Transformer

Apr 9th, 2019

Applied Deep Learning

YUN-NUNG (VIVIAN) CHEN HTTP://ADL.MIULAB.TW

國立臺灣大學
National Taiwan University

Slides credited from Manning, Vaswani & Huang

Representations of Variable Length Data

Input: word sequence, image pixels, audio signal, click logs

Property: continuity, temporal, importance distribution

Example

- Basic combination: average, sum
- Neural combination: network architectures should consider input domain properties
  - CNN (convolutional neural network)
  - RNN (recurrent neural network): temporal information

Network architectures should consider the input domain properties

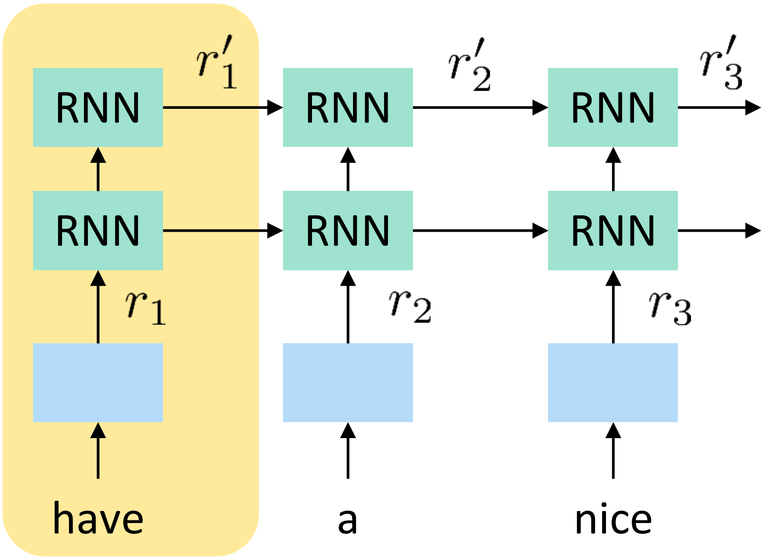
Recurrent Neural Networks

Learning variable-length representations

- Fit for sentences and sequences of values

Sequential computation makes parallelization difficult

No explicit modeling of long and short range dependencies

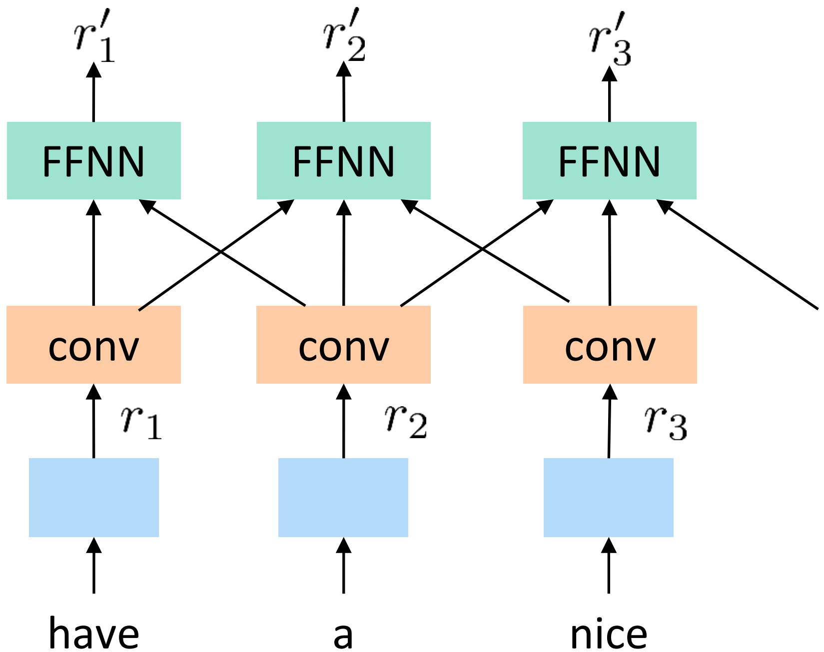


Convolutional Neural Networks

Easy to parallelize

Exploit local dependencies

- **Long-distance** dependencies require many layers



Attention

Encoder-decoder model is important in NMT

RNNs need **attention mechanism** to handle long dependencies

Attention allows us to access any state

Using attention to replace recurrence architectures

Dot-Product Attention

Input: a query q and a set of key-value (k-v) pairs to an output

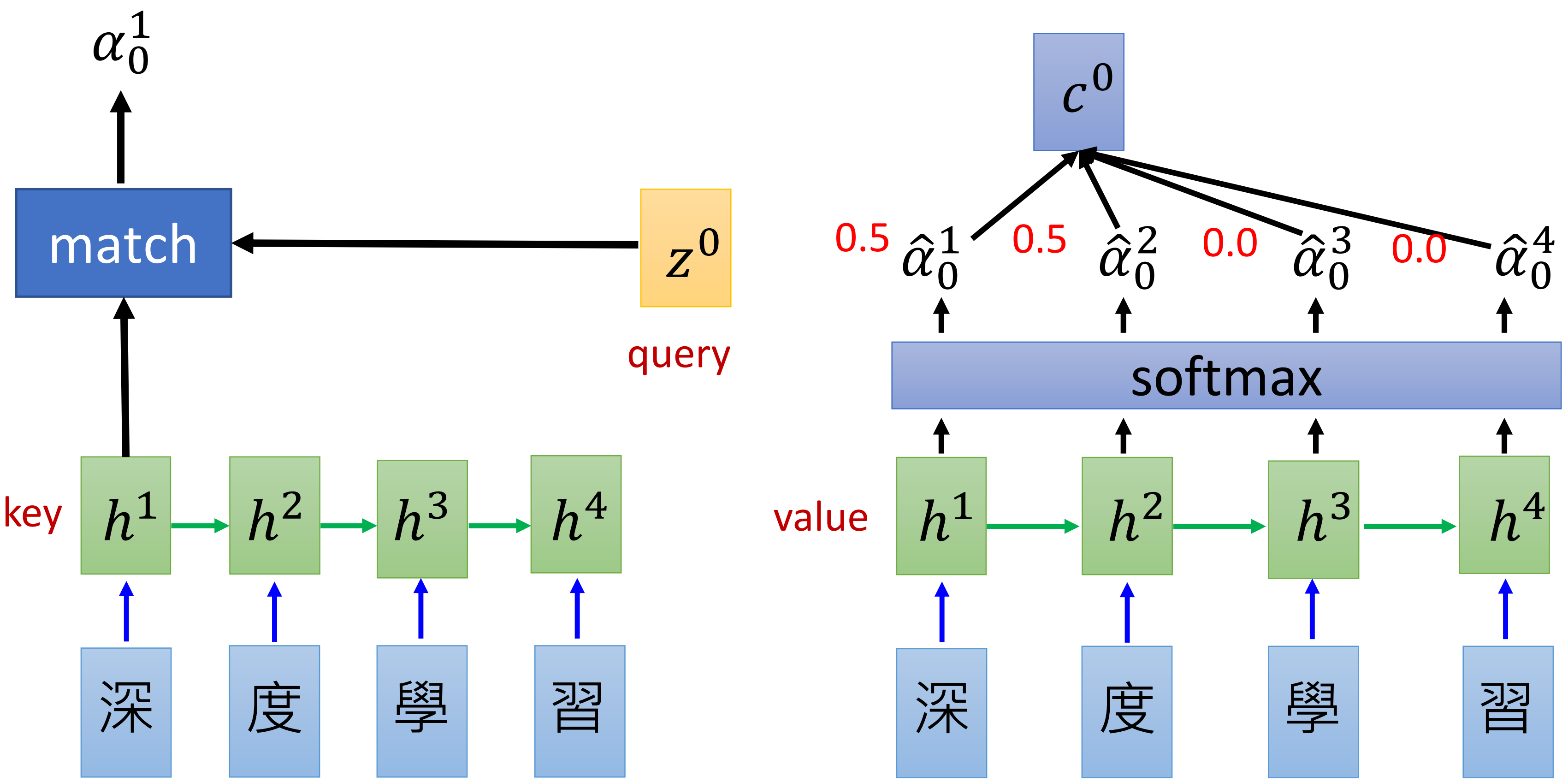
Output: weighted sum of values

Inner product of
query and corresponding key

$$A(q, K, V) = \sum_i \boxed{\frac{\exp(q \cdot k_i)}{\sum_j \exp(q \cdot k_j)}} v_i$$

- Query q is a d_k-dim vector
- Key k is a d_k-dim vector
- Value v is a d_v-dim vector

