Deep Learning
Summer semester ‘24

UNI WÜ CAIDAS

7. Transformers

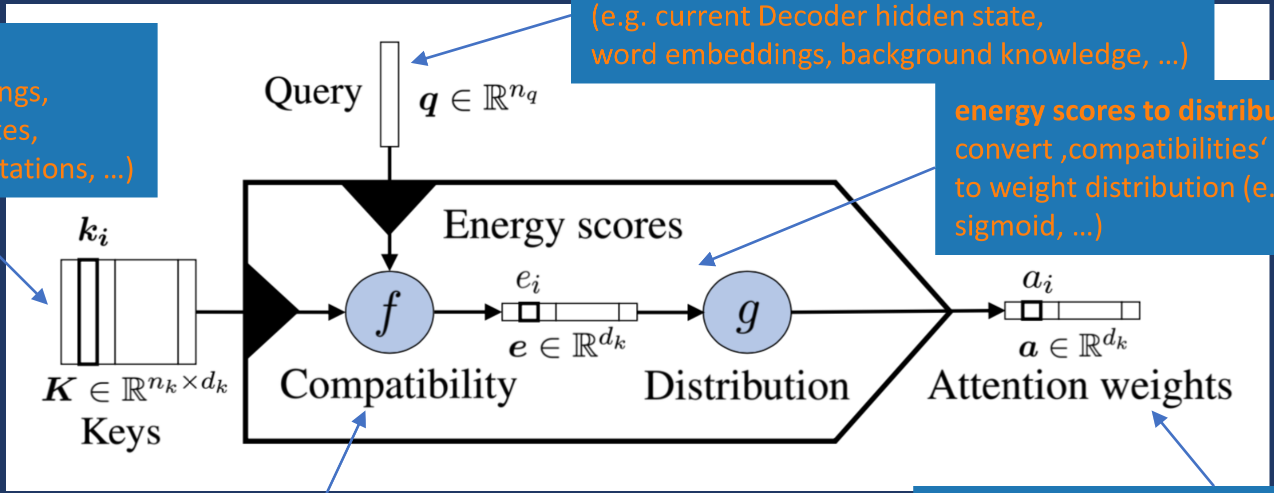
Attention, please!

- In the last few years, many attention mechanisms were introduced
- Always same idea: Compute attention weights for the input sequence to focus on more relevant input steps

query: what to ask for?
(e.g. current Decoder hidden state,
word embeddings, background knowledge, …)

keys: data features
(e.g. word embeddings,
Encoder hidden states,
document representations, …)

energy scores to distribution:
convert ‚compatibilities' of query and keys to weight distribution (e.g. using softmax, sigmoid, …)

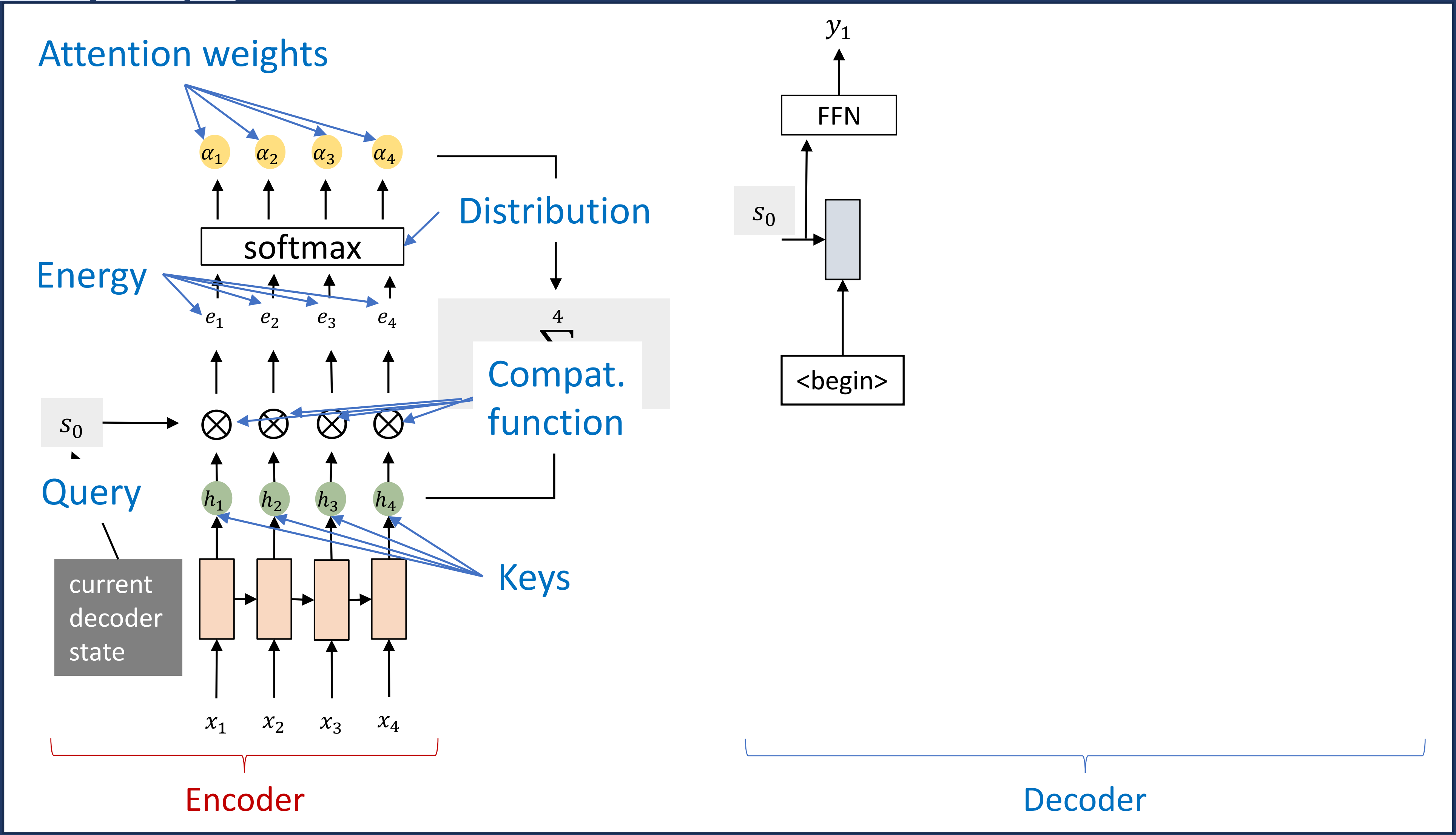


compatibility function: which key is most compatible to query?

attention weights: used to weight the keys

Galassi, Andrea, Marco Lippi, and Paolo Torroni.
"Attention, please! A Critical Review of Neural Attention Models in Natural Language Processing." *arXiv preprint arXiv:1902.02181* (2019).

Recall: Loung-Attention



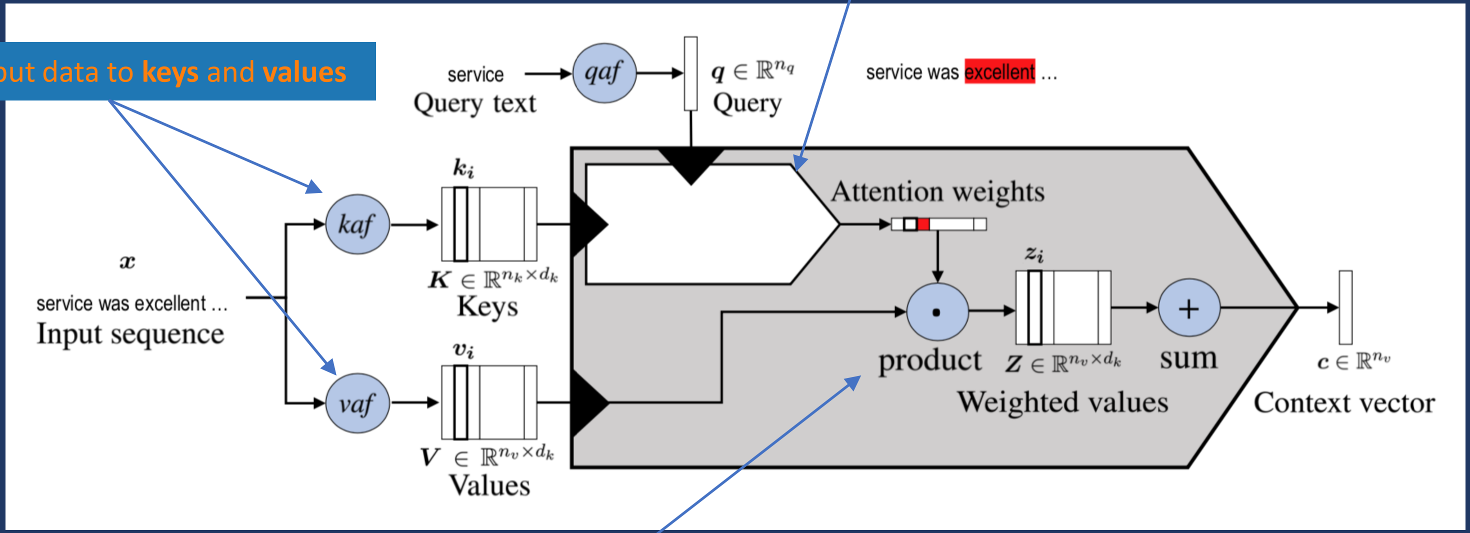
Luong, M. T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation.

Attention, please!

- Sometimes, new key representations are useful → introducing **values**

same mechanism as on previous slide

convert input data to **keys and values**

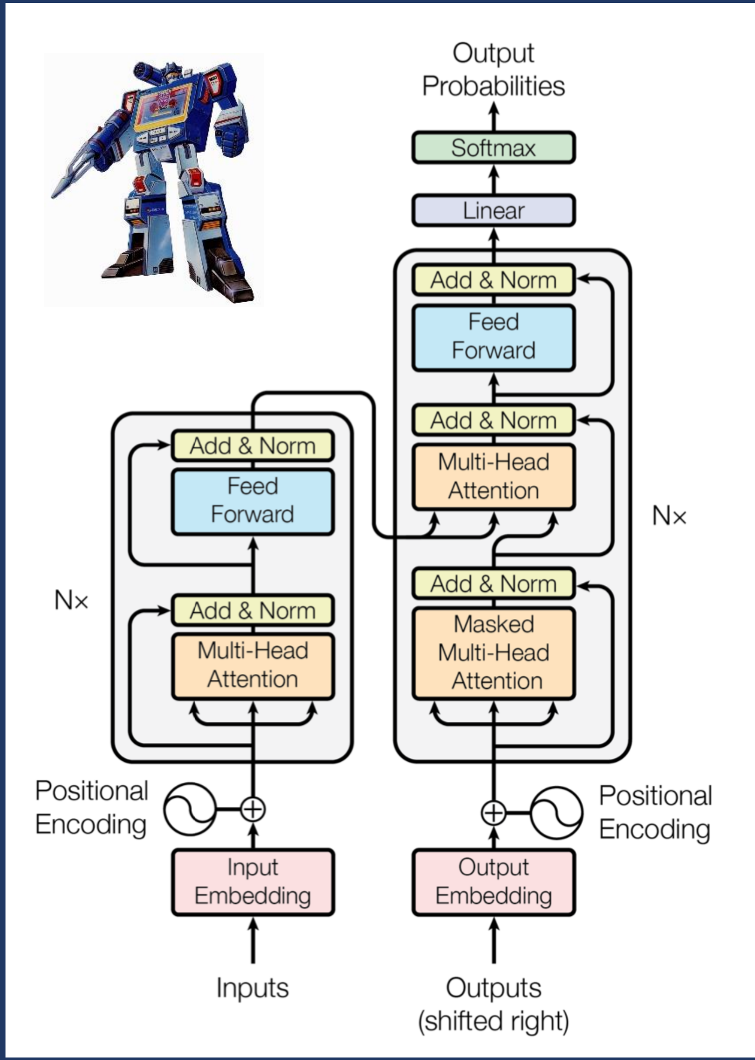


Not the keys are weighted with the resulting distribution, but the values

Galassi, Andrea, Marco Lippi, and Paolo Torroni.
"Attention, please! A Critical Review of Neural Attention Models in Natural Language Processing." *arXiv preprint arXiv:1902.02181* (2019).

Transformer — Is Attention All You Need?

- Transformer: A new neural network architecture based on attention
- Encoder-Decoder structure
- No recurrence!
  - Parallelizable → faster to train
- The encoded sentence is as long as the input sentence!
  - Capturing more information of input
  - „Transforms“ the input into an encoded form



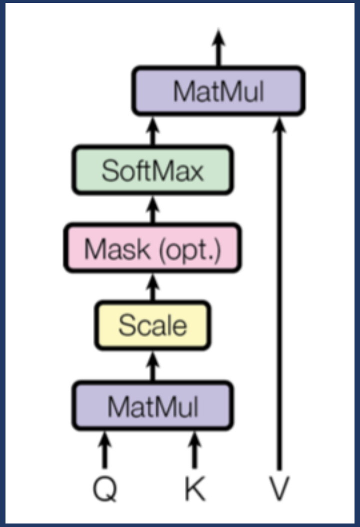
Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.

Transformer Key Idea — (Multi-Head Self-)Attention

Scaled Dot-Product Attention:

- Introduced in Vaswani et al., 2017
- Represents attention by matrix multiplication
- Uses a scaling factor d → Empirically improves results

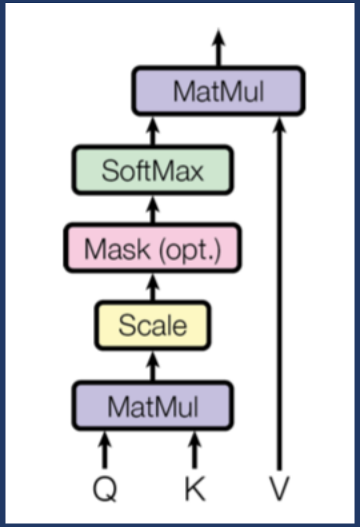
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$



Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.

Transformer Key Idea — (Multi-Head) Self-Attention

- **Self-Attention**
  - Transform an input sequence to a weighted sum of its **own** timesteps
  - Helps to capture long-term dependencies
  - Use Scaled Dot-Product Attention (prev. slide)
  - **Q**uery, **K**eys, and **V**alues are all computed from the input sequence
  - Difference from before:
    Query came from ‚outside' (e.g. Decoder hidden state)



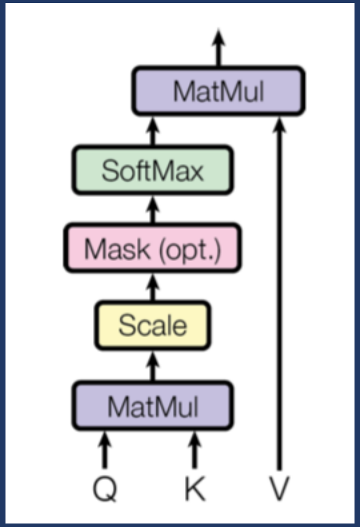
Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.

Transformer Key Idea — (Multi-Head) Self-Attention

- **Self-Attention**
  - Input X
  - Transform X into three different „views“:
    - $K = X \cdot W_k$
    - $V = X \cdot W_v$
    - $Q = X \cdot W_q$

Trainable weights

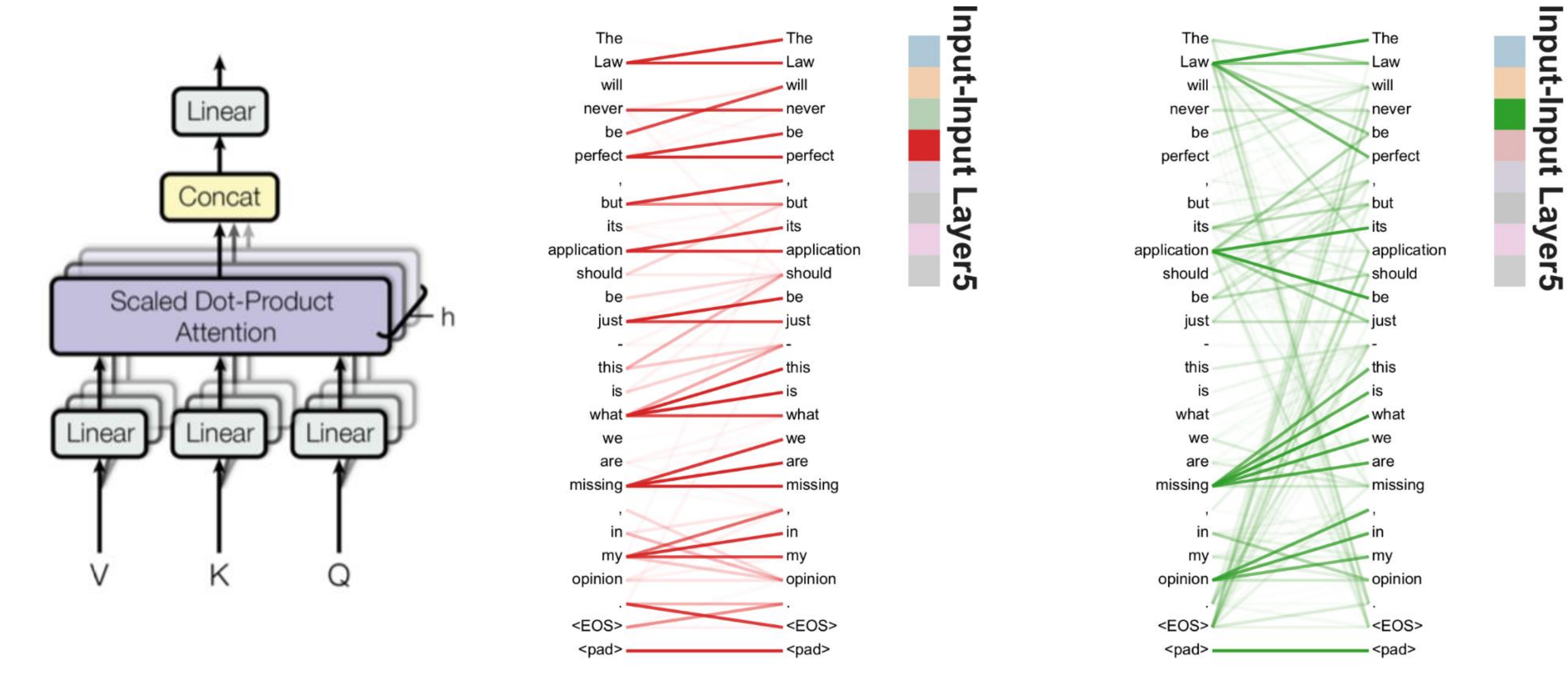
  - $Attention(Q, K, V)$ as before



Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.

