Transformer and BERT

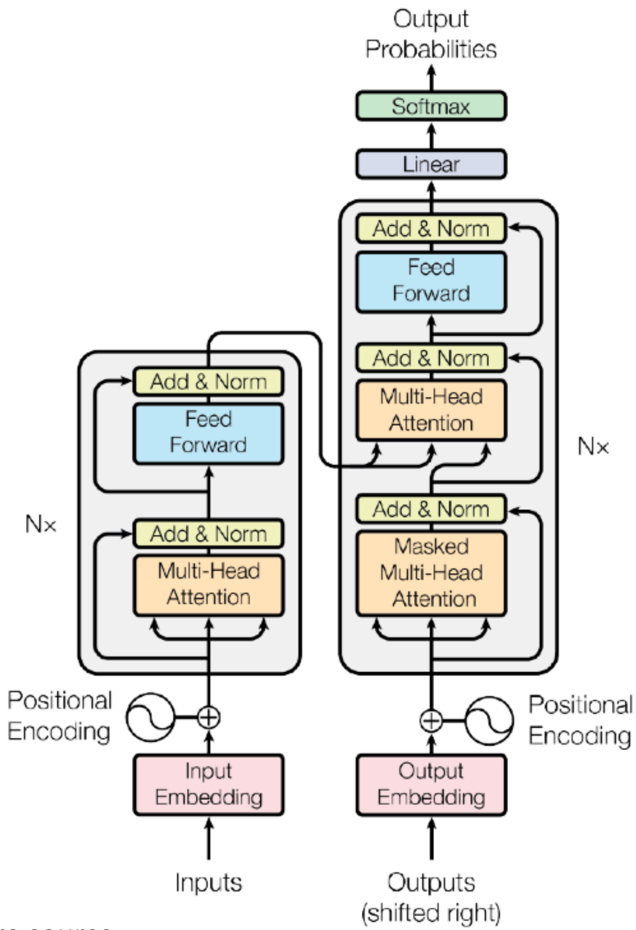
Debapriyo Majumdar
debapriyo@isical.ac.in
June 2021

Transformer: Motivation

- In spite of the improvements with LSTM and GRU, RNNs improve with the help of attention
  - Long range dependencies need to survive through long sequences
  - With attention, it is easier to focus on any part of the input at any time
- RNNs (GRUs, LSTMs) are more expensive to train (the more complex, the harder)
- RNNs are inherently sequential, cannot exploit parallelism
- Idea: can we just have attention and not use RNNs at all?

- **"Attention is all you need" —** Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin

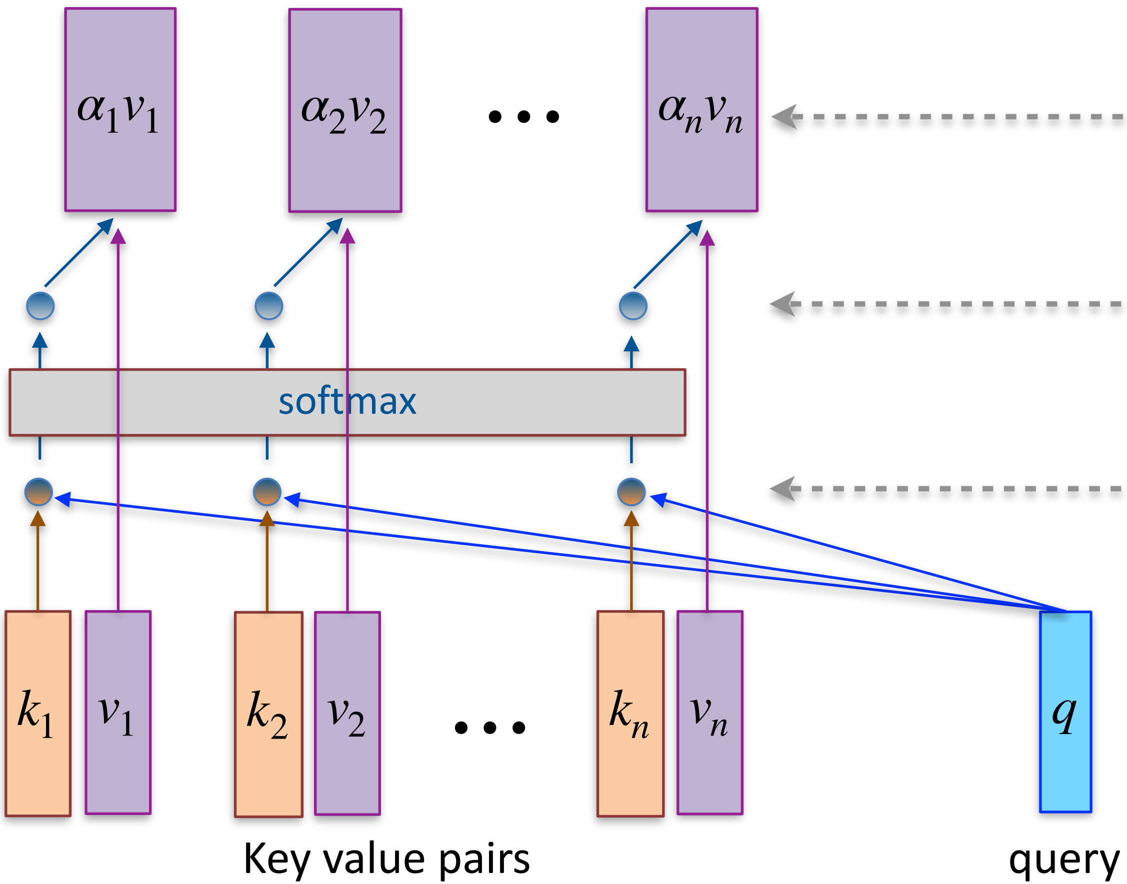
Transformer: architecture at a high level



- Consists of an encoder component and decoder component
- Stack of 6 encoders and 6 decoders
  - Nothing sacred about 6, can be different
- All sublayers produce outputs of dimension $d_{\text{model}} = 512$
- Key points
  - Positional encoding
  - Multi-head self-attention (parallelism)
  - Layer normalization

Picture source:
https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

Recall: Generalized Attention Mechanism



Values weighted by attention

Attention weights α_i

Attention scores (scalar)
Some function $f(q, k_i)$

The dimensions of keys, values and the query may not be the same

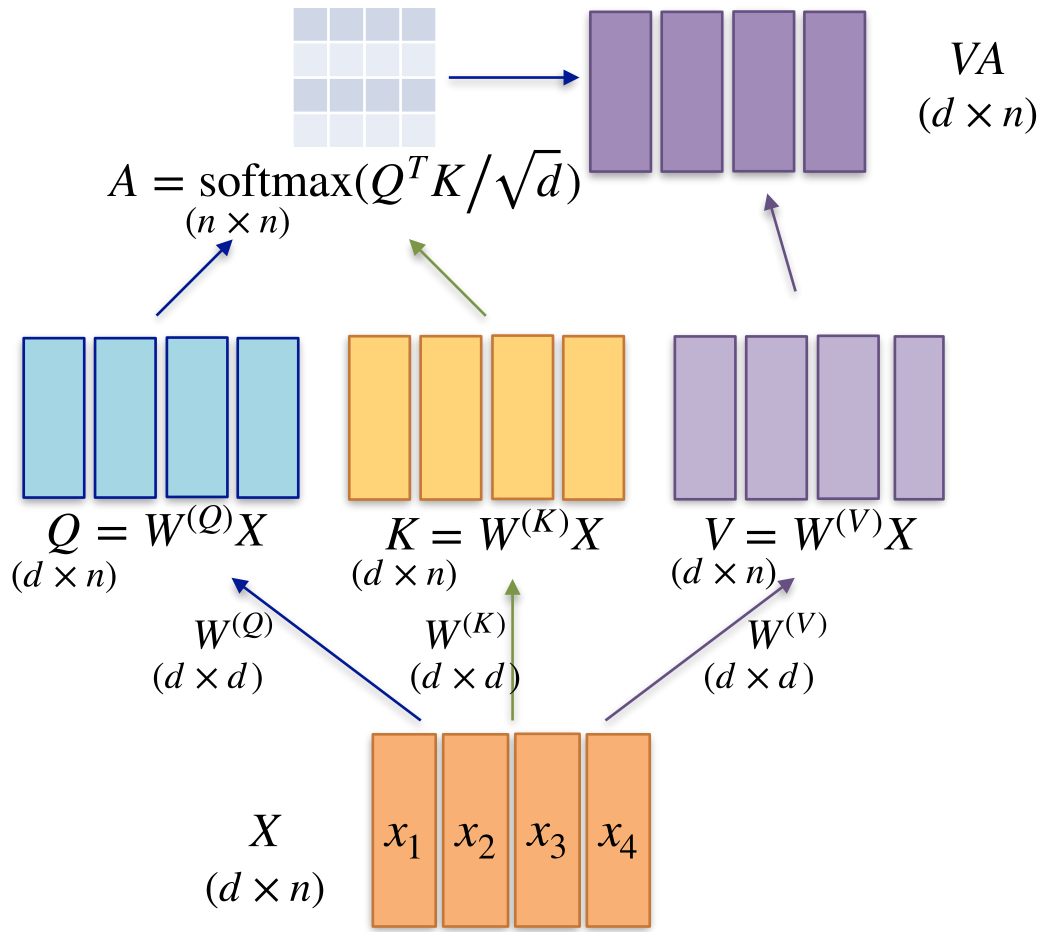
(Keys, values, queries are all vectors)

