Lecture 16

Transformers & Attention

LV
Guest Lecture

Lecture Notes V – Residual and Convolutional Neural Networks + Transformer

Marina Meilă
mmp@uwaterloo.ca

With Thanks to Pascal Poupart & Gautam Kamath
Cheriton School of Computer Science
University of Waterloo

February 8, 2026

Convolutional networks (Convnets) ✓

Residual networks (Resnets) ✓

Some (convolutional) neural network breakthroughs

Transformer networks ←

Reading HTF Ch.: 11.3 Neural networks, Murphy Ch.: (16.5 neural nets), Bach Ch.: –, Deep Learning Book (Goodfellow, Bengio, Courville) 6.1-4, ResNet 7.6, ConvNet 9., Autoencoders 14.1, Dive Into Deep Learning 4.1-4.3.

Transformer networks: Why we need attention

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

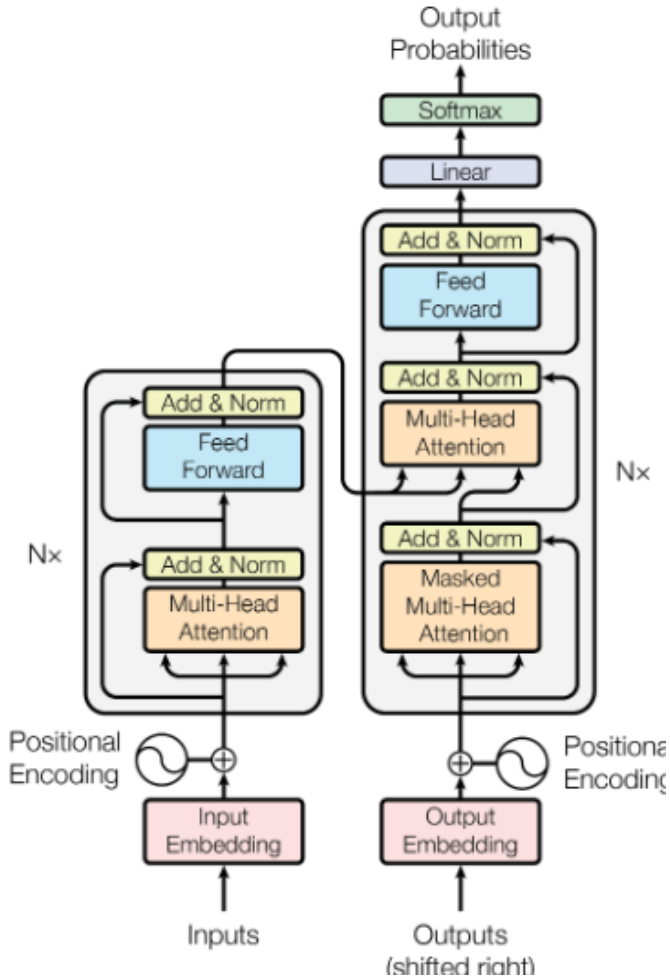
We propose a new simple network architecture, the Transformer,based solely on attention mechanisms, dispensing with recurrence and convolutionsentirely.

Wir schlagen eine neue einfache Netzwerkarchitektur, den Transformer, vor, die ausschließlich auf Aufmerksamkeitsmechanismen basiert und auf Wiederholung und Faltung vollständig verzichtet.

我们提出了一种新的简单的网络架构--Transformer，完全基于注意力机制，摒弃了递归和卷积。

- mapping sequences to sequences (structured prediction)
- both long and short range dependencies
- range depends on input sequence

Basic architecture



- inputs $x_1, x_2, \ldots$, outputs $y_1, y_2, \ldots$ from discrete set (e.g. words in English, Chinese)
- continuous internal representations
- **embedding** modules map input or output space to continuous representations (prelearned)

- recurrence/auto-regression y_t depends on $x_{1:t+k}$ and $y_{1:t-1}$

- **encoder, decoder, encoder-decoder** modules (which use attention)

How to implement attention

- **queries, keys** and **values**
- all **learned**
- Idea: query q matches key k results in selecting the corresponding value v
- q depends on current context, k depends on v
- $q, k \in \mathbb{R}^{d_k}$

