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

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

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.

Siempre que se proporcione la atribución correspondiente, Google otorga por la presente permiso para reproducir las tablas y figuras de este trabajo únicamente para su uso en trabajos periodísticos o académicos.

La Atención es Todo lo que Necesitas

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 extsuperscript{*} extsuperscript{ }
Universidad de Toronto
aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain
lukaszkaiser@google.com

Illia Polosukhin* æxtdagger
illia.polosukhin@gmail.com

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

Resumen

Los modelos de transducción de secuencia dominantes se basan en redes neuronales recurrentes o convolucionales complejas que incluyen un codificador y un decodificador. Los mejores modelos también conectan el codificador y el decodificador a través de un mecanismo de atención. Proponemos una nueva arquitectura de red simple, el Transformer, basado únicamente en mecanismos de atención, eliminando completamente la recurrencia y las convoluciones. Experimentos en dos tareas de traducción automática muestran que estos modelos son superiores en calidad, mientras que son más paralelizables y requieren significativamente menos tiempo de entrenamiento. Nuestro modelo alcanza 28.4 BLEU en la tarea de traducción inglés-alemán WMT 2014, mejorando los resultados existentes más altos, incluyendo conjuntos de modelos (ensembles), en más de 2 BLEU. En la tarea de traducción inglés-francés WMT 2014, nuestro modelo establece una nueva puntuación BLEU de estado del arte para un solo modelo de 41.8 después de entrenar durante 3.5 días en ocho GPUs, una pequeña fracción de los costos de entrenamiento de los mejores modelos de la literatura. Demostramos que el Transformer se generaliza bien a otras tareas al aplicarlo con éxito al análisis de constituyentes del inglés tanto con grandes como con datos de entrenamiento limitados.

* Contribución igual. El orden de listing es aleatorio. Jakob propuso reemplazar las RNN con autoatención y comenzó el esfuerzo para evaluar esta idea. Ashish, con Illia, diseñó e implementó los primeros modelos Transformer y ha estado crucialmente involucrado en cada aspecto de este trabajo. Noam propuso la atención de producto punto escalada, la atención de múltiples cabezas y la representación de posición sin parámetros, y se convirtió en la otra persona involucrada en casi cada detalle. Niki diseñó, implementó, ajustó y evaluó innumerables variantes de modelos en nuestra base de código original y tensor2tensor. Llion también experimentó con nuevas variantes de modelos, fue responsable de nuestra base de código inicial, y de la inferencia y visualización eficientes. Lukasz y Aidan pasaron incontables largos días diseñando varias partes e implementando tensor2tensor, reemplazando nuestra base de código anterior, mejorando enormemente los resultados y acelerando masivamente nuestra investigación.

Trabajo realizado mientras se estaba en Google Brain.

† Trabajo realizado mientras se estaba en Google Research.

31ª Conferencia sobre Sistemas de Procesamiento de Información Neural (NIPS 2017), Long Beach, CA, EE. UU.

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 Introducción

Las redes neuronales recurrentes, la memoria a corto y largo plazo [13] y, en particular, las redes neuronales recurrentes con compuerta [7], se han consolidado como enfoques de vanguardia en la modelización y transducción de secuencias, tales como la modelización del lenguaje y la traducción automática [35, 2, 5]. Desde entonces, numerosos esfuerzos han continuado ampliando los límites de los modelos de lenguaje recurrentes y las arquitecturas codificador-decodificador [38, 24, 15].

Los modelos recurrentes suelen factorizar el cálculo a lo largo de las posiciones de los símbolos en las secuencias de entrada y salida. Al alinear las posiciones con pasos en el tiempo de computación, generan una secuencia de estados ocultos h_t, como función del estado oculto anterior h_{t-1} y la entrada para la posición t. Esta naturaleza inherentemente secuencial impide la paralelización dentro de los ejemplos de entrenamiento, lo cual se vuelve crítico a mayores longitudes de secuencia, ya que las restricciones de memoria limitan el agrupamiento (batching) entre ejemplos. Investigaciones recientes han logrado mejoras significativas en la eficiencia computacional mediante técnicas de factorización [21] y computación condicional [32], mejorando también el rendimiento del modelo en el caso de esta última. Sin embargo, persiste la restricción fundamental de la computación secuencial.

Los mecanismos de atención se han convertido en una parte integral de los modelos de modelado y transducción de secuencias convincentes en diversas tareas, permitiendo el modelado de dependencias sin considerar su distancia en las secuencias de entrada o salida [2, 19]. Sin embargo, en todos menos unos pocos casos [27], dichos mecanismos de atención se utilizan en conjunción con una red recurrente.

En este trabajo proponemos el Transformer, una arquitectura de modelo que evita la recurrencia y en su lugar se basa enteramente en un mecanismo de atención para captar dependencias globales entre la entrada y la salida. El Transformer permite una paralelización significativamente mayor y puede alcanzar un nuevo estado del arte en calidad de traducción tras ser entrenado durante tan solo doce horas en ocho GPUs P100.

2 Antecedentes

El objetivo de reducir la computación secuencial también constituye la base de la GPU Neural Extendida [16], ByteNet [18] y ConvS2S [9], todos los cuales utilizan redes neuronales convolucionales como bloque de construcción básico, calculando representaciones ocultas en paralelo para todas las posiciones de entrada y salida. En estos modelos, el número de operaciones requeridas para relacionar señales de dos posiciones de entrada o salida arbitrarias crece con la distancia entre posiciones, linealmente para ConvS2S y logarítmicamente para ByteNet. Esto dificulta más el aprendizaje de dependencias entre posiciones distantes [12]. En el Transformador, esto se reduce a un número constante de operaciones, aunque a costa de una resolución efectiva reducida debido al promediado de posiciones ponderadas por atención, un efecto que contrarrestamos con Atención de Múltiples Cabezas, como se describe en la sección 3.2.

La autoatención, a veces llamada intraatención, es un mecanismo de atención que relaciona diferentes posiciones de una sola secuencia con el fin de calcular una representación de la misma. La autoatención se ha utilizado con éxito en una variedad de tareas que incluyen comprensión lectora, resumen abtractivo, implicación textual y aprendizaje de representaciones de oraciones independientes de la tarea [4, 27, 28, 22].

Las redes de memoria de extremo a extremo se basan en un mecanismo de atención recurrente en lugar de una recurrencia alineada por secuencias y se ha demostrado que funcionan bien en tareas de respuesta a preguntas con lenguaje sencillo y modelado de lenguaje [34].

Sin embargo, hasta donde de nuestro conocimiento, el Transformer es el primer modelo de transducción que se basa enteramente en la autoatención para calcular representaciones de su entrada y salida sin utilizar RNNs alineadas por secuencia ni convolución. En las siguientes secciones, describiremos el Transformer, fundamentaremos la autoatención y discutiremos sus ventajas sobre modelos como [17, 18] y [9].

