Ch08. Attention

Hwanjo Yu

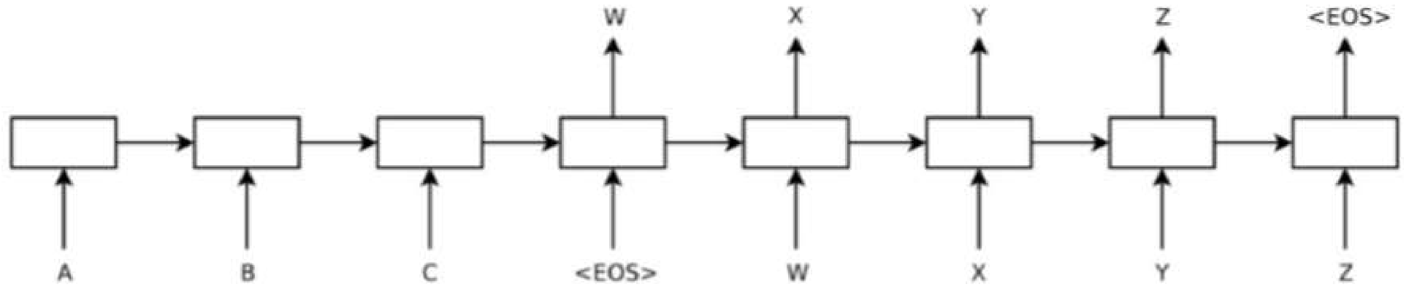
POSTECH

http://hwanjoyu.org

Attention in RNN-Encoder-Decoder

D. Bahdanua, K. Cho, and Y. Bengio (2015), ”Neural machine translation by jointly learning to align and translate,” ICLR.

RNN-Encorder-Decorder: Revisited



- Source sequence: ABC; Target sequence: WXYZ
- Computes the conditional probability

$$p\left(\mathbf{y}_1, \dots, \mathbf{y}_{T_y} | \mathbf{x}_1, \dots, \mathbf{x}_{T_x}\right) = \prod_{t=1}^{T_y} p(\mathbf{y}_t | \mathbf{c}, \mathbf{y}_1, \dots, \mathbf{y}_{t-1}) = \prod_{t=1}^{T_y} g(\mathbf{y}_{t-1}, \mathbf{s}_t, \mathbf{c})$$

- where is $\mathbf{s}_t$ the hidden state of the decoder, $\mathbf{c}$ is the fixed-dimensional representation of the input sequence $(\mathbf{x}_1, \dots, \mathbf{x}_{T_x})$, given by the last hidden state of the encoder,

$$\mathbf{h}_t = f(\mathbf{x}_t, \mathbf{h}_{t-1}),$$
$$\mathbf{c} = q\left(\mathbf{h}_1, \dots, \mathbf{h}_{T_x}\right) = \mathbf{h}_{T_x}.$$

Attention in RNN-Encoder-Decoder

- Compute the probability over the target sequence:

$$p\left(\mathbf{y}_1, \dots, \mathbf{y}_{T_y} | \mathbf{x}_1, \dots, \mathbf{x}_{T_x}\right) = \prod_{t=1}^{T_y} p(\mathbf{y}_t | \mathbf{c}, \mathbf{y}_1, \dots, \mathbf{y}_{t-1})$$

$$= \prod_{t=1}^{T_y} g(\mathbf{y}_{t-1}, \mathbf{s}_t, \mathbf{c}_t)$$

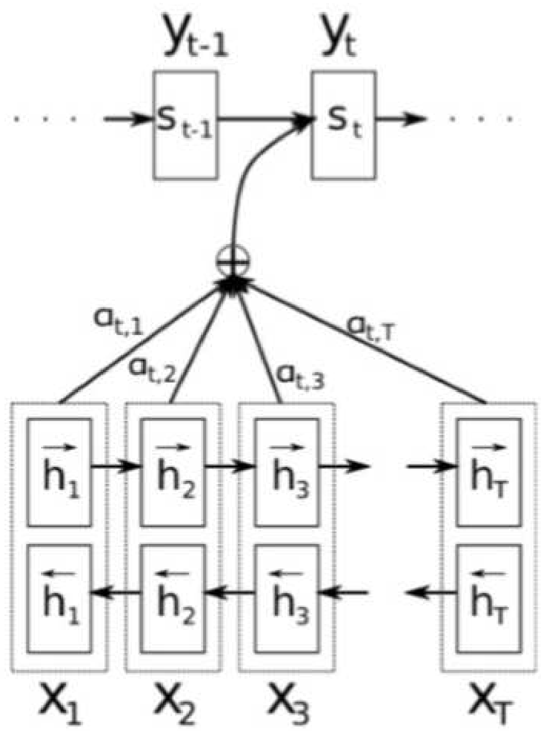
- where is $\mathbf{s}_t$ the hidden state of the decoder, computed by

