LG전자 Deep Learning 과정

Self-Attention and Transformer

Gunhee Kim

Computer Science and Engineering

서울대학교
SEOUL NATIONAL UNIVERSITY

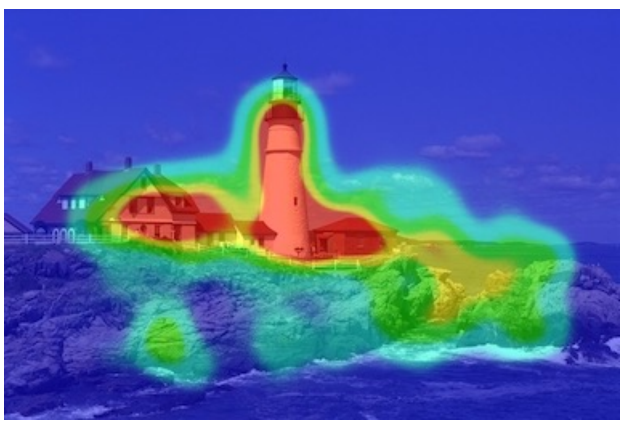
Outline

- Attention
- Self-attention
- Transformer

Attention Mechanism

Attention mechanisms in neural Networks

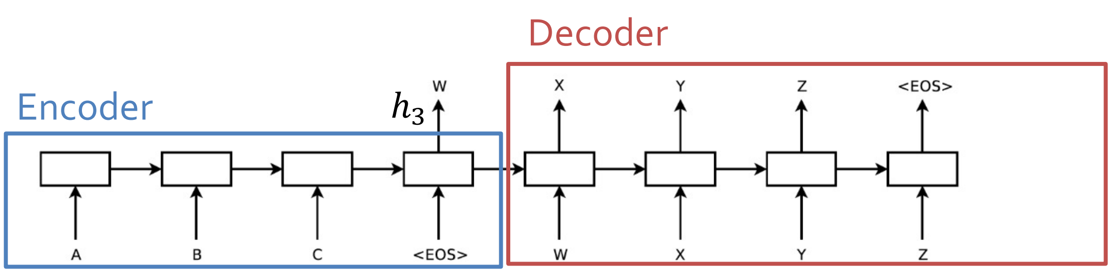
- Loosely based on the visual attention mechanism in humans
- Focus on a certain region of an image with high resolution while perceiving the surrounding image in low resolution
- Adjust the focal point over time



Attention Mechanism

First successfully applied in neural machine translation

- In previous seq2seq model, the decoder generates a translation solely based on the last hidden state (h_3) from the encoder
- Could be unreasonable that (1) a single vector encodes all information about a potentially very long sentence and (2) the decoder produces a good translation based on only that
- What if the input sentence consists of 50 words?
- Multiple (practical) hacks: (i) reverse the input sentence, or (ii) feed the input sentence twice….



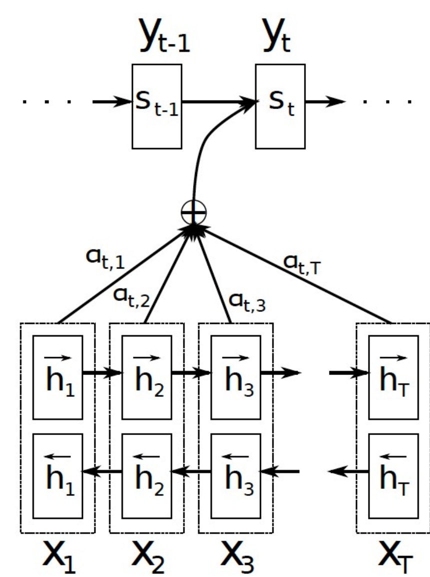
Attention Mechanism

Solution: attention mechanisms!

- The decoder **attends** to different parts of the source sentence at each step of the output generation

Encoder: a bi-directional RNN

- x_i / y_t: a word of source/translated sentence
- Forward RNN reads the sentence, and calculates forward hidden states $(\overrightarrow{h_i}, \ldots, \overrightarrow{h_T})$
- The embedding of each word x_j is obtained by the concatenation $h_i = [\overrightarrow{h_i}^T, \overleftarrow{h_i}^T]$



Bahdanau et al. Neural Machine Translation by Jointly Learning to Align and Translate. ICLR 2015

Attention Mechanism

Solution: attention mechanisms!

- The decoder **attends** to different parts of the source sentence at each step of the output generation

Decoder by soft-attention

- The context vector c_t with weight α_{it}

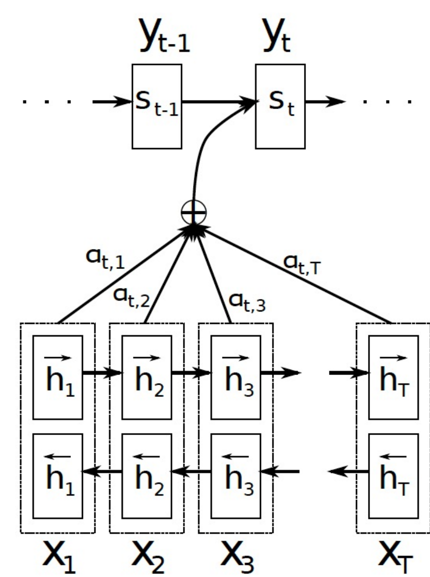
$$c_t = \sum_{i=1}^{T} \alpha_{ti} h_i$$

can consider all input i according to its importance

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{k=1}^{T} \exp(e_{tk})}$$

how much each input i contributes to output t

$$e_{ti} = v^T \tanh(W s_{t-1} + V h_i)$$



Attention Mechanism

Solution: attention mechanisms!

- The decoder **attends** to different parts of the source sentence at each step of the output generation

Decoder by soft-attention

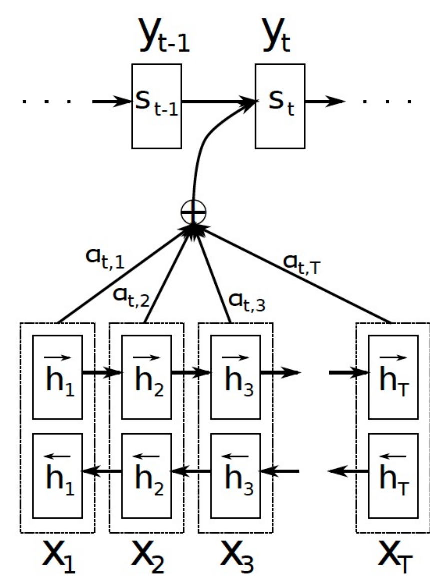
- The language model equation is

$$P(y_t | y_t, \dots, y_{t-1}, \boldsymbol{x}) = g(y_{t-1}, s_t, c_t)$$

where g is a nonlinear (multi-layered) function for probability output (e.g. hidden layers + softmax)