- Q, K, V matrices of queries, keys, values

$$A(Q, K) = \text{softmax}(\frac{1}{\sqrt{d_k}} QK^T) \tag{4}$$

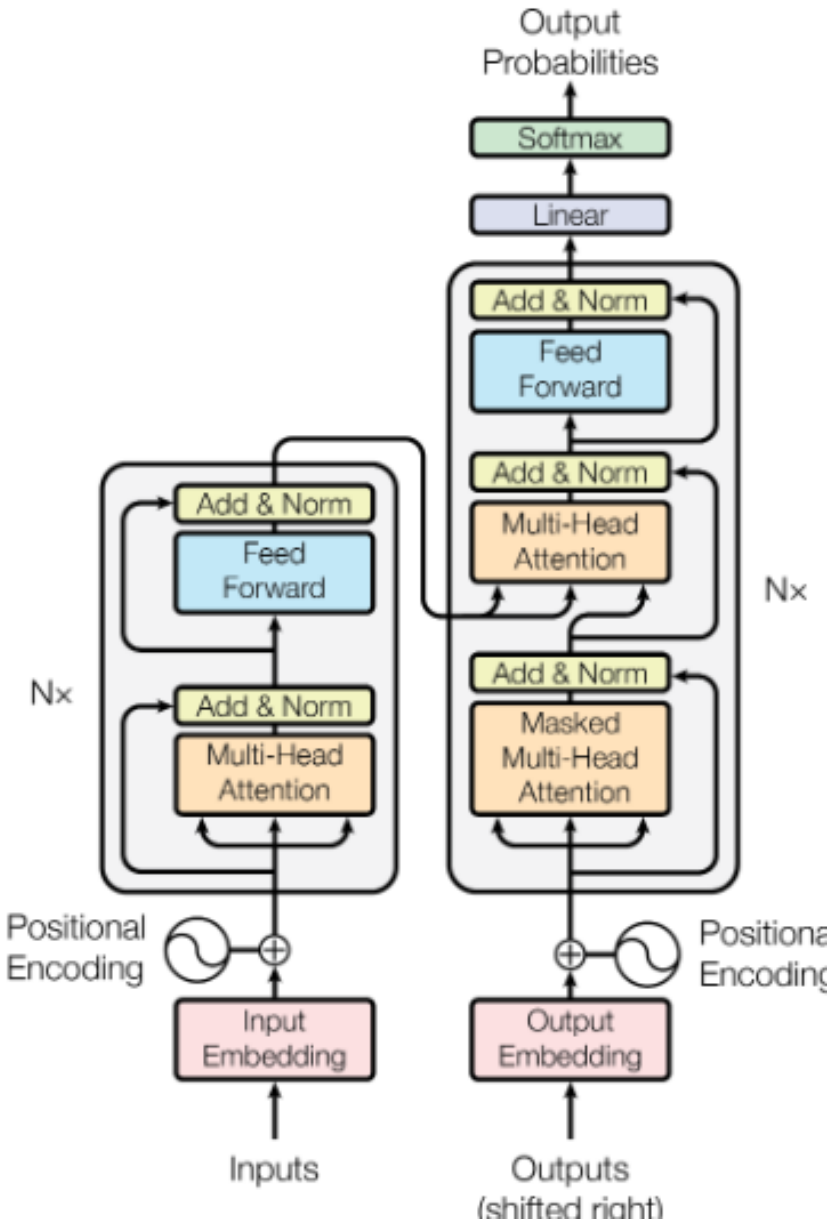
- $A_{q:}$ selects value v for the best matching key for each q

Transformer architecture

- D,E,DE modules: each have $N = 6$ layers of Attention + Feed-forward (FFW) networks of same $d = 512$
- FFW, A are ResNets
- FFW is $W_2 \max(W_1 x, 0)$, $W_{1,2}$ with identical rows
- A is **multihead attention**
  - $h = 8$ parallel attention layers, concatenated

- advantages – implements long distance dependencies with fixed (small) number layers, and parallel computations

Attention mechanism in Transformer



- encoder-decoder
  - queries from previous decode layer
  - keys, values from current encoder output
- encoder – self-attention (=previous encoder layer)
- decoder
  - self-attention, **masked**
  - only from outputs before current step

Lecture 16: Attention, Transformers and Structured State Space Sequence (S4) CS480/680 Intro to Machine Learning

2023-3-9

THANKS!