Self-attention (single-head)

- Input dimension d
- Queries (Q), Keys (K) and values (V) are all obtained from the same set of vectors
- $Q = W^{(Q)}X$, $K = W^{(K)}X$ and $V = W^{(V)}X$, each of dimension d
- Attention probabilities $(n \times n)$ by scaled dot-product attention

$$A = \alpha(Q, K) = \text{softmax}\left(\frac{Q^T K}{\sqrt{d_k}}\right)$$

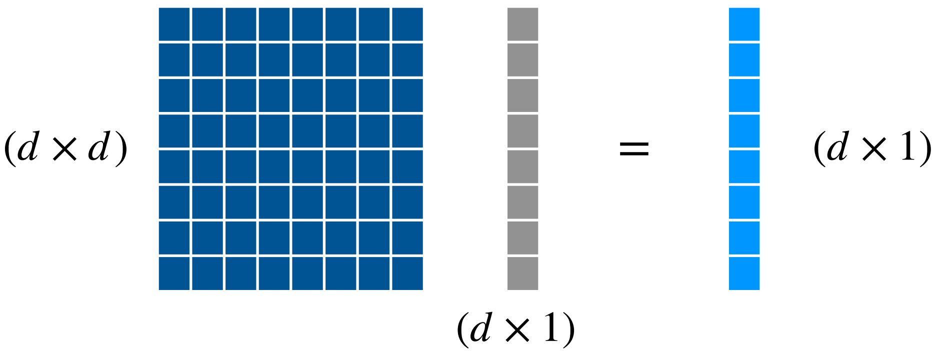
- The values are weighted by the attention probabilities: $\text{Attention}(Q, K, V) = VA$
- Essentially: a function computing $X \mapsto VA$
  - Dimension remains the same
- Parameters: $W^{(Q)}, W^{(K)}, W^{(V)}$
  - Each matrix is $d \times d$
  - Trains the parameters