s_t is an RNN hidden state at time t

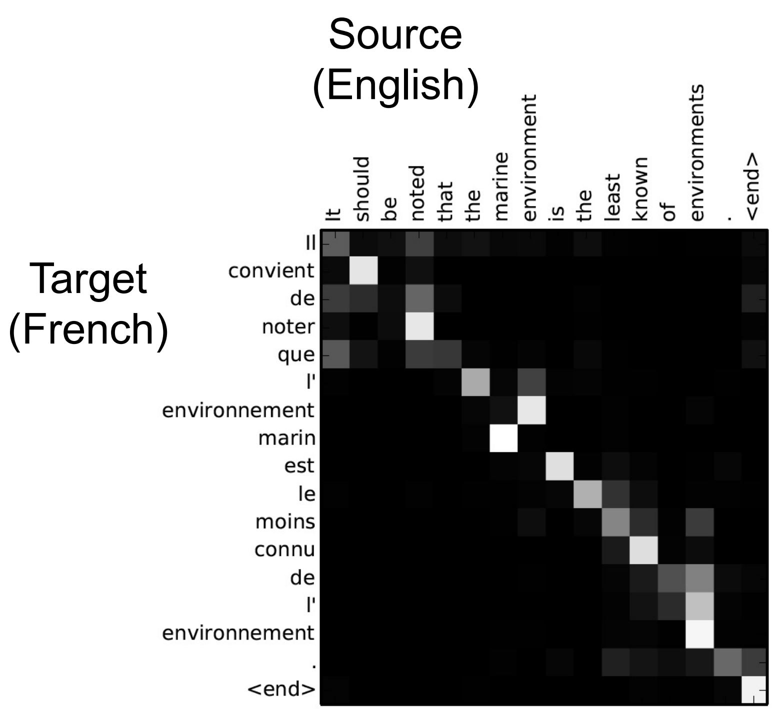
$$s_t = f(s_{t-1}, y_{t-1}, c_t)$$



Attention Mechanism

Visualizing the attention weight matrix $\boldsymbol{\alpha}$

- The decoder **attends** to different parts of the source sentence at each step of the output generation



The model generates each output word by **attending** sequentially to different input state

Cost of Attention

The size of attention weight matrix $\boldsymbol{\alpha}$ is quadratic

- e.g. if input/output sentence is 50 words long, then the size of $\boldsymbol{\alpha}$ is 50x50 = 2,500
- If we consider character-level tokenization, it could be prohibitively expensive

The previous attention mechanism is counter-intuitive

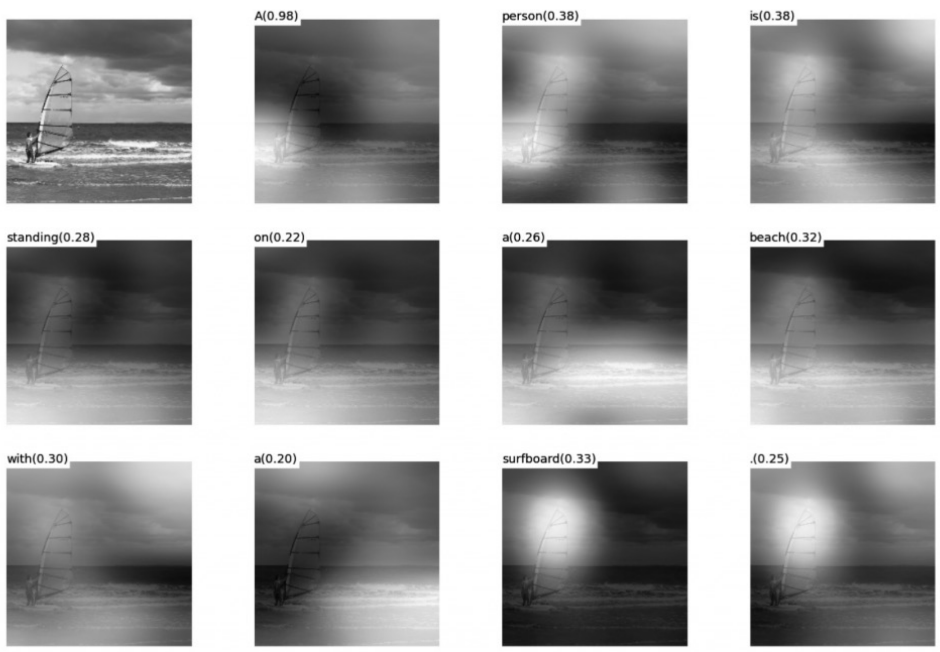
- For each output, we scan through all the input words (i.e. going back through all of our internal memory)
- Human instantaneously knows which parts of memory should be accessed
- Use reinforcement learning to predict an approximation location to focus to [Mnih 2014]

Mnih et al. Recurrent Models of Visual Attention. NIPS 2014

Image Captioning with Visual Attention

Solution: attention mechanisms!

- The decoder attends to different parts of the source ~~sentence~~ image at each step of the output generation



(b) A person is standing on a beach with a surfboard.

