EC-332 Machine Learning

Nazar Khan

Department of Computer Science
University of the Punjab

Transformers

Static vs. Dynamic Inputs

- *Static* signals, such as an image, do not change over time.
  - Ordered with respect to space.
  - Output depends on current input.
- *Dynamic* signals, such as text, audio, video or stock price change over time.
  - Ordered with respect to time.
  - Output depends on current input as well as past (or even future) inputs.
  - Also called *temporal*, *sequential* or *time-series* data.

Context in Text

'The Taj _____ was commissioned by Shah Jahan in 1631, to be built in the memory of ____ wife Mumtaz Mahal, who died on 17 June *that* year, giving birth to *their* 14th child, Gauhara Begum. *Construction* started in 1632, and the mausoleum was completed ____ 1643.'

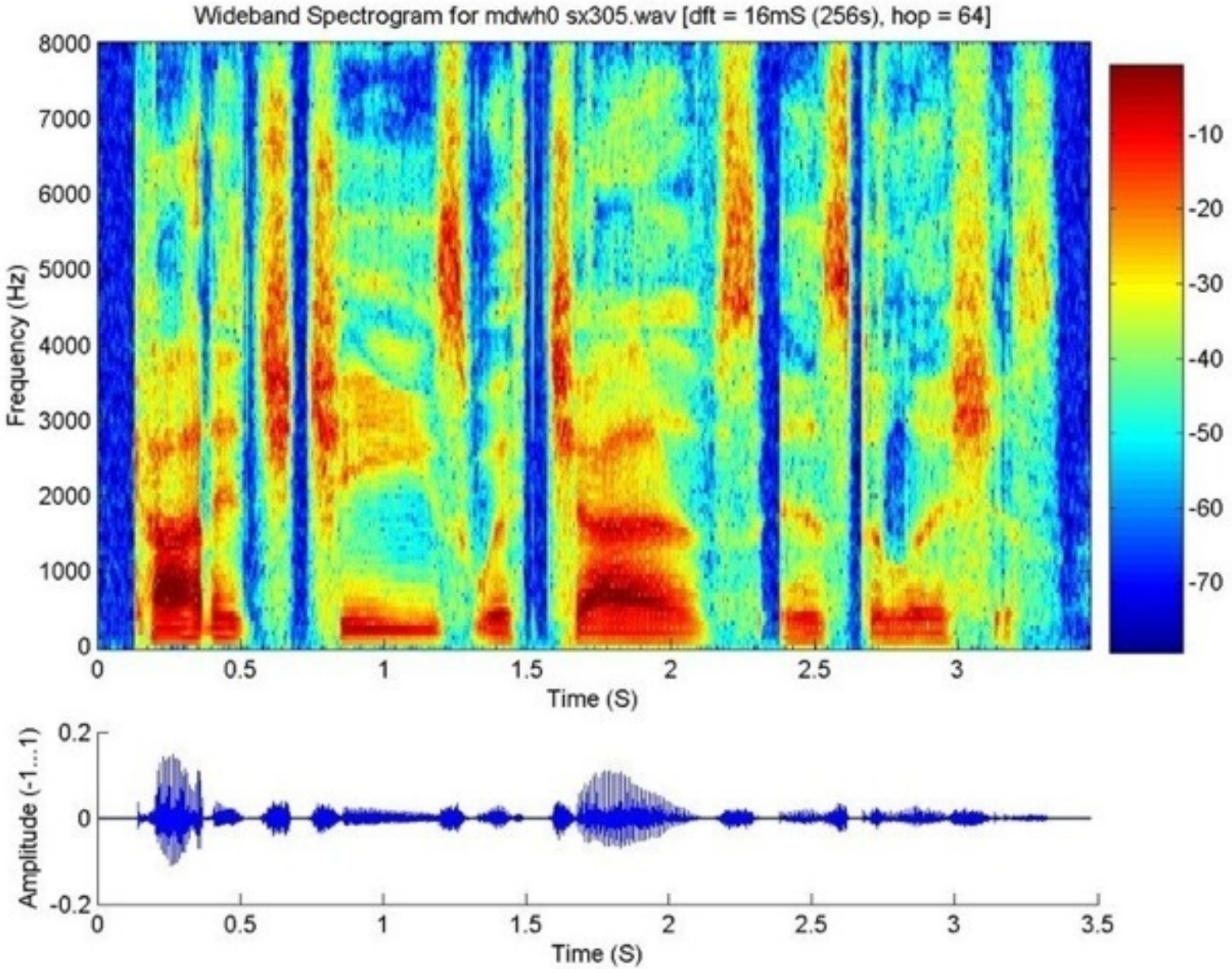
Context in Text

The following are the contents of a viral email from 2003.

> Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe.