Transformer Key Idea — Multi-Head Self-Attention

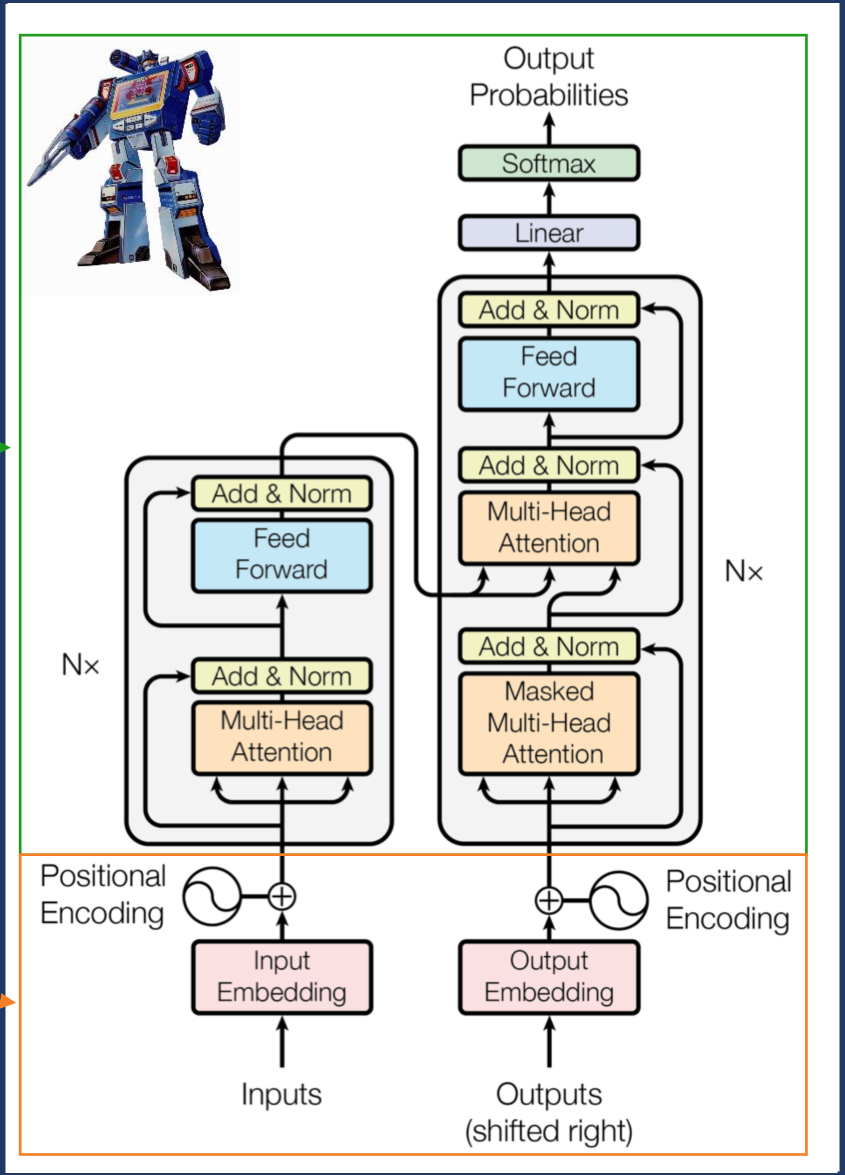
- Multi-Head Attention:
  - Apply self-attention multiple times for the same input sequence (using different weights W_q^i, W_v^i and W_k^i)
  - → Attention with multiple „views“ of the original sequence
  - → Enables capturing different kinds of importance



Transformer — Is Attention All You Need?

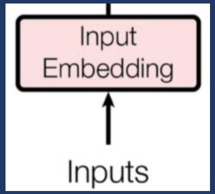
You know this part

We haven‘t talked about these



Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.

Byte Pair Encoding — From Words to Subwords



- A vocabulary of 50,000 words covers ~95% of the text …
- … this gets you 95% of the way

- Imagine a translation task:
  - “The sewage treatment plant smells particularly special today”
  - “Die Abwasser Behandlungs Anlage riecht heute besonders speziell”

- “Die **UNKNOWN** riecht heute besonders speziell”

Abwasserbehandlungsanlage?

Byte Pair Encoding — From Words to Subwords

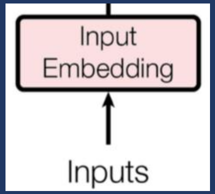
- Traditional NMT has a fixed vocabulary of 30,000 — 50,000 words
  - Rare words are problematic
  - Out-of-vocabulary words even more so
- NMT is an open-vocabulary problem
  - Especially for languages with productive word formation (compounding)
  - E. g. German

→Let’s go a level deeper and use sub-word tokens

- Character-level tokens seem computationally infeasible
- Can we do better than that?

→As so often, information theory comes to rescue

Byte Pair Encoding — From Words to Subwords



- Byte Pair Encoding
  - Starting Point: Character-level representation
  - Repeatedly replace most frequent symbol pair (*a, b*) with (*ab*)
  - Hyperparameter **m**: When to stop → Vocabulary Size
- Bottom-up character merging
- Example with 10 merges (**m** = original vocab. + 10):

1

Word	Frequency
l o w </w>	5
l o w e r </w>	2
n e w e s t </w>	6
w i d e s t </w>	3

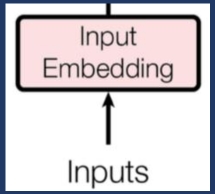
End-of-word symbol to restore original tokenization after translation

Vocabulary: l o w </w> e r n s t i d

Pairs		Frequency
l	o	7
o	w	7
…	…	…
e	**s**	**9**
…	…	…
t	</w>	9

Merge **e** and **s**

Byte Pair Encoding — From Words to Subwords



- Byte Pair Encoding
  - Starting Point: Character-level representation
  - Repeatedly replace most frequent symbol pair (*a, b*) with (*ab*)
  - Hyperparameter **m**: When to stop → Vocabulary Size
- Bottom-up character merging
- Example with 10 merges (**m** = original vocab. + 10):

2

Word	Frequency
l o w </w>	5
l o w e r </w>	2
n e w **es** t </w>	6
w i d **es** t </w>	3

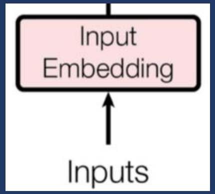
End-of-word symbol to restore original tokenization after translation

Vocabulary: l o w </w> e r n s t i d **es**

Pairs		**Frequency**
l	o	7
o	w	7
…	…	…
es	**t**	**9**
…	…	…
t	</w>	9

➡ Merge **es** and **t**

Byte Pair Encoding — From Words to Subwords



- Byte Pair Encoding
  - Starting Point: Character-level representation
  - Repeatedly replace most frequent symbol pair (*a, b*) with (*ab*)
  - Hyperparameter **m**: When to stop → Vocabulary Size
- Bottom-up character merging
- Example with 10 merges (**m** = original vocab. + 10):

3

Word	Frequency
l o w </w>	5
l o w e r </w>	2
n e w **est** </w>	6
w i d **est** </w>	3

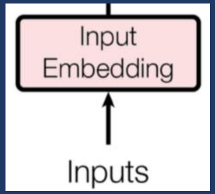
End-of-word symbol to restore original tokenization after translation

Vocabulary: l o w </w> e r n s t i d es **est**

Pairs		**Frequency**
l	o	7
o	w	7
…	…	…
est	**</w>**	**9**
…	…	…
d	est	3

Merge **est** and **</w>**

Byte Pair Encoding — From Words to Subwords



- Byte Pair Encoding
  - Starting Point: Character-level representation
  - Repeatedly replace most frequent symbol pair (*a, b*) with (*ab*)
  - Hyperparameter **m**: When to stop → Vocabulary Size
- Bottom-up character merging
- Example with 10 merges (**m** = original vocab. + 10):

4

Word	Frequency
l o w </w>	5
l o w e r </w>	2
n e w **est</w>**	6
w i d **est</w>**	3

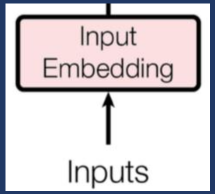
End-of-word symbol to restore original tokenization after translation

Vocabulary: l o w </w> e r n s t i d es est **…**

Pairs		**Frequency**
l	**o**	**7**
o	w	7
…	…	…
w	est</w>	6
…	…	…
d	es	3

Merge **l** and **o**

Byte Pair Encoding — From Words to Subwords



- Byte Pair Encoding
  - Starting Point: Character-level representation
  - Repeatedly replace most frequent symbol pair (*a, b*) with (*ab*)
  - Hyperparameter **m**: When to stop → Vocabulary Size
- Bottom-up character merging
- Example with 10 merges (**m** = original vocab. + 10):

10

Word	Frequency
low</w>	5
low e r </w>	2
newest</w>	6
w i d est</w>	3

End-of-word symbol to restore original tokenization after translation

Vocabulary: l o w </w> e r n s t i d es est est</w> lo low ne new newest</w> low</w> wi

Size: Equal to initial vocabulary + amount merges

Byte Pair Encoding — From Words to Subwords

- How does Tokenization work?
  - Let‘s look at „Abwasserbehandlungsanlage“ again
  - Imagine we learned these merges, best at top to worst at bottom

A b
a s
e r
s er
w as
Ab was
Abwas ser
B e
a n
d l
h an
n g
u ng
Be han
dl ung
Behan dlung
A n
a g
l ag

Byte Pair Encoding — From Words to Subwords

- How does Tokenization work?
  - Let‘s look at „Abwasserbehandlungsanlage“ again
  - Imagine we learned these merges, best at top to worst at bottom

A b
a s
e r
s er
w as
Ab was
Abwas ser
B e
a n
d l
h an
n g
u ng
Be han
dl ung
Behan dlung
A n
a g
l ag

1. Split word into characters

A b w a s s e r b e h a n d l u n g s a n l a g e </w>

Byte Pair Encoding — From Words to Subwords

- How does Tokenization work?
  - Let‘s look at „Abwasserbehandlungsanlage“ again
  - Imagine we learned these merges, best at top to worst at bottom

A b
a s
e r
s er
w as
Ab was
Abwas ser
B e
a n
d l
h an
n g
u ng
Be han
dl ung
Behan dlung
A n
a g
l ag

1. Split word into characters

   A b w a s s e r b e h a n d l u n g s a n l a g e </w>

2. Repeatedly pick best merge

   Ab w a s s e r b e h a n d l u n g s a n l a g e </w>

Byte Pair Encoding — From Words to Subwords

- How does Tokenization work?
  - Let‘s look at „Abwasserbehandlungsanlage“ again
  - Imagine we learned these merges, best at top to worst at bottom

A b
a s
e r
s er
w as
Ab was
Abwas ser
B e
a n
d l
h an
n g
u ng
Be han
dl ung
Behan dlung
A n
a g
l ag

1. Split word into characters

   A b w a s s e r b e h a n d l u n g s a n l a g e </w>

2. Repeatedly pick best merge

   Ab w **as** s e r b e h a n d l u n g s a n l a g e </w>

Byte Pair Encoding — From Words to Subwords

- How does Tokenization work?
  - Let‘s look at „Abwasserbehandlungsanlage“ again
  - Imagine we learned these merges, best at top to worst at bottom

A b
a s
e r
s er
w as
Ab was
Abwas ser
B e
a n
d l
h an
n g
u ng
Be han
dl ung
Behan dlung
A n
a g
l ag

1. Split word into characters

   A b w a s s e r b e h a n d l u n g s a n l a g e </w>

2. Repeatedly pick best merge

   Ab w as s **er** b e h a n d l u n g s a n l a g e </w>

Byte Pair Encoding — From Words to Subwords

- How does Tokenization work?
  - Let‘s look at „Abwasserbehandlungsanlage“ again
  - Imagine we learned these merges, best at top to worst at bottom

A b
a s
e r
s er
w as
Ab was
Abwas ser
B e
a n
d l
h an
n g
u ng
Be han
dl ung
Behan dlung
A n
a g
l ag

1. Split word into characters

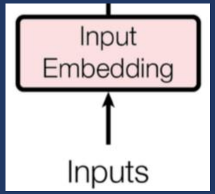
   A b w a s s e r b e h a n d l u n g s a n l a g e </w>

2. Repeatedly pick best merge

   Abwasser b e han dlung s an lag e </w>

3. We now represent our unknown word with ten subtokens

Byte Pair Encoding — From Words to Subwords



- Why Byte Pair Encoding?

- Open Vocabulary
  - Operations learned on training set can be applied to unknown words
- Compression of frequent character sequences (efficiency)

→Trade-off between text length and vocabulary size

Rico Sennrich, Barry Haddow, and Alexandra Birch. (2015). Neural Machine Translation of Rare Words with Subword Units.

Positional Encoding — A Notion of Order

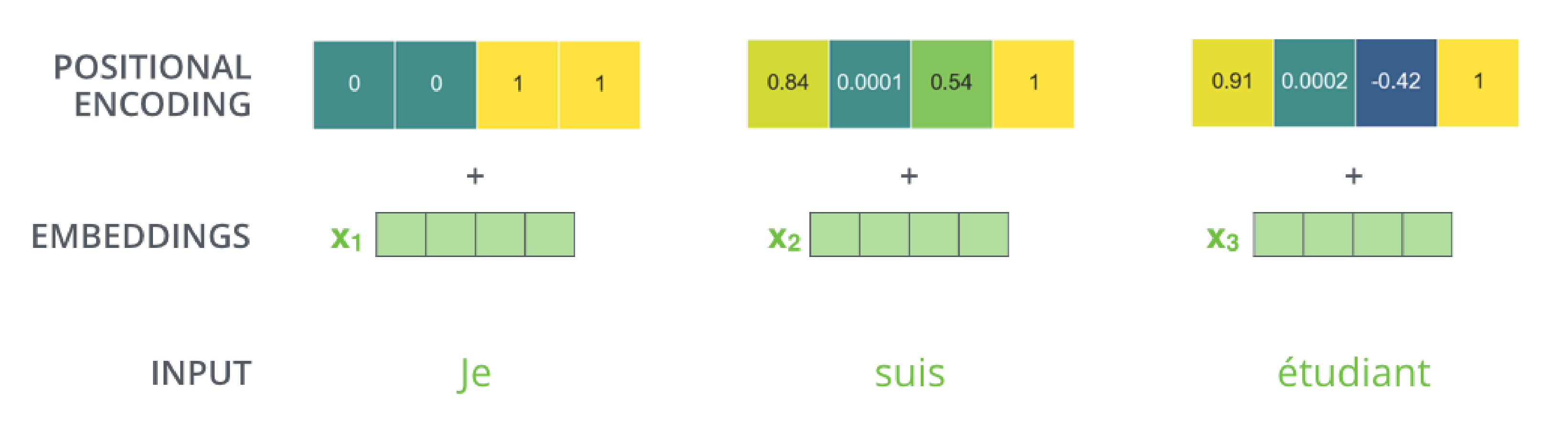
- Position and order of words are essential in any language
- RNNs model these inherently
- Transformers (intentionally) don't have recurrence
  - Massive improvements in speed
  - Potentially longer dependencies are covered
  - But: Inputs loses sequence information

Tip: The image is quite telling

- How can structure be preserved alternatively?
  - Unique encoding for each position in a sentence
  - Distances between positions must be consistent across different length sentences
  - Generalization to longer sentences

Positional Encoding — A Notion of Order

- Idea: Encode this information into our embeddings
  - Add a signal to each embedding that allows meaningful distances between vectors
  - The model learns this pattern



https://jalammar.github.io/illustrated-transformer/

Positional Encoding — A Notion of Order

- Vaswani et al. use sines and cosines of different frequencies
  - There are multiple other options, even learned ones, e. g. Shaw et al.

