Achtung ist alles, was du brauchst

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
Universität Toronto
aidan@cs.toronto.edu

Lukasz Kaiser
Google Brain
lukaszkaiser@google.com

Illia Polosukhin
illia.polosukhin@gmail.com

Zusammenfassung

Die vorherrschenden Sequenztransduktionsmodelle basieren auf komplexen rezidivierenden oder konvolutionäre neuronale Netzwerke, die einen Encoder und einen Decoder enthalten. Ausführung Modelle verbinden auch den Encoder und Decoder durch eine Aufmerksamkeit Wir schlagen eine neue einfache Netzwerkarchitektur vor, den Transformer, ausschließlich auf der Grundlage von Aufmerksamkeitsmechanismen, Dispensing mit Rezidiv und Konvolutionen Die Experimente an zwei maschinellen Übersetzungsaufgaben zeigen, dass diese Modelle in der Qualität überlegen sein, während parallelisierbarer und erfordern erheblich weniger Zeit zum Trainieren. Unser Modell erreicht 28,4 BLEU auf der WMT 2014 Englisch-Deutsch-Übersetzungsaufgabe, Verbesserung über die bestehenden besten Ergebnisse, einschließlich ensembles, von über 2 BLEU. Auf der WMT 2014 Englisch-Französische Übersetzungsaufgabe, unser Modell stellt ein neues Modell auf dem neuesten Stand der Technik BLEU-Score von 41,8 nach Ausbildung für 3,5 Tage auf acht GPUs, ein kleiner Teil der Ausbildungskosten der beste Modelle aus der Literatur. Wir zeigen, dass der Transformer gut verallgemeinert andere Aufgaben, indem sie erfolgreich auf englischen Wahlkreis Parsing beide mit große und begrenzte Trainingsdaten.

Gleicher Beitrag. Listing-Ordnung ist zufällig. Jakob schlug vor, RNNs durch Selbstachtung zu ersetzen und begann der Versuch, diese Idee zu bewerten. Ashish, mit Illia, entworfen und implementiert die ersten Transformer-Modelle und Noam schlug vor, skalierte Punkt-Produkt-Aufmerksamkeit, Mehr-Kopf-Aufmerksamkeit und die parameterfreie Positionsdarstellung und wurde die andere Person, die in fast jedem Detail. Niki konzipiert, implementiert, abgestimmt und bewertet unzählige Modellvarianten in unserer originalen Codebase und tensor2tensor. Llion experimentierte auch mit neuartigen Modellvarianten, war verantwortlich für unsere erste Codebase, und effiziente Folgerungen und Visualisierungen. Lukasz und Aidan verbrachten unzählige lange Tage damit, verschiedene Teile von und Umsetzung von Tensor2tensor, Ersetzung unserer früheren Codebasis, deutliche Verbesserung der Ergebnisse und massive Beschleunigung Unsere Forschung.

Arbeit durchgeführt, während bei Google Brain.

Arbeit durchgeführt, während bei Google Research.

31. Konferenz über neurale Informationsverarbeitungssysteme (NIPS 2017), Long Beach, CA, USA.

1 . .Einleitung . . . . . .

Rezidivierende neuronale Netzwerke, langes Kurzzeitgedächtnis [13] und gated rezidivieren
insbesondere als Stand der Technik in der Sequenzmodellierung fest etabliert sind und
Transduktionsprobleme wie Sprachmodellierung und maschinelle Übersetzung [35, <a href
Bemühungen haben seitdem fortgesetzt, die Grenzen von wiederkehrenden Sprachmodellen und Encoder-Decoder zu verschie
Architekturen [38, 24, 15].

Recurrent-Modelle typischerweise Faktorberechnung entlang der Symbolpositionen der Ein- und Ausgabe
Sequenzen. Richten Sie die Positionen zu Schritten in Rechenzeit, erzeugen sie eine Sequenz von versteckten
Staaten , als Funktion des vorherigen versteckten Zustandes und der Eingang für Position . Dies ist inhärent
sequenzieller Charakter schließt Parallelisierung innerhalb von Ausbildungsbeispielen aus, die länger kritisch wird
Sequenzlängen, als Speicher-Beschränkungen begrenzen Batching über Beispiele. Jüngste Arbeit hat erreicht
signifikante Verbesserungen der Recheneffizienz durch Factorisierungstricks [21] und bedi
Berechnung [32], bei gleichzeitiger Verbesserung der Modellleistung bei letzterem.
Einschränkungen der sequentiellen Berechnung bleiben jedoch bestehen.

