# Transformer, BERT

Milan Straka

May 10, 2021

EUROPEAN UNION
European Structural and Investment Fund
Operational Programme Research, Development and Education

Charles University in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

# Attention is All You Need

For some sequence processing tasks, *sequential* processing (as performed by recurrent neural networks) of its elements might be too restrictive.

Instead, we may want to be able to combine sequence elements independently on their distance.

Such processing is allowed in the *Transformer* architecture, originally proposed for neural machine translation in 2017 in *Attention is All You Need* paper.

# Transformer

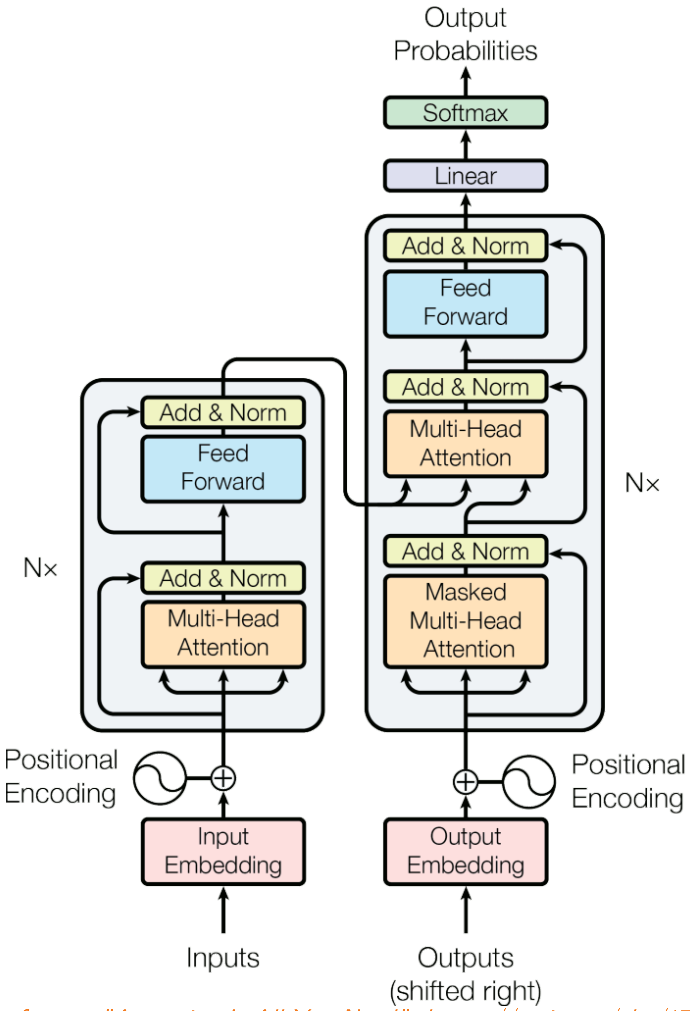
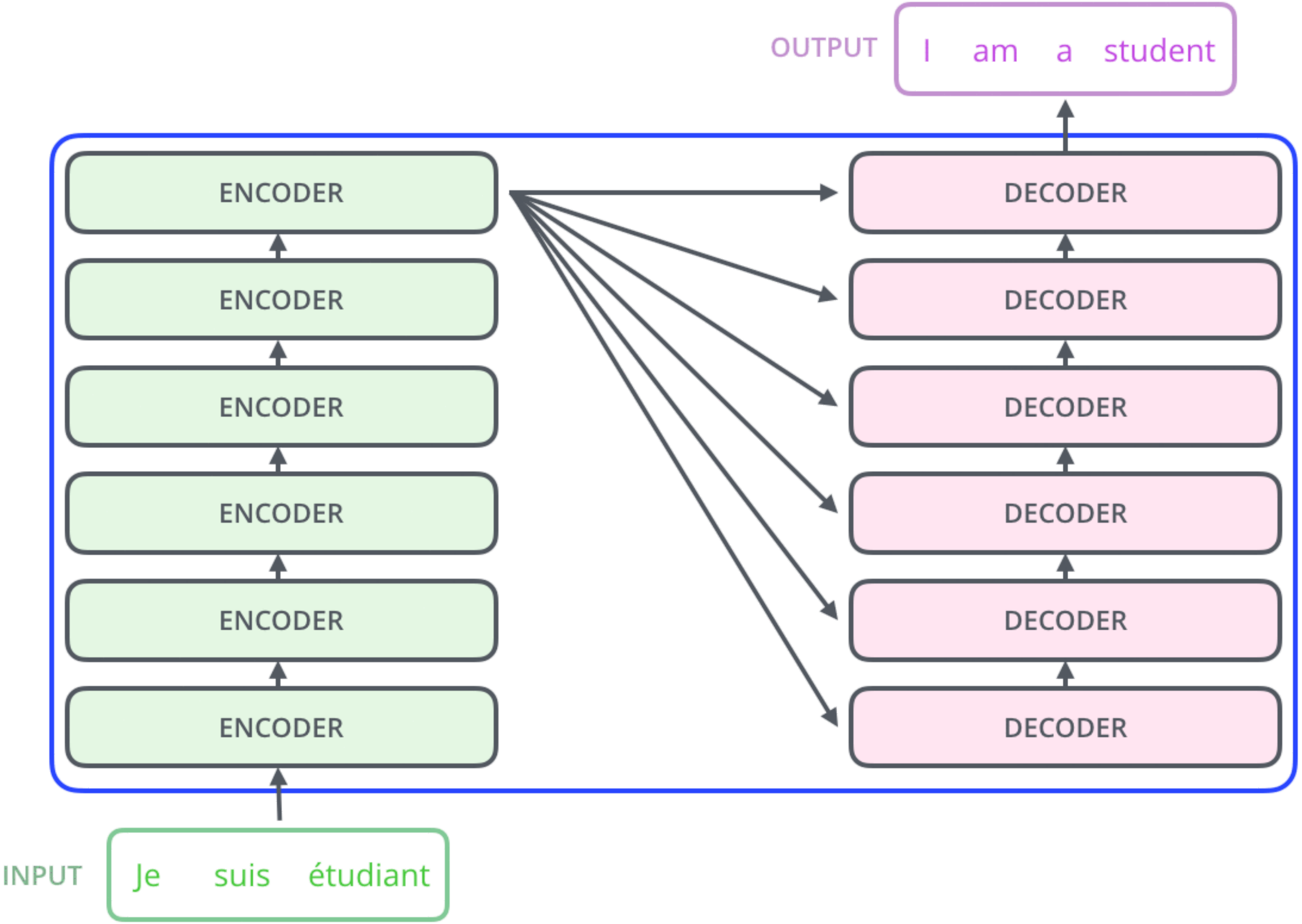


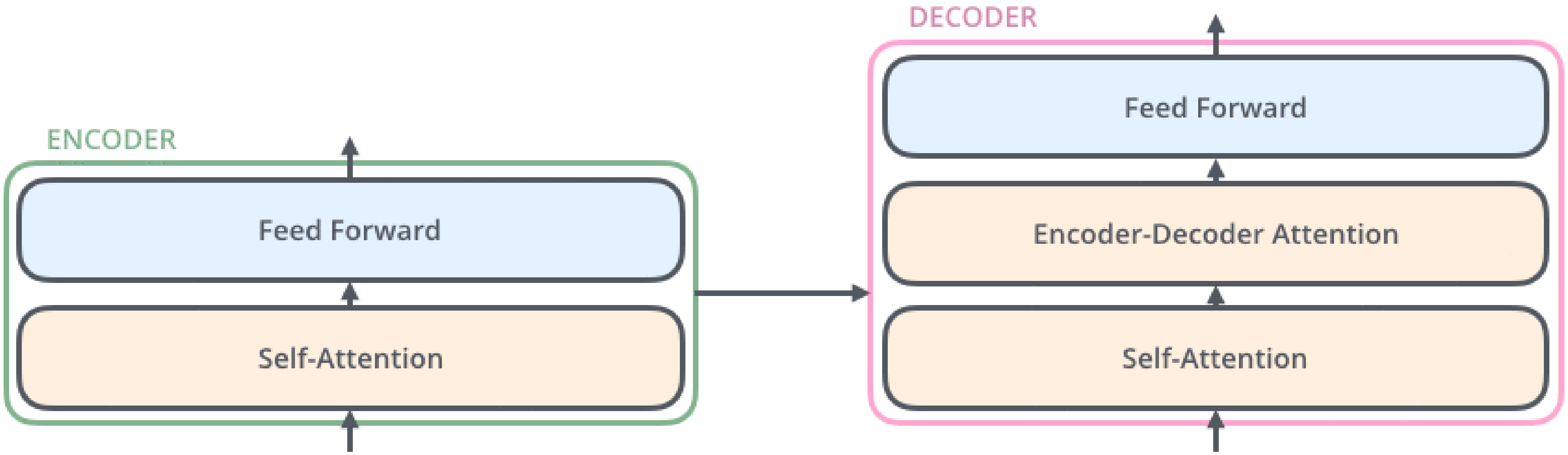

*Figure 1 of paper "Attention Is All You Need", https://arxiv.org/abs/1706.03762*

# Transformer



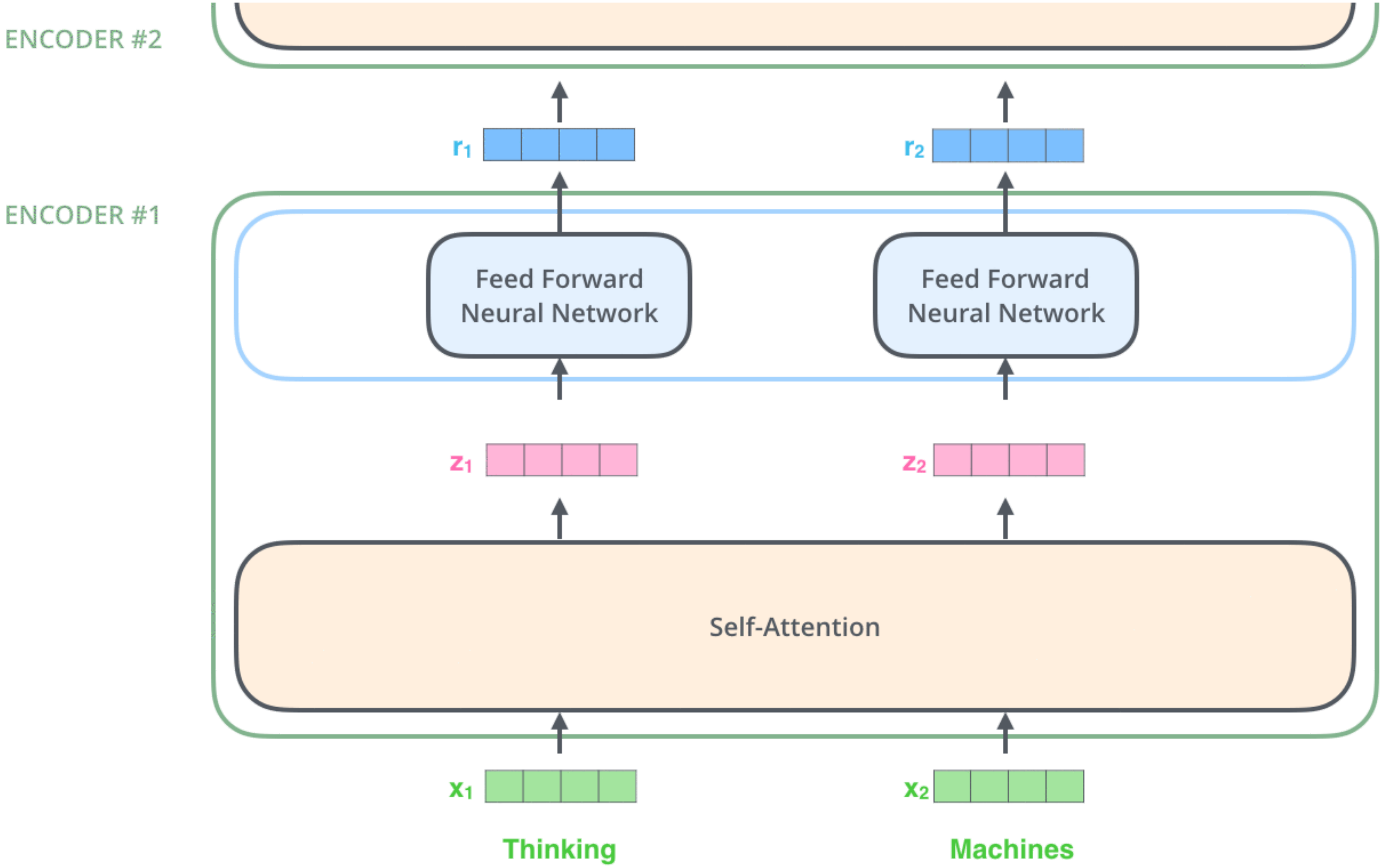

*http://jalammar.github.io/images/t/The_transformer_encoder_decoder_stack.png*

# Transformer




*http://jalammar.github.io/images/t/Transformer_decoder.png*

# Transformer




*http://jalammar.github.io/images/t/encoder_with_tensors_2.png*

Assume that we have a sequence of $n$ words represented using a matrix $\boldsymbol{X} \in \mathbb{R}^{n \times d}$.

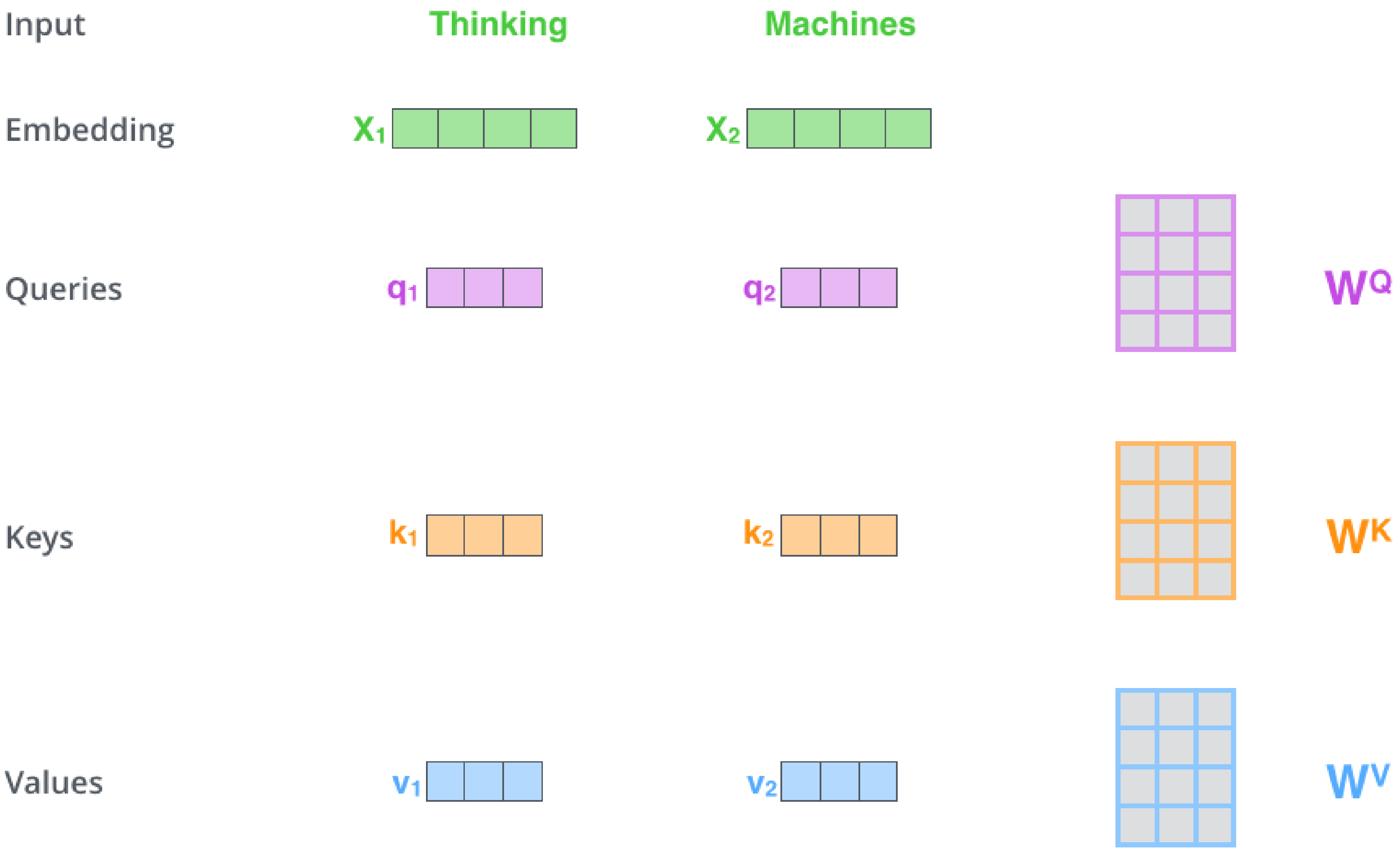
The attention module for a queries $\boldsymbol{Q} \in \mathbb{R}^{n \times d_k}$, keys $\boldsymbol{K} \in \mathbb{R}^{n \times d_k}$ and values $\boldsymbol{V} \in \mathbb{R}^{n \times d_v}$ is defined as:

$$\operatorname{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \operatorname{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}}\right)\boldsymbol{V}.$$