Xu et al. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. ICML 2015

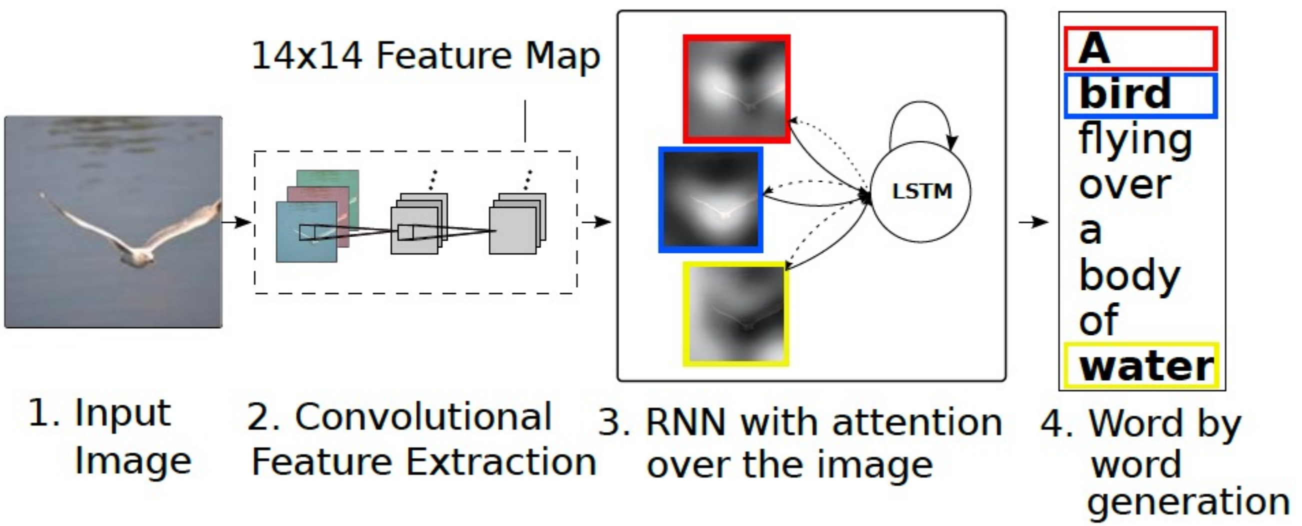
Image Captioning with Visual Attention

Encoder: 14x14 CNN feature map

- Let the decoder to selectively focus on certain parts of an image

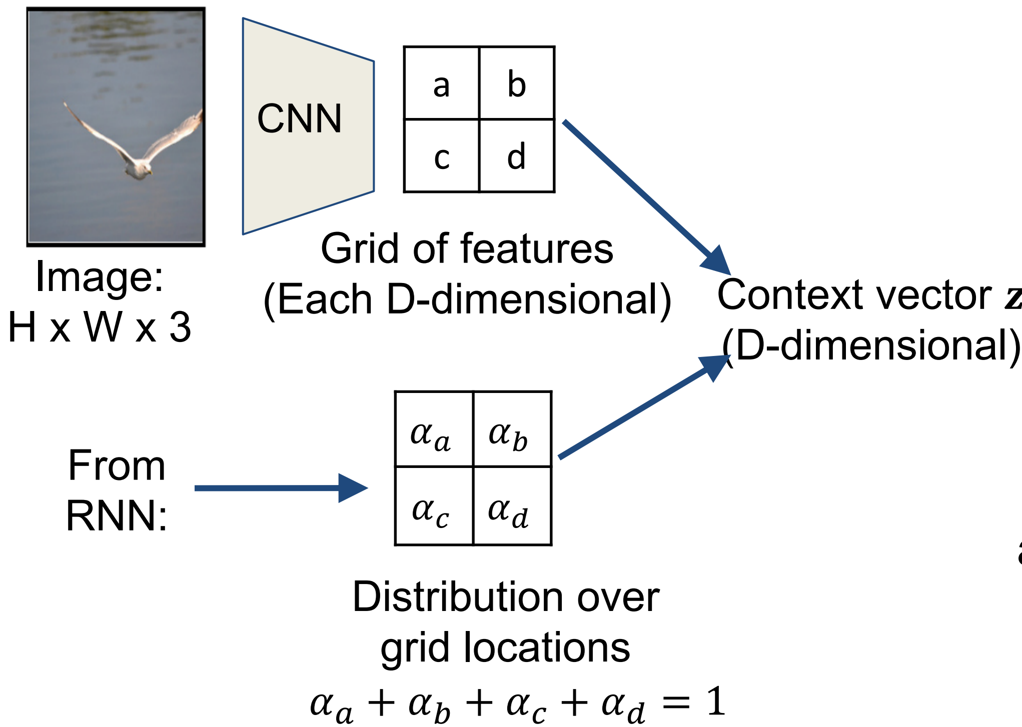
Decoder: LSTM layer

- Similar to the decoder of [Bahdanau et al. ICLR 2015]



Xu et al. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. ICML 2015

Language Model Pretraining



Soft attention:
Summarize ALL locations

$$z = \alpha_a a + \alpha_b b + \alpha_c c + \alpha_d d$$

Derivative $dz/d\alpha$ is nice!
Train with gradient descent

Hard attention:
(image cropping)
Sample ONE location
according to α, z = that vector

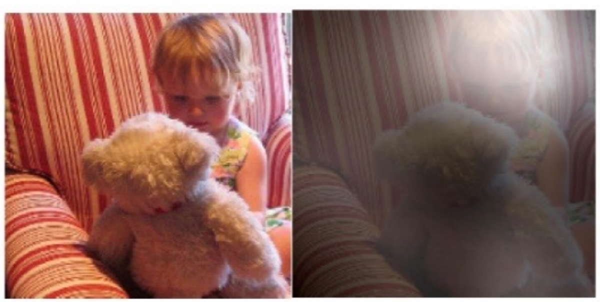
With argmax, $dz/d\alpha$ is zero
almost everywhere …
Can't use gradient descent;
need reinforcement learning

Image Captioning with Visual Attention

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.



A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Video Captioning with Temporal Attention

Solution: attention mechanisms!

- The decoder attends to different parts of the source ~~sentence~~ frame at each step of the output generation

Video captioning

- Given a short video clip, generate a descriptive sentence

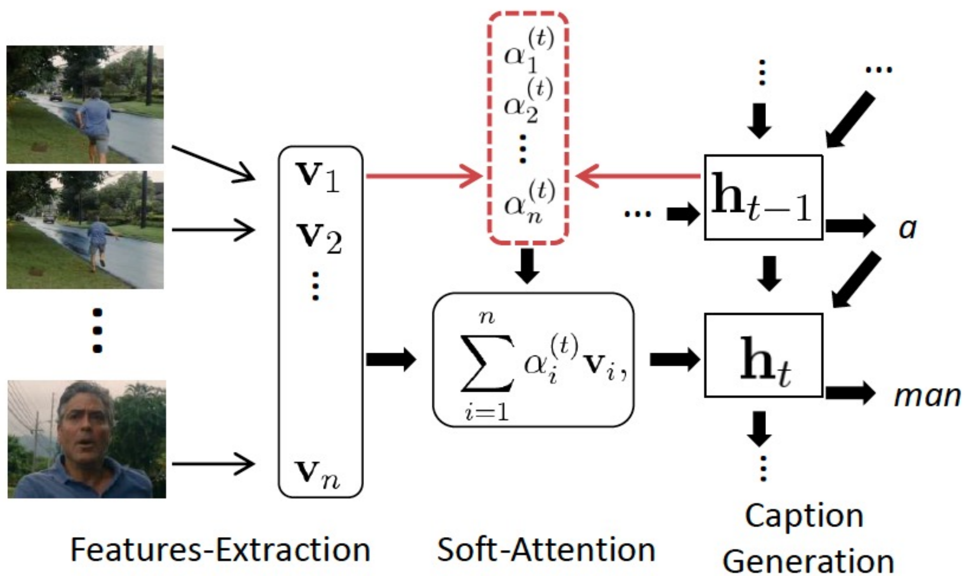
Yao et al. Describing Videos by Exploiting Temporal Structure. CVPR 2015

Video Captioning with Temporal Attention

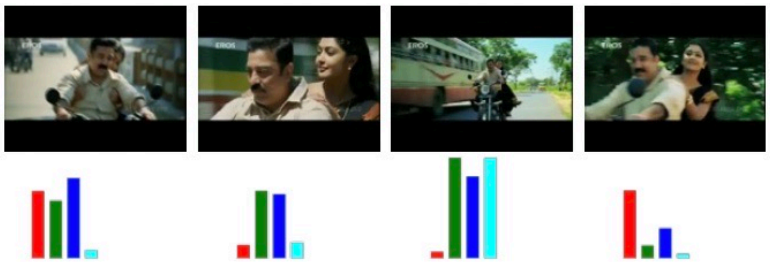
Encoder: 3D spatio-temporal CNN features for frames