Die Aufmerksamkeitsmechanismen sind zu einem festen Bestandteil der zwingenden Sequenzmodellierung und Transduc-
modellen in verschiedenen Aufgaben, die die Modellierung von Abhängigkeiten ohne Rücksicht auf ihre Entfernung in
die Eingabe- oder Ausgabesequenzen [2, 19]. In allen bis
in Verbindung mit einem wiederkehrenden Netzwerk verwendet werden.

In dieser Arbeit schlagen wir den Transformer vor, eine Modellarchitektur, die ein Wiederaufleben verhindert und stattdessen
sich vollständig auf einen Aufmerksamkeitsmechanismus zu verlassen, um globale Abhängigkeiten zwischen Input und Outpu
Der Transformer ermöglicht eine deutlich mehr Parallelisierung und kann einen neuen Stand der Technik in erreichen
Übersetzungsqualität, nachdem sie auf acht P100 GPUs für nur zwölf Stunden trainiert wurde.

2 Hintergrund

Das Ziel, die sequenzielle Berechnung zu reduzieren, bildet auch die Grundlage der erweiterten Neural-GPU
[16], ByteNet [18] und ConvS2S <a href="attention.html
Block, Berechnung versteckter Darstellungen parallel für alle Eingangs- und Ausgangspositionen. In diesen Modellen,
die Anzahl der Operationen, die erforderlich sind, um Signale von zwei beliebigen Eingangs- oder Ausgangspositionen zu bez
im Abstand zwischen den Positionen, linear für ConvS2S und logarithmisch für ByteNet.
Es ist schwieriger, Abhängigkeiten zwischen entfernten Positionen zu lernen [12]. Im Trans
auf eine konstante Anzahl von Operationen reduziert, wenn auch auf Kosten einer reduzierten effektiven Abwicklung
um aufmerksamkeitsgewichtete Positionen zu durchschnittlich, ein Effekt, den wir mit Multi-Head Attention als
beschrieben in Abschnitt 3.2.

Selbstaufmerksamkeit, manchmal Intra-Aufmerksamkeit genannt, ist ein Aufmerksamkeitsmechanismus, der unterschiedliche
einer einzelnen Sequenz, um eine Darstellung der Sequenz zu berechnen.
erfolgreich in einer Vielzahl von Aufgaben eingesetzt werden, einschließlich Leseverständnis, abstrakte Zusammenfassung,
textual mitwirkende und lernende aufgabenunabhängige Satzdarstellungen [4, <a href="atte

End-to-End-Speichernetzwerke basieren auf einem wiederkehrenden Aufmerksamkeitsmechanismus anstelle von Sequenz-
ein abgestimmtes Rezidiv und haben gezeigt, dass gut auf einfach-sprachliche Frage beantworten und
Sprachmodellierungsaufgaben [34].

Nach bestem Wissen ist der Transformer jedoch das erste Transduktionsmodell, das sich
vollständig auf Selbstachtung, um Darstellungen seiner Ein- und Ausgabe zu berechnen, ohne Sequenz-
ausgerichtete RNNs oder Konvolution. In den folgenden Abschnitten beschreiben wir den Transformer, motivieren
Selbstachtung und diskutieren ihre Vorteile gegenüber Modellen wie [17, 18] und <a href="

3 Modellarchitektur

Die meisten wettbewerbsfähigen neuronalen Sequenztransduktionsmodelle haben eine Encoder-Decoder-Struktur <a href="att
Hier zeigt der Encoder eine Eingabefolge von Symboldarstellungen zu einer Sequenz
von kontinuierlichen Darstellungen . Gegeben , erzeugt der Decoder dann einen Ausgang
Reihenfolge von Symbolen ein Element zu einem Zeitpunkt. Bei jedem Schritt ist das Modell auto-regressiv
[10], die zuvor erzeugten Symbole als zusätzliche Eingabe bei der Erzeugung des nächsten.

