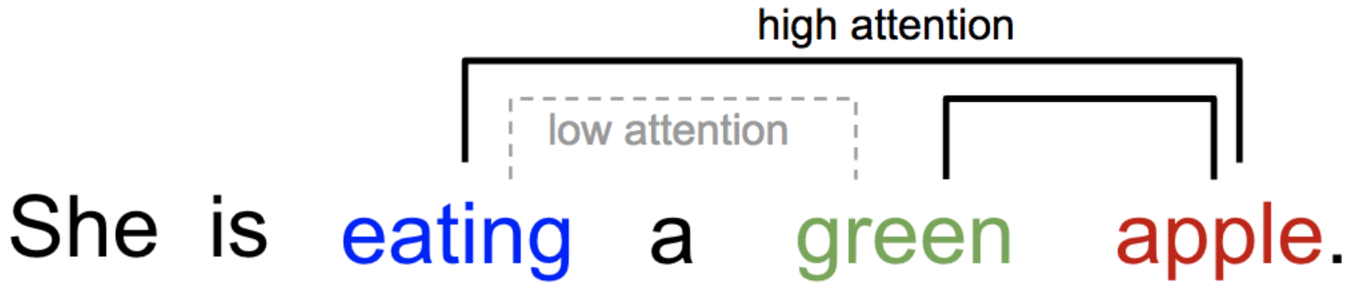
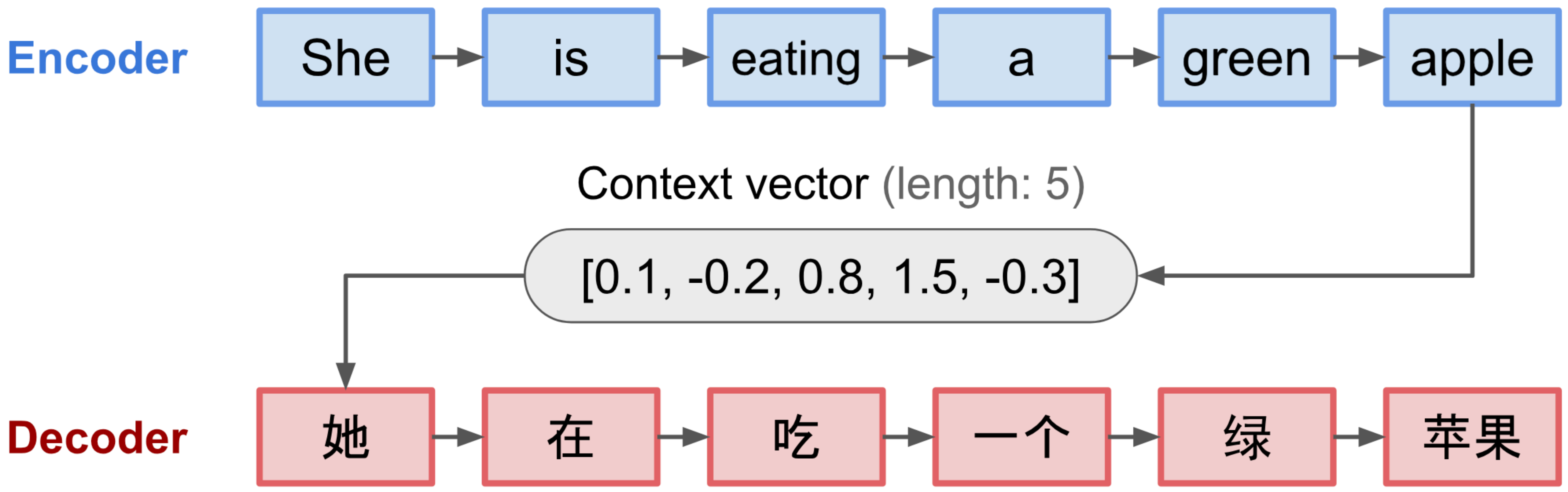
Attention RNNs and Transformers