$$\mathbf{s}_t = f(\mathbf{s}_{t-1}, \mathbf{y}_{t-1}, \mathbf{c}_t),$$

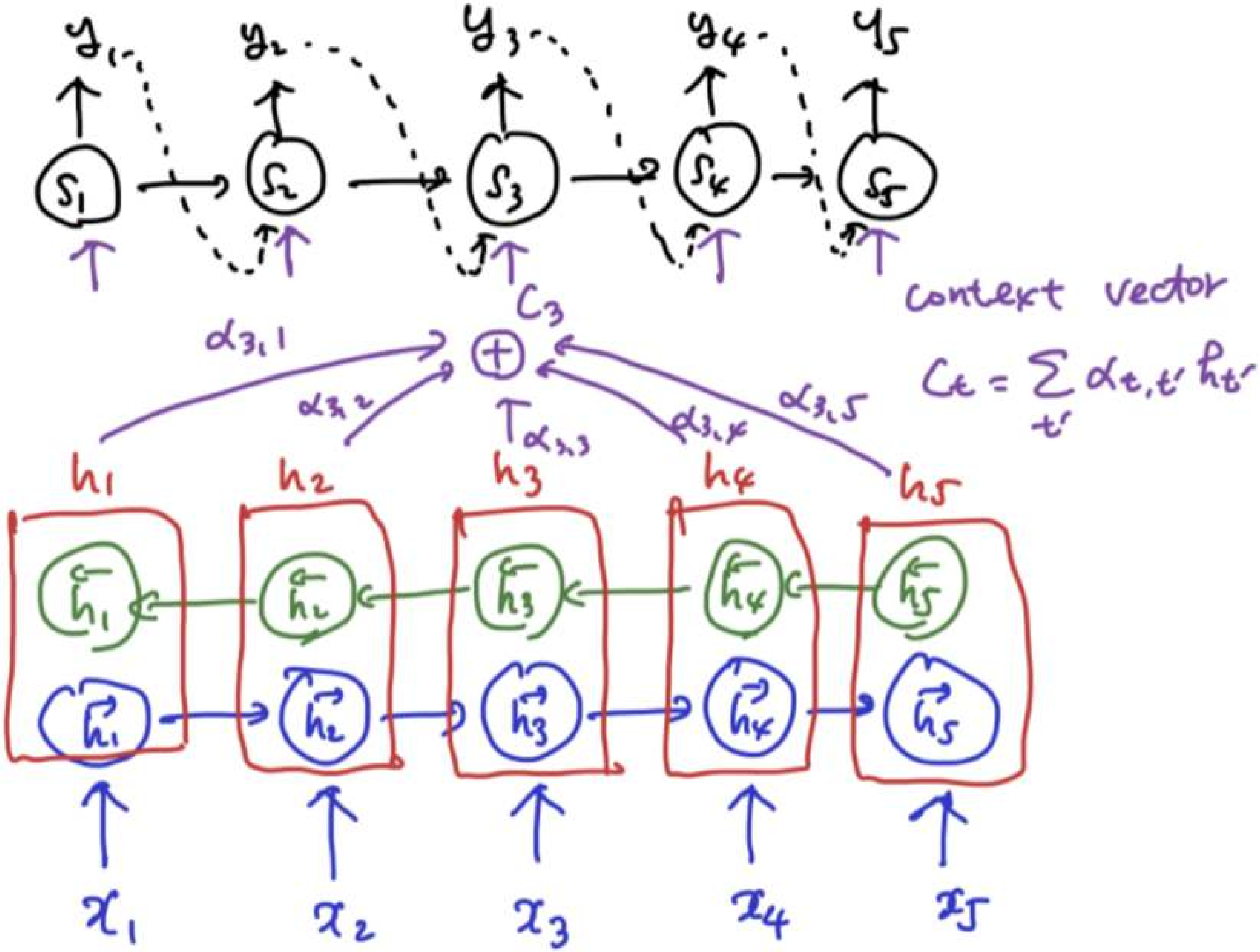
- and $\mathbf{c}_t$ the context vector, computed by a weighted sum of encoder hidden states $\{\mathbf{h}_t\}$:

