Provided proper attribution is provided, Google hereby grants permission to reproduce the tables and figures in this paper solely for use in journalistic or scholarly works.

Attention Is All You Need

Ashish Vaswani[*]
Google Brain
avaswani@google.com

Noam Shazeer[*]
Google Brain
noam@google.com

Niki Parmar[*]
Google Research
nikip@google.com

Jakob Uszkoreit[*]
Google Research
usz@google.com

Llion Jones[*]
Google Research
llion@google.com

Aidan N. Gomez[*] [†]
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser[*]
Google Brain
lukaszkaiser@google.com

Illia Polosukhin[*] [‡]
illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

[*]Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

[†]Work performed while at Google Brain.

[‡]Work performed while at Google Research.

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

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

При условии надлежащего указания авторства Google настоящим предоставляет разрешение на воспроизведение таблиц и рисунков в данной статье исключительно для использования в журналистских или научных работах.

Внимания вам достаточно

Ашиш Васвани*
Google Brain
avaswani@google.com

Ноам Шазиер*
Google Brain
noam@google.com

Ники Пармар*
Google Research
nikip@google.com

Якоб Ускорайт*
Google Research
usz@google.com

Ллион Джонс*
Google Research
llion@google.com

Aidan N. Gomez*
Торонтский университет
aidan@cs.toronto.edu

Łukasz Kaiser*
Google Brain
lukaszkaiser@google.com

Илля Полосухин*
illia.polosukhin@gmail.com

Аннотация

Доминирующие модели трансдукции последовательностей основаны на сложных рекуррентных или сверточных нейронных сетях, включающих энкодер и декодер. Наилучшие модели также связывают энкодер и декодер с помощью механизма внимания. Мы предлагаем новую простую архитектуру сети — Трансформер, основанный исключительно на механизмах внимания, полностью исключающий рекуррентность и свертки. Эксперименты по двум задачам машиного перевода показывают превосходство этих моделей по качеству при большей параллелизируемости и значительно меньшем времени обучения. Наша модель достигает 28,4 BLEU в задаче перевода с английского на немецкий языка WMT 2014, превосходя существующие лучшие результаты, включая ансамбли, более чем на 2 BLEU. В задаче перевода с английского на французский язык WMT 2014 наша модель устанавливает новый рекорд state-of-the-art для одиночных моделей со счетом BLEU 41,8 после 3,5 дней обучения на восьми GPU, что составляет малую долю от затрат на обучение лучших моделей из литературы. Мы показываем, что Трансформер хорошо обобщается на другие задачи благодаря успешному применению его для синтаксического разбора на constituency (английский язык) как с большими, так и с ограниченными обучающими данными.

***Равный вклад. Порядок перечисления произволен. Якоб предложил заменить рекуррентные нейронные сети (RNN) на механизм самовнимания и предпринял усилия для оценки этой идеи. Ашиш вместе с Иллией разработал и реализовал первые модели Transformer, а также был ключевым участником во всех аспектах данной работы. Ноам предложил масштабированное внимание на основе скалярного произведения, многоголовое внимание и параметрически независимое позиционное кодирование, став соучастником почти в каждой детали. Ники спроектировал, реализовал, настроил и протестировал бесчисленное множество вариантов моделей в нашем исходном коде, а также tensor2tensor. Ллион также экспериментировал с новыми вариантами архитектур, отвечал за наш первоначальный код, а также за эффективное инференс-вычисление и визуализацию. Лукаш и Эйдан провели无数个 долгих дней, разрабатывая и реализуя различные части tensor2tensor, заменив нашу предыдущую кодовую базу, что значительно улучшило результаты и существенно ускорило наши исследования.**

[†]Работа выполнена во время работы в Google Brain.

[‡]Работа выполнена в Google Research.

31-я конференция по системам нейронной обработки информации (NIPS 2017), Лонг-Бич, штат Калифорния, США

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

1 Introduction

Recurrent neural networks, long short-term memory [13] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation [35, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [38, 24, 15].

Recurrent models typically factor computation along the symbol positions of the input and output sequences. Aligning the positions to steps in computation time, they generate a sequence of hidden states h_t, as a function of the previous hidden state h_{t-1} and the input for position t. This inherently sequential nature precludes parallelization within training examples, which becomes critical at longer sequence lengths, as memory constraints limit batching across examples. Recent work has achieved significant improvements in computational efficiency through factorization tricks [21] and conditional computation [32], while also improving model performance in case of the latter. The fundamental constraint of sequential computation, however, remains.

Attention mechanisms have become an integral part of compelling sequence modeling and transduction models in various tasks, allowing modeling of dependencies without regard to their distance in the input or output sequences [2, 19]. In all but a few cases [27], however, such attention mechanisms are used in conjunction with a recurrent network.

In this work we propose the Transformer, a model architecture eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output. The Transformer allows for significantly more parallelization and can reach a new state of the art in translation quality after being trained for as little as twelve hours on eight P100 GPUs.

2 Background

The goal of reducing sequential computation also forms the foundation of the Extended Neural GPU [16], ByteNet [18] and ConvS2S [9], all of which use convolutional neural networks as basic building block, computing hidden representations in parallel for all input and output positions. In these models, the number of operations required to relate signals from two arbitrary input or output positions grows in the distance between positions, linearly for ConvS2S and logarithmically for ByteNet. This makes it more difficult to learn dependencies between distant positions [12]. In the Transformer this is reduced to a constant number of operations, albeit at the cost of reduced effective resolution due to averaging attention-weighted positions, an effect we counteract with Multi-Head Attention as described in section 3.2.

Self-attention, sometimes called intra-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence. Self-attention has been used successfully in a variety of tasks including reading comprehension, abstractive summarization, textual entailment and learning task-independent sentence representations [4, 27, 28, 22].

End-to-end memory networks are based on a recurrent attention mechanism instead of sequence-aligned recurrence and have been shown to perform well on simple-language question answering and language modeling tasks [34].

To the best of our knowledge, however, the Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or convolution. In the following sections, we will describe the Transformer, motivate self-attention and discuss its advantages over models such as [17, 18] and [9].

3 Model Architecture

Most competitive neural sequence transduction models have an encoder-decoder structure [5, 2, 35]. Here, the encoder maps an input sequence of symbol representations $(x_1, ..., x_n)$ to a sequence of continuous representations $\mathbf{z} = (z_1, ..., z_n)$. Given $\mathbf{z}$, the decoder then generates an output sequence $(y_1, ..., y_m)$ of symbols one element at a time. At each step the model is auto-regressive [10], consuming the previously generated symbols as additional input when generating the next.

1 Введение

Рекуррентные нейронные сети, в частности long short-term memory [13] и gating-рекуррентные нейронные сети [7], прочно закрепились в качестве передовых методов в задачах моделирования последовательностей и трансдукции, таких как языковое моделирование и машинный перевод [35, 2, 5]. С тех пор Numerous efforts have continued to push the boundaries of recurrent language models and encoder-decoder architectures [38, 24, 15].

Рекуррентные модели, как правило, разлагают вычисления вдоль позиций символов входной и выходной последовательностей. Сопоставляя позиции со шагами во времени вычислений, они генерируют последовательность скрытых состояний h_t в качестве функции предыдущего скрытого состояния h_{t-1} и входа для позиции t. Эта присущая им последовательная природа исключает параллелизацию внутри обучающих примеров, что становится критичным при больших длинах последовательностей, поскольку ограничения памяти ограничивают формирование батчей между примерами. Недавние работы достигли значительного повышения вычислительной эффективности за счет приемов факторизации [21] и условных вычислений [32], одновременно улучшая производительность модели в случае второго подхода. Фундаментальное ограничение последовательных вычислений, однако, сохраняется.

Механизмы внимания стали неотъемлемой частью убедительного моделирования и трансдукции последовательностей в различных задачах, позволяя учитывать зависимости без учета расстояния между элементами во входных или выходных последовательностях [2, 19]. Однако, за исключением нескольких случаев [27], такие механизмы внимания используются в сочетании с рекуррентными сетями.

В данной работе мы предлагаем Трансформер — архитектуру модели, исключающую рекуррентность и полностью полагающуюся на механизм внимания для установления глобальных зависимостей между входными и выходными данными. Трансформер обеспечивает значительно более высокую степень параллелизма и может достичь нового уровня передовых достижений в качестве перевода после обучения всего в течение двенадцати часов на восьми графических процессорах P100.