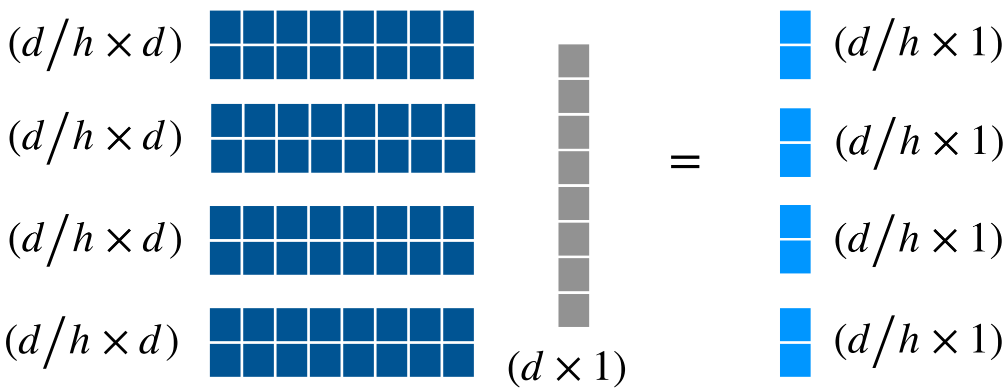
In our example, number of words $n = 4$

Intuition: parallelizing the process with multiple-heads

Intuition using a matrix-vector multiplication



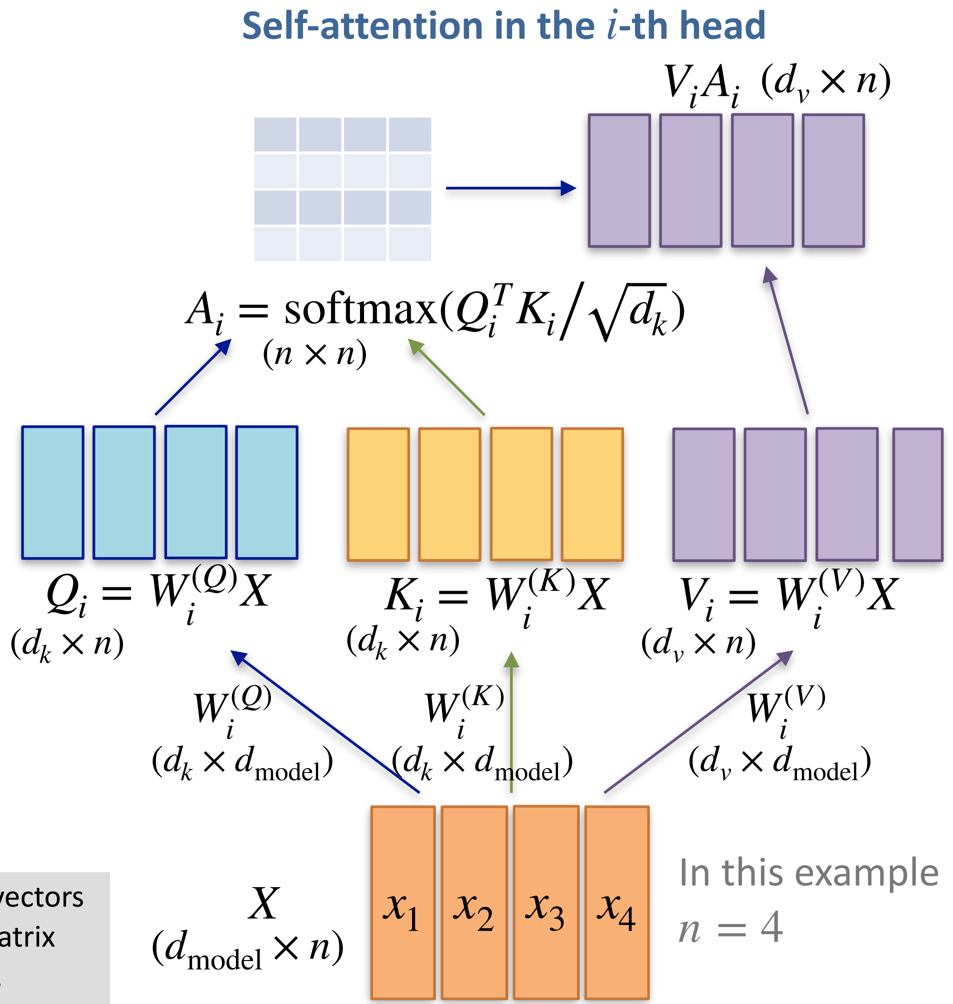
Single head transformation



h computing heads
Can be computed in parallel

Multi-head self attention

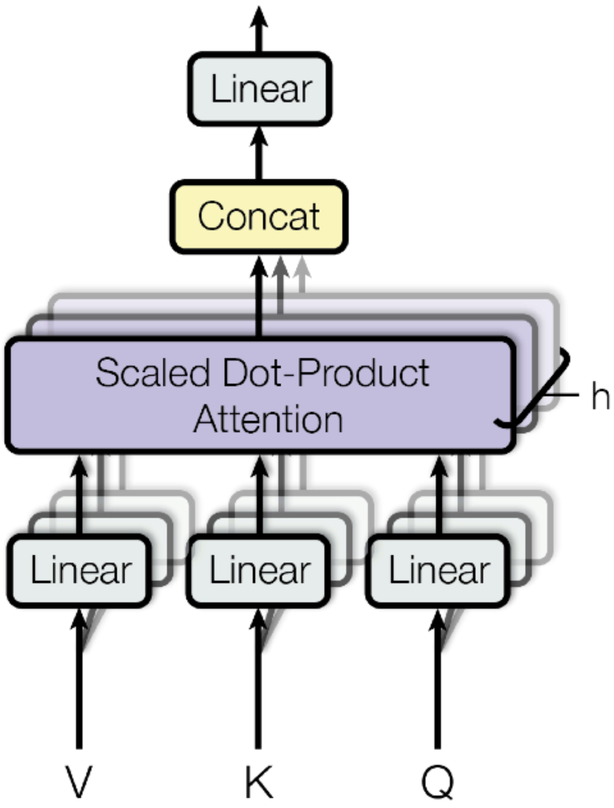
- Divide and parallelize the computation
- Multiheaded: perform attention in parallel, different projection matrices
- Hyperparameters: number of heads $h = 8$, $d_{\text{model}} = 512, d_k = d_v = d_{\text{model}}/h = 64$
- For each head i, train projection matrices (paramters) $W_i^{(Q)}, W_i^{(K)}, W_i^{(V)}$
- Each attention head produces output of dimension d_v
- Concatenating them, obtain: d_{model}



For the sake of consistency, embeddings are considered to be column vectors here (the authors consider them to be row vectors). Accordingly the matrix dimensions have been switched and the formulae have been modified.

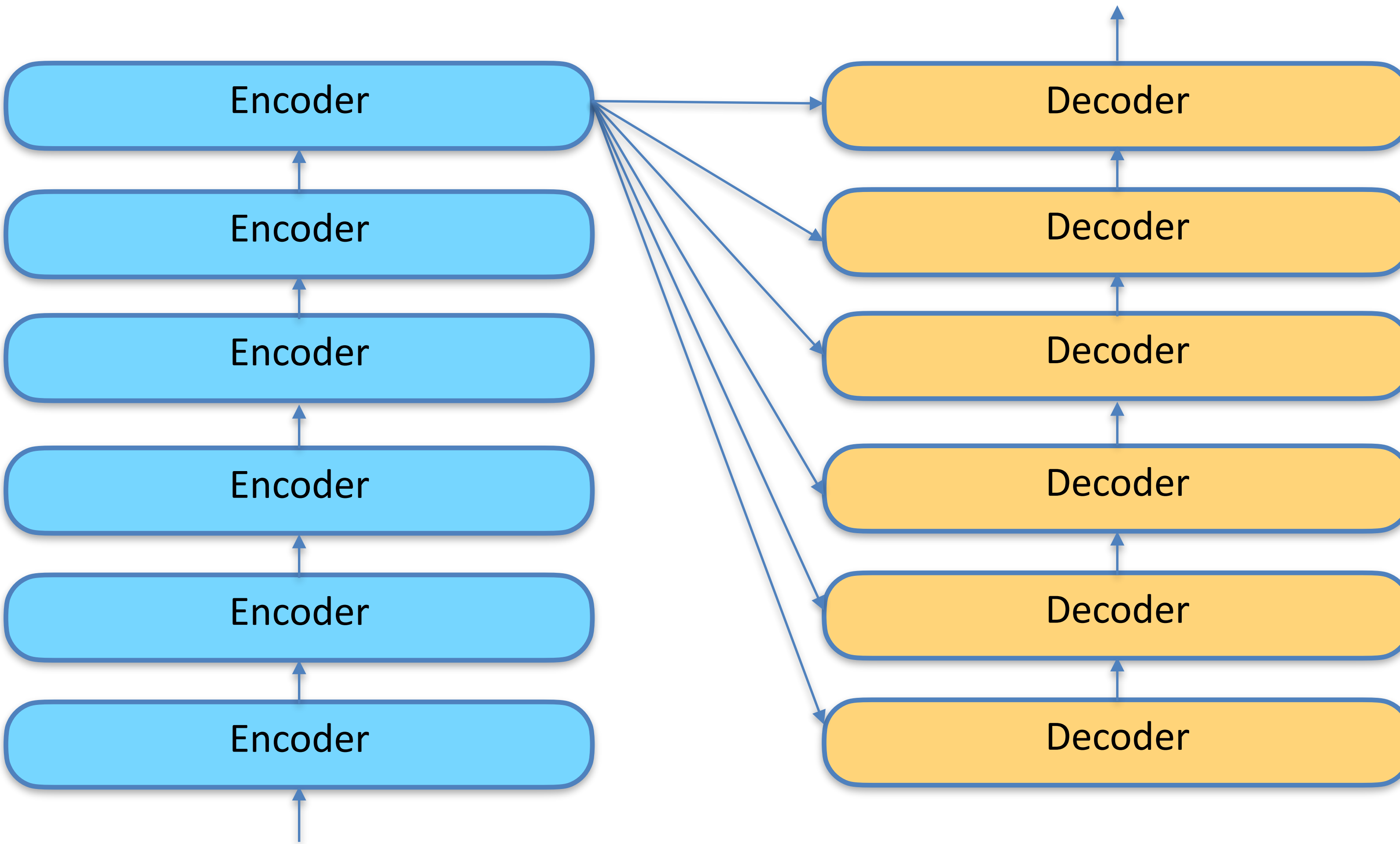
Multi-head attention

- Each attention head produces an output of dimension $d_v \times n$
- A total of h heads, concatenate the output → obtain output of dimension $d_{\text{model}} \times n$
- Computational complexity equivalent to a single attention head with full dimension
- Advantage:
  - Learn different attention representation subspaces at different positions (8 focus points!)
  - Parallelism in training

