Deep Learning and Its Applications in Signal Processing

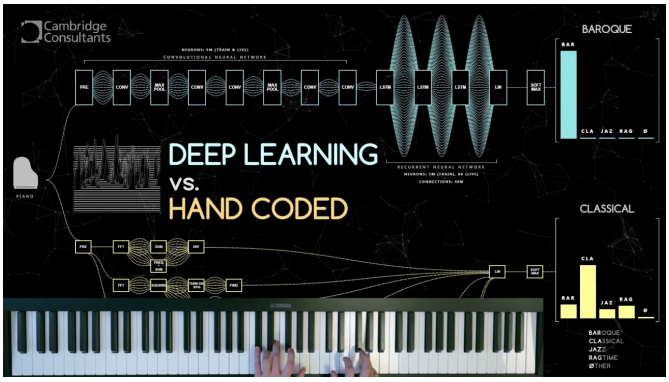
Lesson 5: Attention-based Neural Networks for Signal Processing

Liang Dong, ECE

Baylor University
SCHOOL OF ENGINEERING & COMPUTER SCIENCE
Department of Electrical & Computer Engineering

Recurrent Neural Network Demo

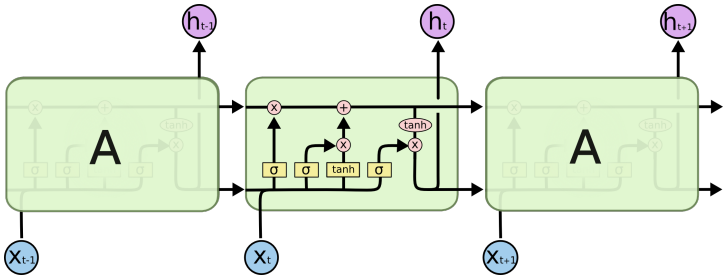
Real-time music classification with RNN/LSTM



RNN/LSTM ← Click here

Recurrent Neural Network Limits

- The basic recurrent neural network (RNN) design struggles with long sequences. In practice, it does not learn long-term dependencies well.
- A special variant, long short-term memory (LSTM) network, can work well with long sequences, achieving remarkable results in voice recognition, translation, and image captioning.
- LSTM, like ResNet, can bypass units of RNN and remember for longer time steps. It removes some of the vanishing gradient problem of the RNN. "Selectively remember or forget."



Recurrent Neural Network Limits

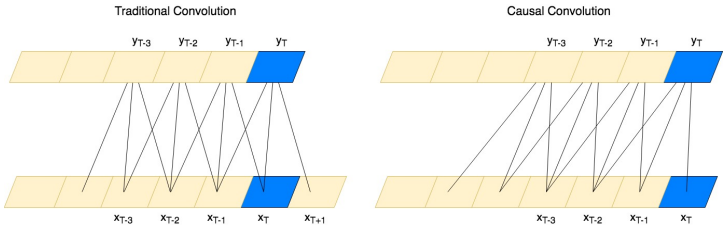
However, RNN and LSTM have limits:

1. Long sequence problem. When sequences are long, the model often forgets the content of distant positions. The probability of keeping the context from a far-away word decreases exponetially with the distance from it.
2. Parallelization problem. The sentence is processed word by word sequentially which inhibits parallelization. Not fitting for hardware acceleration.
3. Modeling problem. There is no explicit modeling of long and short range dependencies.

Temporal Convolutional Network

- Temporal convolutional network (TCN) – A variation of CNN architecture for modeling sequences.

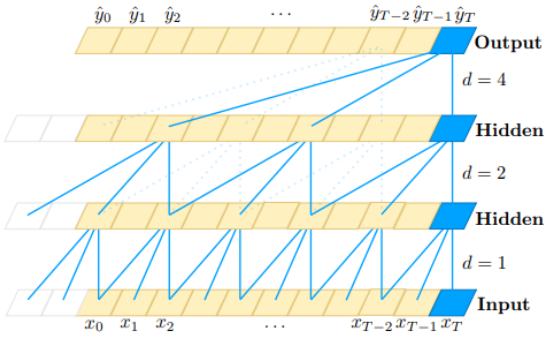
- TCN uses causal convolution – It convolves only with the elements from current and earlier timestamps in the previous layer.



- Temporal convolutional networks (TCN) "outperform canonical recurrent networks such as LSTMs across a diverse range of tasks and datasets, while demonstrating longer effective memory".