Perumadura De Silva

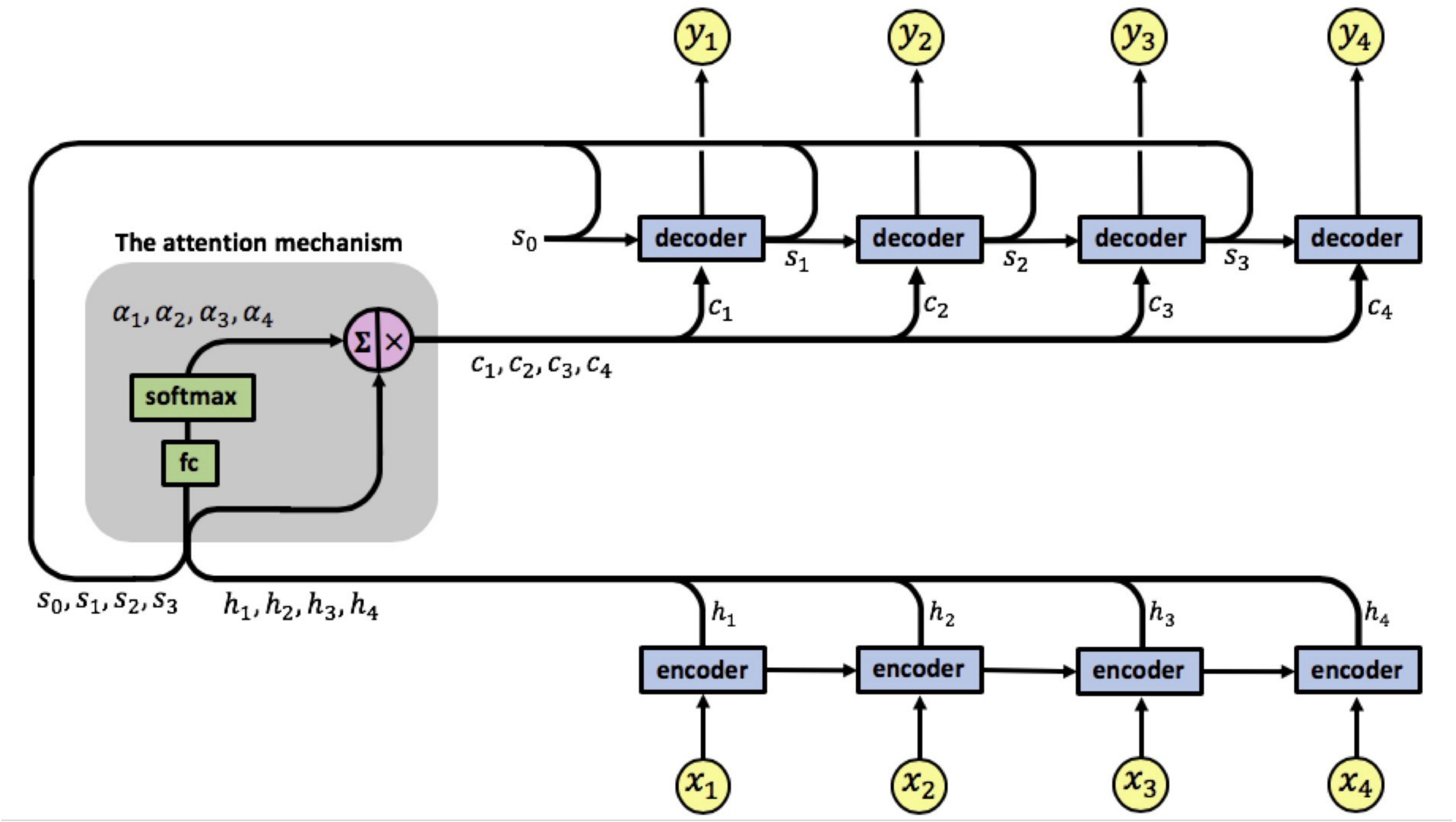
Attention RNN
Introduction



Ref.1: Attention in a sentence



Ref.1: Typical Seq2Seq encoder decoder model



Ref.2: Attention RNN with additive attention

Attention RNN

Encoder and decoder

- Encoder: Bi-Directional RNN/LSTM
  - Input: Source sentence of length N; $X = \{x_1, x_2, \ldots x_N\}$
  - Hidden state: $h = [h^{\Leftarrow}; h^{\Rightarrow}]$
- Decoder: RNN/LSTM
  - Output: Target sentence of length M; $Y = \{y_2, y_3, \ldots., y_M\}$
  - Input: Context vector and target sentence $[C; Y]$; $Y' = \{y_1, y_2, \ldots y_{M-1}\}$
  - Hidden state: $S = \{s_1, s_2, \ldots. s_M\}$

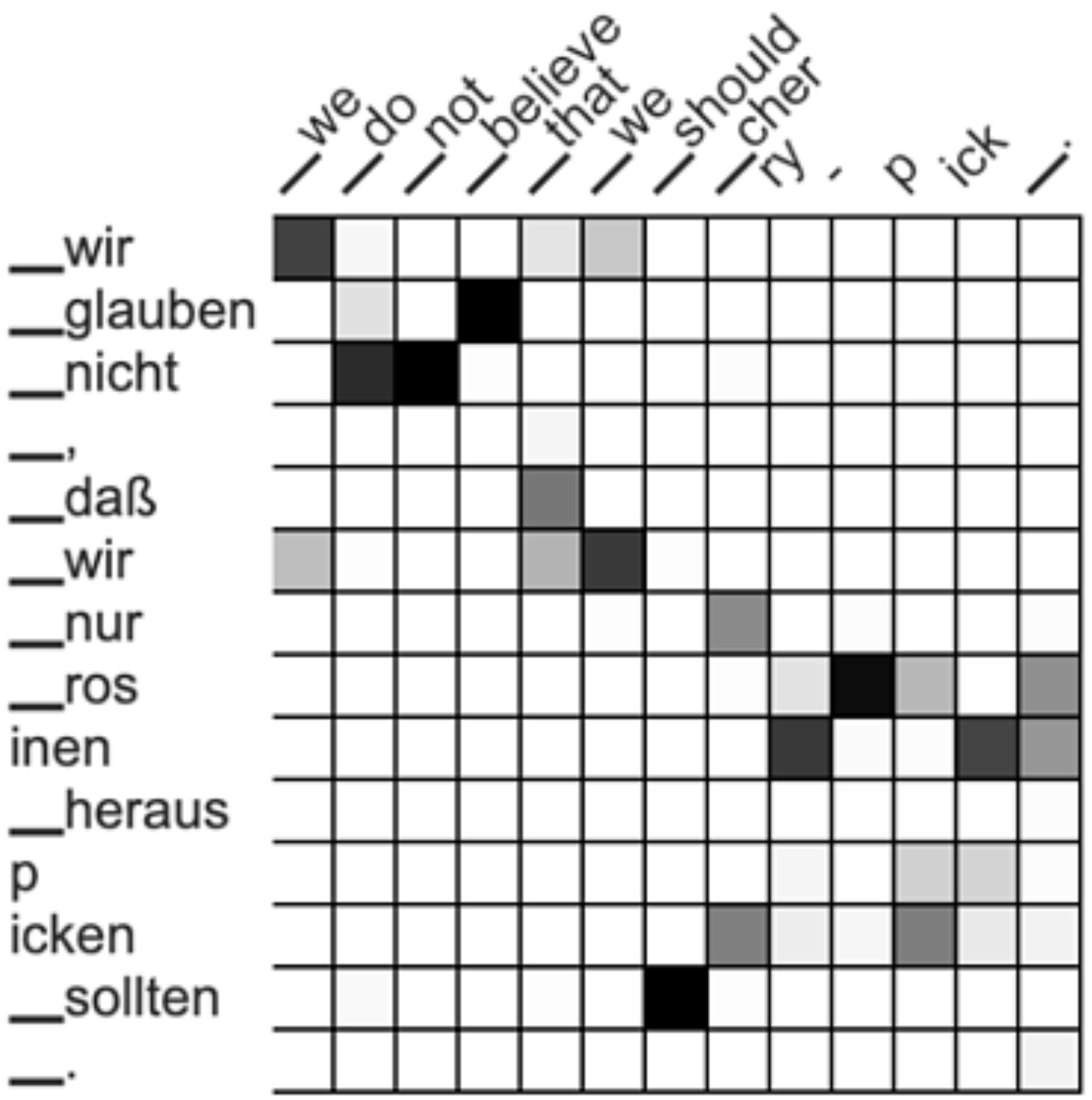
Attention RNN
Algorithm for context computation

1. Given the previous decoder state s_{t-1} and encoder hidden state h
   - $score(s_{t-1}, h_i) = v_a^T tanh(W_a[s_{t-1}; h_i]) = e_{ti}$
   - v_a and W_a are trainable weights
2. Compute the alignment α
   - $\alpha_{ti} = \frac{exp(e_{ti})}{\sum_{k=1}^{N} exp(e_{tk})}$; Softmax taken across the input time axis
3. Compute the context vector for time step t
   - $c_t = \sum_{i=1}^{N} \alpha_{ti} h_i$