3 Arquitectura del modelo

La mayoría de los modelos competitivos de transducción de secuencias neuronales poseen una estructura de codificador-decodificador [5, 2, 35]. Aquí, el codificador mapea una secuencia de entrada de representaciones de símbolos $(x_1, \ldots, x_n)$ a una secuencia de representaciones continuas $\mathbf{z}=(z_1, \ldots, z_n)$. Dado $\mathbf{z}$, el decodificador genera entonces una secuencia de salida $(y_1, \ldots, y_m)$ de símbolos un elemento a la vez. En cada paso, el modelo es autoregresivo [10], consumiendo los símbolos generados previamente como entrada adicional al generar el siguiente.

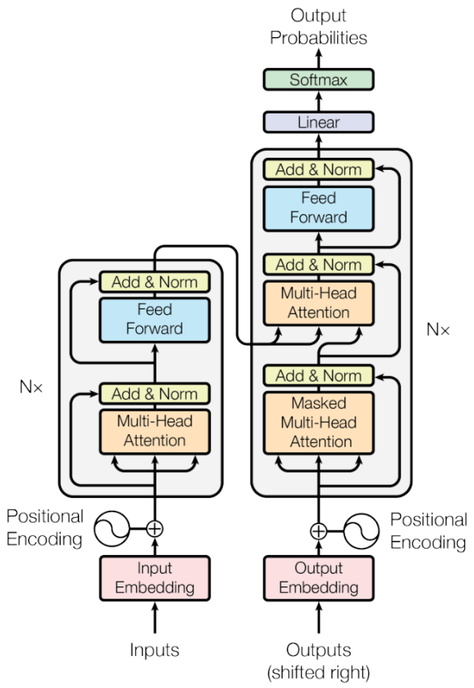


Figure 1: The Transformer - model architecture.

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

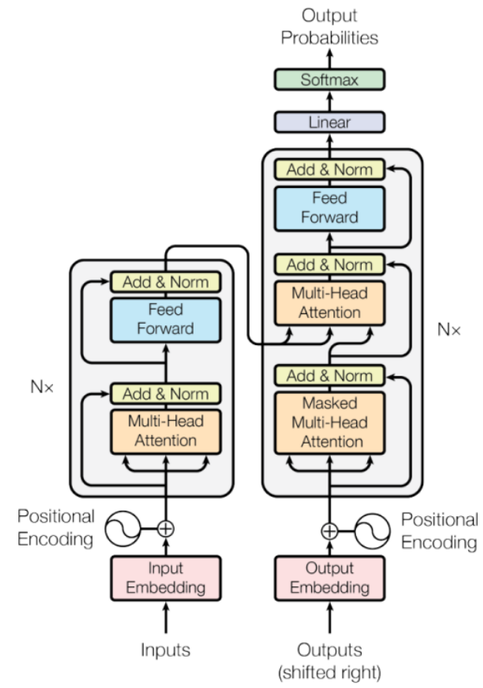


Figura 1: La arquitectura del modelo Transformer.

El Transformer sigue esta arquitectura general utilizando capas apiladas de autoatención y capas completamente conectadas punto a punto tanto para el codificador como para el decodificador, mostradas en las mitades izquierda y derecha de la Figura 1, respectivamente.

3.1 Pilas de codificador y decodificador

Codificador: El codificador está compuesto por una pila de $N=6$ capas idénticas. Cada capa tiene dos subcapas. La primera es un mecanismo de autoatención multicanal, y la segunda es una red de propagación hacia adelante completamente conectada simple y punto a punto. Empleamos una conexión residual [11] alrededor de cada una de las dos subcapas, seguida de una normalización de capa [1]. Es decir, la salida de cada subcapa es LayerNorm(x + Sublayer(x)), donde Sublayer(x) es la función implementada por la propia subcapa. Para facilitar estas conexiones residuales, todas las subcapas en el modelo, así como las capas de incrustación, producen salidas de dimensión $d_{\text{model}} = 512$.

Decodificador: El decodificador también está compuesto por una pila de $N=6$ capas idénticas. Además de los dos sub-capas en cada capa del codificador, el decodificador inserta una tercera sub-capa, que realiza atención multi-cabeza sobre la salida de la pila del codificador. Similar al codificador, empleamos conexiones residuales alrededor de cada una de las sub-capas, seguidas de normalización de capa. También modificamos la sub-capa de auto-atención en la pila del decodificador para evitar que las posiciones atiendan a posiciones subsiguientes. Este enmascaramiento, combinado con el hecho de que los incrustaciones de salida están desplazadas una posición, asegura que las predicciones para la posición i solo puedan depender de las salidas conocidas en posiciones menores que i.

3.2 Atención

Una función de atención puede describirse como mapear una consulta y un conjunto de pares clave-valor en una salida, donde la consulta, las claves, los valores y la salida son todos vectores. La salida se calcula como una suma ponderada

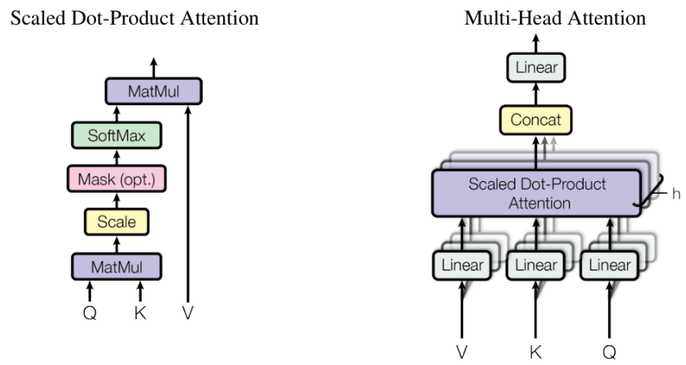


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

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.

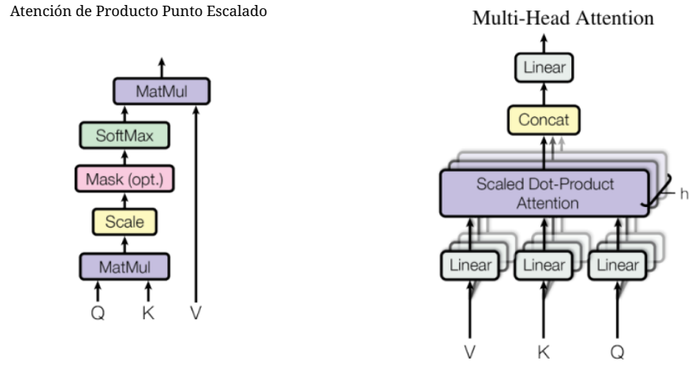


Figura 2: (izquierda) Atención de Producto Escalado. (derecha) La Atención de Múltiples Cabezas consiste en varias capas de atención que se ejecutan en paralelo.

de los valores, donde el peso asignado a cada valor se calcula mediante una función de compatibilidad de la consulta con la clave correspondiente.

3.2.1 Atención de Producto Escalar

Llamamos a nuestra atención particular "Atención de Producto Punto Escalada" (Figura 2). La entrada consiste en consultas y claves de dimensión d_k, y valores de dimensión d_v. Calculamos los productos punto de la consulta con todas las claves, dividimos cada uno por $\sqrt{d_k}$, y aplicamos una función softmax para obtener los pesos sobre los valores.