Decoder: LSTM layer

- Similar to the decoder of [Bahdanau et al. ICLR 2015]
- $\boldsymbol{h}_t$: hidden state at time t

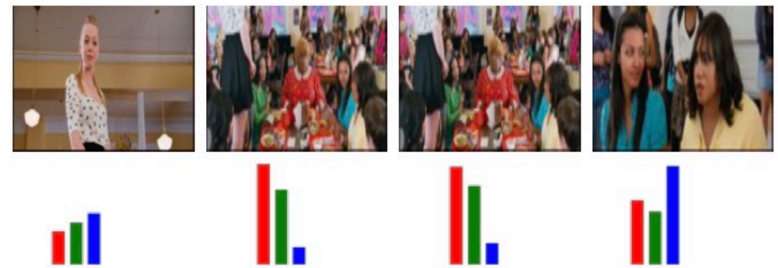


Video Captioning with Temporal Attention



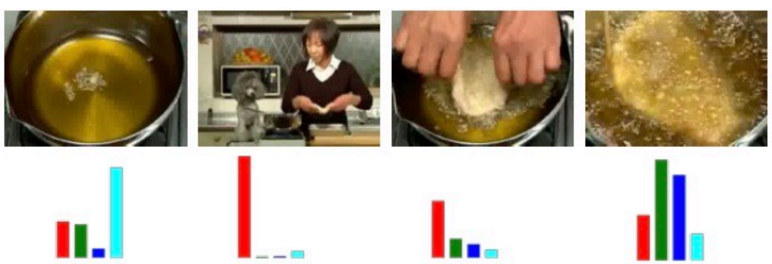
+Local+Global: A man and a woman are talking on the road

Ref: A man and a woman ride a motorcycle



+Local+Global: the girl grins at him

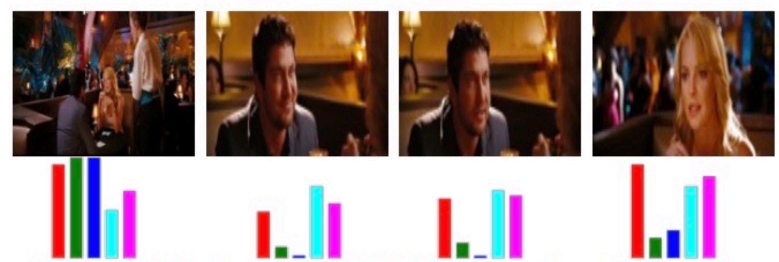
Ref: SOMEONE and SOMEONE swap a look



+Local+Global: Someone is frying a fish in a pot

+Local: Someone is frying something
+Global: The person is cooking
Basic: A man cooking its kitchen

Ref: A woman is frying food



+Local+Global: as SOMEONE sits on the table, SOMEONE shifts his gaze to SOMEONE

+Local: with a smile SOMEONE arrives
+Global: SOMEONE sits at a table
Basic: now, SOMEONE grins

Ref: SOMEONE gaze at SOMEONE

Memory Networks

Attention mechanisms

- The decoder attends to different parts of the source sentence/image/frame at each step of the output generation

Another interpretation: memory

- If we regard the source as the information in the memory, attention corresponds to selective memory access with weights

Memory networks = neural networks with external memory

- Prevents suffering from vanishing gradient problem

http://www.wildml.com/2016/01/attention-and-memory-in-deep-learning-and-nlp/

End-to-End Memory Networks

Neural networks (RNN) + explicit storage + attention

Application: question answering

- (i) input: a set of sentences $\{x_i\}$, (ii) query q, and (iii) a single word answer
- Word/sentence order matters

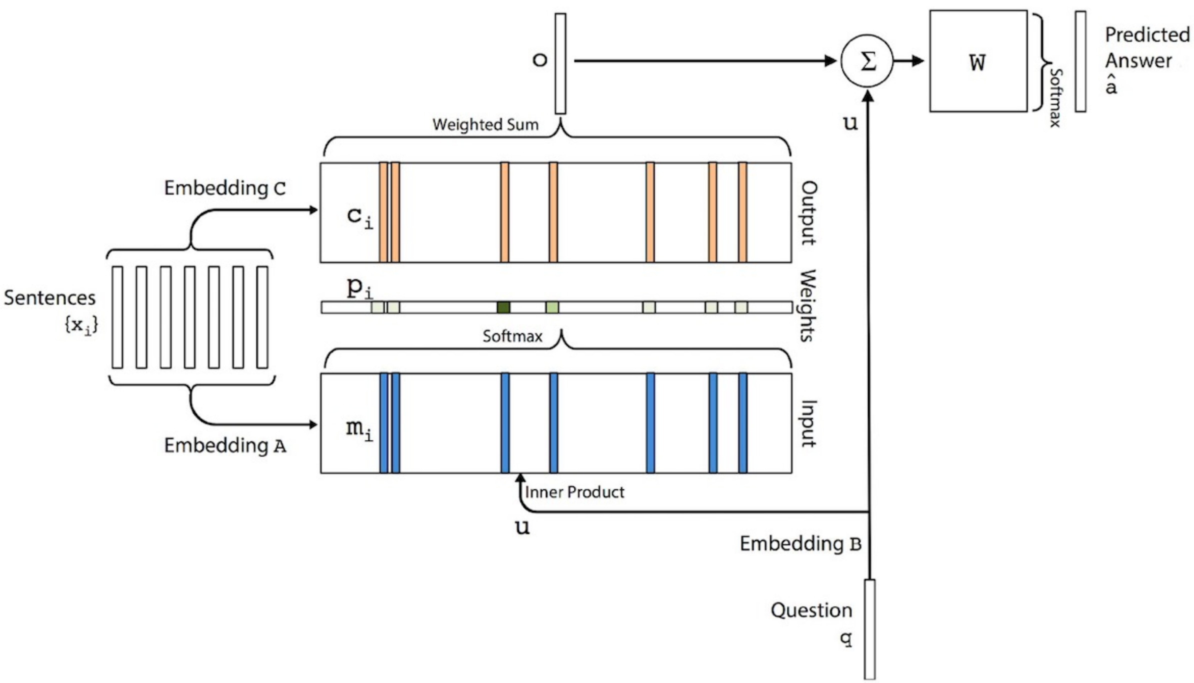
Sam walks into the kitchen. Sam picks up an apple. Sam walks into the bedroom. Sam drops the apple. Q: Where is the apple? A. Bedroom	Brian is a lion. Julius is a lion. Julius is white. Bernhard is green. Q: What color is Brian? A. White

Sukhbaatar et al. End-To-End Memory Networks. NIPS 2015

End-to-End Memory Networks

Neural networks (RNN) + explicit storage + attention

- Input $\{x_i\}$ are embedded by A, B into m_i, c_i (See the paper)
- Embedding matrices A, B, C, W are learned during training