The queries, keys and values are computed from the input word representations $\boldsymbol{X}$ using a linear transformation as

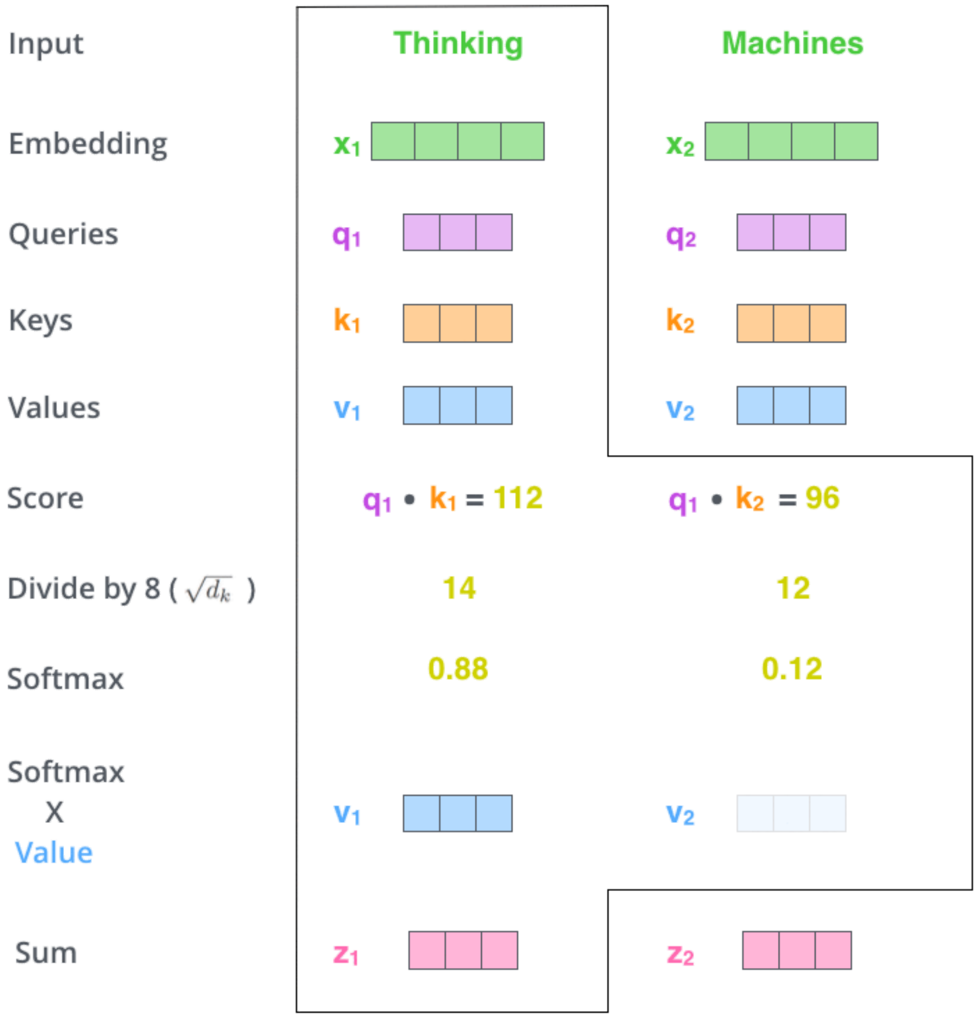
$$\begin{aligned} \boldsymbol{Q} &= \boldsymbol{W}^Q \cdot \boldsymbol{X} \\ \boldsymbol{K} &= \boldsymbol{W}^K \cdot \boldsymbol{X} \\ \boldsymbol{V} &= \boldsymbol{W}^V \cdot \boldsymbol{X} \end{aligned}$$

# Transformer – Self-Attention

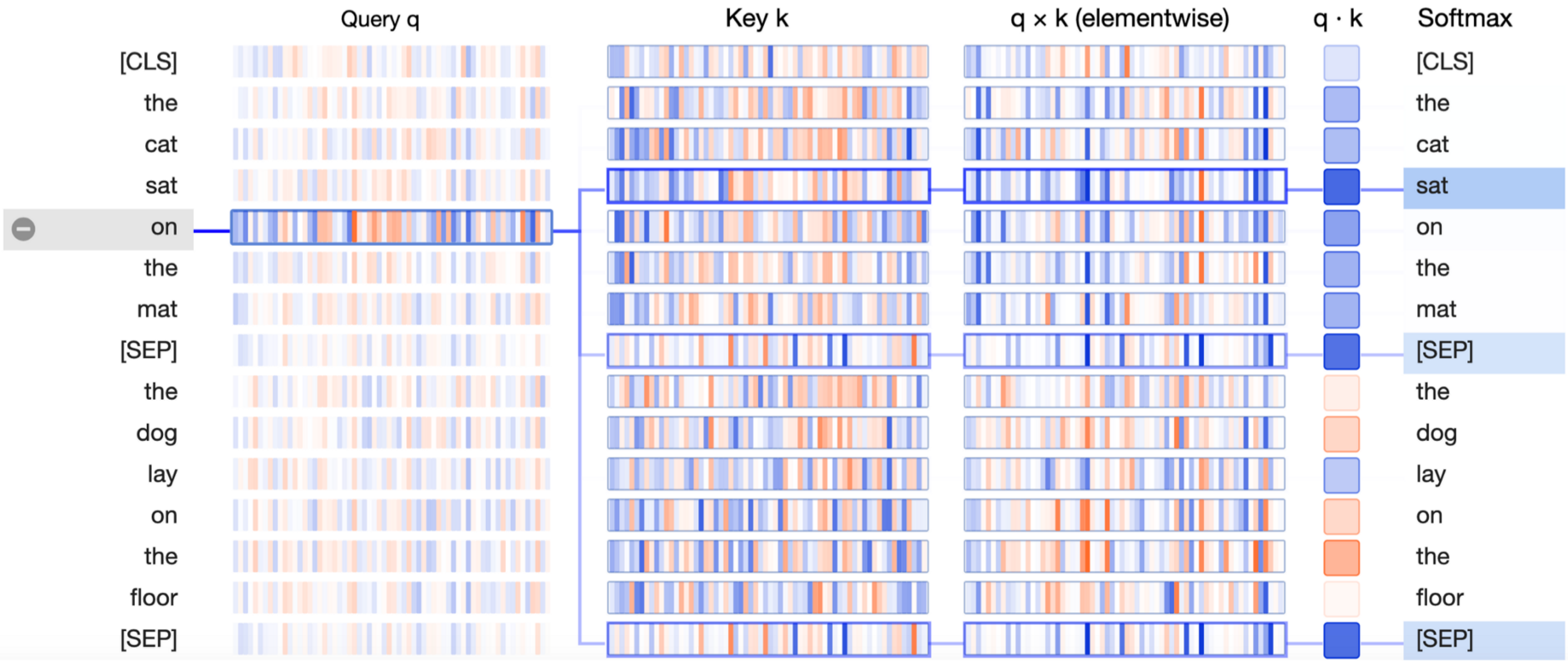



http://jalammar.github.io/images/t/transformer_self_attention_vectors.png

# Transformer – Self-Attention

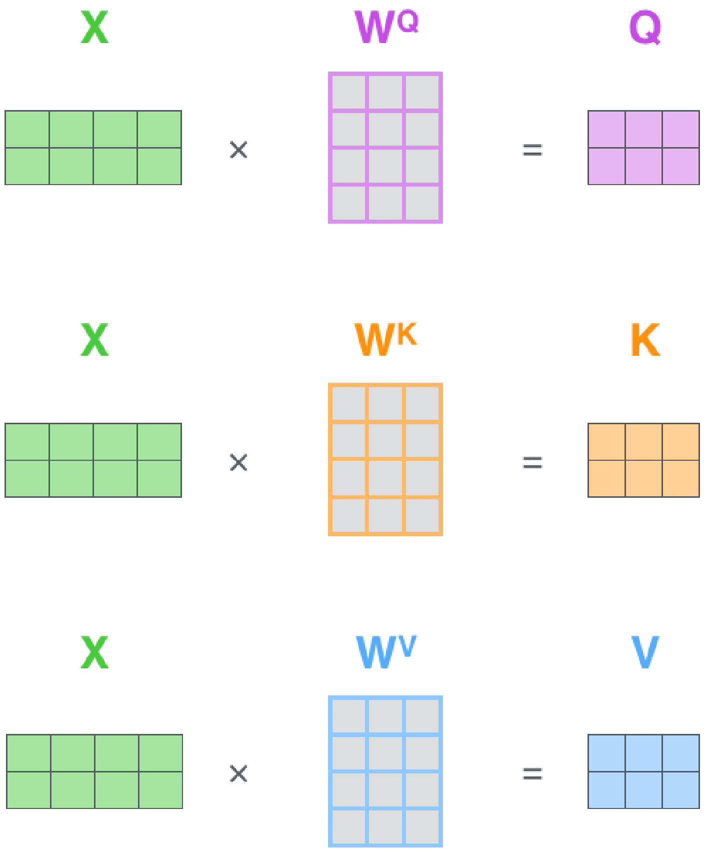



*http://jalammar.github.io/images/t/self-attention-output.png*

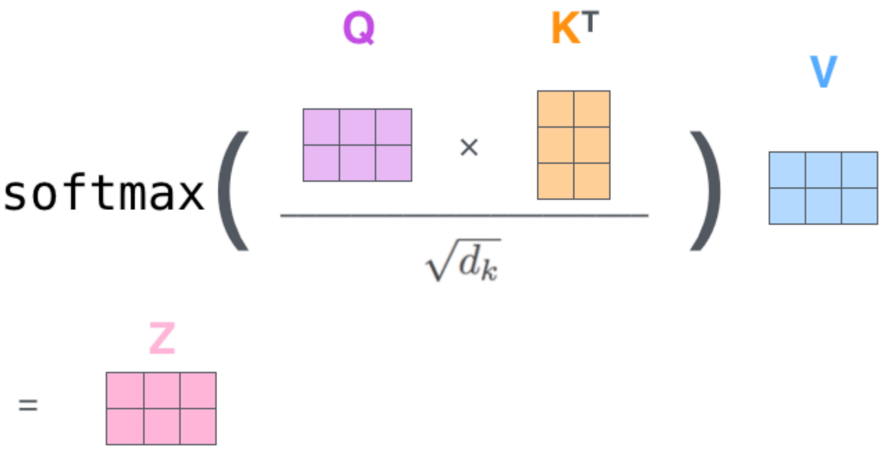



*https://miro.medium.com/max/2000/1*jBsfVNOOcJ-I3tsLVgni_w.png*

# Transformer – Self-Attention




*http://jalammar.github.io/images/t/self-attention-matrix-calculation.png*




*http://jalammar.github.io/images/t/self-attention-matrix-calculation-2.png*

# Transformer – Multihead Attention

Multihead attention is used in practice. Instead of using one huge attention, we split queries, keys and values to several groups (similar to how ResNeXt works), compute the attention in each of the groups separately, concatenate the results and multiply them by a matrix $\boldsymbol{W}^O$.