Machine Translation with Attention



Dot-Product Attention in Matrix

Input: *multiple* queries q and a set of key-value (k-v) pairs to an output

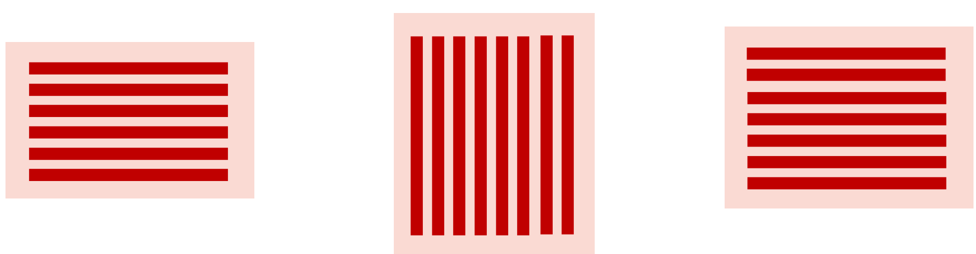
Output: a set of weighted sum of values

$$A(q, K, V) = \sum_i \frac{\exp(q \cdot k_i)}{\sum_j \exp(q \cdot k_j)} v_i$$

$$A(Q, K, V) = \mathrm{softmax}(QK^T)V$$

$$[|Q| \times d_k] \times [d_k \times |K|] \times [|K| \times d_v]$$

softmax
row-wise