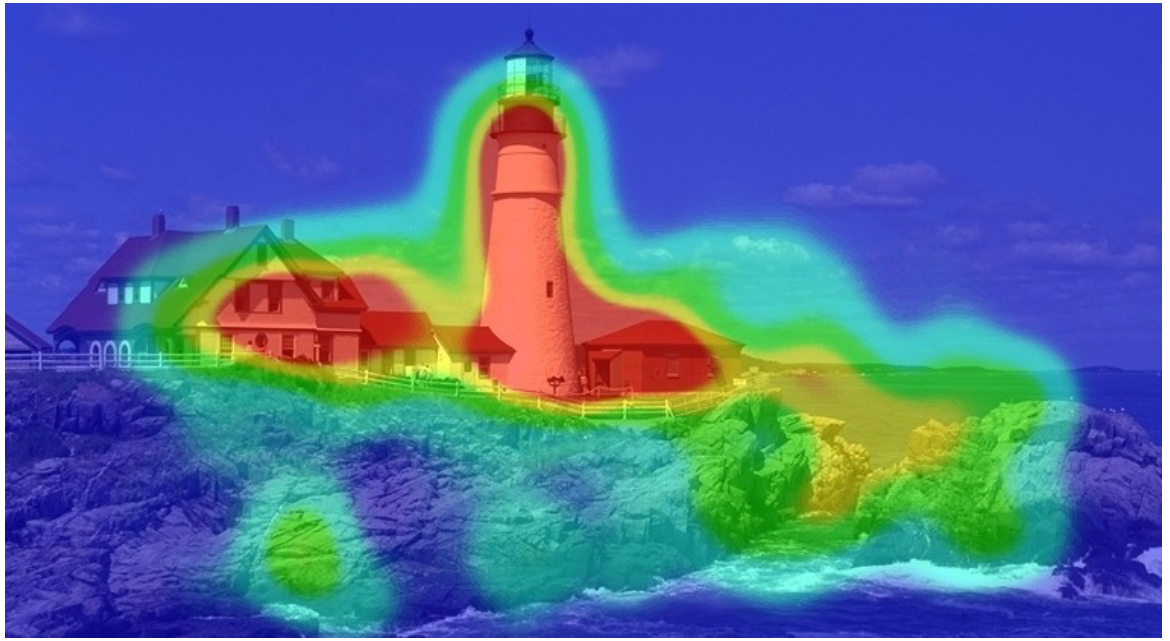
Pascal Poupart
David R. Cheriton School of Computer Science

UNIVERSITY OF WATERLOO

Attention

Mixture of experts

- Attention in Computer Vision
  - 2014: Attention used to highlight important parts of an image that contribute to a desired output



- Attention in NLP
  - 2015: Aligned machine translation
  - 2017: Language modeling with **Transformer networks**

Sequence Modeling

input tokens $x = [x_1 \dots x_T]^T$ sequence $x_t \in X$

output tokens $y = [y_1 \dots y_{T'}]^T$ $y_t \in Y$

Challenges with RNNs

- Long range dependencies
- Gradient vanishing and explosion
- Large # of training steps
- Recurrence prevents parallel computation