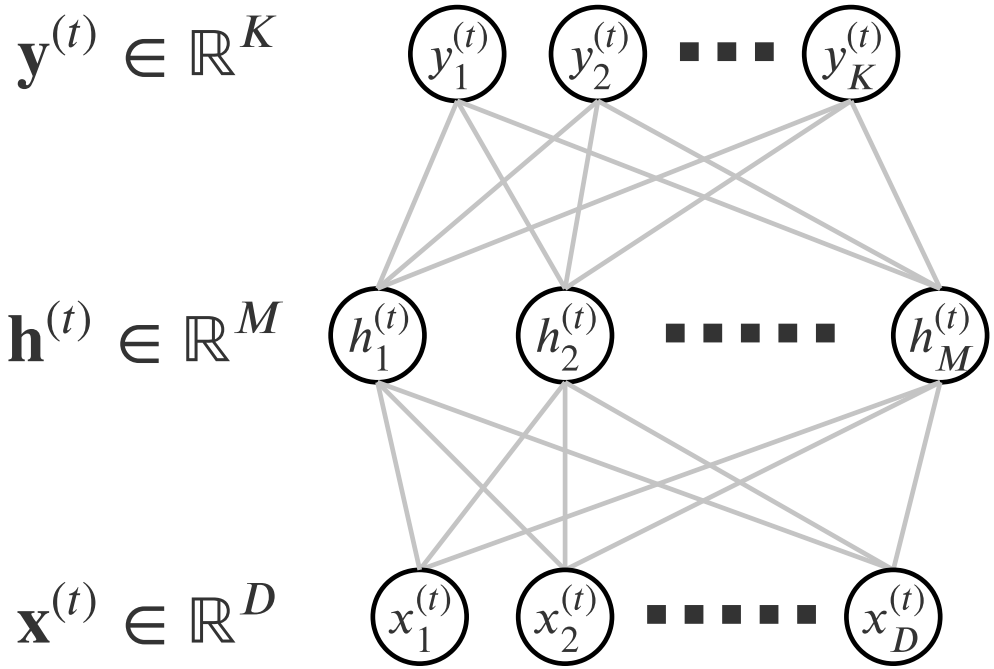
Context in Video

Context in Audio



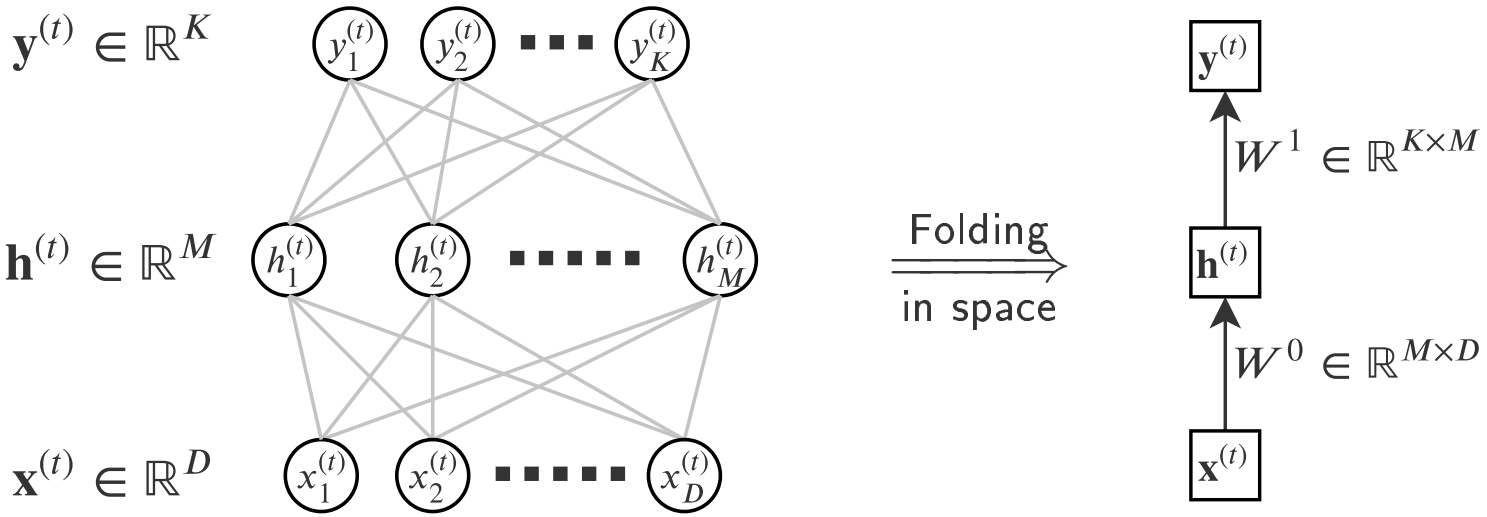
Time-series Data

- A *single* input will be a *series of vectors* $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^T$.



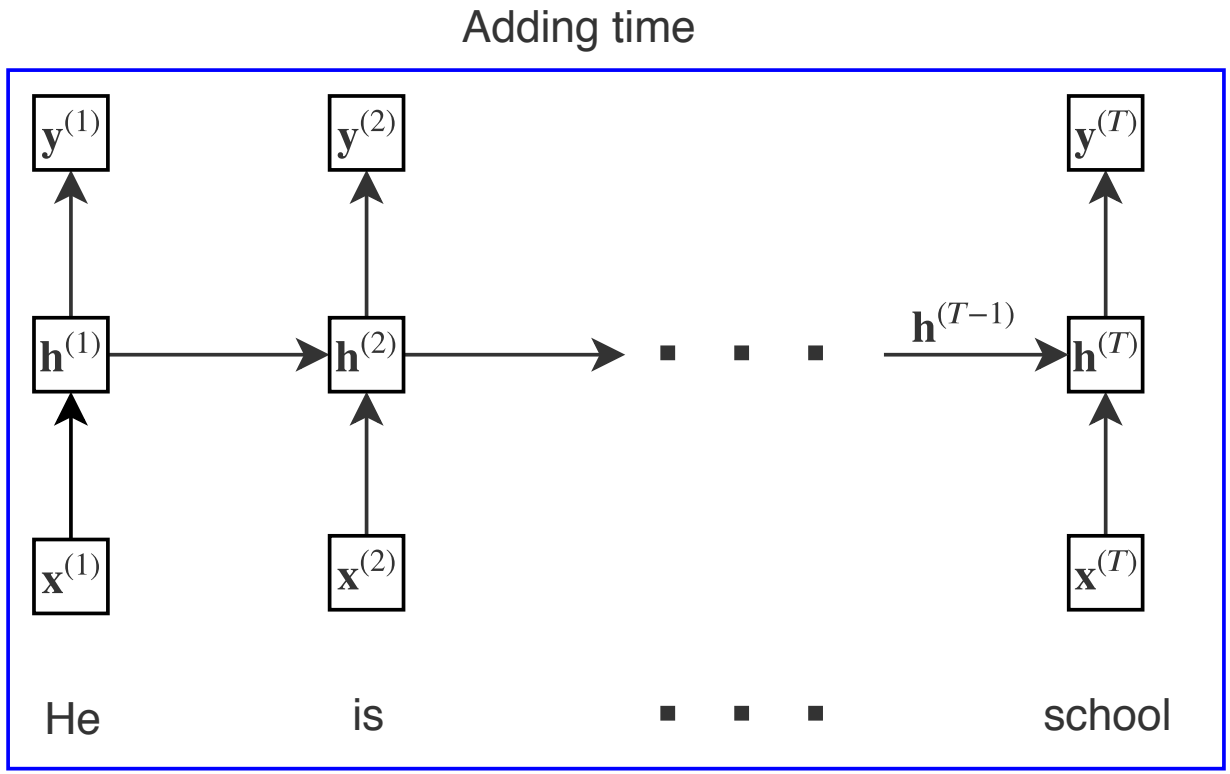
Input component at time t forward propagated through a network.

Representational Shortcut 1 – Space Folding



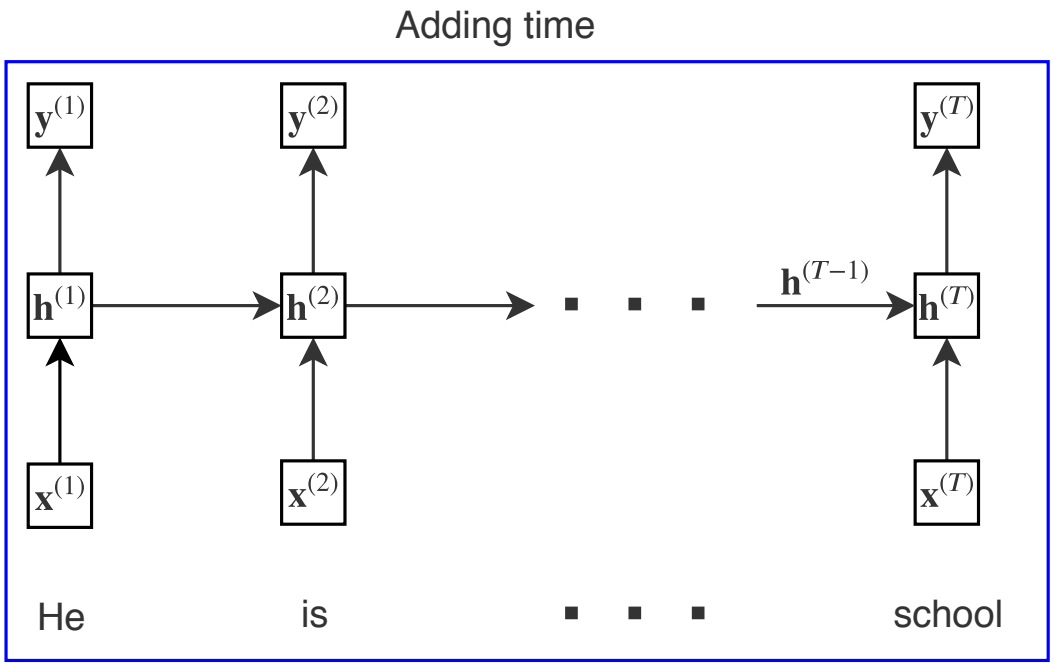
Each box represents a layer of neurons.