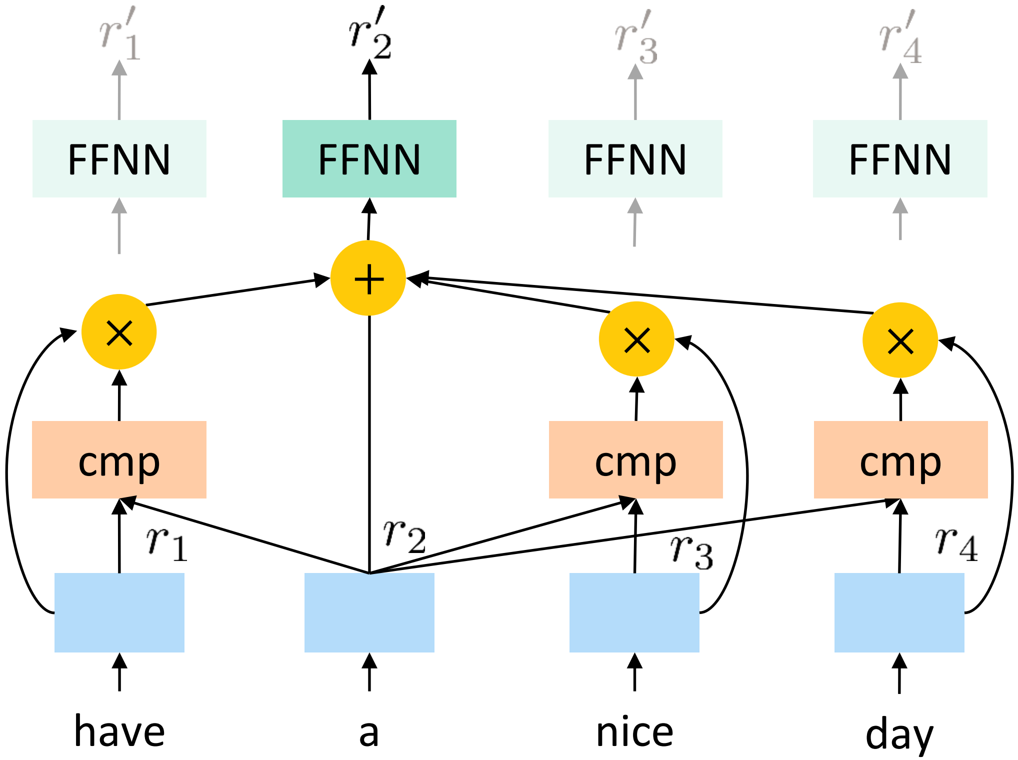
$$= [|Q| \times d_v]$$

Self-Attention

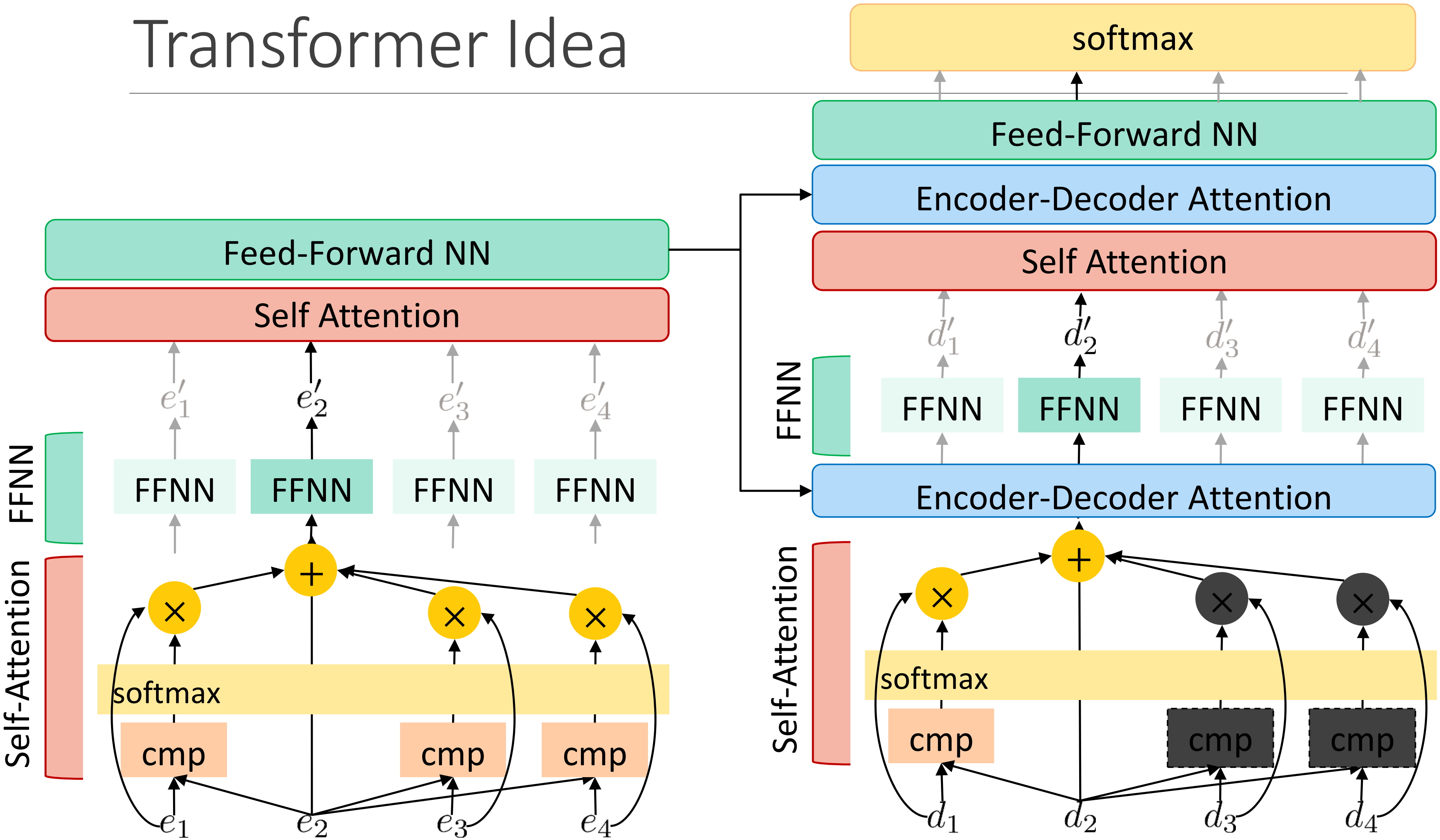
Self-Attention

Constant "path length" between two positions

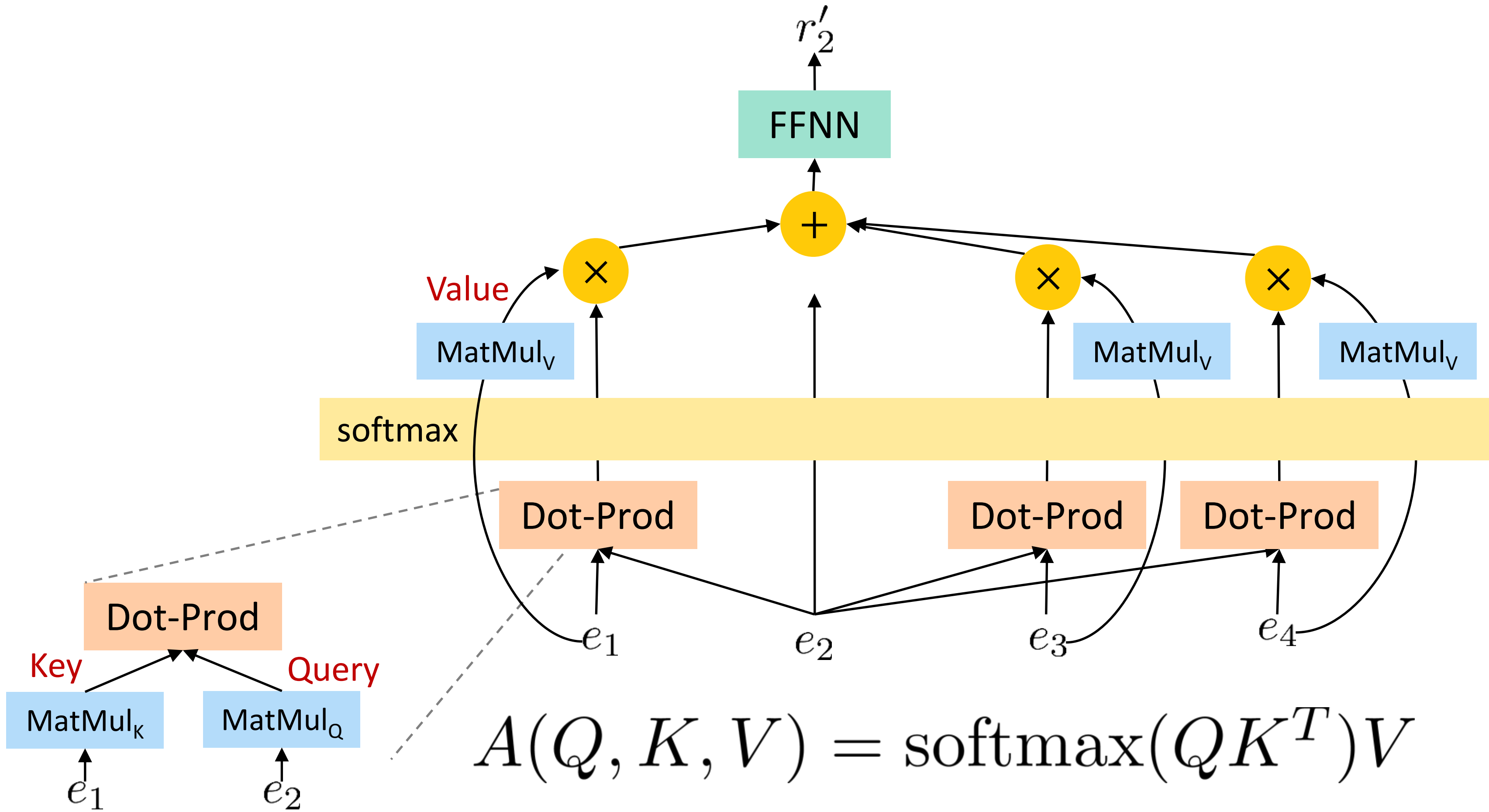
Easy to parallelize



Transformer Idea

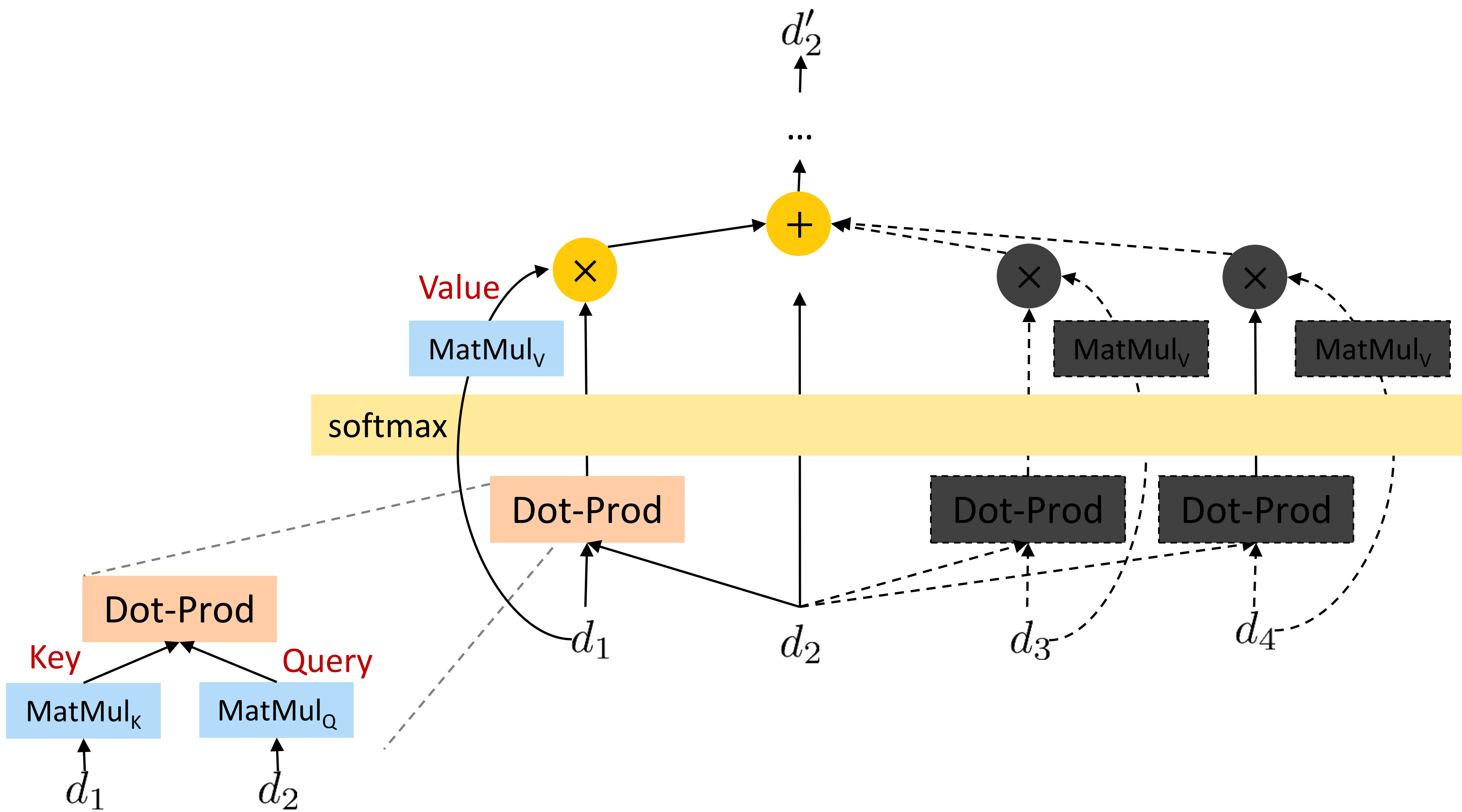


Encoder Self-Attention (Vaswani+, 2017)



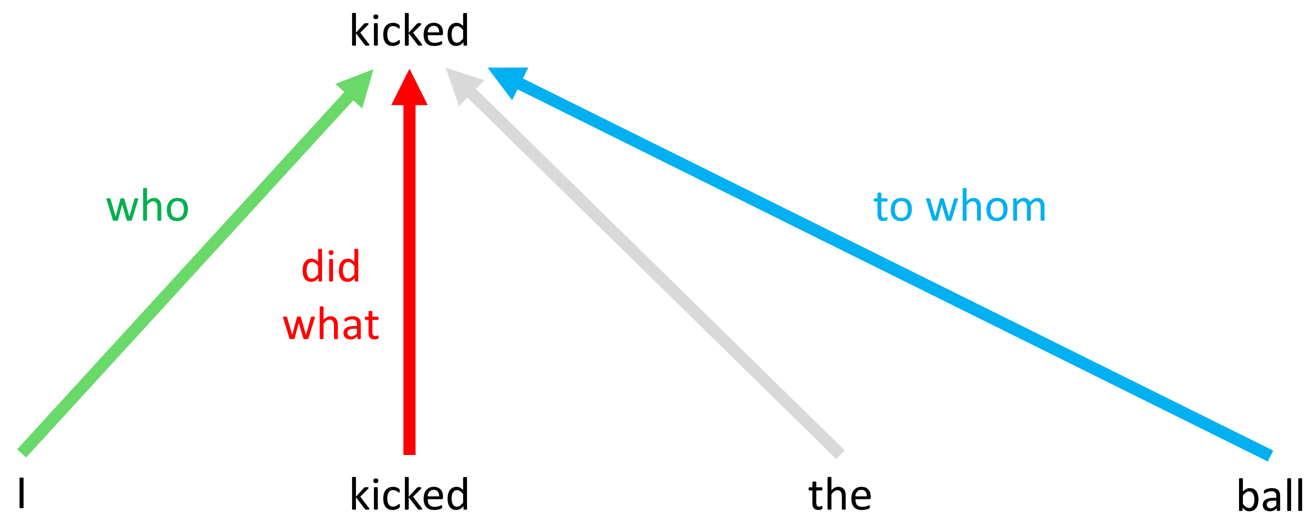
$$A(Q,K,V) = \text{softmax}(QK^T)V$$

Decoder Self-Attention (Vaswani+, 2017)