$$\mathbf{c}_t = \sum_{t'=1}^{T_x} \alpha_{t,t'} \mathbf{h}_{t'} .$$

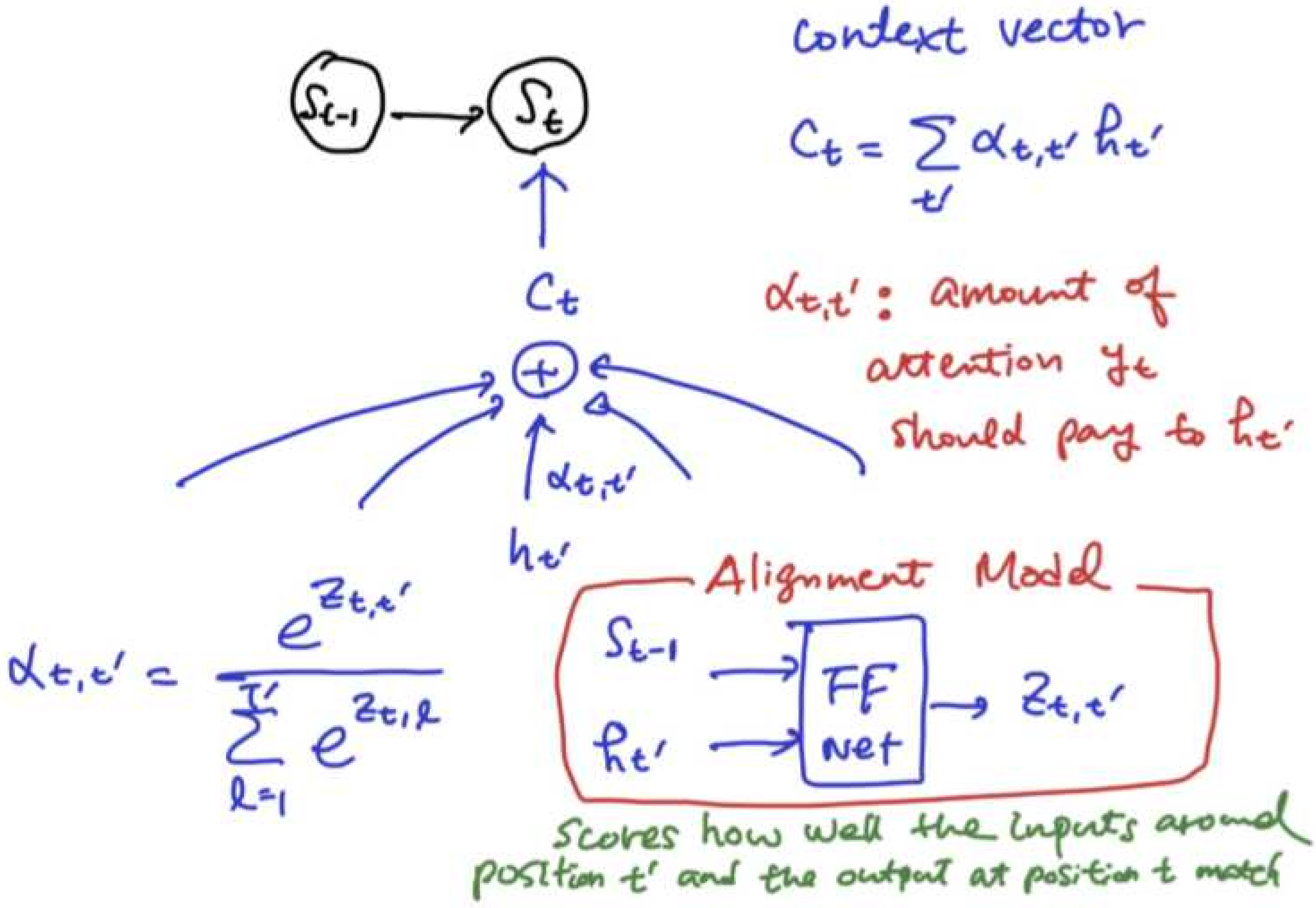
- Note that in the previous model, we have $\mathbf{c}_1 = \mathbf{h}_{T_x}$ and $\mathbf{c}_t = 0$ for $t = 2, \dots, T_y$.