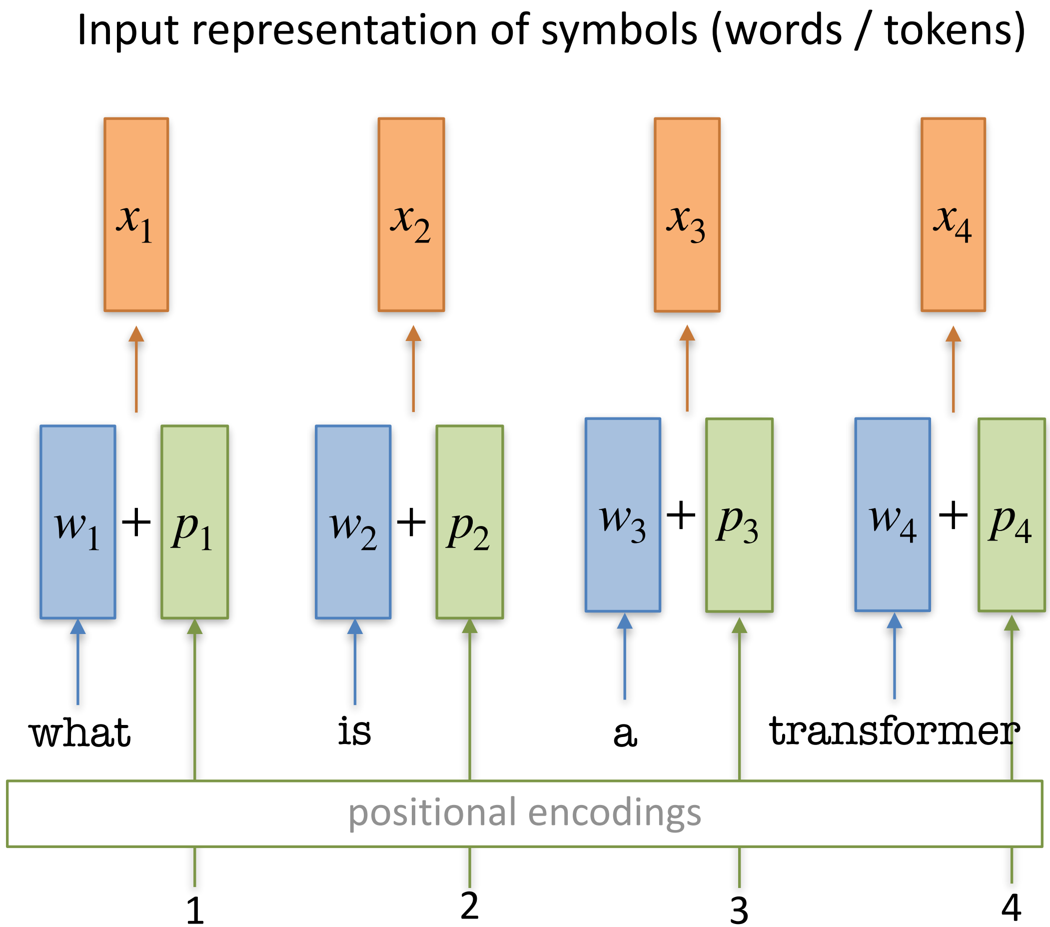


Picture source:
https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

The encoder decoder connections



Input: embedding with positional encoding

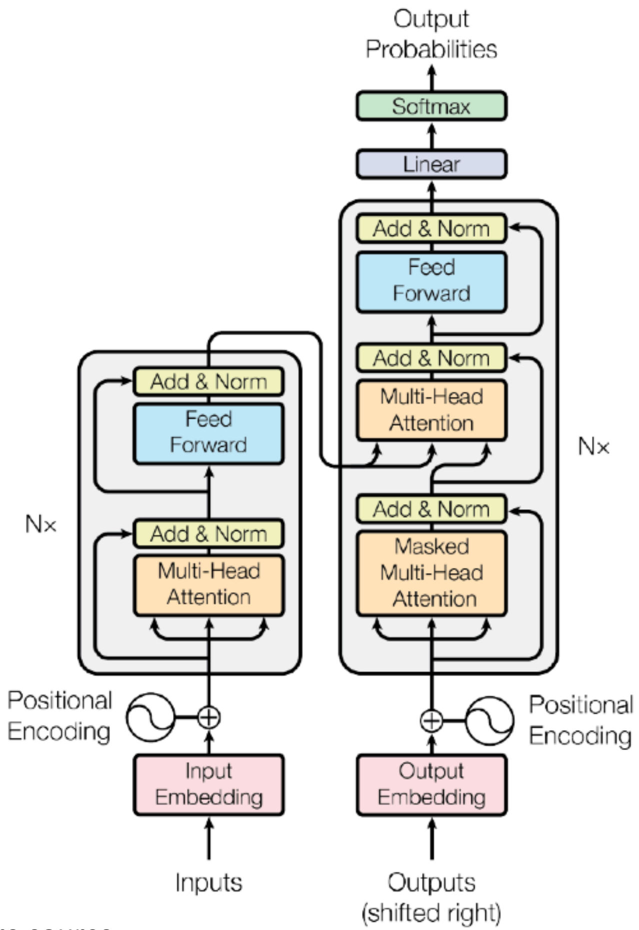


- Positional encodings and input embeddings (for example word embeddings) have same dimension (they are summed)
- Positional encodings can be learned, or fixed
- In transformer, the authors use

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

where pos is the position of the symbol and i is the dimension

Transformer: architecture at a high level



Picture source:
https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

- Each encoder consists of a multi-head attention, followed by a fully connected feedforward layer
- Residual connection: each layer employs a residual connection and layer normalization
- Feedforward layer: two linear transformations with a ReLU in between

$$\mathrm{FFN}(x) = W_2(\max(0, W_1 x + b_1)) + b_2$$

- The feedforward layers on different symbols are independent, trained in parallel
- The decoder is similar, consists of three sub-layers with residual connection and layer normalization for each sub-layer:
  - Multi-head self attention (but mask future positions, so only attend to earlier positions)
  - Multi-head attention over encoder output
  - Fully connection feedforward layer

Transformer: results

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

- Base version (the one we have seen) and the big version ($d_{\text{model}} = 1024$, $h = 16$)
- Essentially outperformed all state-of-the-art (at that point of time) in NMT
- Breakthrough innovation, inspired follow up work that made transfer learning possible (and essentially highly recommended) for all NLP related tasks

Table source: https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf
Please see the original paper for the baselines cited here.

Transfer learning

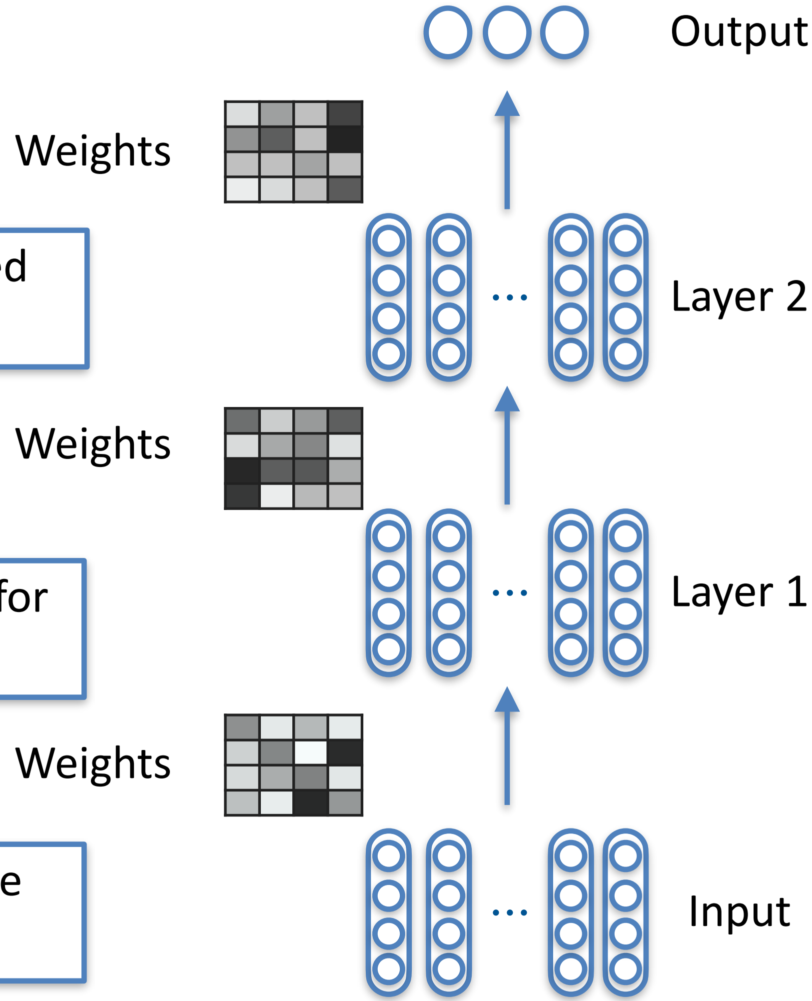
- A basic assumption is supervised learning: the training samples and test samples are drawn from the same *distribution*
- Intuitively: same task on same domain
- Random initialization of a model: training the model from the scratch
  - Is that necessary?
- Human analogy (not all would perfectly resonate)
  - Learn to drive in the city $\Longrightarrow$ can adapt (with little training) in the hills
  - Learn mathematics $\Longrightarrow$ learn computer science relatively easily
  - Learn English $\Longrightarrow$ learn German (compared to those who don’t know English) easily
- Many approaches in transfer learning (very important subfield of research)
  - Freeze initial layers and only **fine - tune** final layers
  - Gradually unfreeze layers, slow down learning rate