(1) Query embedding

$u = Bq$

(2) Attention in memory

$p_i = \text{softmax}(u^T m_i)$

(3) Memory output

$o = \sum_i p_i c_i$

(4) Answer selection

$\hat{a} = \text{softmax}(W(o + u))$

Outline

- Attention
- Self-attention
- Transformer

Representation Learning of Sequential Data

Basic building block of sequence-to-sequence learning

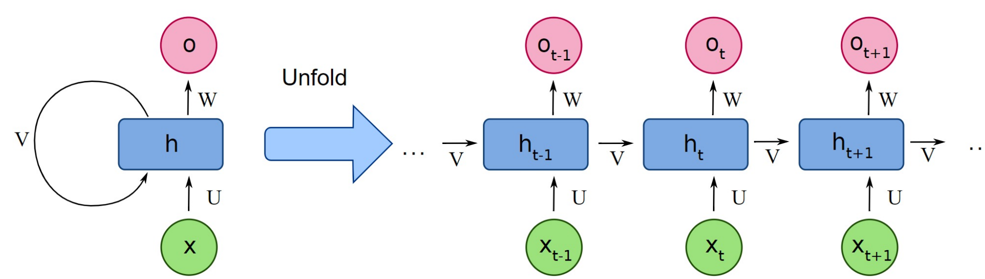
- Neural machine translation, summarization, QA, …

Recurrent Neural Networks (LSTMs, GRUs and variants)

- Natural fit for sentences and sequences of pixels

However,

- Sequential computation inhibits parallelization
- No explicit modeling of long and short range dependencies
- No model hierarchy



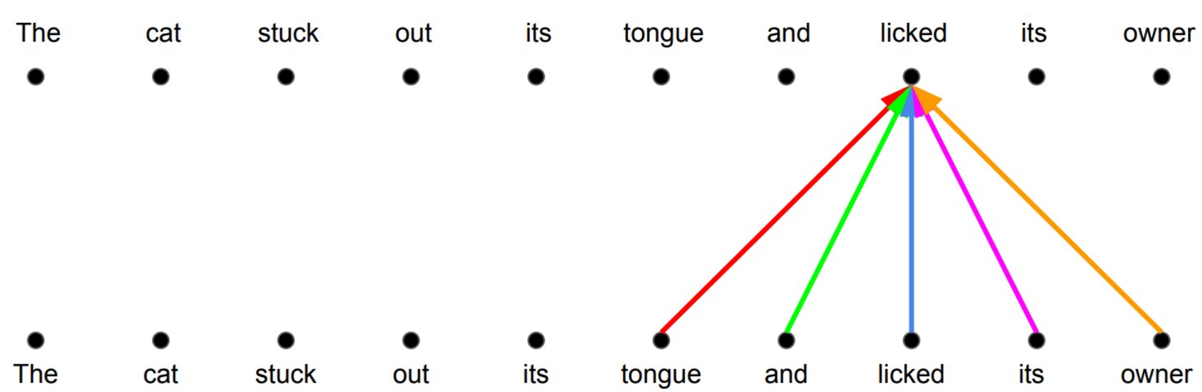
Representation Learning of Sequential Data

Convolutional neural networks

- Trivial to parallelize (per layer)
- Exploits local dependencies
- ‘Interaction distance’ between positions linear or logarithmic …

However,

- Long-distance dependencies require many layers



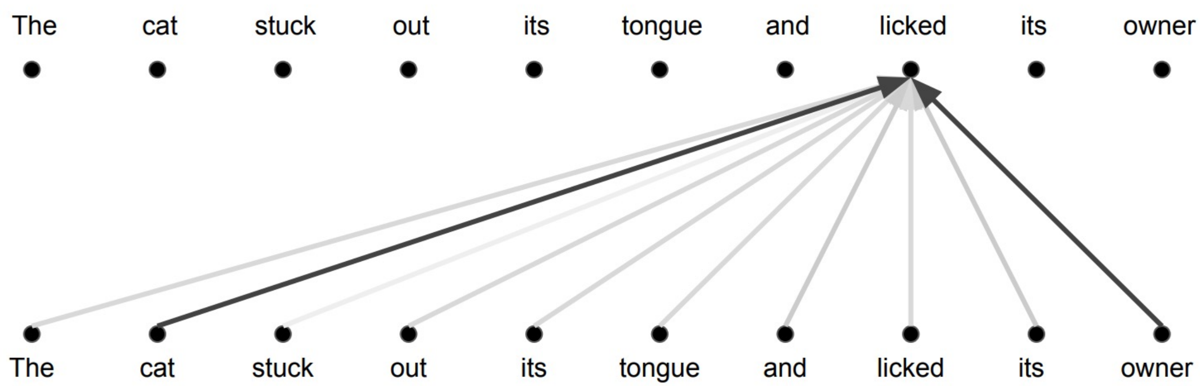
Self-Attention

Why not use attention for representations?

- Self-attention directly models relationships between all words in a sentence, regardless of their respective position

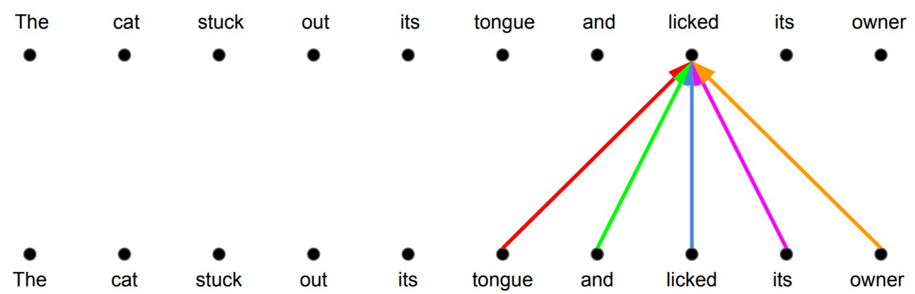
I arrived at the bank after crossing the river

- For representation of “*bank*”, use a weighted average of all words’ representations (the attention scores are used as weights)
- Trivial to parallelize (per layer)
- Constant ‘path length’ between any two positions