Attention in RNN-Encoder-Decoder



Attention in RNN-Encoder-Decoder



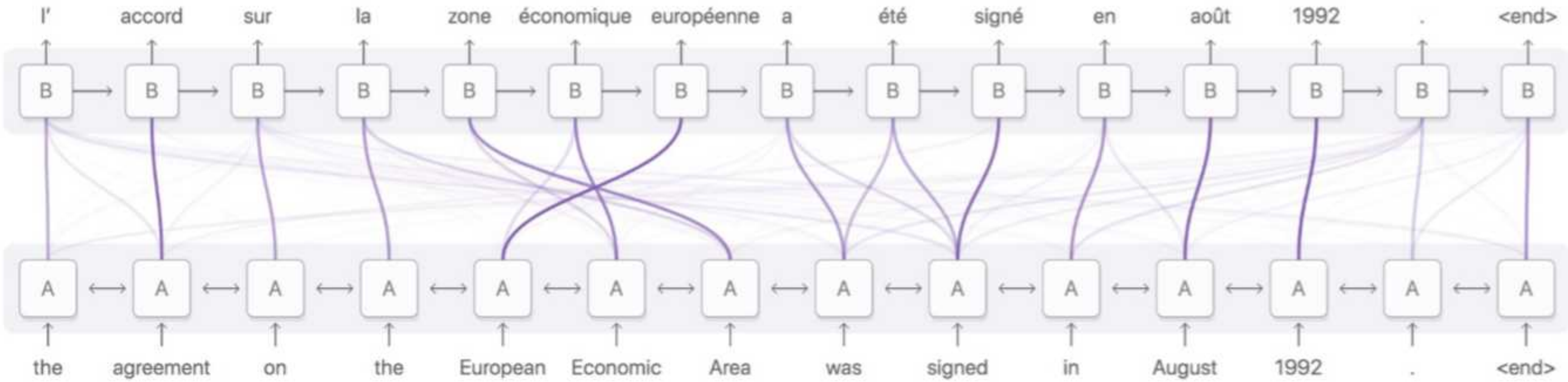


Diagram derived from Fig. 3 of Bahdanau, *et al.* 2014

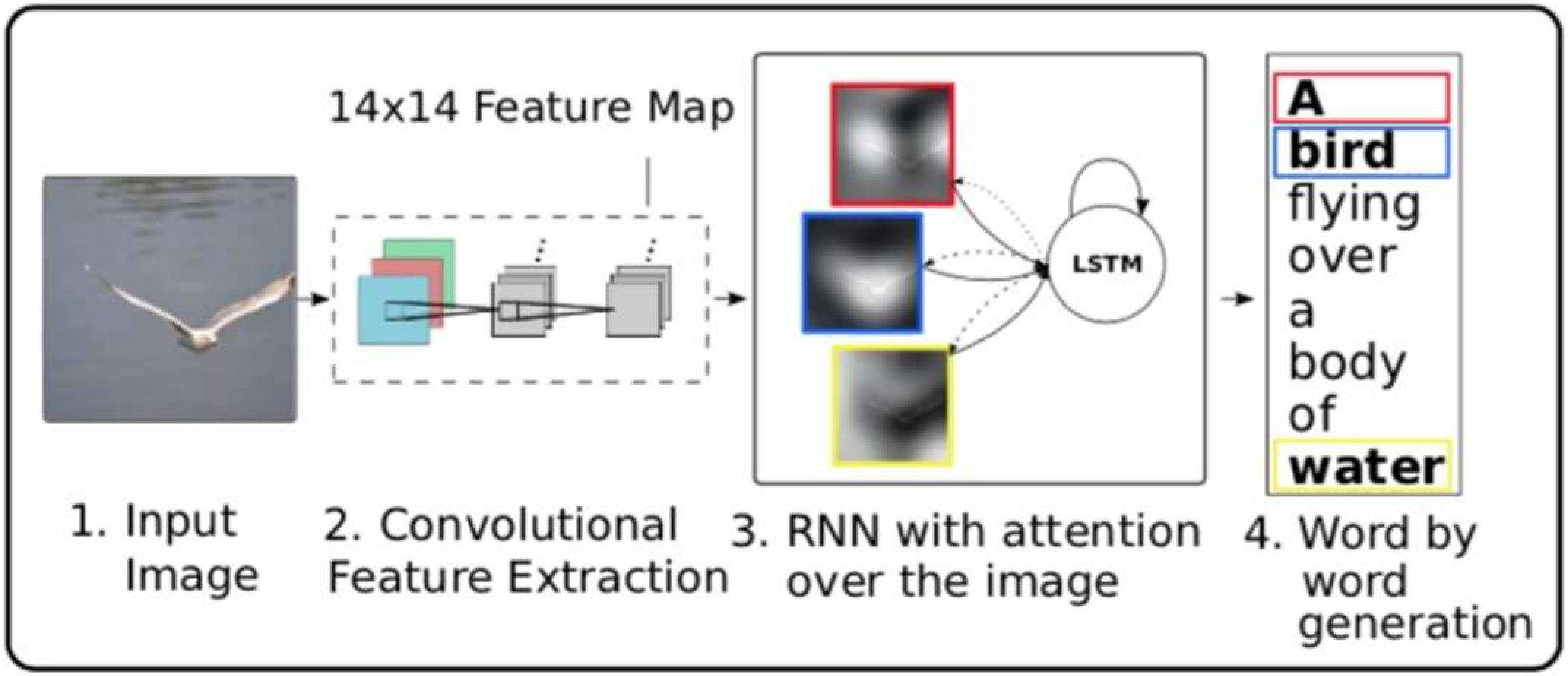
Visual Attention

A. Karpathy and L. Fei-Fei (2015), ”Deep Visual-Semantic Alignments for Generating Image Descriptions,” CVPR.

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

Learning words-image alignment

- Input: Raw image
- Output: A sequence of C words from vocabulary of size K, $\{\mathbf{y}_1, \ldots, \mathbf{y}_C\}, \mathbf{y}_i \in \mathbb{R}^K$.



Visual Attention