2. Фоновая информац ия

Цель сокращения последовательных вычислений также лежит в основе Extended Neural GPU [16], ByteNet [18] и ConvS2S [9], все из которых используют сверточные нейронные сети в качестве основного строительного блока, вычисляя скрытые представления параллельно для всех входных и выходных позиций. В этих моделях количество операций, необходимых для установления связи между сигналами из двух произвольных входных или выходных позиций, растет с расстоянием между позициями: линейно для ConvS2S и логарифмически для ByteNet. Это затрудняет обучение зависимостям между удаленными позициями [12]. В Трансформере это сводится к постоянному числу операций, albeit at the cost of reduced effective resolution due to averaging attention-weighted positions, an effect we counteract with Multi-Head Attention as described in section 3.2.

Self-attention, иногда называемый intra-attention, — это механизм внимания, устанавливающий связи между различными позициями в рамках одной последовательности для вычисления её представления. Self-attention успешно применяется в различных задачах, включая понимание прочитанного, абстрактное суммирование, текстовую энтайлмент и обучение представлению предложений, независимому от задачи [4, 27, 28, 22].

Конец-к-концу сети памяти основаны на рекуррентном механизме внимания вместо последовательностно-выровненной рекурренции и, как было показано, хорошо показывают себя в задачах ответов на вопросы с использованием простого языка и моделирования языка [34].

Однако, насколько нам известно, Transformer является первой моделью трансдукции, полностью полагающейся на механизм самовнимания для вычисления представлений входных и выходных данных без использования рекуррентных нейронных сетей или свертки. В последующих разделах мы опишем архитектуру Transformer, обоснуем необходимость применения самовнимания и обсудим его преимущества перед такими моделями, как [17, 18] и [9].

3 Архитектура модели

Большинство конкурентоспособных моделей нейронной последовательностной трансдукции имеют структуру «энкодер-декодер» [5, 2, 35]. Здесь энкодер отображает входную последовательность представлений символов $(x_1, \ldots, x_n)$ в последовательность непрерывных представлений $\mathbf{z}=(z_1, \ldots, z_n)$. Зная $\mathbf{z}$, декодер затем генерирует выходную последовательность $(y_1, \ldots, y_m)$ символов по одному элементу за раз. На каждом шаге модель является авто регрессивной [10], используя ранее сгенерированные символы в качестве дополнительного входа при генерации следующего.

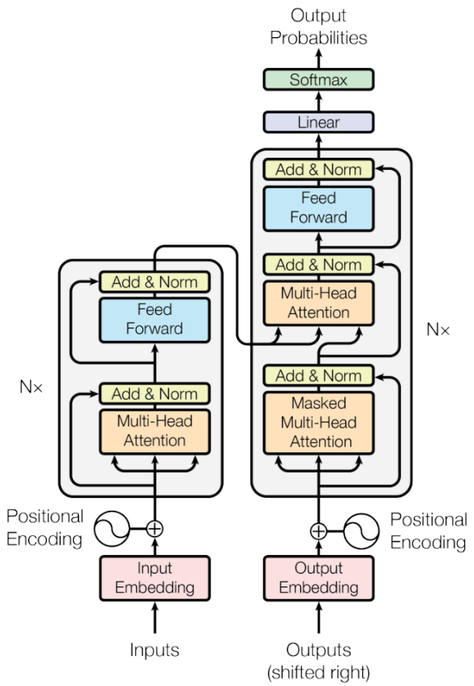


Figure 1: The Transformer - model architecture.

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

3.1 Encoder and Decoder Stacks

Encoder: The encoder is composed of a stack of $N = 6$ identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. We employ a residual connection [11] around each of the two sub-layers, followed by layer normalization [1]. That is, the output of each sub-layer is $\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x))$, where $\mathrm{Sublayer}(x)$ is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension $d_{\text{model}} = 512$.

Decoder: The decoder is also composed of a stack of $N = 6$ identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, we employ residual connections around each of the sub-layers, followed by layer normalization. We also modify the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with fact that the output embeddings are offset by one position, ensures that the predictions for position i can depend only on the known outputs at positions less than i.

3.2 Attention

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

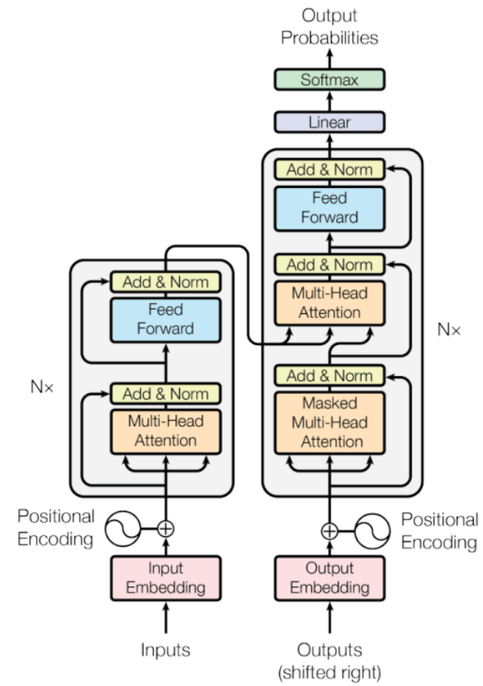


Рисунок 1: Архитектура модели Transformer.

Трансформер следует этой общей архитектуре, используя многоуровневое само внимание и точечно-полносвязные слои как для энкодера, так и для декодера, показанных в левой и правой частях Рисунка 1 соответственно.

3.1 Стек энкодера и декодера

Кодировщик: Кодировщик состоит из стека из $N = 6$ идентичных слоев. Каждый слой имеет два подслоя. Первый — это механизм многоголовного само внимания, а второй — простая поэлементно полностью связная сеть прямого распространения. Мы используем остаточное соединение [11] вокруг каждого из двух подслоев, за которым следует нормализация слоя [1]. То есть выход каждого подслоя равен $\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x))$, где $\mathrm{Sublayer}(x)$ — функция, реализованная самим подслоем. Для облегчения реализации этих остаточных соединений все подслои в модели, а также слои внедрения (эмбеддингов), выдают выходы размерности $d_{\text{модели}} = 512$.

Декодер: Декодер также состоит из стека из $N = 6$ идентичных слоев. В дополнение к двум подслоям в каждом слое энкодера, декодер вставляет третий подслой, который выполняет многоголовое внимание по выходу стека энкодера. Как и в энкодере, мы применяем остаточные соединения вокруг каждого из подслоев, за которыми следует нормализация слоя. Мы также модифицируем подслой само-внимания в стеке декодера, чтобы предотвратить внимание позициям на последующих позициях. Это маскирование, в сочетании с тем фактом, что выходные эмбеддинги смещены на одну позицию, обеспечивает, что предсказания для позиции i могут зависеть только от известных выходов на позициях, меньших i.

3.2 Внимание

Функцию внимания можно описать как отображение, преобразующее запрос и набор пар «ключ-значение» в результат, при этом запрос, ключи, значения и результат представляют собой векторы. Результат вычисляется как взвешенная сумма

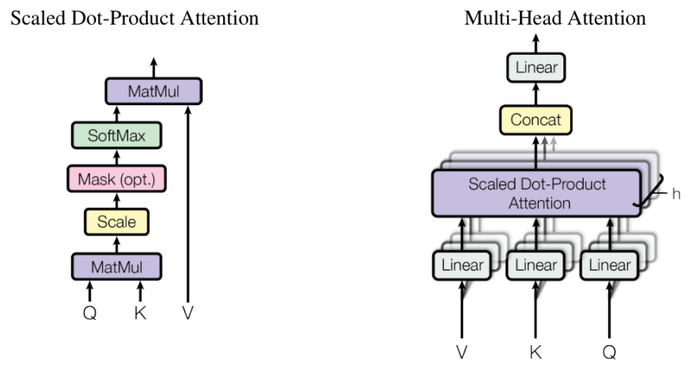


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

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

3.2.1 Scaled Dot-Product Attention

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

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

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

The two most commonly used attention functions are additive attention [2], and dot-product (multiplicative) attention. Dot-product attention is identical to our algorithm, except for the scaling factor of $\frac{1}{\sqrt{d_k}}$. Additive attention computes the compatibility function using a feed-forward network with a single hidden layer. While the two are similar in theoretical complexity, dot-product attention is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code.