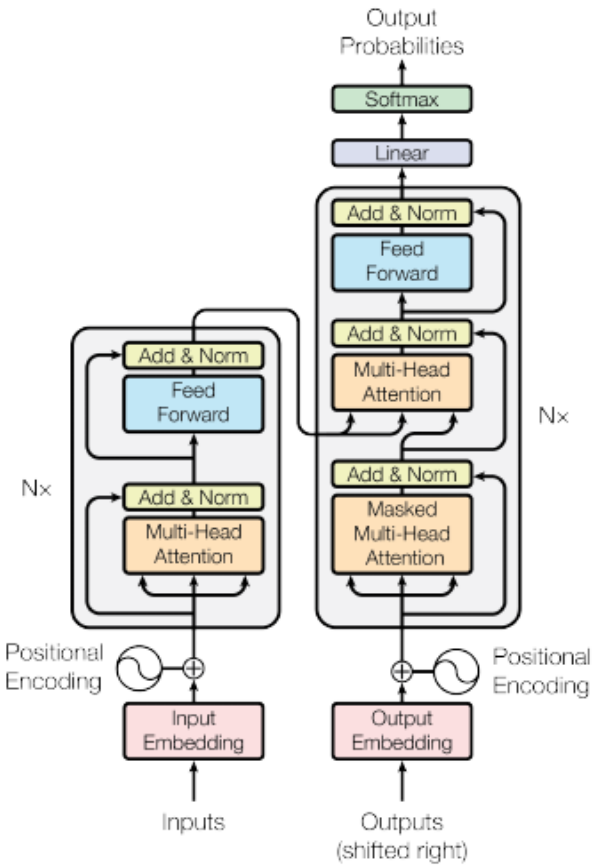


Figure 1: The Transformer - model architecture.

The Transformer follows this overall architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder, shown in the left and right halves of Figure 1, respectively.

3.1 Encoder and Decoder Stacks

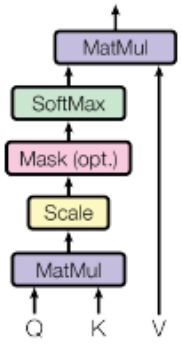
Encoder: The encoder is composed of a stack of identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. We employ a residual connection [11] around e the two sub-layers, followed by layer normalization [1]. That is, the output of each sub-laye , where is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension .

Decoder: The decoder is also composed of a stack of identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, we employ residual connections around each of the sub-layers, followed by layer normalization. We also modify the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with fact that the output embeddings are offset by one position, ensures that the predictions for position can depend only on the known outputs at positions less than .

3.2 Attention

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum

Scaled Dot-Product Attention



Multi-Head Attention

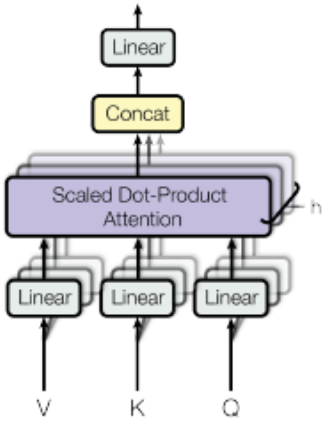


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

3.2.1 Scaled Dot-Product Attention

We call our particular attention "Scaled Dot-Product Attention" (Figure 2). The input consists queries and keys of dimension , and values of dimension . We compute the dot products of the query with all keys, divide each by —, and apply a softmax function to obtain the weights on the values.

In practice, we compute the attention function on a set of queries simultaneously, packed together into a matrix . The keys and values are also packed together into matrices and . We compute the matrix of outputs as:

(1)

The two most commonly used attention functions are additive attention [2], and dot-product plicative) attention. Dot-product attention is identical to our algorithm, except for the scaling factor of √ . Additive attention computes the compatibility function using a feed-forward network with a single hidden layer. While the two are similar in theoretical complexity, dot-product attention is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code.

While for small values of the two mechanisms perform similarly, additive attention outperforms dot product attention without scaling for larger values of [3]. We suspect that for large va , the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients . To counteract this effect, we scale the dot products by

3.2.2 Multi-Head Attention

Instead of performing a single attention function with -dimensional keys, values and queries, we found it beneficial to linearly project the queries, keys and values times with different, learned linear projections to , and dimensions, respectively. On each of these projected versions of queries, keys and values we then perform the attention function in parallel, yielding -dimensional

To illustrate why the dot products get large, assume that the components of and are independent random variables with mean and variance . Then their dot product, , has mean and variance .

output values. These are concatenated and once again projected, resulting in the final values, as depicted in Figure 2.

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.

where

Where the projections are parameter matrices × , × , × and × .

In this work we employ parallel attention layers, or heads. For each of these we use . Due to the reduced dimension of each head, the total computational cost is similar to that of single-head attention with full dimensionality.

3.2.3 Applications of Attention in our Model

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

- In "encoder-decoder attention" layers, the queries come from the previous decoder layer, and the memory keys and values come from the output of the encoder. This allows every position in the decoder to attend over all positions in the input sequence. This mimics the typical encoder-decoder attention mechanisms in sequence-to-sequence models such as [38, 2, 9].
- The encoder contains self-attention layers. In a self-attention layer all of the keys, values and queries come from the same place, in this case, the output of the previous layer in the encoder. Each position in the encoder can attend to all positions in the previous layer of the encoder.
- Similarly, self-attention layers in the decoder allow each position in the decoder to attend to all positions in the decoder up to and including that position. We need to prevent leftward information flow in the decoder to preserve the auto-regressive property. We implement this inside of scaled dot-product attention by masking out (setting to) all values in the input of the softmax which correspond to illegal connections. See Figure 2.