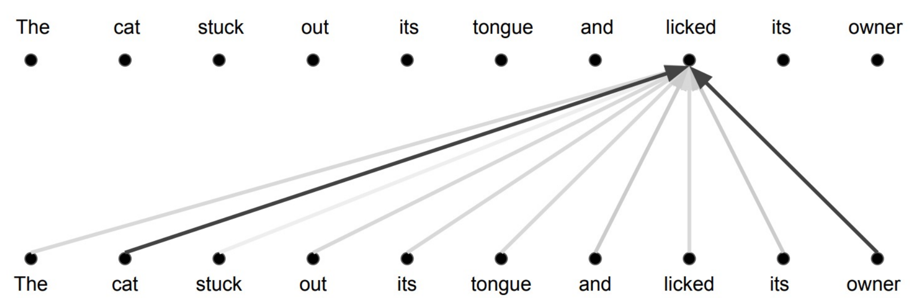


Convolution vs Attention

Convolution: different linear transformations by relative position



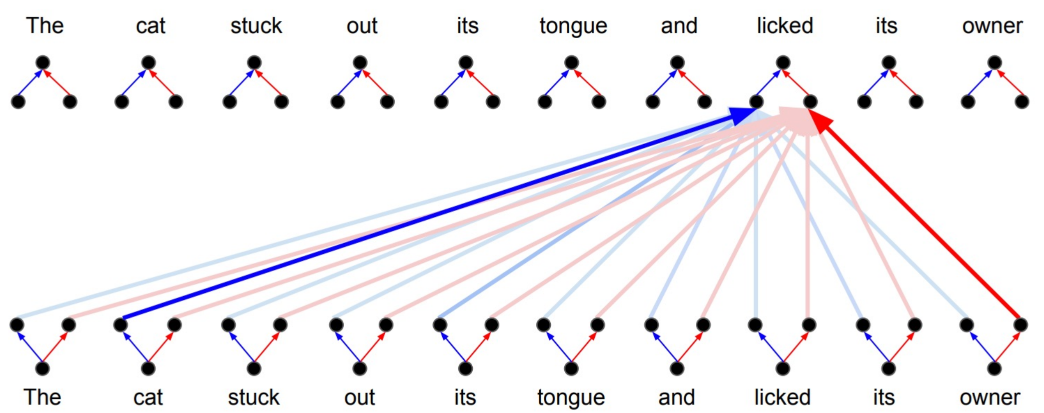
Convolution: a weighted average for all inputs



Multi-head Attention

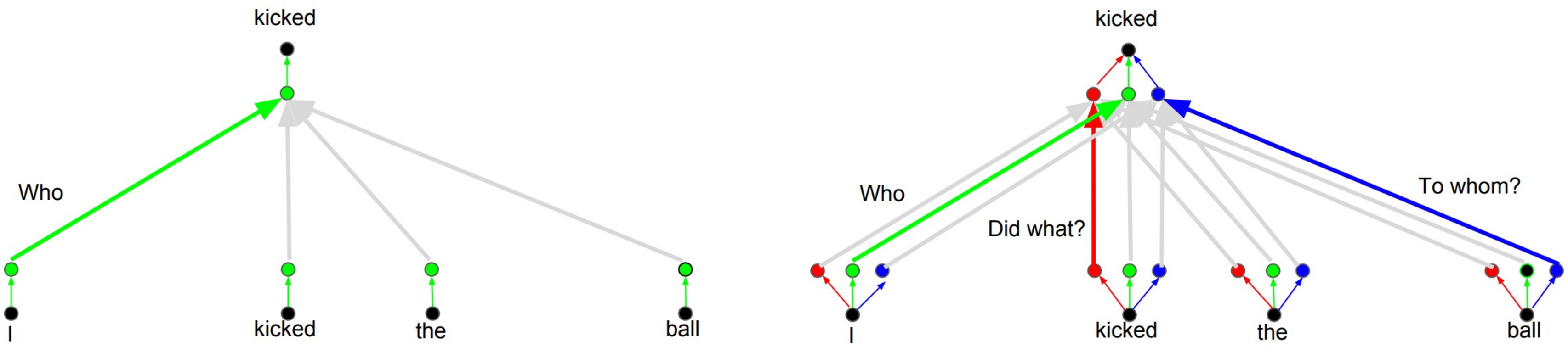
Parallel attention layers with different linear transformations on input and output

- It turns out that multiple different attentions are helpful



Multi-head Attention

Can combine the knowledge explored by multiple heads or agents instead of doing it by one



Comparison with Other Architectures

Self-attention connects all positions with a constant number of sequentially executed operations

- Moreover, $n \ll d$ usually

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

The amount of computation that can be parallelized

The maximum path length between any two input/output

Image Credit: Vaswani et al. Attention Is All You Need. NIPS 2017

Outline

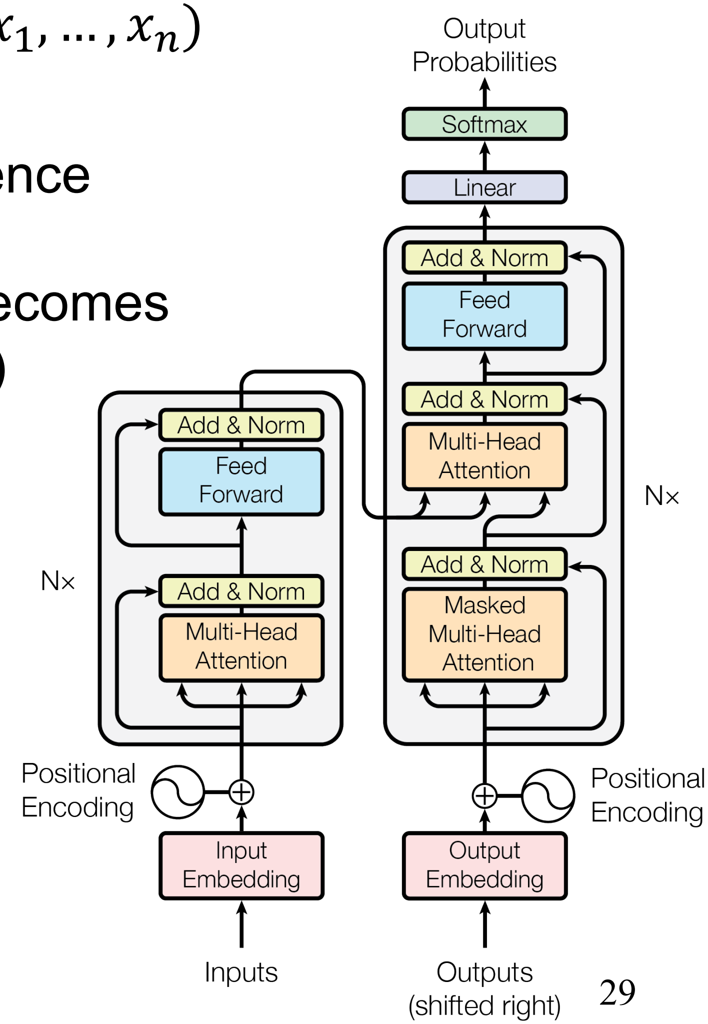
- Attention
- Self-attention
- Transformer

A. Vaswani, N. Shazeer N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser I. Polosukhin .
Attention is All you Need. NeurIPS 2017

Transformer

An encoder-decode structure

- Encoder maps an input sequence $(x_1, \ldots, x_n)$ to a latent sequence $(z_1, \ldots, z_n)$
- Decoder generates an output sequence $(y_1, \ldots, y_n)$ from $\boldsymbol{z}$ autoregressively (i.e. previously generated outputs becomes an input to generating a next output)



Transformer

Encoding

- Start initial representations for each word (unfilled circles)
- Using self-attention, it aggregates information from all of the other words, generating a new representation per word informed by the entire context (filled balls)
- This step is then repeated multiple times in parallel