While for small values of d_k the two mechanisms perform similarly, additive attention outperforms dot product attention without scaling for larger values of d_k [3]. We suspect that for large values of d_k, the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients [4]. To counteract this effect, we scale the dot products by $\frac{1}{\sqrt{d_k}}$.

3.2.2 Multi-Head Attention

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

[4] To illustrate why the dot products get large, assume that the components of q and k are independent random variables with mean 0 and variance 1. Then their dot product, $q \cdot k = \sum_{i=1}^{d_k} q_i k_i$, has mean 0 and variance d_k.

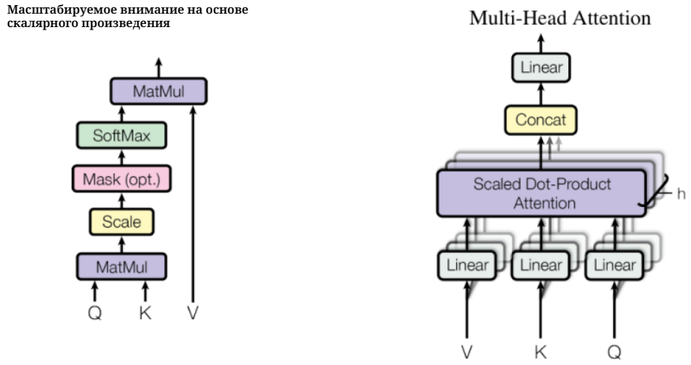


Рисунок 2: (слева) Масштабированное скалярное произведение внимания. (справа) Многоголовое внимание состоит из нескольких слоев внимания, работающих параллельно.

значений, где вес, присвоенный каждому значению, вычисляется с помощью функции совместимости запроса с соответствующим ключом.

3.2.1 Масштабируемое внимание на основе скалярного произведения

Мы называем наше конкретное внимание "Масштабируемым скалярным произведением внимания" (Рисунок 2). Входные данные состоят из запросов и ключей размерности d_k и значений размерности d_v. Мы вычисляем скалярные произведения запроса со всеми ключами, делим каждое на $\sqrt{d_k}$ и применяем функцию softmax для получения весов по значениям.

На практике мы вычисляем функцию внимания для набора запросов одновременно, упаковывая их в матрицу Q. Ключи и значения также объединяются в матрицы K и V. Матрицу outputs вычисляем как:

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

Два наиболее часто используемых типа функций внимания — это аддитивное внимание [2] и внимание на основе скалярного произведения (мультипликативное). Внимание на основе скалярного произведения идентично нашему алгоритму, за исключением множителя масштабирования $\frac{1}{\sqrt{d_k}}$. Аддитивное внимание вычисляет функцию совместимости с помощью сети прямого распространения с одним скрытым слоем. Хотя оба подхода схожи по теоретической сложности, внимание на основе скалярного произведения на практике работает значительно быстрее и требует меньше памяти, поскольку его можно реализовать с использованием высокооптимизированного кода для умножения матриц.

Хотя для малых значений d_k оба механизма показывают схожие результаты, аддитивное внимание превосходит внимание по скалярному произведению без масштабирования при больших значениях d_k [3]. Мы предполагаем, что при больших значениях d_k скалярные произведения становятся большими по модулю, выталкивая функцию softmax в области, где она имеет чрезвычайно малые градиенты 4. Чтобы противодействовать этому эффекту, мы масштабируем скалярные произведения на $\frac{1}{\sqrt{d_k}}$.

3.2.2 Многоголовое внимание

Вместо выполнения одной функции внимания с ключами, значениями и запросами размерности d_{model}, мы сочли целесообразным линейно спроецировать запросы, ключи и значения h раз с помощью различных обучаемых линейных проекций в размерности d_k, d_k и d_v соответственно. Для каждой из этих спроецированных версий запросов, ключей и значений мы затем выполняем функцию внимания параллельно, что дает результат размерности d_v.

[4] Для иллюстрации того, почему скалярные произведения становятся большими, предположим, что компоненты q и k являются независимыми случайными величинами со средним значением 0 и дисперсией 1. Тогда их скалярное произведение, $q \cdot k = \sum_{i=1} q_i k_i$, имеет среднее значение 0 и дисперсию d_k.

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

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

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

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_\text{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_\text{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_\text{model} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_\text{model}}$.

In this work we employ $h = 8$ parallel attention layers, or heads. For each of these we use $d_k = d_v = d_\text{model}/h = 64$. Due to the reduced dimension of each head, the total computational cost is similar to that of single-head attention with full dimensionality.

3.2.3 Applications of Attention in our Model

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

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

3.3 Position-wise Feed-Forward Networks

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

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

While the linear transformations are the same across different positions, they use different parameters from layer to layer. Another way of describing this is as two convolutions with kernel size 1. The dimensionality of input and output is $d_\text{model} = 512$, and the inner-layer has dimensionality $d_{ff} = 2048$.

3.4 Embeddings and Softmax

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

выходные значения. Они объединяются и снова проецируются, в результате чего получаются окончательные значения, как показано на Рисунке 2.

Многоголовое внимание позволяет модели совместно фокусироваться на информации из различных подпространств представлений в разных позициях. При использовании одного внимания-головы усреднение подавляет этот механизм.

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

где проекционные матрицы $W_i^Q \in \mathbb{R}^{d_\text{модель} \times d_k}$, $W_i^K \in \mathbb{R}^{d_\text{модель} \times d_k}$, $W_i^V \in \mathbb{R}^{d_\text{модель} \times d_v}$ **и** $W^O \in \mathbb{R}^{hd_v \times d_\text{модель}}$.

В данной работе мы используем $h = 8$ **параллельных слоев внимания, или голов. Для каждой из них мы применяем** $d_k = d_v = d_\text{модель}/h = 64$. **В связи со сниженной размерностью каждой головы, общие вычислительные затраты сопоставимы с таковыми для одноголовного внимания с полной размерностью.**

3.2.3 Применение механизма внимания в нашей модели

Трансформер использует многоголовое внимание тремя различными способами:

В слоях «внимания энкодер-декодер» запросы поступают из предыдущего слоя декодера, а ключи памяти и значения — из выхода энкодера. Это позволяет каждой позиции в декодере обращать внимание на все позиции входной последовательности. Это имитирует типичные механизмы внимания энкодер-декодер в моделях последовательность-к-последовательности, таких как [38, 2, 9].

Энкодер содержит слои само-внимания. В слое само-внимания все ключи, значения и запросы поступают из одного источника, в данном случае — из выхода предыдущего слоя энкодера. Каждая позиция в энкодере может взаимодействовать со всеми позициями в предыдущем слое энкодера.

Аналогично, слои самовнимания в декодере позволяют каждому положению в декодере обращать внимание на все позиции до этой включительно. Мы должны предотвратить поток информации слева направо в декодере для сохранения авто регрессивного свойства. Мы реализуем это внутри масштабированного поэлементного произведения внимания путем маскирования (установки значения $-\infty$**) всех значений во входных данных функции softmax, которые соответствуют недопустимым связям. См. Рисунок 2.**

3.3 Позиционно-ориентированные сети прямого распространения

Помимо подслоев внимания, каждый из слоев в нашем энкодере и декодере содержит полностью связную прямопроходную сеть, которая применяется к каждой позиции отдельно и идентичным образом. Она состоит из двух линейных преобразований с активацией ReLU между ними.

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

Хотя линейные преобразования одинаковы для различных позиций, они используют различные параметры от слоя к слою. Другой способ описать это — как две свертки с размером ядра 1. Размерность входа и выхода составляет $d_\text{model} = 512$, **а размерность внутреннего слоя —** $d_{ff} = 2048$.

3.4 Вложения и Softmax

Аналогично другим моделям трансдукции последовательностей, мы используем обученные эмбеддинги для преобразования входных и выходных токенов в векторы размерности d_model. **Мы также применяем обычную обученную линейную трансформацию и функцию softmax для преобразования выхода декодера в предсказанные вероятности следующих токенов. В нашей модели мы используем одну и ту же матрицу весов между двумя слоями эмбеддингов и пред-softmax линейной трансформацией, аналогично [30]. В слоях эмбеддингов мы умножаем эти веса на** $\sqrt{d_\text{model}}$.