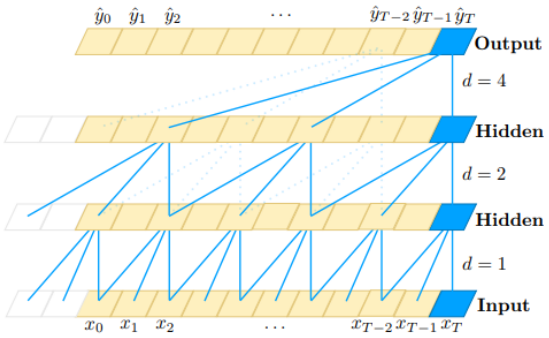
Shaojie Bai, J. Zico Kolter, and Vladlen Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," *arXiv*, 2018.

Temporal Convolutional Network



- TCN uses 1-D fully-convolutional network, where each hidden layer is the same length as the input layer (with zero padding). The output sequence is of the same length as the input sequence (like RNN).
- TCN uses causal convolution. No information leakage from future to past.

Temporal Convolutional Network



- TCN uses dilated convolution. It can capture long-term dependencies (long history).

$$(\mathbf{x} \otimes_d \mathbf{f})(n) = \sum_{i=0}^{k-1} f(i)x(n - di)$$

where $\mathbf{x}$ is the input sequence, $\mathbf{f}$ is a filter of length k, and d is the dilation factor.

Temporal Convolutional Network

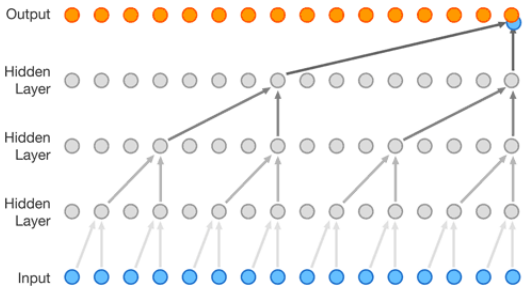
Advantages of TCN:

- In both training and inference, a long input sequence can be processed as a whole, because convolutions can be done in parallel as the same filter is used in each layer.
- Convolution helps capture (temporally) local information.
- TCN can change its receptive field size easily, hence better control of the model's memory size.
- TCN filters are shared in a layer with back-propagation only depending on the network depth. "Distance" in the order of $\log(n)$.

C. Lea, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks: A unified approach to action segmentation," *European Conference on Computer Vision*, 2016.

N. Kalchbrenner, et al., "Neural machine translation in linear time," *arXiv* 2016.

WaveNet



- WaveNet for audio signal synthesis – WaveNet directly models the raw waveform of the audio signal, one sample at a time.
- Sequence modeling – Autoregressive feedforward models model the last n steps using dilated convolutions.
- At each step during inference time, a value is drawn from the probability distribution computed by the network. This value is then fed back to the input and a new prediction for the next step is made.

Aaron van den Oord, et al., "WaveNet: A Generative Model for Raw Audio," *arXiv*, 2016.

Attention-based Neural Network

- Attention-based models require less resources to train and run than RNN/LSTM.
- Attention – Neural networks focus on a subset of the information they are given.
- Attention can be an important component of neural network to understand sequences.
- Attention mechanism parses spatio-temporal information.

Attention

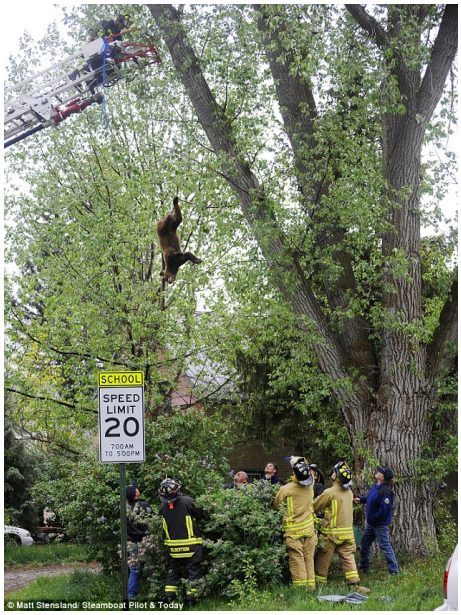


Figure: Bear falls from the tree.

- Our brains implement attention at many levels, in order to select only the important information to process, and eliminate the overwhelming amount of background information.
- Human visual attention allows us to focus on a certain region with "high resolution" while perceiving the surrounding image in "low resolution", and then adjust the focal point or do the inference accordingly.