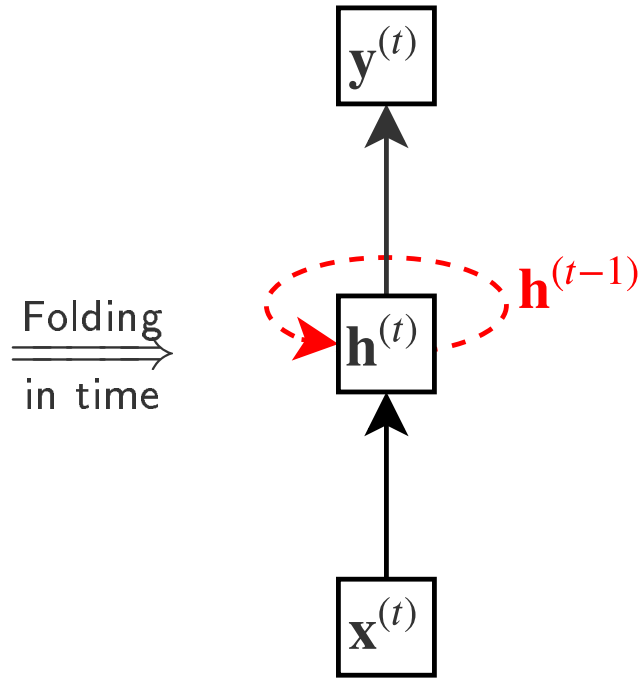
Recurrent Neural Networks



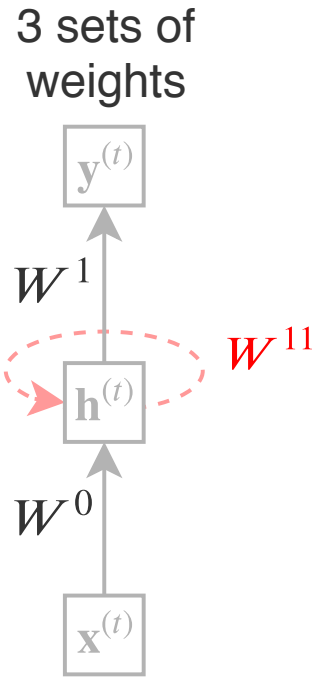
- A recurrent neural network (RNN) makes hidden state at time t directly dependent on the hidden state at time $t-1$ and therefore indirectly *on all previous times*.
- Output $\mathbf{y}_t$ depends on all that the network has already seen so far.

Representational Shortcut 2 – Time Folding





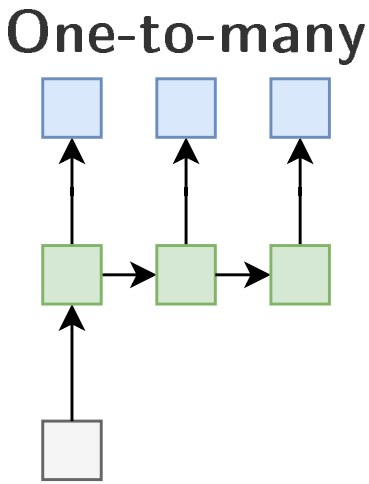
Recurrent Neural Networks

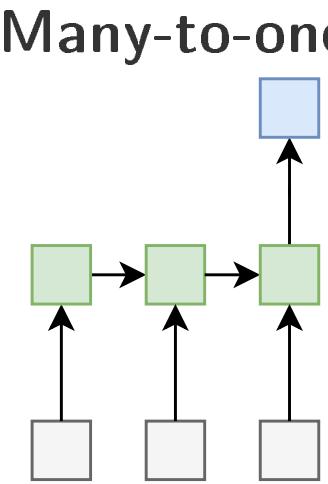


$$\mathbf{y}^{(t)} = f(\overbrace{W^1\mathbf{h}^{(t)} + \mathbf{b}_1}^{\mathbf{a}^{1(t)}})$$

$$\mathbf{h}^{(t)} = \tanh(\underbrace{W^0\mathbf{x}^{(t)} + W^{11}\mathbf{h}^{(t-1)} + \mathbf{b}_0}_{\mathbf{a}^{0(t)}})$$

Sequence Mappings





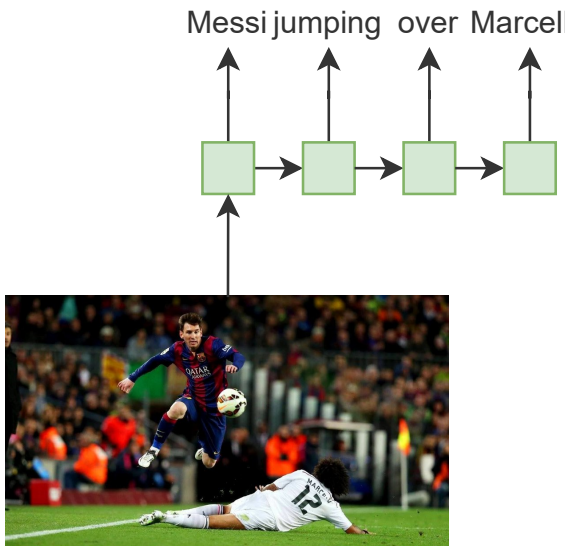
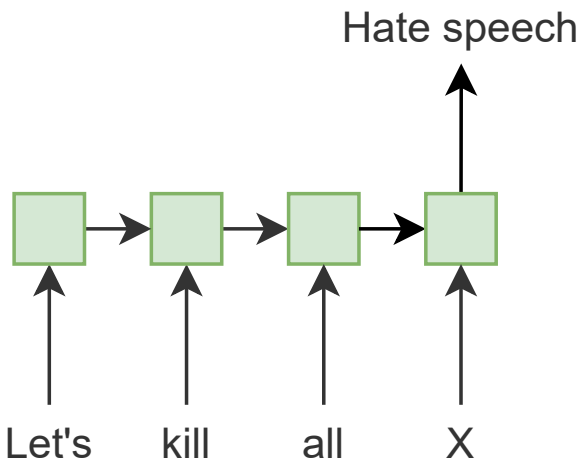