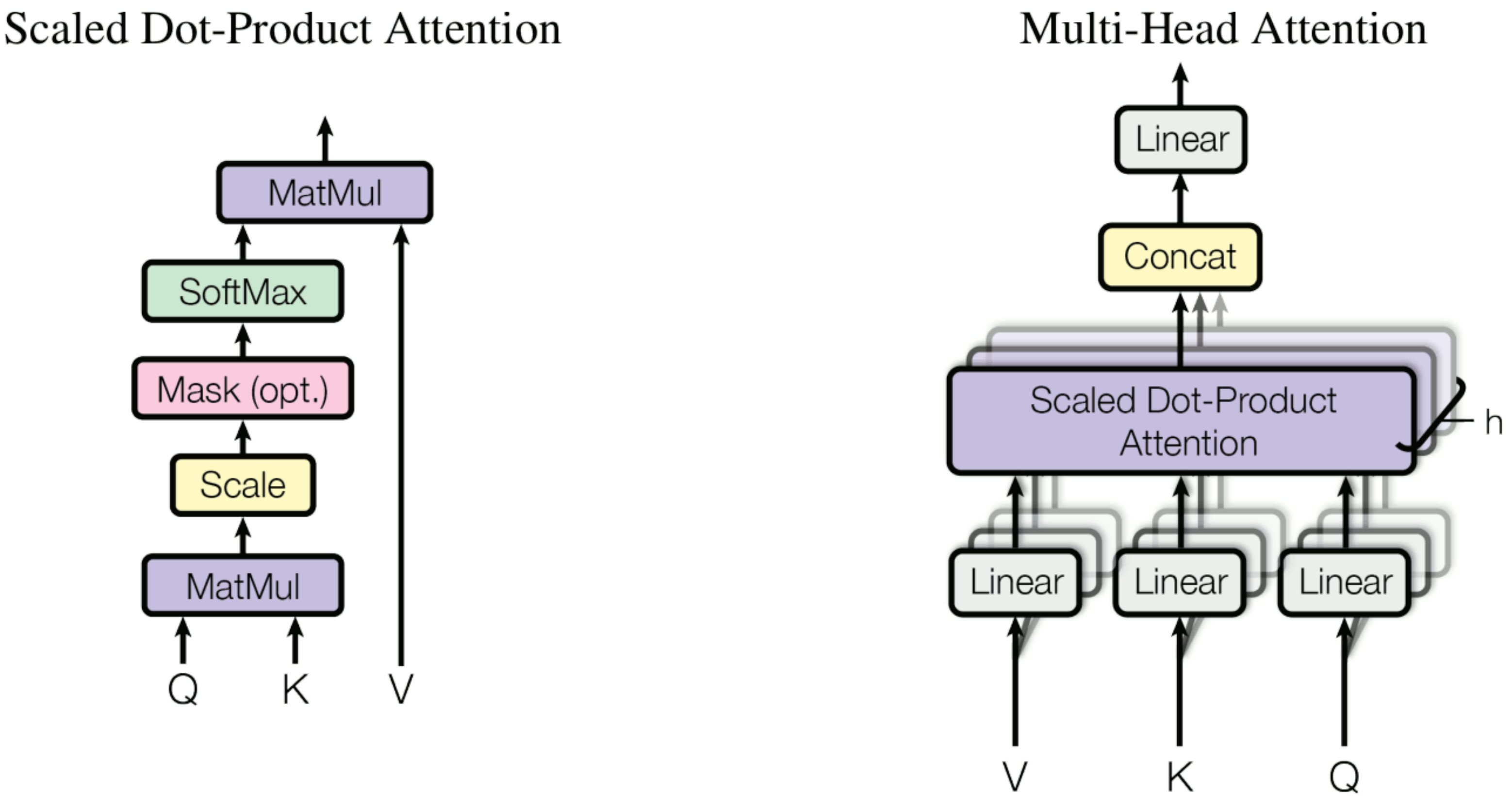
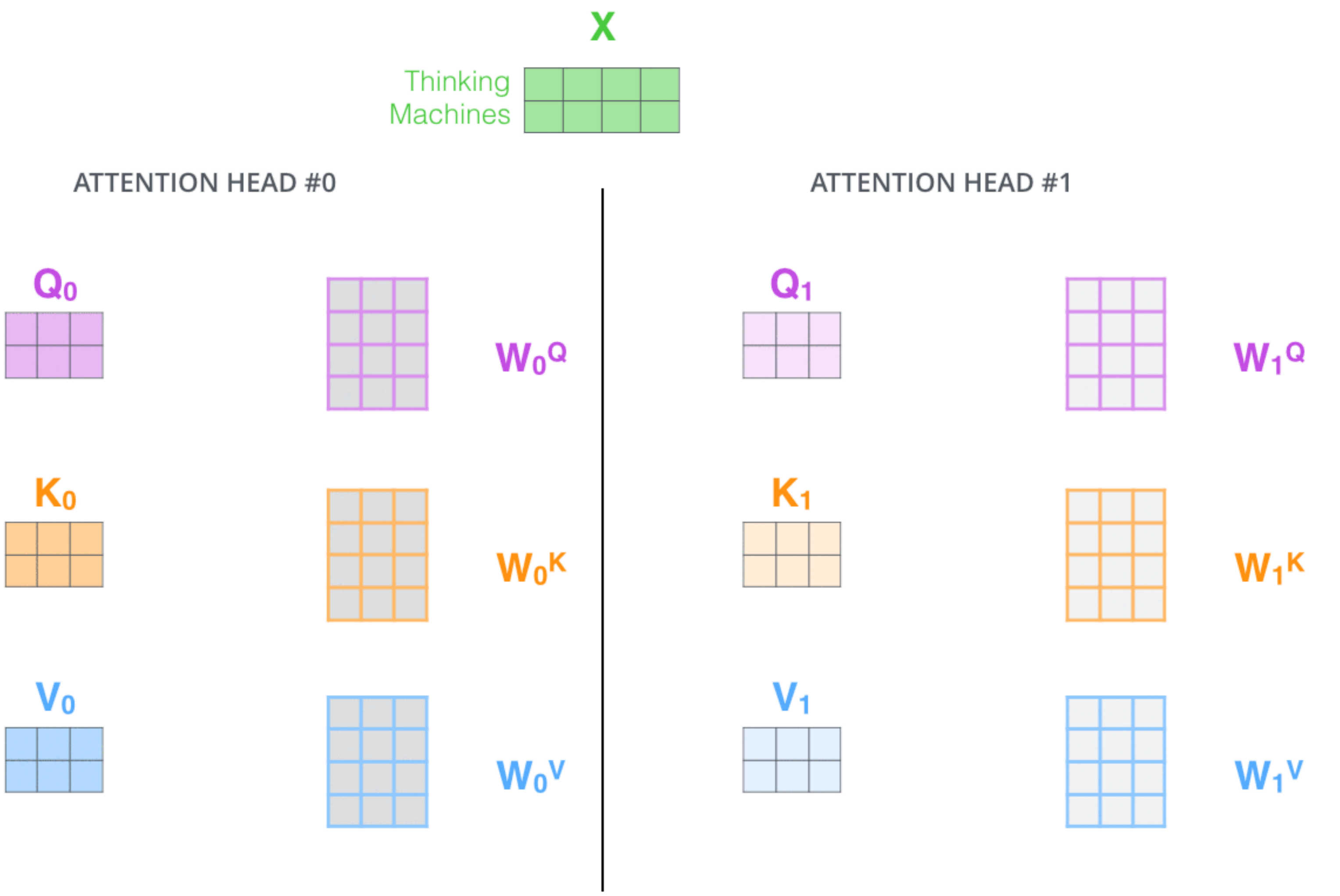
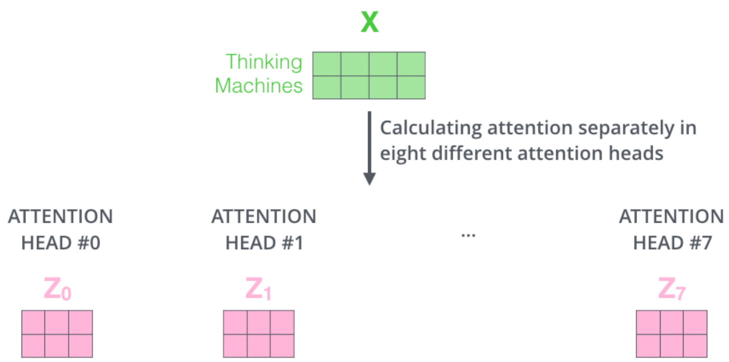


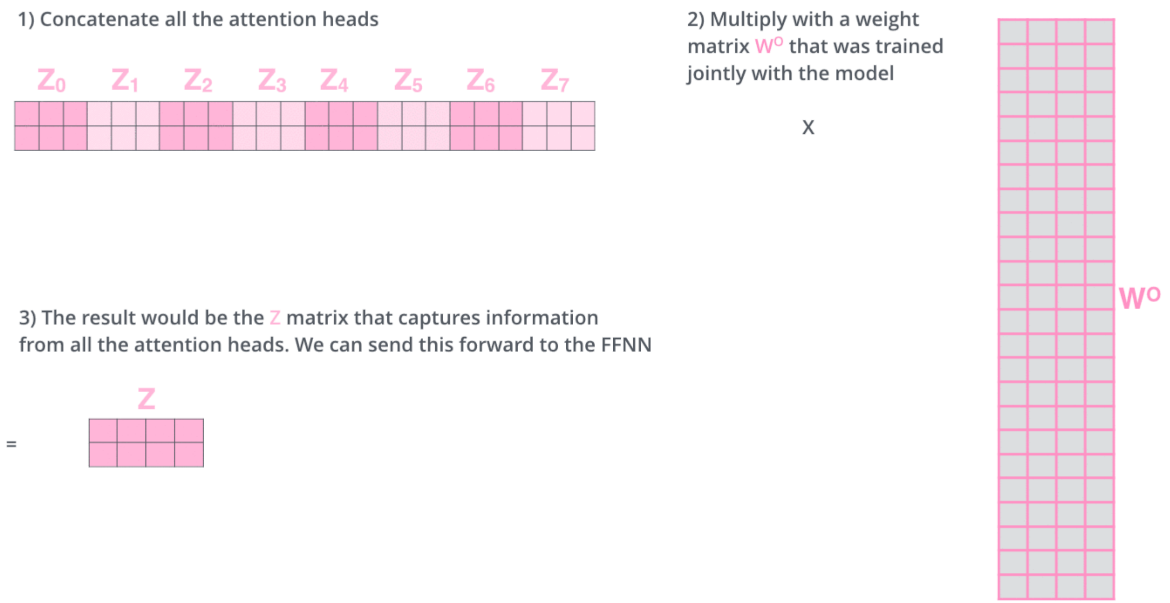

*Figure 2 of paper "Attention Is All You Need", https://arxiv.org/abs/1706.03762*




http://jalammar.github.io/images/t/transformer_attention_heads_qkv.png

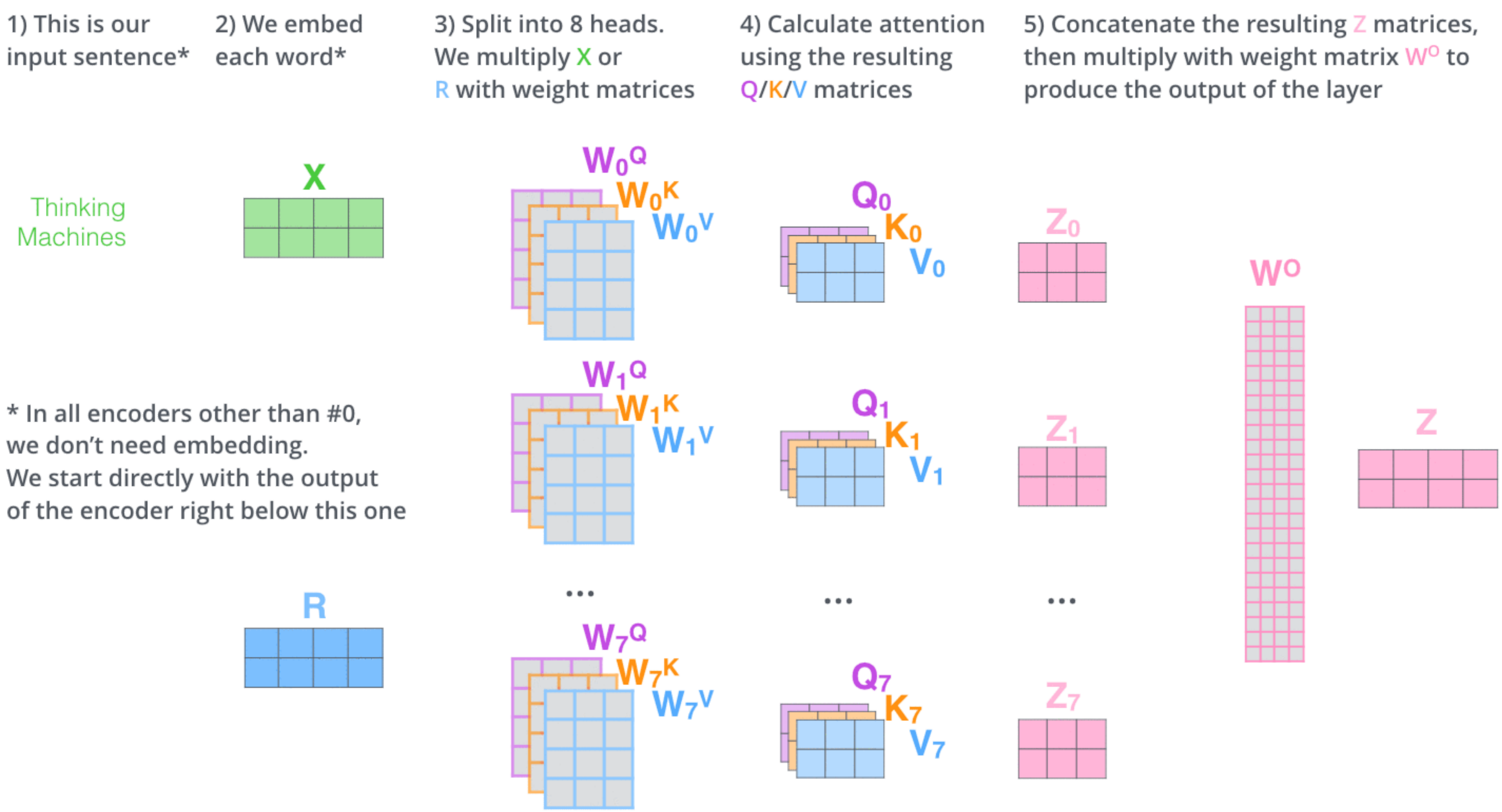



*http://jalammar.github.io/images/t/transformer_attention_heads_z.png*




*http://jalammar.github.io/images/t/transformer_attention_heads_weight_matrix_o.png*

# Transformer – Multihead Attention




*http://jalammar.github.io/images/t/transformer_multi-headed_self-attention-recap.png*

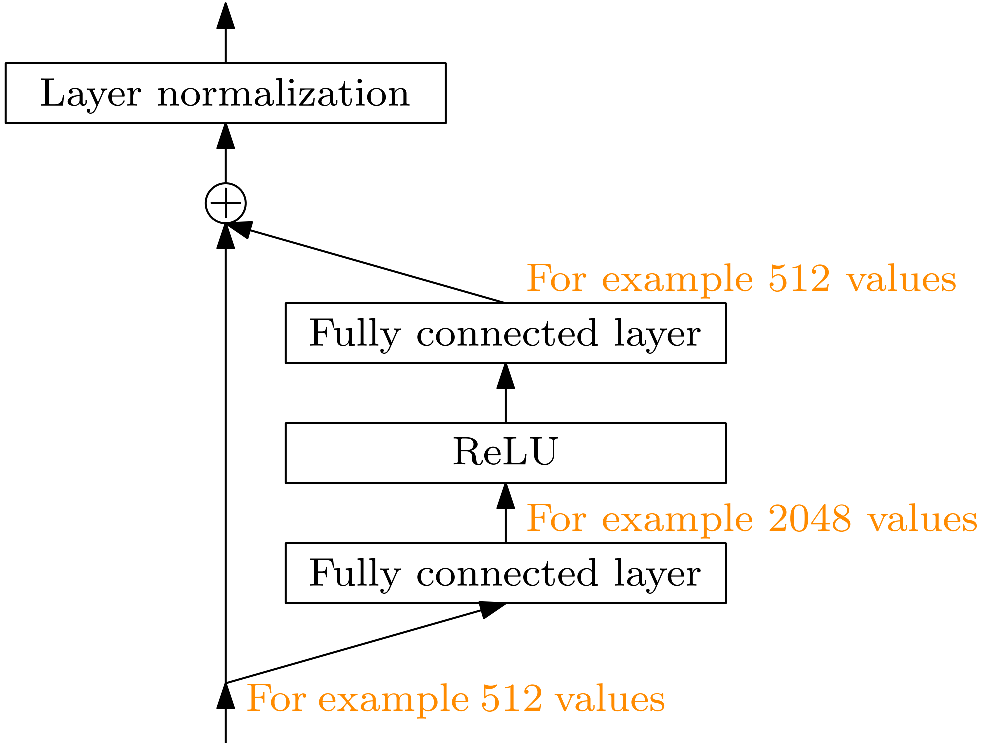
Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. $n$ is the sequence length, $d$ is the representation dimension, $k$ is the kernel size of convolutions and $r$ the size of the neighborhood in restricted self-attention.

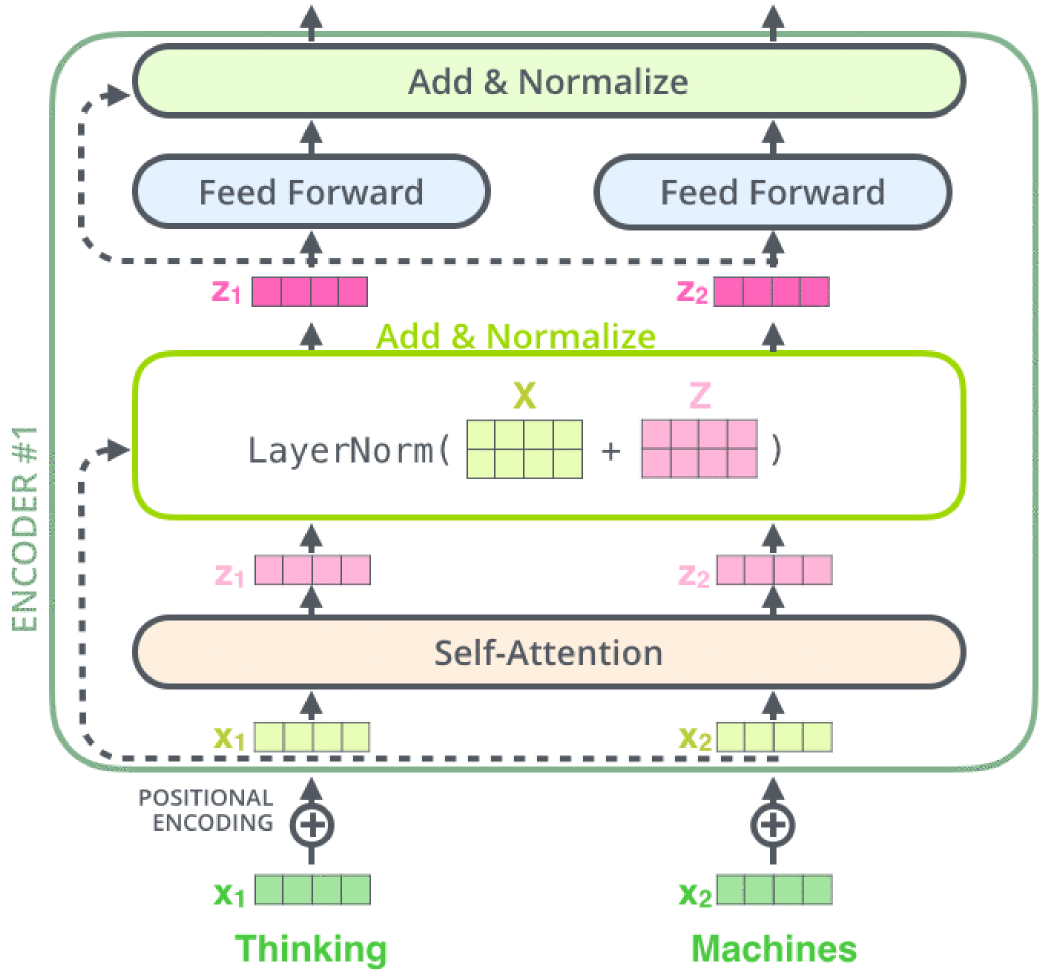
| Layer Type | Complexity per Layer | Sequential Operations | Maximum Path Length |
|---|---|---|---|
| Self-Attention | $O(n^2 \cdot d)$ | $O(1)$ | $O(1)$ |
| Recurrent | $O(n \cdot d^2)$ | $O(n)$ | $O(n)$ |
| Convolutional | $O(k \cdot n \cdot d^2)$ | $O(1)$ | $O(log_k(n))$ |
| Self-Attention (restricted)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