3.3 Position-wise Feed-Forward Networks

In addition to attention sub-layers, each of the layers in our encoder and decoder contains a fully connected feed-forward network, which is applied to each position separately and identically. This consists of two linear transformations with a ReLU activation in between.

(2)

While the linear transformations are the same across different positions, they use different parameters from layer to layer. Another way of describing this is as two convolutions with kernel size 1. The dimensionality of input and output is , and the inner-layer has dimensionality .

3.4 Embeddings and Softmax

Similarly to other sequence transduction models, we use learned embeddings to convert the input tokens and output tokens to vectors of dimension . We also use the usual learned linear transformation and softmax function to convert the decoder output to predicted next-token probabilities. In our model, we share the same weight matrix between the two embedding layers and the pre-softmax linear transformation, similar to [30]. In the embedding layers, we multiply those weights by .

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. is the sequence length, is the representation dimension, is the kernel size of convolutions and the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention			
Recurrent			
Convolutional			
Self-Attention (restricted)			

3.5 Positional Encoding

Since our model contains no recurrence and no convolution, in order for the model to make use of the order of the sequence, we must inject some information about the relative or absolute position of the tokens in the sequence. To this end, we add "positional encodings" to the input embeddings at the bottoms of the encoder and decoder stacks. The positional encodings have the same dimension

as the embeddings, so that the two can be summed. There are many choices of positional encodings, learned and fixed [9].

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

where is the position and is the dimension. That is, each dimension of the positional encoding corresponds to a sinusoid. The wavelengths form a geometric progression from to . We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset , can be represented as a linear function of .

We also experimented with using learned positional embeddings [9] instead, and found that versions produced nearly identical results (see Table 3 row (E)). We chose the sinusoidal vers because it may allow the model to extrapolate to sequence lengths longer than the ones encountered during training.

4 Why Self-Attention

In this section we compare various aspects of self-attention layers to the recurrent and convolutional layers commonly used for mapping one variable-length sequence of symbol representations to another sequence of equal length , with , such as a hidden layer in a typical sequence transduction encoder or decoder. Motivating our use of self-attention we consider three desiderata.

One is the total computational complexity per layer. Another is the amount of computation that can be parallelized, as measured by the minimum number of sequential operations required.

The third is the path length between long-range dependencies in the network. Learning long-range dependencies is a key challenge in many sequence transduction tasks. One key factor affecting the ability to learn such dependencies is the length of the paths forward and backward signals have to traverse in the network. The shorter these paths between any combination of positions in the input and output sequences, the easier it is to learn long-range dependencies [12]. Hence we also c the maximum path length between any two input and output positions in networks composed of the different layer types.

As noted in Table 1, a self-attention layer connects all positions with a constant number of se executed operations, whereas a recurrent layer requires sequential operations. In terms of computational complexity, self-attention layers are faster than recurrent layers when the sequence

length is smaller than the representation dimensionality , which is most often the case with sentence representations used by state-of-the-art models in machine translations, such as word-piece [38] and byte-pair [31] representations. To improve comp very long sequences, self-attention could be restricted to considering only a neighborhood of size in the input sequence centered around the respective output position. This would increase the maximum path length to . We plan to investigate this approach further in future work.

A single convolutional layer with kernel width does not connect all pairs of input and output positions. Doing so requires a stack of convolutional layers in the case of contiguous kernels, or in the case of dilated convolutions [18], increasing the length of the longest p between any two positions in the network. Convolutional layers are generally more expensive than recurrent layers, by a factor of . Separable convolutions [6], however, decrease the compl considerably, to . Even with , however, the complexity of a separable convolution is equal to the combination of a self-attention layer and a point-wise feed-forward layer, the approach we take in our model.

As side benefit, self-attention could yield more interpretable models. We inspect attention distributions from our models and present and discuss examples in the appendix. Not only do individual attention heads clearly learn to perform different tasks, many appear to exhibit behavior related to the syntactic and semantic structure of the sentences.

5 Training

This section describes the training regime for our models.

5.1 Training Data and Batching

We trained on the standard WMT 2014 English-German dataset consisting of about 4.5 million sentence pairs. Sentences were encoded using byte-pair encoding [3], which has a shared so target vocabulary of about 37000 tokens. For English-French, we used the significantly larger WMT 2014 English-French dataset consisting of 36M sentences and split tokens into a 32000 word-piece vocabulary [38]. Sentence pairs were batched together by approximate sequence length. Eac batch contained a set of sentence pairs containing approximately 25000 source tokens and 25000 target tokens.