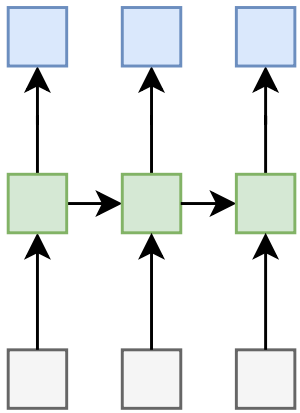
Image caption generation



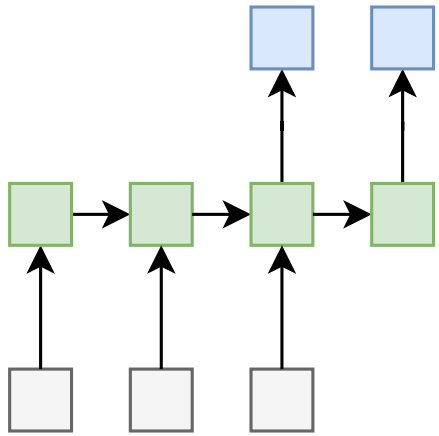
Sentiment classification

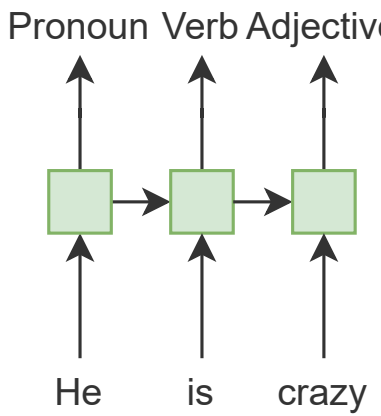
Sequence Mappings

Many-to-many

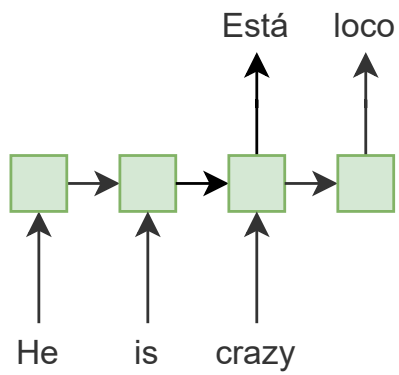


Many-to-many delayed





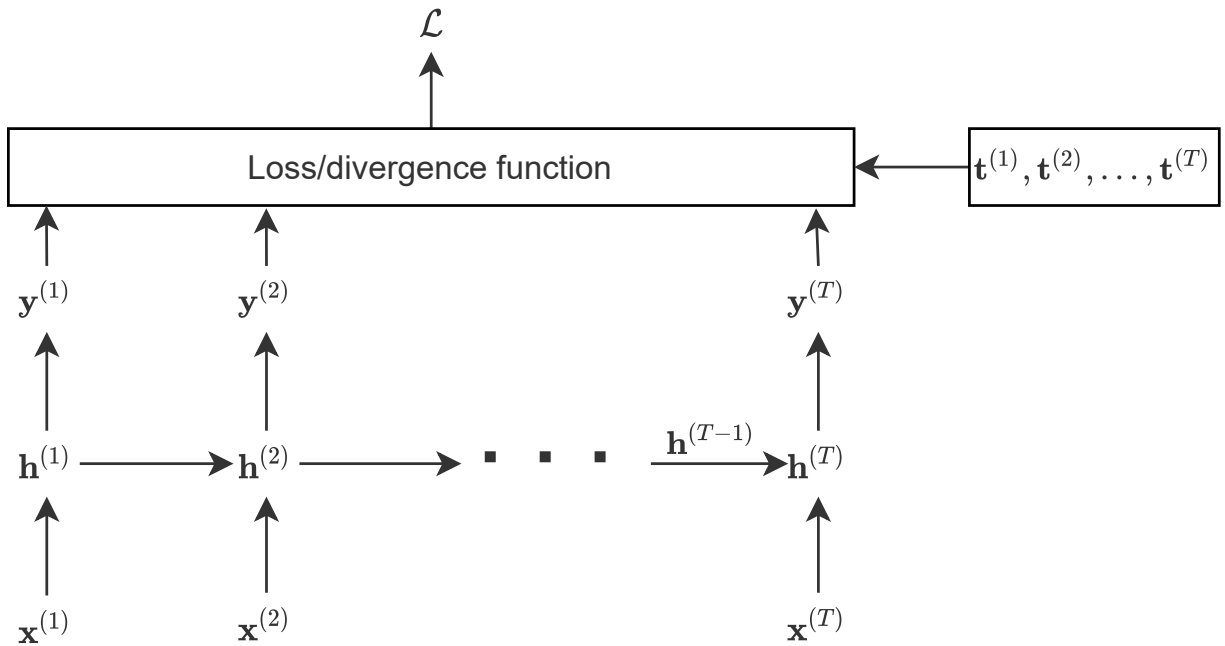
POS tagging



Language translation

Loss Functions for Sequences

- For recurrent nets, loss is between *series* of output and target vectors. That is $\mathcal{L}(\{\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(T)}\}, \{\mathbf{t}^{(1)}, \ldots, \mathbf{t}^{(T)}\})$.

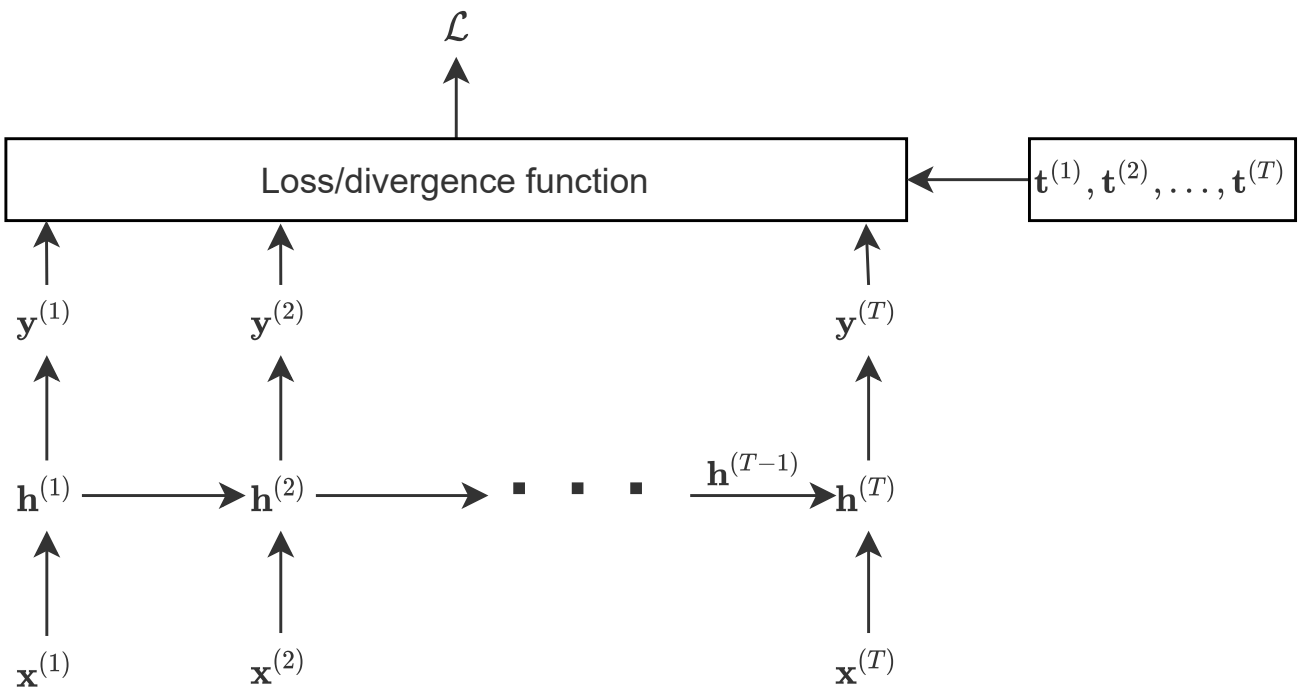


Forward propagation in an RNN unfolded in time.