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

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

3.5 Positional Encoding

Since our model contains no recurrence and no convolution, in order for the model to make use of the order of the sequence, we must inject some information about the relative or absolute position of the tokens in the sequence. To this end, we add "positional encodings" to the input embeddings at the bottoms of the encoder and decoder stacks. The positional encodings have the same dimension d_{model} as the embeddings, so that the two can be summed. There are many choices of positional encodings, learned and fixed [9].

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

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

where pos is the position and i is the dimension. That is, each dimension of the positional encoding corresponds to a sinusoid. The wavelengths form a geometric progression from 2π to $10000 \cdot 2\pi$. We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset k, PE_{pos+k} can be represented as a linear function of PE_{pos}.

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

4 Why Self-Attention

In this section we compare various aspects of self-attention layers to the recurrent and convolutional layers commonly used for mapping one variable-length sequence of symbol representations $(x_1, ..., x_n)$ to another sequence of equal length $(z_1, ..., z_n)$, with $x_i, z_i \in \mathbb{R}^d$, such as a hidden layer in a typical sequence transduction encoder or decoder. Motivating our use of self-attention we consider three desiderata.

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

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

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

Таблица 1: Максимальные длины путей, сложность на слой и минимальное количество последовательных операций для различных типов слоев. n — длина последовательности, d — размерность представления, k — размер ядра свертки, а r — размер окрестности в ограниченном механизме само внимания.

Тип слоя	Сложность на слой	Последовательный Операции	Максимальная длина пути
Самовнимание	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Рекуррентный	$O(n \cdot d^2)$	null	$O(n)$
Сверточный	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Самовнимание (ограниченное)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

3.5 Позиционное кодирование

Поскольку наша модель не содержит рекуррентных и сверточных операций, для того чтобы модель могла учитывать порядок последовательности, необходимо добавить информацию о относительном или абсолютном положении токенов в последовательности. С этой целью мы добавляем «позиционные кодировки» к входным эмбеддингам внизу стеков энкодера и декодера. Позиционные кодировки имеют ту же размерность d_{model}, что и эмбеддинги, чтобы их можно было складывать. Существует множество вариантов позиционных кодировок: как обучаемые, так и фиксированные [9].

В данной работе мы используем синусоидальные и косинусоидальные функции различных частот:

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

где pos — позиция, а i — размерность. Таким образом, каждая размерность позиционного кодирования соответствует синусоиде. Длины волн образуют геометрическую прогрессию от 2π до $10000 \cdot 2\pi$. Мы выбрали эту функцию, потому что предположили, что она позволит модели легко обучаться обращать внимание на относительные позиции, поскольку для любого фиксированного смещения k, PE_{pos+k} может быть представлено как линейная функция от PE_{pos}.

Мы также экспериментировали с использованием обученных позиционных вложений [9] вместо них и обнаружили, что обе версии дают практически идентичные результаты (см. Таблицу 3, строка (E)). Мы выбрали синусоидальную версию, поскольку она может позволить модели экстраполировать до длин последовательностей, превышающих те, которые встречались во время обучения.

4 Почему само-внимание

В этом разделе мы сравниваем различные аспекты слоев само-внимания со рекуррентными и сверточными слоями, которые обычно используются для преобразования одной последовательности представлений символов переменной длины $(x_1, \ldots, x_n)$ в другую последовательность той же длины $(z_1, \ldots, z_n)$, где $x_i, z_i \in \mathbb{R}^d$, например, в скрытом слое типичного энкодера или декодера трансдукции последовательностей. Мотивируя наше использование само-внимания, мы рассматриваем три предпочтительных требования.

Одним является общая вычислительная сложность на слой. Другим — объем вычислений, которые могут быть распараллелены, что измеряется минимальным количеством последовательных операций, необходимых для выполнения.

Третьим является длина пути между дальнодействующими зависимостями в сети. Обучение дальнодействующим зависимостям представляет собой ключевую задачу во многих задачах трансдукции последовательностей. Одним из основных факторов, влияющих на способность к обучению таким зависимостям, является длина путей, которые должны пройти прямые и обратные сигналы в сети. Чем короче эти пути между любой комбинацией позиций во входной и выходной последовательностях, тем легче обучаться дальнодействующим зависимостям [12]. Следовательно, мы также сравниваем максимальную длину пути между любыми двумя позициями входа и выхода в сетях, состоящих из различных типов слоев.

Как отмечено в Таблице 1, слой само-внимания связывает все позиции с постоянным количеством последовательно выполняемых операций, в то время как рекуррентный слой требует $O(n)$ последовательных операций. С точки зрения вычислительной сложности, слои само-внимания работают быстрее рекуррентных слоев, когда последовательность

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

A single convolutional layer with kernel width $k < n$ does not connect all pairs of input and output positions. Doing so requires a stack of $O(n/k)$ convolutional layers in the case of contiguous kernels, or $O(log_k(n))$ in the case of dilated convolutions [18], increasing the length of the longest paths between any two positions in the network. Convolutional layers are generally more expensive than recurrent layers, by a factor of k. Separable convolutions [6], however, decrease the complexity considerably, to $O(k \cdot n \cdot d + n \cdot d^2)$. Even with $k = n$, however, the complexity of a separable convolution is equal to the combination of a self-attention layer and a point-wise feed-forward layer, the approach we take in our model.

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

5 Training

This section describes the training regime for our models.

5.1 Training Data and Batching

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

5.2 Hardware and Schedule

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

5.3 Optimizer

We used the Adam optimizer [20] with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. We varied the learning rate over the course of training, according to the formula:

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

This corresponds to increasing the learning rate linearly for the first $warmup_steps$ training steps, and decreasing it thereafter proportionally to the inverse square root of the step number. We used $warmup_steps = 4000$.

5.4 Regularization

We employ three types of regularization during training:

длина n меньше размерности представления d, что наиболее часто встречается в representations предложений, используемых в передовых моделях машинного перевода, таких как word-piece [38] и byte-pair [31]. Для повышения вычислительной эффективности при решении задач, связанных с очень длинными последовательностями, механизм self-attention может быть ограничен рассмотрением только окрестности размером r во входной последовательности, центрированной вокруг соответствующей выходной позиции. Это увеличит максимальную длину пути до $O(n/r)$. Мы планируем дальнейшее исследование данного подхода в будущих работах.

Один свертывающий слой с шириной ядра $k<n$ не связывает все пары входных и выходных позиций. Для этого требуется стек из $O(n/k)$ свертывающих слоев в случае непрерывных ядер или $O(log_k(n))$ в случае разреженных сверток [18], что увеличивает длину наивысших путей между любыми двумя позициями в сети. Сворачивающие слои, как правило, требуют больше вычислительных ресурсов, чем рекуррентные слои, в k раз. Раздельные свертки [6], однако, значительно снижают сложность — до $O(k \cdot n \cdot d + n \cdot d^2)$. Даже при $k=n$, сложность раздельной свертки равна комбинации слоя само-внимания и послойного полносвязного слоя, подход, который мы используем в нашей модели.

В качестве дополнительного преимущества, механизм самовнимания может обеспечивать более интерпретируемые модели. Мы анализируем распределения внимания в наших моделях и приводим, а также обсуждаем примеры в приложении. Не только отдельные головы внимания явно обучаются выполнению различных задач, но многие из них, по-видимому, демонстрируют поведение, связанное с синтаксической и семантической структурой предложений.

5 Обучение

В данном разделе описывается режим обучения наших моделей.

5.1 Обучающие данные и пакетная обработка

Мы обучались на стандартном наборе данных WMT 2014 «английско-немецкий», состоящем из примерно 4,5 миллионов пар предложений. Предложения были закодированы с использованием байт-парного кодирования [3], которое имеет общий словарь исходного и целевого языков объемом около 37 000 токенов. Для пары «английско-французский» мы использовали значительно более крупный набор данных WMT 2014, состоящий из 36 миллионов предложений, и разбили токены на словарь word-piece объемом 32 000 единиц [38]. Пары предложений группировались в батчи приблизительно по длине последовательности. Каждый обучающий батч содержал набор пар предложений, включающий примерно 25 000 исходных токенов и 25 000 целевых токенов.

5.2 Аппаратное обеспечение и расписание

Мы обучали наши модели на одной машине с 8 графическими процессорами NVIDIA P100. Для наших базовых моделей, использующих гиперпараметры, описанные в данной работе, каждый шаг обучения занимал около 0,4 секунды. Базовые модели обучались в общей сложности 100 000 шагов или 12 часов. Для наших больших моделей (описанных в последней строке таблицы 3) время одного шага составило 1,0 секунды. Большие модели обучались 300 000 шагов (3,5 дня).