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE by Dzmitry Bahdanau

Attention RNN

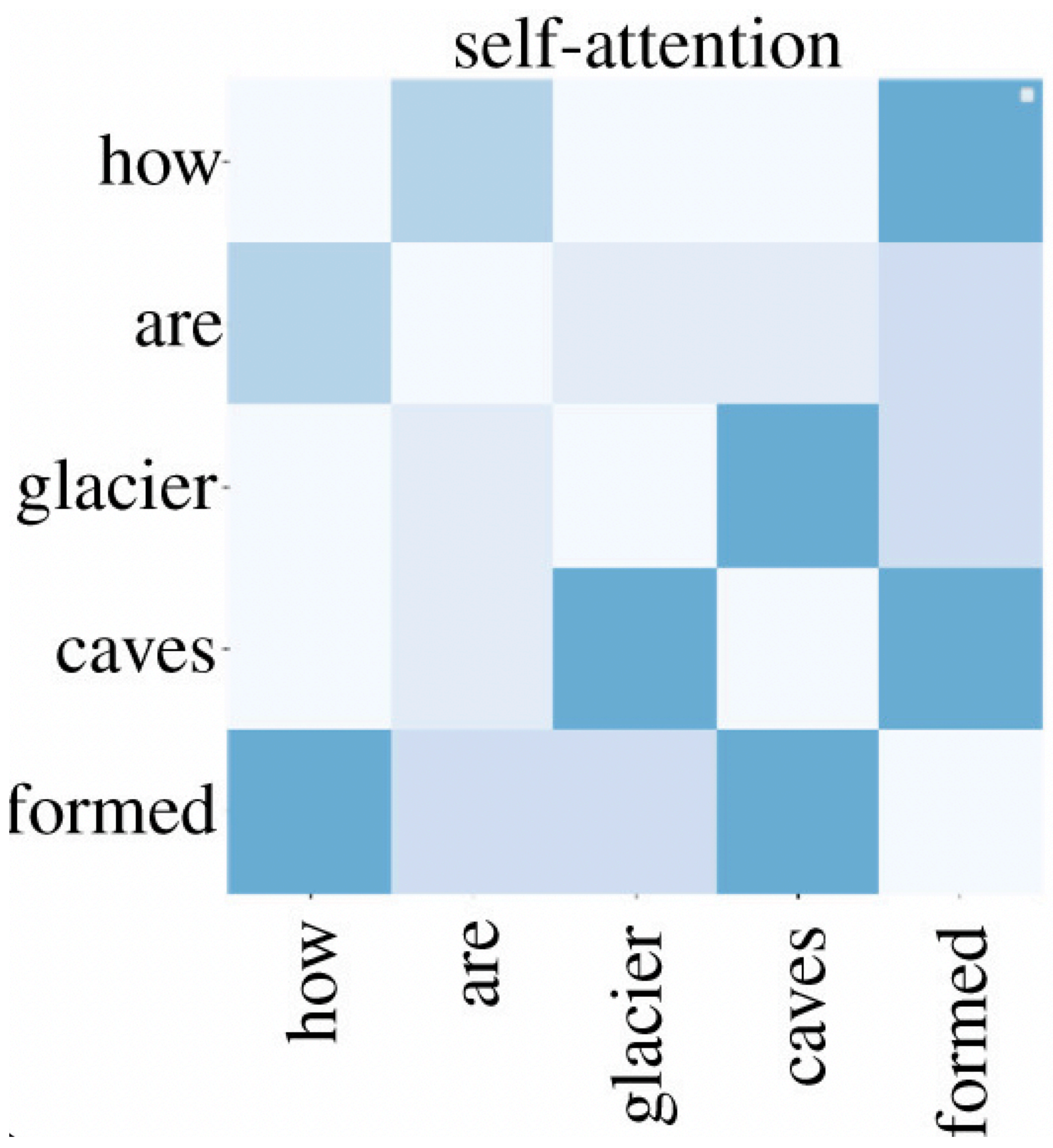
Attention Matrix



Ref-3: Additive/Bahdanau attention computes an alignment between a source sentence and output word

Transformers
Self attention

- Transformers built on top of the idea of self attention
  - How each word in a sentence related to each other
  - Self attention matrix is used for:
    - classification
    - autoregressive
    - translation tasks