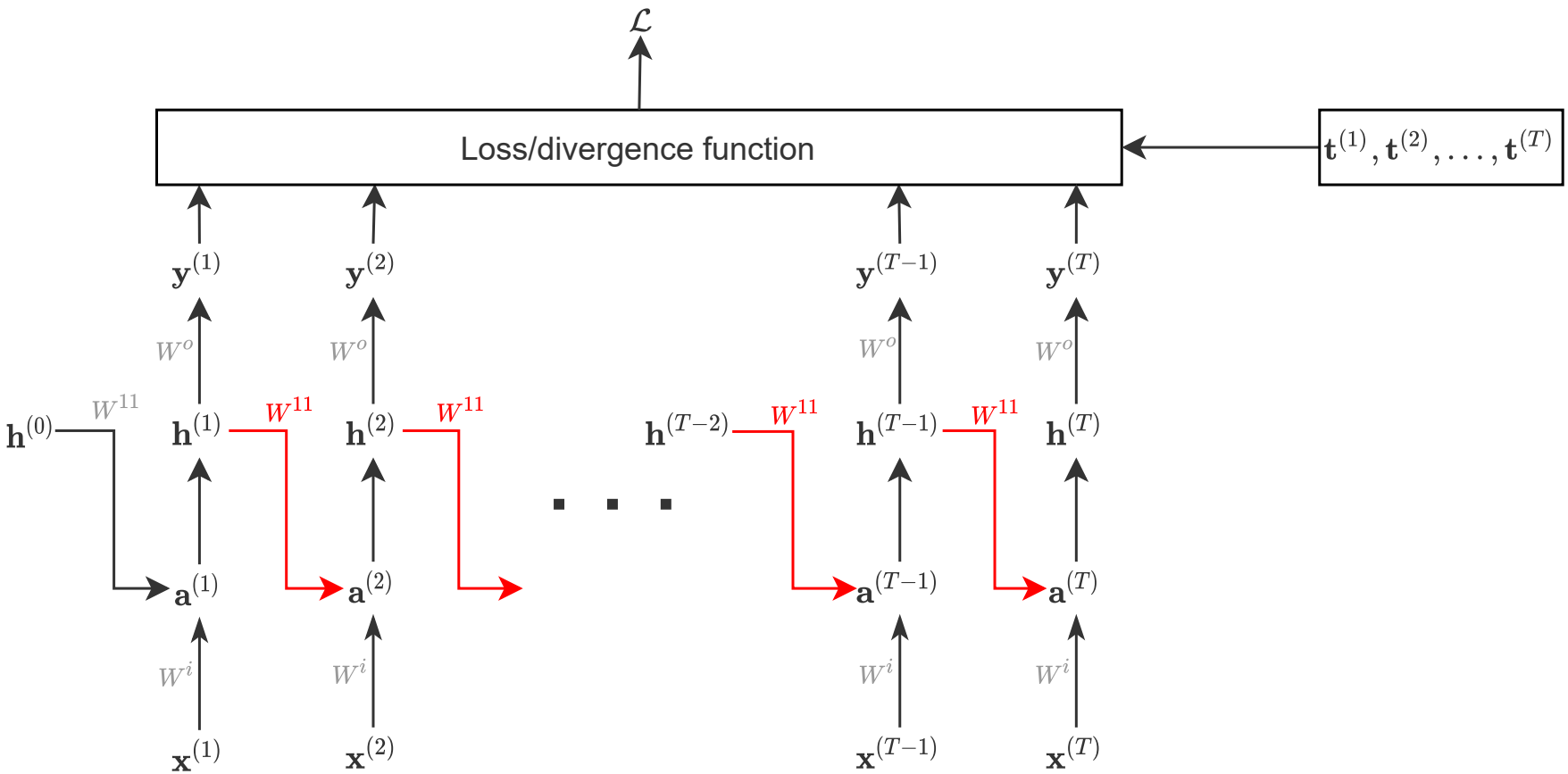
- Notice that loss $\mathcal{L}$ can be computed only after $\mathbf{y}^{(T)}$ has been computed.

Loss Functions for Sequences

- Loss is *not necessarily* decomposable.
- In the following, we will assume decomposable loss $\mathcal{L} = \sum_{t=1}^{T} \mathcal{L}(\mathbf{y}^{(t)}, \mathbf{t}^{(t)})$.
- In both cases, as long as $\frac{\partial L}{\partial \mathbf{y}^{(t)}}$ has been computed, backpropagation can proceed.