- **Encoder:** Use a CNN to extract a set of D-dimensional feature vectors, referred to as **annotation vectors**:

$$\mathbf{a} = [\mathbf{a}_1, \dots, \mathbf{a}_L] \in \mathbb{R}^{L \times D}$$

- which contains the output $L_1 \times L_2 \times D$ $(L = L_1 \times L_2)$ of a lower convolutional layer (before max pooling)

- **Decoder:** Use a **RNN with attention modules** to produce a caption generating one word every time step conditioned on a context vector, the previous hidden state, and the previously generated words.

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.

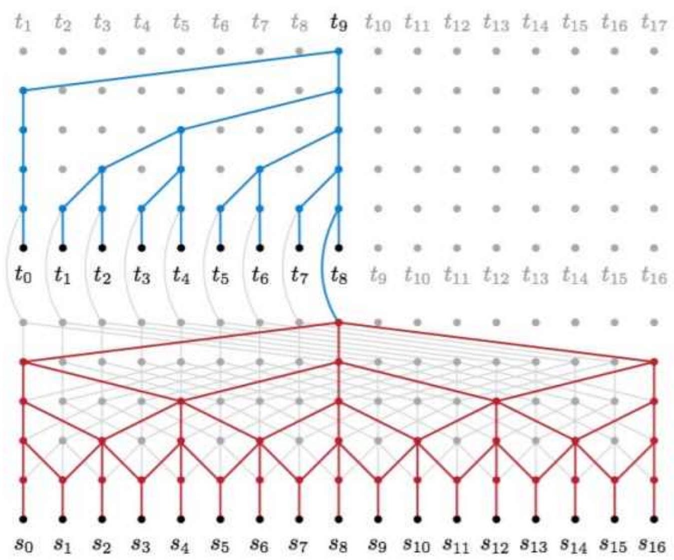
A group of people sitting on a boat in the water.