Decoding

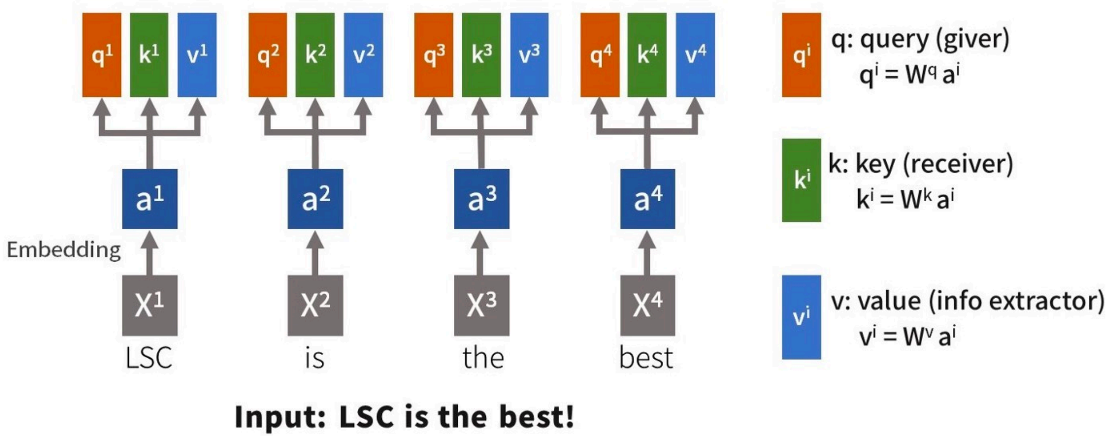
- The decoder generates one word at a time from left to right
- It attends not only to the other previously generated words, but also to the final representations by the encoder

Image Credit: https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

Self-Attention Layer

For every input word x_i

- First encode into a vector x_i and then obtain query q_i, key k_i value v_i as
- $q_i = W_q a_i$, $k_i = W_k a_i$, $v_i = W_v a_i$ where W_* are learnable parameters



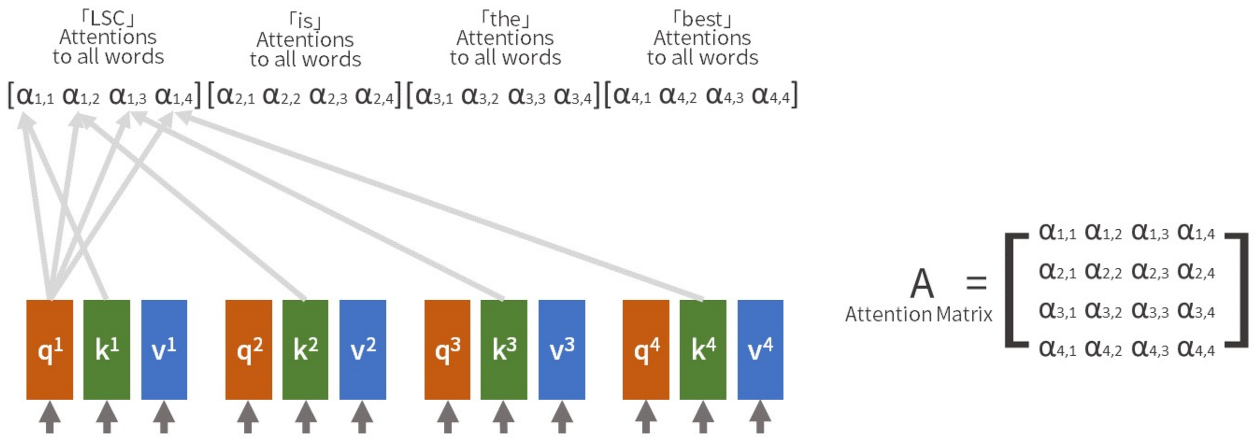
Neil Wu, Introduction of Self-Attention Layer in Transformer

Self-Attention Layer

Attention $\alpha_{i,j}$

- Defined as inner product of Query q_i and Key k_j divided by square root of its dimensions

$$\alpha_{i,j} = \text{softmax}\left(\frac{q_i \cdot k_j}{\sqrt{d}}\right)$$



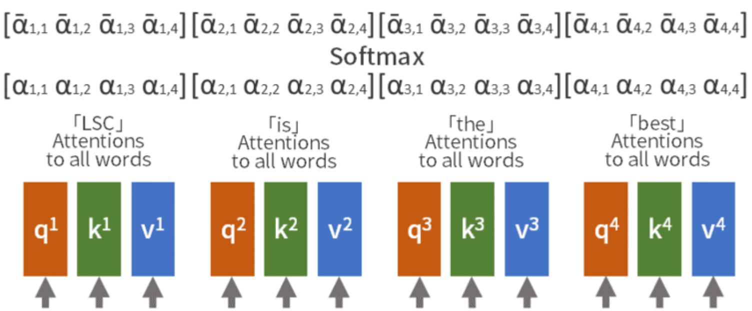
Neil Wu, Introduction of Self-Attention Layer in Transformer

Self-Attention Layer

Output b_i

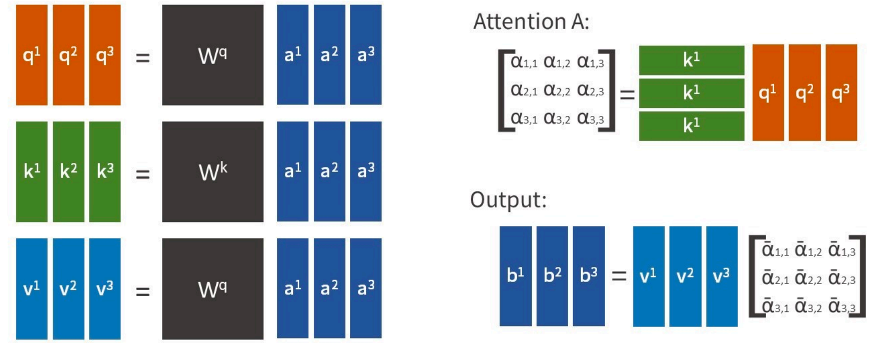
- Calculated by sum of attentions multiply by extract information from value of each word paid attention to

$$b_i = \sum_j \alpha_{i,j} \cdot v_j$$



Self-Attention Layer

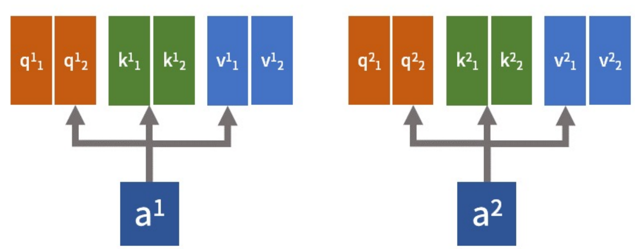
Matricize!