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

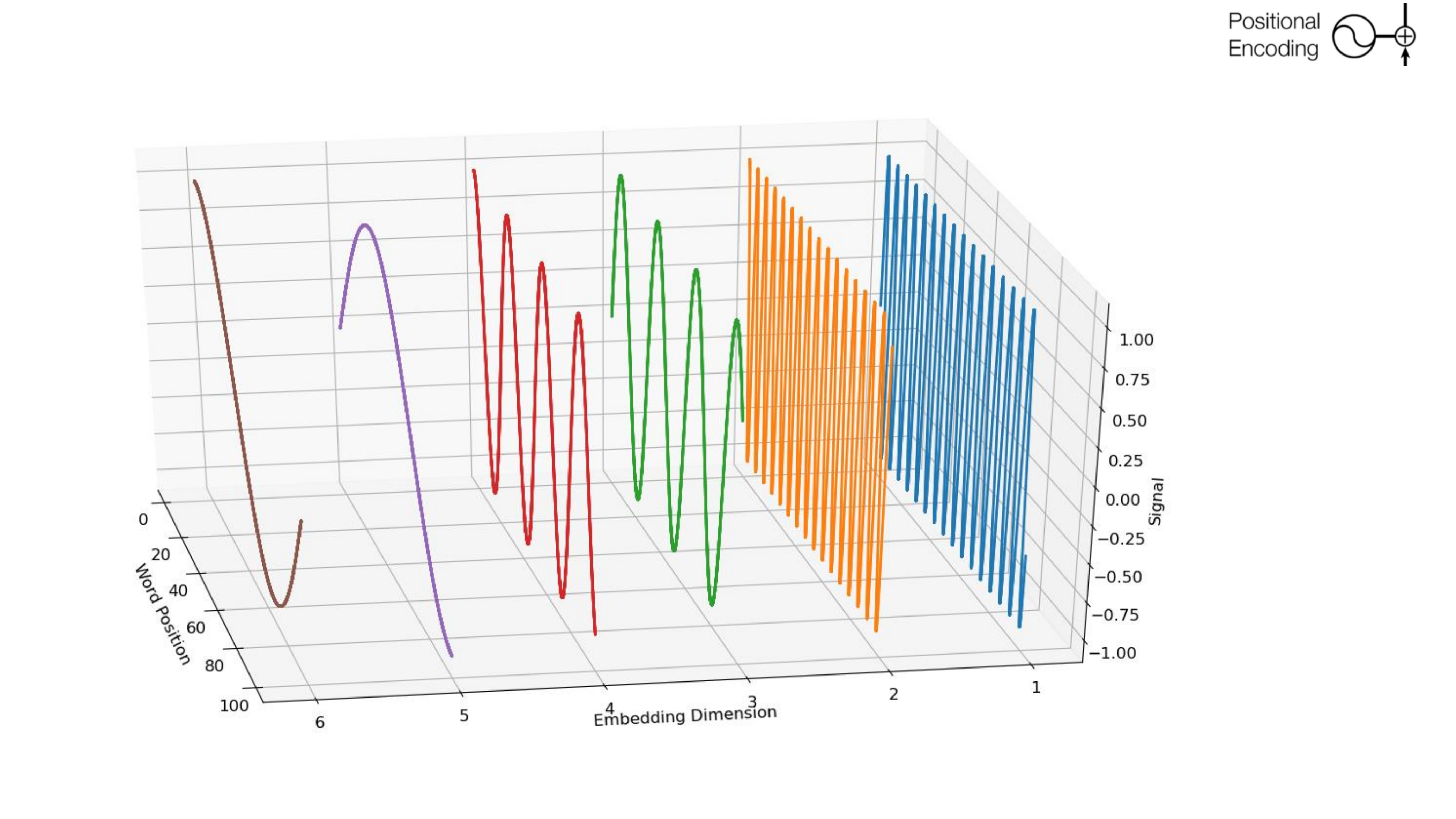
$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

- *pos* = Word Position, d_{model} = Embedding Dimension, *i* = i-th Dimension

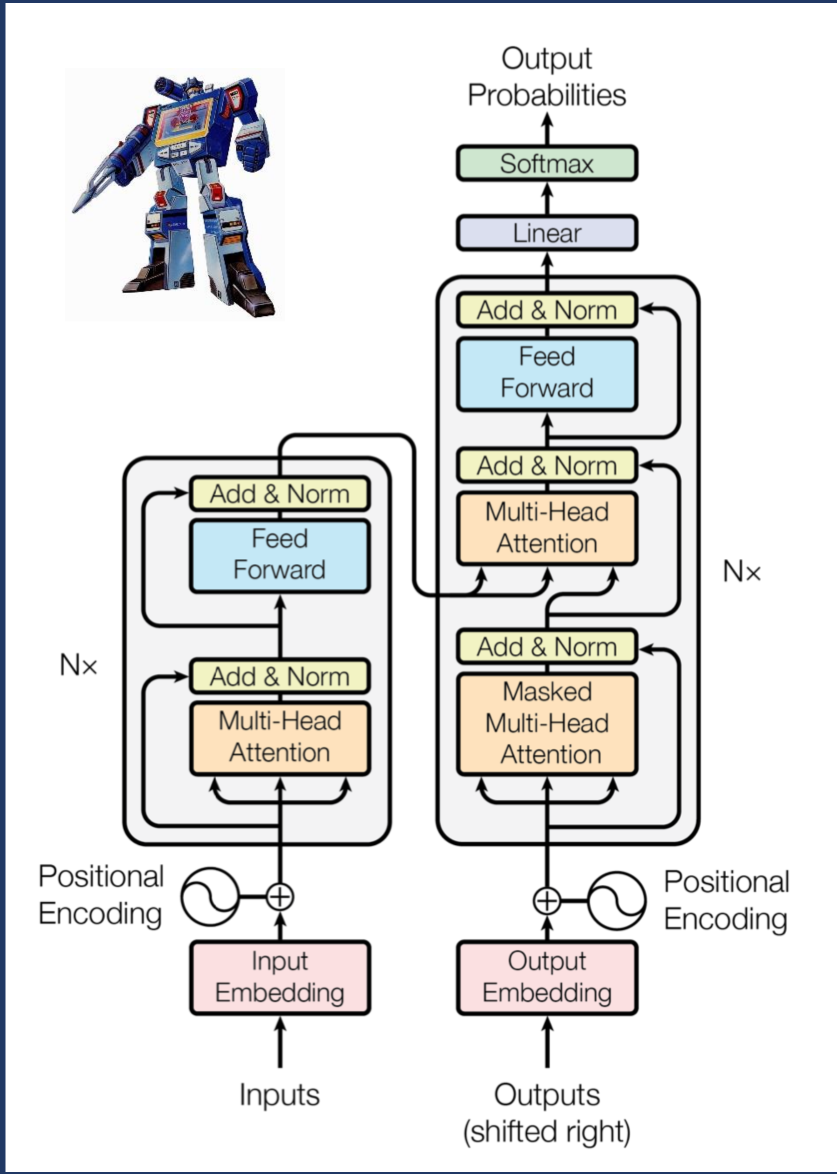
- Longest sequence with unique position representations: 10000 steps
- For any fixed offset *k*, PE_{pos+k} can be represented as linear function of PE_{pos}

Ashish Vaswani, et al. (2017). Attention Is All You Need.
Peter Shaw, Jakob Uszkoreit, & Ashish Vaswani. (2018). Self-Attention with Relative Position Representations.

Positional Encoding — A Notion of Order



Transformer — Is Attention All You Need?



Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.

Transformer — Results

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 [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	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**	**41.0**	$2.3 \cdot 10^{19}$	

Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.

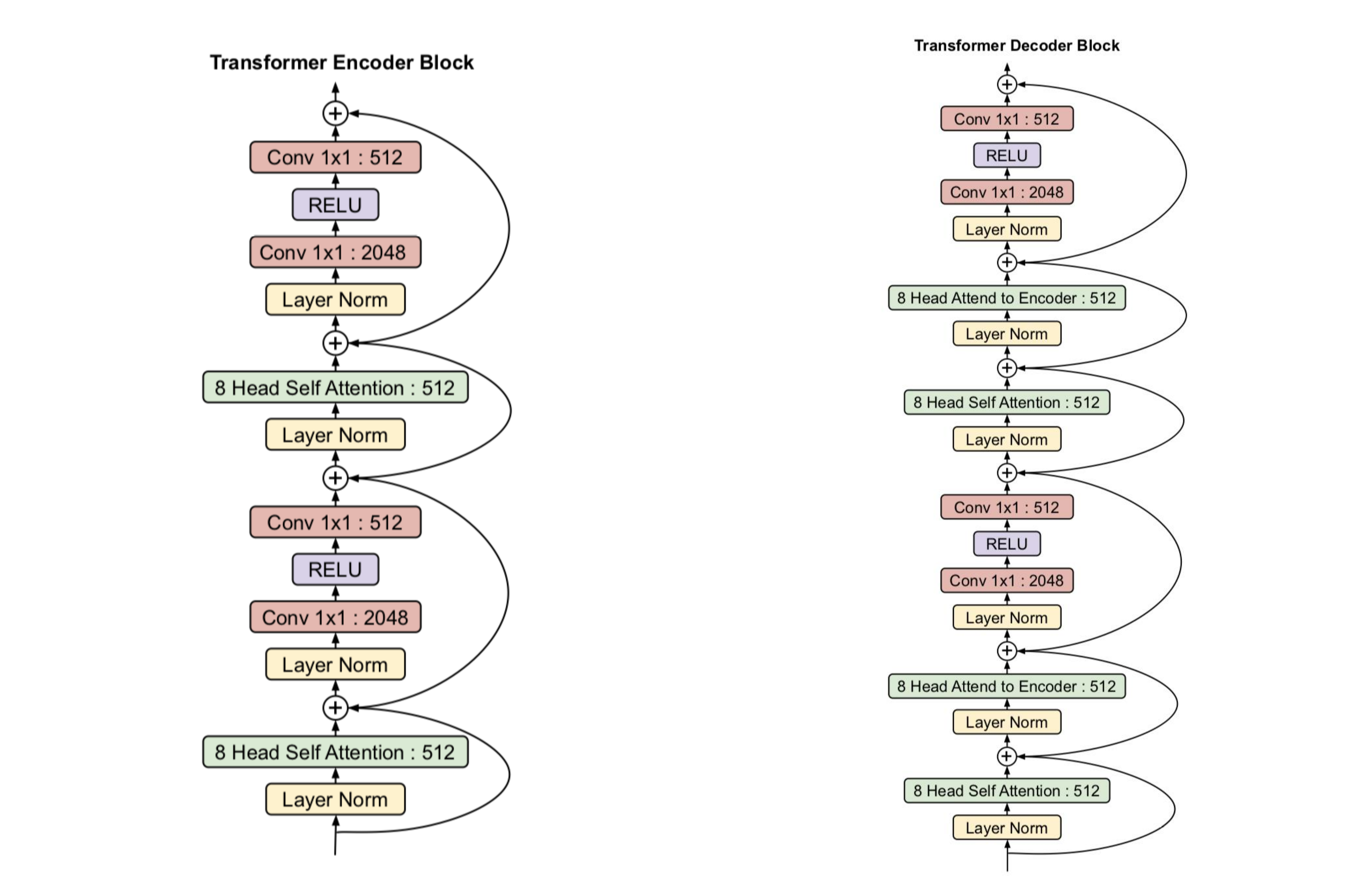
Next Step: The Evolved Transformer

- Transformer architecture is hand-engineered
- Why not let the computer find the best architecture?
- Apply a *neural architecture search* using an Evolution Strategy
  - Randomly create different architectures and test them on the data
  - Mutate the best architectures and repeat testing

→ The Evolved Transformer

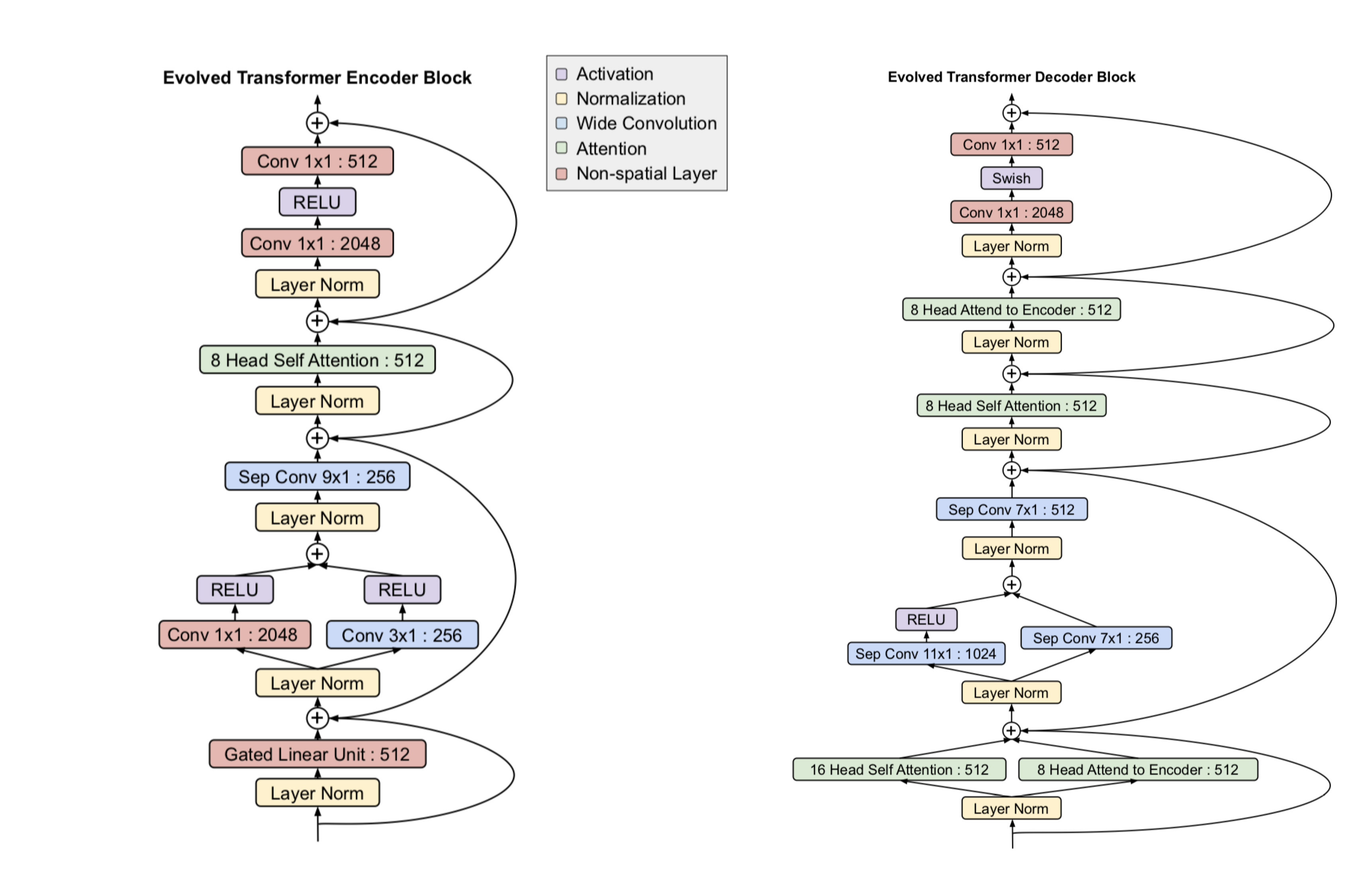
So, David R., Chen Liang, and Quoc V. Le. "The Evolved Transformer." *arXiv preprint arXiv:1901.11117* (2019).

The Transformer



So, David R., Chen Liang, and Quoc V. Le. "The Evolved Transformer." *arXiv preprint arXiv:1901.11117* (2019).

The Evolved Transformer



So, David R., Chen Liang, and Quoc V. Le. "The Evolved Transformer." *arXiv preprint arXiv:1901.11117* (2019).

Evolved Transformer vs. Transformer — Results

Model	Embedding Size	Parameters	Perplexity	BLEU	Δ BLEU
Transformer	128	7.0M	8.62 ± 0.03	21.3 ± 0.1	-
ET	128	7.2M	$\mathbf{7.62} \pm 0.02$	$\mathbf{22.0} \pm 0.1$	**+ 0.7**
Transformer	432	45.8M	4.65 ± 0.01	27.3 ± 0.1	-
ET	432	47.9M	$\mathbf{4.36} \pm 0.01$	$\mathbf{27.7} \pm 0.1$	+ 0.4
Transformer	512	61.1M	4.46 ± 0.01	27.7 ± 0.1	-
ET	512	64.1M	$\mathbf{4.22} \pm 0.01$	$\mathbf{28.2} \pm 0.1$	+ 0.5
Transformer	768	124.8M	4.18 ± 0.01	28.5 ± 0.1	-
ET	768	131.2M	$\mathbf{4.00} \pm 0.01$	$\mathbf{28.9} \pm 0.1$	+ 0.4
Transformer	1024	210.4M	4.05 ± 0.01	28.8 ± 0.2	-
ET	1024	221.7M	$\mathbf{3.94} \pm 0.01$	$\mathbf{29.0} \pm 0.1$	+ 0.2

Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems*. 2017.

Machine Translation — State of the Art

- Neural Machine Translation beats SMT
- Large differences between language pairs:
  Translating between English and French is much easier than between English and German!