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

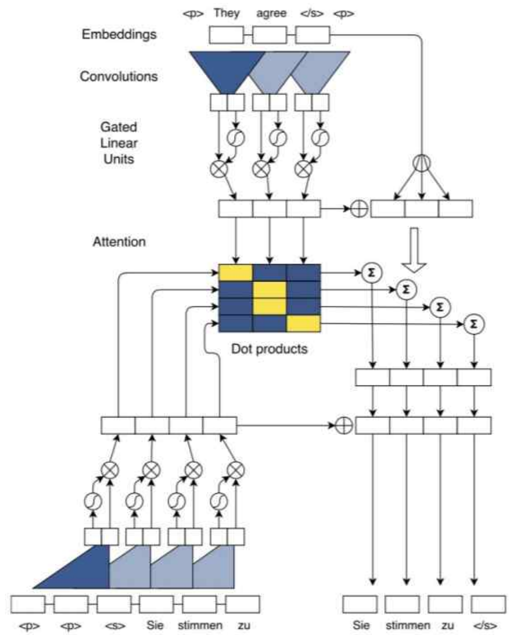
Google Transformer

A. Vaswani et al. (2017), ”Attention is all you need,” NIPS.

- RNN: Sequential computations (autoregressive models in decoders) are expensive and are not easy to be parallelized.
    - N. Kalchbrenner et al. (2017), ”Neural machine translation in linear time,” arXiv:1610.10099.
- CNN (ByteNet, ConvS2S): Need multiple layers to catch long-term dependencies
    - J. Gehring et al. (2017), ”Convolutional sequence to sequence learning,” arXiv:1705.03122



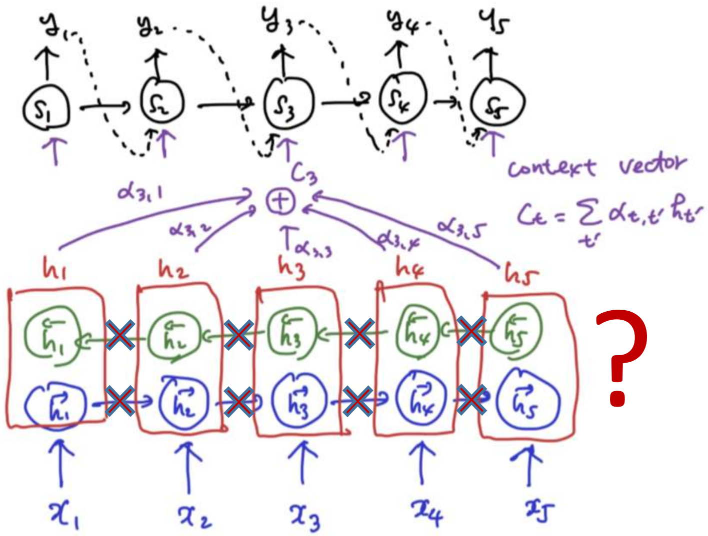
(a) ByteNet



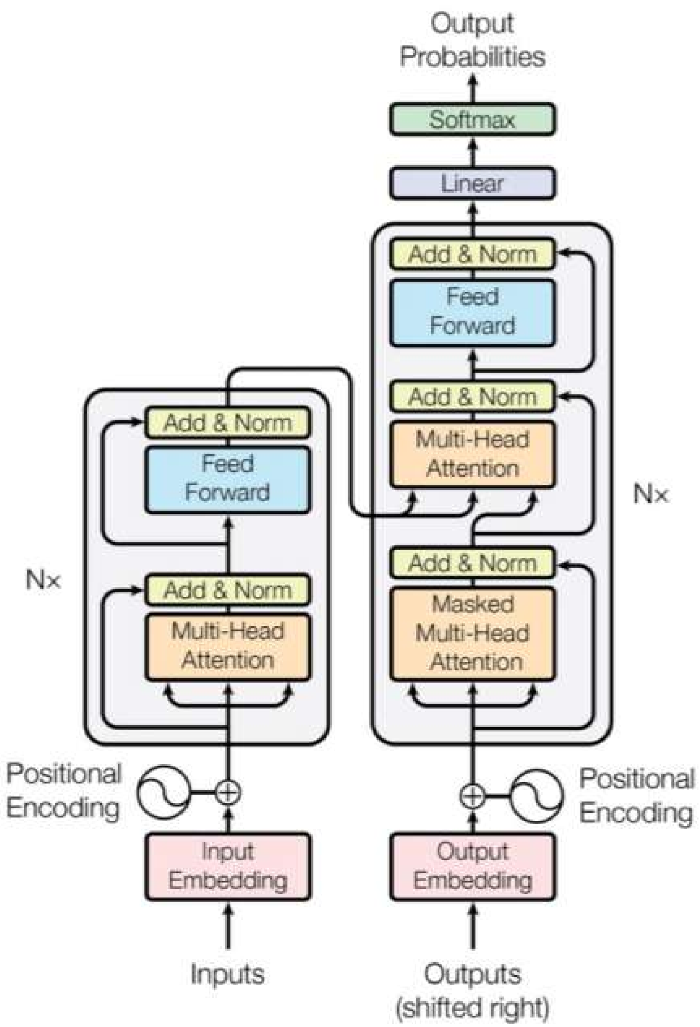
(b) ConvS2S

Self-attention

- Disconnect recurrent links (to facilitate parallel processing). Instead, use **self-attention with positional encoding**.