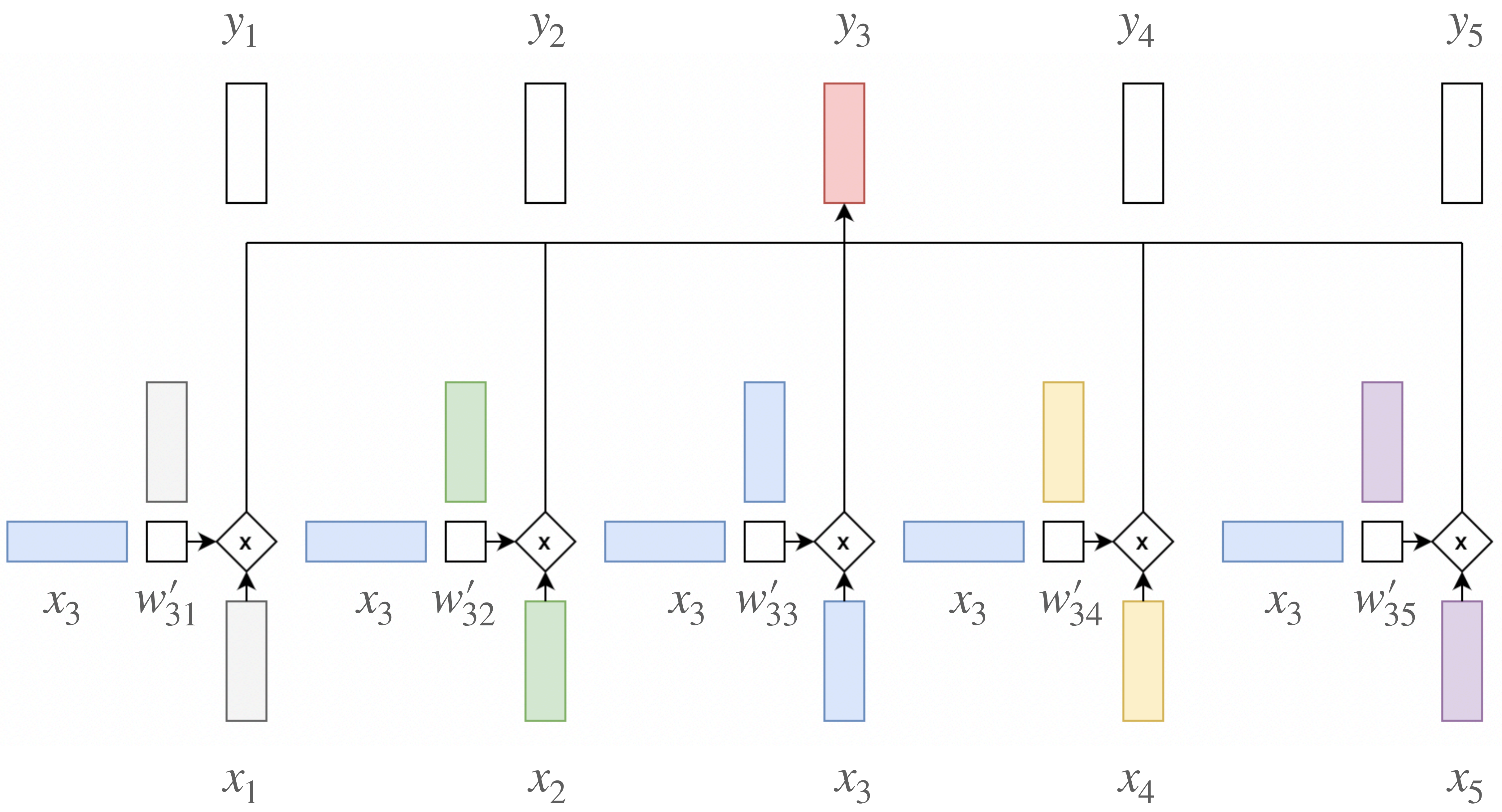


Ref-4: Self attention matrix

Transformers

Self attention

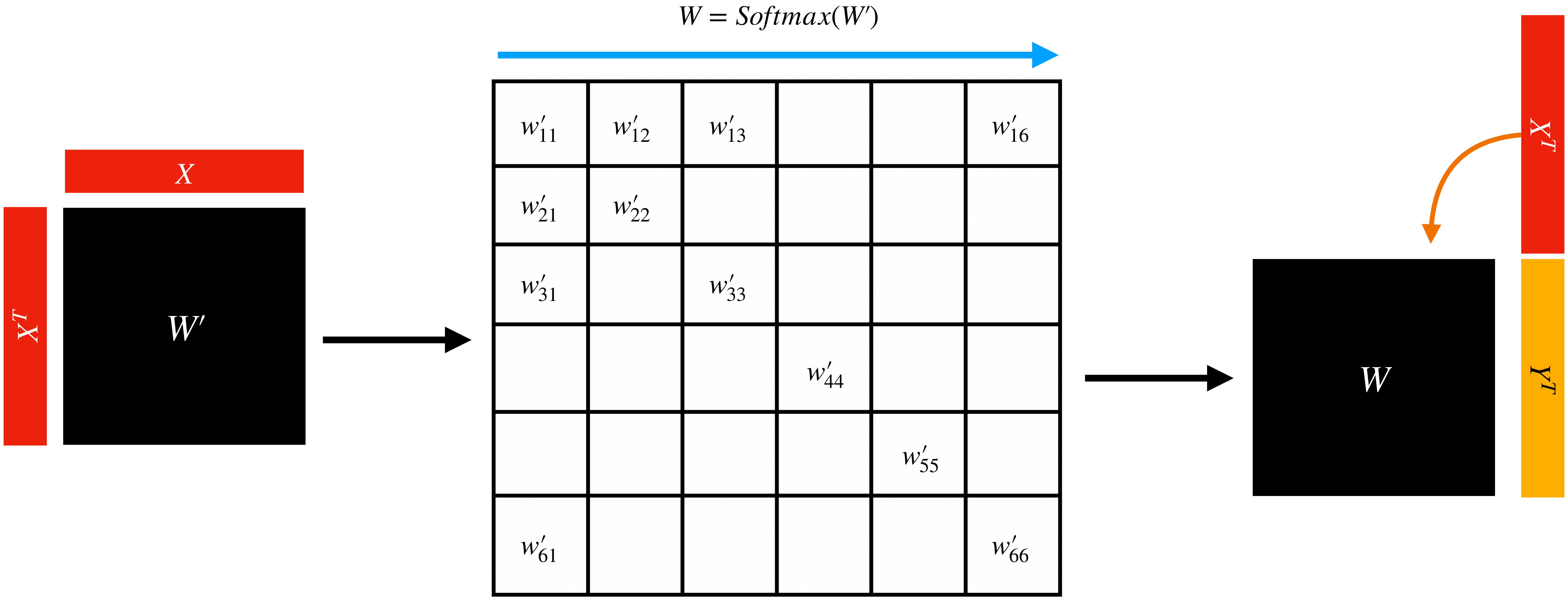
- Calculate how one input relate to the other
  - $w'_{ij} = x_i^T x_j$
  - $w_{ij} = \frac{exp(w'_{ij})}{\sum_j w'_{ij}}$ Attention score
- Finally calculate the output
  - $y_i = \sum_j w_{ij} x_j$



Reference: https://www.youtube.com/watch?v=KmAISyVvE1Y

Transformers

Self attention



Reference: https://www.youtube.com/watch?v=KmAISyVvE1Y

Transformers

Properties

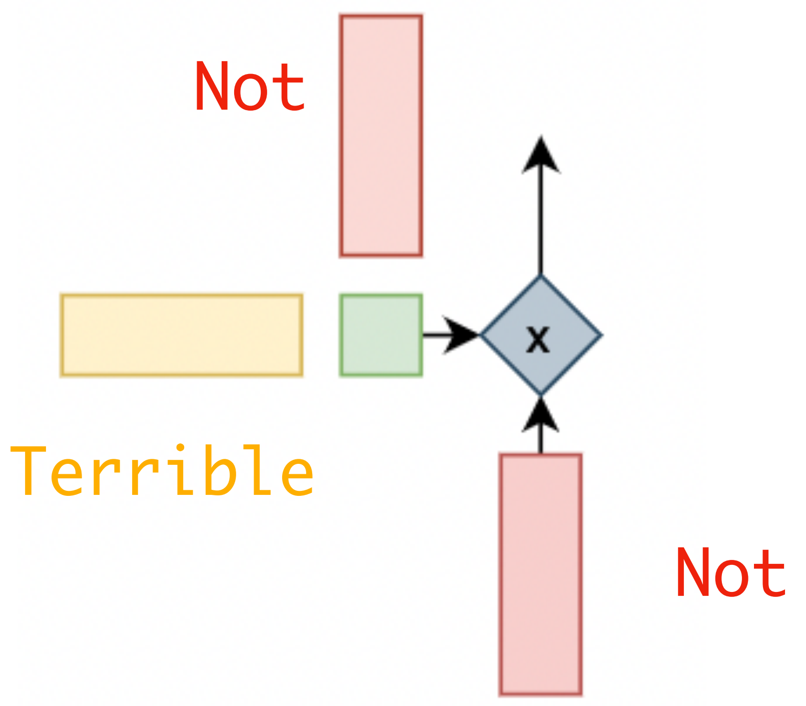
- Simple self attention has no parameters:
    - Only the input embeddings control the self attention
- The self weights (w_{ii}) are the strongest
- A linear operation between X and Y
- Can look far back in the sequence:
    - More computations in RNNs when attending early steps
- A non-sequential model; No information about the sequence order
- Permutation Equivariant: $Perm(SA(X)) = SA(Perm(X))$; $Perm$: permutation, SA: Self attention

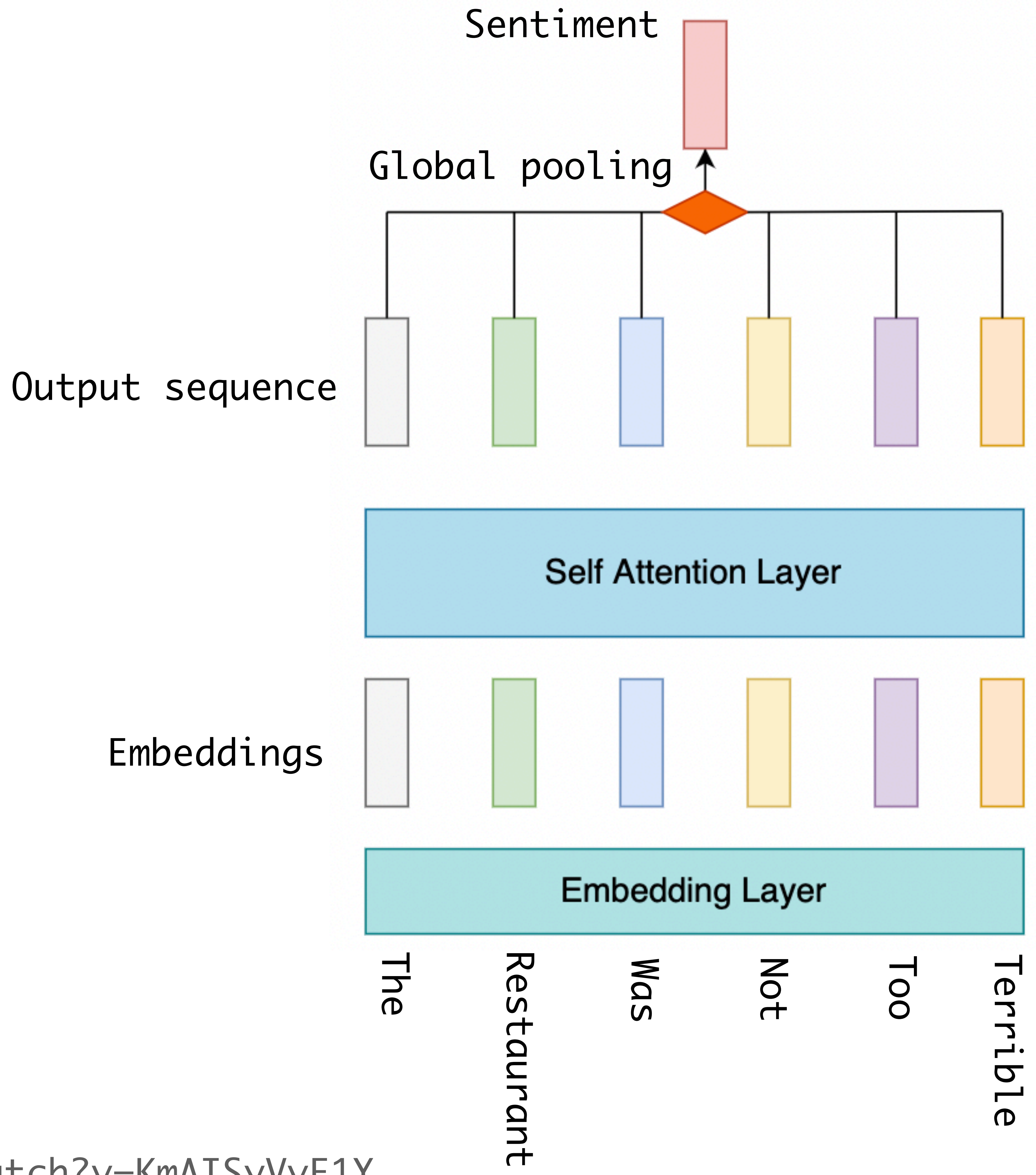
Reference: https://www.youtube.com/watch?v=KmAISyVvE1Y