- Current research:
  - Machine Translation without parallel data
  - Machine Translation in low resource languages

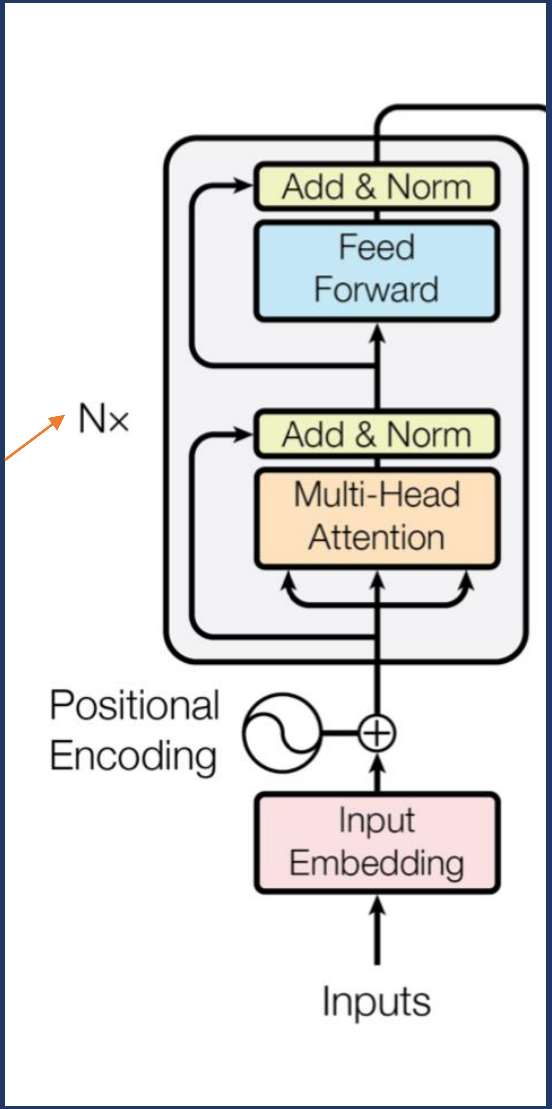
BERT

BERT — Bidirectional Encoder Representations from Transformers

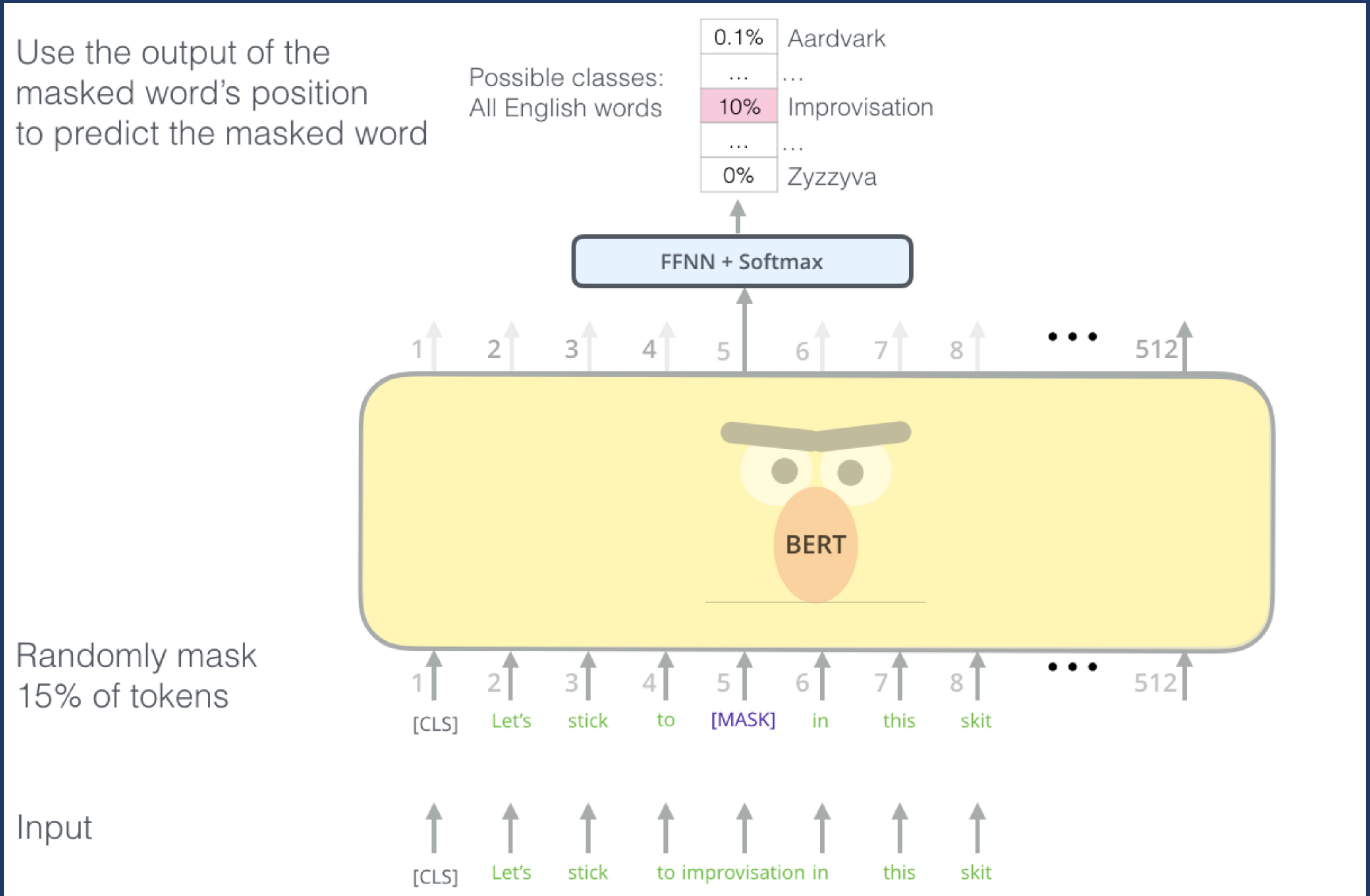
- Train a **Transformer** encoder
- Feed whole sentence to network but mask out words
  - Get bidirectional information with one model

- Train model on multiple tasks for which a big amount of data exists:
  - Predict hidden word („masked language model“)
  - Randomly replace words (instead of hiding) and let the model predict the correct word
  - Ask model if a sentence follows on another sentence

up to N=24

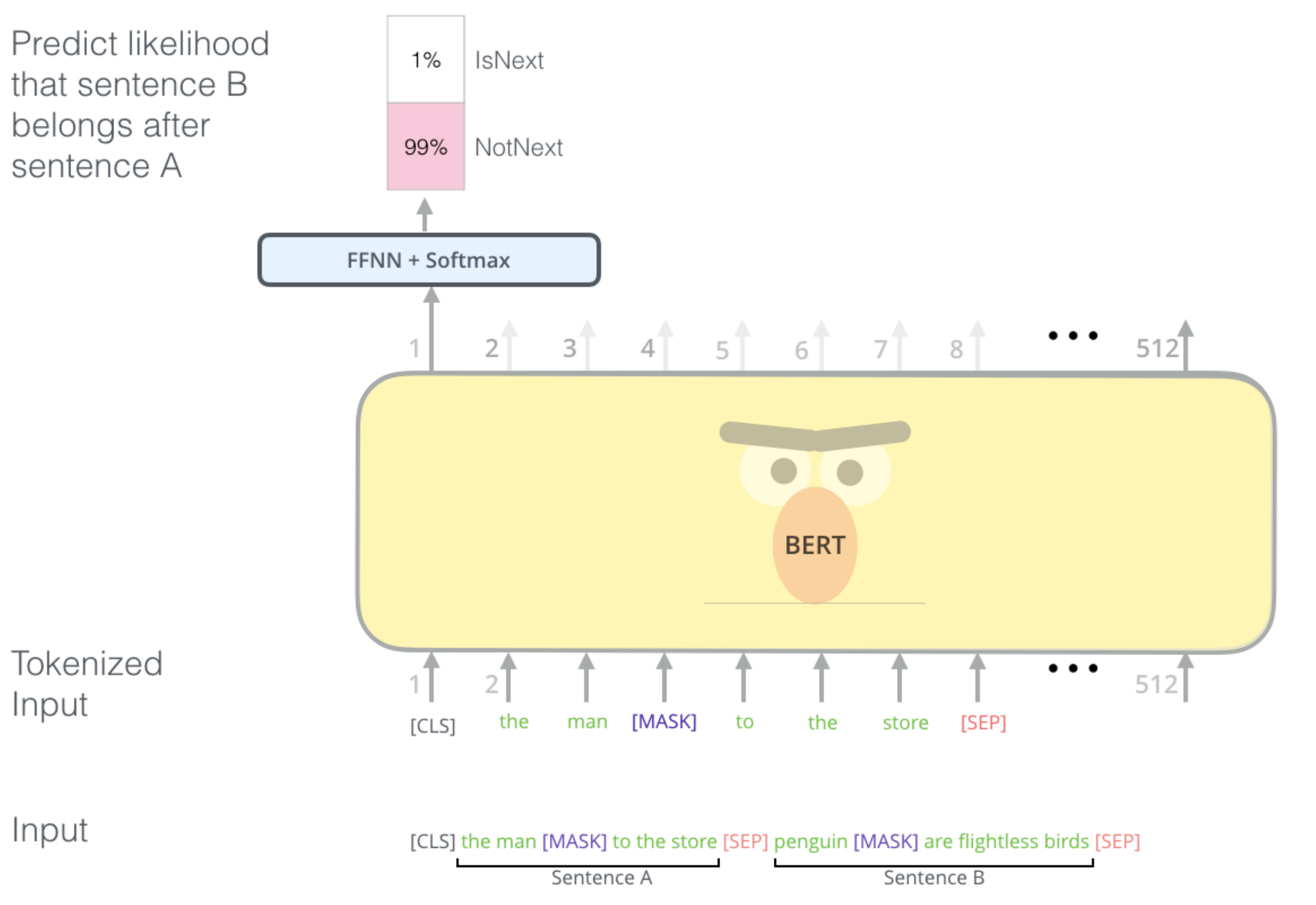


BERT — Masked Language Model



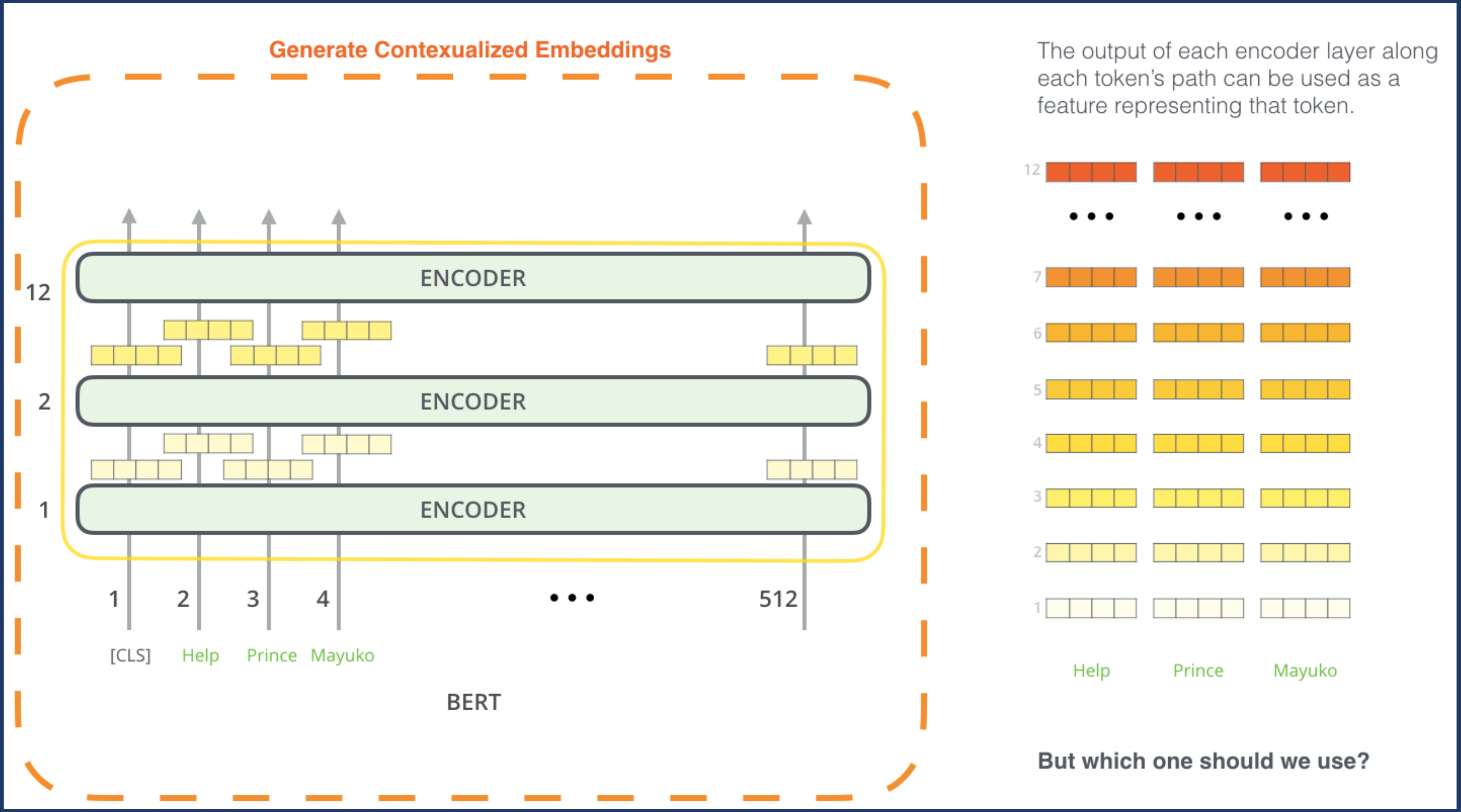
http://jalammar.github.io/illustrated-bert/

BERT — Next Sentence Prediction



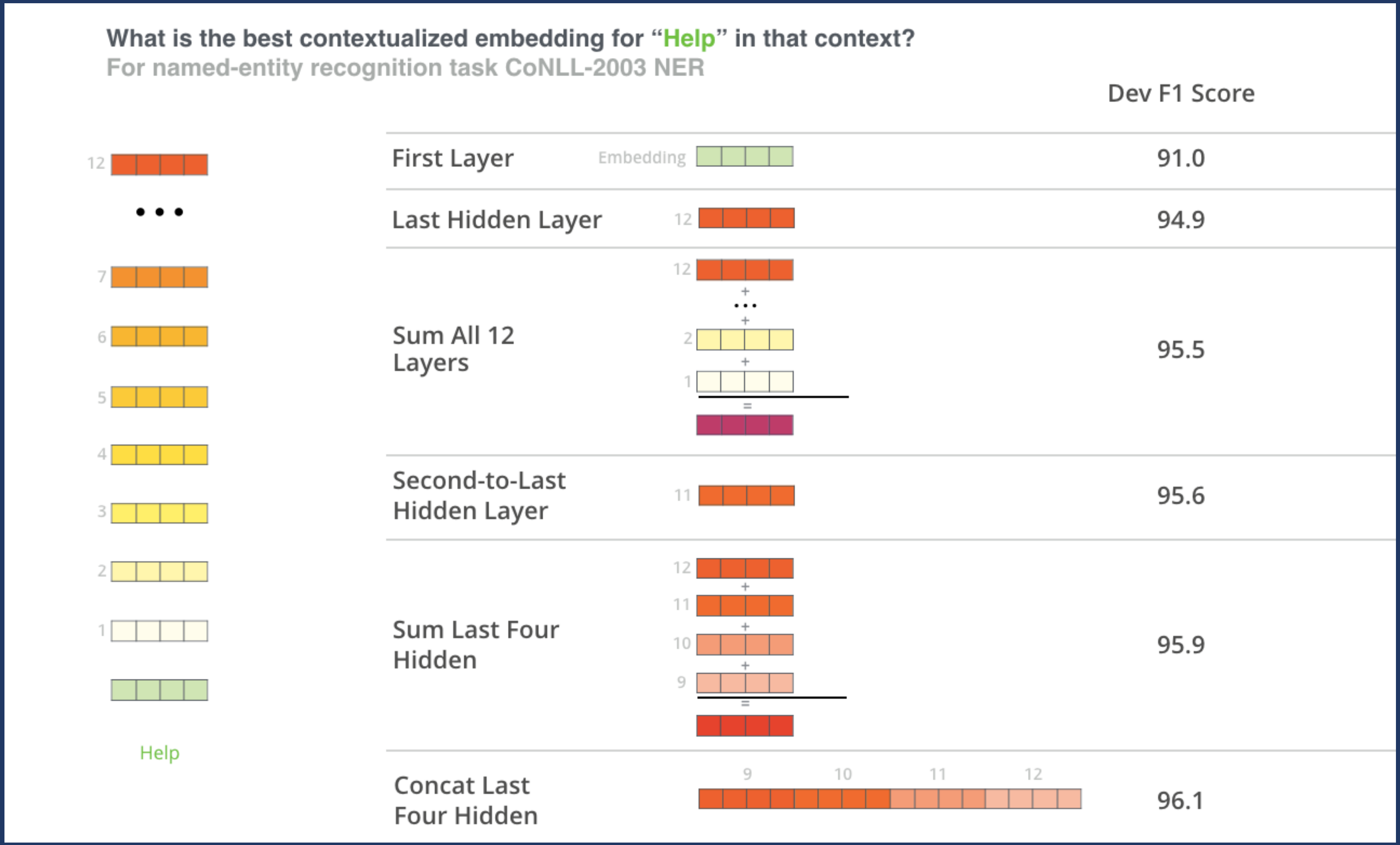
http://jalammar.github.io/illustrated-bert/

