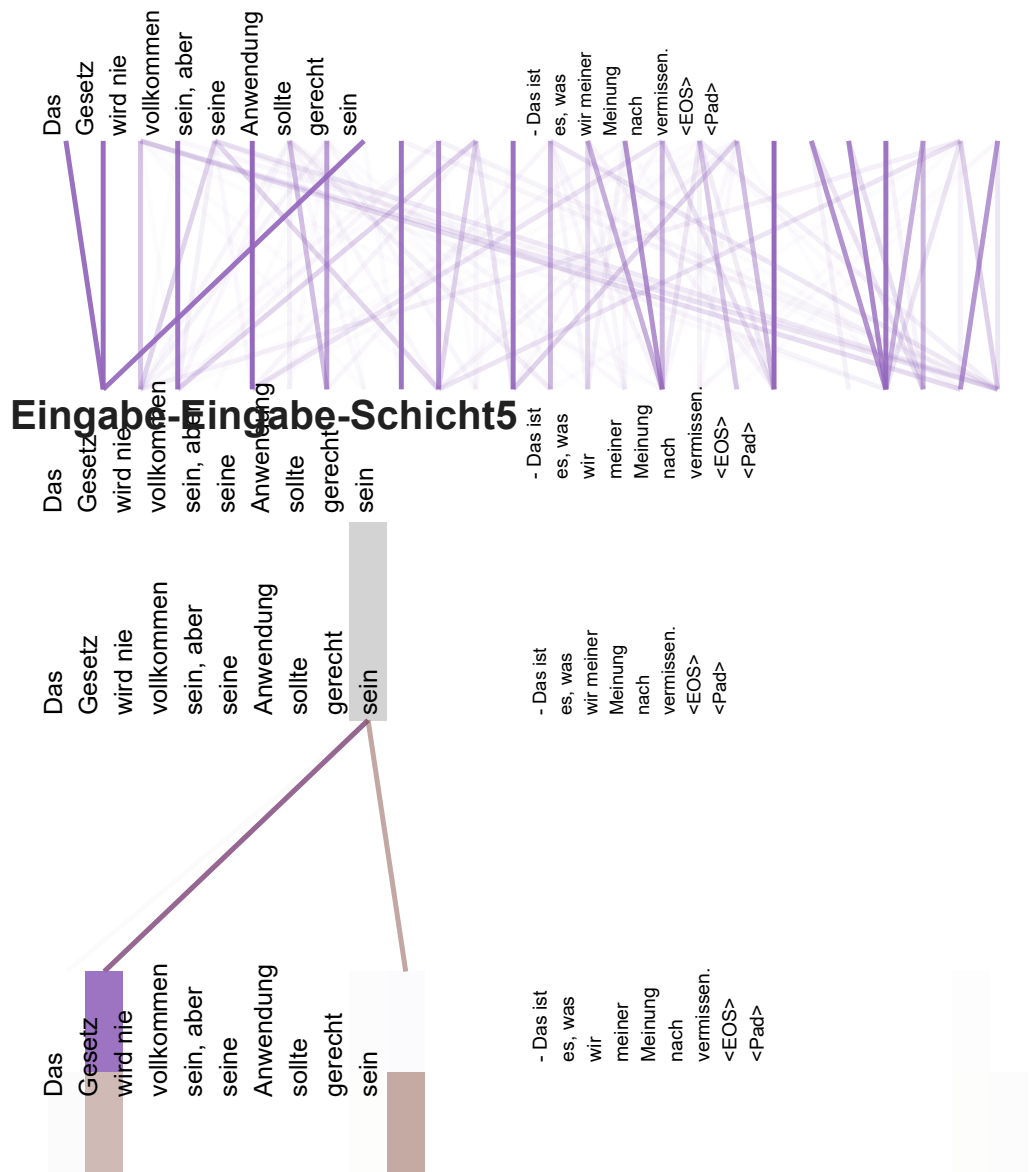


Abbildung 4: Zwei Aufmerksamkeitsköpfe, ebenfalls in Schicht 5 von 6, offenbar an der Anaphorenauflösung beteiligt. Oben: Volle Aufmerksamkeit für Kopf 5. Unten: Isolierte Aufmerksamkeiten nur aus dem Wort "its" für die Aufmerksamkeitsköpfe 5 und 6. Beachten Sie, dass die Aufmerksamkeit für dieses Wort sehr scharf ist.

Eingabe-Eingabe-Schicht5

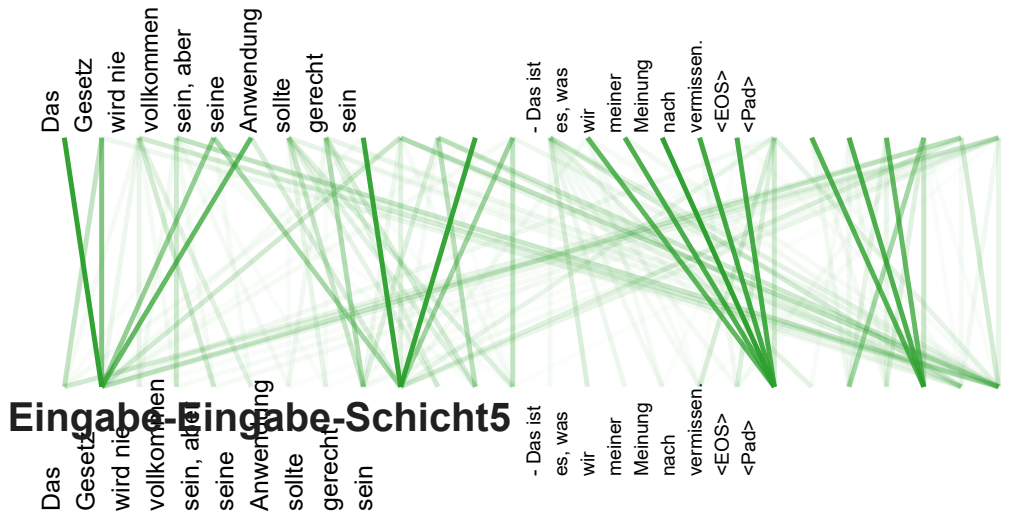


Abbildung 5: Viele der Aufmerksamkeitsköpfe zeigen ein Verhalten, das mit der Struktur des Satzes zusammenzuhängen scheint. Wir geben oben zwei solcher Beispiele von zwei verschiedenen Köpfen aus der Selbstaufmerksamkeit des Encoders auf Schicht 5 von 6. Die Köpfe haben deutlich gelernt, unterschiedliche Aufgaben zu erfüllen.