*Table 1 of paper "Attention Is All You Need", https://arxiv.org/abs/1706.03762*

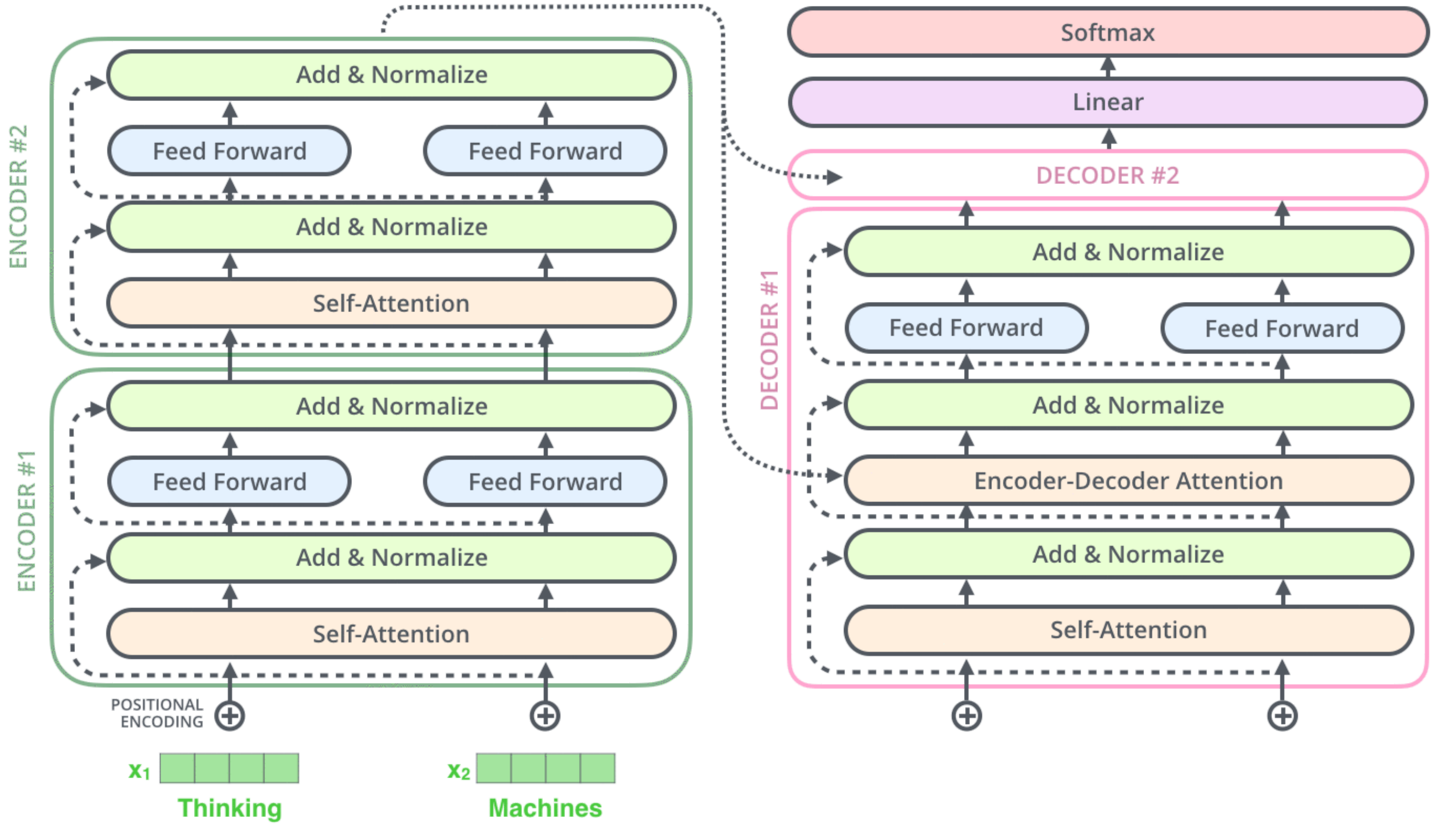
## Feed Forward Networks

The self-attention is complemented with FFN layers, which is a fully connected ReLU layer with four times as many hidden units as inputs, followed by another fully connected layer without activation.






*http://jalammar.github.io/images/t/transformer_resideual_layer_norm_2.png*




*http://jalammar.github.io/images/t/transformer_resideual_layer_norm_3.png*

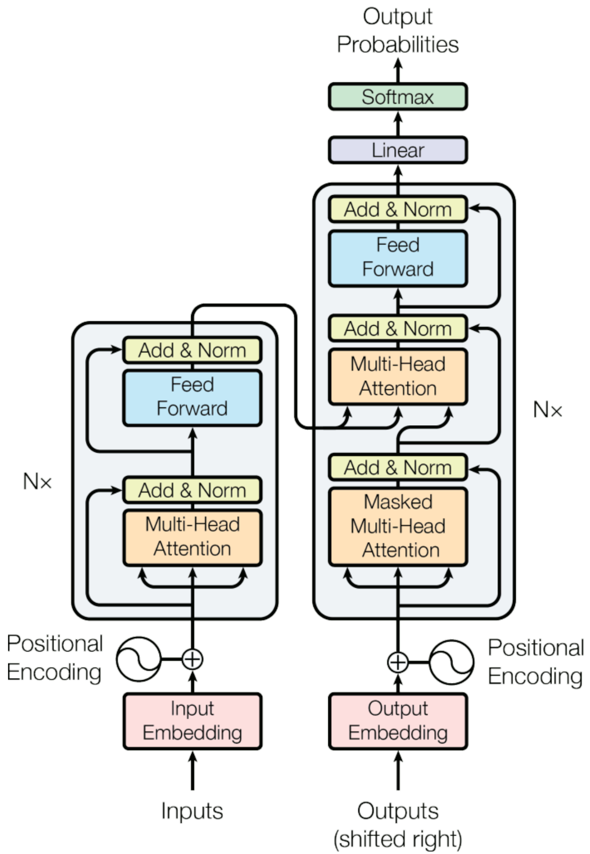



*Figure 1 of paper "Attention Is All You Need", https://arxiv.org/abs/1706.03762*

## Masked Self-Attention

During decoding, the self-attention must attent only to earlier positions in the output sequence.

This is achieved by *masking* future positions, i.e., zeroing their weights out, which is usually implemented by setting them to $-\infty$ before the $\mathrm{softmax}$ calculation.

## Encoder-Decoder Attention

In the encoder-decoder attentions, the *queries* comes from the decoder, while the *keys* and the *values* originate from the encoder.

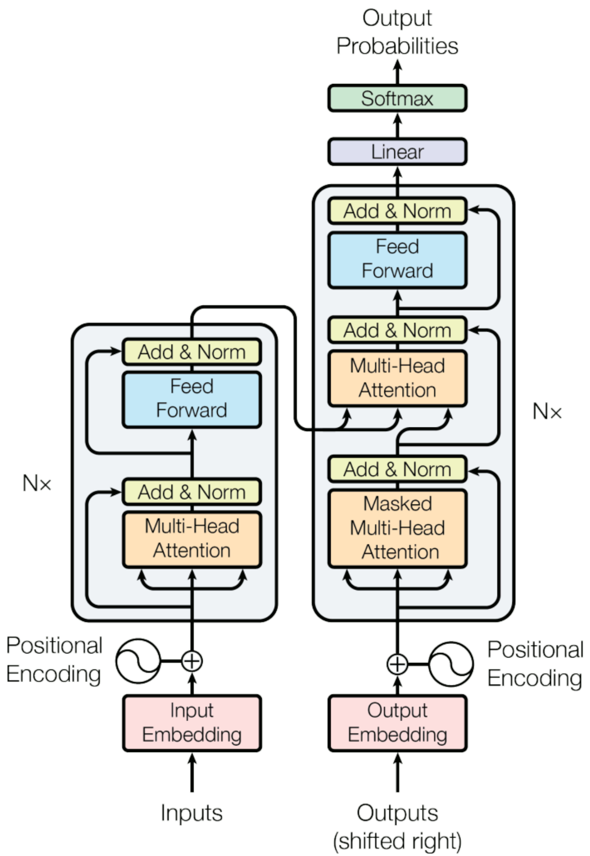
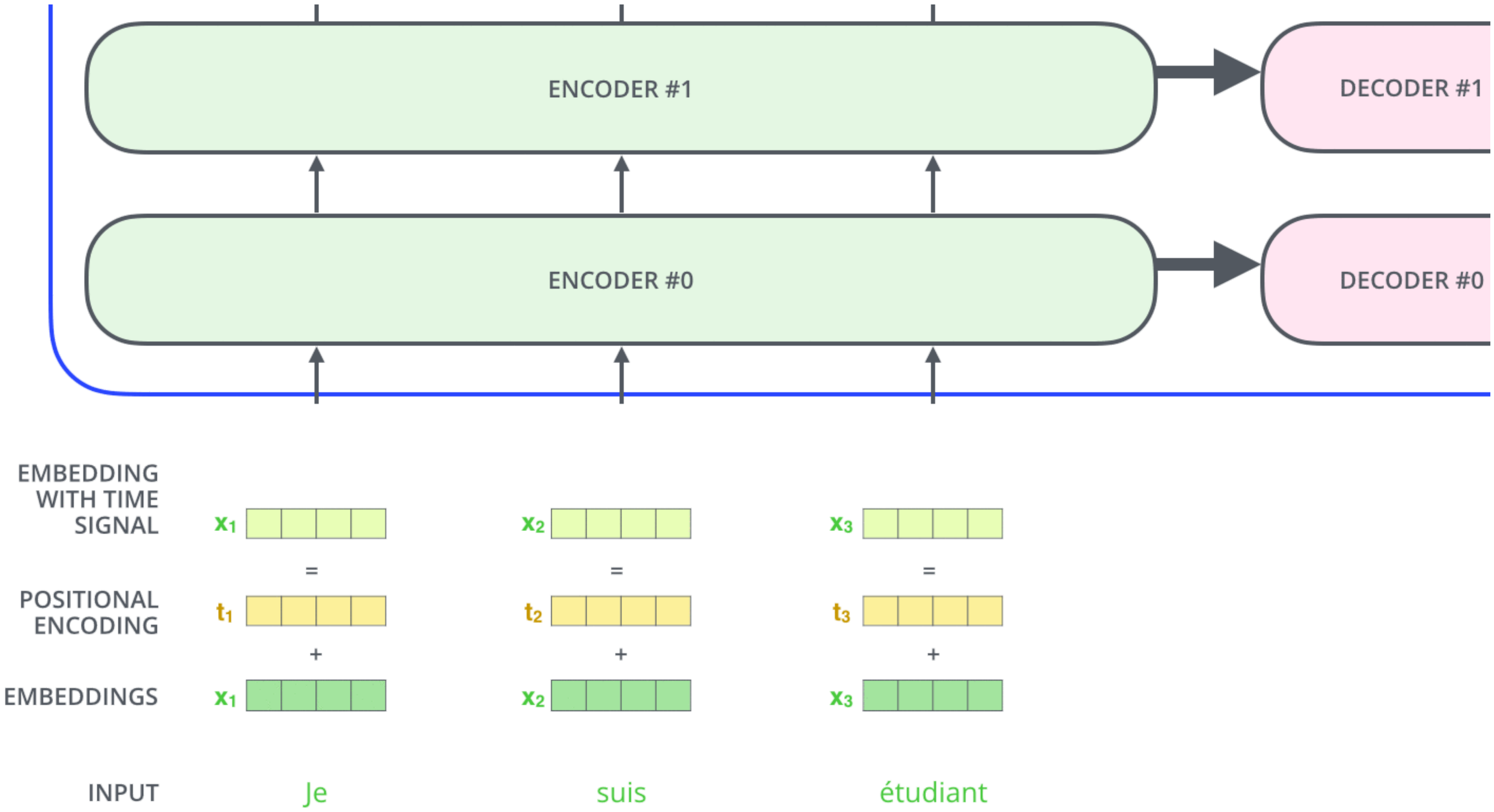



*Figure 1 of paper "Attention Is All You Need", https://arxiv.org/abs/1706.03762*

# Transformer – Positional Embedding




*http://jalammar.github.io/images/t/transformer_positional_encoding_vectors.png*

## Positional Embeddings

We need to encode positional information (which was implicit in RNNs).

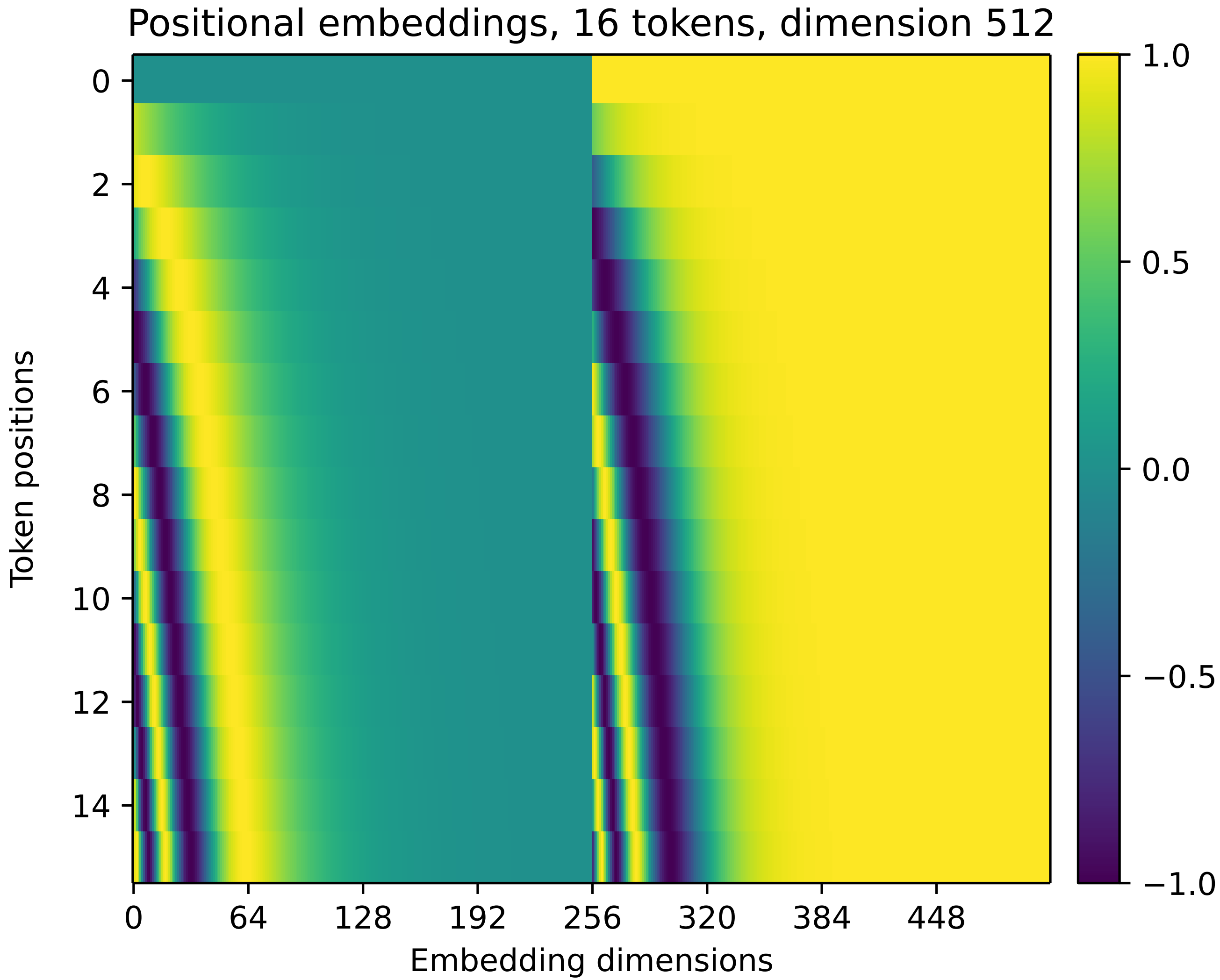
- Learned embeddings for every position.
- Sinusoids of different frequencies:

$$\begin{aligned}\mathrm{PE}_{(pos,2i)} &= \sin\left(pos/10000^{2i/d}\right)\\ \mathrm{PE}_{(pos,2i+1)} &= \cos\left(pos/10000^{2i/d}\right)\end{aligned}$$