Recurrent

Transformer Networks

Σ discrete sets

- Facilitate long range dependencies
- No gradient vanishing and explosion
- Fewer training steps
- No recurrence that facilitate parallel computation

not so much

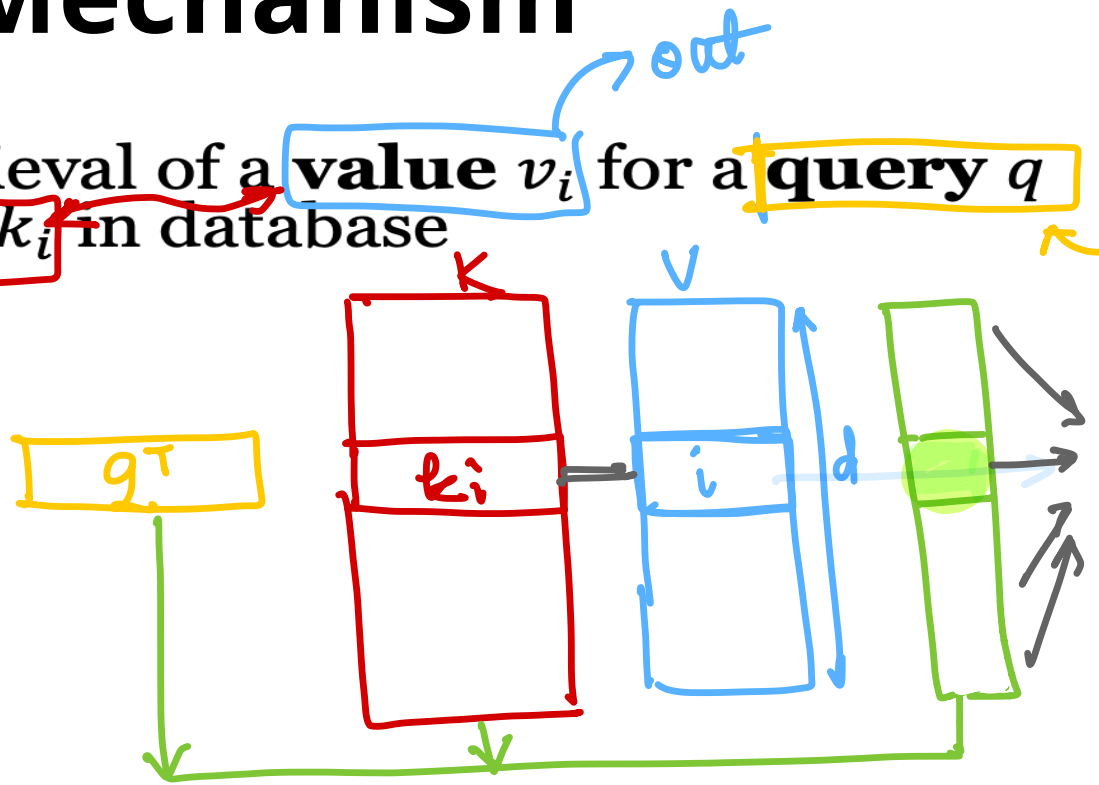
Ex: machine translation x = English words y = chinese chars y

Pb: predict y_t ($x_{1:T}$, $y_{1:t-1}$) context

Attention Mechanism

φ = softmax

- Mimics the retrieval of a **value** v_i for a **query** q based on a **key** k_i in database
- Picture



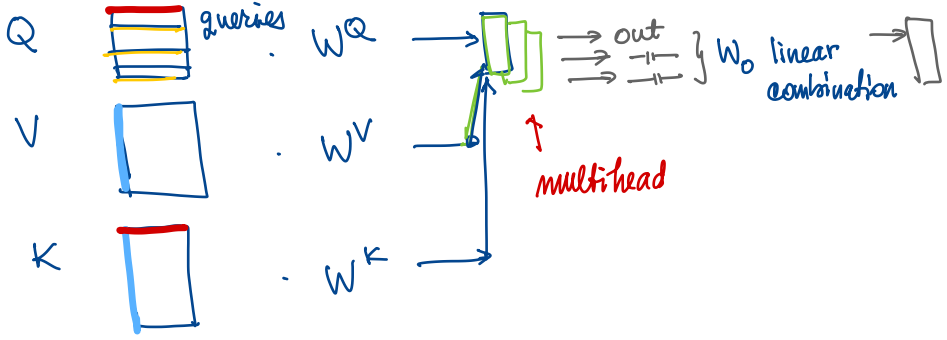
$\pi_{1:d} = \varphi(q^T k_{1:d})$

$$attention(q, \boldsymbol{k}, \boldsymbol{v}) = \sum_i similarity(q, k_i) \times v_i$$