5.3 Оптимизатор

Мы использовали оптимизатор Adam [20] со значениями $\beta_1 = 0.9$, $\beta_2 = 0.98$ и $\varepsilon = 10^{-9}$. В ходе обучения мы изменяли скорость обучения согласно формуле:

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

Это соответствует линейному увеличению скорости обучения в течение первых warmup_steps шагов обучения, а последующему его пропорциональному уменьшению обратно квадратному корню из номера шага. Мы использовали warmup_steps = 4000.

5.4 Регуляризация

Мы используем три вида регуляризации во время обучения:

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

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Residual Dropout We apply dropout [33] to the output of each sub-layer, before it is added to the sub-layer input and normalized. In addition, we apply dropout to the sums of the embeddings and the positional encodings in both the encoder and decoder stacks. For the base model, we use a rate of $P_{drop} = 0.1$.

Label Smoothing During training, we employed label smoothing of value $\epsilon_{ls} = 0.1$ [36]. This hurts perplexity, as the model learns to be more unsure, but improves accuracy and BLEU score.

6 Results

6.1 Machine Translation

On the WMT 2014 English-to-German translation task, the big transformer model (Transformer (big) in Table 2) outperforms the best previously reported models (including ensembles) by more than 2.0 BLEU, establishing a new state-of-the-art BLEU score of 28.4. The configuration of this model is listed in the bottom line of Table 3. Training took 3.5 days on 8 P100 GPUs. Even our base model surpasses all previously published models and ensembles, at a fraction of the training cost of any of the competitive models.

On the WMT 2014 English-to-French translation task, our big model achieves a BLEU score of 41.0, outperforming all of the previously published single models, at less than $1/4$ the training cost of the previous state-of-the-art model. The Transformer (big) model trained for English-to-French used dropout rate $P_{drop} = 0.1$, instead of 0.3.

For the base models, we used a single model obtained by averaging the last 5 checkpoints, which were written at 10-minute intervals. For the big models, we averaged the last 20 checkpoints. We used beam search with a beam size of 4 and length penalty $\alpha = 0.6$ [38]. These hyperparameters were chosen after experimentation on the development set. We set the maximum output length during inference to input length + 50, but terminate early when possible [38].

Table 2 summarizes our results and compares our translation quality and training costs to other model architectures from the literature. We estimate the number of floating point operations used to train a model by multiplying the training time, the number of GPUs used, and an estimate of the sustained single-precision floating-point capacity of each GPU [5].

6.2 Model Variations

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

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

Таблица 2: Трансформер показывает лучшие результаты по метрике BLEU по сравнению с предыдущими передовыми моделями на тестах English-to-German и English-to-French newstest2014 при значительно меньших затратах на обучение.

Модель	BLEU		Стоимость обучения (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**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Трансформер (базовая модель)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Трансформер (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Residual Dropout Применяем dropout [33] к выходу каждого подслоя до того, как он добавляется ко входу подслоя и нормализуется. Кроме того, применяем dropout к суммам эмбеддингов и позиционных кодирований как в стеках энкодера, так и декодера. Для базовой модели используем вероятность отключения $P_{drop} = 0.1$.

Сглаживание меток Во время обучения мы применяли сглаживание меток со значением $\epsilon_{ls} = 0.1$ [36]. Это ухудшает перплексивность, поскольку модель учится быть более неуверенной, но повышает точность и показатель BLEU.

6 Результаты

6.1 Машинный перевод

На задаче машиного перевода с английского на немецкий языка WMT 2014 большая модель трансформера (Transformer (big) в Таблице 2) превосходит лучшие ранее опубликованные модели (включая ансамбли) более чем на 2.0 BLEU, устанавливая новый рекорд по показателю BLEU на уровне 28.4. Конфигурация данной модели приведена в последней строке Таблицы 3. Обучение заняло 3.5 дней на 8 видеоускорителях P100. Даже наша базовая модель превосходит все ранее опубликованные модели и ансамбли при значительно меньших затратах на обучение по сравнению с конкурентными моделями.

На задаче машинного перевода с английского на французский языка WMT 2014 наша большая модель достигла показателя BLEU, равного 41,0, превзойдя все ранее опубликованные одномерные модели при затратах на обучение, составляющих менее 1/4 от затрат предыдущей передовой модели. Модель Transformer (большая), обученная для перевода с английского на французский, использовала ставку dropout $P_{drop} = 0.1$ вместо 0.3.

Для базовых моделей мы использовали одну модель, полученную путем усреднения последних 5 контрольных точек, которые создавались с интервалом в 10 минут. Для больших моделей мы усредняли последние 20 контрольных точек. Мы применяли поиск лучами с размером луча 4 и штрафом за длину $\alpha = 0.6$ [38]. Эти гиперпараметры были выбраны после экспериментов на наборе для разработки. Мы установили максимальную длину вывода во время инференса равной длине ввода + 50, но досрочно завершали процесс, когда это возможно [38].

Таблица 2 обобщает наши результаты и сравнивает качество нашего перевода и затраты на обучение с другими архитектурами моделей из литературы. Мы оцениваем количество операций с плавающей точкой, использованных при обучении модели, умножая время обучения, количество использованных видеокарт (GPU) и оценку производительности каждой видеокарты в режиме непрерывной работы с одинарной точностью 5.

6.2 Варианты модели

Чтобы оценить важность различных компонентов Трансформера, мы изменяли нашу базовую модель различными способами, измеряя изменения в производительности при переводе с английского на немецкий языка на

[5]Мы использовали значения 2,8, 3,7, 6,0 и 9,5 ТФЛОПС для K80, K40, M40 и P100 соответственно.

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

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

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

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

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

6.3 English Constituency Parsing

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

We trained a 4-layer transformer with $d_{model} = 1024$ on the Wall Street Journal (WSJ) portion of the Penn Treebank [25], about 40K training sentences. We also trained it in a semi-supervised setting, using the larger high-confidence and BerkleyParser corpora from with approximately 17M sentences [37]. We used a vocabulary of 16K tokens for the WSJ only setting and a vocabulary of 32K tokens for the semi-supervised setting.

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

Таблица 3: Варианты архитектуры Transformer. Неуказанные значения идентичны таковым базовой модели. Все метрики приведены для набора разработки English-to-German translation, newstest2013. Указанные значения перплексии рассчитаны на единицу wordpiece согласно нашей байт-парной кодировке и не должны сравниваться с перплексией на слово.

	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)	позиционная вставка вместо синусоидальных									4.92	25.7	
big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

набор для разработки, newstest2013. Мы использовали поиск лучом, как описано в предыдущем разделе, но без усреднения по контрольным точкам. Представим эти результаты в Таблице 3.

В строках (A) Таблицы 3 мы варьируем количество голов внимания, а также размерности ключа и значения внимания, сохраняя объем вычислений постоянным, как описано в Разделе 3.2.2. Хотя одноголовое внимание уступает лучшему варианту на 0,9 по метрике BLEU, качество также снижается при слишком большом количестве голов.

В строках (B) Таблицы 3 мы наблюдаем, что уменьшение размера ключа внимания d_k ухудшает качество модели. Это указывает на то, что определение совместимости является непростой задачей и что более сложная функция совместимости, чем скалярное произведение, может быть предпочтительнее. В строках (C) и (D) мы further observed, что, как и ожидалось, более крупные модели работают лучше, а dropout очень помогает в предотвращении переобучения. В строке (E) мы заменяем нашу синусоидальную позиционную кодировку на обучаемые позиционные эмбеддинги [9] и получаем результаты, практически идентичные базовой модели.

6.3 Синтаксический анализ английского языка (разбор на Constituency)

Чтобы оценить, может ли Трансформер обобщаться на другие задачи, мы провели эксперименты по синтаксическому разбору фраз (constituency parsing) на английском языке. Эта задача представляет собой определенные сложности: выходная последовательность подчиняется строгим структурным ограничениям и значительно превышает входную по длине. Кроме того, рекуррентные нейронные сети последовательность-в-последовательность (RNN sequence-to-sequence) не смогли достичь наилучших современных результатов в условиях ограниченного объема данных [37].