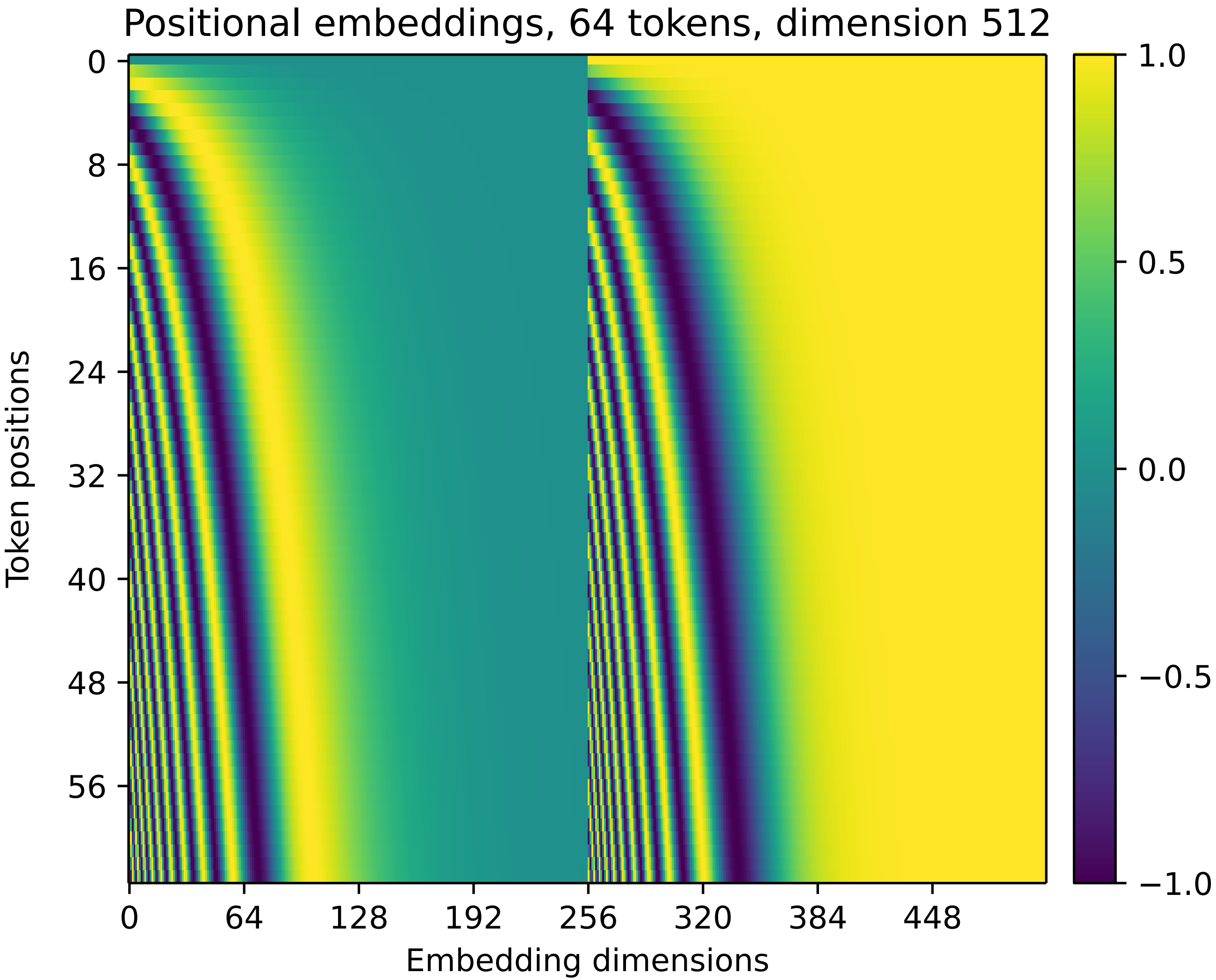
This choice of functions should allow the model to attend to relative positions, since for any fixed $k$, $\mathrm{PE}_{pos+k}$ is a linear function of $\mathrm{PE}_{pos}$, because

$$\begin{aligned}\mathrm{PE}_{(pos+k,2i)} &= \sin\left((pos+k)/10000^{2i/d}\right)\\ &= \sin\left(pos/10000^{2i/d}\right)\cdot\cos\left(k/10000^{2i/d}\right) + \cos\left(pos/10000^{2i/d}\right)\cdot\sin\left(k/10000^{2i/d}\right)\\ &= \mathit{offset}_{(k,2i)}\cdot\mathrm{PE}_{(pos,2i)} + \mathit{offset}_{(k,2i+1)}\cdot\mathrm{PE}_{(pos,2i+1)}.\end{aligned}$$

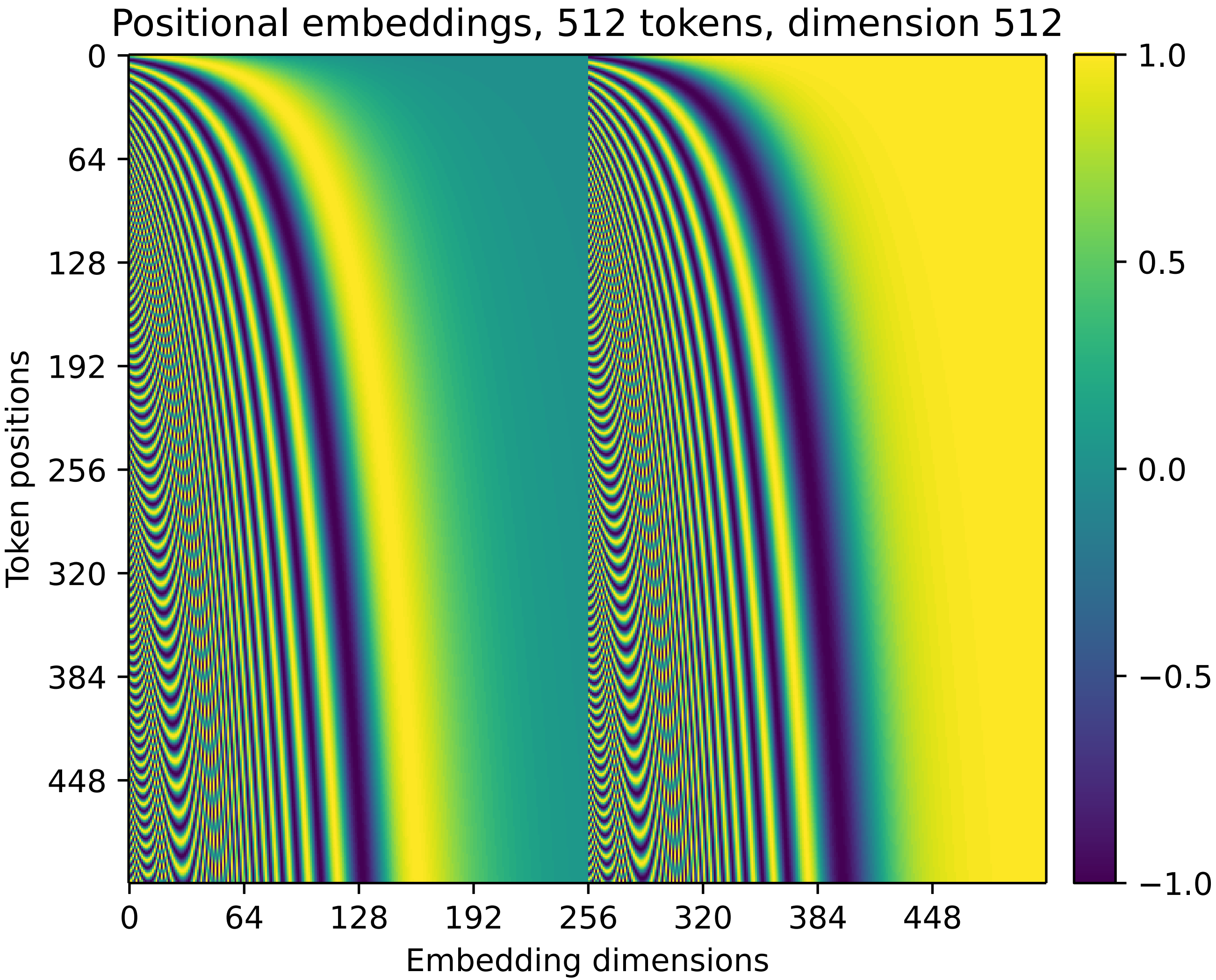
# Transformer – Positional Embeddings



# Transformer – Positional Embeddings



# Transformer – Positional Embeddings



## Regularization

The network is regularized by:

- dropout of input embeddings,
- dropout of each sub-layer, just before before it is added to the residual connection (and then normalized),
- label smoothing.

Default dropout rate and also label smoothing weight is 0.1.

## Parallel Execution

Because of the *masked attention*, training can be performed in parallel.

However, inference is still sequential.

## Optimizer

Adam optimizer (with $\beta_2 = 0.98$, smaller than the default value of $0.999$) is used during training, with the learning rate decreasing proportionally to inverse square root of the step number.

## Warmup

Furthermore, during the first *warmup_steps* updates, the learning rate is increased linearly from zero to its target value.

$$learning\_rate = \frac{1}{\sqrt{d_\textrm{model}}} \min\left(\frac{1}{\sqrt{step\_num}}, \frac{step\_num}{warmup\_steps} \cdot \frac{1}{\sqrt{warmup\_steps}}\right).$$

In the original paper, 4000 warmup steps were proposed.

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** | $2.3 \cdot 10^{19}$ | |

*Table 2 of paper "Attention Is All You Need", https://arxiv.org/abs/1706.03762*

# Transformers Results

| | $N$ | $d_{\text{model}}$ | $d_{\text{ff}}$ | $h$ | $d_k$ | $d_v$ | $P_{drop}$ | $\epsilon_{ls}$ | 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 |

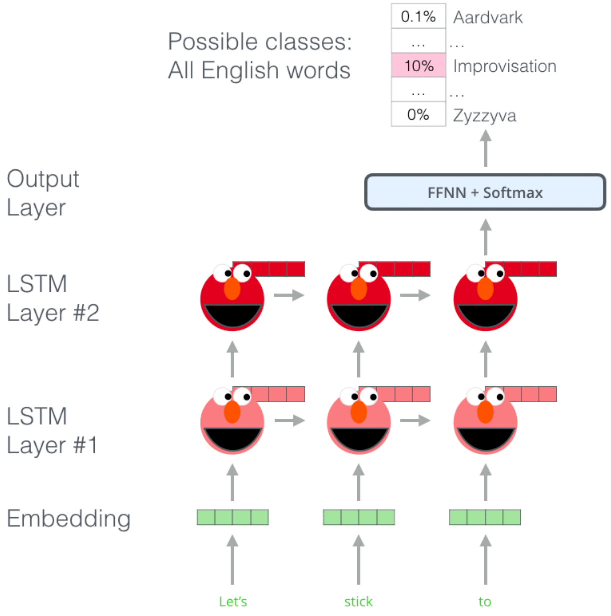
*Table 4 of paper "Attention Is All You Need", https://arxiv.org/abs/1706.03762*

## Main Takeaway

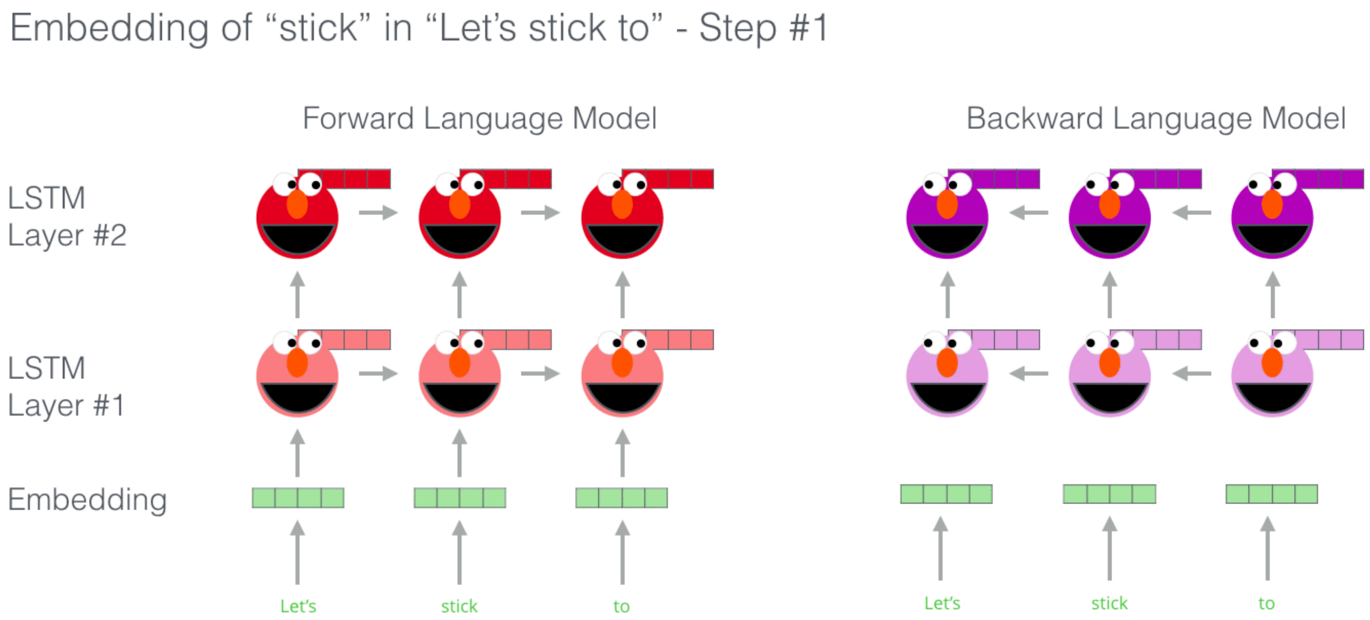
Generally, Transformer provides more powerful sequence-to-sequence architecture and also sequence element representation architecture than RNNs, but usually requires substantially more data.

At the end of 2017, a new type of *deep contextualized* word representations was proposed by Peters et al., called ELMo, **E**mbeddings from **L**anguage **Mo**dels.

The ELMo embeddings were based on a two-layer pre-trained LSTM language model, where a language model predicts following word based on a sentence prefix. Specifically, two such models were used, one for the forward direction and the other one for the backward direction.