Multi-heads

- Create multiple Attention matrices in one layer
- By simply double the Query, Key and Value combinations, and independently calculates attention matrix

$$\underset{\text{Attention Matrix 1}}{A^1} = \begin{bmatrix} \alpha^1_{1,1} & \alpha^1_{1,2} \\ \alpha^1_{2,1} & \alpha^1_{2,2} \end{bmatrix}$$



$$\underset{\text{Attention Matrix 2}}{A^2} = \begin{bmatrix} \alpha^2_{1,1} & \alpha^2_{1,2} \\ \alpha^2_{2,1} & \alpha^2_{2,2} \end{bmatrix}$$

Neil Wu, Introduction of Self-Attention Layer in Transformer

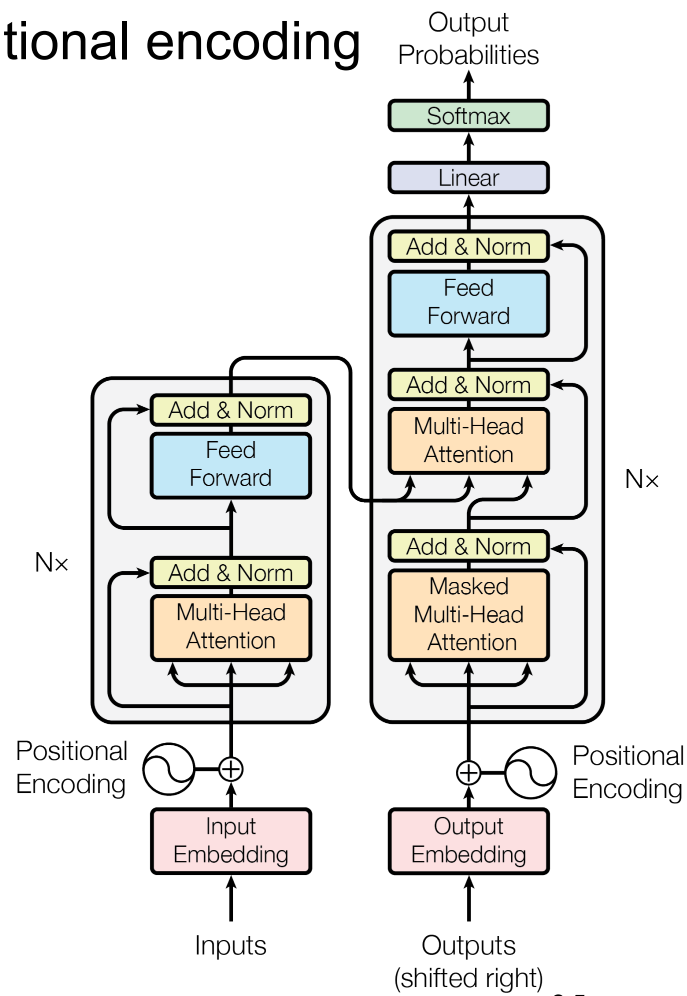
Transformer Architecture

Input

- Add the Input embedding and the positional encoding

Encoder

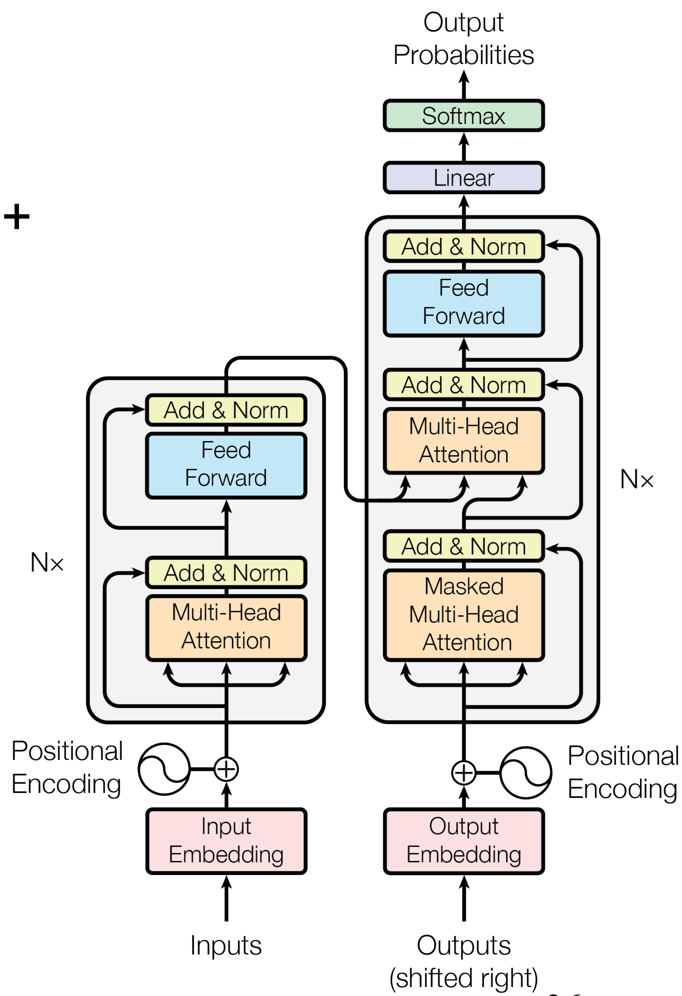
- A stack of *N* (=6) identical layers
- (1) Multi-head attention layer: already discussed!
- (2) Position-wise feedforward neural network: a little FFN has identical parameters for each position
- Dropouts and layer normalization per sublayer



Transformer Architecture

Decoder

- Similar to encoder
- A stack of N (=6) identical layers
- Decoder Input: the output Embedding + Positional Encoding up to i-1 outputs
- Masked Multi-Head Attention prevents future words to be part of the attention

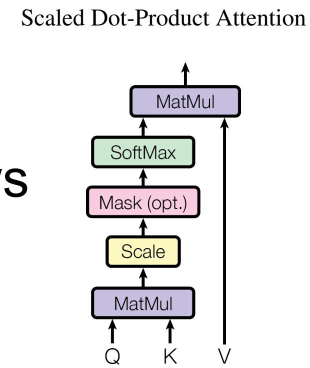


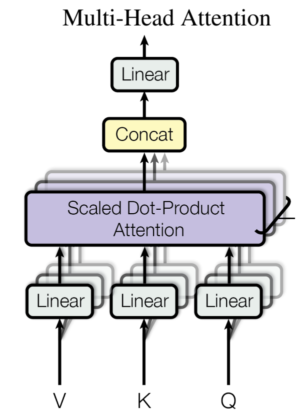
Transformer Architecture

Multi-head attention

- The original pictures are as follows

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





Position encoding

- Unlike sequential RNNs, the transformer is parallel and non-sequential
- Represent the information on the absolute (or relative) position in a sequence

$$PE_{pos,i} = \begin{cases} \sin\left(\dfrac{pos}{10000^{\frac{i}{d}}}\right) & \text{if } i \text{ is even} \\ \cos\left(\dfrac{pos}{10000^{\frac{i}{d}}}\right) & \text{if } i \text{ is odd} \end{cases}$$

where pos is the position and i is the dimension

Jay Alammar, The Illustrated Transformer