Transformers

Self attention sentiment analysis

- Take the following sentence:
  - This restaurant was not too terrible
  - Score between not and terrible will contribute positive to the sentiment





Reference: https://www.youtube.com/watch?v=KmAISyVvE1Y

Transformers

Modifications to self attention

1. Scaled self attention:
   - When the input dimension is large; $x_i^T x_j$ will be large
   - So scale by dimension size k: $w'_{ij} = \frac{x_i^T x_j}{\sqrt{k}}$
2. Key (k) Query (q) and value (v) transformations:
   - The idea comes from information retrieval

Reference: https://www.youtube.com/watch?v=KmAISyVvE1Y

Transformers

Modifications to self attention

- Consider the following python dictionary d

$d = \left\{ \begin{array}{c} a : car \\ b : bicycle \\ c : airplane \end{array} \right\}$; where a, b, c are the keys and $1,2,3$ are the values

When we query the dictionary for a key c, we get a hard value; $d['c'] = airplane$

We can write the probability of getting each value as:

$$d = \left\{ \begin{array}{c} a : 0 \\ b : 0 \\ c : 1 \end{array} \right\}$$

Now if we multiply the probabilities by a value vector $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} car \\ bicycle \\ airplane \end{bmatrix}$ we get $\begin{bmatrix} 0 \\ 0 \\ airplane \end{bmatrix}$

Transformers

Modifications to self attention

- Taking the same idea of python dictionary (hard); we can retrieve a soft value in attention

  $Output = Softmax\left(\frac{QK^T}{\sqrt{k}}\right)V$; Where Q, K and V are the transformed embeddings

  where the $Softmax\left(\frac{QK^T}{\sqrt{k}}\right)$ gives the probabilities

  Since we are taking the softmax we get the soft values; e.g $\begin{bmatrix} \frac{1}{6}car \\ \frac{1}{6}bicycle \\ \frac{2}{3}airplane \end{bmatrix}$

Transformers

Modifications to self attention

- In order to do this transformations of the embeddings an MLP can be used:
  - $k_i = W_k x_i + b_k$
  - $q_i = W_q x_i + b_q$
  - $v_i = W_v x_i + b_v$