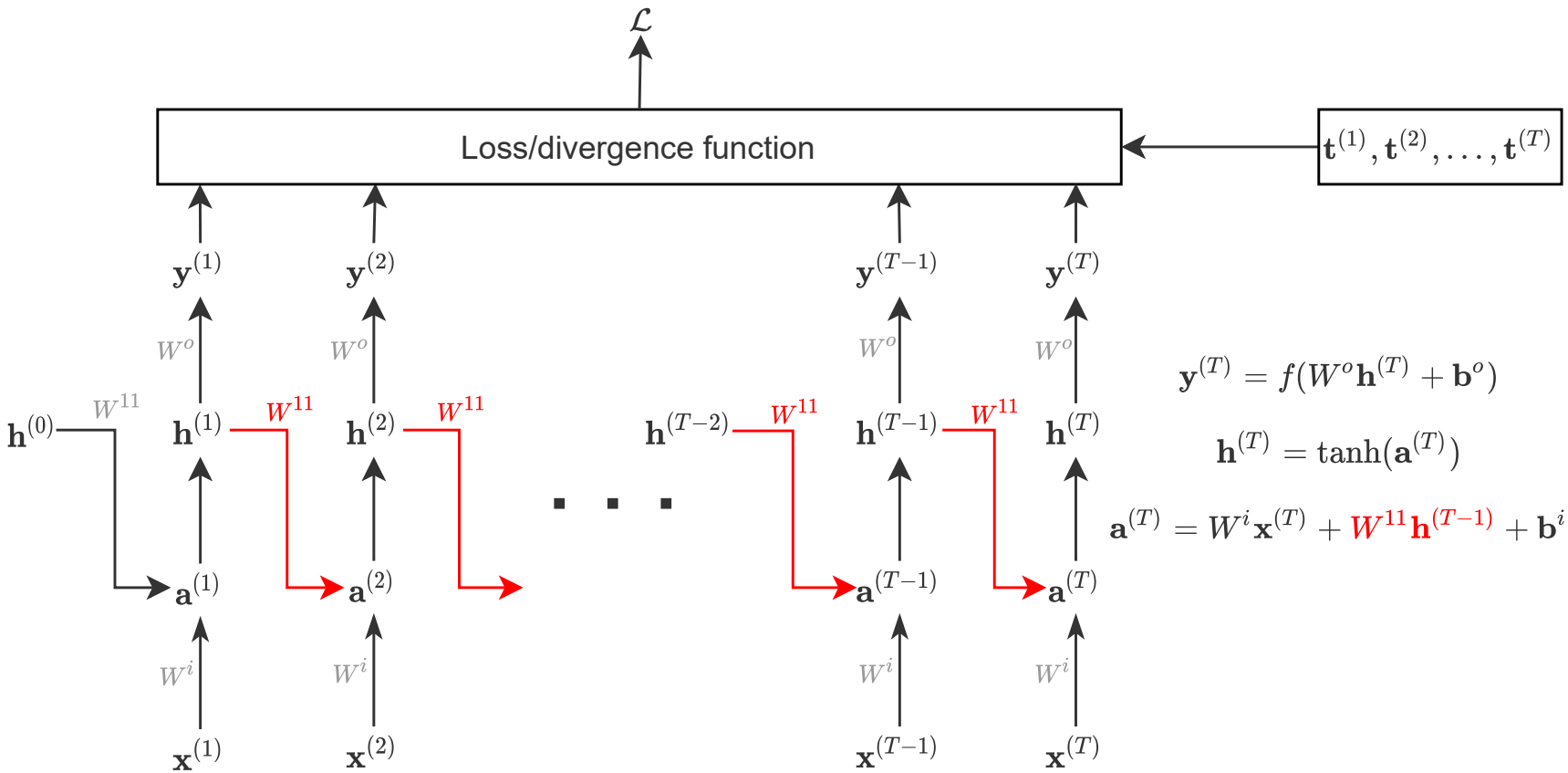


Forward Propagation Through Time



Forward propagation in an RNN unfolded in time. Recurrence between hidden states through pre-activation $\mathbf{a}^{(t)}$ is shown in red.

Forward Propagation Through Time



Forward propagation in an RNN unfolded in time. Recurrence between hidden states through pre-activation $\mathbf{a}^{(t)}$ is shown in red.

The problem with RNNs

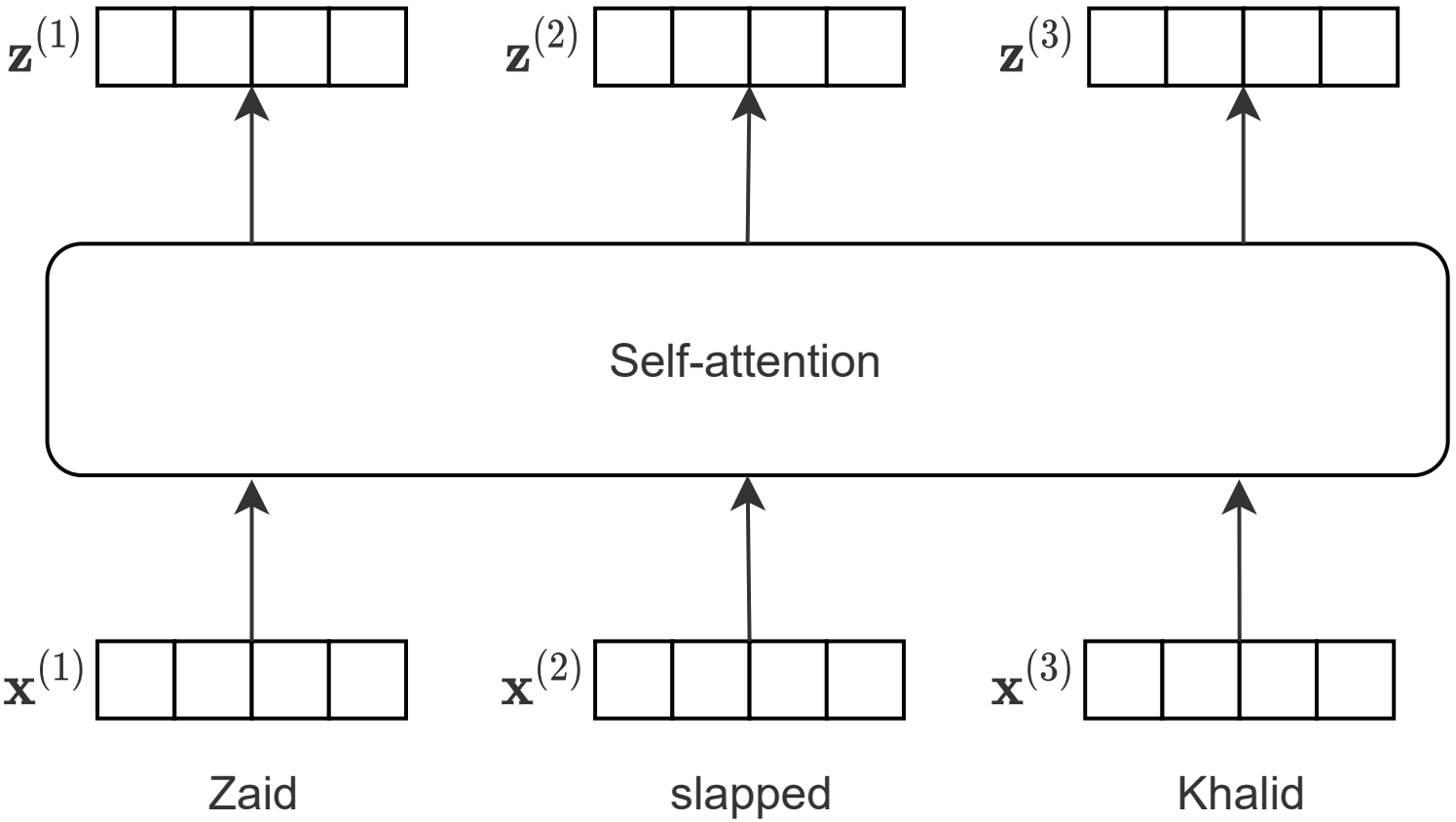
- Recurrence is a sequential process.
- It cannot be parallelized.
- Makes training slow.

Self-attention and Transformers[1]

- A sequence-to-sequence model without convolution and without recurrence.
- Instead of looking at previous inputs, look at all inputs.
- Transformer contains parallelizable modules and can therefore be trained faster.
- Transformers achieve state-of-the-art performance on sequence modelling tasks.

[1] Ashish Vaswani et al. 'Attention is All You Need'. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA, 2017, 6000–6010.

Self-attention



We will assume 512-dimensional input embeddings $\mathbf{x}^{(t)}$ as well as 512-dimensional encodings $\mathbf{z}^{(t)}$.