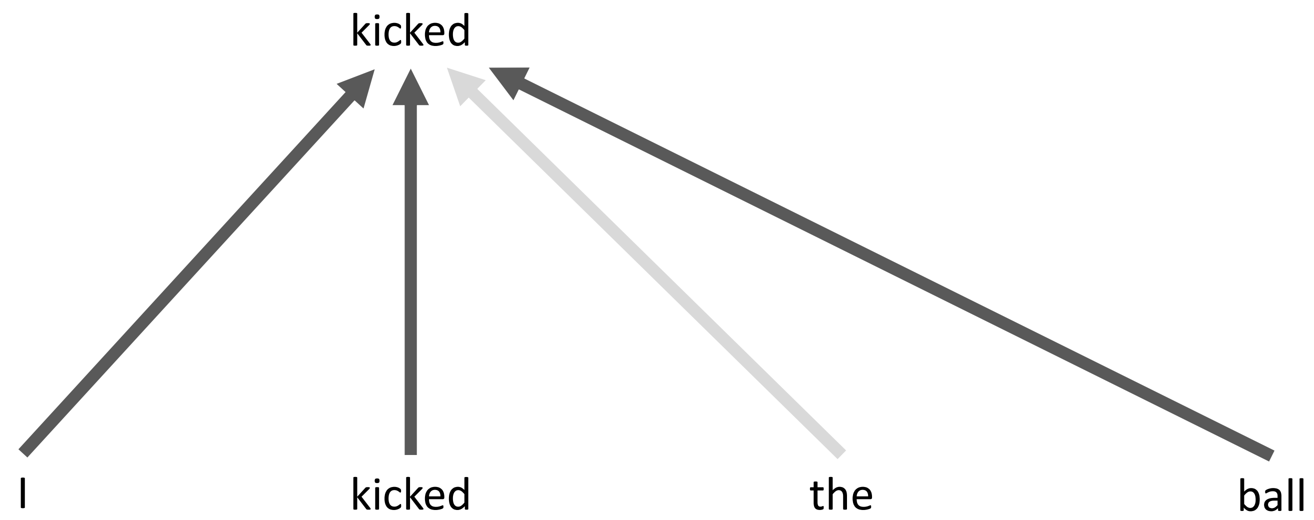


Multi-Head Attention

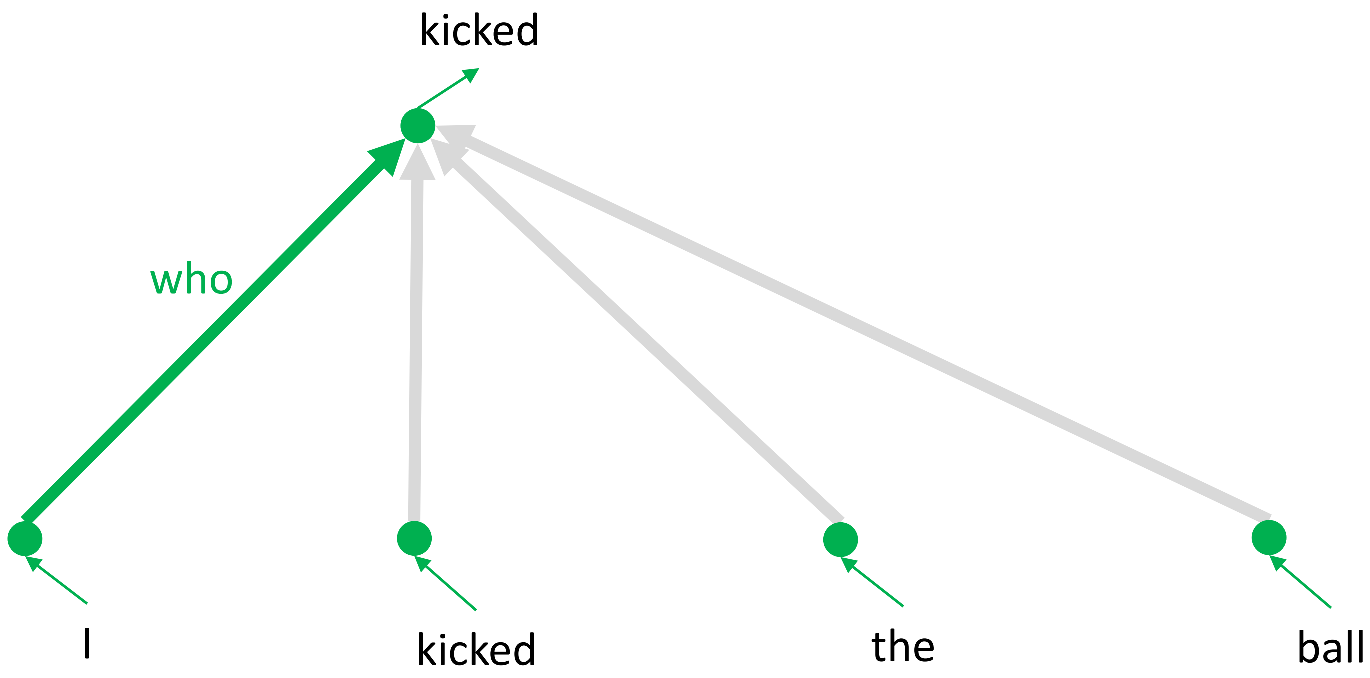
Convolutions



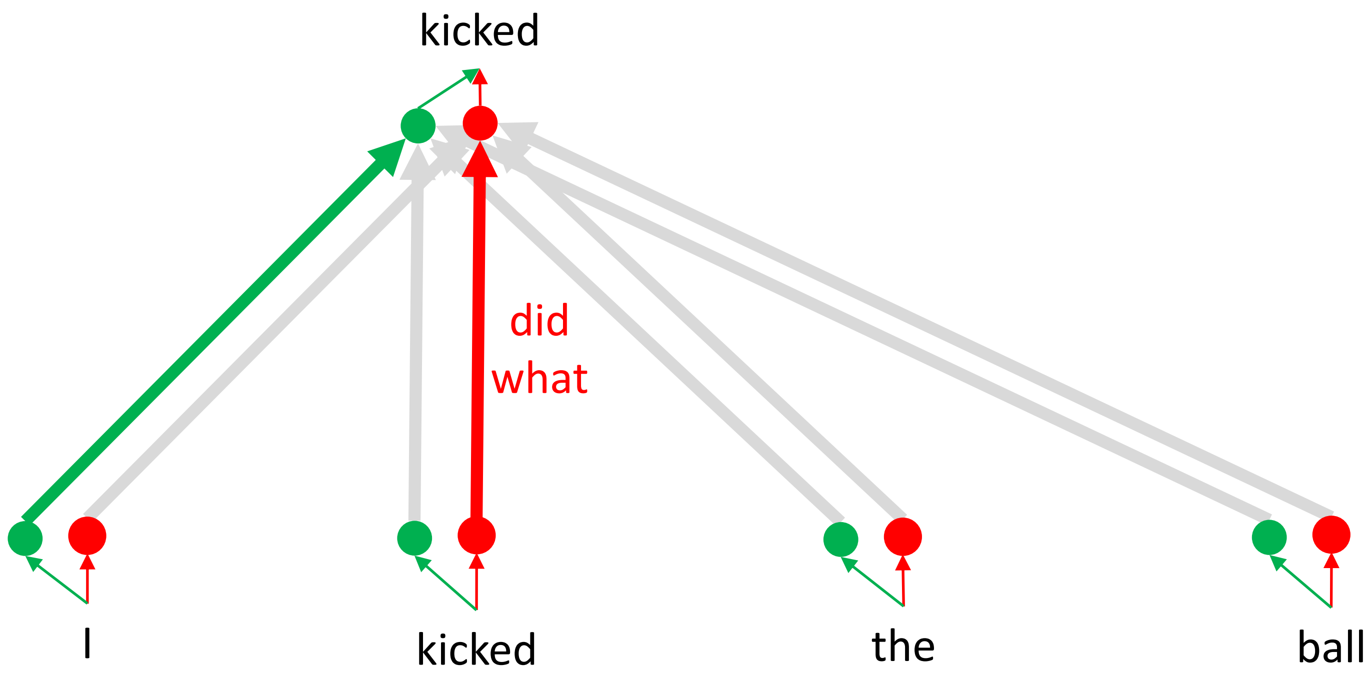
Self-Attention



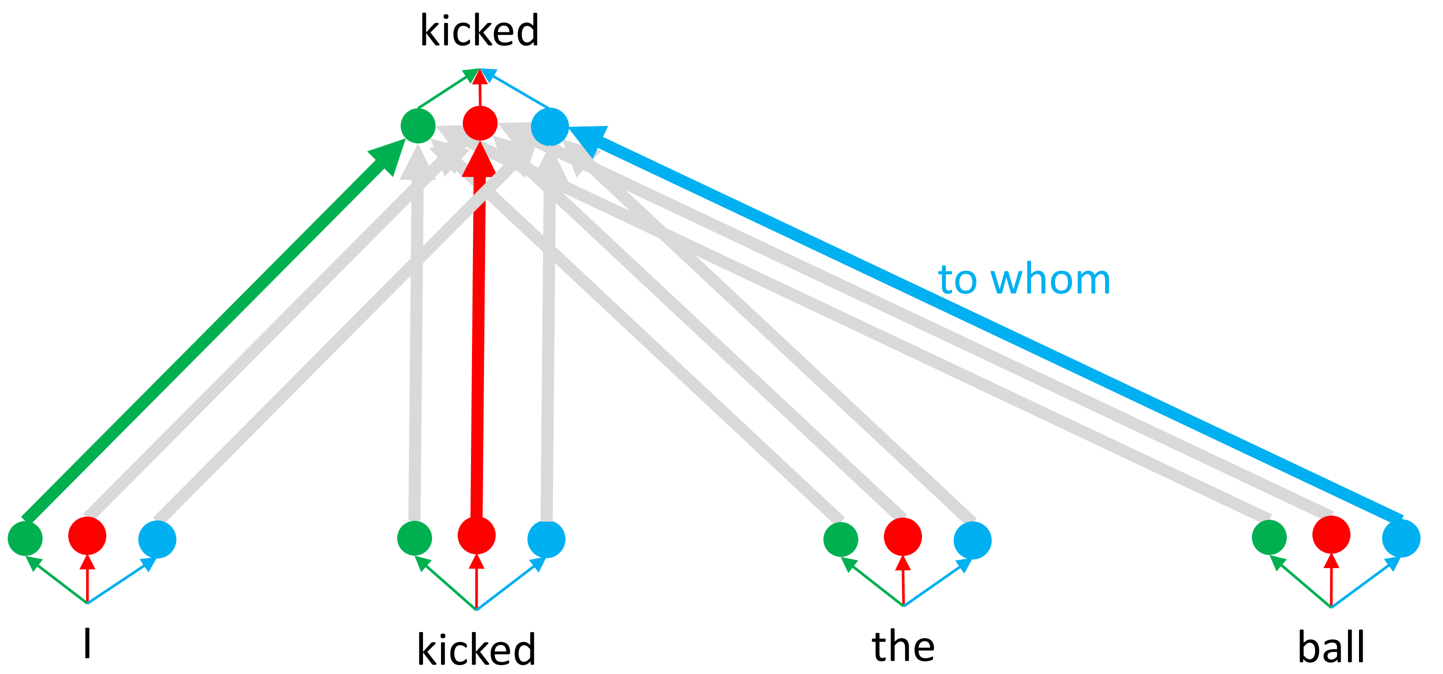
Attention Head: who



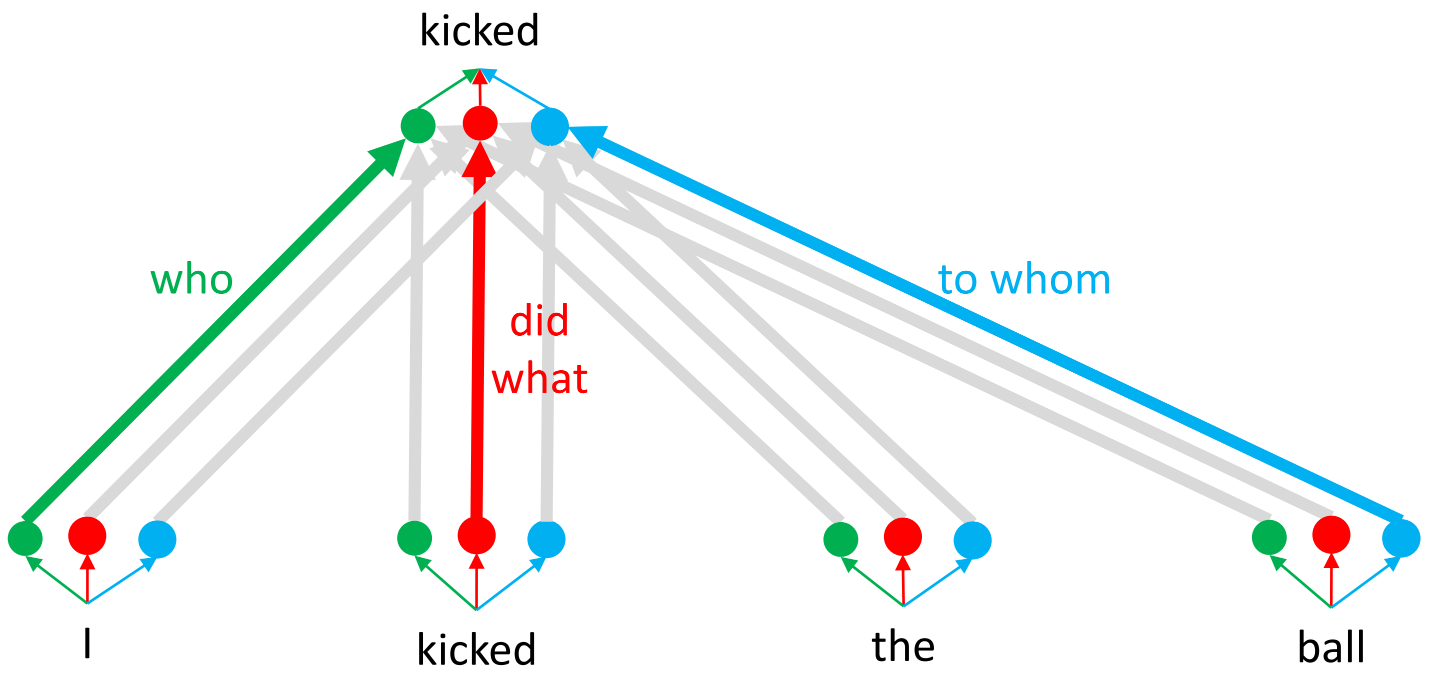
Attention Head: did what



Attention Head: to whom