3. Multi-Headed attention:
   - Idea: Different words has different relationships to each other

     Inverts Moderate

   - E.g: This restaurant was not too terrible

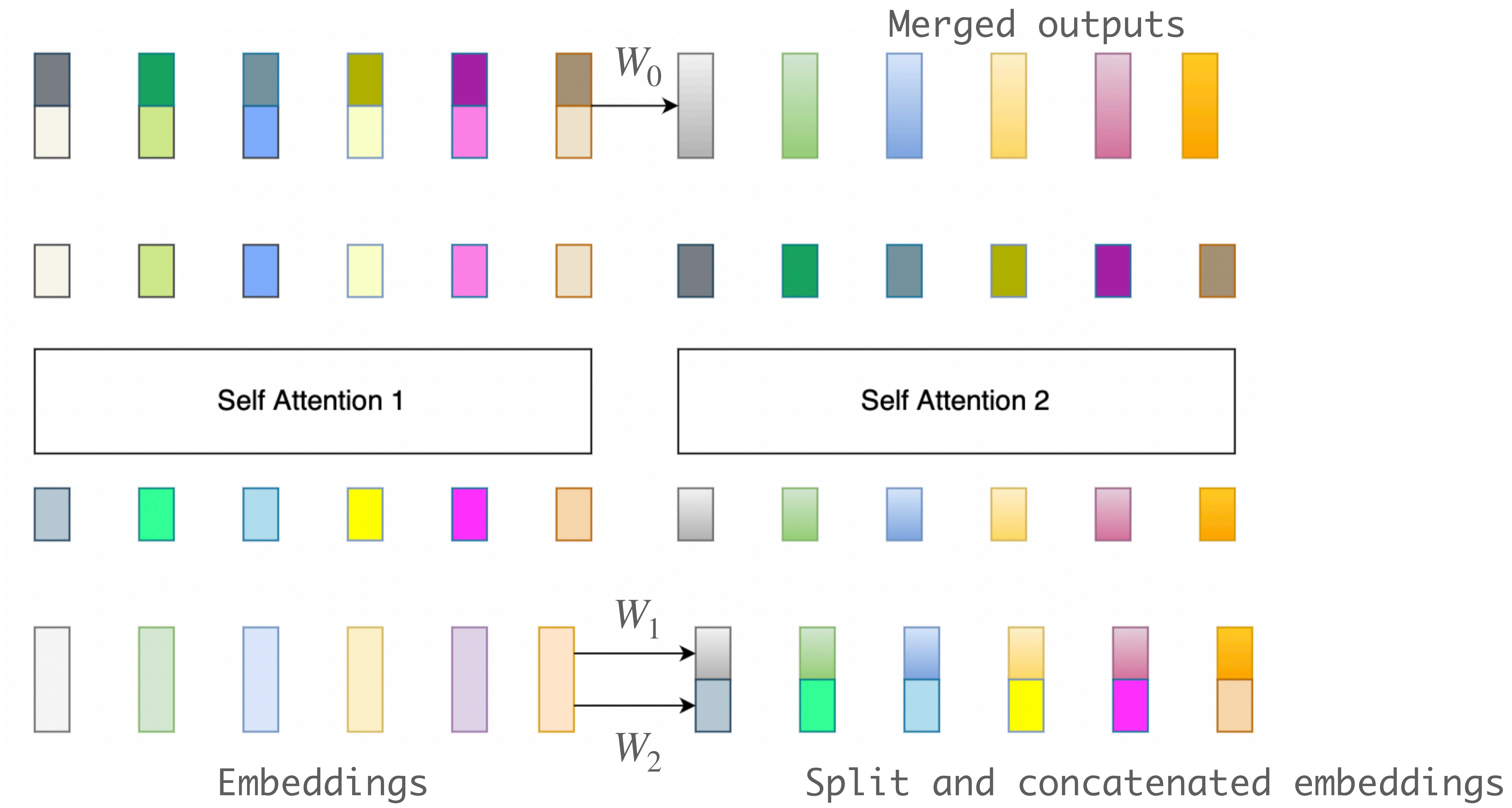
     Property of

Reference: https://www.youtube.com/watch?v=KmAISyVvE1Y

Transformers

Modifications to self attention

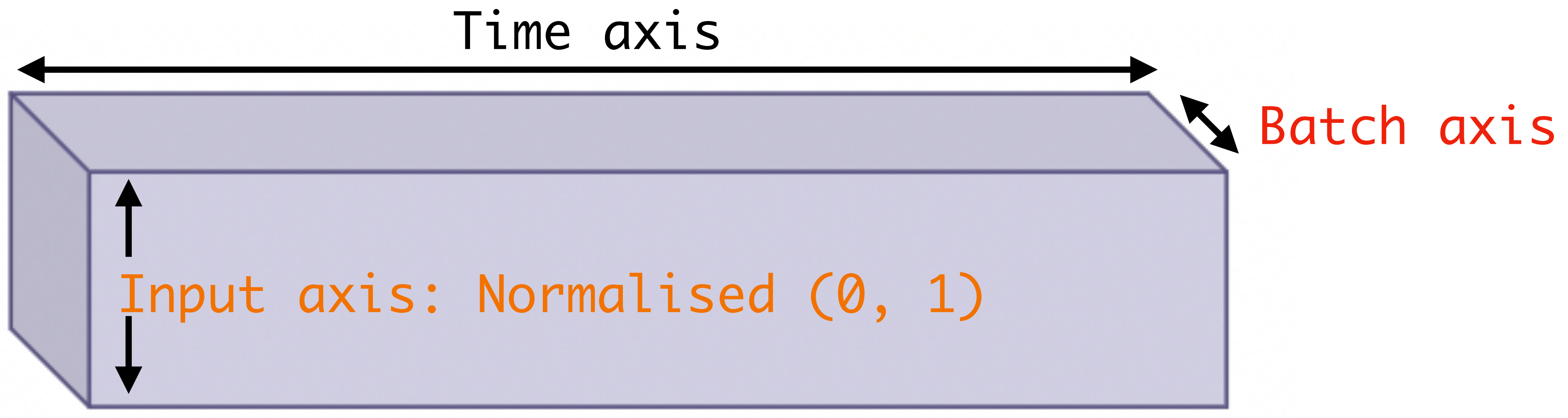
- Multi-Headed attention model these relationships by splitting the embeddings to lower dimension

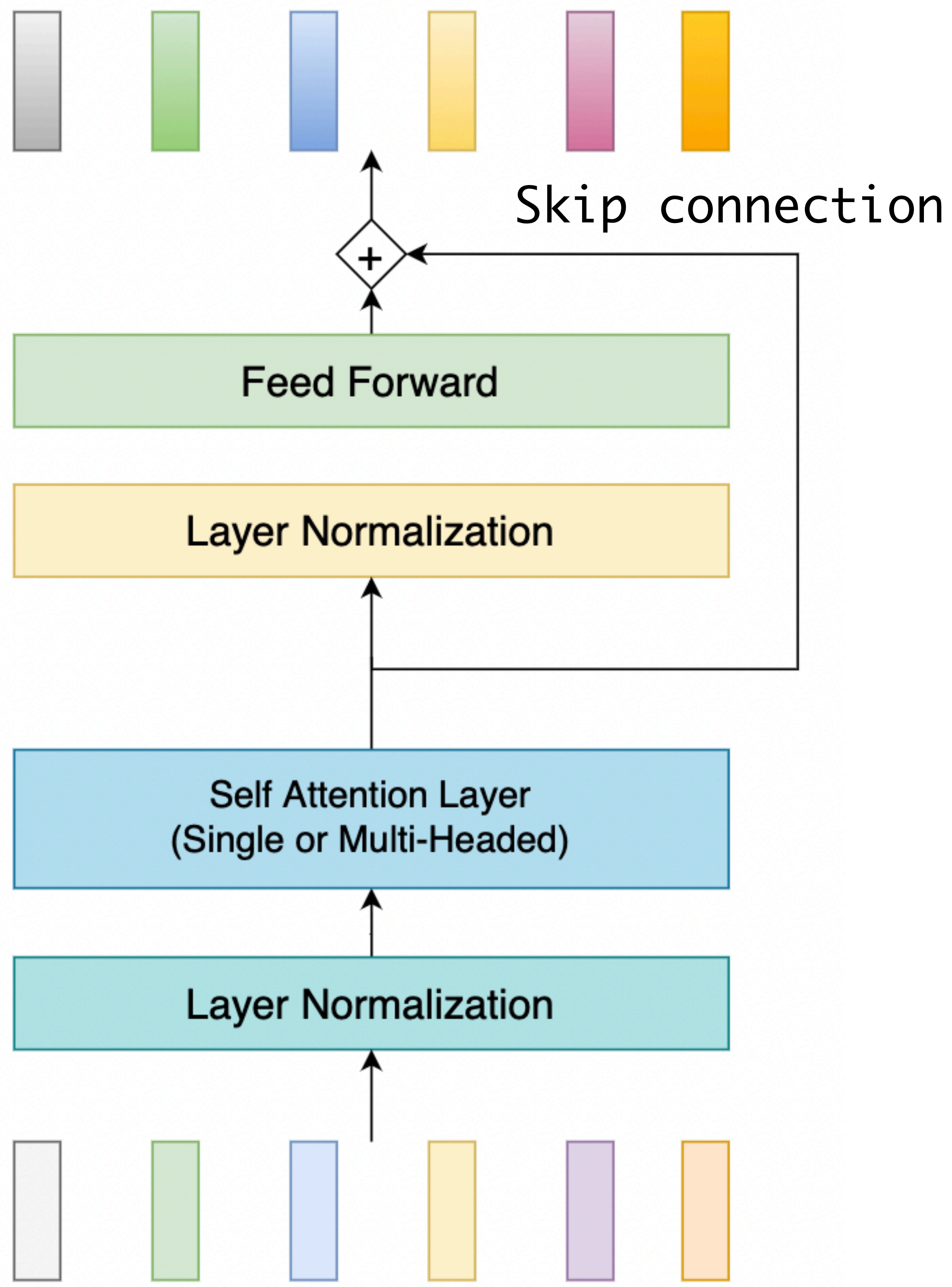


Reference: https://www.youtube.com/watch?v=KmAISyVvE1Y

Transformers
Transformer block

- A transformer block is simply a stack of different layers with skip connections
- Layer normalisation:
  - Similar to batch normalisation; but performed on the input dimesion