En la práctica, calculamos la función de atención sobre un conjunto de consultas simultáneamente, agrupadas en una matriz Q. Las claves y los valores también se agrupan en las matrices K y V. Calculamos la matriz de salidas como:

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

Las dos funciones de atención más comúnmente utilizadas son la atención aditiva [2] y la atención por producto punto (multiplicativa). La atención por producto punto es idéntica a nuestro algoritmo, excepto por el factor de escalado de $\frac{1}{\sqrt{d_k}}$. La atención aditiva calcula la función de compatibilidad utilizando una red de propagación hacia adelante con una sola capa oculta. Aunque las dos son similares en complejidad teórica, la atención por producto punto es mucho más rápida y eficiente en espacio en la práctica, ya que puede implementarse utilizando código de multiplicación de matrices altamente optimizado.

Aunque para valores pequeños de d_k ambos mecanismos se comportan de manera similar, la atención aditiva supera a la atención de producto punto sin escalado para valores mayores de d_k [3]. Sospechamos que para valores grandes de d_k, los productos punto crecen en magnitud, empujando la función softmax hacia regiones donde esta presenta gradientes extremadamente pequeños [4]. Para contrarrestar este efecto, escalamos los productos punto por $\frac{1}{\sqrt{d_k}}$.

3.2.2 Atención de Múltiples Cabezas

En lugar de realizar una única función de atención con claves, valores y consultas de d_{model} dimensiones, encontramos que era beneficioso proyectar linealmente las consultas, claves y valores h veces con diferentes proyecciones lineales aprendidas a dimensiones de d_k, d_k y d_v, respectivamente. En cada una de estas versiones proyectadas de consultas, claves y valores, realizamos entonces la función de atención en paralelo, produciendo [resultados] de d_v dimensiones.

[4] Para ilustrar por qué los productos escalares se vuelven grandes, suponga que las componentes de q y k son variables aleatorias independientes con media 0 y varianza 1. Entonces su producto escalar, $q \cdot k = \sum_{i=1} q_i k_i$, tiene media 0 y varianza 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 $\sqrt{d_\text{model}}$.

valores de salida. Estos se concatenan y una vez más se proyectan, dando como resultado los valores finales, como se muestra en la Figura 2.

La atención de múltiples cabezas permite al modelo prestar atención conjuntamente a información de diferentes subespacios de representación en distintas posiciones. Con una única cabeza de atención, el promediado inhibe esto.

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

Donde las proyecciones son matrices de parámetros $W_i^Q \in \mathbb{R}^{d_\text{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_\text{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_\text{model} \times d_v}$ y $W^O \in \mathbb{R}^{hd_v \times d_\text{model}}$.

En este trabajo empleamos $h = 8$ capas de atención paralelas, o cabezas. Para cada una de estas utilizamos $d_k = d_v = d_\text{model}/h = 64$. Debido a la reducción de la dimensión de cada cabeza, el costo computacional total es similar al de la atención de una sola cabeza con dimensionalidad completa.

3.2.3 Aplicaciones de la Atención en nuestro Modelo

El Transformer utiliza atención de múltiples cabezas de tres formas diferentes:

En las capas de "atención codificador-decodificador", las consultas provienen de la capa anterior del decodificador, y las claves de memoria y los valores provienen de la salida del codificador. Esto permite que cada posición en el decodificador atienda a todas las posiciones en la secuencia de entrada. Esto imita los mecanismos típicos de atención codificador-decodificador en modelos de secuencia a secuencia como [38, 2, 9].

El codificador contiene capas de autoatención. En una capa de autoatención, todas las claves, valores y consultas provienen del mismo lugar; en este caso, la salida de la capa anterior del codificador. Cada posición en el codificador puede atender a todas las posiciones de la capa anterior del codificador.

De manera similar, las capas de autoatención en el decodificador permiten que cada posición en el decodificador preste atención a todas las posiciones del decodificador hasta y incluyendo esa posición. Necesitamos prevenir el flujo de información hacia la izquierda en el decodificador para preservar la propiedad autoregresiva. Implementamos esto dentro de la atención de producto punto escalada enmascarando (estableciendo en $-\infty$) todos los valores en la entrada del softmax que corresponden a conexiones ilegales. Ver Figura 2.

3.3 Redes de Alimentación Directa Por Posición

Además de las subcapas de atención, cada una de las capas en nuestro codificador y decodificador contiene una red de propagación hacia adelante completamente conectada, la cual se aplica a cada posición de forma separada e idéntica. Esta consiste en dos transformaciones lineales con una activación ReLU en medio.

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

Aunque las transformaciones lineales son las mismas en diferentes posiciones, utilizan parámetros distintos de capa a capa. Otra forma de describir esto es como dos convoluciones con tamaño de kernel 1. La dimensionalidad de entrada y salida es $d_\text{model} = 512$, y la capa interna tiene dimensionalidad $d_{ff} = 2048$.

3.4 Incrustaciones y Softmax

De manera similar a otros modelos de transducción de secuencias, utilizamos incrustaciones aprendidas para convertir los tokens de entrada y los tokens de salida en vectores de dimensión d_model. También utilizamos la transformación lineal aprendida habitual y la función softmax para convertir la salida del decodificador en probabilidades predichas del siguiente token. En nuestro modelo, compartimos la misma matriz de pesos entre las dos capas de incrustación y la transformación lineal pre-softmax, similar a [30]. En las capas de incrustación, multiplicamos esos pesos por $\sqrt{d_\text{model}}$.

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

Tabla 1: Longitudes máximas de trayectoria, complejidad por capa y número mínimo de operaciones secuenciales para diferentes tipos de capas. n es la longitud de la secuencia, d es la dimensión de la representación, k es el tamaño del kernel de las convoluciones y r el tamaño del vecindario en la autoatención restringida.

Tipo de capa	Complejidad por capa	Secuencial Operaciones	Longitud máxima del camino
Autoatención	$O(n^2 \cdot d)$	O(1)	O(1)
Recurrente	$O(n \cdot d^2)$	O()	O(n)
Convolucional	$O(k \cdot n \cdot d^2)$	O(1)	$O(log_k(n))$
Autoatención (restringida)	$O(r \cdot n \cdot d)$	O(1)	O(n/r)

3.5 Codificación Posicional

Dado que nuestro modelo no contiene ni recurrencias ni convoluciones, a fin de que el modelo pueda hacer uso del orden de la secuencia, debemos inyectar alguna información sobre la posición relativa o absoluta de los tokens en la secuencia. Con este fin, añadimos "codificaciones posicionales" a las incrustaciones de entrada en la base de las pilas del codificador y del decodificador. Las codificaciones posicionales tienen la misma dimensión d_{model} que las incrustaciones, de modo que ambas pueden sumarse. Existen muchas opciones de codificaciones posicionales, ya sean aprendidas o fijas [9].