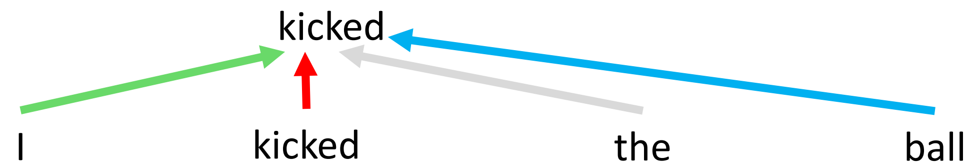


Multi-Head Attention

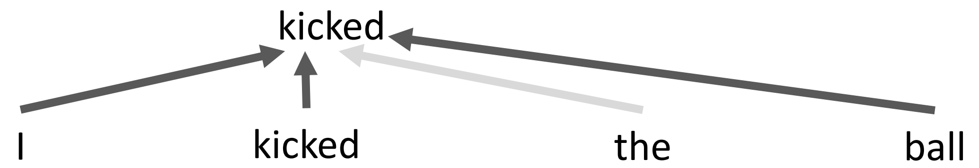


Comparison

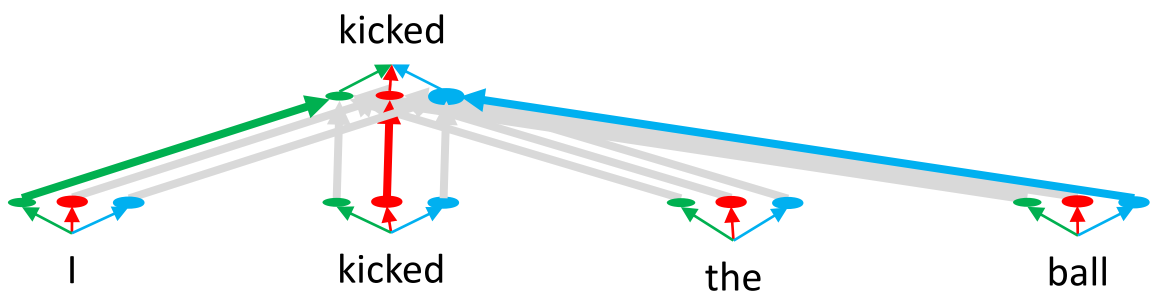
Convolution: different linear transformations by relative positions



Attention: a weighted average



Multi-Head Attention: parallel attention layers with different linear transformations on input/output