$\varphi = \text{softmax}(q^T k_i,\ i = 1:d) \in (0,1)^d$

$\pi = [.95 \quad .01 \quad .01 \ldots .01 \quad 0 \quad 0 \ldots] \Rightarrow out = v_1 \times .95 + .01 \times v_2 + \ldots$

Multi head ~~Attention~~



Ex: Is there red car

V_i =  k_i = {"car"}

k_i = "color"

q = "car"

q' = "color"

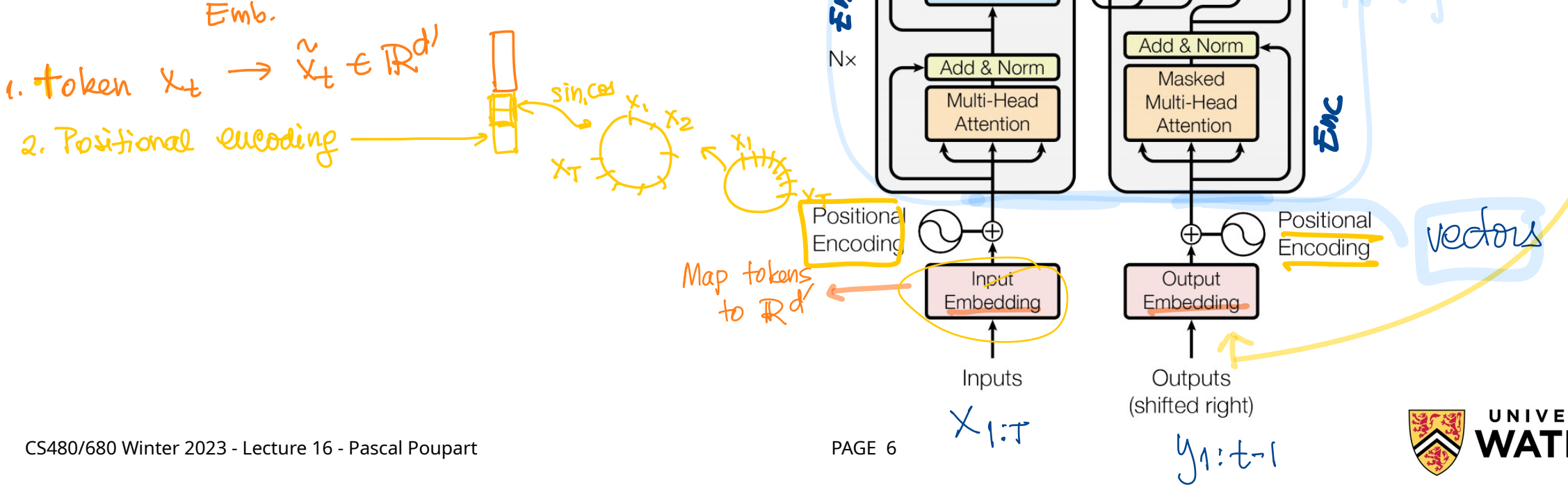
Attention Mechanism (Neural Architecture)

-

- Example: machine translation
  - Query: s_{i-1} (hidden vector for $i-1^{th}$ output word)
  - Key: h_j (hidden vector for j^{th} input word)
  - Value: h_j (hidden vector for j^{th} input word)