En este trabajo, utilizamos funciones seno y coseno de diferentes frecuencias:

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

donde pos es la posición e i es la dimensión. Es decir, cada dimensión del codificado posicional corresponde a un sinusoid. Las longitudes de onda forman una progresión geométrica desde 2π hasta $10000 \cdot 2\pi$. Elegimos esta función porque hipotetizamos que permitiría al modelo aprender fácilmente a atender por posiciones relativas, ya que para cualquier desplazamiento fijo k, PE_{pos+k} puede representarse como una función lineal de PE_{pos}.

También experimentamos con el uso de incrustaciones posicionales aprendidas [9] en su lugar, y descubrimos que las dos versiones produjeron resultados casi idénticos (ver Tabla 3, fila (E)). Elegimos la versión sinusoidal porque podría permitir que el modelo extrapole a longitudes de secuencia más largas que las encontradas durante el entrenamiento.

4 ¿Por qué Autoatención

En esta sección comparamos varios aspectos de las capas de autoatención con las capas recurrentes y convolucionales comúnmente utilizadas para mapear una secuencia de longitud variable de representaciones de símbolos $(x_1, \ldots, x_n)$ a otra secuencia de igual longitud $(z_1, \ldots, z_n)$, con $x_i, z_i \in \mathbb{R}^d$, como una capa oculta en un codificador o decodificador típico de transducción de secuencias. Para motivar nuestro uso del autoatención, consideramos tres requisitos.

Una es la complejidad computacional total por capa. La otra es la cantidad de computations que se pueden parallelizar, medida por el número mínimo de operaciones secuenciales requeridas.

El tercero es la longitud del camino entre las dependencias a largo alcance en la red. Aprender dependencias a largo alcance es un desafío clave en muchas tareas de transducción de secuencias. Un factor clave que afecta la capacidad de aprender tales dependencias es la longitud de los caminos que las señales hacia adelante y hacia atrás deben traversar en la red. Cuanto más cortos son estos caminos entre cualquier combinación de posiciones en las secuencias de entrada y salida, más fácil es aprender dependencias a largo alcance [12]. Por lo tanto, también comparamos la longitud máxima del camino entre cualquier par de posiciones de entrada y salida en redes compuestas por los diferentes tipos de capas.

Como se señaló en la Tabla 1, una capa de autoatención conecta todas las posiciones con un número constante de operaciones ejecutadas secuencialmente, mientras que una capa recurrente requiere $O(n)$ operaciones secuenciales. En términos de complejidad computacional, las capas de autoatención son más rápidas que las capas recurrentes cuando la secuencia

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:

la longitud n es menor que la dimensionalidad de la representación d, lo cual es el caso más frecuente con representaciones de oraciones utilizadas por modelos de vanguardia en traducción automática, como las representaciones word-piece [38] y byte-pair [31]. Para mejorar el rendimiento computacional en tareas que involucran secuencias muy largas, la autoatención podría restringirse a considerar únicamente un vecindario de tamaño r en la secuencia de entrada centrado alrededor de la respectiva posición de salida. Esto aumentaría la longitud máxima del camino a $O(n/r)$. Planificamos investigar este enfoque más a fondo en trabajos futuros.

Una sola capa convolucional con ancho de kernel $k<n$ no conecta todos los pares de posiciones de entrada y salida. Hacerlo requiere una pila de $O(n/k)$ capas convolucionales en el caso de kernels contiguos, o $O(log_k(n))$ en el caso de convoluciones dilatadas [18], aumentando la longitud de los caminos más largos entre cualquier par de posiciones en la red. Las capas convolucionales son generalmente más costosas que las capas recurrentes, por un factor de k. Sin embargo, las convoluciones separables [6] reducen considerablemente la complejidad, a $O(k \cdot n \cdot d + n \cdot d^2)$. Incluso con $k=n$, sin embargo, la complejidad de una convolución separable es igual a la combinación de una capa de autoatención y una capa de propagación hacia adelante punto a punto, el enfoque que adoptamos en nuestro modelo.

Como beneficio secundario, la autoatención podría producir modelos más interpretables. Inspeccionamos las distribuciones de atención de nuestros modelos y presentamos y discutimos ejemplos en el apéndice. No solo los heads de atención individuales aprenden claramente a realizar diferentes tareas, sino que muchos parecen exhibir comportamientos relacionados con la estructura sintáctica y semántica de las oraciones.

5 Entrenamiento

Esta sección describe el régimen de entrenamiento para nuestros modelos.

5.1 Datos de entrenamiento y agrupación en lotes

Entrenamos con el conjunto de datos estándar WMT 2014 Inglés-Alemán, que consta de aproximadamente 4,5 millones de pares de oraciones. Las oraciones se codificaron utilizando codificación byte-pareado [3], la cual posee un vocabulario compartido fuente-destino de aproximadamente 37000 tokens. Para Inglés-Francés, utilizamos el notablemente más grande conjunto de datos WMT 2014 Inglés-Francés, compuesto por 36M oraciones, y dividimos los tokens en un vocabulario de word-piece de 32000 [38]. Los pares de oraciones se agruparon por longitud de secuencia aproximada. Cada lote de entrenamiento contenía un conjunto de pares de oraciones que incluían aproximadamente 25000 tokens de origen y 25000 tokens de destino.

5.2 Hardware y Programación

Entrenamos nuestros modelos en una máquina con 8 GPUs NVIDIA P100. Para nuestros modelos base que utilizan los hiperparámetros descritos a lo largo del artículo, cada paso de entrenamiento tardó aproximadamente 0.4 segundos. Entrenamos los modelos base por un total de 100.000 pasos o 12 horas. Para nuestros modelos grandes (descritos en la última línea de la tabla 3), el tiempo por paso fue de 1.0 segundos. Los modelos grandes se entrenaron durante 300.000 pasos (3.5 días).

5.3 Optimizador

Utilizamos el optimizador Adam [20] con $\beta_1 = 0.9$, $\beta_2 = 0.98$ y $\varepsilon = 10^{-9}$. Variamos la tasa de aprendizaje a lo largo del entrenamiento, según la fórmula:

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

Esto corresponde a aumentar la tasa de aprendizaje linealmente durante los primeros warmup_steps pasos de entrenamiento, y disminuirla posteriormente proporcionalmente a la raíz inversa cuadrada del número de paso. Utilizamos warmup_steps = 4000.

5.4 Regularización

Empleamos tres tipos de regularización durante el entrenamiento:

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	**$3.3 \cdot 10^{18}$**	
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.

Tabla 2: El Transformer obtiene mejores puntuaciones BLEU que los modelos state-of-the-art anteriores en las pruebas newstest2014 de inglés-alemán e inglés-francés, a una fracción del costo de entrenamiento.