How fine-tuning works

Model already trained on some task A

Weights are trained for task A

Want to fine-tune the weights for task B



Start with pre-trained weights

Change the output layer to suit the task B

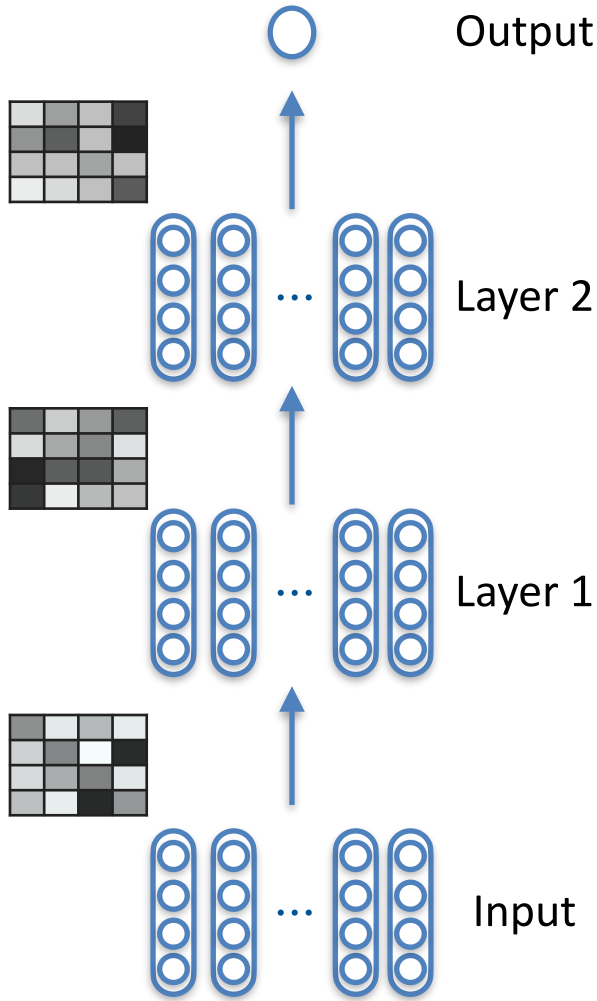
Sometimes, add some layers too

How fine-tuning works

Model already trained on some task A

Weights are trained for task A

Want to fine-tune the weights for task B

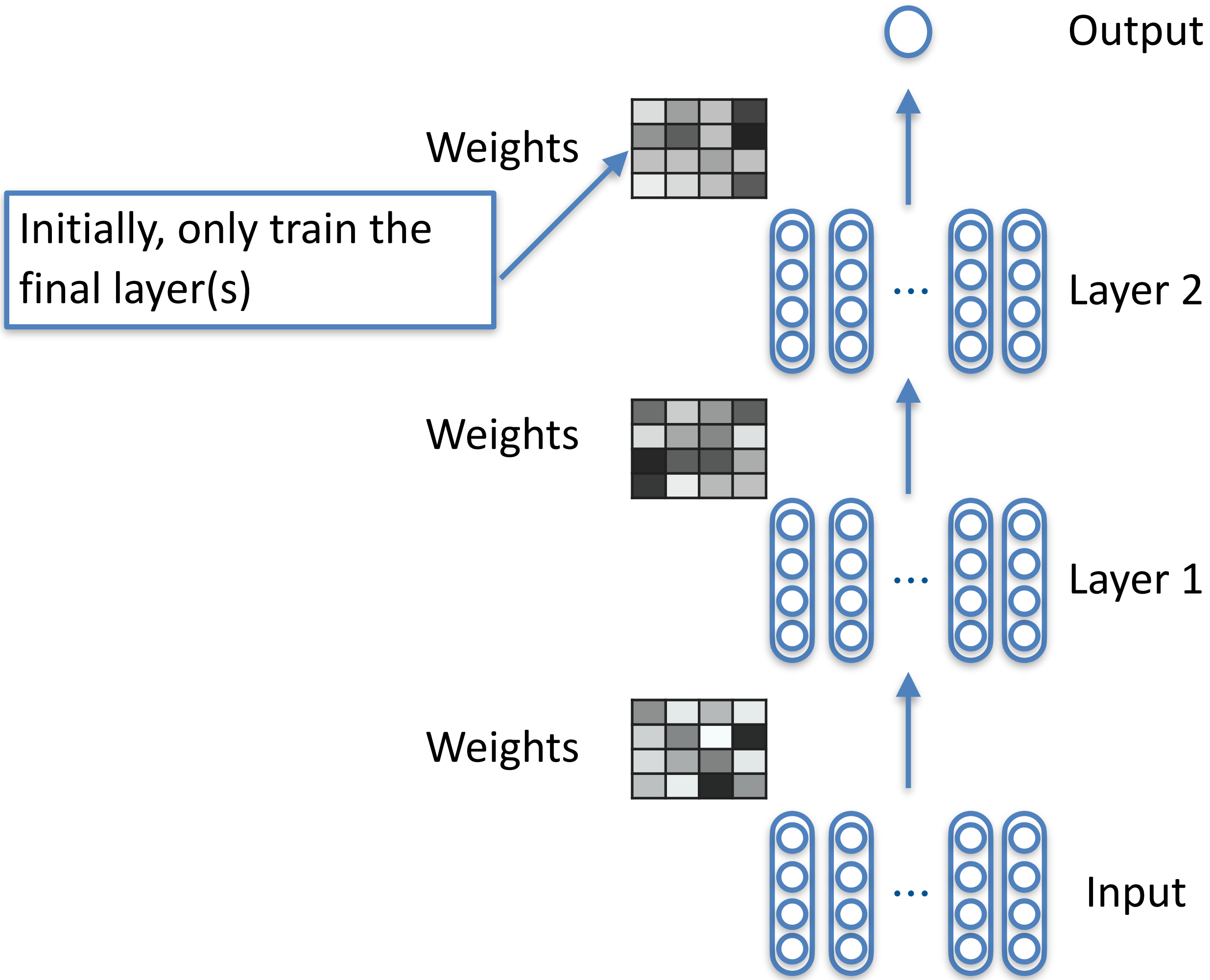


Start with pre-trained weights

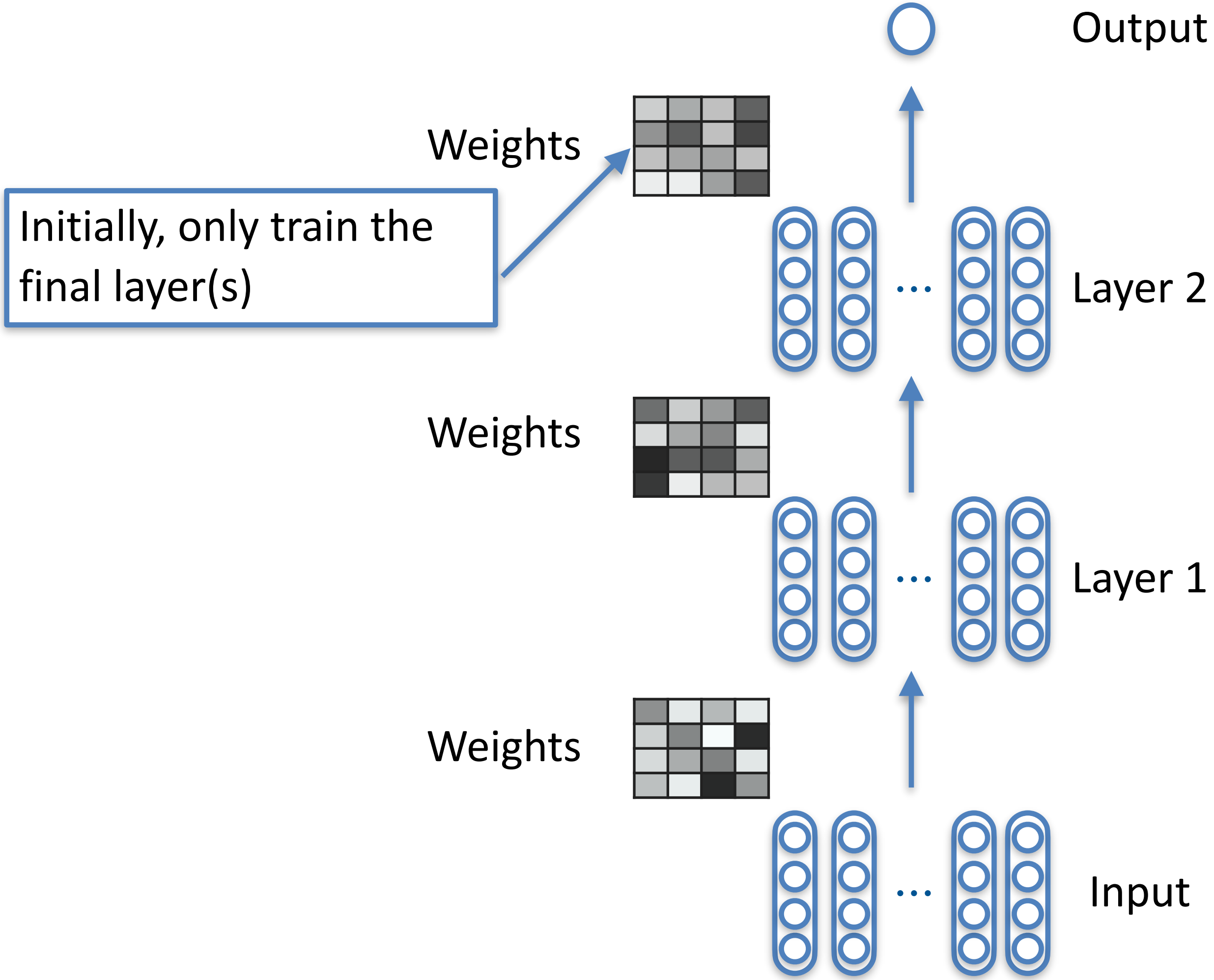
Change the output layer to suit the task B