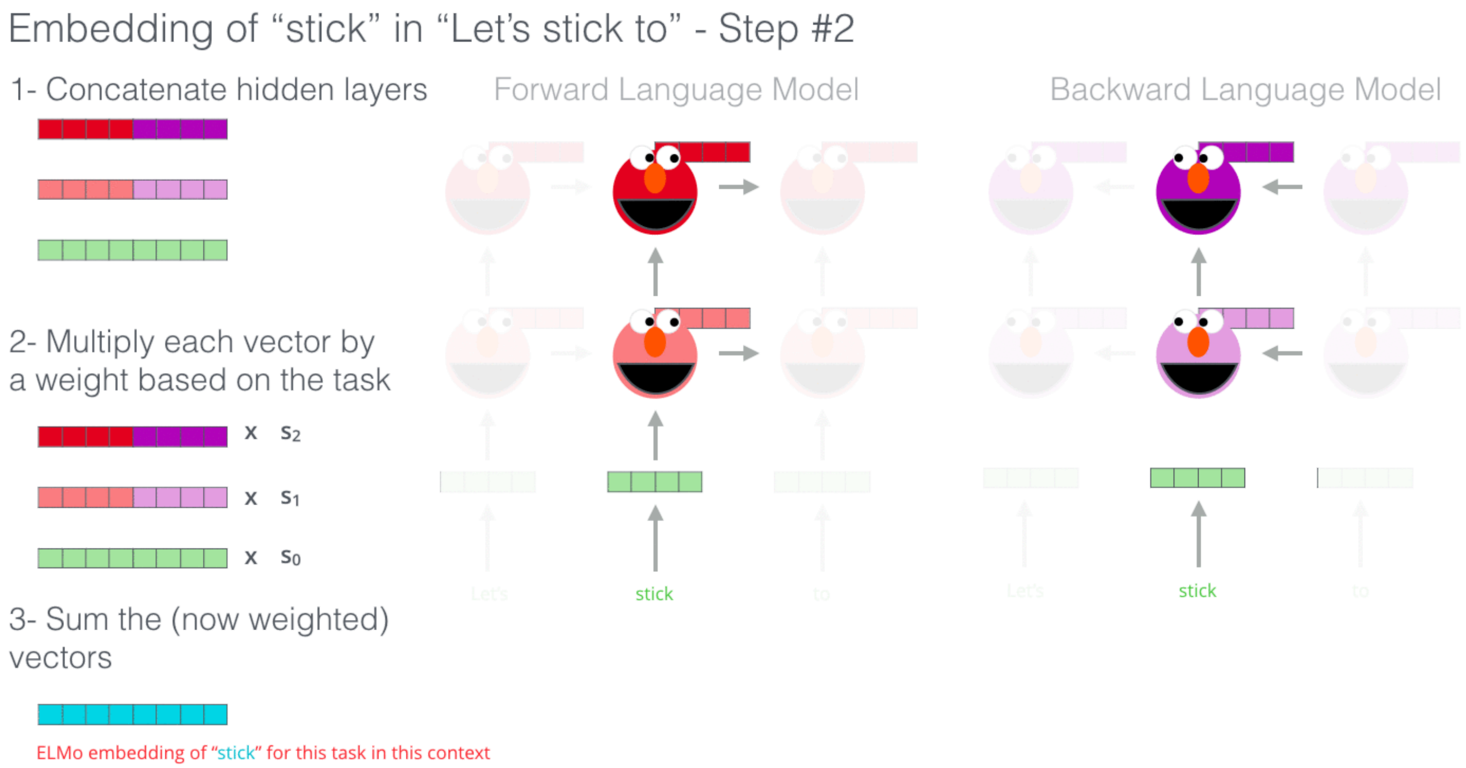

*http://jalammar.github.io/images/Bert-language-modeling.png*




*http://jalammar.github.io/images/elmo-forward-backward-language-model-embedding.png*

# ELMo

To compute an embedding of a word in a sentence, the concatenation of the two language model's hidden states is used.




*http://jalammar.github.io/images/elmo-embedding.png*

Pre-trained ELMo embeddings substantially improved several NLP tasks.

A year later after ELMo, at the end of 2018, a new model called BERT (standing for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers) was proposed. It is nowadays one of the most dominating approaches for pre-training word embeddings and for paragraph representation.


*https://www.sesameworkshop.org/sites/default/files/imageservicecache/2019-03/header5120x1620_50thanniversary.png/4b00e17bb509f5c630c57c318b37d0da.webp*

In the BERT model computes contextualized representations using a bidirectional Transformer architecture.

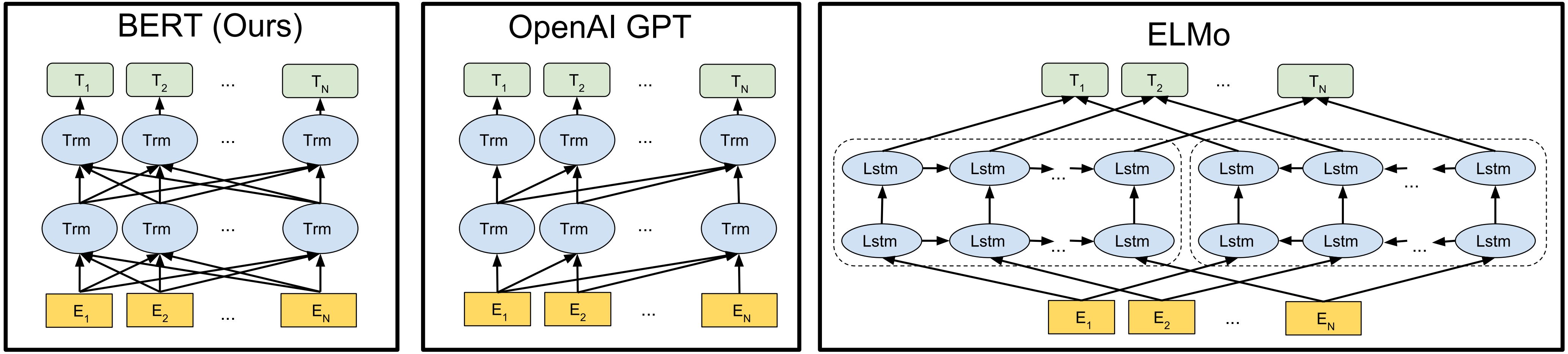
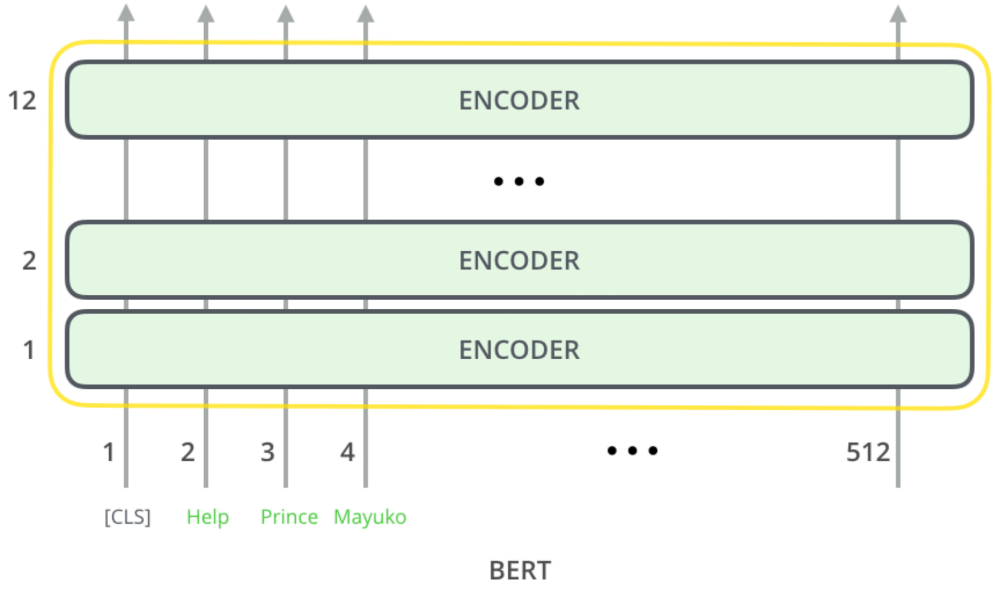



*Figure 3 of paper "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" , https://arxiv.org/abs/1810.04805*

The baseline BERT base model consists of 12 Transformer layers:




*http://jalammar.github.io/images/bert-encoders-input.png*

The bidirectionality is important, but it makes training difficult.

The input to the BERT model are two so-called *sentences*, but they are in fact pieces of text with hundreds of subwords (512 maximum in total). The first token is a special `CLS` token and every sentence is ended by a `SEP` token. Additionally, a trainable embedding indicating if a token belongs to a sentence `A` (inclusively up to its `SEP` token) or to sentence `B` is used.

The BERT model is pretrained using two objectives:

- **masked language model** – 15% of the input words are *masked*, and the model tries to predict them.
  - 80% of them are replaced by a special `MASK` token;
  - 10% of them are replaced by a random word;
  - 10% of them are left intact.
- **next sentence prediction** – the model tries to predict whether the second *sentence* followed the first one in the raw corpus.
  - 50% of the time the second sentence is the actual next sentence;
  - 50% of the time the second sentence is a random sentence from the corpus.

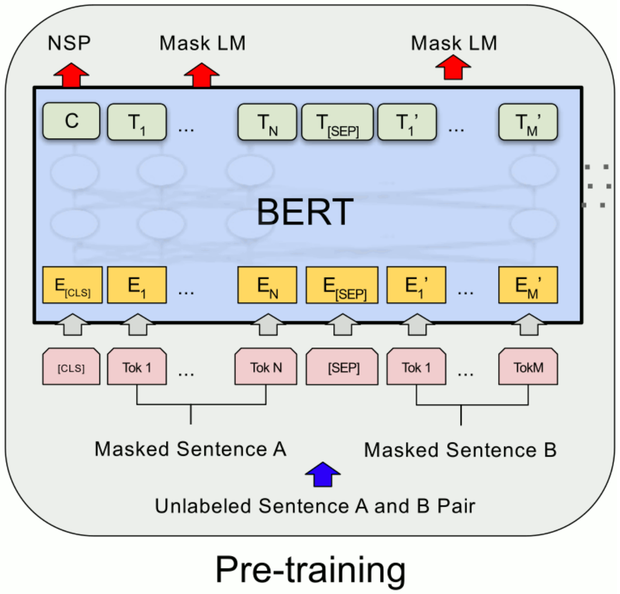



*Figure 1 of paper "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", https://arxiv.org/abs/1810.04805.*

For pre-training, English BookCorpus (800M words) and Wikipedia (2,500M words) are used, with a 30k WordPieces vocabulary.

Batch size is 256 sequences, each 512 subwords, giving 128k tokens per batch. Adam with learning rate 1e-4 is used, with linear learning rate warmup for the first 10k steps, followed by a linear learning rate decay to 0. Standard momentum parameters are used, and L2 weight decay of 0.01 is utilized.

Dropout of 0.1 on all layers is used, and GELU activation is used instead of ReLU.

Furthermore, because longer sequences are quadratically more expensive, first 90% of the pre-training is performed on sequences of length 128, and only the last 10% use sequences of length 512.

Two variants are considered:

- BERT *base* with 12 layers, 12 attention heads and hidden size 768 (110M parameters),
- BERT *large* with 24 layers, 16 attention heads and hidden size 1024 (340M parameters).

ReLU multiplies the input by zero or one, depending on its value.

Dropout stochastically multiplies the input by zero or one.

Both these functionalities are merged in Gaussian error linear units (GELUs), where the input value is multiplied by $m \sim \operatorname{Bernoulli}(\Phi(x))$, where $\Phi(x) = P(x' \leq x)$ for $x' \sim \mathcal{N}(0, 1)$ is the cumulative density function of the standard normal distribution.

The GELUs compute the expectation of this value, i.e.,

$$\operatorname{GELU}(x) = x \cdot \Phi(x) + 0 \cdot \big(1 - \Phi(x)\big) = x\Phi(x).$$

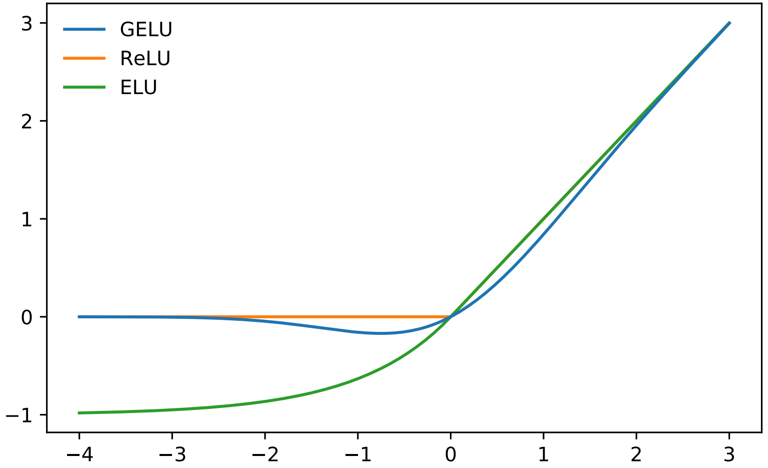



Figure 1: The GELU ($\mu = 0, \sigma = 1$), ReLU, and ELU ($\alpha = 1$).

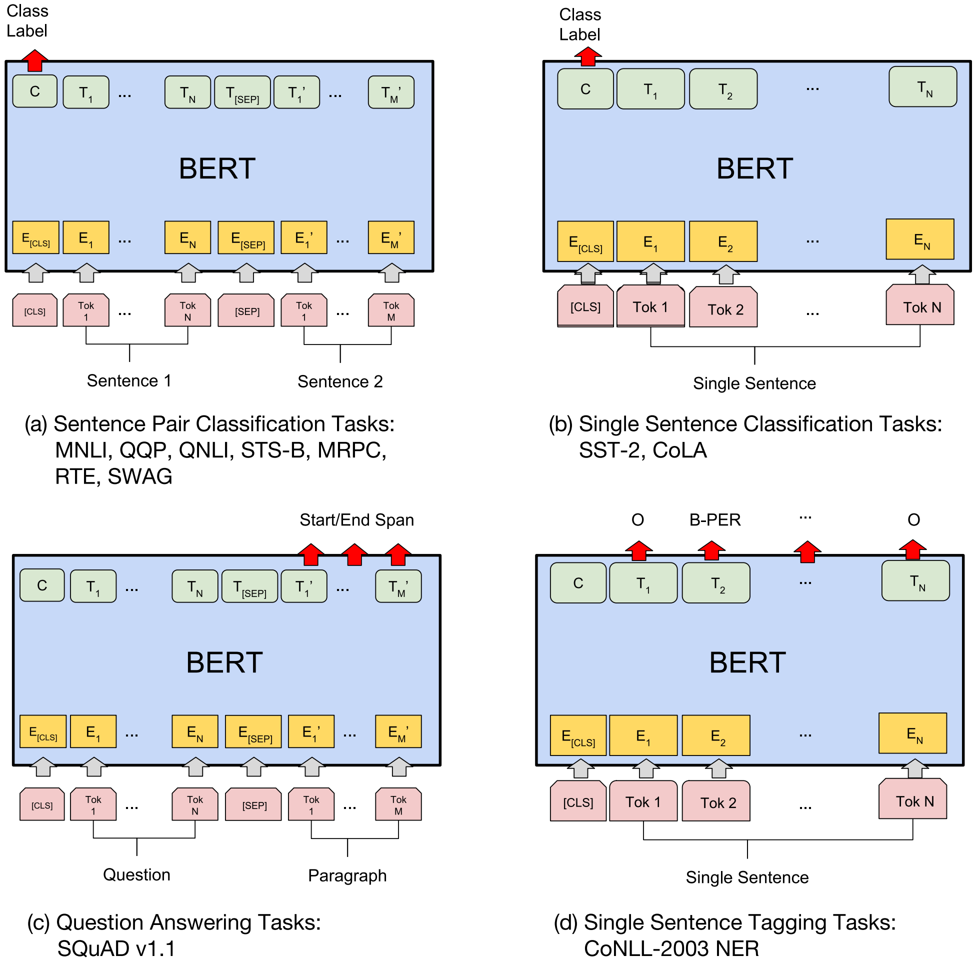
*Figure 1 of paper "Gaussian Error Linear Units (GELUs)", https://arxiv.org/abs/1606.08415*

GELUs can be approximated using

$$0.5x\left(1 + \tanh\left[\sqrt{2/\pi}(x + 0.044715x^3)\right]\right) \text{ or } x\sigma(1.702x).$$

The pre-trained BERT model can be finetuned on a range of tasks:

- **sentence element representation**
  - PoS tagging
  - named entity recognition
  - …
- **sentence representation**
  - text classification
- **sentence relation representation**
  - textual entailment, aka natural language inference (the second sentence is *implied by/contradicts/has no relation to* the first sentence)
  - textual similarity
  - paraphrase detection
  - natural language inference