Modelo	BLEU		Coste de entrenamiento (FLOPs)	
	EN-ALE	EN-FR	EN-ALE	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 Conjunto [39]		40,4		$8,0 \cdot 10^{20}$
GNMT + RL Conjunto [38]	26,30	41,16	$1,8 \cdot 10^{20}$	$1,1 \cdot 10^{21}$
ConvS2S Conjunto [9]	26,36	41,29	$7,7 \cdot 10^{19}$	$1,2 \cdot 10^{21}$
Transformer (modelo base)	27,3	38,1	$3,3 \cdot 10^{18}$	
Transformer (grande)	28,4	41,8	$2,3 \cdot 10^{19}$	

Dropout Residual Aplicamos dropout [33] a la salida de cada subcapa, antes de que se sume a la entrada de la subcapa y se normalice. Además, aplicamos dropout a la suma de los embeddings y los codificaciones posicionales tanto en las pilas del codificador como del decodificador. Para el modelo base, utilizamos una tasa de $P_{drop} = 0.1$.

Suavizado de etiquetas Durante el entrenamiento, empleamos suavizado de etiquetas con valor $\epsilon_{ls} = 0.1$ [36]. Esto perjudica la perplexidad, ya que el modelo aprende a ser más inseguro, pero mejora la precisión y la puntuación BLEU.

6 Resultados

6.1 Traducción automática

En la tarea de traducción inglés-alemán WMT 2014, el gran modelo transformer (Transformer (big) en la Tabla 2) supera a los mejores modelos previamente reportados (incluyendo conjuntos) en más de 2.0 BLEU, estableciendo una nueva puntuación estado del arte BLEU de 28.4. La configuración de este modelo se lista en la última línea de la Tabla 3. El entrenamiento duró 3.5 días en 8 GPUs P100. Incluso nuestro modelo base supera a todos los modelos y conjuntos publicados anteriormente, a una fracción del costo de entrenamiento de cualquiera de los modelos competitivos.

En la tarea de traducción del inglés al francés WMT 2014, nuestro modelo grande obtiene una puntuación BLEU de 41.0, superando a todos los modelos individuales publicados anteriormente, a un costo de entrenamiento inferior a 1/4 del del modelo anterior más avanzado. El modelo Transformer (grande) entrenado para la traducción del inglés al francés utilizó una tasa de dropout $P_{drop} = 0.1$, en lugar de 0.3.

Para los modelos base, utilizamos un único modelo obtenido promediando los últimos 5 puntos de control, que se escribieron a intervalos de 10 minutos. Para los modelos grandes, promediamos los últimos 20 puntos de control. Utilizamos búsqueda de haz con un tamaño de haz de 4 y una penalización de longitud $\alpha = 0.6$ [38]. Estos hiperparámetros se seleccionaron tras experimentación en el conjunto de desarrollo. Establecimos la longitud máxima de salida durante la inferencia en longitud de entrada + 50, pero terminamos anticipadamente cuando es posible [38].

La Tabla 2 resume nuestros resultados y compara nuestra calidad de traducción y costos de entrenamiento con otras arquitecturas de modelo de la literatura. Estimamos el número de operaciones de punto flotante utilizadas para entrenar un modelo multiplicando el tiempo de entrenamiento, el número de GPUs utilizadas y una estimación de la capacidad sostenida de punto flotante de precisión simple de cada GPU 5.

6.2 Variaciones del modelo

Para evaluar la importancia de los diferentes componentes del Transformador, variamos nuestro modelo base de distintas maneras, midiendo el cambio en el rendimiento en la traducción de inglés a alemán en el

[5]Utilizamos valores de 2,8, 3,7, 6,0 y 9,5 TFLOPS para K80, K40, M40 y P100, respectivamente.

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

Tabla 3: Variaciones en la arquitectura Transformer. Los valores no listados son idénticos a los del modelo base. Todas las métricas se encuentran en el conjunto de desarrollo de traducción inglés-alemán, newstest2013. Las perplexidades listadas son por palabrapiece, según nuestro codificación byte-pair, y no deben compararse con las perplexidades por palabra.

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	entrenamiento 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)	incrustación posicional en lugar de sinusoides									4.92	25.7	
grande	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

conjunto de desarrollo, newstest2013. Utilizamos búsqueda de haz como se describió en la sección anterior, pero sin promediado de checkpoints. Presentamos estos resultados en la Tabla 3.

En la Tabla 3, filas (A), variamos el número de cabezas de atención y las dimensiones de la clave y el valor de atención, manteniendo constante la cantidad de computación, como se describe en la Sección 3.2.2. Si bien la atención de una sola cabeza es 0.9 BLEU peor que el mejor ajuste, la calidad también disminuye con demasiadas cabezas.

En las filas (B) de la Tabla 3, observamos que reducir el tamaño de la clave de atención d_k perjudica la calidad del modelo. Esto sugiere que determinar la compatibilidad no es fácil y que una función de compatibilidad más sofisticada que el producto punto puede ser beneficiosa. Observamos además en las filas (C) y (D) que, como era de esperarse, los modelos más grandes son mejores, y el dropout es muy útil para evitar el sobreajuste. En la fila (E) reemplazamos nuestro codificación posicional sinusoidal por incrustaciones posicionales aprendidas [9], y observamos resultados casi idénticos a los del modelo base.

6.3 Análisis de Constituencias en Inglés

Para evaluar si el Transformer puede generalizarse a otras tareas, realizamos experimentos de análisis de constituyentes en inglés. Esta tarea presenta desafíos específicos: la salida está sujeta a fuertes restricciones estructurales y es significativamente más larga que la entrada. Además, los modelos de secuencia a secuencia basados en RNN no han logrado obtener resultados de estado del arte en regímenes de datos reducidos [37].

Entrenamos un transformador de 4 capas con $d_{model} = 1024$ en la sección del Wall Street Journal (WSJ) del Penn Treebank [25], aproximadamente 40K oraciones de entrenamiento. También lo entrenamos en un entorno semisupervisado, utilizando los corpora BerkleyParser y de alta confianza más grandes, que contienen aproximadamente 17M de oraciones [37]. Utilizamos un vocabulario de 16K tokens para el entorno exclusivo de WSJ y un vocabulario de 32K tokens para el entorno semisupervisado.

Realizamos solo un pequeño número de experimentos para seleccionar el dropout, tanto la atención como el residual (sección 5.4), las tasas de aprendizaje y el tamaño del haz en el conjunto de desarrollo de la Sección 22; todos los demás parámetros permanecieron inalterados respecto al modelo base de traducción del inglés al alemán. Durante la inferencia, 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 + 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.

Tabla 4: El Transformer se generaliza bien al análisis de constituyentes en inglés (Los resultados son de la Sección 23 del WSJ)

Analizador	Entrenamiento	F1 WSJ 23
Vinyals y Kaiser et al. (2014) [37]	Solo WSJ, discriminativo	88,3
Petrov et al. (2006) [29]	Solo WSJ, discriminativo	90,4
Zhu et al. (2013) [40]	Solo WSJ, discriminativo	90,4
Dyer et al. (2016) [8]	Solo WSJ, discriminativo	91,7
Transformador (4 capas)	Solo WSJ, discriminativo	91,3
Zhu et al. (2013) [40]	semisupervisado	91,3
Huang & Harper (2009) [14]	semisupervisado	91,3
McClosky et al. (2006) [26]	semisupervisado	92,1
Vinyals y Kaiser et al. (2014) [37]	semisupervisado	92,1
Transformador (4 capas)	semisupervisado	92,7
Luong et al. (2015) [23]	multitarea	93,0
Dyer et al. (2016) [8]	generativo	93,3

aumentamos la longitud de salida máxima a longitud de entrada + 300. Utilizamos un tamaño de haz de 21 y α=0.3 tanto para el caso solo de WSJ como para el entorno semisupervisado.