BERT — Contextualised Word Embeddings



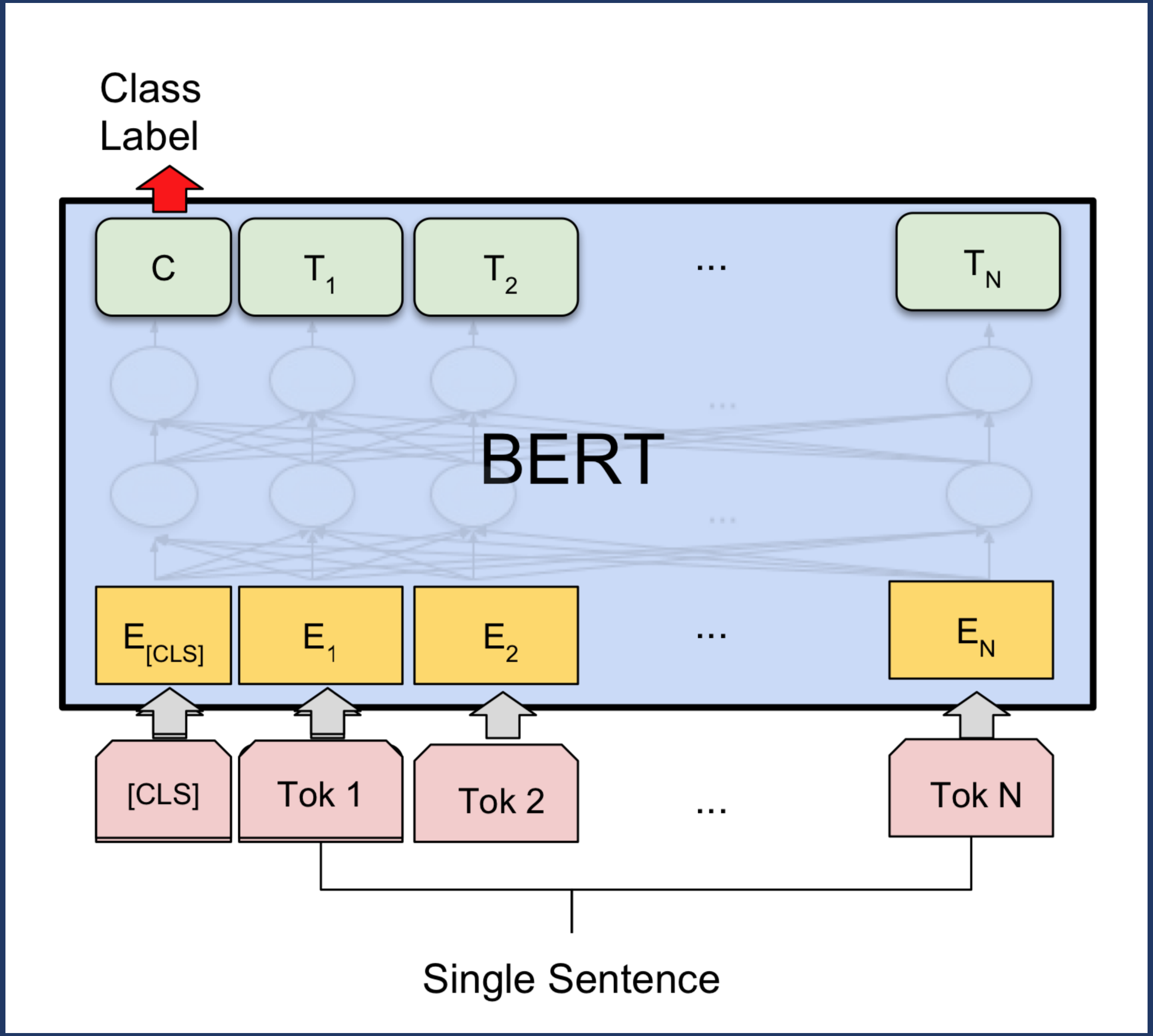
http://jalammar.github.io/illustrated-bert/

BERT — Contextualised Word Embeddings



http://jalammar.github.io/illustrated-bert/

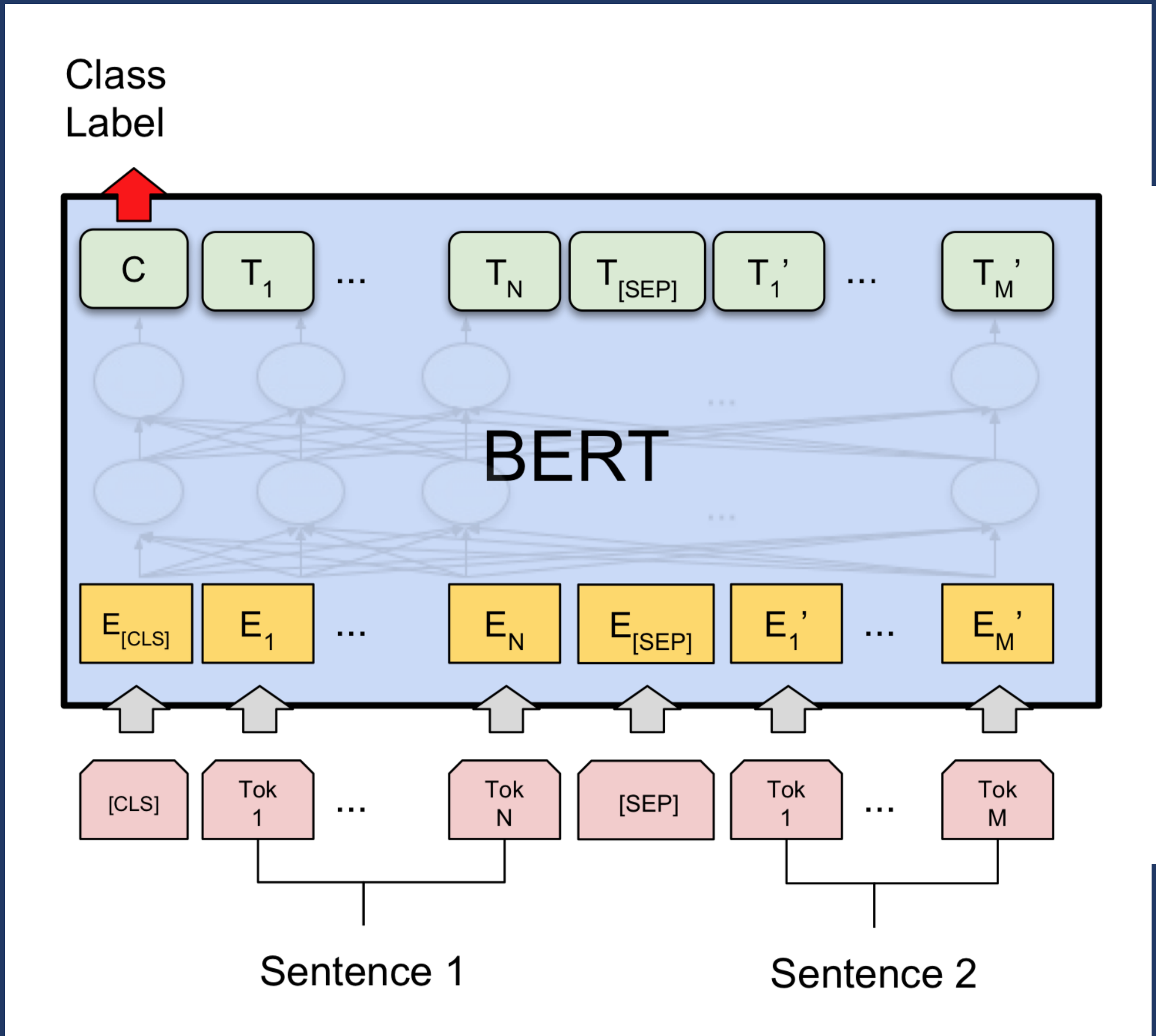
BERT — Single Sentence Classification Tasks



E.g. Sentiment Analysis
on Stanford Sentiment Treebank

Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

BERT — Sentence Pair Classification Tasks



E.g. SWAG dataset:

On stage, a woman takes a seat at the piano. She
- a) sits on a bench as her sister plays with the doll.
- b) smiles with someone as the music plays.
- c) is in the crowd, watching the dancers.
- **d) nervously sets her fingers on the keys.**

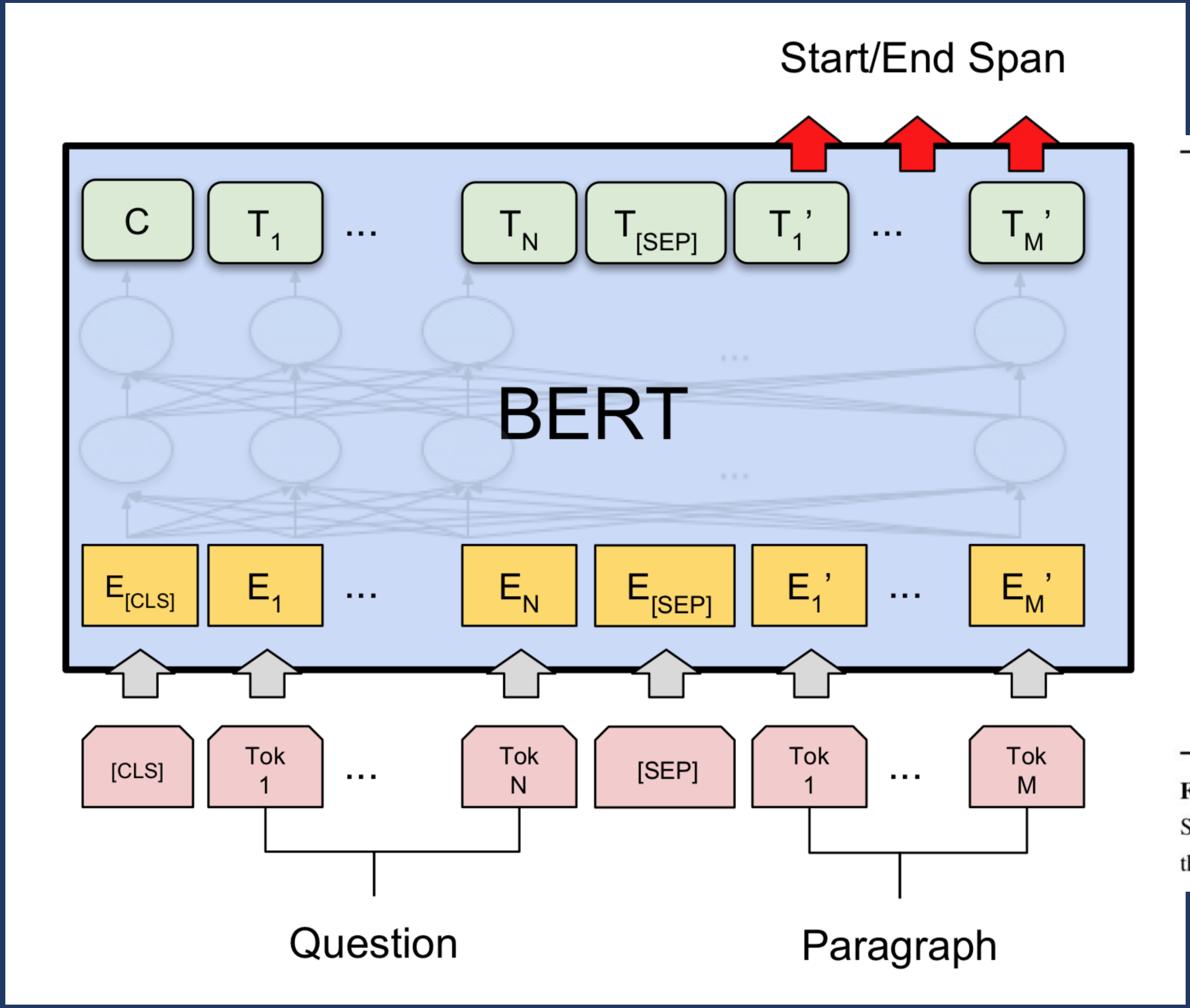
A girl is going across a set of monkey bars. She
- a) jumps up across the monkey bars.
- b) struggles onto the monkey bars to grab her head.
- **c) gets to the end and stands on a wooden plank.**
- d) jumps up and does a back flip.

The woman is now blow drying the dog. The dog
- **a) is placed in the kennel next to a woman's feet.**
- b) washes her face with the shampoo.
- c) walks into frame and walks towards the dog.
- d) tried to cut her face, so she is trying to do something very close to her face.

Table 1: Examples from SWAG; the correct answer is **bolded**. Adversarial Filtering ensures that stylistic models find all options equally appealing.

Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).
Zellers, Rowan, et al. "Swag: A large-scale adversarial dataset for grounded commonsense inference." *arXiv preprint arXiv:1808.05326* (2018).

BERT — Question Answering Tasks



E.g. SQuAD:

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under **gravity**. The main forms of precipitation include drizzle, rain, sleet, snow, **graupel** and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals **within a cloud**. Short, intense periods of rain in scattered locations are called "showers".

What causes precipitation to fall?
gravity

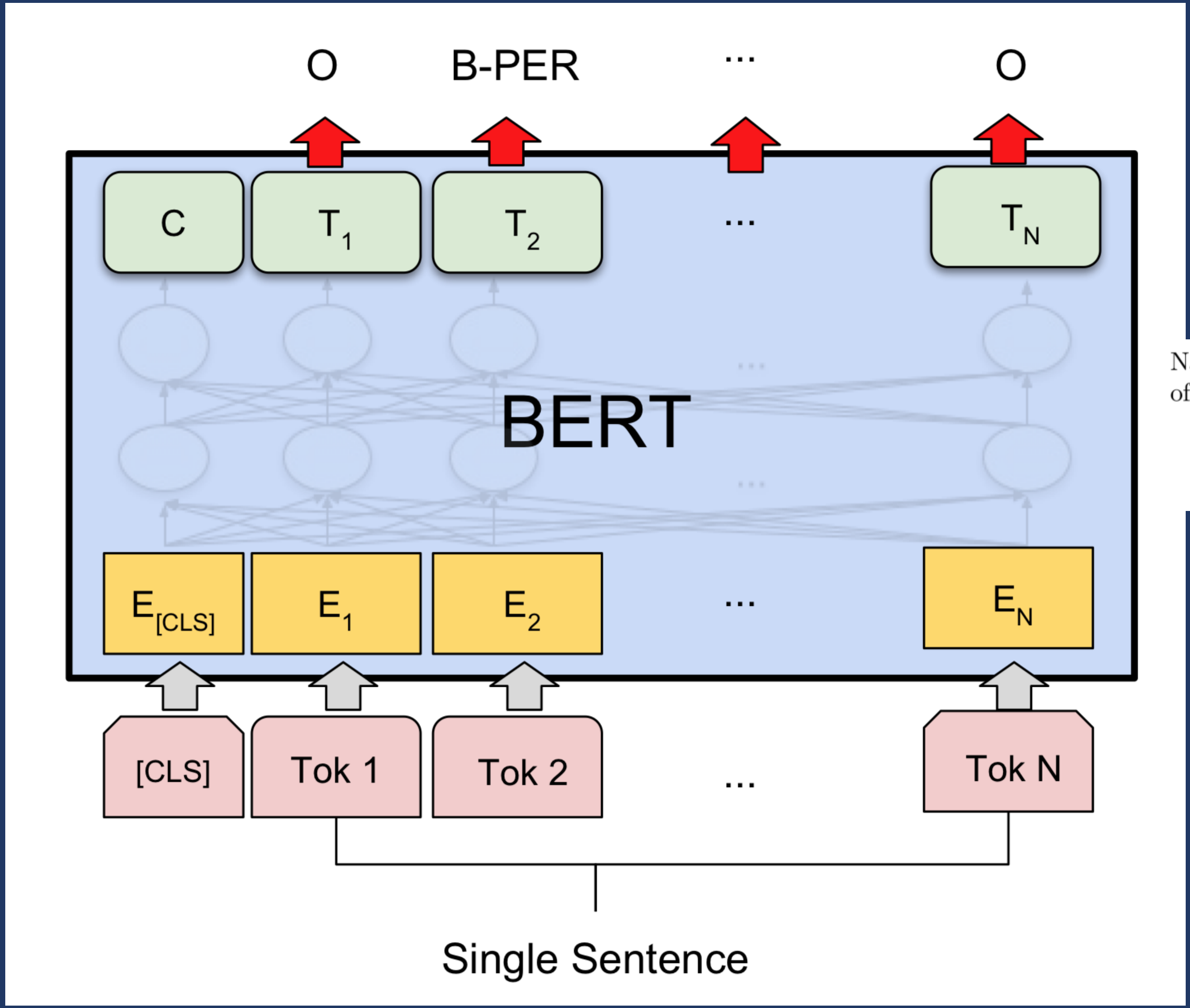
What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud

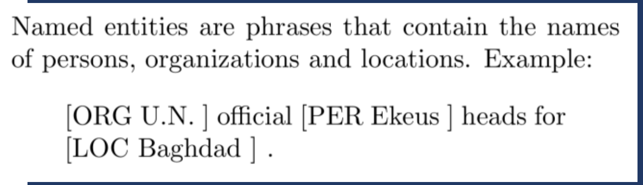
Figure 1: Question-answer pairs for a sample passage in the SQuAD dataset. Each of the answers is a segment of text from the passage.

Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).
Rajpurkar, Pranav, et al. "Squad: 100,000+ questions for machine comprehension of text." *arXiv preprint arXiv:1606.05250* (2016).

BERT — Single Sentence Tagging Tasks



E.g. CoNLL-2003 NER:


Named entities are phrases that contain the names of persons, organizations and locations. Example:

[ORG U.N.] official [PER Ekeus] heads for [LOC Baghdad] .

Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).
Sang, Erik F., and Fien De Meulder. "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition." *arXiv preprint cs/0306050* (2003).

BERT — Results

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.

BERT — Derivatives

- Since then various derivatives have been developed…

1. BERT (from Google) released with the paper BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding by Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova.
2. RoBERTa (from Facebook), released together with the paper a Robustly Optimized BERT Pretraining Approach by Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov.
3. DistilBERT (from HuggingFace) released together with the paper DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter by Victor Sanh, Lysandre Debut and Thomas Wolf. The same method has been applied to compress GPT2 into DistilGPT2.
4. CamemBERT (from FAIR, Inria, Sorbonne Université) released together with the paper CamemBERT: a Tasty French Language Model by Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suarez, Yoann Dupont, Laurent Romary, Eric Villemonte de la Clergerie, Djame Seddah, and Benoît Sagot.
5. ALBERT (from Google Research), released together with the paper a ALBERT: A Lite BERT for Self-supervised Learning of Language Representations by Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, Radu Soricut.
6. XLM-RoBERTa (from Facebook AI), released together with the paper Unsupervised Cross-lingual Representation Learning at Scale by Alexis Conneau*, Kartikay Khandelwal*, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer and Veselin Stoyanov.
7. FlauBERT (from CNRS) released with the paper FlauBERT: Unsupervised Language Model Pre-training for French by Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoît Crabbé, Laurent Besacier, Didier Schwab.