Мы обучили 4-слойный трансформер с $d_{model} = 1024$ на части Wall Street Journal (WSJ) из Penn Treebank [25], comprising около 40K обучающих предложений. Мы также обучили его в полу-supervised режиме, используя более крупные корпуса high-confidence и BerkleyParser с примерно 17M предложениями [37]. Мы использовали словарь из 16K токенов для режима только WSJ и словарь из 32K токенов для полу-supervised режима.

Мы провели лишь небольшое количество экспериментов по подбору dropout, как attention, так и residual (раздел 5.4), а также скоростей обучения и размера пучка на наборе данных для разработки в Разделе 22; все остальные параметры остались неизменными по сравнению с базовой моделью перевода с английского на немецкий. Во время инференса мы

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

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

increased the maximum output length to input length + 300. We used a beam size of 21 and $\alpha = 0.3$ for both WSJ only and the semi-supervised setting.

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

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

7 Conclusion

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

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

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

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

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

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

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

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

[4] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016.

Таблица 4: Трансформатор хорошо обобщается на синтаксический анализ constituent-структуры английского языка (Результаты представлены в Разделе 23 WSJ)

Парсер	Обучение	WSJ 23 F1
Vinyals и Kaiser et al. (2014) [37]	Только WSJ, дискриминативный	88,3
Petrov et al. (2006) [29]	Только WSJ, дискриминативный	90,4
Zhu et al. (2013) [40]	Только WSJ, дискриминативный	90,4
Dyer et al. (2016) [8]	Только WSJ, дискриминативный	91,7
Трансформер (4 слоя)	Только WSJ, дискриминативный	91,3
Zhu et al. (2013) [40]	полусупервизорный	91,3
Huang & Harper (2009) [14]	полусупервизорный	91,3
McClosky et al. (2006) [26]	полусупервизорный	92,1
Vinyals и Kaiser et al. (2014) [37]	полусупервизорный	92,1
Трансформер (4 слоя)	полусупервизорный	92,7
Luong et al. (2015) [23]	многозадачный	93,0
Dyer et al. (2016) [8]	генеративный	93,3

увеличили максимальную длину вывода до длины ввода + 300. Для обоих режимов — только WSJ и полу-supervised — мы использовали размер пучка 21 и α=0.3.

Наши результаты в Таблице 4 показывают, что, несмотря на отсутствие специфической для задач настройки, наша модель демонстрирует удивительно высокие результаты, уступая по показателям всем ранее опубликованным моделям, за исключением Рекуррентной Нейронной Сетевой Грамматики [8].

В отличие от рекуррентных нейронных сетей последовательность-к-последовательности [37], Трансформатор превосходит Berkeley-Parser [29] даже при обучении только на обучающем наборе WSJ из 40 000 предложений.

7 Заключение

В данной работе мы представили Трансформер — первую модель трансдукции последовательностей, полностью основанную на механизме внимания, которая заменяет рекуррентные слои, наиболее часто используемые в архитектурах «энкодер-декодер», многоголовным само вниманием.

Для задач перевода модель Transformer может обучаться значительно быстрее, чем архитектуры на основе рекуррентных или сверточных слоев. На обеих задачах перевода WMT 2014 (английский-немецкий и английский-французский) мы достигли нового рекорда. В первой задаче наша лучшая модель превосходит даже все ранее опубликованные ансамбли.

Мы с воодушевлением смотрим в будущее моделей, основанных на механизме внимания, и планируем применять их для решения других задач. Мы намерены расширить возможности Трансформатора на задачи,涉及 входные и выходные модальности, отличные от текста, а также исследовать локальные ограниченные механизмы внимания для эффективной обработки больших объемов входных и выходных данных, таких как изображения, аудио и видео. Снижение степени последовательности генерации является еще одной нашей исследовательской целью.

Код, который мы использовали для обучения и оценки наших моделей, доступен по адресу: https://github.com/tensorflow/tensor2tensor.

Благодарности Мы благодарим Нала Кальхбреннера и Стефана Гауса за их плодотворные комментарии, исправления и вдохновение.

Список литературы

[1] Джимми Лей Ба, Джейми Райан Кирос и Джеффри Э. Хинтон. Слойная нормализация. arXiv preprint arXiv:1607.06450, 2016.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Нейронный машинный перевод путем совместного обучения выравниванию и переводу. CoRR, abs/1409.0473, 2014.

[3] Денни Бритц, Анна Голди, Минь-Тханг Луонг и Куок В. Ле. Масштабное исследование архитектур нейронного машинного перевода. CoRR, abs/1703.03906, 2017.

[4] Цзянпэн Чэн, Ли Дун и Мирелла Лапарта. Долгосрочные краткосрочные сети памяти для машинного чтения. arXiv preprint arXiv:1601.06733, 2016.

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

[6] Francois Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357*, 2016.

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

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

[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122v2*, 2017.

[10] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

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

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

[14] Zhongqiang Huang and Mary Harper. Self-training PCFG grammars with latent annotations across languages. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 832–841. ACL, August 2009.

[15] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

[16] Łukasz Kaiser and Samy Bengio. Can active memory replace attention? In *Advances in Neural Information Processing Systems, (NIPS)*, 2016.

[17] Łukasz Kaiser and Ilya Sutskever. Neural GPUs learn algorithms. In *International Conference on Learning Representations (ICLR)*, 2016.

[18] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099v2*, 2017.

[19] Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. Structured attention networks. In *International Conference on Learning Representations*, 2017.

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

[21] Oleksii Kuchaiev and Boris Ginsburg. Factorization tricks for LSTM networks. *arXiv preprint arXiv:1703.10722*, 2017.

[22] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

[23] Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*, 2015.

[24] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk и Yoshua Bengio. Обучение представлений фраз с использованием RNN-кодера-декодера для статистического машинного перевода. CoRR, abs/1406.1078, 2014.

[6] Франсуа Шолле. Xception: Глубокое обучение с помощью поэлементно-раздельных сверток. arXiv preprint arXiv:1610.02357, 2016.

[7] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Эмпирическая оценка затворенных рекуррентных нейронных сетей для моделирования последовательностей. CoRR, abs/1412.3555, 2014.

[8] Крис Дайер, Адхигуна Кункоро, Мигель Бальестерос и Ноа А. Смит. Рекуррентные нейронные сетевые грамматики. В: Proc. of NAACL, 2016.

[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, и Yann N. Dauphin. Сверточная последовательность к последовательности: обучение. arXiv preprint arXiv:1705.03122v2, 2017.

[10] Alex Graves. Генерация последовательностей с помощью рекуррентных нейронных сетей. arXiv preprint arXiv:1308.0850, 2013.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, и Jian Sun. Глубокое остаточное обучение для распознавания изображений. В: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, страницы 770 – 778, 2016.

[12] Сепп Хохрейтер, Йошуа Бенджо, Паоло Фраскони и Юрген Шмидхубер. Поток градиента в рекуррентных сетях: сложность обучения долгосрочным зависимостям, 2001.

[13] Sepp Hochreiter и Юрген Шмидхубер. Долгосрочная краткосрочная память. Neural computation, 9(8):1735 – 1780, 1997.

[14] Чжунцян Хуанг и Мэри Харпер. Самообучающиеся грамматики PCFG с латентными аннотациями для различных языков. В: Сборник материалов конференции 2009 года по эмпирическим методам в обработке естественного языка, страницы 832 – 841. ACL, август 2009 г.

[15] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer и Yonghui Wu. Исследование границ языкового моделирования. arXiv preprint arXiv:1602.02410, 2016.

[16] Łukasz Kaiser и Самми Бенджо. Может ли активная память заменить внимание? В: Advances in Neural Information Processing Systems, (NIPS), 2016.

[17] Łukasz Kaiser и Илья Сутскевер. Нейронные графические процессоры осваивают алгоритмы. В: International Conference on Learning Representations (ICLR), 2016.

[18] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves и Koray Kavukcuoglu. Нейронный машинный перевод за линейное время. arXiv preprint arXiv:1610.10099v2, 2017.

[19] Йон Ким, Карл Дентон, Луонг Хоанг и Александер М. Раш. Структурные сети внимания. В: Международная конференция по представлению знаний, 2017.

[20] Дидерик Кингма и Джимми Ба. Adam: метод стохастической оптимизации. В ICLR, 2015.

[21] Олексий Кучаев и Борис Гинсбург. Методы факторизации для сетей LSTM. arXiv preprint arXiv:1703.10722, 2017.