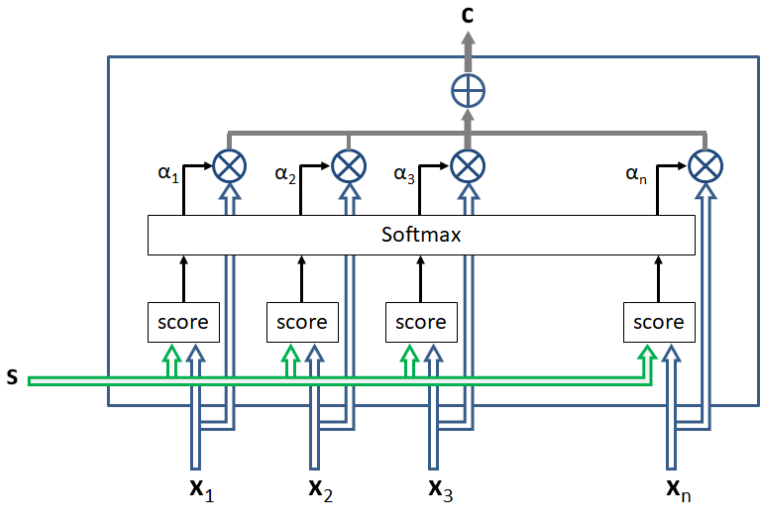
Attention

Notice

There no is exam final for this class!

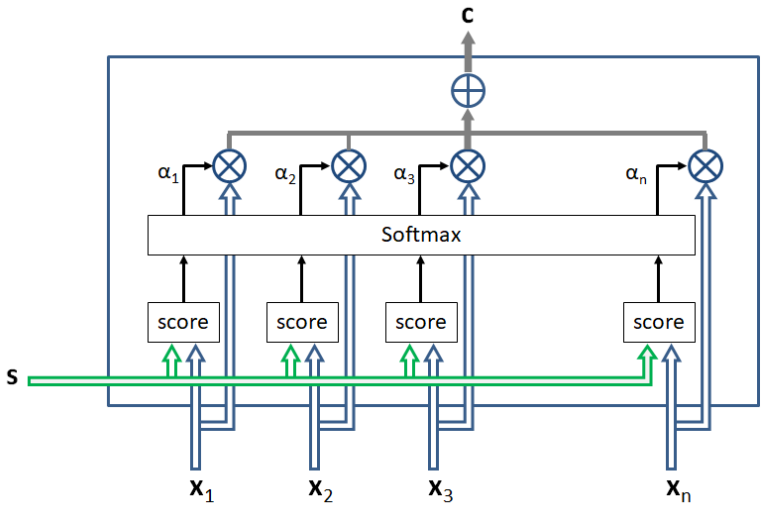
- Similarly, we select the important information in a sentence to process.
- We do not read sequentially. In fact, we interpret characters, words, and sentences as a group.
- An attention-based or convolutional module perceives the sequence and projects a representation in our mind.

Attention Model



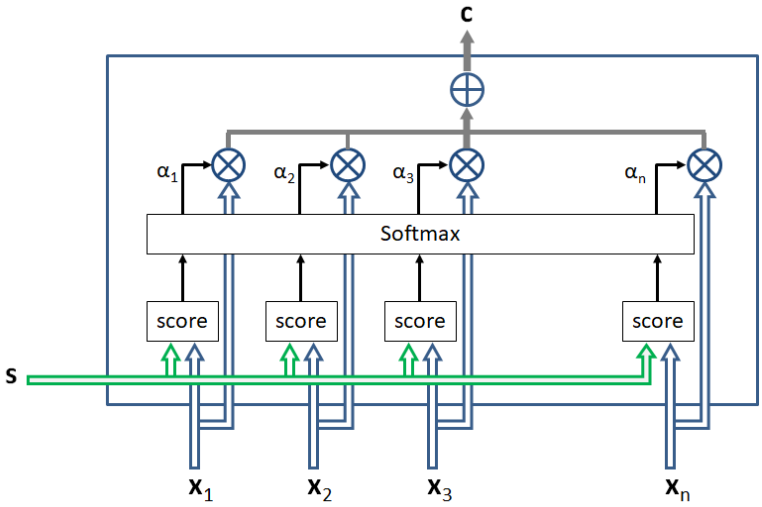
- Reduce data to a focal set and pay attention to the set.
- Make attention differentiable so that it can be "learned". How?
  Focus everywhere, just to different extents.