Sometimes, add some layers too

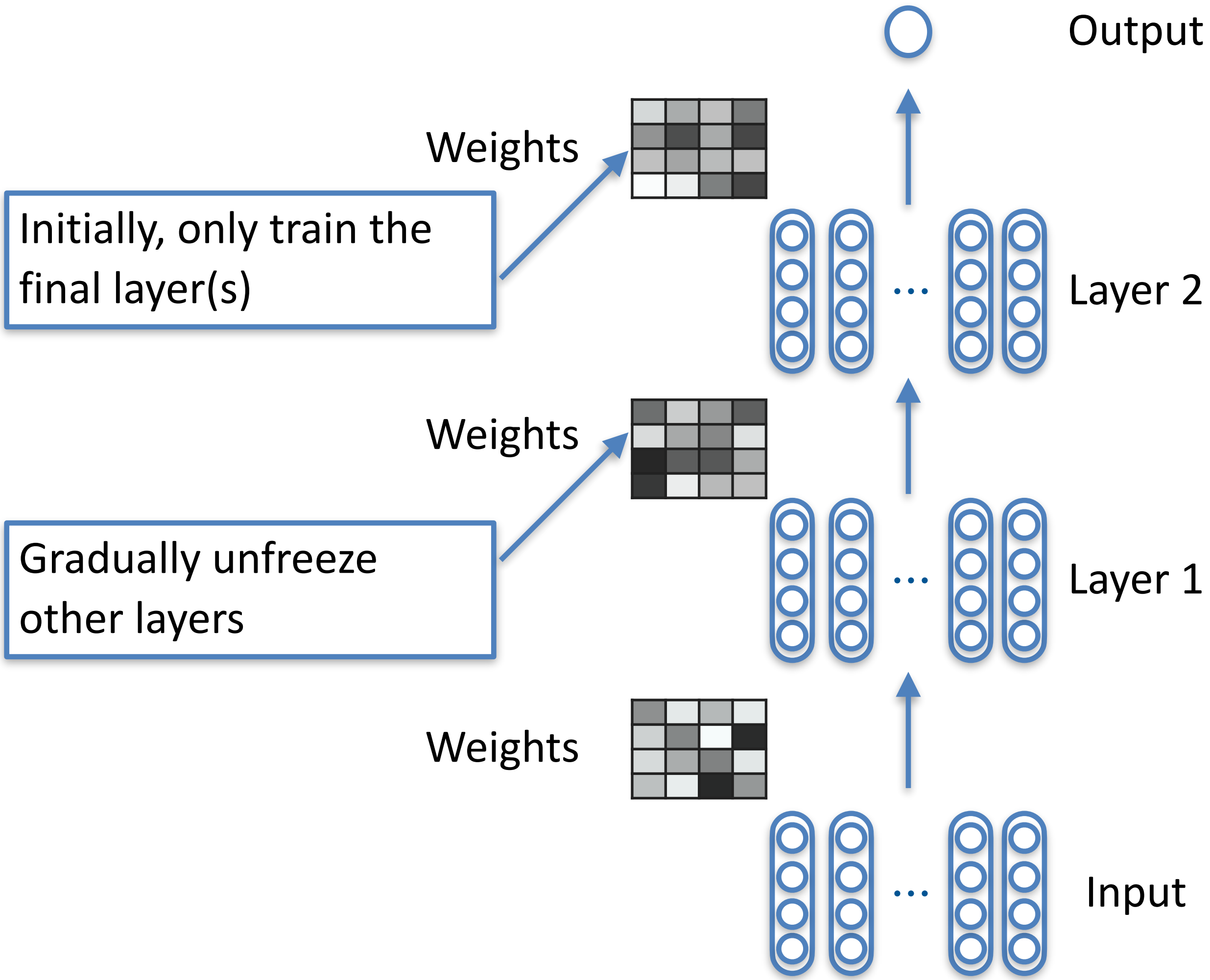
How fine-tuning works



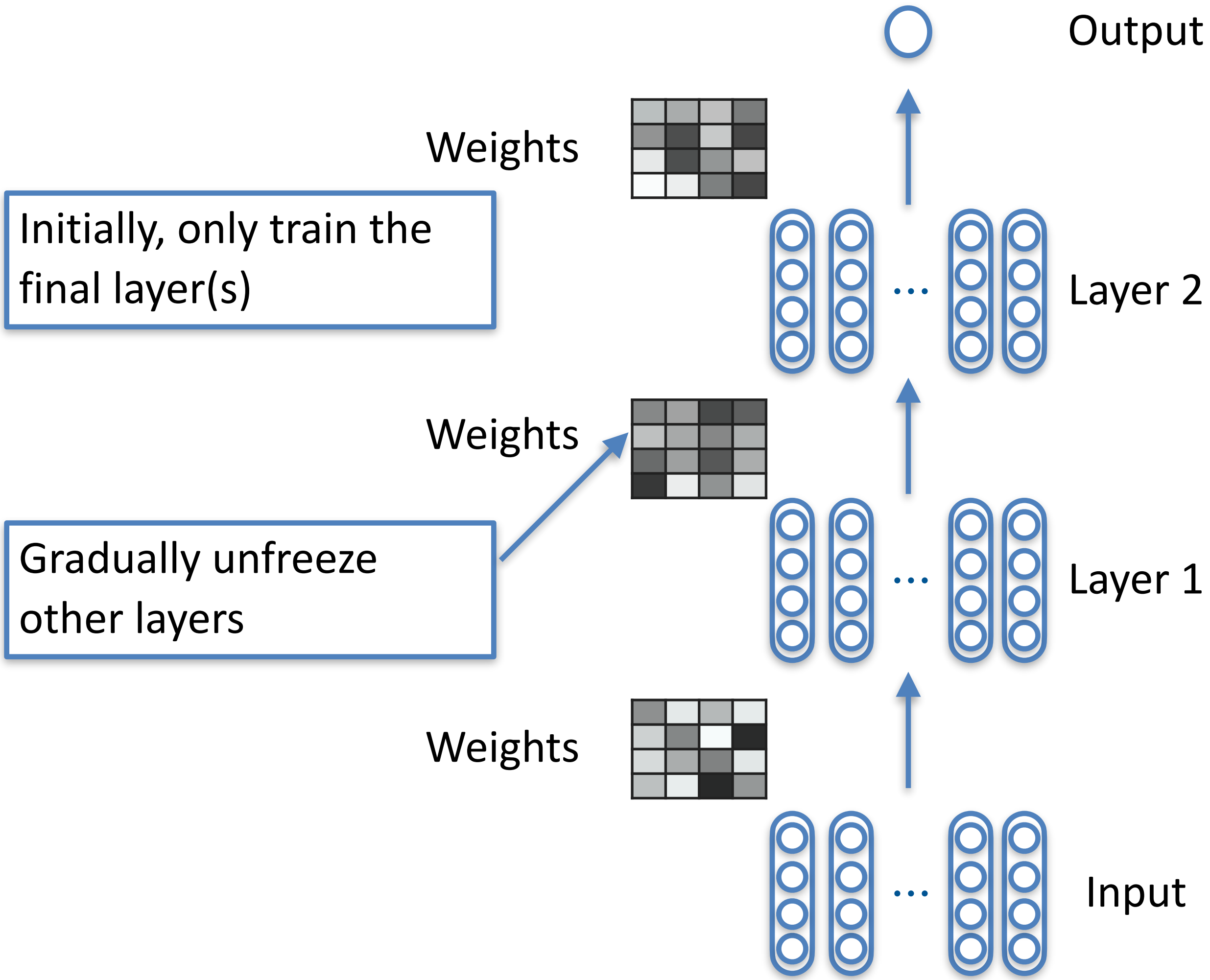
How fine-tuning works



How fine-tuning works



How fine-tuning works



Language Model

A probability distribution over sequences of words $w_1, w_2, \ldots, w_m$

$$P(w_1 w_2 \cdots w_m)$$

Example 1: n-gram language model

$$\begin{aligned} P(w_1 w_2 w_3) &= P(w_3 \mid w_1 w_2) \cdot P(w_1 w_2) \\ &= P(w_3 \mid w_1 w_2) \cdot P(w_2 \mid w_1) \cdot P(w_1) \end{aligned}$$

Example 2: unigram language model ($n = 1$)
In other words, assume independence

$$P(w_j \mid w_i) = P(w_j)$$

$$P(w_1 w_2 w_3) = P(w_3) P(w_2) P(w_1)$$

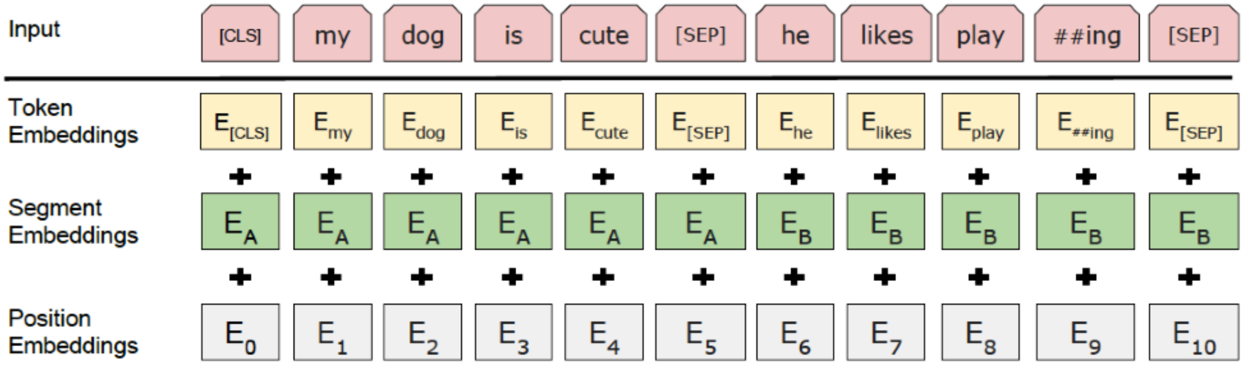
Language models have several use cases in NLP and IR
For example, predict next word, or fill in the blanks

In general, a language model is a model which can predict a target word given the context

BERT: Motivation

- RNN based language models are trained either left-to-right or right-to-left
- Usual application: predict the *next* word
- Idea: Masked LM (Cloze procedure, Taylor, 1953)
  - Fill in the blanks:
    - (a) I approached the ___bank___ for a loan.
    - (b) I need some ___documents___ for a loan.
- RNN to Transformer — change of approach
  - Instead of reading in one direction, read the whole sentence and pay attention to whatever parts necessary at any point of time