Positional Encoding

- Transformers have no recurrence, so positional encodings are added:

$$P_{(pos,2i)} = \sin(pos/10000^{2i/d})$$

$$P_{(pos,2i+1)} = \cos(pos/10000^{2i/d})$$

- pos: index of the token in the sequence. For example, if a word is the third word in a sentence, its pos would be 2 (since indexing is 0-based).
- d: dimension of the embedding vector (typically 512).
- i: index within embedding vector. Ranges from 0 to $d-1$.
- 10000 is an arbitrary scaling factor large enough to ensure positional information is spread out across different frequencies.
- The term $10000^{2i/d}$ changes the wavelength of the sine and cosine functions so that different dimensions of the positional encodings capture positional dependencies at different scales.

Positional Encodings
Intuition

- The sine and cosine functions generate a unique positional encoding for each word position.
- The frequency of the functions is controlled by the exponent term $2i/d$, which varies across the dimensions.
- Even dimensions $(2i)$ use the sine function, while odd dimensions $(2i + 1)$ use the cosine function.
- This design allows each position to be uniquely represented in a continuous way, and it enables the model to learn relative positions, not just absolute ones.

Self-attention

1. Place embeddings of all words in a matrix $E \in \mathbb{R}^{512 \times T^{\text{in}}}$.
2. Add positional encodings.
3. Consider 3 *learnable* matrices $W_Q, W_K \in \mathbb{R}^{64 \times 512}$ and $W_V \in \mathbb{R}^{512 \times 512}$ and apply linear transformations

$$Q = W_Q E \in \mathbb{R}^{64 \times T^{\text{in}}}$$
$$K = W_K E \in \mathbb{R}^{64 \times T^{\text{in}}}$$
$$V = W_V E \in \mathbb{R}^{512 \times T^{\text{in}}}$$

   to each word. *Parallelizable in time*.
4. Compute similarity scores between the representations in Q and K.

$$S = \text{row-wise softmax}\left(\frac{Q^T K}{\sqrt{64}}\right) \in \mathbb{R}^{T^{\text{in}} \times T^{\text{in}}}$$

Self-attention

5. Compute the encoding of each word

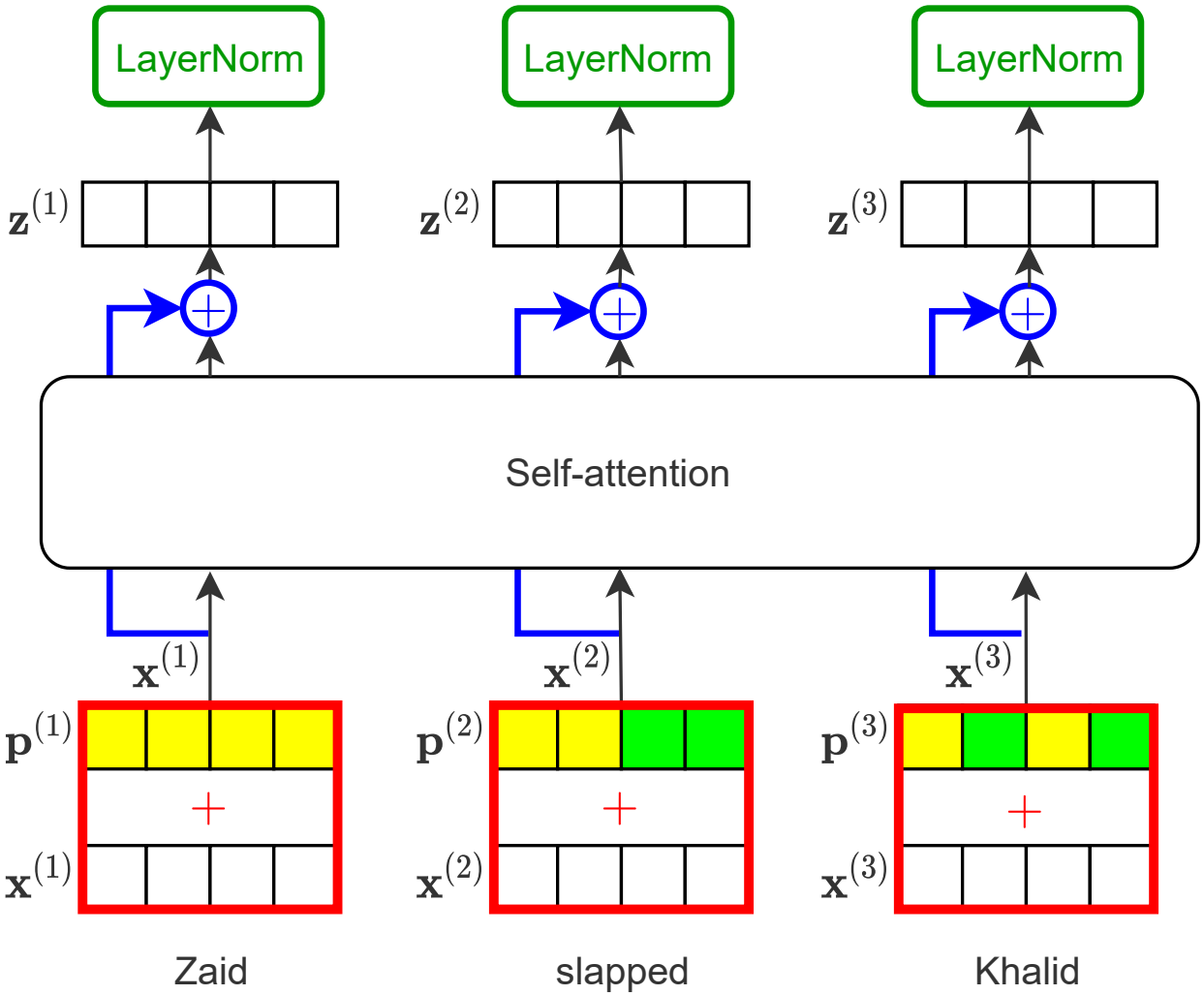
$$Z = VS^T \in \mathbb{R}^{512 \times T^{\text{in}}}$$

where each column of Z is a 512-dimensional encoding of the corresponding word.

Note that *each word has now been encoded by attending to all words in the sentence*.

The scaled dot-product scores in S are the attention weights.

Self-attention
Additional details



Multi-headed attention

- Replicate 8 self-attention modules, each with its own learnable matrices W_{Qi}, W_{Ki}, W_{Vi}.
- Compute encodings $Z_1, \ldots, Z_8$.
- Compute final encoding Z by concatenating the Z_i and projecting onto 512-dimensional space using another learnable matrix $W_O \in \mathbb{R}^{512 \times (512*8)}$.

$$Z = W_O \begin{bmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_8 \end{bmatrix}$$

- This way, the model can learn 8 different ways of encoding the input sentence.