Attention Model



- The attention model gives a context vector $\mathbf{c}$ which is the summary of the source $\mathbf{x}$ focusing on the information linked to the focus $\mathbf{s}$.
- The score is the relevance of each $\mathbf{x}_i$ given the focus $\mathbf{s}$.

Attention Model



- The score function can be
  $\text{score}(\mathbf{s}, \mathbf{x}_i) = \mathbf{v}_a^T \tanh\left(\mathbf{W}_a[\mathbf{s}; \mathbf{x}_i]\right)$ ($\mathbf{s}$ and $\mathbf{x}_i$ linearly combined)
  $\text{score}(\mathbf{s}, \mathbf{x}_i) = \mathbf{s}^T \mathbf{x}_i$ (dot product indicating relevance)

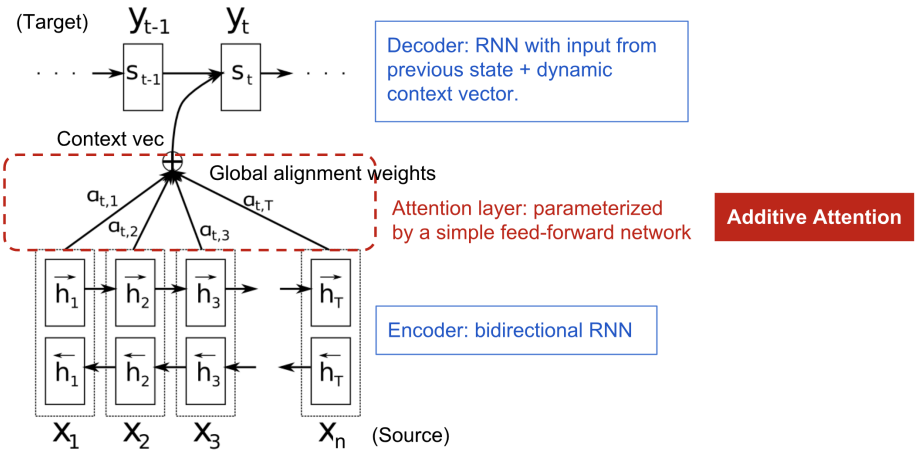
Attention in Image Captioning

Figure: Attention-weight visualization: "A woman is throwing a frisbee in a park."

The image is first encoded by a CNN to extract features. Then a LSTM decoder consumes the features to produce descriptive words one by one, where the weights are learned through attention.

K. Xu, et al., "Show, attend and tell: Neural image caption generation with visual attention," *ICML*, 2015.

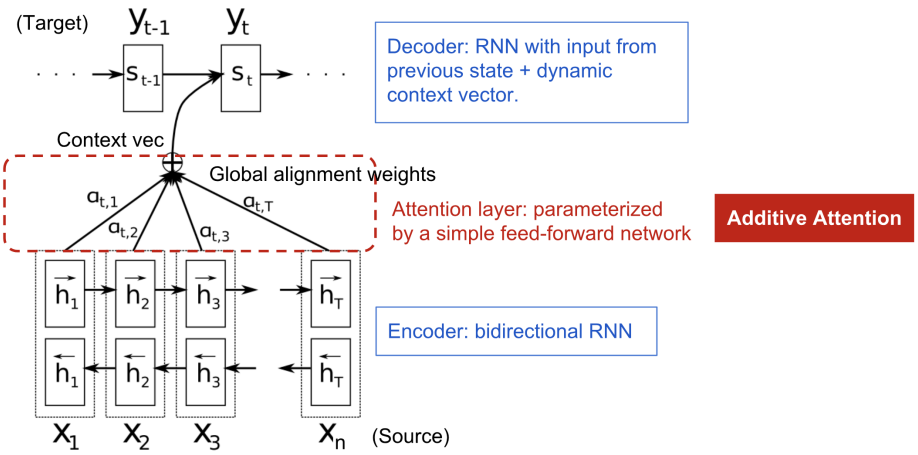
Attention in Neural Machine Translation



- The attention mechanism originally helps memorize long source sentences in neural machine translation (NMT).