Nuestros resultados en la Tabla 4 muestran que, a pesar de la falta de ajuste específico para la tarea, nuestro modelo obtiene un rendimiento sorprendentemente bueno, produciendo mejores resultados que todos los modelos reportados anteriormente con la excepción de la Red Neuronal Recurrente Gramatical [8].

A diferencia de los modelos RNN de secuencia a secuencia [37], el Transformer supera al Berkeley-Parser [29] incluso cuando se entrena únicamente con el conjunto de entrenamiento WSJ de 40K oraciones.

7 Conclusión

En este trabajo, presentamos el Transformer, el primer modelo de transducción de secuencias basado enteramente en atención, reemplazando las capas recurrentes más comúnmente utilizadas en arquitecturas codificador-decodificador con autoatención de múltiples cabezas.

Para las tareas de traducción, el Transformer puede entrenarse significativamente más rápido que las arquitecturas basadas en capas recurrentes o convolucionales. En ambas tareas de traducción WMT 2014 de inglés a alemán y de inglés a francés, alcanzamos un nuevo estado del arte. En la primera tarea, nuestro mejor modelo supera incluso a todos los ensembles reportados anteriormente.

Estamos entusiasmados con el futuro de los modelos basados en atención y planeamos aplicarlos a otras tareas. Planiamos extender el Transformador a problemas que involucren modalidades de entrada y salida distintas al texto e investigar mecanismos de atención locales y restringidos para manejar de manera eficiente grandes entradas y salidas, como imágenes, audio y video. Hacer que la generación sea menos secuencial es otro de nuestros objetivos de investigación.

El código que utilizamos para entrenar y evaluar nuestros modelos está disponible en https://github.com/tensorflow/tensor2tensor.

Agradecimientos Estamos agradecidos con Nal Kalchbrenner y Stephan Gouws por sus fructíferos comentarios, correcciones e inspiración.

Referencias

[1] Jimmy Lei Ba, Jamie Ryan Kiros, y Geoffrey E Hinton. Normalización por capas. arXiv preprint arXiv:1607.06450, 2016.

[2] Dzmitry Bahdanau, Kyunghyun Cho, y Yoshua Bengio. Traducción de máquina neural mediante el aprendizaje conjunto para alinear y traducir. CoRR, abs/1409.0473, 2014.

[3] Denny Britz, Anna Goldie, Minh-Thang Luong, y Quoc V. Le. Exploración masiva de arquitecturas de traducción automática neuronal. CoRR, abs/1703.03906, 2017.

[4] Jianpeng Cheng, Li Dong y Mirella Lapata. Redes de memoria de largo y corto plazo para lectura automática. 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, y Yoshua Bengio. Aprendizaje de representaciones de frases utilizando un codificador-decodificador rnn para traducción automática estadística. 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, y Yoshua Bengio. Evaluación empírica de redes neuronales recurrentes con compuerta en el modelado de secuencias. CoRR, abs/1412.3555, 2014.

[8] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros y Noah A. Smith. Gramáticas de redes neuronales recurrentes. En Proc. of NAACL, 2016.

[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, y Yann N. Dauphin. Aprendizaje de secuencia a secuencia convolucional. arXiv preprint arXiv:1705.03122v2, 2017.

[10] Alex Graves. Generación de secuencias con redes neuronales recurrentes. arXiv preprint arXiv:1308.0850, 2013.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, y Jian Sun. Aprendizaje residual profundo para el reconocimiento de imágenes. En Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, páginas 770 – 778, 2016.

[12] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi y Jürgen Schmidhuber. Flujo de gradiente en redes recurrentes: la dificultad de aprender dependencias a largo plazo, 2001.

[13] Sepp Hochreiter y Jürgen Schmidhuber. Memoria a largo plazo de corto término. Neural computation, 9(8):1735 – 1780, 1997.

[14] Zhongqiang Huang y Mary Harper. Gramáticas PCFG de autoentrenamiento con anotaciones latentes a través de idiomas. En Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, páginas 832 – 841. ACL, agosto de 2009.

[15] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, y Yonghui Wu. Explorando los límites del modelado de lenguaje. arXiv preprint arXiv:1602.02410, 2016.

[16] Łukasz Kaiser y Samy Bengio. ¿Puede la memoria activa reemplazar a la atención? En Advances in Neural Information Processing Systems, (NIPS), 2016.

[17] Łukasz Kaiser y Ilya Sutskever. Las GPU neuronales aprenden algoritmos. En International Conference on Learning Representations (ICLR), 2016.

[18] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, y Koray Kavukcuoglu. Traducción de máquina neural en tiempo lineal. arXiv preprint arXiv:1610.10099v2, 2017.

[19] Yoon Kim, Carl Denton, Luong Hoang y Alexander M. Rush. Redes de atención estructuradas. En International Conference on Learning Representations, 2017.

[20] Diederik Kingma y Jimmy Ba. Adam: Un método para optimización estocástica. En ICLR, 2015.

[21] Oleksii Kuchaiev y Boris Ginsburg. Trucos de factorización para redes LSTM. arXiv preprint arXiv:1703.10722, 2017.

[22] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou y Yoshua Bengio. Un incrustado de oraciones con autoatención estructurada. arXiv preprint arXiv:1703.03130, 2017.

[23] Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals y Lukasz Kaiser. Aprendizaje de secuencia a secuencia multi-tarea. arXiv preprint arXiv:1511.06114, 2015.

[24] Minh-Thang Luong, Hieu Pham, y Christopher D Manning. Enfoques efectivos para la traducción automática neural basada en atención. 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] Mitchell P Marcus, Mary Ann Marcinkiewicz y Beatrice Santorini. Construcción de un corpus anotado extenso de inglés: El Penn Treebank. Lingüística computacional, 19(2):313 – 330, 1993.

[26] David McClosky, Eugene Charniak y Mark Johnson. Entrenamiento autoeficaz para el análisis sintáctico. En Proceedings of the Human Language Technology Conference of the NAACL, Main Conference, páginas 152 – 159. ACL, junio de 2006.

[27] Ankur Parikh, Oscar Täckström, Dipanjan Das y Jakob Uszkoreit. Un modelo de atención descomponible. En Empirical Methods in Natural Language Processing, 2016.

[28] Romain Paulus, Caiming Xiong y Richard Socher. Un modelo de refuerzo profundo para la resumención abstractiva. arXiv preprint arXiv:1705.04304, 2017.