Transformer

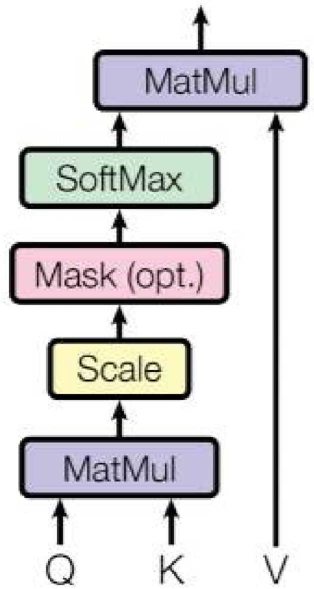


- Transformer = encoder + decoder
- Encoder or decoder = attention + positional encoding + feedforward net

Scaled Dot-Product Attention

- Queries: $\mathbf{Q} \in \mathbb{R}^{N \times D_k}$ (N: # of words; D_k: dimension size)
  - $\mathbf{Q} = (\mathbf{X} + \mathbf{P}) * \mathbf{W}_Q \in \mathbb{R}^{D_{model} \times D_k}$ (D_{model}: dimension size of $\mathbf{X}$)
    - $\mathbf{X}$: input
    - $\mathbf{P}$: positional encoding (explained later)
    - $\mathbf{W}_Q$: learning parameter
- Keys: $\mathbf{K} \in \mathbb{R}^{N \times D_k}$
  - $\mathbf{K} = (\mathbf{X} + \mathbf{P}) * \mathbf{W}_K$
- Values: $\mathbf{V} \in \mathbb{R}^{N \times D_v}$
  - $\mathbf{V} = (\mathbf{X} + \mathbf{P}) * \mathbf{W}_V$
- $\mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{D_k}}\right)\mathbf{V}$
- $\mathbf{Q}\mathbf{K}^{\mathrm{T}}$: kind (but not exactly) of pair-words similarity matrix (note that $\mathbf{Q} \neq \mathbf{K}$)
- $\mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$: $N \times D_v$ matrix where each row is a word containing the context within the sentence.
- Self-attention: Queries, keys, and values are from the same word sequence (but with different embeddings, as $\mathbf{W}_Q \neq \mathbf{W}_K \neq \mathbf{W}_V$).

Scaled Dot-Product Attention



Multi-Head Attention

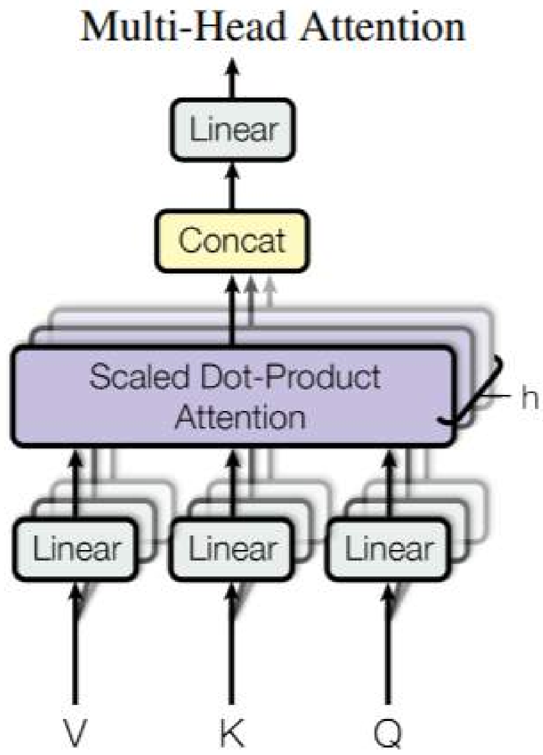
- Allows the model to jointly attend to information from different representation subspaces at different positions:

$$\mathrm{MultiHead}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_h)\mathbf{W}^o,$$
$$\mathrm{head}_i = \mathit{Attention}\left(\mathbf{Q}(\mathbf{W}_i^Q), \mathbf{K}(\mathbf{W}_i^K), \mathbf{V}(\mathbf{W}_i^V)\right),$$