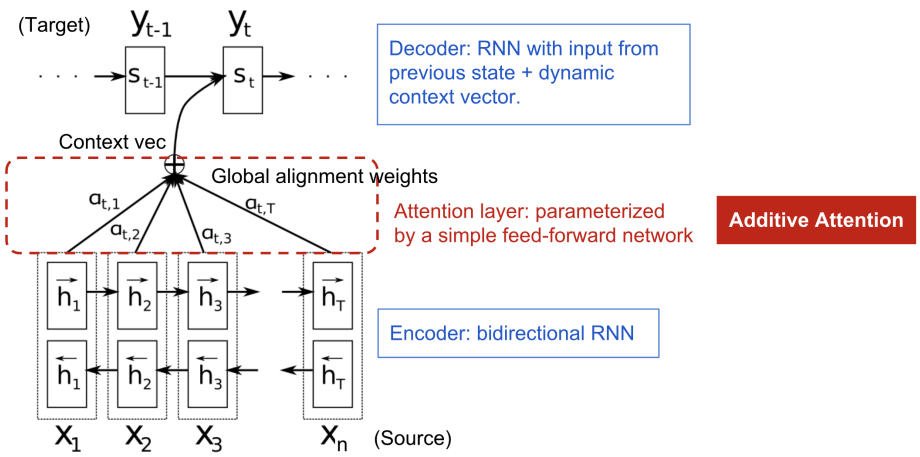
Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," *ICLR*, 2015.

Attention in Neural Machine Translation



- The context vector has access to the entire input sequence. The alignment between the source and target is learned and controlled by the context vector.

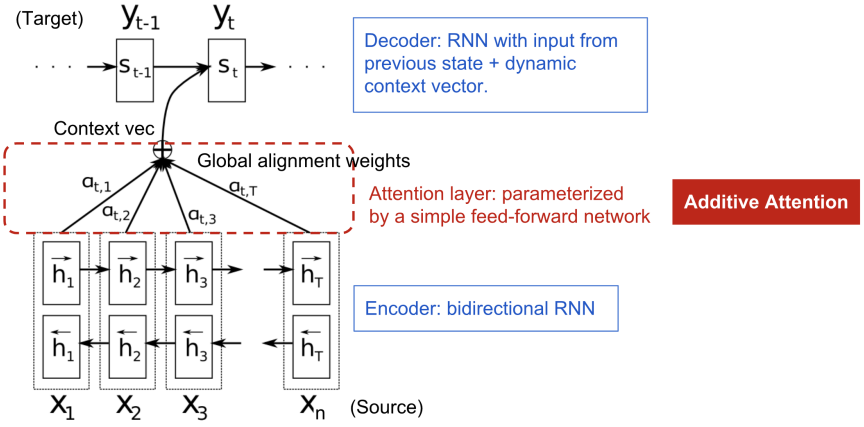
Attention in Neural Machine Translation



- The context vector $\mathbf{c}_t$ is a sum of hidden states $\mathbf{h}_i$ (bidirectional RNN) of the input sequence, weighted by alignment scores $\{\alpha_{t,i}\}$.

$$\mathbf{c}_t = \sum_{i=1}^{n} \alpha_{t,i} \mathbf{h}_i$$

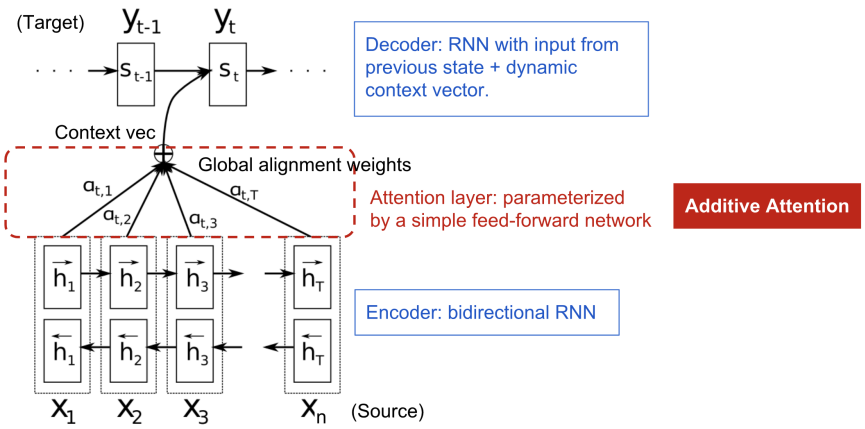
Attention in Neural Machine Translation



- The alignment model assigns a score $\alpha_{t,i}$ to the pair of input at position i and output at position t, (y_t, x_i). That is, how much of each source hidden state should be considered for each output.

$$\alpha_{t,i} = \text{softmax}_i(\text{score}(\mathbf{s}_t, \mathbf{h}_i))$$

Attention in Neural Machine Translation



- The alignment score α is parameterized by a feed-forward network.

$$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \mathbf{v}_a^T \tanh\left(\mathbf{W}_a[\mathbf{s}_t; \mathbf{h}_i]\right)$$

where $\mathbf{v}_a$ and $\mathbf{W}_a$ are weight matrices to be learned.