[29] Slav Petrov, Leon Barrett, Romain Thibaux y Dan Klein. Aprendizaje de anotación de árboles precisa, compacta e interpretable. En Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, páginas 433 – 440. ACL, julio de 2006.

[30] Ofir Press y Lior Wolf. Usando la incrustación de salida para mejorar los modelos de lenguaje. arXiv preprint arXiv:1608.05859, 2016.

[31] Rico Sennrich, Barry Haddow, y Alexandra Birch. Traducción automática neural de palabras raras con unidades de subpalabra. arXiv preprint arXiv:1508.07909, 2015.

[32] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton y Jeff Dean. Redes neuronales outrageosamente grandes: la capa de mezcla de expertos con compuertas dispersas. arXiv preprint arXiv:1701.06538, 2017.

[33] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever y Ruslan Salakhutdinov. Dropout: una forma sencilla de evitar el sobreajuste en redes neuronales. Journal of Machine Learning Research, 15(1):1929 – 1958, 2014.

[34] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston y Rob Fergus. Redes de memoria de extremo a extremo. En C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama y R. Garnett, editores, Avances en Procesamiento de Información con Redes Neuronales 28, páginas 2440 – 2448. Curran Associates, Inc., 2015.

[35] Ilya Sutskever, Oriol Vinyals, y Quoc VV Le. Aprendizaje de secuencia a secuencia con redes neuronales. En Advances in Neural Information Processing Systems, páginas 3104 – 3112, 2014.

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

[37] Vinyals & Kaiser, Koo, Petrov, Sutskever, y Hinton. Gramática como idioma extranjero. En 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. El sistema de traducción neural de Google: cerrando la brecha entre la traducción humana y la máquina. arXiv preprint arXiv:1609.08144, 2016.

[39] Jie Zhou, Ying Cao, Xuguang Wang, Peng Li y Wei Xu. Modelos recurrentes profundos con conexiones de avance rápido para traducción automática neural. CoRR, abs/1606.04199, 2016.

[40] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang y Jingbo Zhu. Análisis constituyente rápido y preciso basado en desplazamiento-reducción. En Proceedings of the 51st Annual Meeting of the ACL (Volume 1: Long Papers), páginas 434 – 443. ACL, agosto de 2013.

Attention Visualizations

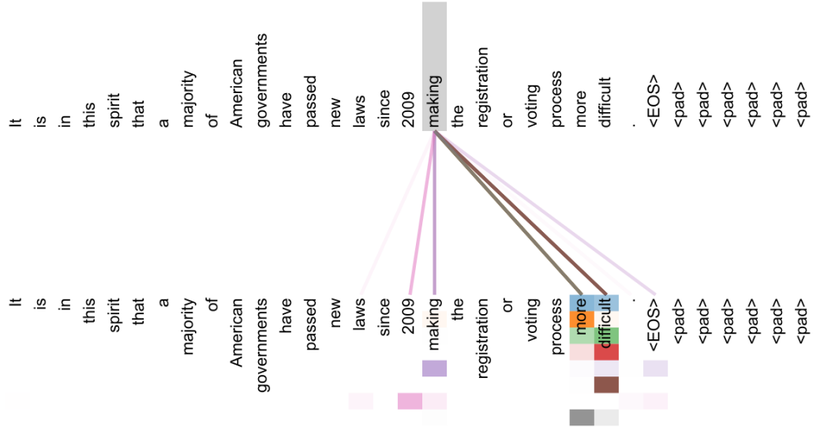


Figure 3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb ‘making’, completing the phrase ‘making...more difficult’. Attentions here shown only for the word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

Visualizaciones de Atención

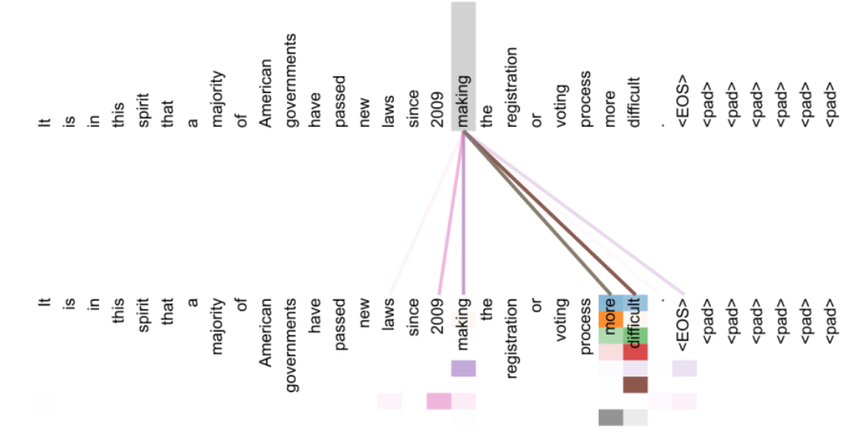
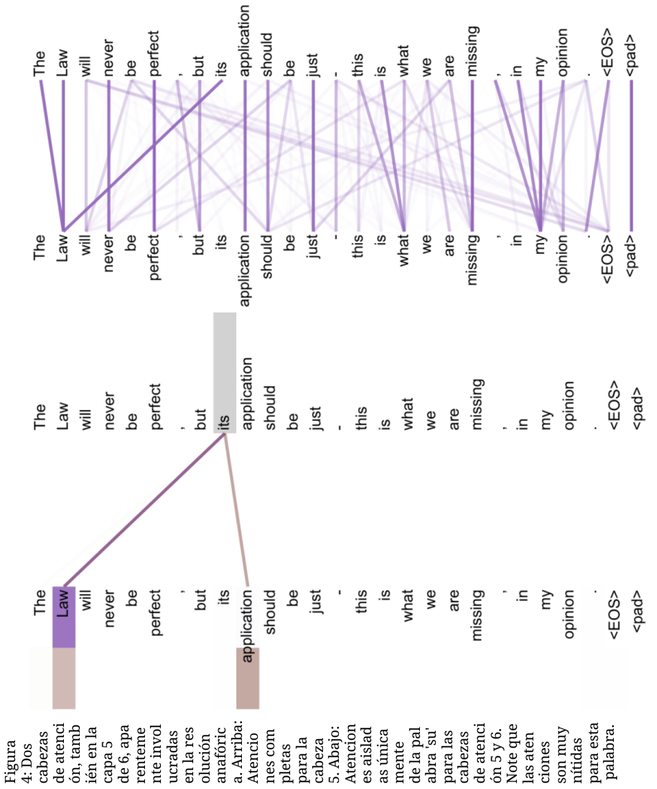


Figura 3: Un ejemplo del mecanismo de atención que sigue dependencias a larga distancia en la autoatención del codificador en la capa 5 de 6. Muchas de las cabezas de atención se centran en una dependencia distante del verbo 'making' (construcción/producción), completando la frase ' making...more difficult' (haciendo...más difícil). Las atenciones mostradas aquí son solo para la palabra ' making'. Los diferentes colores representan diferentes cabezas. Se visualiza mejor en color.