  - Better than either of the one directions (or a concatenation of the two directions)
  - Understanding of the context comes for *free*

BERT: input representation

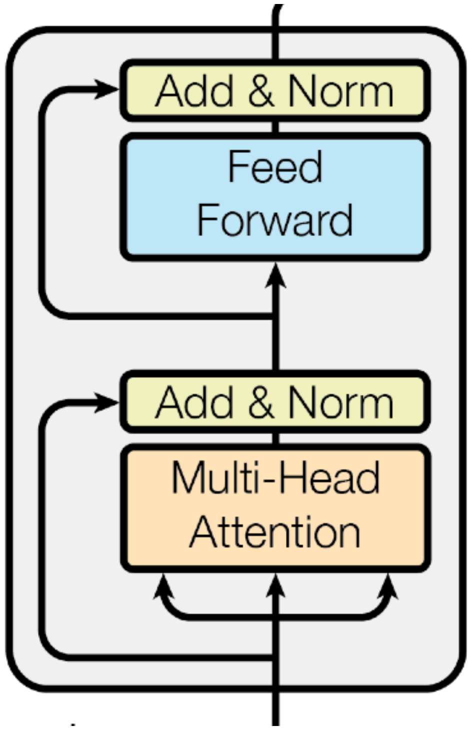


- Input pieces of texts are arbitrary span of contiguous text (may be more than one usual sentence packed together)
- The first token is [CLS], sentence separators [SEP]
- Segment embedding to denote which segment of the text the token is coming from
- Position embedding to denote the position of the token in the whole text
- Token embedding: WordPiece embedding (Wu et al., 2016), with vocabulary size = 30,000
- Each token is the sum of the three embeddings

BERT: encoder

- Transformer encoder
- The input embedding contains segment and positional information
- Multi-headed self-attention
  - Long-distance context has equal opportunity (advantage over LSTM / GRU)
- Layer norm and residual for training deep network
- Feed-forward layer with ReLU
  - Non-linear hierarchical features
- BERT-base: 12-layer, 768 hidden (d_{model}), 12-head
  - For fair comparison with then-state-of-the-art OpenAI GPT
- BERT-large: 24-layer, 1024 hidden, 12-head
- Trained on two unsupervised tasks: *masked LM* and *next sentence prediction*

BERT encoding



BERT
input representation

Masked LM

- Generate training data from unlabelled text by masking out $k\ \%$ ($k = 15$) of the input words by the token [MASK]