Different Attentions and Alignment Score Functions

Attention Name	Alignment Score Function $\text{score}(\mathbf{s}_t, \mathbf{h}_i)$
General	$\mathbf{s}_t^T \mathbf{W}_a \mathbf{h}_i$ $\mathbf{W}_a$ is a trainable weight matrix for attention.
Content-based	$\cos(\mathbf{s}_t, \mathbf{h}_i)$
Additive	$\mathbf{v}_a^T \tanh(\mathbf{W}_a[\mathbf{s}_t; \mathbf{h}_i])$
Location-based	$\text{softmax}(\mathbf{W}_a \mathbf{s}_t)$ Alignment only depends on the target position.
Dot-product	$\mathbf{s}_t^T \mathbf{h}_i$
Scaled dot-product	$\mathbf{s}_t^T \mathbf{h}_i / \sqrt{n}$ n is the dimension of the source hidden states.

Thang Luong, Hieu Pham, Christopher D. Manning. "Effective Approaches to Attention-based Neural Machine Translation," *EMNLP*, 2015.

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

Hierarchical Attention Network

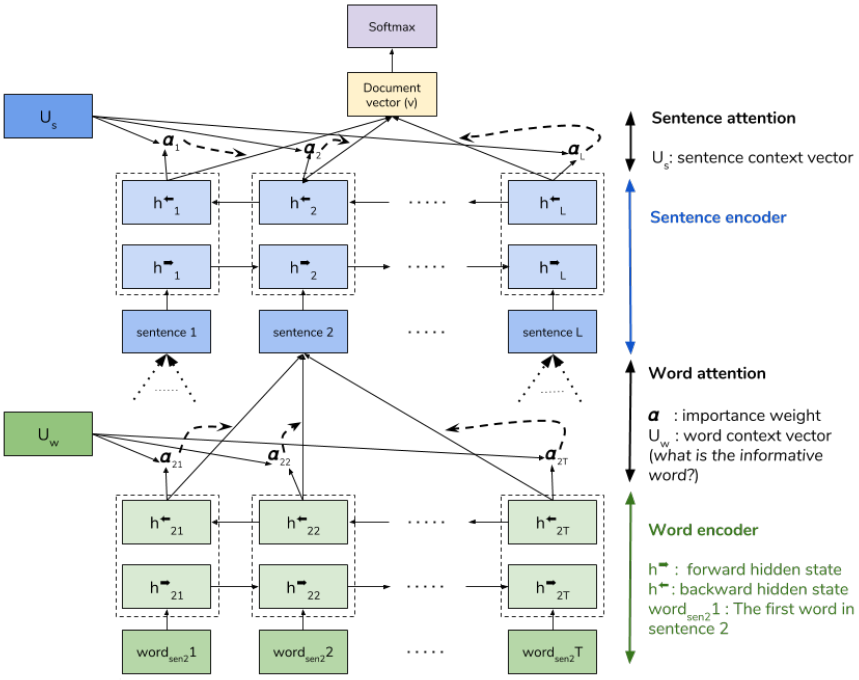


Figure: The transformer has an encoding and a decoding component. For machine translation, it takes a sentence and outputs its translation.

Hierarchical Attention Network

- Hierarchical: Deriving sentence meaning from the words and then deriving the meaning of the document from these sentences.
- Attention model is used so that a sentence vector has more attention on "important" words.
- The attention model consists of bidirectional RNN and attention network. It aggregates the representation of the words to form a sentence vector. This weighted sum embodies the whole sentence.
- The same procedure applies to sentence vectors so that the final vector embodies the gist of the whole document.

Z. Yang, et al., "Hierarchical Attention Networks for Document Classification," *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Jun. 2016.

Transformer

- Temporal convolutional network (TCN) does not do well with dependencies.
- Attention-based RNN does not do well with parallelization.
- Transformer uses TCN together with attention models.