5.2 Hardware and Schedule

We trained our models on one machine with 8 NVIDIA P100 GPUs. For our base models using the hyperparameters described throughout the paper, each training step took about 0.4 seconds. We trained the base models for a total of 100,000 steps or 12 hours. For our big models,(described on the bottom line of table 3), step time was 1.0 seconds. The big models were trained for 300,000 s (3.5 days).

5.3 Optimizer

We used the Adam optimizer [20] with , , and . We varied the learning rate over the course of training, according to the formula:

$$\text{(3)}$$

This corresponds to increasing the learning rate linearly for the first training steps, and decreasing it thereafter proportionally to the inverse square root of the step number. We used .

5.4 Regularization

We employ three types of regularization during training:

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		
GNMT + RL [38]	24.6	39.92		
ConvS2S [9]	25.16	40.46		
MoE [32]	26.03	40.56		
Deep-Att + PosUnk Ensemble [39]		40.4		
GNMT + RL Ensemble [38]	26.30	41.16		
ConvS2S Ensemble [9]	26.36	41.29		
Transformer (base model)	27.3	38.1		
Transformer (big)	**28.4**	**41.8**		

Residual Dropout We apply dropout [33] to the output of each sub-layer, before it is adde sub-layer input and normalized. In addition, we apply dropout to the sums of the embeddings and the positional encodings in both the encoder and decoder stacks. For the base model, we use a rate of .

Label Smoothing During training, we employed label smoothing of value [36]. hurts perplexity, as the model learns to be more unsure, but improves accuracy and BLEU score.

6 Results

6.1 Machine Translation

On the WMT 2014 English-to-German translation task, the big transformer model (Transformer (big) in Table 2) outperforms the best previously reported models (including ensembles) by more t BLEU, establishing a new state-of-the-art BLEU score of . The configuration of this model is listed in the bottom line of Table . Training took days on 8 P100 GPUs. Even our base model surpasses all previously published models and ensembles, at a fraction of the training cost of any of the competitive models.

On the WMT 2014 English-to-French translation task, our big model achieves a BLEU score of , outperforming all of the previously published single models, at less than the training cost of the previous state-of-the-art model. The Transformer (big) model trained for English-to-French used dropout rate , instead of .

For the base models, we used a single model obtained by averaging the last 5 checkpoints, which were written at 10-minute intervals. For the big models, we averaged the last 20 checkpoints. We used beam search with a beam size of and length penalty [38]. These hyperpara were chosen after experimentation on the development set. We set the maximum output length during inference to input length + , but terminate early when possible [38].

Table 2 summarizes our results and compares our translation quality and training costs to oth architectures from the literature. We estimate the number of floating point operations used to train a model by multiplying the training time, the number of GPUs used, and an estimate of the sustained single-precision floating-point capacity of each GPU .

6.2 Model Variations

To evaluate the importance of different components of the Transformer, we varied our base model in different ways, measuring the change in performance on English-to-German translation on the

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

Table 3: Variations on the Transformer architecture. Unlisted values are identical to those of the base model. All metrics are on the English-to-German translation development set, newstest2013. Listed perplexities are per-wordpiece, according to our byte-pair encoding, and should not be compared to per-word perplexities.

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	train steps	PPL (dev)	BLEU (dev)	params $\times 10^6$
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65
(A)				1	512	512				5.29	24.9	
				4	128	128				5.00	25.5	
				16	32	32				4.91	25.8	
				32	16	16				5.01	25.4	
(B)					16					5.16	25.1	58
					32					5.01	25.4	60
(C)	2									6.11	23.7	36
	4									5.19	25.3	50
	8									4.88	25.5	80
		256			32	32				5.75	24.5	28
		1024			128	128				4.66	26.0	168
			1024							5.12	25.4	53
			4096							4.75	26.2	90
(D)							0.0			5.77	24.6	
							0.2			4.95	25.5	
								0.0		4.67	25.3	
								0.2		5.47	25.7	
(E)	positional embedding instead of sinusoids									4.92	25.7	
big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

development set, newstest2013. We used beam search as described in the previous section, but no checkpoint averaging. We present these results in Table 3.

In Table 3 rows (A), we vary the number of attention heads and the attention key and value d keeping the amount of computation constant, as described in Section 3.2.2. While single-head attention is 0.9 BLEU worse than the best setting, quality also drops off with too many heads.