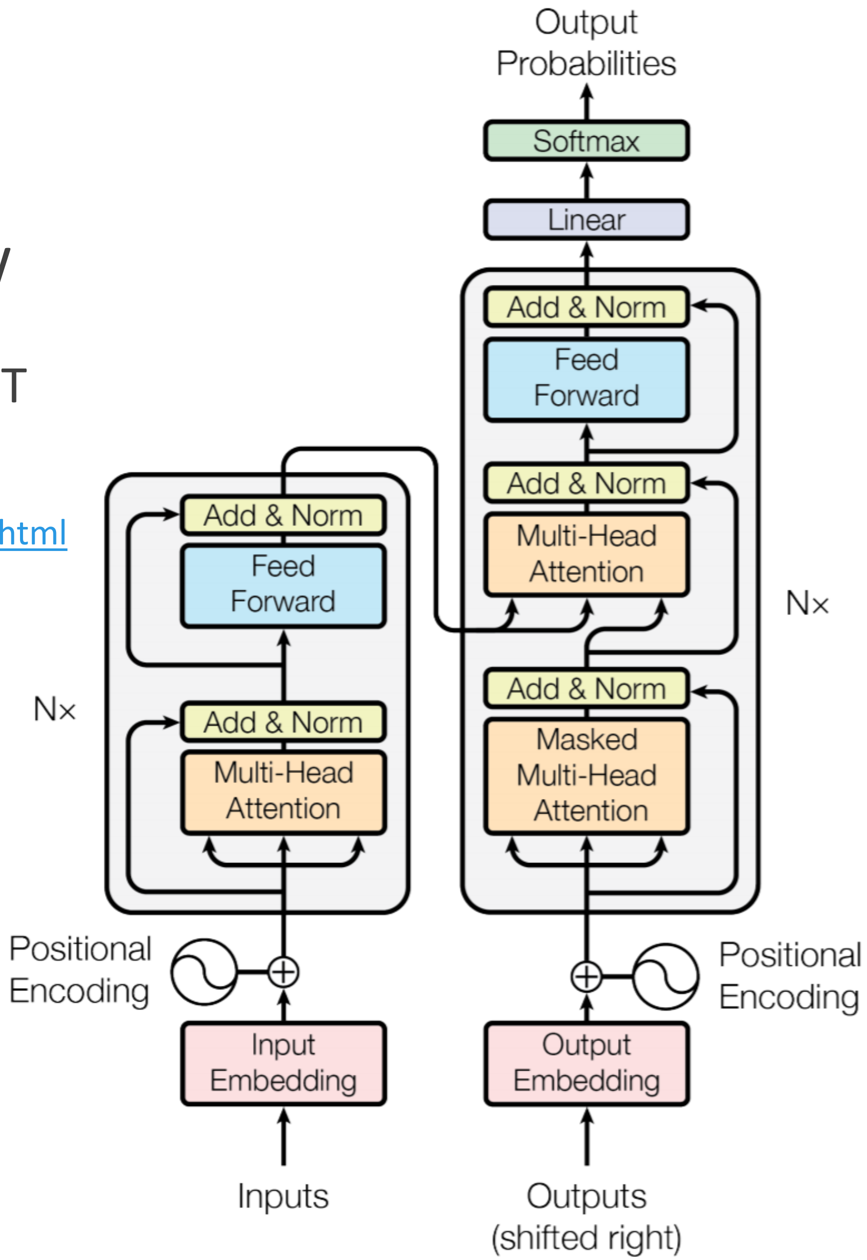


Transformer Overview

Non-recurrent encoder-decoder for MT

PyTorch explanation by Sasha Rush

- http://nlp.seas.harvard.edu/2018/04/03/attention.html

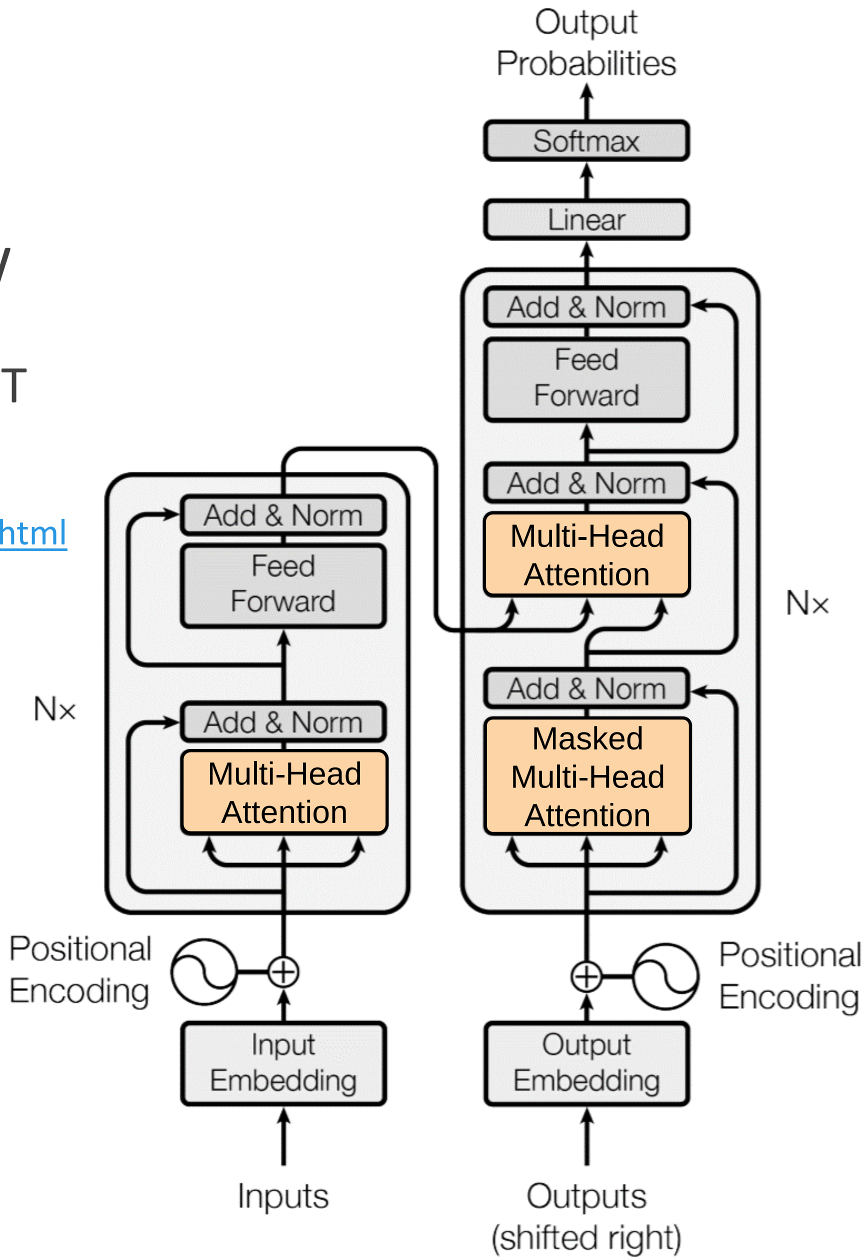


Transformer Overview

Non-recurrent encoder-decoder for MT

PyTorch explanation by Sasha Rush

- http://nlp.seas.harvard.edu/2018/04/03/attention.html



Multi-Head Attention

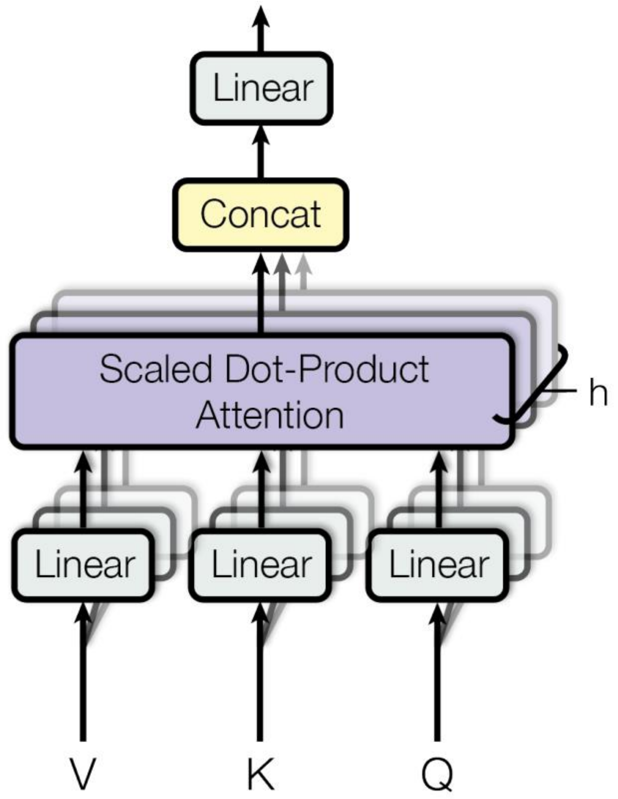
Idea: allow words to interact with one another

Model

- Map V, K, Q to lower dimensional spaces
- Apply attention, concatenate outputs
- Linear transformation

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head}_1, \cdots, \mathrm{head}_h) W^O$$

$$\mathrm{head}_i = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



Scaled Dot-Product Attention

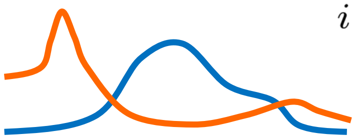
Problem: when d_k gets large, the variance of $q^T k$ increases