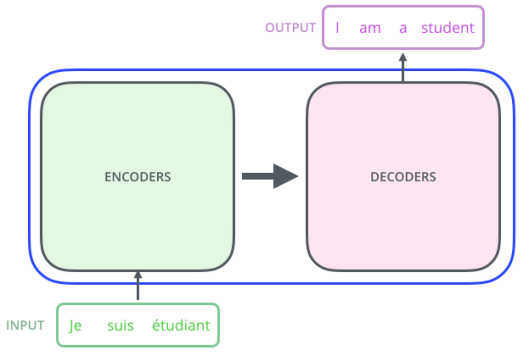


Figure: The transformer has an encoding and a decoding component. For machine translation, it takes a sentence and outputs its translation.

Transformer

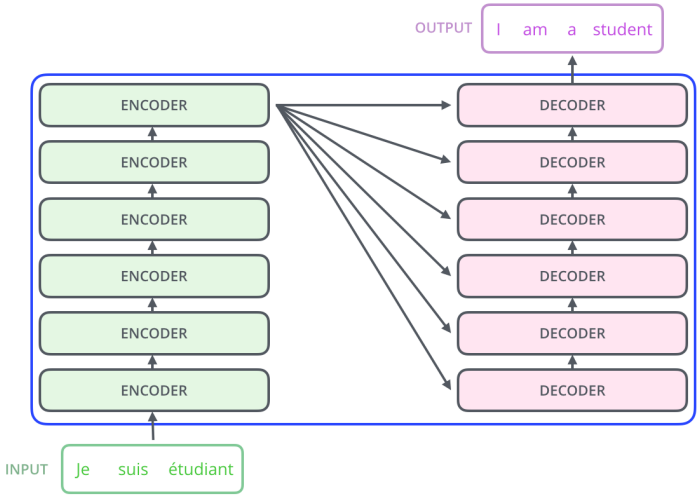
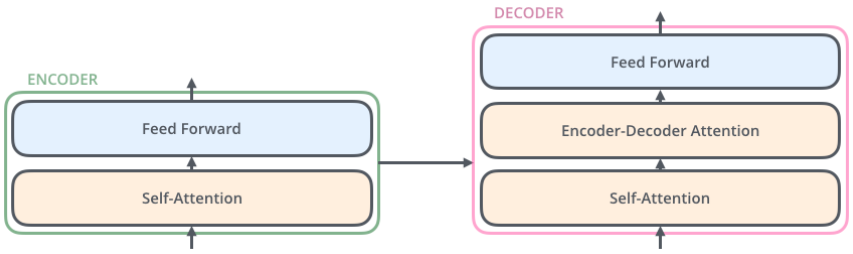


Figure: The encoding component is a stack of encoders (six encoders used in the paper). The decoding component is a stack of decoders of the same number.

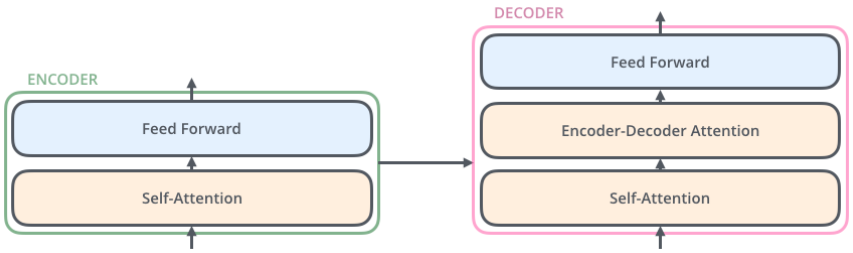
Ashish Vaswani, et al., "Attention Is All You Need," *NIPS*, 2017.

Transformer



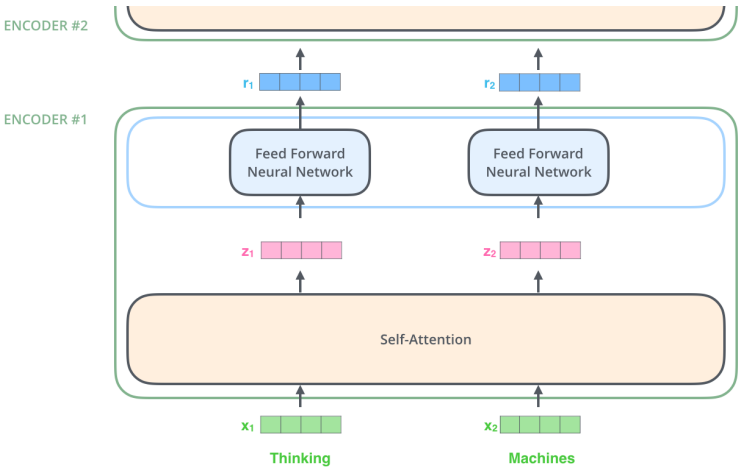
- The encoder has a self-attention layer followed by a feed-forward neural network.
- The self-attention layer helps the encoder look at other words in the input sentence as it encodes a specific word.
- The same feed-forward network is applied to each position.