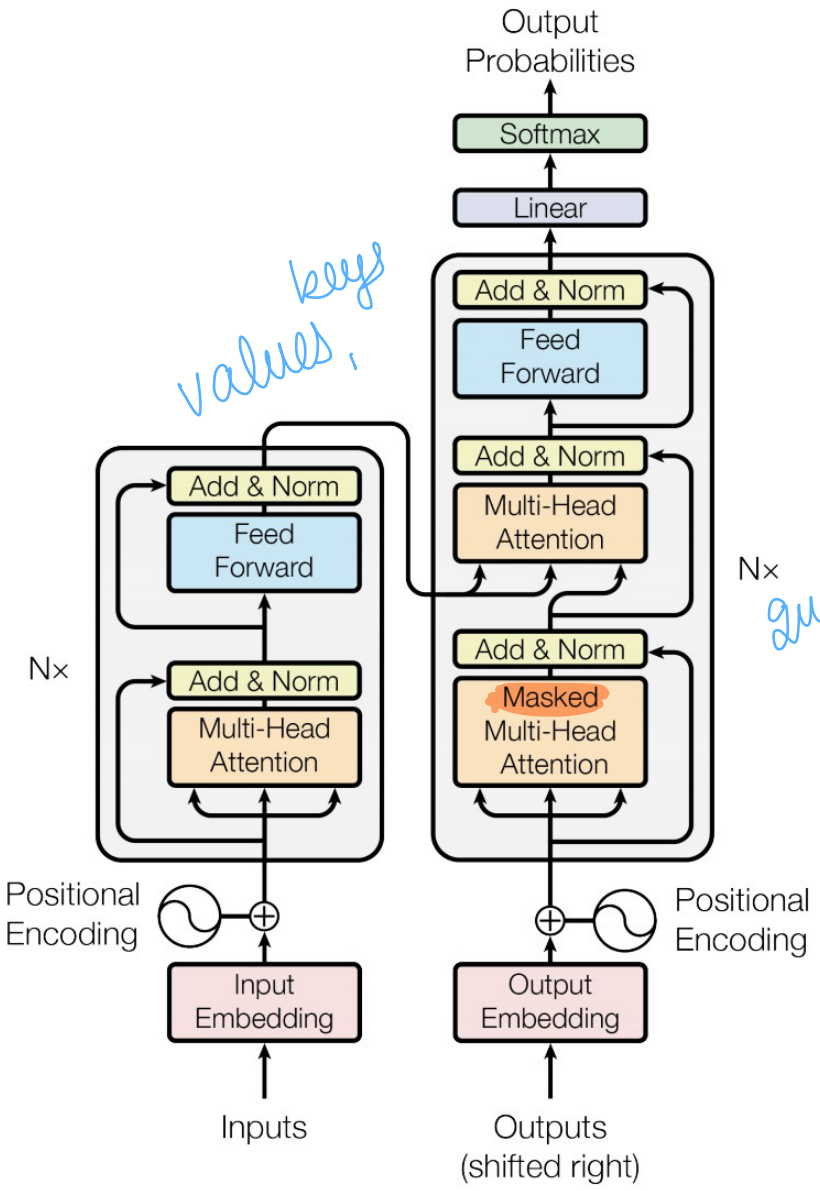
Transformer Networ

- Vaswani et al., (2017) Attention is all you need.
- Encoder-decoder based on attention (no recurrence)



Transformer Networ

- Vaswani et al., (2017) Attention is all you need.
- Encoder-decoder based on attention (no recurrence)



Multihead attention

- Multihead attention: compute multiple attentions per query with different weights

$$multihead(Q,K,V) = W^O concat(head_1, head_2, \dots, head_h)$$
$$head_i = attention\left(W_i^Q Q, W_i^K K, W_i^V V\right)$$
$$attention(Q,K,V) = softmax\left(\frac{Q^T K}{\sqrt{d_k}}\right)V$$

Masked Multi-head attention

- Masked multi-head attention: multi-head where some values are masked (i.e., probabilities of masked values are nullified to prevent them from being selected).
- When decoding, an output value should only depend on previous outputs (not future outputs). Hence we mask future outputs.

$$attention(Q, K, V) = softmax\left(\frac{Q^T K}{\sqrt{d_k}}\right)V$$

$$maskedAttention(Q, K, V) = softmax\left(\frac{Q^T K + M}{\sqrt{d_k}}\right)V$$

where M is a mask matrix of 0's and $-\infty$'s

Other layers

- Layer normalization:
  - Normalize values in each layer to have 0 mean and 1 variance
  - For each hidden unit h_i compute $h_i \leftarrow \frac{g}{\sigma}(h_i - \mu)$

    where g is a variable, $\mu = \frac{1}{H}\sum_{i=1}^{H} h_i$ and $\sigma = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(h_i - \mu)^2}$
  - This reduces "covariate shift" (i.e., gradient dependencies between each layer) and therefore fewer training iterations are needed