→ some values inside softmax get large

→ the softmax gets very peaked

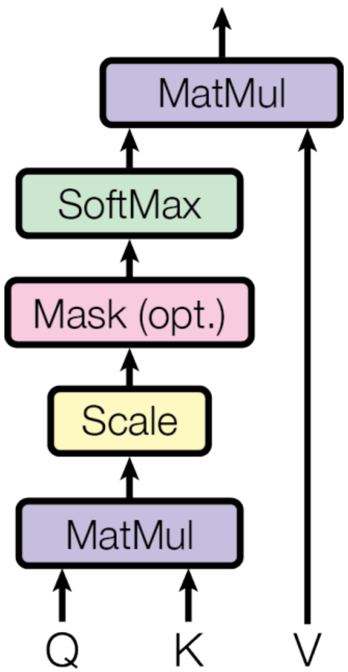
→ hence its gradient gets smaller

$$A(q, K, V) = \sum_i \frac{\exp(q \cdot k_i)}{\sum_j \exp(q \cdot k_j)} v_i$$



Solution: scale by length of query/key vectors

$$A(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

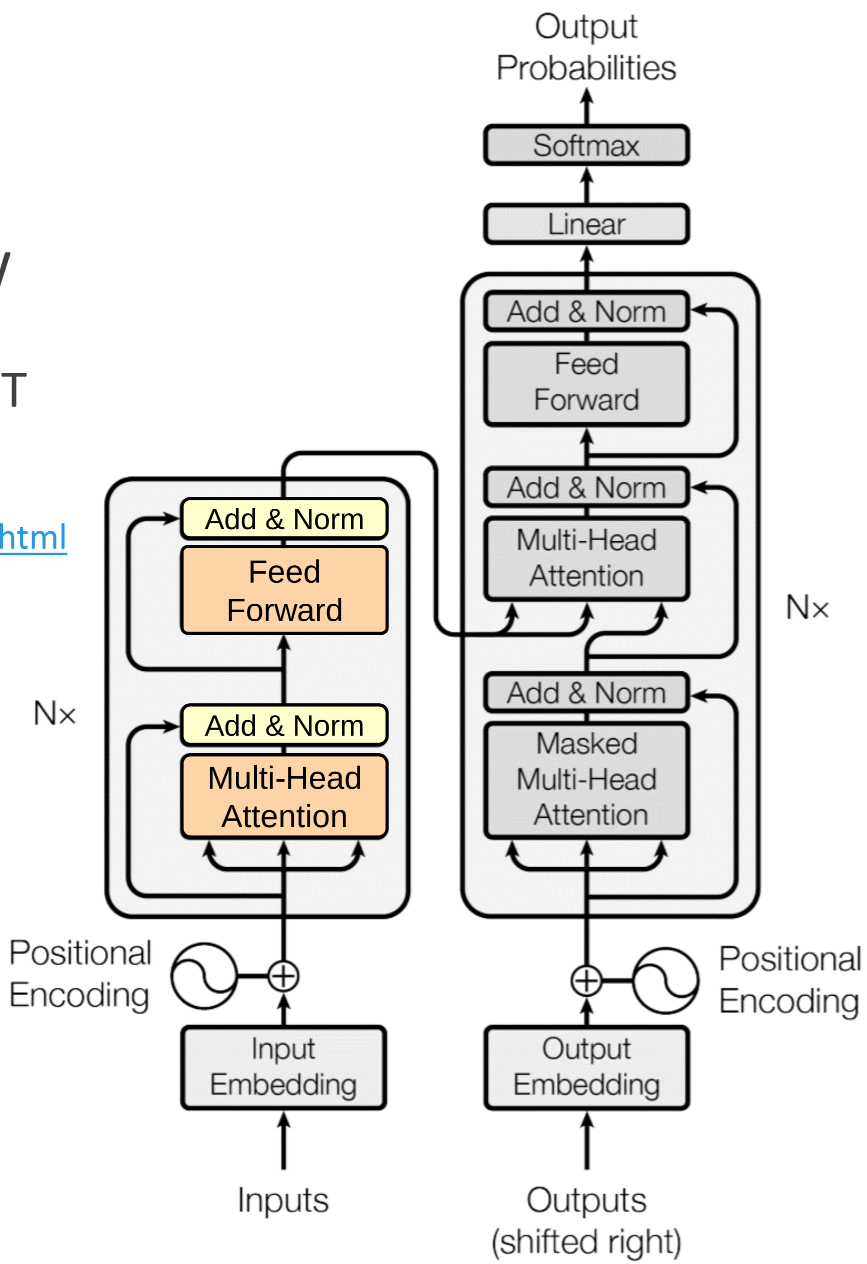


Transformer Overview

Non-recurrent encoder-decoder for MT

PyTorch explanation by Sasha Rush

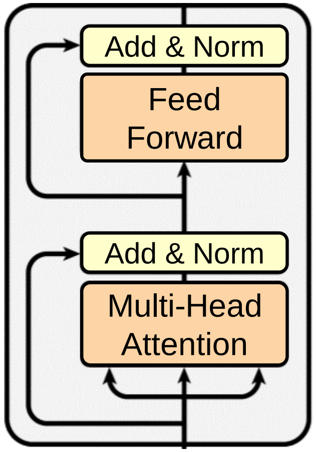
- http://nlp.seas.harvard.edu/2018/04/03/attention.html



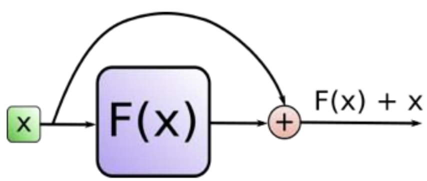
Transformer Encoder Block

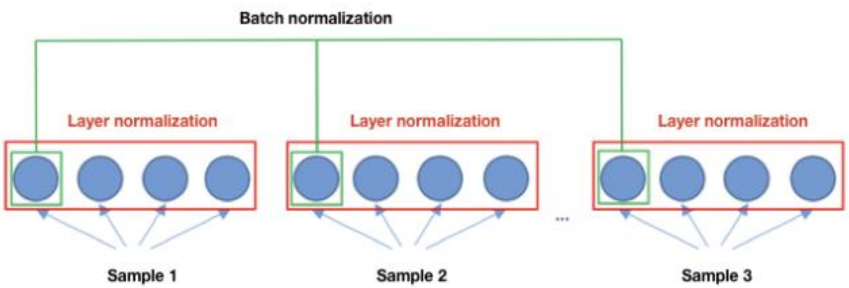
Each block has
- multi-head attention
- 2-layer feed-forward NN (w/ ReLU)

Both parts contain
- Residual connection & layer normalization (LayerNorm)
  - LayerNorm(x + sublayer(x))
  - Change input to have 0 mean and 1 variance per layer & per training point



$$H(x) = g(x) = x + F(x)$$





$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l \qquad \sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(a_i^l - \mu^l)^2} \qquad h_i = f(\frac{g_i}{\sigma_i}(a_i - \mu_i) + b_i)$$

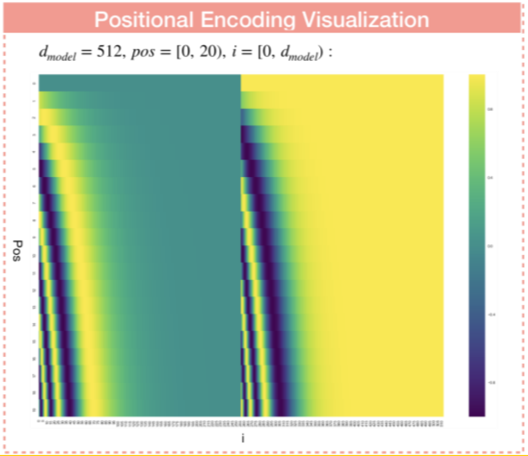
Encoder Input

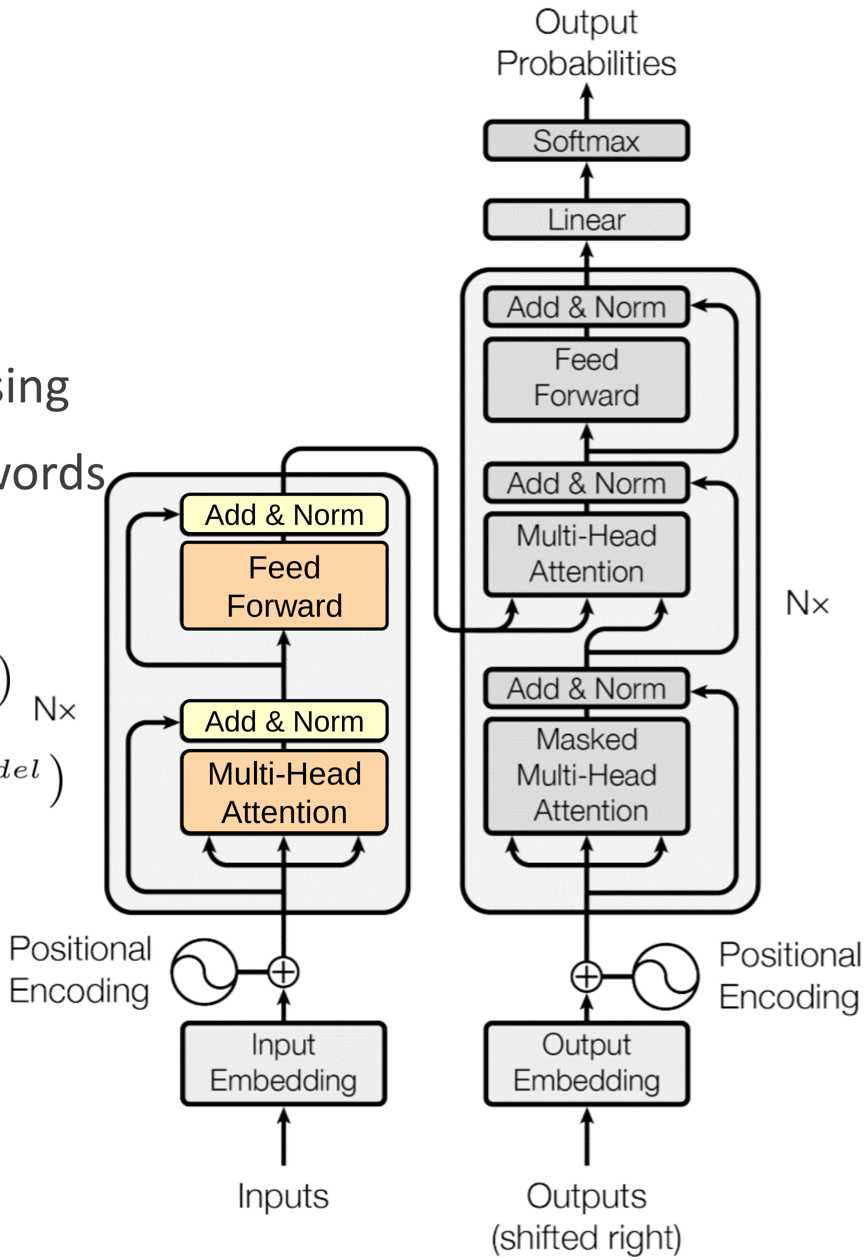
Problem: temporal information is missing