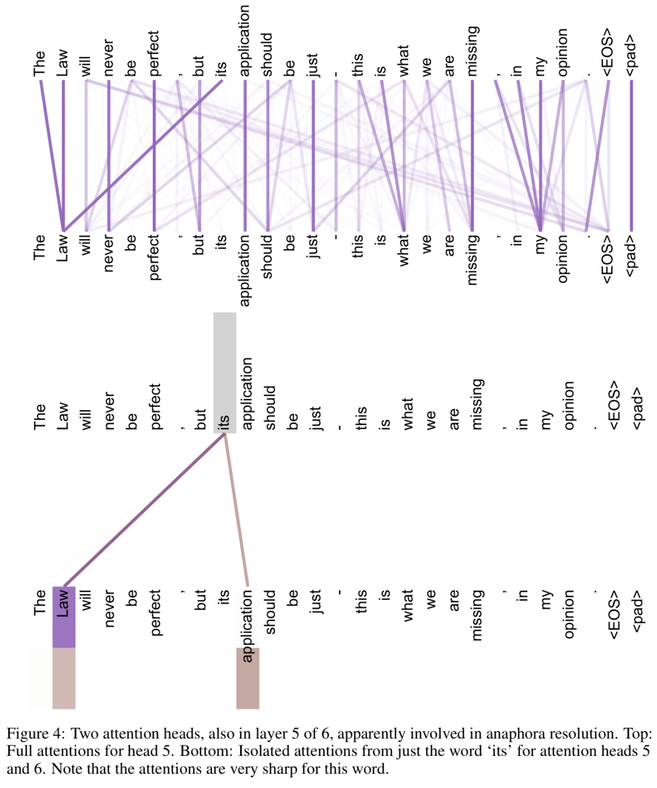


Figure 4: Two attention heads, also in layer 5 of 6, apparently involved in anaphora resolution. Top: Full attentions for head 5. Bottom: Isolated attentions from just the word 'its' for attention heads 5 and 6. Note that the attentions are very sharp for this word.

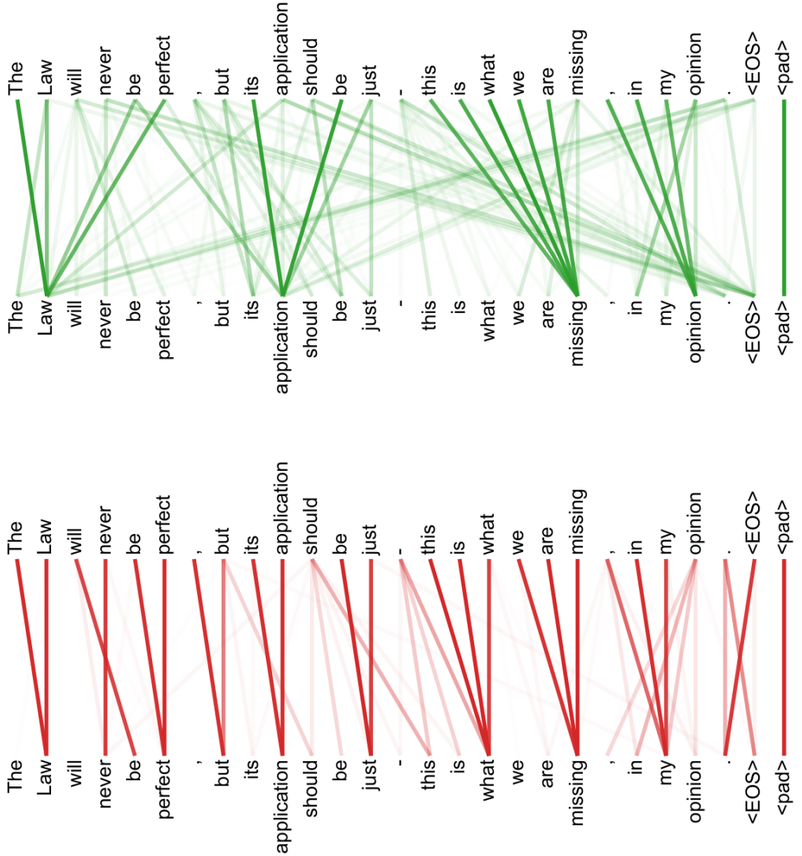


Figure 5: Many of the attention heads exhibit behaviour that seems related to the structure of the sentence. We give two such examples above, from two different heads from the encoder self-attention at layer 5 of 6. The heads clearly learned to perform different tasks.

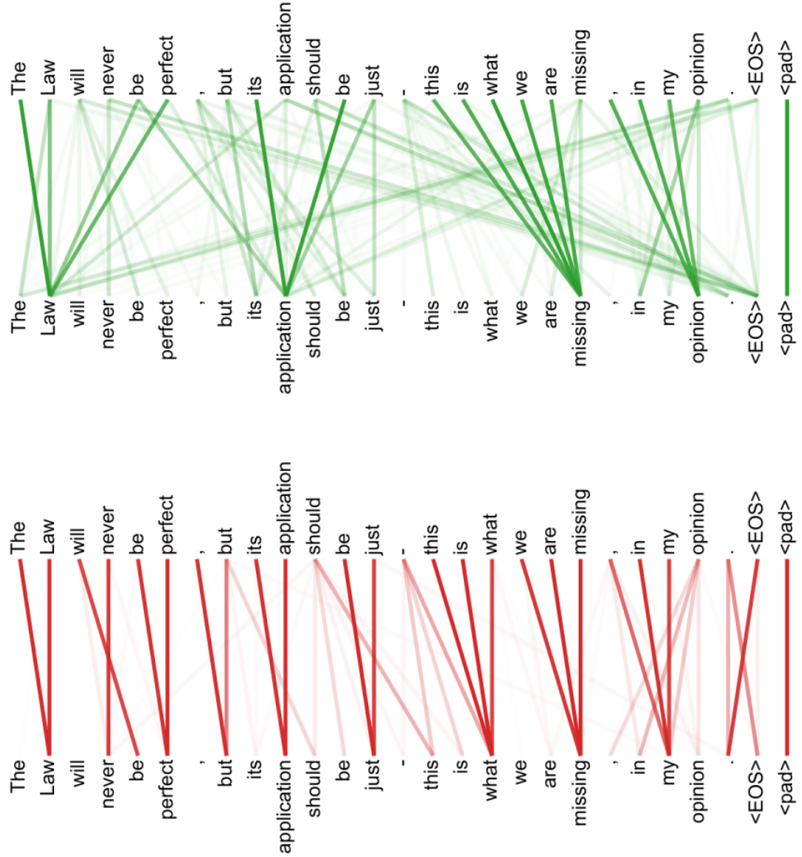


Figura 5: Muchas de las cabezas de atención exhiben un comportamiento que parece estar relacionado con la estructura de la oración. Presentamos dos ejemplos de estos arriba, provenientes de dos cabezas diferentes de la autoatención del codificador en la capa 5 de 6. Las cabezas aprendieron claramente a realizar tareas distintas.