- Positional embedding (embedding to distinguish each position):

$$PE_{position,2i} = \sin(position/10000^{2i/d})$$
$$PE_{position,2i+1} = \cos(position/10000^{2i/d})$$

Comparison

- Attention reduces sequential operations and maximum path length, which facilitates long range dependencies

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)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

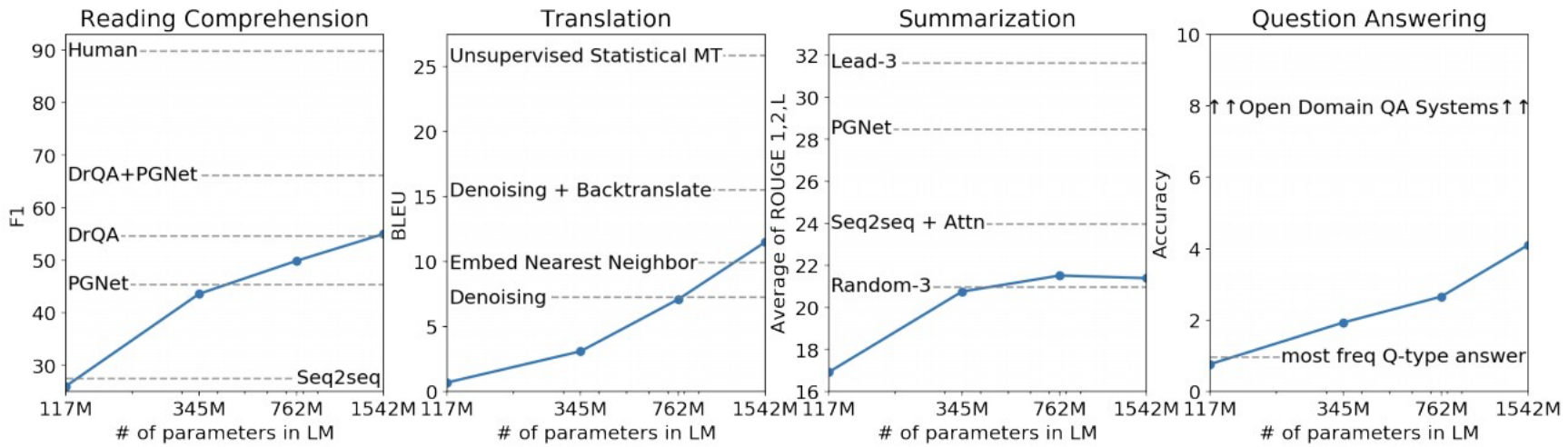
Results

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	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.0**	$2.3 \cdot 10^{19}$	

GPT and GPT-2

- Radford et al., (2018) Language models are unsupervised multitask learners
  - Decoder transformer that predicts next word based on previous words by computing $P(x_t|x_{1..t-1})$
  - SOTA in "zero-shot" setting for 7/8 language tasks (where zero-shot means no task training, only unsupervised language modeling)



BERT (Bidirectional Encoder Representations from Transformers)

- Devlin et al., (2019) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
  - Decoder transformer that predicts a missing word based on surrounding words by computing $P(x_t|x_{1..t-1,t+1..T})$
  - Mask missing word with masked multi-head attention
  - Improved state of the art on 11 tasks

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
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**

Limitation

- Transformers scale quadratically with sequence length
  - In practice, sequence length often limited to 512 tokens

- How can we process long sequences?
  - Hierarchy of transformers (i.e., words→sentences→documents→corpus)
  - Approximate transformers (i.e., longformer, reformer, performer, etc.)
  - **Structured State Space Sequence (S4) model**

- S4: Very recent approach (Gu, Goel & Re, ICLR 2022)
  - Potential to displace transformers
  - S4 achieved state of the art on Long Range Arena benchmark
  - Scales linearly with sequence length