Reference: https://www.youtube.com/watch?v=oUhGZMCTHtI

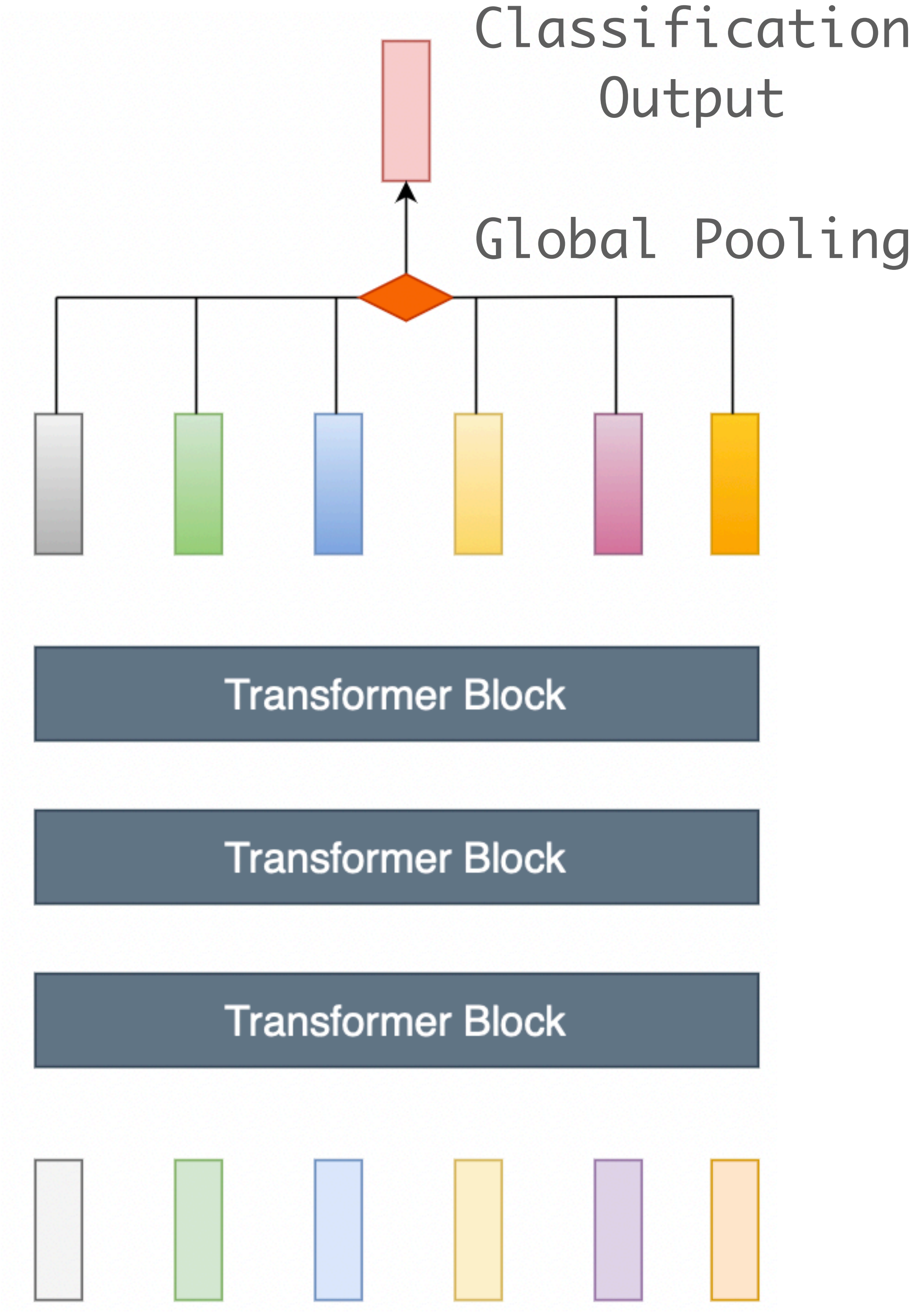
Transformers
Transformer block

- Layer normalisation:
  - $\mu^{b_t} = \frac{1}{d}\sum_{i}^{d} x_i^{b_t}$; Mean over the input dimension
  - $\sigma^{b_t} = \frac{1}{d}\sum_{i}^{d} (x_i^{b_t} - \mu^{b_t})^2$; variance over the input dimension
  - $\hat{x}^{b_t} = \frac{x^{b_t - \mu^{b_t}}}{\sqrt{(\sigma^{b_t} + \epsilon)}}$; Standardised inputs
  - $y^{b_t} = \gamma^T \hat{x}^{b_t} + \beta$; normalised layer outputs
    - γ and β are learnable parameters

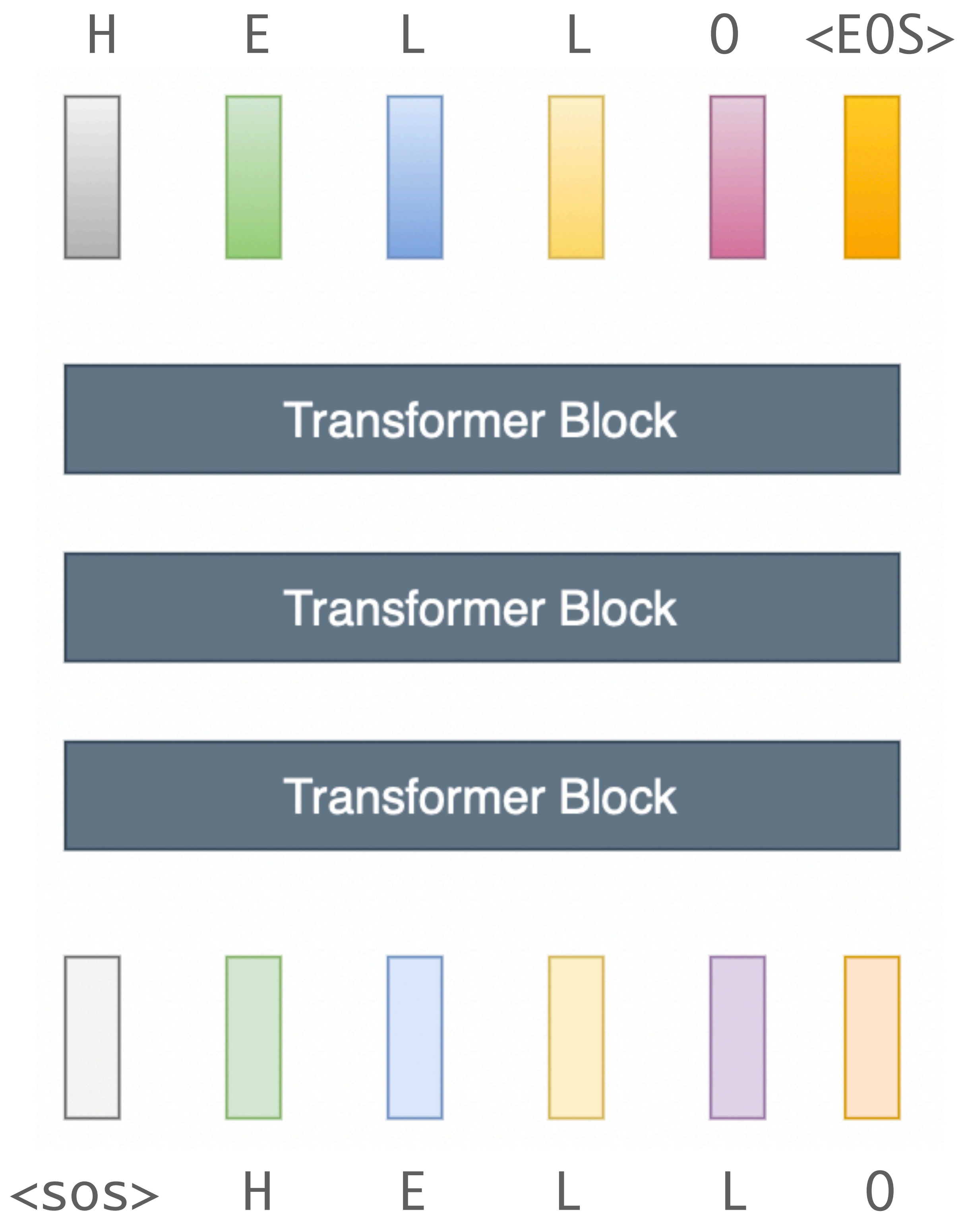
Reference: https://www.youtube.com/watch?v=oUhGZMCTHtI