(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

(b) Single Sentence Classification Tasks: SST-2, CoLA

(c) Question Answering Tasks: SQuAD v1.1

(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

*Figure 4 of paper "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", https://arxiv.org/abs/1810.04805*

# BERT – Results

For finetuning, dropout 0.1 is used, usually very small number of epochs (2-4) suffice, and a good learning rate is usually one of 5e-5, 3e-5, 2e-5.

| System | MNLI-(m/mm) | QQP | QNLI | SST-2 | CoLA | STS-B | MRPC | RTE | **Average** |
|---|---|---|---|---|---|---|---|---|---|
| | 392k | 363k | 108k | 67k | 8.5k | 5.7k | 3.5k | 2.5k | - |
| Pre-OpenAI SOTA | 80.6/80.1 | 66.1 | 82.3 | 93.2 | 35.0 | 81.0 | 86.0 | 61.7 | 74.0 |
| BiLSTM+ELMo+Attn | 76.4/76.1 | 64.8 | 79.8 | 90.4 | 36.0 | 73.3 | 84.9 | 56.8 | 71.0 |
| OpenAI GPT | 82.1/81.4 | 70.3 | 87.4 | 91.3 | 45.4 | 80.0 | 82.3 | 56.0 | 75.1 |
| $\text{BERT}_{\text{BASE}}$ | 84.6/83.4 | 71.2 | 90.5 | 93.5 | 52.1 | 85.8 | 88.9 | 66.4 | 79.6 |
| $\text{BERT}_{\text{LARGE}}$ | **86.7/85.9** | **72.1** | **92.7** | **94.9** | **60.5** | **86.5** | **89.3** | **70.1** | **82.1** |

Table 1: GLUE Test results, scored by the evaluation server (https://gluebenchmark.com/leaderboard). The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set.[8] BERT and OpenAI GPT are single-model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

*Table 1 of paper "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" , https://arxiv.org/abs/1810.04805*

| System | Dev EM | Dev F1 | Test EM | Test F1 |
|---|---|---|---|---|
| **Top Leaderboard Systems (Dec 10th, 2018)** | | | | |
| Human | - | - | 82.3 | 91.2 |
| #1 Ensemble - nlnet | - | - | 86.0 | 91.7 |
| #2 Ensemble - QANet | - | - | 84.5 | 90.5 |
| **Published** | | | | |
| BiDAF+ELMo (Single) | - | 85.6 | - | 85.8 |
| R.M. Reader (Ensemble) | 81.2 | 87.9 | 82.3 | 88.5 |
| **Ours** | | | | |
| $\text{BERT}_{\text{BASE}}$ (Single) | 80.8 | 88.5 | - | - |
| $\text{BERT}_{\text{LARGE}}$ (Single) | 84.1 | 90.9 | - | - |
| $\text{BERT}_{\text{LARGE}}$ (Ensemble) | 85.8 | 91.8 | - | - |
| $\text{BERT}_{\text{LARGE}}$ (Sgl.+TriviaQA) | **84.2** | **91.1** | **85.1** | **91.8** |
| $\text{BERT}_{\text{LARGE}}$ (Ens.+TriviaQA) | **86.2** | **92.2** | **87.4** | **93.2** |

Table 2: SQuAD 1.1 results. The BERT ensemble is 7x systems which use different pre-training checkpoints and fine-tuning seeds.

*Table 2 of paper "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", https://arxiv.org/abs/1810.04805*

| System | Dev EM | Dev F1 | Test EM | Test F1 |
|---|---|---|---|---|
| **Top Leaderboard Systems (Dec 10th, 2018)** | | | | |
| Human | 86.3 | 89.0 | 86.9 | 89.5 |
| #1 Single - MIR-MRC (F-Net) | - | - | 74.8 | 78.0 |
| #2 Single - nlnet | - | - | 74.2 | 77.1 |
| **Published** | | | | |
| unet (Ensemble) | - | - | 71.4 | 74.9 |
| SLQA+ (Single) | - | | 71.4 | 74.4 |
| **Ours** | | | | |
| $\text{BERT}_{\text{LARGE}}$ (Single) | 78.7 | 81.9 | 80.0 | 83.1 |

Table 3: SQuAD 2.0 results. We exclude entries that use BERT as one of their components.

*Table 3 of paper "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", https://arxiv.org/abs/1810.04805*

| System | Dev | Test |
|---|---|---|
| ESIM+GloVe | 51.9 | 52.7 |
| ESIM+ELMo | 59.1 | 59.2 |
| OpenAI GPT | - | 78.0 |
| $\text{BERT}_{\text{BASE}}$ | 81.6 | - |
| $\text{BERT}_{\text{LARGE}}$ | **86.6** | **86.3** |
| Human (expert)† | - | 85.0 |
| Human (5 annotations)† | - | 88.0 |

Table 4: SWAG Dev and Test accuracies. †Human performance is measured with 100 samples, as reported in the SWAG paper.

*Table 4 of paper "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", https://arxiv.org/abs/1810.04805*

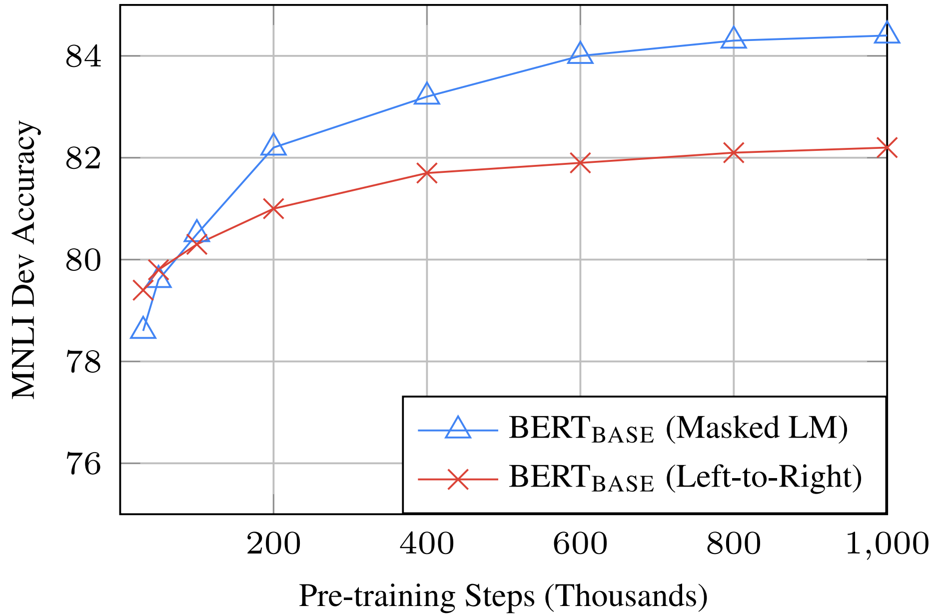



Figure 5: Ablation over number of training steps. This shows the MNLI accuracy after fine-tuning, starting from model parameters that have been pre-trained for $k$ steps. The x-axis is the value of $k$.

*Figure 5 of paper "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", https://arxiv.org/abs/1810.04805*

| Masking Rates | | | Dev Set Results | | |
|---|---|---|---|---|---|
| MASK | SAME | RND | MNLI Fine-tune | NER Fine-tune | NER Feature-based |
| 80% | 10% | 10% | 84.2 | 95.4 | 94.9 |
| 100% | 0% | 0% | 84.3 | 94.9 | 94.0 |
| 80% | 0% | 20% | 84.1 | 95.2 | 94.6 |
| 80% | 20% | 0% | 84.4 | 95.2 | 94.7 |
| 0% | 20% | 80% | 83.7 | 94.8 | 94.6 |
| 0% | 0% | 100% | 83.6 | 94.9 | 94.6 |

Table 8: Ablation over different masking strategies.

*Table 8 of paper "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", https://arxiv.org/abs/1810.04805*

# Multilingual BERT

The Multilingual BERT is pre-trained on 102-104 largest Wikipedias, including the Czech one.

There are two versions, the *cased* one has WordPieces including case, and the *uncased* one with subwords all in lower case and *without diacritics*.

Even if only very small percentage of input sentences were Czech, it works surprisingly well for Czech NLP.

Furthermore, without any explicit supervision, mBERT is able to represent the input languages in a *shared* space, allowing cross-lingual transfer.

# Cross-lingual Transfer with Multilingual BERT

Consider a *reading comprehension* task, where for a given paragraph and a question an answer needs to be located in the paragraph.

Then training the model in English and then directly running it on a different language works comparably to translating the data to English and then back.

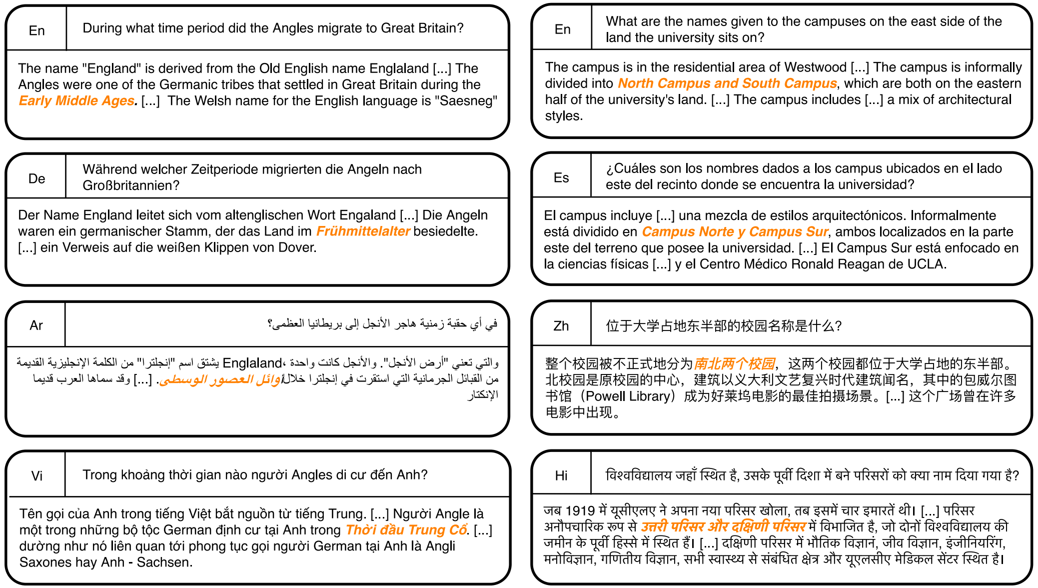



*Figure 2 of paper "MLQA: Evaluating Cross-lingual Extractive Question Answering", https://arxiv.org/abs/1910.07475*

| F1 / EM | en | es | de | ar | hi | vi | zh |
|---|---|---|---|---|---|---|---|
| BERT-Large | **80.2 / 67.4** | - | - | - | - | - | - |
| Multilingual-BERT | 77.7 / 65.2 | 64.3 / 46.6 | 57.9 / 44.3 | 45.7 / 29.8 | 43.8 / 29.7 | 57.1 / 38.6 | 57.5 / 37.3 |
| XLM | 74.9 / 62.4 | **68.0 / 49.8** | **62.2 / 47.6** | **54.8 / 36.3** | 48.8 / 27.3 | 61.4 / 41.8 | 61.1 / **39.6** |
| Translate test, BERT-L | - | 65.4 / 44.0 | 57.9 / 41.8 | 33.6 / 20.4 | 23.8 / 18.9* | 58.2 / 33.2 | 44.2 / 20.3 |
| Translate train, M-BERT | - | 53.9 / 37.4 | 62.0 / 47.5 | 51.8 / 33.2 | **55.0 / 40.0** | **62.0 / 43.1** | **61.4** / 39.5 |
| Translate train, XLM | - | 65.2 / 47.8 | 61.4 / 46.7 | 54.0 / 34.4 | 50.7 / 33.4 | 59.3 / 39.4 | 59.8 / 37.9 |

*Table 5 of paper "MLQA: Evaluating Cross-lingual Extractive Question Answering", https://arxiv.org/abs/1910.07475*

# RoBERTa – NSP

The *next sentence prediction* was originally hypothesized to be an important factor during training of the BERT model, as indicated by ablation experiments. However, later experiments indicated removing it might improve results.

The RoBERTa authors therefore performed the following experiments:

- SEGMENT-PAIR: pair of segments with at most 512 tokens in total;
- SENTENCE-PAIR: pair of *natural sentences*, usually significantly shorter than 512 tokens;
- FULL-SENTENCES: just one segment on input with 512 tokens, can cross document boundary;
- DOC-SENTENCES: just one segment on input with 512 tokens, cannot cross document boundary.

| **Model** | **SQuAD 1.1/2.0** | **MNLI-m** | **SST-2** | **RACE** |
|---|---|---|---|---|
| *Our reimplementation (with NSP loss):* | | | | |
| SEGMENT-PAIR | 90.4/78.7 | 84.0 | 92.9 | 64.2 |
| SENTENCE-PAIR | 88.7/76.2 | 82.9 | 92.1 | 63.0 |
| *Our reimplementation (without NSP loss):* | | | | |
| FULL-SENTENCES | 90.4/79.1 | 84.7 | 92.5 | 64.8 |
| DOC-SENTENCES | 90.6/79.7 | 84.7 | 92.7 | 65.6 |
| $\text{BERT}_{\text{BASE}}$ | 88.5/76.3 | 84.3 | 92.8 | 64.3 |
| $\text{XLNet}_{\text{BASE}}$ (K = 7) | –/81.3 | 85.8 | 92.7 | 66.1 |
| $\text{XLNet}_{\text{BASE}}$ (K = 6) | –/81.0 | 85.6 | 93.4 | 66.7 |

*Table 2 of paper "RoBERTa: A Robustly Optimized BERT Pretraining Approach", https://arxiv.org/abs/1907.11692*

# RoBERTa – Larger Batches

BERT is trained for 1M steps with a learning rate of 1e-4.

The RoBERTa authors also considered larger batches (with linearly larger learning rate).

| bsz | steps | lr | ppl | MNLI-m | SST-2 |
|---|---|---|---|---|---|
| 256 | 1M | 1e-4 | 3.99 | 84.7 | 92.7 |
| 2K | 125K | 7e-4 | **3.68** | **85.2** | **92.9** |
| 8K | 31K | 1e-3 | 3.77 | 84.6 | 92.8 |

Table 3: Perplexity on held-out training data (*ppl*) and development set accuracy for base models trained over BOOKCORPUS and WIKIPEDIA with varying batch sizes (*bsz*). We tune the learning rate (*lr*) for each setting. Models make the same number of passes over the data (epochs) and have the same computational cost.

*Table 3 of paper "RoBERTa: A Robustly Optimized BERT Pretraining Approach", https://arxiv.org/abs/1907.11692*

The RoBERTa model, **R**bustly **o**ptimized **BERT** **a**pproach, is trained with dynamic masking, FULL-SENTENCES without NSP, large 8k minibatches and byte-level BPE with 50k subwords.

| Model | data | bsz | steps | SQuAD (v1.1/2.0) | MNLI-m | SST-2 |
|---|---|---|---|---|---|---|
| RoBERTa | | | | | | |
| with BOOKS + WIKI | 16GB | 8K | 100K | 93.6/87.3 | 89.0 | 95.3 |
| + additional data (§3.2) | 160GB | 8K | 100K | 94.0/87.7 | 89.3 | 95.6 |
| + pretrain longer | 160GB | 8K | 300K | 94.4/88.7 | 90.0 | 96.1 |
| + pretrain even longer | 160GB | 8K | 500K | **94.6/89.4** | **90.2** | **96.4** |
| $\text{BERT}_{\text{LARGE}}$ | | | | | | |
| with BOOKS + WIKI | 13GB | 256 | 1M | 90.9/81.8 | 86.6 | 93.7 |
| $\text{XLNet}_{\text{LARGE}}$ | | | | | | |
| with BOOKS + WIKI | 13GB | 256 | 1M | 94.0/87.8 | 88.4 | 94.4 |
| + additional data | 126GB | 2K | 500K | 94.5/88.8 | 89.8 | 95.6 |

Table 4: Development set results for RoBERTa as we pretrain over more data (16GB $\rightarrow$ 160GB of text) and pretrain for longer (100K $\rightarrow$ 300K $\rightarrow$ 500K steps). Each row accumulates improvements from the rows above. RoBERTa matches the architecture and training objective of $\text{BERT}_{\text{LARGE}}$. Results for $\text{BERT}_{\text{LARGE}}$ and $\text{XLNet}_{\text{LARGE}}$ are from Devlin et al. (2019) and Yang et al. (2019), respectively. Complete results on all GLUE tasks can be found in the Appendix.