In Table 3 rows (B), we observe that reducing the attention key size hurts model quality. This suggests that determining compatibility is not easy and that a more sophisticated compatibility function than dot product may be beneficial. We further observe in rows (C) and (D) that, as expected, bigger models are better, and dropout is very helpful in avoiding over-fitting. In row (E) we replace our sinusoidal positional encoding with learned positional embeddings [9], and observe nearly i results to the base model.

6.3 English Constituency Parsing

To evaluate if the Transformer can generalize to other tasks we performed experiments on English constituency parsing. This task presents specific challenges: the output is subject to strong structural constraints and is significantly longer than the input. Furthermore, RNN sequence-to-sequence models have not been able to attain state-of-the-art results in small-data regimes [37].

We trained a 4-layer transformer with on the Wall Street Journal (WSJ) portion of the Penn Treebank [25], about 40K training sentences. We also trained it in a semi-supervised s using the larger high-confidence and BerkleyParser corpora from with approximately 17M sentences [37]. We used a vocabulary of 16K tokens for the WSJ only setting and a vocabulary of 32K for the semi-supervised setting.

We performed only a small number of experiments to select the dropout, both attention and residual (section 5.4), learning rates and beam size on the Section 22 development set, all other param remained unchanged from the English-to-German base translation model. During inference, we

Table 4: The Transformer generalizes well to English constituency parsing (Results are on Section 23 of WSJ)

Parser	Training	WSJ 23 F1
Vinyals & Kaiser el al. (2014) [37]	WSJ only, discriminative	88.3
Petrov et al. (2006) [29]	WSJ only, discriminative	90.4
Zhu et al. (2013) [40]	WSJ only, discriminative	90.4
Dyer et al. (2016) [8]	WSJ only, discriminative	91.7
Transformer (4 layers)	WSJ only, discriminative	91.3
Zhu et al. (2013) [40]	semi-supervised	91.3
Huang & Harper (2009) [14]	semi-supervised	91.3
McClosky et al. (2006) [26]	semi-supervised	92.1
Vinyals & Kaiser el al. (2014) [37]	semi-supervised	92.1
Transformer (4 layers)	semi-supervised	92.7
Luong et al. (2015) [23]	multi-task	93.0
Dyer et al. (2016) [8]	generative	93.3

increased the maximum output length to input length + . We used a beam size of and for both WSJ only and the semi-supervised setting.

Our results in Table 4 show that despite the lack of task-specific tuning our model performs prisingly well, yielding better results than all previously reported models with the exception of the Recurrent Neural Network Grammar [8].

In contrast to RNN sequence-to-sequence models [37], the Transformer outperforms the Be Parser [29] even when training only on the WSJ training set of 40K sentences.

7 Conclusion

In this work, we presented the Transformer, the first sequence transduction model based entirely on attention, replacing the recurrent layers most commonly used in encoder-decoder architectures with multi-headed self-attention.

For translation tasks, the Transformer can be trained significantly faster than architectures based on recurrent or convolutional layers. On both WMT 2014 English-to-German and WMT 2014 English-to-French translation tasks, we achieve a new state of the art. In the former task our best model outperforms even all previously reported ensembles.

We are excited about the future of attention-based models and plan to apply them to other tasks. We plan to extend the Transformer to problems involving input and output modalities other than text and to investigate local, restricted attention mechanisms to efficiently handle large inputs and outputs such as images, audio and video. Making generation less sequential is another research goals of ours.

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

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

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. , 2016.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. , abs/1409.0473, 2014.

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

[4] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. , 2016.

[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. , abs/1406.1078, 2014.

[6] Francois Chollet. Xception: Deep learning with depthwise separable convolutions. , 2016.

[7] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. , abs/1412.3555, 2014.

[8] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. Recurrent neural network grammars. In , 2016.

[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. , 2017.

[10] Alex Graves. Generating sequences with recurrent neural networks. , 2013.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In , pages 770–778, 2016.

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

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

[14] Zhongqiang Huang and Mary Harper. Self-training PCFG grammars with latent annotations across languages. In , pages 832–841. ACL, August 2009.

[15] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. , 2016

[16] Łukasz Kaiser and Samy Bengio. Can active memory replace attention? In , 2016.

[17] Łukasz Kaiser and Ilya Sutskever. Neural GPUs learn algorithms. In , 2016.

[18] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. , 2017.

[19] Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. Structured attention networks. In , 2017.

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

[21] Oleksii Kuchaiev and Boris Ginsburg. Factorization tricks for LSTM networks. , 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. , 2017.

[23] Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. , 2015.

[24] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. , 201

[25] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. , 19(2):313–330, 1993.

[26] David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In , pages 152–159. ACL, June 2006.

[27] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model. In , 2016.