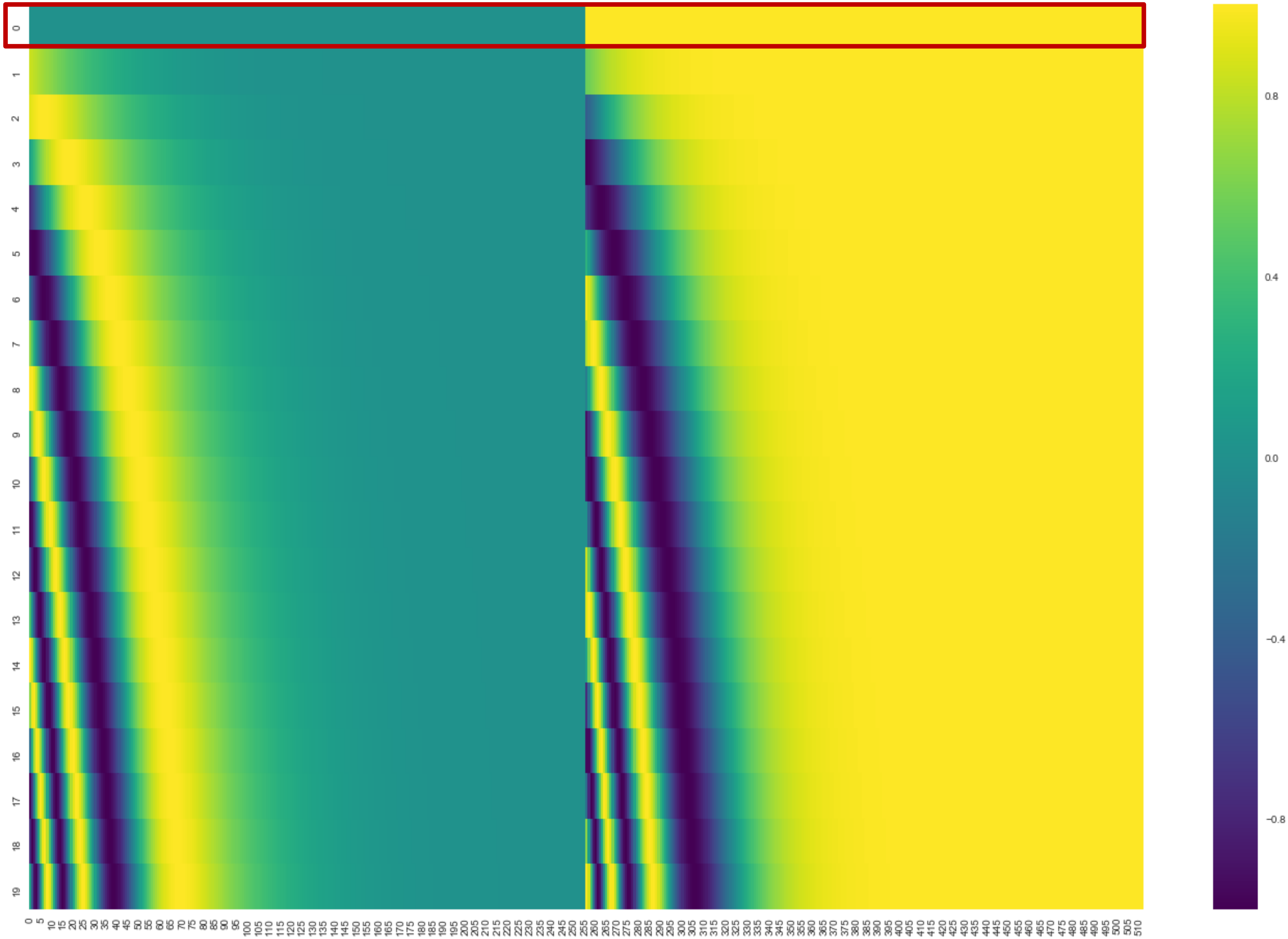
Solution: **positional encoding** allows words at different locations to have different embeddings with fixed dimensions

$$\mathrm{PE}_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

$$\mathrm{PE}_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$






0.8
0.4
0.0
−0.4
−0.8

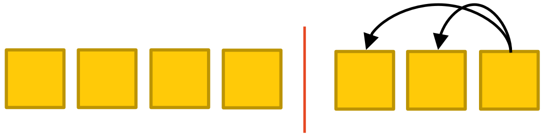
Multi-Head Attention Details

encoder self attention

1. Multi-head Attention
2. Query=Key=Value

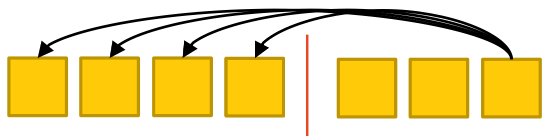
decoder self attention

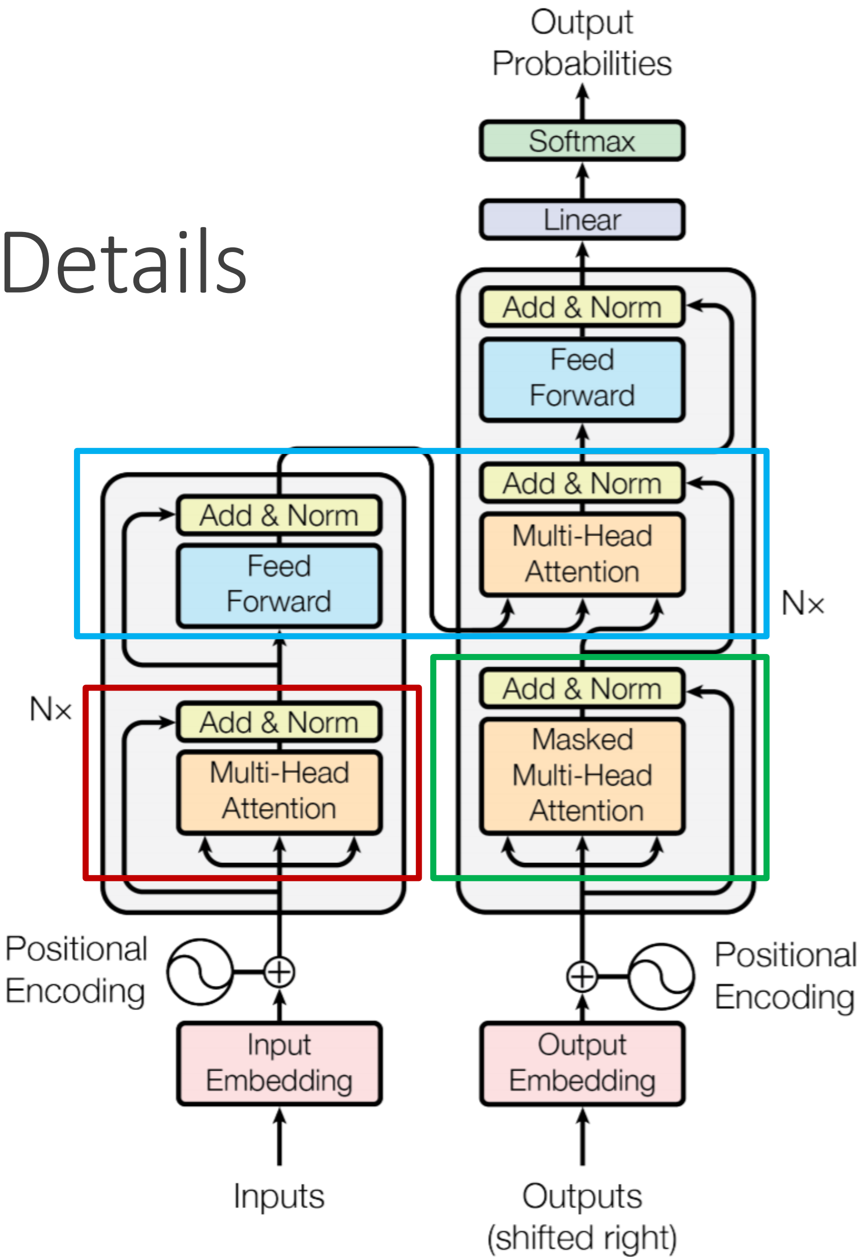
1. Masked Multi-head Attention
2. Query=Key=Value



encoder-decoder attention

1. Multi-head Attention
2. Encoder Self attention=Key=Value
3. Decoder Self attention=Query





Training Tips

Byte-pair encodings

Checkpoint averaging

ADAM optimizer with learning rate changes

Dropout during training at every layer just before adding residual

Label smoothing

Auto-regressive decoding with beam search and length penalties

MT Experiments

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

Vaswani et al., "Attention Is All You Need", in *NIPS*, 2017.

Parsing Experiments

Parser	Training	WSJ 23 F1
Vinyals & Kaiser el al. (2014) [37]	WSJ only, discriminative	88.3
Petrov et al. (2006) [29]	WSJ only, discriminative	90.4
Zhu et al. (2013) [40]	WSJ only, discriminative	90.4
Dyer et al. (2016) [8]	WSJ only, discriminative	91.7
Transformer (4 layers)	WSJ only, discriminative	91.3
Zhu et al. (2013) [40]	semi-supervised	91.3
Huang & Harper (2009) [14]	semi-supervised	91.3
McClosky et al. (2006) [26]	semi-supervised	92.1
Vinyals & Kaiser el al. (2014) [37]	semi-supervised	92.1
Transformer (4 layers)	semi-supervised	92.7
Luong et al. (2015) [23]	multi-task	93.0
Dyer et al. (2016) [8]	generative	93.3

Vaswani et al., “Attention Is All You Need”, in *NIPS*, 2017.

Concluding Remarks

Non-recurrence model is easy to parallelize

Multi-head attention captures different aspects by interacting between words

Positional encoding captures location information

Each transformer block can be applied to diverse tasks