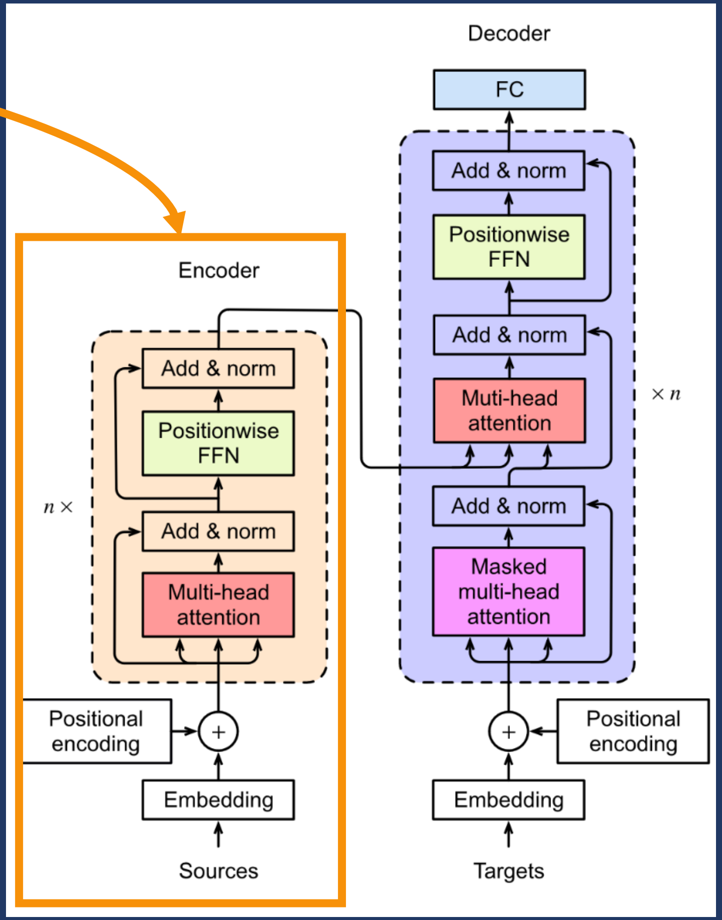
…

GPT

What about the Decoder?

BERT

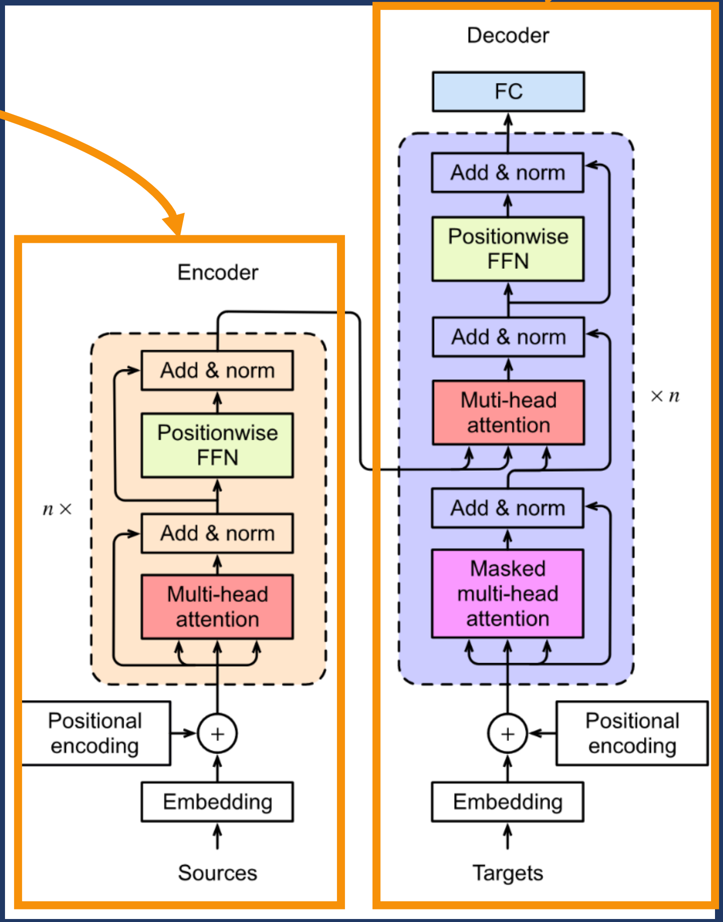
- Encoder-only
- Output:
  - Word Embeddings
- Creates representations of Input for:
  - Question Answering
  - Summarization
  - Named Entity Recognition
  - Semantic Similarity (SentenceBERT)
  - Recommendation
  - …



What about the Decoder?

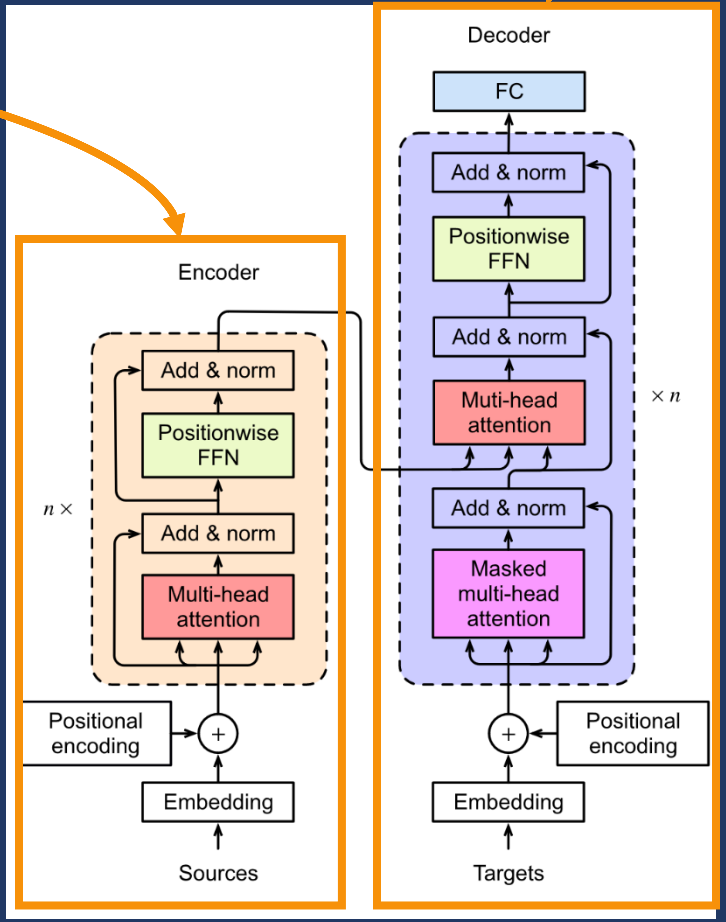
BERT
- Encoder-only
- Output:
  - Word Embeddings
- Creates representations of Input for:
  - Question Answering
  - Summarization
  - Named Entity Recognition
  - Semantic Similarity (SentenceBERT)
  - Recommendation
  - …

What if we use the Decoder only?



What about the Decoder?

BERT
- Encoder-only
- Output:
  - Word Embeddings
- Creates representations of Input for:
  - Question Answering
  - Summarization
  - Named Entity Recognition
  - Semantic Similarity (SentenceBERT)
  - Recommendation
  - …

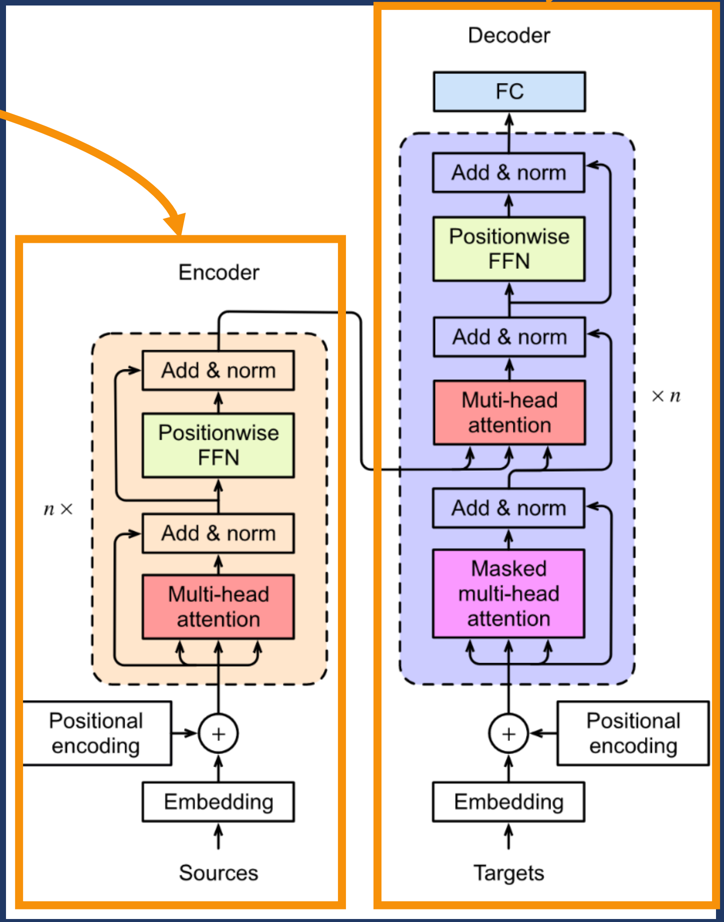


GPT
(Generative Pretrained Transformer)
- Decoder-only
- Output
  - Probability of next word/token
- Predicts continuation of text
  - Output is based on Input text
  - Answer question
  - Follow Instructions
  - "Learn" from examples
- Most famous example: ChatGPT

What about the Decoder?

BERT

- Encoder-only
- Output:
  - Word Embeddings
- Creates representations of Input for:
  - Question Answering
  - Summarization
  - Named Entity Recognition
  - Semantic Similarity (SentenceBERT)
  - Recommendation
  - …



GPT
(Generative Pretrained Transformer)

- Decoder-only
- Output
  - Probability of next word/token
- Predicts continuation of text
  - Output is based on Input text
  - Answer question
  - Follow Instructions
  - “Learn” from examples
- Most famous example: ChatGPT

Trained autoregressively

- Predict next token… (like RNN)
- … but based on entire input, not just hidden state
- Does not get to see the End of the Sequence (unlike BERT)

Vision Transformer

Slides in this section are based on the Lecture “Advanced Deep Learning - Large Language Models” by Katharina Breininger and Vincent Christlein at Friedrich-Alexander-Universität Erlangen-Nürnberg

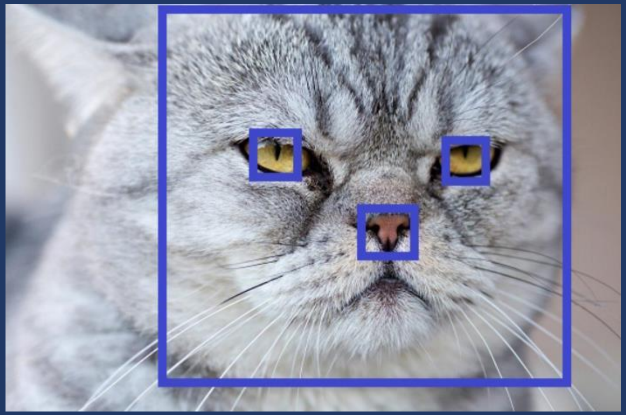
Revisiting CNNs

CNNs incorporate inductive bias

- Hierarchical organization
- Local connectivity
- Translational equivariance

→ Reduces what the network can represent
→ Receptive field strongly linked to network depth

Can we get rid of this Restriciton?
Yes*

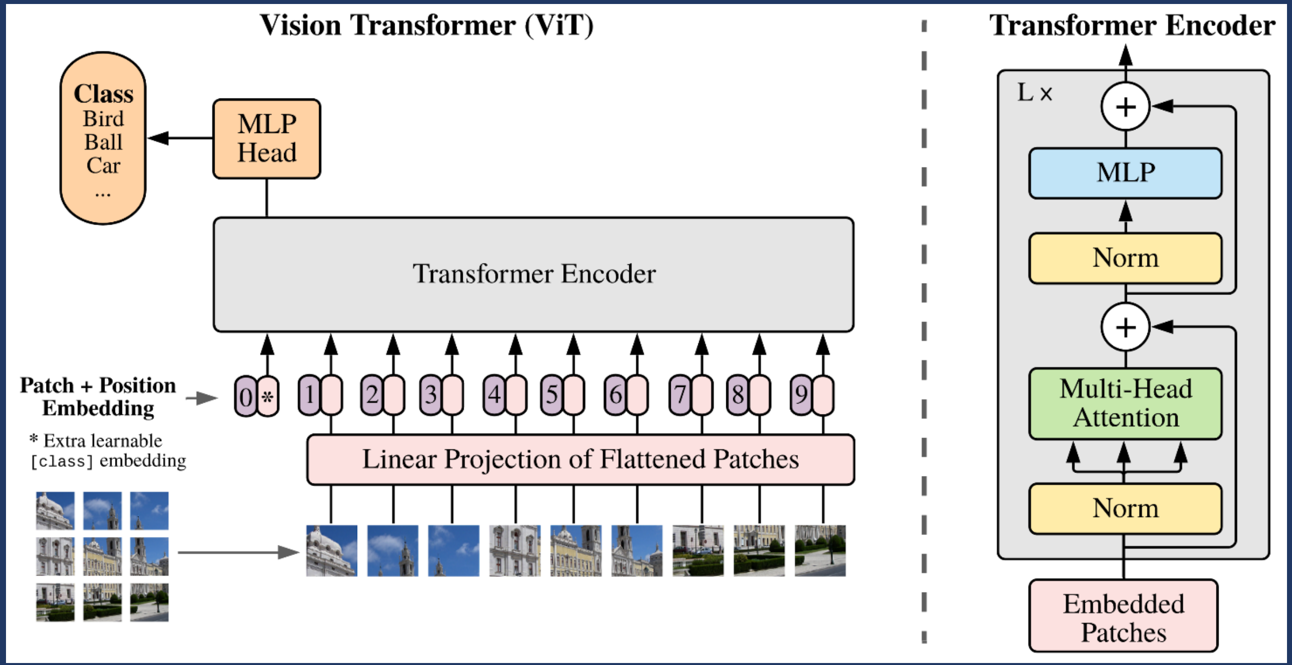


*Term and Conditions apply

Vision Transformer (ViT) - “An image is worth 16 × 16 words”

Core idea: Images are also just “sequences”

- Separate images into patches
- Transform patches to tokens
- Encode patch-tokens using Transformer



Source: Dosovitskiy, Beyer, Kolesnikov, et al. „An image is worth 16 × 16 words”, 2021

Vision Transformer (ViT) - “An image is worth 16 × 16 words”

Main parameters:

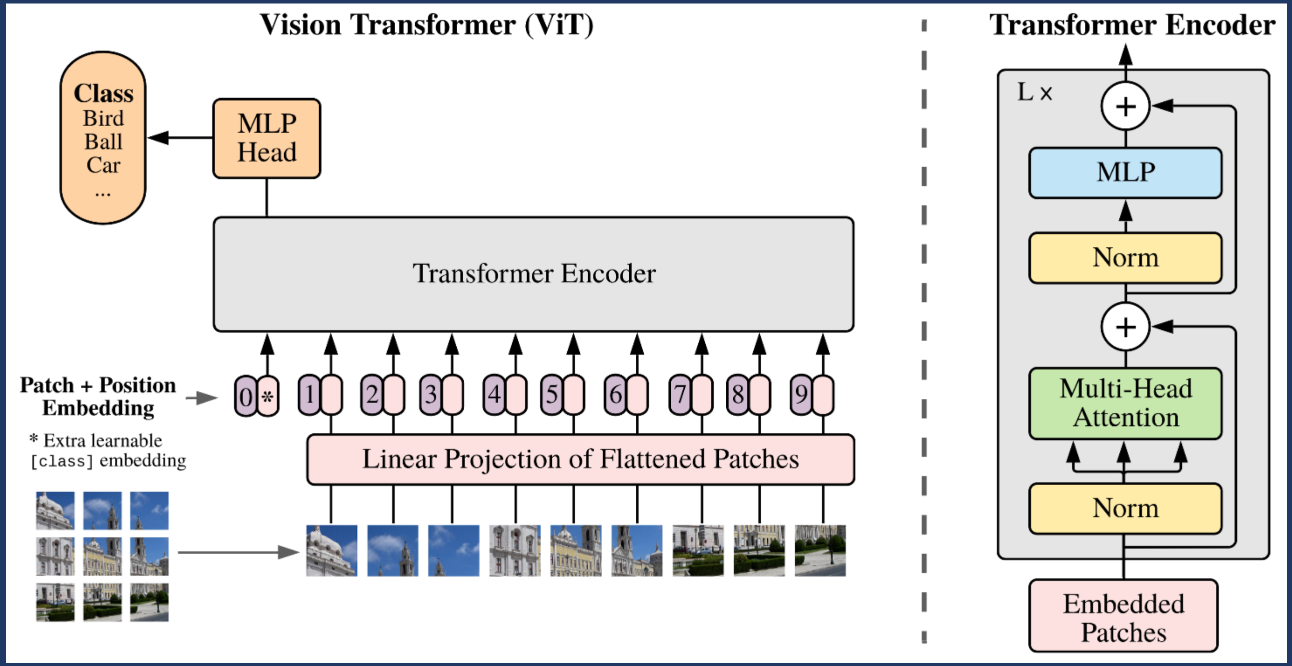
Size of input patches:

16×16 input patches - ViT-X/16

Transformer parameters:

Layers, hidden size, MLP size, heads ...