[28] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. , 2017.

[29] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In , pages 433–440. ACL, July 2006.

[30] Ofir Press and Lior Wolf. Using the output embedding to improve language models. , 2016.

[31] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. , 2015.

[32] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. , 2017.

[33] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. , 15(1):1929–1958, 2014.

[34] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, , pages 2440–2448. Curran Associates, Inc., 2015.

[35] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In , pages 3104–3112, 2014.

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

[37] Vinyals & Kaiser, Koo, Petrov, Sutskever, and Hinton. Grammar as a foreign language. In , 2015.

[38] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. , 2016.

[39] Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. Deep recurrent models with fast-forward connections for neural machine translation. , abs/1606.04199, 2016.

[40] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. Fast and accurate shift-reduce constituent parsing. In , pages 434–443. ACL, August 2013.

Attention Visualizations

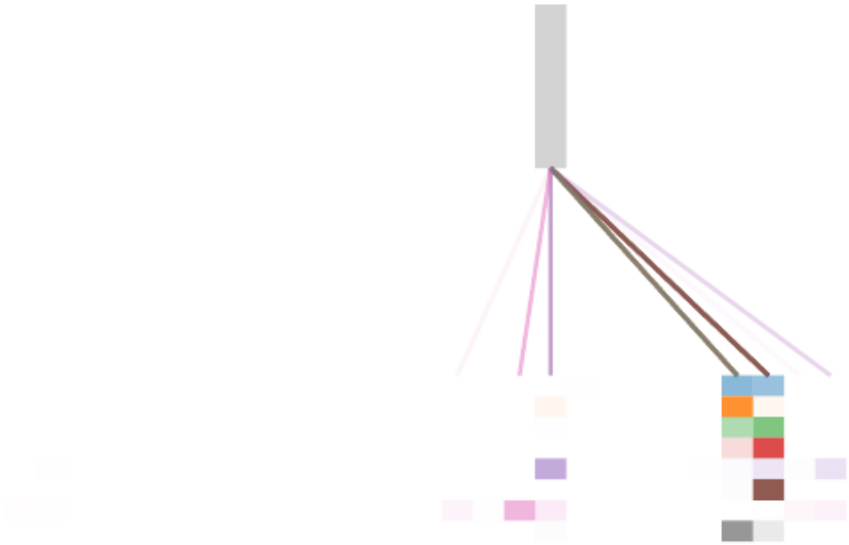


Figure 3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb ‘making’, completing the phrase ‘making...more difficult’. Attentions here shown only for the word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