Transformers

Transformer examples



Transformer network for
sequence classification

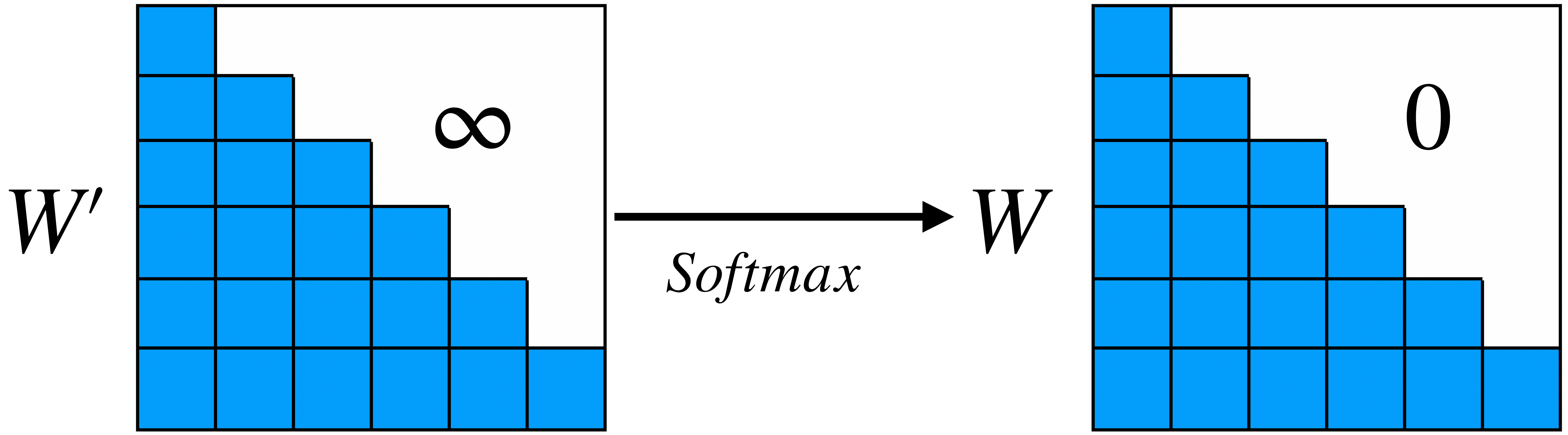


Transformer network for
autoregressive task

Reference: https://www.youtube.com/watch?v=oUhGZMCTHtI

Transformers
Issues

- Self attention can be non-causal
  - Has access to the information in the future
  - Problem for autoregressive tasks
- Solution: mask the forward connections



Reference: https://www.youtube.com/watch?v=oUhGZMCTHtI

Transformers
Issues

- Self attention is non-sequential
  - This is not a real restaurant, its a filthy burger joint
  - This is not a filthy burger joint, its a real restaurant
  - Both the sentences will give the same outcome
- Solution-1: Positional embedding
  - Word embeddings: $v_{the}, v_{man}, v_{pets}, v_{the}, v_{cat}, v_{again}$
  - Positional embeddings: $v_1, v_2, v_3, v_4, v_5, v_6$; Learnable vectors
  - Final embeddings: $(v_{the} + v_1), (v_{man} + v_2), (v_{pets} + v_3), (v_{the} + v_4), (v_{cat} + v_5), (v_{again} + v_6)$

Reference: https://www.youtube.com/watch?v=oUhGZMCTHtI

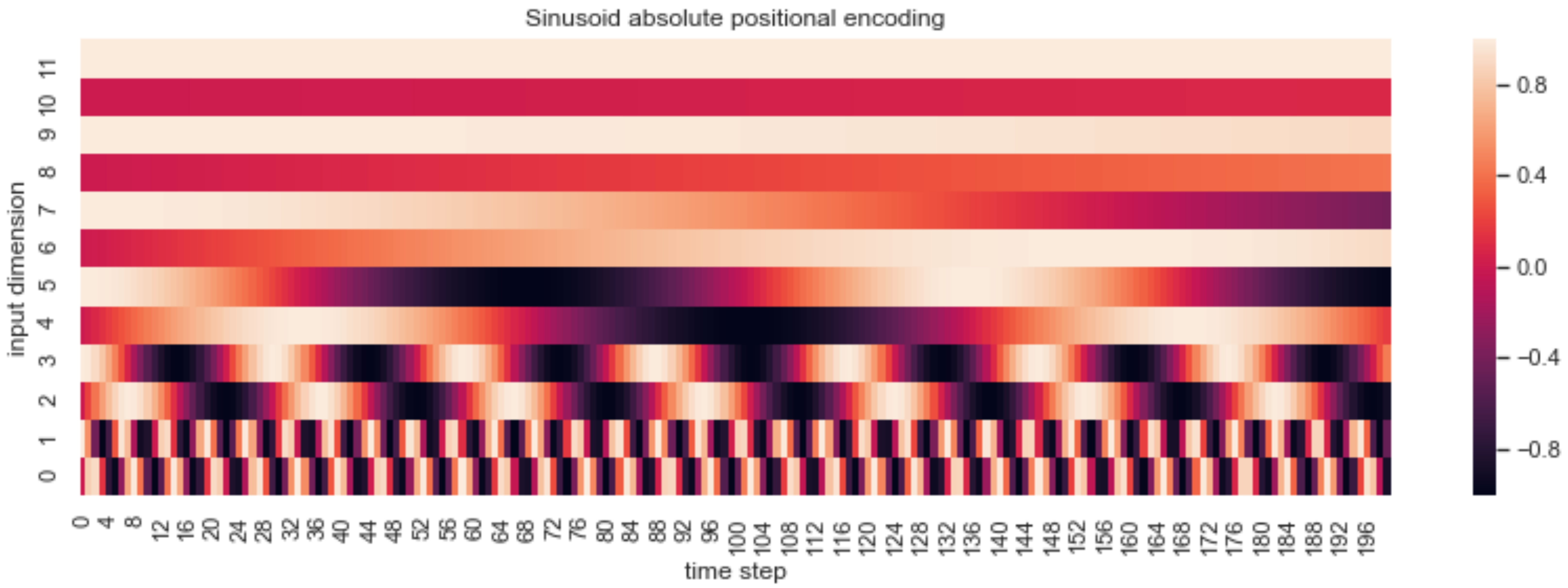
Transformers
Issues

- Solution-2: Positional encoding:
  - Use a deterministic periodic function such as $sine$ and $cosine$ function to encode the positional information

    Sine: $PE_{(pos,2i)} = sin\left(\frac{pos}{1000^{\frac{2i}{d_{model}}}}\right)$, $PE_{(pos,2i+1)} = cos\left(\frac{pos}{1000^{\frac{2i}{d_{model}}}}\right)$

    pos: position, i: dimension and d_{model}: number of dimensions
  - Non-learnable vectors
  - Apply multiple functions to each dimension
  - Suitable for long sequences

Transformers
Sinusoid Position Encoding



Ref-5

Reference

- Ref-1: https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html
- Ref-2: https://medium.datadriveninvestor.com/attention-in-rnns-321fbcd64f05
- Ref-3: https://labs.lilt.com/a-new-age-for-word-alignment-in-machine-translation
- Ref-4: https://royalsocietypublishing.org/doi/10.1098/rsos.191517
- Ref-5: https://www.inovex.de/de/blog/positional-encoding-everything-you-need-to-know/
- Ref-6: https://towardsdatascience.com/transformers-explained-visually-part-3-multi-head-attention-deep-dive-1c1ff1024853