- ViT-Base (86M)
- ViT-Large (307M)
- ViT-Huge (632M)



Source: Dosovitskiy, Beyer, Kolesnikov, et al. „An image is worth 16 × 16 words”, 2021

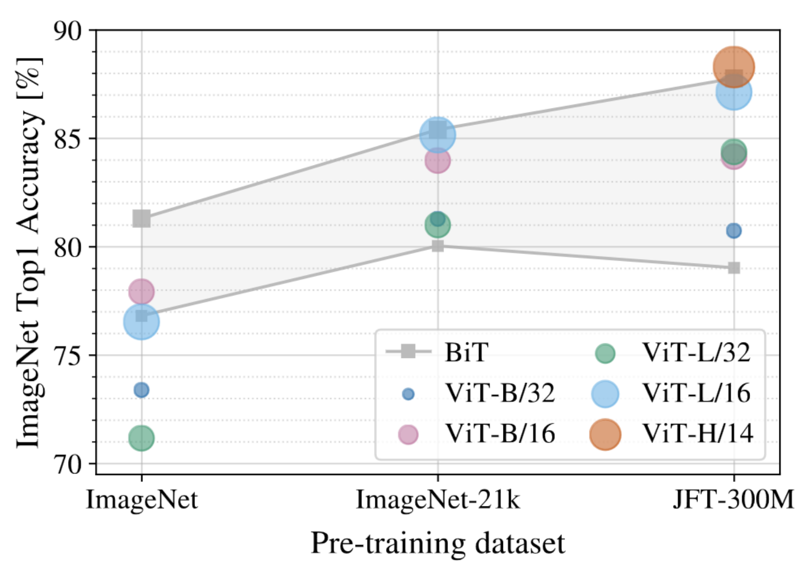
Vision Transformer (ViT) - “An image is worth 16 × 16 words”

Core insight: It works!

- SOTA for various image recognition benchmarks ...
- ... when pre-training on large-scale (!) datasets (JFT-300M)
- More efficient pre-training compared to (large) CNNs

	Ours-JFT (ViT-H/14)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** ± 0.04	87.54 ± 0.02	88.4
ImageNet ReaL	**90.72** ± 0.05	90.54	90.55
CIFAR-10	**99.50** ± 0.06	99.37 ± 0.06	—
CIFAR-100	**94.55** ± 0.04	93.51 ± 0.08	—
Oxford-IIIT Pets	**97.56** ± 0.03	96.62 ± 0.23	—
Oxford Flowers-102	**99.68** ± 0.02	99.63 ± 0.03	—
VTAB (19 tasks)	**77.63** ± 0.23	76.29 ± 1.70	—
TPUv3-core-days	2.5k	9.9k	12.3k

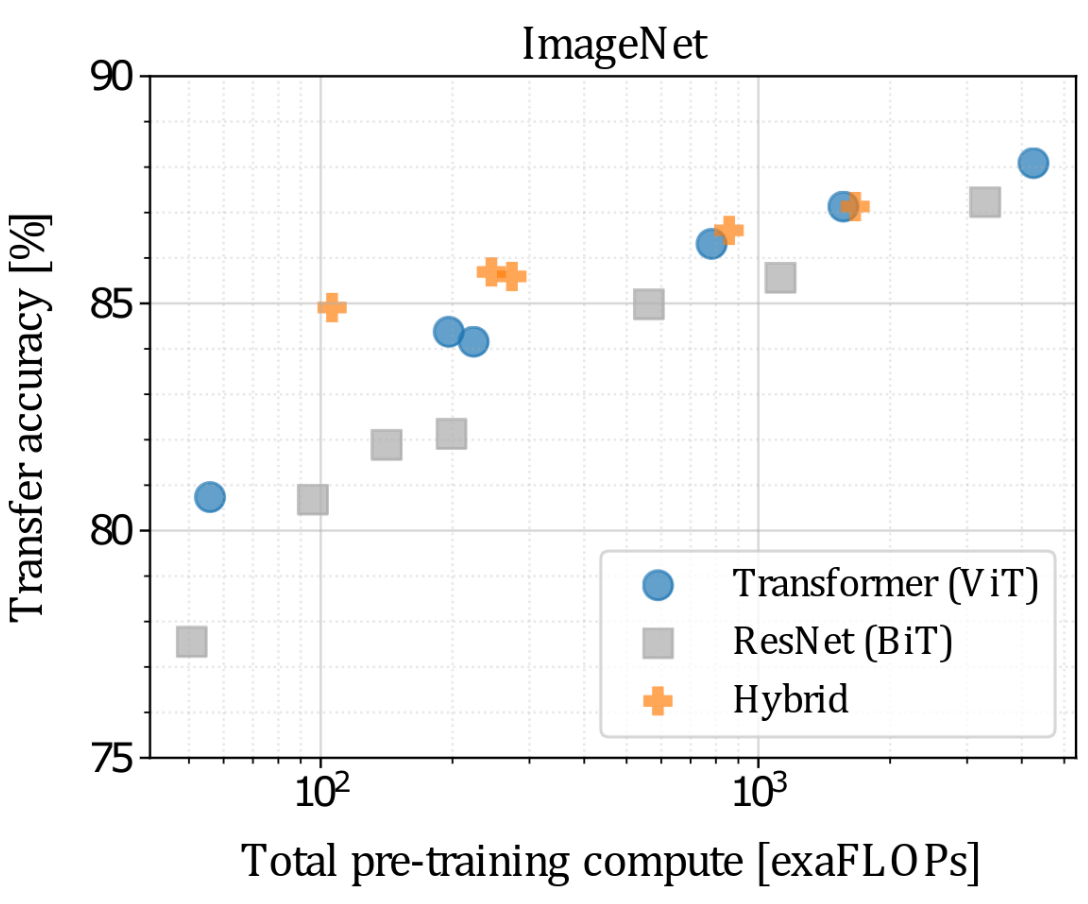
Source: Dosovitskiy, Beyer, Kolesnikov, et al. „An image is worth 16 × 16 words”, 2021 (adapted)



Vision Transformer (ViT) - "An image is worth 16 × 16 words"

Additional insights:

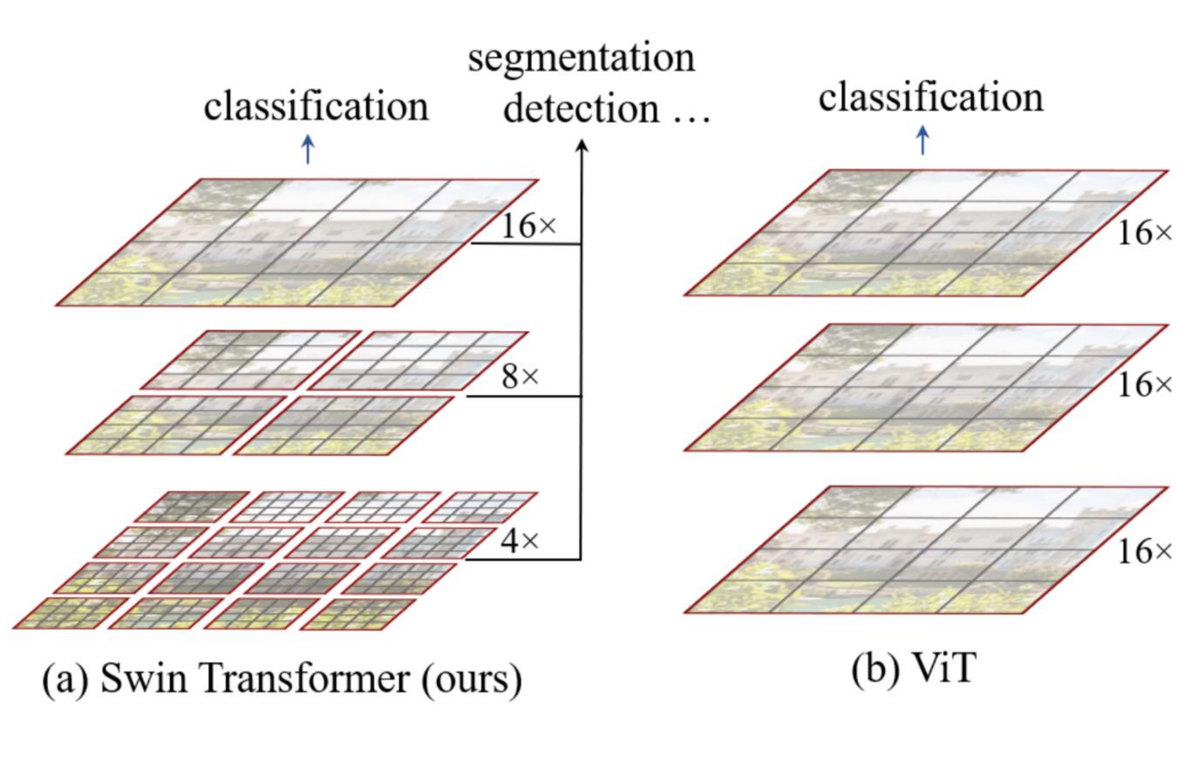
- Hybrid models are possible
  - boost performance for smaller data regimes
- Position-embeddings can be successfully learned
- Efficient implementations (LLMs) can be re-used



Source: Dosovitskiy, Beyer, Kolesnikov, et al. „An image is worth 16 × 16 words", 2021 (adapted)

Beyond Re-using Transformers - Swin-Transformers

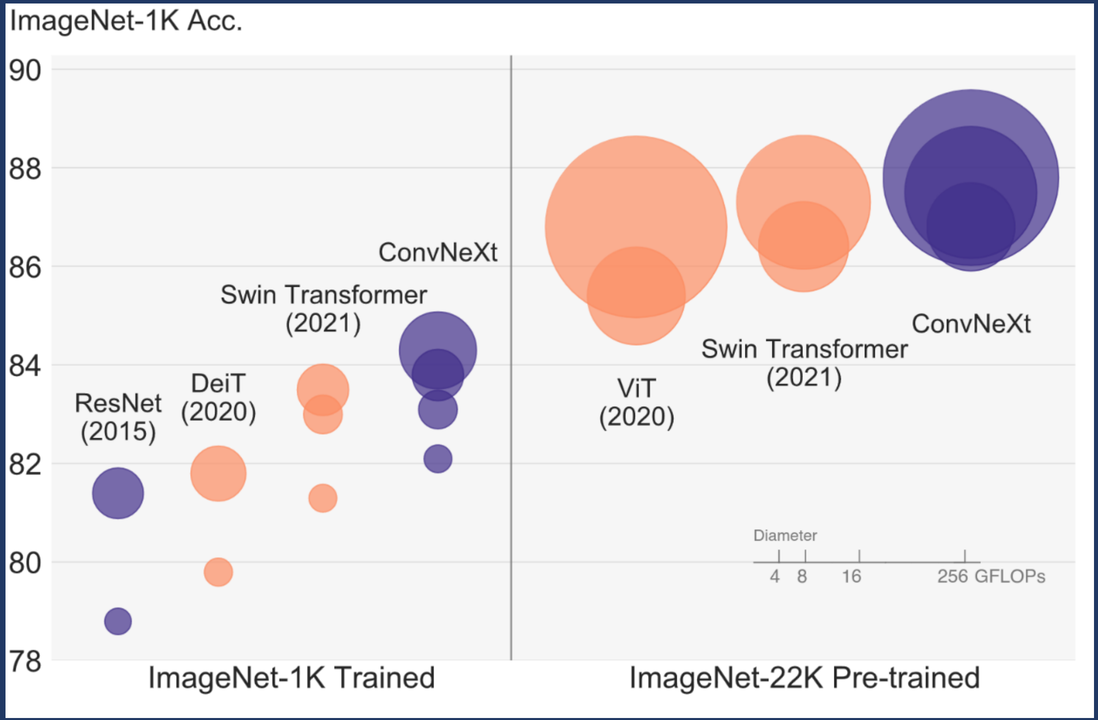
- Vanilla ViTs use uniform “resolution”
  → Can give a lot of freedom but data hungry
- Introduce inductive bias again:
  Hierarchical representation
- Hierarchy allows tasks at multiple scales
  - Classification
  - Object detection
  - Segmentation
- Similar concepts: SegFormer



Source: Ze Liu, Yutong Lin, Yue Cao, et al. “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows”, 2021 (adapted)

Beyond Transformers? CNNs or Transformers

- ConvNeXt: A Convnet for the 2020s
  - Transfers ideas from from Transformer Architectures to Convolutional Architectures
  - Pretrained on large datasets, CNNS scale similarly
  - CNNs are not obsolete
  - Attention appears to work better in transfer learning and multi-task learning



Source: Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, et al. “A ConvNet for the 2020s”, 2022 (adapted)

Finetuning your own Model

Finetuning your own Model

- Problem: Training a model from scratch can take a lot of data and time

- Solution: use pretrained models
  - Already knows how to interpret Language/Images/…

    → Transfer knowledge from pretraining

  - Fine-tune with your own data for your own task

Finetuning your own Model

But what if we cannot even run the model?

	Time (GPU hours)	Power Consumption (W)	Carbon Emitted(tCO2eq)
Llama 3 8B	1.3M	700	390
Llama 3 70B	6.4M	700	1900
Total	7.7M		2290

https://huggingface.co/meta-llama/Meta-Llama-3-8B

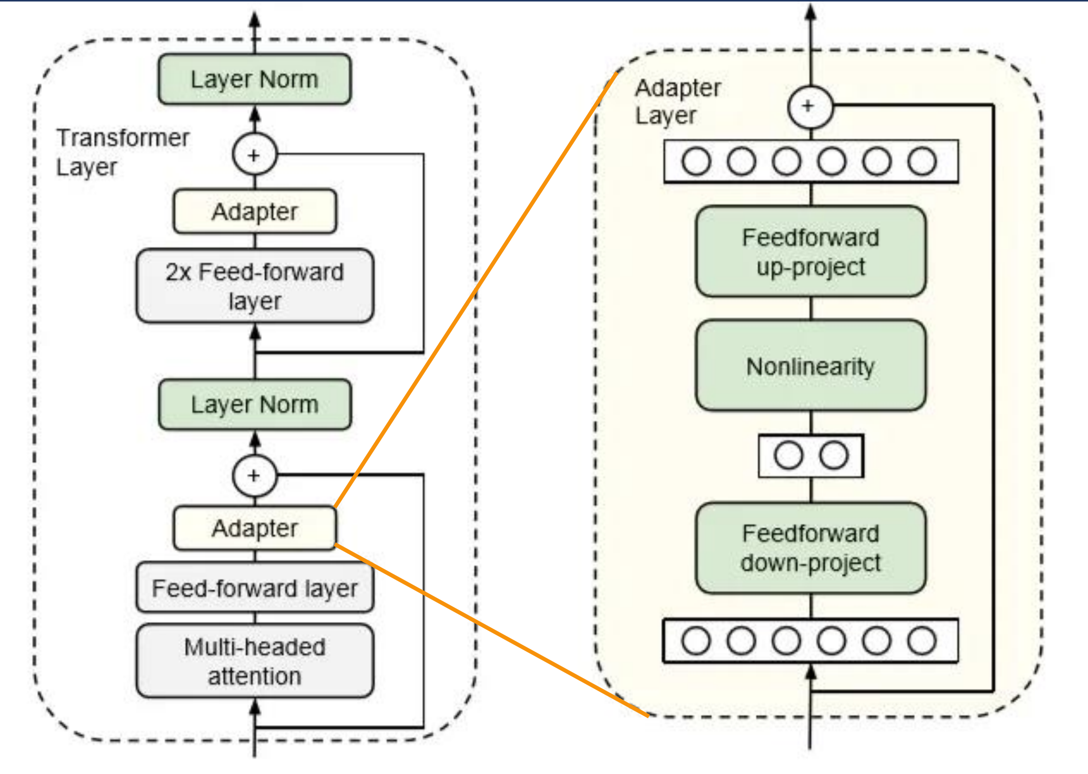
At 16 bit per parameter that's 140 GB VRAM just to load the model and perform inference

At least twice to train the model one sample at a time.