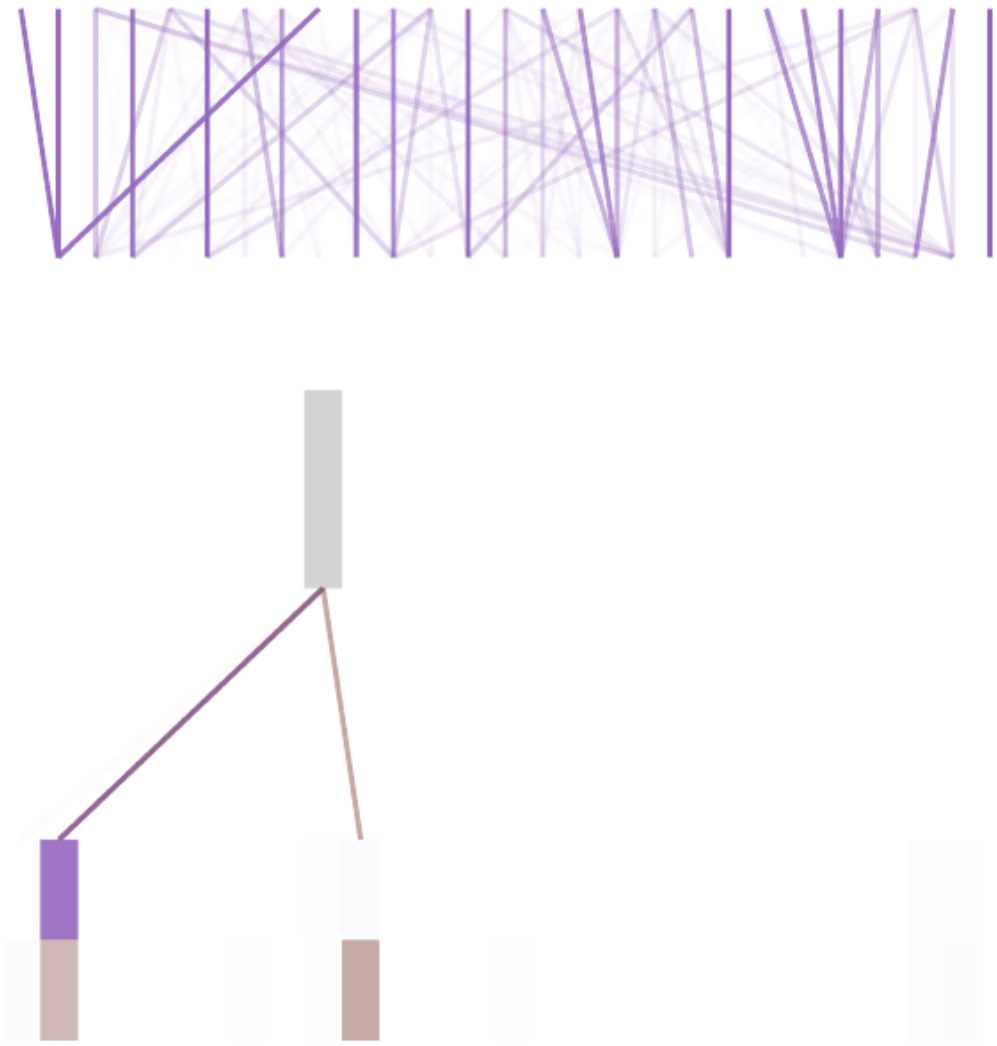


Figure 4: Two attention heads, also in layer 5 of 6, apparently involved in anaphora resolution. Top: Full attentions for head 5. Bottom: Isolated attentions from just the word ‘its’ for attention heads 5 and 6. Note that the attentions are very sharp for this word.

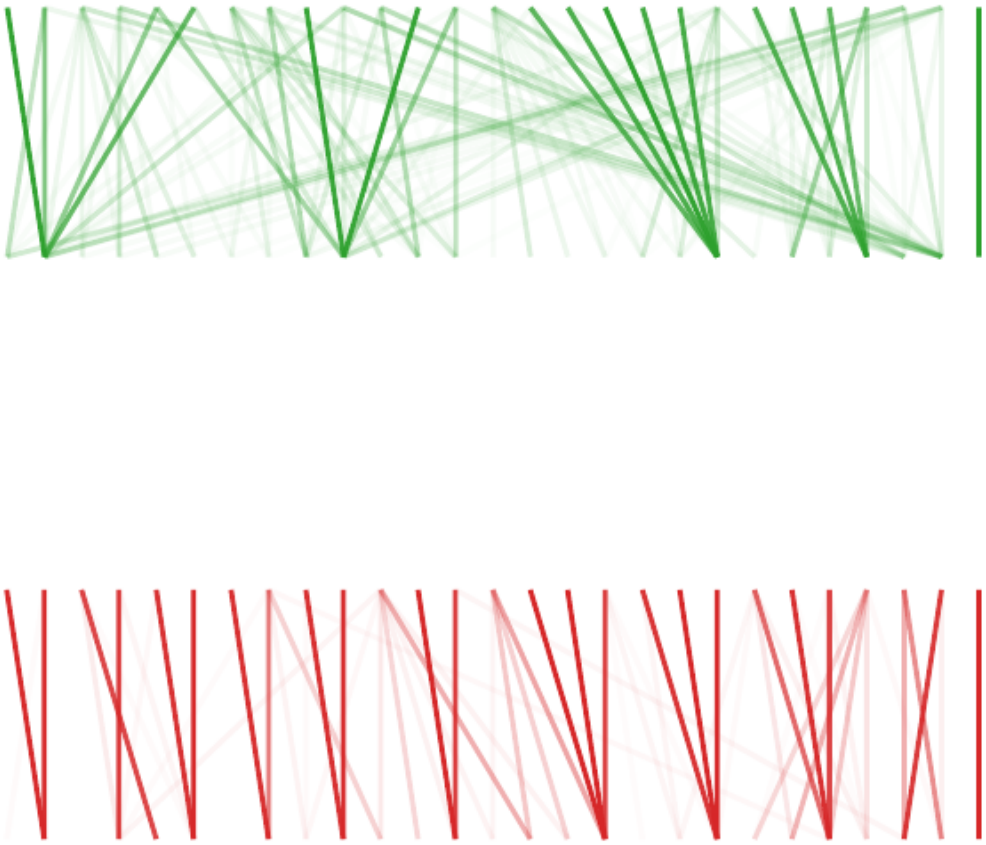


Figure 5: Many of the attention heads exhibit behaviour that seems related to the structure of the sentence. We give two such examples above, from two different heads from the encoder self-attention at layer 5 of 6. The heads clearly learned to perform different tasks.