[22] Чжоуань Лин, Минвэй Фэн, Сіцеро Ногейра дус Сантуш, Мо Ю, Бин Сиян, Боувэн Чжоу и Йосуа Бенджо. Структурированное самовнимательное вложение предложений. arXiv preprint arXiv:1703.03130, 2017.

[23] Минь-Тханг Луонг, Куок В. Ле, Илья Сутскевер, Ориол Виньялс и Лукаш Кайзер. Множественное задание: обучение последовательность к последовательности. arXiv preprint arXiv:1511.06114, 2015.

[24] Минь-Тханг Луонг, Хьеу Фам и Кристофер Д. Мэннинг. Эффективные подходы к нейромашинному переводу на основе внимания. arXiv preprint arXiv:1508.04025, 2015.

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

[26] David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159. ACL, June 2006.

[27] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model. In *Empirical Methods in Natural Language Processing*, 2016.

[28] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.

[29] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 433–440. ACL, July 2006.

[30] Ofir Press and Lior Wolf. Using the output embedding to improve language models. *arXiv preprint arXiv:1608.05859*, 2016.

[31] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[32] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

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

[34] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2440–2448. Curran Associates, Inc., 2015.

[35] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

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

[37] Vinyals & Kaiser, Koo, Petrov, Sutskever, and Hinton. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, 2015.

[38] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

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

[40] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the ACL (Volume 1: Long Papers)*, pages 434–443. ACL, August 2013.

[25] Митчелл П. Маркус, Мэри Энн Марчинкiewicz и Битрис Санторини. Построение большого размеченного корпуса английского языка: Penn Treebank. Вычисления в лингвистике, 19(2):313 – 330, 1993.

[26] Давид МакКлоски, Юджин Чарниак и Марк Джонсон. Эффективное самообучение для разбора. В: Сборник конференции по технологиям человеческих языков NAACL, основная конференция, страницы 152 – 159. ACL, июнь 2006.

[27] Анкур Парикх, Оскар Тёкстрём, Дипанджан Дас и Якоб Ускорейт. Разлагаемая модель внимания. В: Empirical Methods in Natural Language Processing, 2016.

[28] Ромэн Полюс, Цаймин Сюн и Ричард Сошер. Глубокая модель с усиленным обучением для абстрактного суммирования. arXiv preprint arXiv:1705.04304, 2017.

[29] Slav Petrov, Leon Barrett, Romain Thibaux и Dan Klein. Обучение точной, компактной и интерпретируемой аннотации деревьев. В: Труды 21-й Международной конференции по вычислительной лингвистике и 44-го ежегодного собрания ACL, страницы 433 – 440. ACL, июль 2006.

[30] Офир Пресс и Лиор Вольф. Использование выходного вложения для улучшения языковых моделей. arXiv preprint arXiv:1608.05859, 2016.

[31] Рико Сеннрих, Барри Хаддоу и Александра Бирч. Нейронный машинный перевод редких слов с использованием субсловных единиц. arXiv preprint arXiv:1508.07909, 2015.

[32] Ноам Шазиер, Азалия Мирхосейни, Кшysztof Маз з, Энди Дэвис, Куок Ле, Джеффри Хинтон и Джефф Дин. Невероятно большие нейронные сети: слой с разреженными вентильными смесями экспертов. arXiv preprint arXiv:1701.06538, 2017.

[33] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever и Ruslan Salakhutdinov. Dropout: простой способ предотвращения переобучения нейронных сетей. Journal of Machine Learning Research, 15(1):1929 – 1958, 2014.

[34] Сайнбаяр Сухбаатар, Артур Сзлам, Джейсон Уэстон и Роб Фергус. Концевые сети памяти. В: C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama и R. Garnett, редакторы, Достижения в обработке нейронной информации 28, страницы 2440 – 2448. Curran Associates, Inc., 2015.

[35] Илья Сутскевер, Ориол Виньялс и Куок ВВ Ле. Обучение последовательностей с помощью нейронных сетей. В: Advances in Neural Information Processing Systems, страницы 3104 – 3112, 2014.

[36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, и Zbigniew Wojna. Переосмысление архитектуры Inception для компьютерного зрения. CoRR, abs/1512.00567, 2015.

[37] Виналс, Кейзер, Кук, Петров, Сутскевер и Хинтон. Грамматика как иностранный язык. В: Достижения в системах обработки нейронной информации, 2015.

[38] Юнхуи Ву, Майк Шустер, Чифэн Чэнь, Куок В. Ле, Мохаммад Нороузи, Вольфганг Махереи, Максим Крикун, Юань Цао, Цин Гао, Клаус Махереи и др. Нейронная система машинного перевода Google: сокращение разрыва между человеческим и машинным переводом. arXiv preprint arXiv:1609.08144, 2016.

[39] Цзе Чжоу, Ин Цао, Сюгуан Ван, Пэн Ли и Вэй Сю. Глубокие рекуррентные модели с прямыми связями для нейронного машинного перевода. CoRR, abs/1606.04199, 2016.

[40] Мухуа Чжу, Юэ Чжан, Вэнлянь Чэнь, Мин Чжан и Цинбо Чжу. Быстрое и точное синтаксическое разложение на составные части методом сдвига-редукции. В: Proceedings of the 51st Annual Meeting of the ACL (Volume 1: Long Papers), страницы 434 – 443. ACL, август 2013.

Attention Visualizations

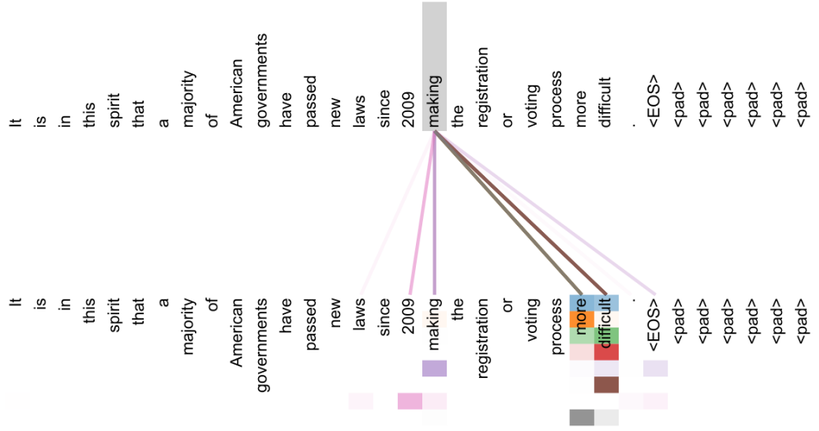


Figure 3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb ‘making’, completing the phrase ‘making...more difficult’. Attentions here shown only for the word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

Визуализация внимания

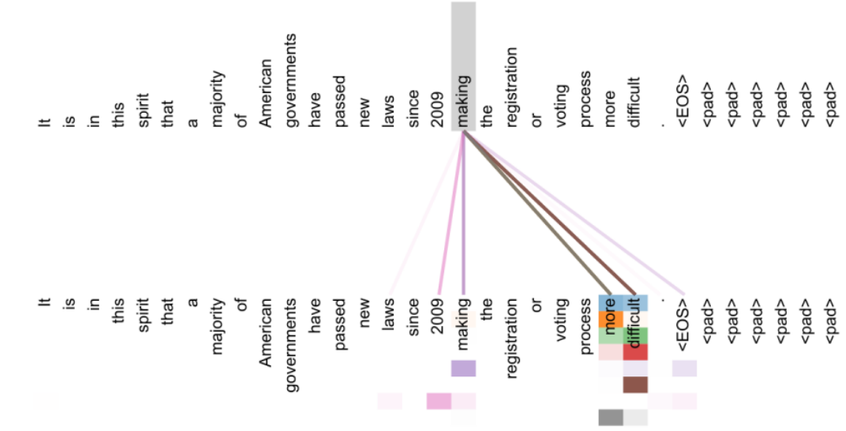


Рисунок 3: Пример работы механизма внимания, выявляющего дальние зависимости в самовнимании энкодера на слое 5 из 6. Многие головы внимания фокусируются на удаленной зависимости глагола «making», дополняя фразу «making...more difficult». Здесь показано внимание только для слова «making». Разные цвета соответствуют разным головам. Наиболее эффективно воспринимается в цвете.