We'd really like to make use of all this pretraining

Finetuning your own Model - Adapters

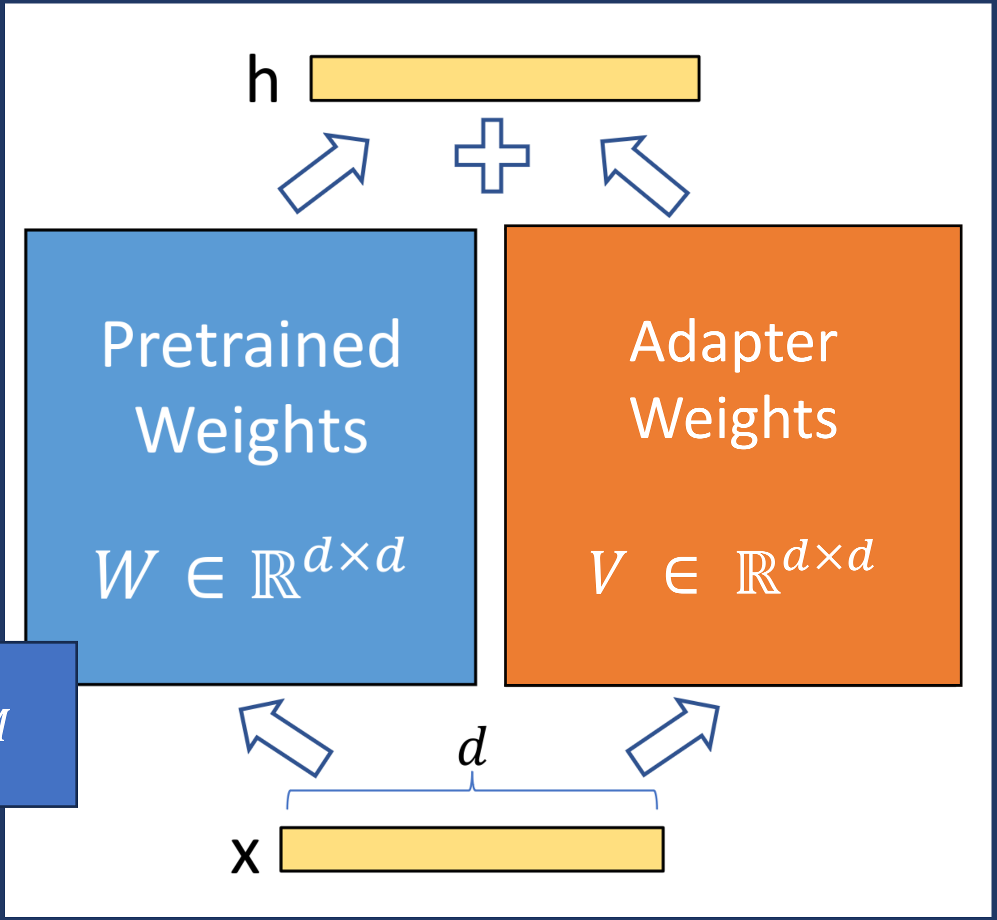
- Add adapter-layers between (some) pretrained layers
- Freeze the original model's weights
- Resulting model has comparable performance to full finetuning
- Need to train fewer weights, but still need to load the original model + added layers



Houlsby, Neil, et al. "Parameter-efficient transfer learning for NLP." International conference on machine learning. PMLR, 2019. (adapted)

Finetuning your own Model – LoRa (Low Rank Adaptation)

- Instead of inserting new layers, train an "offset" to the existing layers
$$h = Wx + Vx$$
- Updates in finetuning tend to focus on some specific aspects of the internal representation
  - few weights in $V \in \mathbb{R}^{d \times d}$ contain most of the information
  - $V \in \mathbb{R}^{d \times d}$ can be approximated via a low-rank matrix

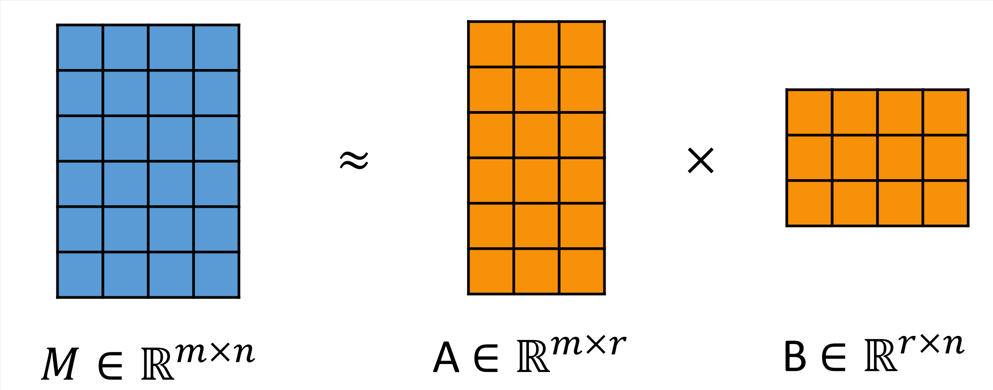


Hu, Edward J., et al. "Lora: Low-rank adaptation of large language models." arXiv preprint arXiv:2106.09685 (2021). (adapted)

$rank(M)$ is the number of linearly independent columns in M

Finetuning your own Model – LoRa (Low Rank Adaptation)

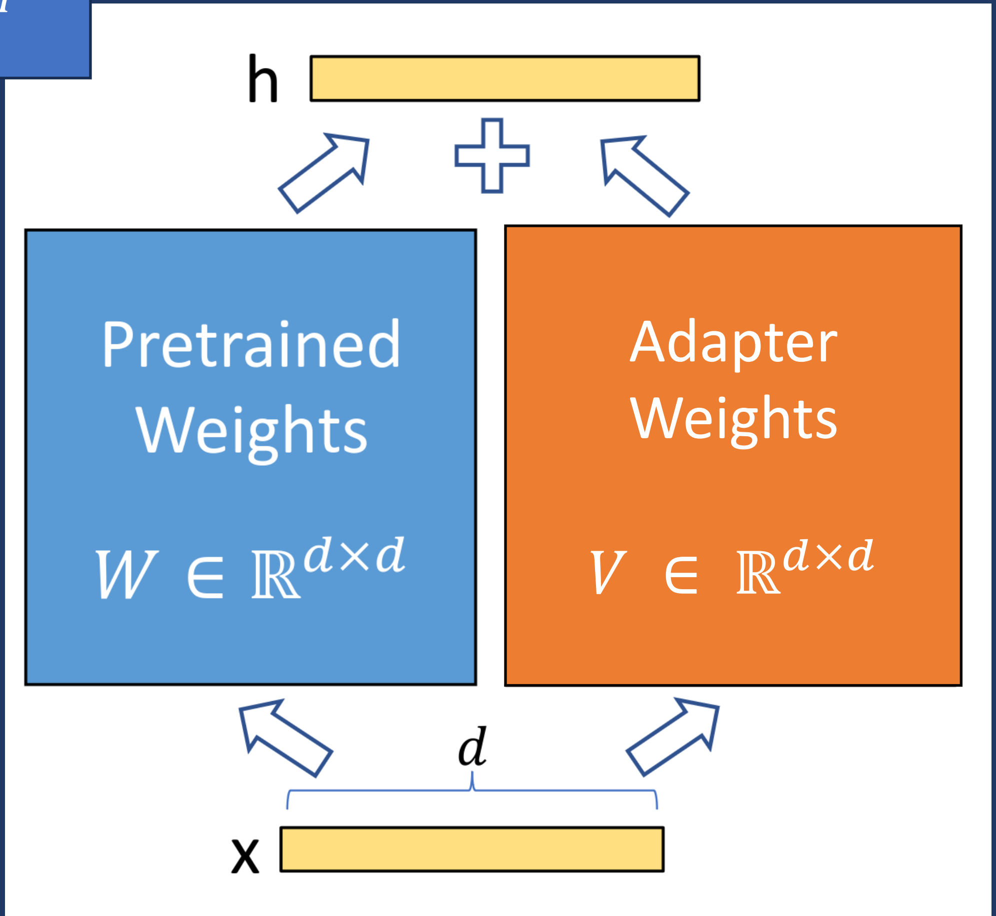
$rank(M)$ is the number of linearly independent columns in M



If $r \ll m, n$ this takes a lot fewer parameters

M has $m \times n$ parameters

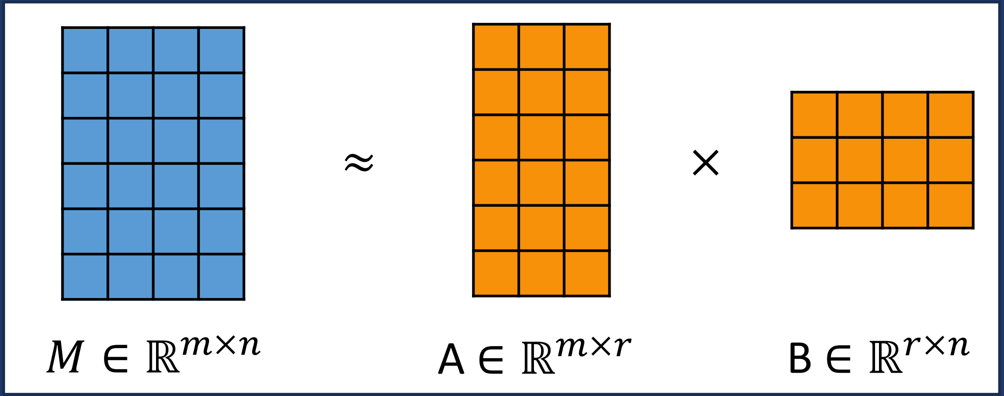
AB has $r(m+n)$ parameters



Hu, Edward J., et al. "Lora: Low-rank adaptation of large language models." arXiv preprint arXiv:2106.09685 (2021).

Finetuning your own Model – LoRa (Low Rank Adaptation)

$rank(M)$ is the number of linearly independent columns in M

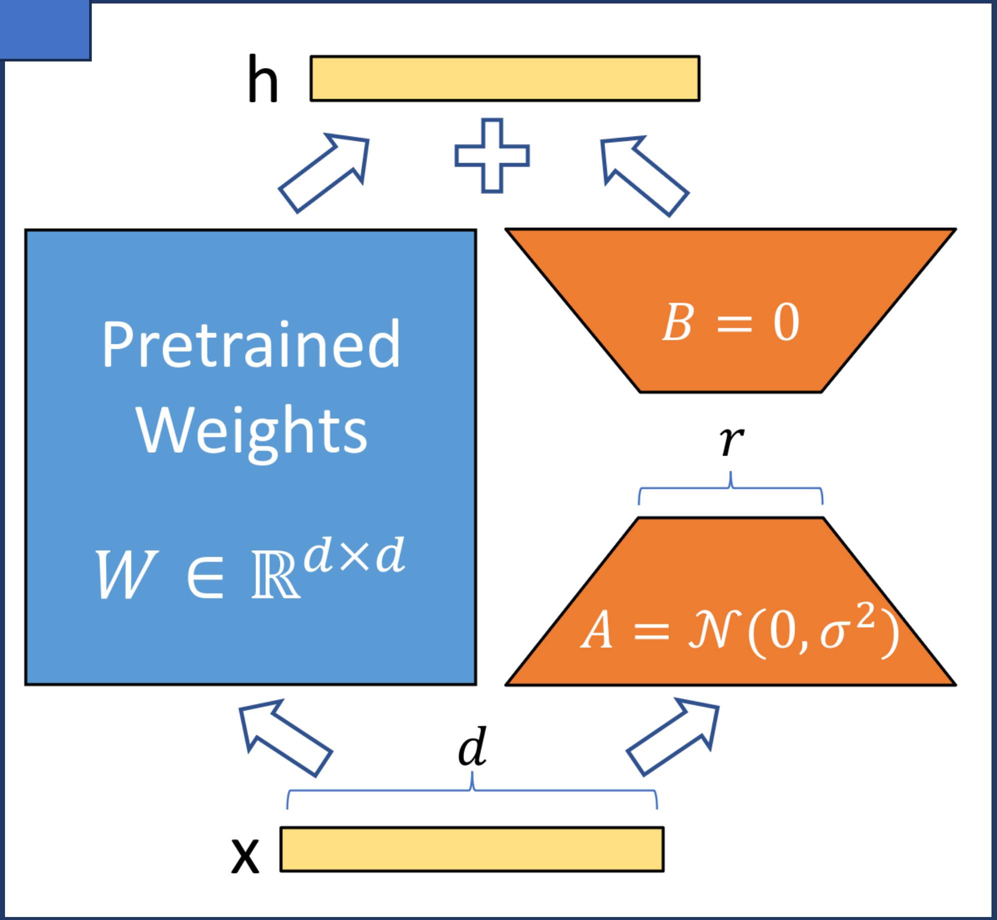


If $r \ll m, n$ this takes a lot fewer parameters

M has $m \times n$ parameters
AB has $r(m + n)$ parameters

We can learn two small matrices rather than a full adapter Matrix

$$h = Wx + Vx \approx Wx + BAx$$



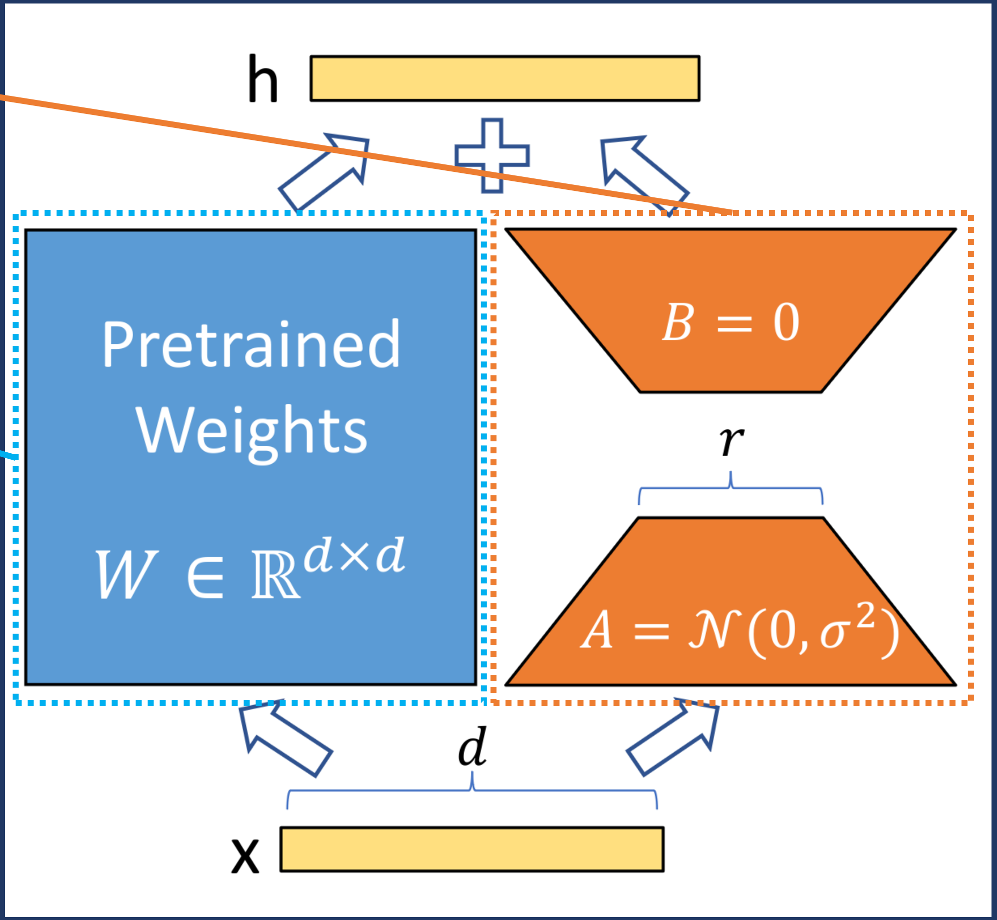
Hu, Edward J., et al. "Lora: Low-rank adaptation of large language models." arXiv preprint arXiv:2106.09685 (2021).

Finetuning your own Model – QLoRa

We saved parameters here

But we still need to perform the forward pass through the entire model

Solution: Model Quantization



Hu, Edward J., et al. "Lora: Low-rank adaptation of large language models." arXiv preprint arXiv:2106.09685 (2021).

Finetuning your own Model – QLoRa

Model Quantization

Model weights are typically stored in 16 or 32 Bits as floating point numbers

Idea: Map weights to a smaller number of approximate values

A constant, depending on the largest input

$$X^{Int8} = round\left(\boxed{\frac{127}{absmax(X^{Int8})}X^{Int8}}\right) = \mathrm{round}\left(c^{FP32} \times X^{FP32}\right)$$

1. Normalize weights with regard to largest input value
2. Map values to integers
3. Store integer weights
4. Dequantize these values only during computation:

$$dequant\left(c^{FP32}, X^{FP32}\right) = \frac{X^{Int8}}{c^{FP32}} \approx X^{FP32}$$

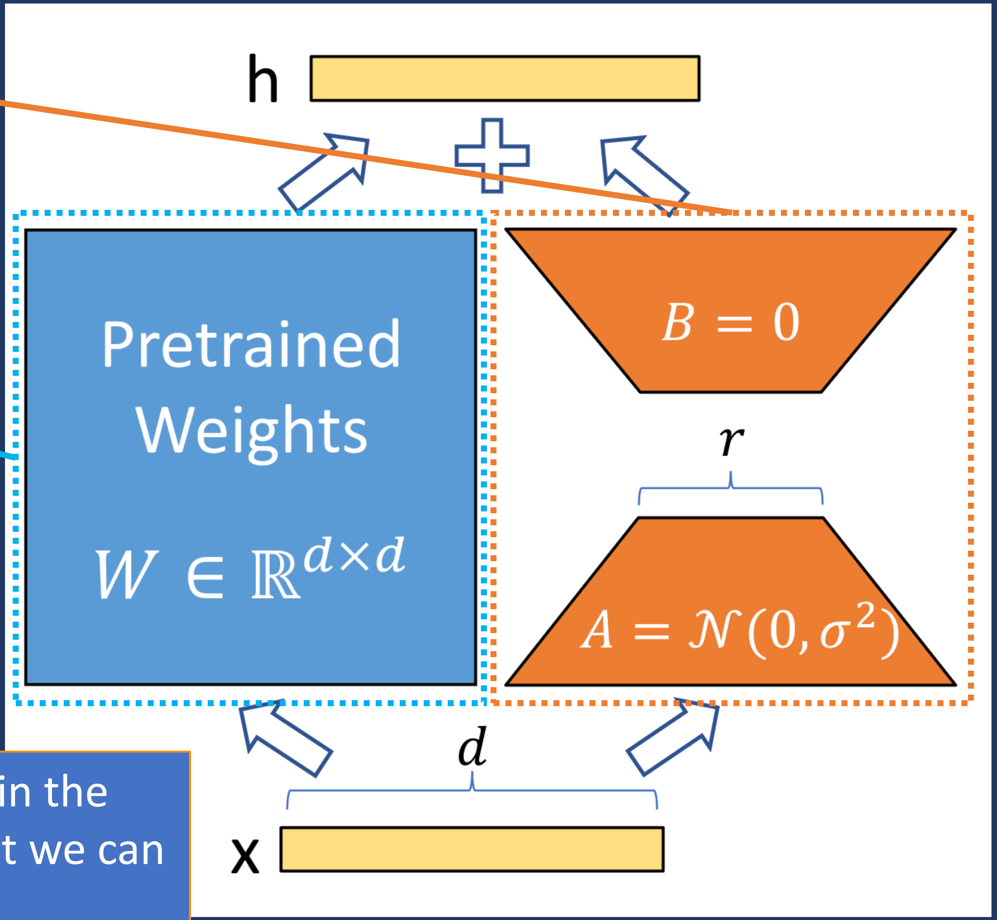
Outliers can cause issues → Split Matrix up into smaller chunks with their own c

Finetuning your own Model – QLoRa

We saved parameters here

And significantly reduced the memory requirements on this part

→ We can finetune many large models (like LLMs) on consumer Hardware



Note: We cannot train the quantized model, but we can merge in the trained adapters afterwards

Hu, Edward J., et al. "Lora: Low-rank adaptation of large language models." arXiv preprint arXiv:2106.09685 (2021).