- where linear projections are done via parameters matrices

$$\mathbf{W}_i^Q \in \mathbb{R}^{D_{model} \times D_k}, \qquad \mathbf{W}_i^K \in \mathbb{R}^{D_{model} \times D_k},$$
$$\mathbf{W}_i^V \in \mathbb{R}^{D_{model} \times D_v}, \qquad \mathbf{W}_i^O \in \mathbb{R}^{hD_v \times D_{model}},$$

- where $h = 8$ is the number of parallel attention layers.



Position-wise FFN, Layer normalization

- Applied to each position separately and identically (with same $\mathbf{W}$).
- FFN with single hidden layer

$$FFN\,(\mathbf{x}) = \max(0, \mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\,\mathbf{W}_2 + \mathbf{b}_2$$

- After that, apply layer normalization

Attention in Encoder and Decoder

- Encoder:
  - Contains self-attention layers.
  - Each position in the encoder can attend to all positions in the previous layer of the encoder.
- Decoder:
  - Contains self-attention layers.
  - Each position in the decoder can attend to all positions in the decoder up to and including that position
  - Shift to the right by one position in decoder input. (Predict $\mathbf{y}_i$ from $\mathbf{y}_0$ to $\mathbf{y}_{i-1}$.)
- Encoder-decoder attention:
  - Queries come from the previous decoder layer and keys/values come from the output of the encoder.
  - Allows every position in the decoder attend over all positions in the input sequence.
  - Mimics the encoder-decoder attention in seq2seq learning.
- Training
  - Use teacher-forcing.
- Inference
  - Predict the first word ($\mathbf{y}_1$) from input of <start> ($\mathbf{y}_0$) token.
  - Predict $\mathbf{y}_2$ from input of $\mathbf{y}_0, \mathbf{y}_1$
  - And so on..

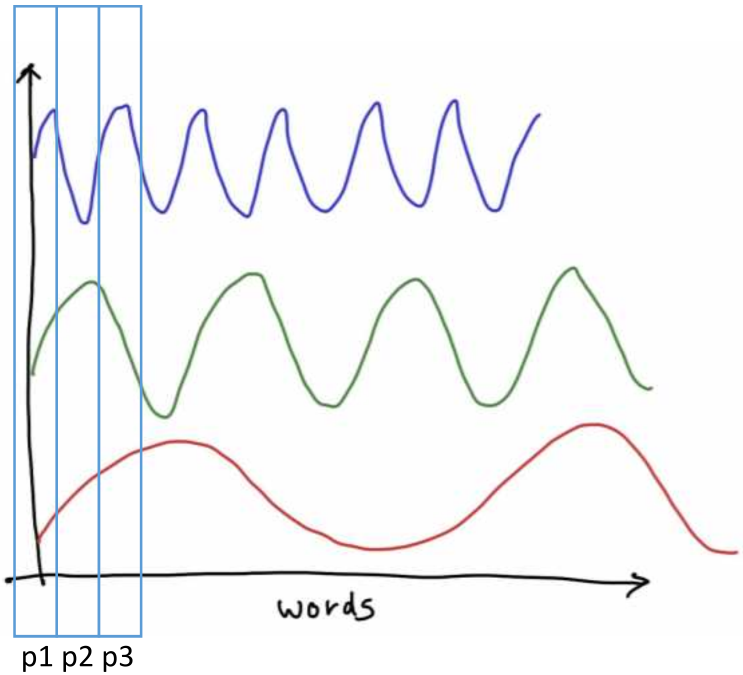
Positional Encoding

- Inject some information about the relative or absolute position of the tokens in the sequence.
- Add ”positional encodings” to the input embeddings.

$$\mathrm{PE}_{pos,2i} = \sin\left(\frac{pos}{10000^{2i/D_{model}}}\right),$$
$$\mathrm{PE}_{pos,2i+1} = \cos\left(\frac{pos}{10000^{2i/D_{model}}}\right).$$

For example,

- p1 = (mid, low, low)
- p2 = (low, high, mid)
- p3 = (high, mid, high)
- …



The encoder self-attention distribution for the word "it" from the 5th to the 6th layer of a Transformer trained on English to French translation (one of eight attention heads).

[Figure source: https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html]