Feed-forward NN

- Pass each encoding in Z through the same 2-layer network.

$$E = W_2 * ReLU(W_1 Z + \mathbf{b}_1 \mathbf{1}^T) + \mathbf{b}_2 \mathbf{1}^T$$

  where W_1 has 2048 rows and W_2 has 512 rows.
- Add residual connection.

$$E = W_2 * ReLU(W_1 Z + \mathbf{b}_1 \mathbf{1}^T) + \mathbf{b}_2 \mathbf{1}^T + Z$$

- Perform LayerNorm on each column of E.
- *Parallelizable in time*.

Stacked Encoders

- An encoder involves the transformation

$$\text{Embeddings} \longrightarrow \text{Self-attention} \longrightarrow \text{FFNN} \longrightarrow \text{Encodings}$$

- Encoders can be stacked on top of each other.
- Encoding produced by one encoder becomes the input embedding for the next encoder.
- Final encoded output is the result of the last encoder.

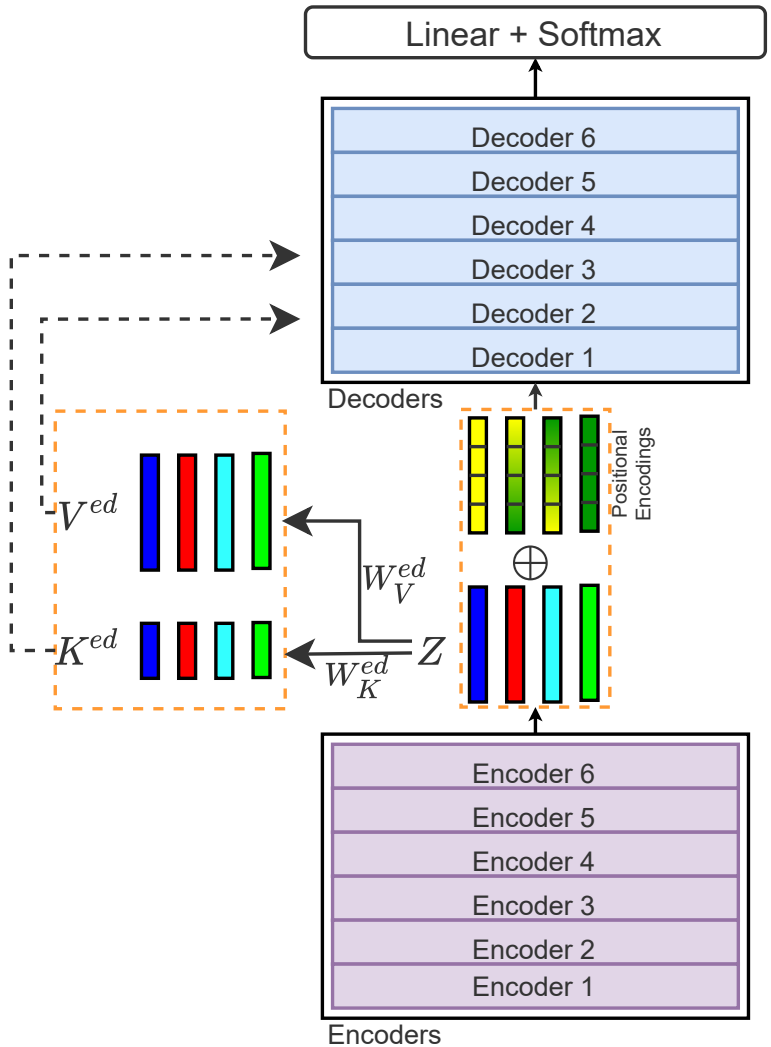
From Encoder to Decoder

Cross-Attention

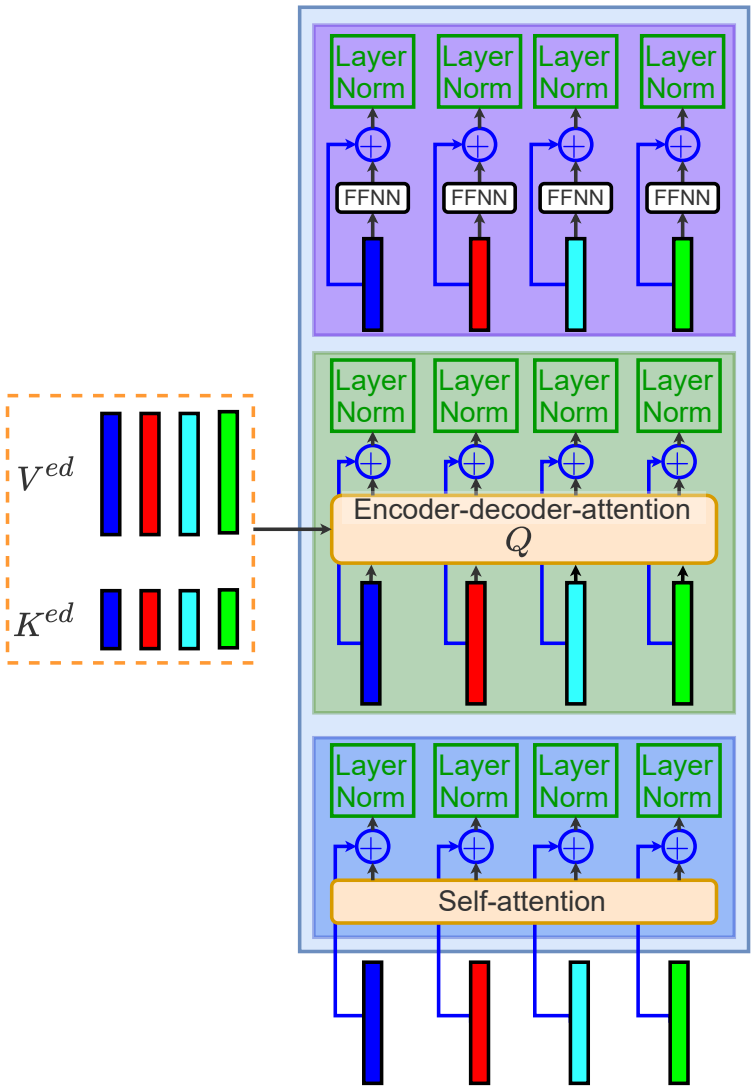
- Cross-attention is used in the Transformer decoder to attend to encoder outputs.
- In the decoder:
  - Queries (Q) come from the decoder's previous layer.
  - Keys (K) and Values (V) come from the encoder's output.
- The mechanism allows the decoder to focus on relevant parts of the input sequence while generating each token.
- *The attention weights determine which encoded input tokens are most influential for the current decoding step.*

From Encoder to Decoder

Cross-Attention



Inside a Decoder



Decoder and the future

- Self-attention in decoder attends only to the words generated so far in the output sequence.
- Achieved by setting future times to $-\infty$ in the softmax.

Summary

- Transformers represent the state of the art in deep learning architectures.
- Key progress is due to self-attention – representation of token at any time depends on tokens at all other times.
  - Complete input sequence observed in one go. Therefore, long-term context.
  - No sequential processing. Therefore, parallelizable and fast.
- Regularization through layer normalization to retain parallelism.
- Multiple attention heads for attending in different ways.
- Positional encoding to exploit sequential order.