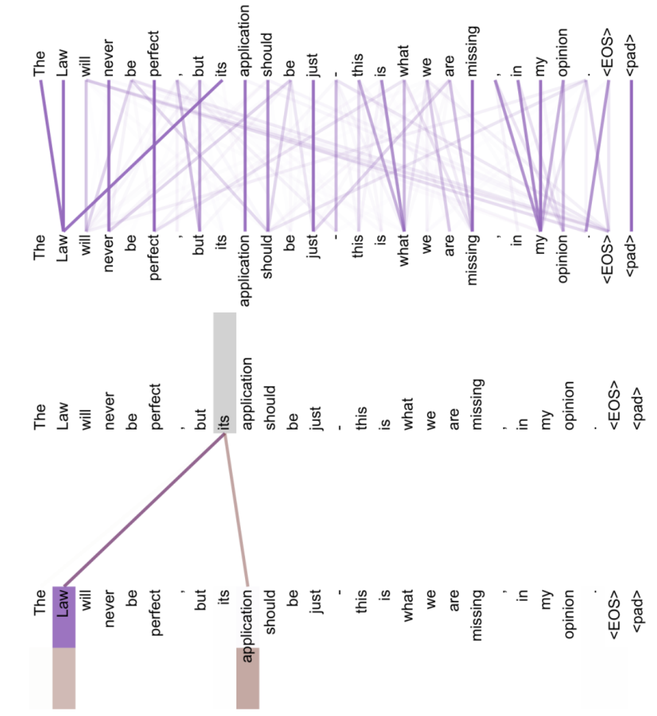


Рисунок 4: Две головки внимания, также участвующие в слое 5 из 6, по-видимому, вовлеченные в разрешение анафоры. Сверху: Полное внимание для головы 5. Снизу: Изолированное внимание только от слова «its» для головы внимания 5 и 6. Обратите внимание, что внимание очень четкое для этого слова.

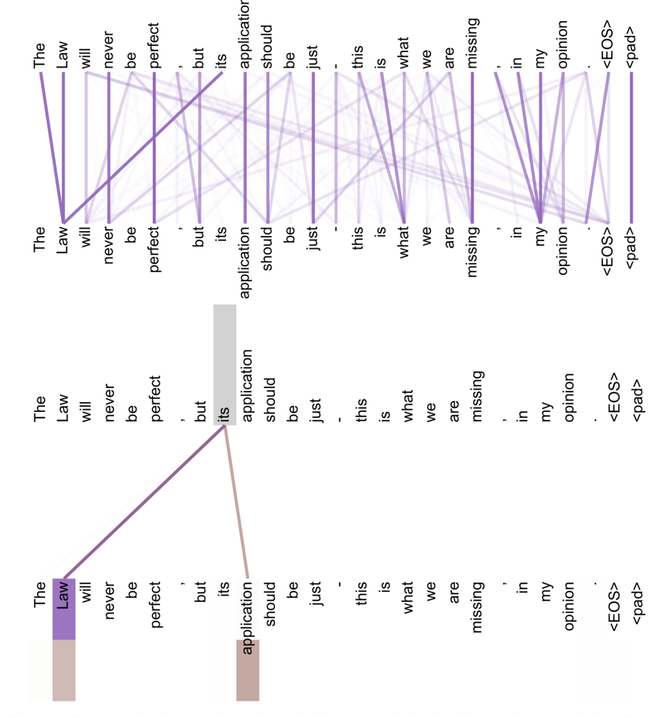


Figure 4: Two attention heads, also in layer 5 of 6, apparently involved in anaphora resolution. Top: Full attentions for head 5. Bottom: Isolated attentions from just the word ‘its’ for attention heads 5 and 6. Note that the attentions are very sharp for this word.

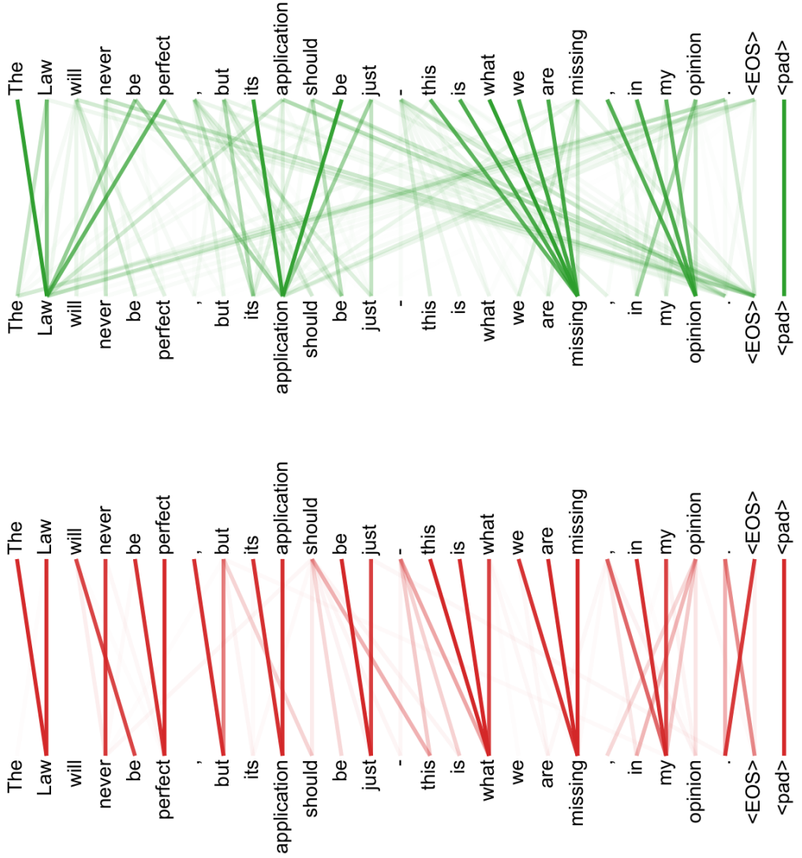


Figure 5: Many of the attention heads exhibit behaviour that seems related to the structure of the sentence. We give two such examples above, from two different heads from the encoder self-attention at layer 5 of 6. The heads clearly learned to perform different tasks.

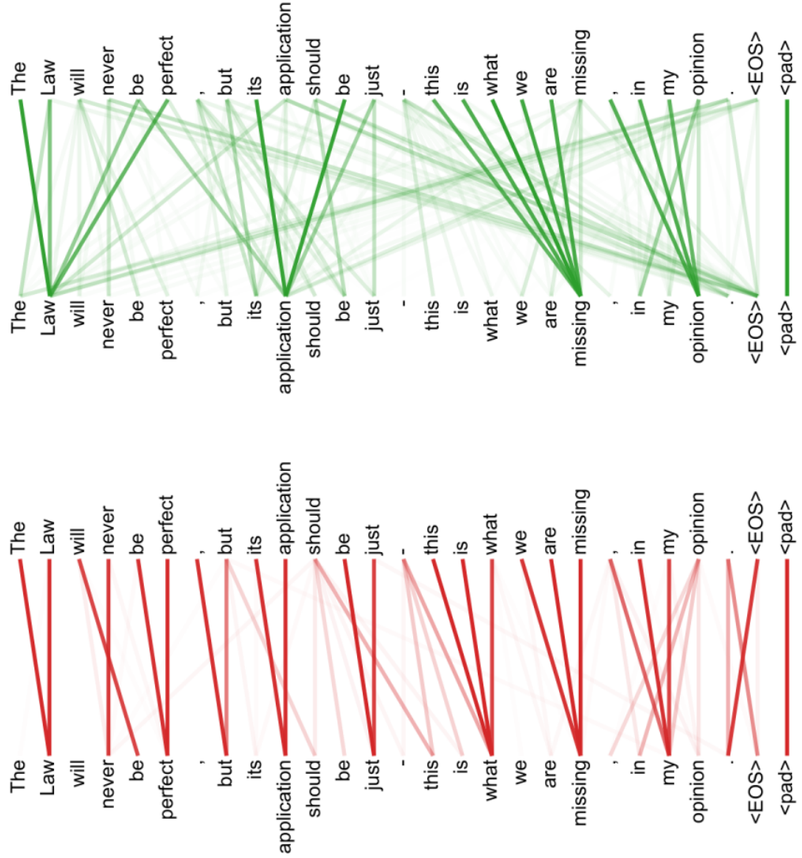


Рисунок 5: Многие головы внимания демонстрируют поведение, которое, по-видимому, связано со структурой предложения. Приведены два таких примера выше, взятые из двух различных голов само внимание энкодера на 5-м слое из 6. Головы явно научились выполнять различные задачи.