*Table 4 of paper "RoBERTa: A Robustly Optimized BERT Pretraining Approach", https://arxiv.org/abs/1907.11692*

# RoBERTa Results

| | MNLI | QNLI | QQP | RTE | SST | MRPC | CoLA | STS | WNLI | Avg |
|---|---|---|---|---|---|---|---|---|---|---|
| *Single-task single models on dev* | | | | | | | | | | |
| $\text{BERT}_{\text{LARGE}}$ | 86.6/- | 92.3 | 91.3 | 70.4 | 93.2 | 88.0 | 60.6 | 90.0 | - | - |
| $\text{XLNet}_{\text{LARGE}}$ | 89.8/- | 93.9 | 91.8 | 83.8 | 95.6 | 89.2 | 63.6 | 91.8 | - | - |
| RoBERTa | **90.2/90.2** | **94.7** | **92.2** | **86.6** | **96.4** | **90.9** | **68.0** | **92.4** | **91.3** | - |
| *Ensembles on test (from leaderboard as of July 25, 2019)* | | | | | | | | | | |
| ALICE | 88.2/87.9 | 95.7 | **90.7** | 83.5 | 95.2 | 92.6 | **68.6** | 91.1 | 80.8 | 86.3 |
| MT-DNN | 87.9/87.4 | 96.0 | 89.9 | 86.3 | 96.5 | 92.7 | 68.4 | 91.1 | 89.0 | 87.6 |
| XLNet | 90.2/89.8 | 98.6 | 90.3 | 86.3 | **96.8** | **93.0** | 67.8 | 91.6 | **90.4** | 88.4 |
| RoBERTa | **90.8/90.2** | **98.9** | 90.2 | **88.2** | 96.7 | 92.3 | 67.8 | **92.2** | 89.0 | **88.5** |

Table 5: Results on GLUE. All results are based on a 24-layer architecture. $\text{BERT}_{\text{LARGE}}$ and $\text{XLNet}_{\text{LARGE}}$ results are from Devlin et al. (2019) and Yang et al. (2019), respectively. RoBERTa results on the development set are a median over five runs. RoBERTa results on the test set are ensembles of *single-task* models. For RTE, STS and MRPC we finetune starting from the MNLI model instead of the baseline pretrained model. Averages are obtained from the GLUE leaderboard.

*Table 5 of paper "RoBERTa: A Robustly Optimized BERT Pretraining Approach", https://arxiv.org/abs/1907.11692*

| Model | SQuAD 1.1 EM | SQuAD 1.1 F1 | SQuAD 2.0 EM | SQuAD 2.0 F1 |
|---|---|---|---|---|
| *Single models on dev, w/o data augmentation* | | | | |
| $\text{BERT}_{\text{LARGE}}$ | 84.1 | 90.9 | 79.0 | 81.8 |
| $\text{XLNet}_{\text{LARGE}}$ | **89.0** | 94.5 | 86.1 | 88.8 |
| RoBERTa | 88.9 | **94.6** | **86.5** | **89.4** |
| *Single models on test (as of July 25, 2019)* | | | | |
| $\text{XLNet}_{\text{LARGE}}$ | | | $86.3^{\dagger}$ | $89.1^{\dagger}$ |
| RoBERTa | | | 86.8 | 89.8 |
| XLNet + SG-Net Verifier | | | **$87.0^{\dagger}$** | **$89.9^{\dagger}$** |

*Table 6 of paper "RoBERTa: A Robustly Optimized BERT Pretraining Approach", https://arxiv.org/abs/1907.11692*

| Model | Accuracy | Middle | High |
|---|---|---|---|
| *Single models on test (as of July 25, 2019)* | | | |
| $\text{BERT}_{\text{LARGE}}$ | 72.0 | 76.6 | 70.1 |
| $\text{XLNet}_{\text{LARGE}}$ | 81.7 | 85.4 | 80.2 |
| RoBERTa | **83.2** | **86.5** | **81.3** |

Table 7: Results on the RACE test set. $\text{BERT}_{\text{LARGE}}$ and $\text{XLNet}_{\text{LARGE}}$ results are from Yang et al. (2019).

*Table 7 of paper "RoBERTa: A Robustly Optimized BERT Pretraining Approach", https://arxiv.org/abs/1907.11692*

ALBERT model, **A Lite BERT**, was proposed with small model size in mind.

To achieve smaller size, the authors consider

- factorized embeddings representation; and
- parameter sharing across layers.

The following configurations are evaluated in the paper:

| Model | | Parameters | Layers | Hidden | Embedding | Parameter-sharing |
|---|---|---|---|---|---|---|
| BERT | base | 108M | 12 | 768 | 768 | False |
| | large | 334M | 24 | 1024 | 1024 | False |
| ALBERT | base | 12M | 12 | 768 | 128 | True |
| | large | 18M | 24 | 1024 | 128 | True |
| | xlarge | 60M | 24 | 2048 | 128 | True |
| | xxlarge | 235M | 12 | 4096 | 128 | True |

Table 1: The configurations of the main BERT and ALBERT models analyzed in this paper.

*Table 1 of paper "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations", https://arxiv.org/abs/1909.11942*

# ALBERT – Factorized Embeddings

The subword embeddings have the hidden size dimensionality $H$ in BERT, which results in quite a large number of parameters.

Instead, authors propose to represent the subwords using only embeddings of size $E$ and then use a matrix of size $E \times H$ to generate the correctly-sized embeddings for the first layer.

| Model | $E$ | Parameters | SQuAD1.1 | SQuAD2.0 | MNLI | SST-2 | RACE | Avg |
|---|---|---|---|---|---|---|---|---|
| ALBERT base not-shared | 64 | 87M | 89.9/82.9 | 80.1/77.8 | 82.9 | 91.5 | 66.7 | 81.3 |
| | 128 | 89M | 89.9/82.8 | 80.3/77.3 | 83.7 | 91.5 | 67.9 | 81.7 |
| | 256 | 93M | 90.2/83.2 | 80.3/77.4 | 84.1 | 91.9 | 67.3 | 81.8 |
| | 768 | 108M | 90.4/83.2 | 80.4/77.6 | 84.5 | 92.8 | 68.2 | 82.3 |
| ALBERT base all-shared | 64 | 10M | 88.7/81.4 | 77.5/74.8 | 80.8 | 89.4 | 63.5 | 79.0 |
| | 128 | 12M | 89.3/82.3 | 80.0/77.1 | 81.6 | 90.3 | 64.0 | 80.1 |
| | 256 | 16M | 88.8/81.5 | 79.1/76.3 | 81.5 | 90.3 | 63.4 | 79.6 |
| | 768 | 31M | 88.6/81.5 | 79.2/76.6 | 82.0 | 90.6 | 63.3 | 79.8 |

Table 3: The effect of vocabulary embedding size on the performance of ALBERT-base.

*Table 3 of paper "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations", https://arxiv.org/abs/1909.11942*

# ALBERT – Shared Parameters

To improve parameter efficiency, the parameters of both the soft-attention and the feed-forward networks are shared across layers.

| Model | | Parameters | SQuAD1.1 | SQuAD2.0 | MNLI | SST-2 | RACE | Avg |
|---|---|---|---|---|---|---|---|---|
| ALBERT base $E$=768 | all-shared | 31M | 88.6/81.5 | 79.2/76.6 | 82.0 | 90.6 | 63.3 | 79.8 |
| | shared-attention | 83M | 89.9/82.7 | 80.0/77.2 | 84.0 | 91.4 | 67.7 | 81.6 |
| | shared-FFN | 57M | 89.2/82.1 | 78.2/75.4 | 81.5 | 90.8 | 62.6 | 79.5 |
| | not-shared | 108M | 90.4/83.2 | 80.4/77.6 | 84.5 | 92.8 | 68.2 | 82.3 |
| ALBERT base $E$=128 | all-shared | 12M | 89.3/82.3 | 80.0/77.1 | 82.0 | 90.3 | 64.0 | 80.1 |
| | shared-attention | 64M | 89.9/82.8 | 80.7/77.9 | 83.4 | 91.9 | 67.6 | 81.7 |
| | shared-FFN | 38M | 88.9/81.6 | 78.6/75.6 | 82.3 | 91.7 | 64.4 | 80.2 |
| | not-shared | 89M | 89.9/82.8 | 80.3/77.3 | 83.2 | 91.5 | 67.9 | 81.6 |

Table 4: The effect of cross-layer parameter-sharing strategies, ALBERT-base configuration.

*Table 4 of paper "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations", https://arxiv.org/abs/1909.11942*

# ALBERT – Sentence Order Prediction

An alternative to *next sentence prediction* is considered – given two consecutive segments, predict which one appeared first in the original document.

| | Intrinsic Tasks | | | Downstream Tasks | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| SP tasks | MLM | NSP | SOP | SQuAD1.1 | SQuAD2.0 | MNLI | SST-2 | RACE | Avg |
| None | 54.9 | 52.4 | 53.3 | 88.6/81.5 | 78.1/75.3 | 81.5 | 89.9 | 61.7 | 79.0 |
| NSP | 54.5 | 90.5 | 52.0 | 88.4/81.5 | 77.2/74.6 | 81.6 | **91.1** | 62.3 | 79.2 |
| SOP | 54.0 | 78.9 | 86.5 | **89.3/82.3** | **80.0/77.1** | **82.0** | 90.3 | **64.0** | **80.1** |

Table 5: The effect of sentence-prediction loss, NSP vs. SOP, on intrinsic and downstream tasks.

*Table 5 of paper "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations", https://arxiv.org/abs/1909.11942*

# ALBERT Results and Ablations

| Model | | Parameters | SQuAD1.1 | SQuAD2.0 | MNLI | SST-2 | RACE | Avg | Speedup |
|---|---|---|---|---|---|---|---|---|---|
| BERT | base | 108M | 90.4/83.2 | 80.4/77.6 | 84.5 | 92.8 | 68.2 | 82.3 | 4.7x |
| | large | 334M | 92.2/85.5 | 85.0/82.2 | 86.6 | 93.0 | 73.9 | 85.2 | 1.0 |
| ALBERT | base | 12M | 89.3/82.3 | 80.0/77.1 | 81.6 | 90.3 | 64.0 | 80.1 | 5.6x |
| | large | 18M | 90.6/83.9 | 82.3/79.4 | 83.5 | 91.7 | 68.5 | 82.4 | 1.7x |
| | xlarge | 60M | 92.5/86.1 | 86.1/83.1 | 86.4 | 92.4 | 74.8 | 85.5 | 0.6x |
| | xxlarge | 235M | **94.1/88.3** | **88.1/85.1** | **88.0** | **95.2** | **82.3** | **88.7** | 0.3x |

*Table 2 of paper "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations", https://arxiv.org/abs/1909.11942*

| | SQuAD1.1 | SQuAD2.0 | MNLI | SST-2 | RACE | Avg |
|---|---|---|---|---|---|---|
| No additional data | **89.3/82.3** | **80.0/77.1** | 81.6 | 90.3 | 64.0 | 80.1 |
| With additional data | 88.8/81.7 | 79.1/76.3 | **82.4** | **92.8** | **66.0** | **80.8** |

Table 7: The effect of additional training data using the ALBERT-base configuration.

*Table 7 of paper "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations", https://arxiv.org/abs/1909.11942*

| | SQuAD1.1 | SQuAD2.0 | MNLI | SST-2 | RACE | Avg |
|---|---|---|---|---|---|---|
| With dropout | 94.7/89.2 | 89.6/86.9 | 90.0 | 96.3 | 85.7 | 90.4 |
| Without dropout | **94.8/89.5** | **89.9/87.2** | **90.4** | **96.5** | **86.1** | **90.7** |

*Table 8 of paper "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations", https://arxiv.org/abs/1909.11942*

# ALBERT SoTA Results

| Models | MNLI | QNLI | QQP | RTE | SST | MRPC | CoLA | STS | WNLI | Avg |
|---|---|---|---|---|---|---|---|---|---|---|
| *Single-task single models on dev* | | | | | | | | | | |
| BERT-large | 86.6 | 92.3 | 91.3 | 70.4 | 93.2 | 88.0 | 60.6 | 90.0 | - | - |
| XLNet-large | 89.8 | 93.9 | 91.8 | 83.8 | 95.6 | 89.2 | 63.6 | 91.8 | - | - |
| RoBERTa-large | 90.2 | 94.7 | **92.2** | 86.6 | 96.4 | **90.9** | 68.0 | 92.4 | - | - |
| ALBERT (1M) | 90.4 | 95.2 | 92.0 | 88.1 | 96.8 | 90.2 | 68.7 | 92.7 | - | - |
| ALBERT (1.5M) | **90.8** | **95.3** | **92.2** | **89.2** | **96.9** | **90.9** | **71.4** | **93.0** | - | - |
| *Ensembles on test (from leaderboard as of Sept. 16, 2019)* | | | | | | | | | | |
| ALICE | 88.2 | 95.7 | **90.7** | 83.5 | 95.2 | 92.6 | **69.2** | 91.1 | 80.8 | 87.0 |
| MT-DNN | 87.9 | 96.0 | 89.9 | 86.3 | 96.5 | 92.7 | 68.4 | 91.1 | 89.0 | 87.6 |
| XLNet | 90.2 | 98.6 | 90.3 | 86.3 | 96.8 | 93.0 | 67.8 | 91.6 | 90.4 | 88.4 |
| RoBERTa | 90.8 | 98.9 | 90.2 | 88.2 | 96.7 | 92.3 | 67.8 | 92.2 | 89.0 | 88.5 |
| Adv-RoBERTa | 91.1 | 98.8 | 90.3 | 88.7 | 96.8 | 93.1 | 68.0 | 92.4 | 89.0 | 88.8 |
| ALBERT | **91.3** | **99.2** | 90.5 | **89.2** | **97.1** | **93.4** | 69.1 | **92.5** | **91.8** | **89.4** |

Table 9: State-of-the-art results on the GLUE benchmark. For single-task single-model results, we report ALBERT at 1M steps (comparable to RoBERTa) and at 1.5M steps. The ALBERT ensemble uses models trained with 1M, 1.5M, and other numbers of steps.

*Table 9 of paper "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations", https://arxiv.org/abs/1909.11942*

# ALBERT SoTA Results

| Models | SQuAD1.1 dev | SQuAD2.0 dev | SQuAD2.0 test | RACE test (Middle/High) |
|---|---|---|---|---|
| *Single model (from leaderboard as of Sept. 23, 2019)* | | | | |
| BERT-large | 90.9/84.1 | 81.8/79.0 | 89.1/86.3 | 72.0 (76.6/70.1) |
| XLNet | 94.5/89.0 | 88.8/86.1 | 89.1/86.3 | 81.8 (85.5/80.2) |
| RoBERTa | 94.6/88.9 | 89.4/86.5 | 89.8/86.8 | 83.2 (86.5/81.3) |
| UPM | - | - | 89.9/87.2 | - |
| XLNet + SG-Net Verifier++ | - | - | 90.1/87.2 | - |
| ALBERT (1M) | 94.8/89.2 | 89.9/87.2 | - | 86.0 (88.2/85.1) |
| ALBERT (1.5M) | **94.8/89.3** | **90.2/87.4** | **90.9/88.1** | **86.5 (89.0/85.5)** |
| *Ensembles (from leaderboard as of Sept. 23, 2019)* | | | | |
| BERT-large | 92.2/86.2 | - | - | - |
| XLNet + SG-Net Verifier | - | - | 90.7/88.2 | - |
| UPM | - | - | 90.7/88.2 | |
| XLNet + DAAF + Verifier | - | - | 90.9/88.6 | - |
| DCMN+ | - | - | - | 84.1 (88.5/82.3) |
| ALBERT | **95.5/90.1** | **91.4/88.9** | **92.2/89.7** | **89.4 (91.2/88.6)** |

Table 10: State-of-the-art results on the SQuAD and RACE benchmarks.

*Table 10 of paper "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations", https://arxiv.org/abs/1909.11942*