  I approached the [MASK] for a loan because [MASK] wanted to buy a car

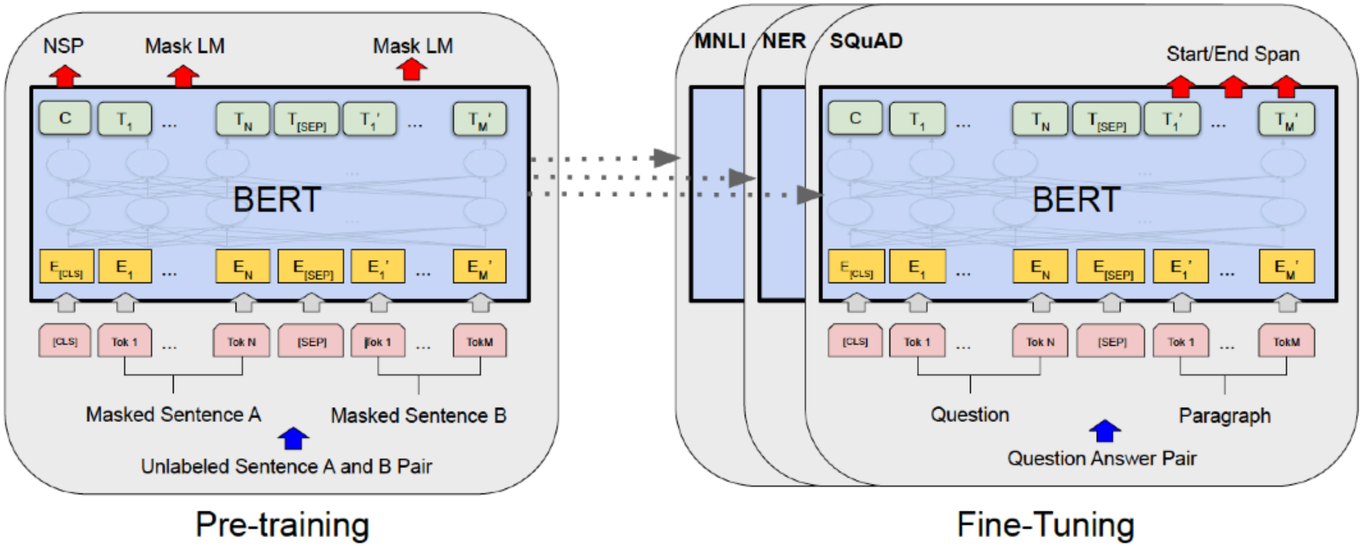
- Tradeoff: too little masking $\rightarrow$ expensive to train, too much making $\rightarrow$ context lost
- Task: predict the masked words
- Details: for other tasks, the model needs to be fined tuned and the token [MASK] won't be present there
- Solution: out of the $k\ \%$ cases to be masked
  - Replace with [MASK] 80% of the time
  - Replace with a random word 10% of the time
  - Keep unchanged 10% of the time
  - While training, compute loss only corresponding to the [MASK] tokens

Next Sentence Prediction

- Learning relationships between sentences: predict whether a sentence B can be the sentence next to sentence A
- Case 1:
  Sentence A: The man went to the store.
  Sentence B: He bought a gallon of milk.
  Label: IsNextSentence
- Case 2
  Sentence A: The man went to the store.
  Sentence B: Penguins are flightless.
  Label: NotNextSentence
- Training data: 50% of the cases, have actual next sentence and the other 50% of the cases, have a random sentence

BERT: fine tuning

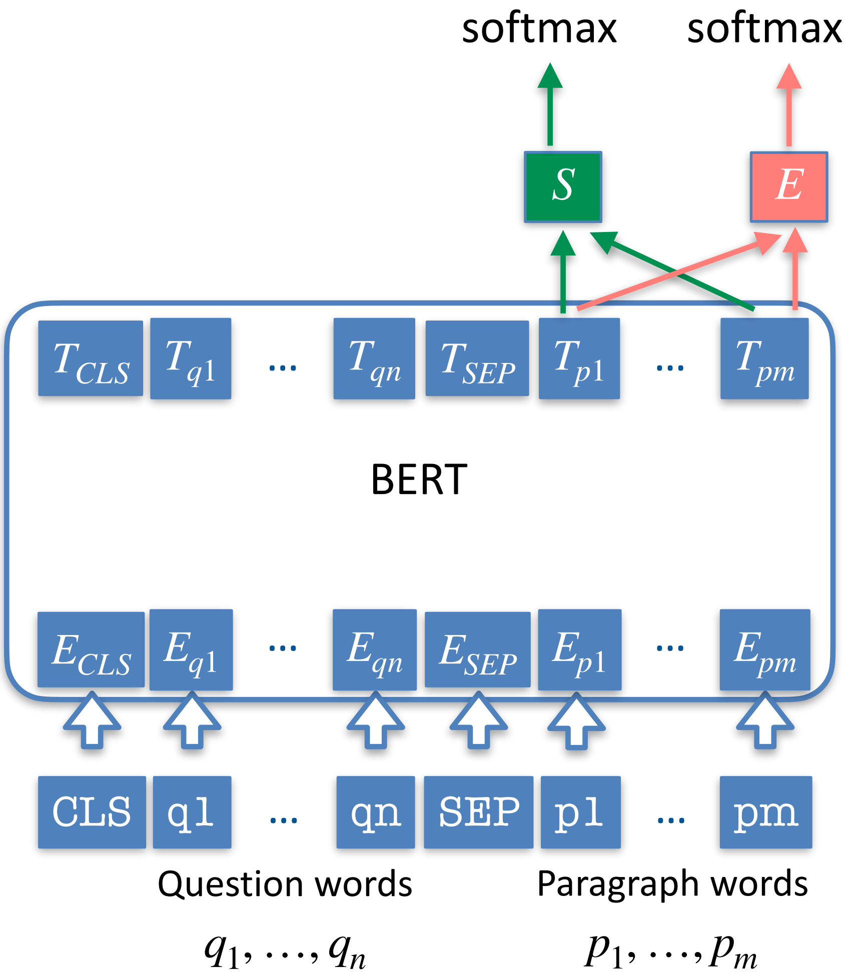
- Pre-training: trained on the two tasks on BooksCorpus (800M words) and English Wikipedia (2,500M words)
- Fine-tuning: add an appropriate output layer depending on the task
- Tasks involving text-pairs (question - answer, sentence pair in paraphrasing) are analogous to *next sentence prediction*
- Usually fine-tuning takes significantly lesser time than pre-training



BERT on SQuAD 1.1

- SQuAD 1.1: 100k crowdsourced QA pairs
  - Task: given a question and a passage from Wikipedia containing the answer, find the answer text span in the passage
- Question and paragraph packed together to construct the input representation
- Shallow neural network at the output layer: **start** and **end** vectors (parameters **S** and **E**) to be learnt
- Probability of a token T_{pi} being the start (or end) are

$$P_i^S = \frac{e^{S \cdot T_{pi}}}{\sum_{j=1}^{m} e^{S \cdot T_{pj}}}, P_i^E = \frac{e^{E \cdot T_{pi}}}{\sum_{j=1}^{m} e^{E \cdot T_{pj}}}$$



BERT Results on QA

Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar et al. '16)	82.304	91.221
1 Oct 05, 2018	BERT (ensemble) *Google AI Language* https://arxiv.org/abs/1810.04805	**87.433**	**93.160**
2 Oct 05, 2018	BERT (single model) *Google AI Language* https://arxiv.org/abs/1810.04805	85.083	91.835
2 Sep 26, 2018	nlnet (ensemble) *Microsoft Research Asia*	85.954	91.677
5 Sep 09, 2018	nlnet (single model) *Microsoft Research Asia*	83.468	90.133
3 Jul 11, 2018	QANet (ensemble) *Google Brain & CMU*	84.454	90.490

- SQuAD 1.1 QA dataset
- 100,000+ question answer pairs on 500+ articles: https://rajpurkar.github.io/SQuAD-explorer/
- BERT single model did very well in SQuAD 1.1, only a BERT emsemble beat it

BERT: GLUE benchmark results

GLUE Benchmark 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	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

Sentence pairs: *entailment, contradition or neutral?*

Quora question pairs: are they equivalent?

SQaAD data: question - paragraph pairs: does the paragraph contain the correct answer?

Binary single sentence sentiment classification from movie reviews

Is this English sentence linguistically acceptable?

Semantic textual similarity (1-5) between a pair of sentences

Are two sentences paraphrases of each other?

Similar to MNLI, but with much less training data

References

- Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In *Advances in neural information processing systems*, pp. 5998-6008. 2017.
- Alammar, Jay (2018). “The Illustrated Transformer” [Blog post]. Retrieved from jalammar.github.io/illustrated-transformer/
- Lio Dirac. “LSTM is dead. Long Live Transformers.” Seattle Applied Deep Learning: www.youtube.com/watch?v=S27pHKBEp30
- Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).
- Khalid, Samia. “BERT Explained: A Complete Guide with Theory and Tutorial.” [Blog post] towardsml.com/2019/09/17/bert-explained-a-complete-guide-with-theory-and-tutorial/
- Slides by Jacob Devlin on BERT: https://nlp.stanford.edu/seminar/details/jdevlin.pdf
- Zhang, Zhuosheng, Junjie Yang, and Hai Zhao. “Retrospective reader for machine reading comprehension.” *arXiv preprint arXiv:2001.09694* (2020).