Transformer



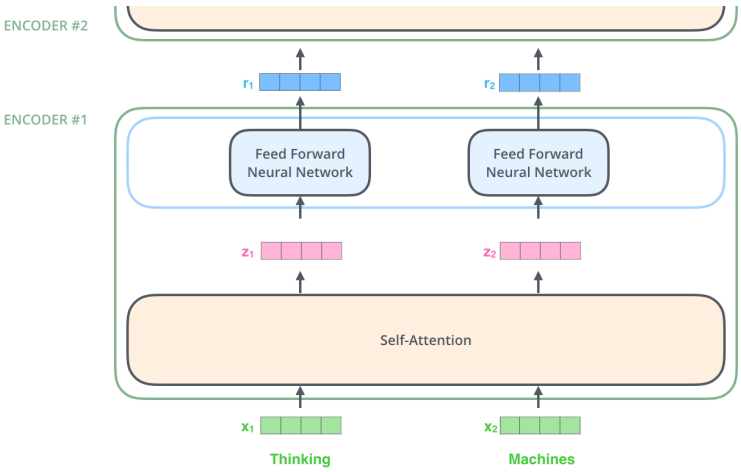
- The decoder has these two layers, but between them is an attention layer that helps the decoder focus on the relevant parts of the input sentence.

Transformer – Encoder



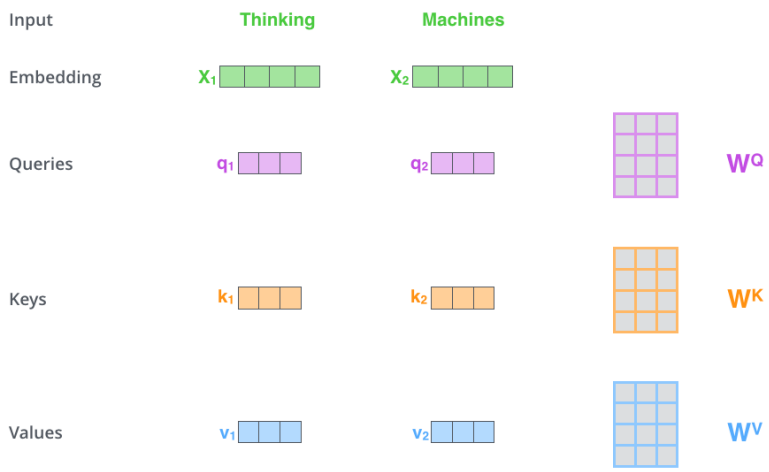
- Word embedding: word $\rightarrow$ vector
- At the bottom encoder, each input word is embedded into a vector of length 512.
- Other encoders take the output vectors from the encoder below.

Transformer – Encoder



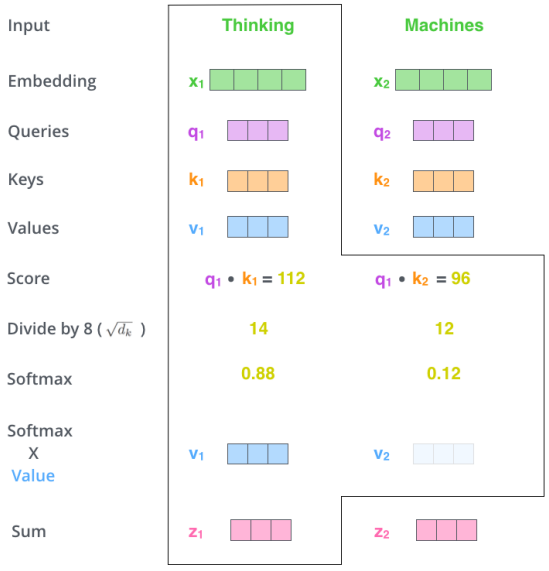
- The word dependencies are captured in the self-attention layer.
- The vectors are executed in parallel while flowing through their own paths in the feed-forward layer.

Transformer – Self-Attention Layer



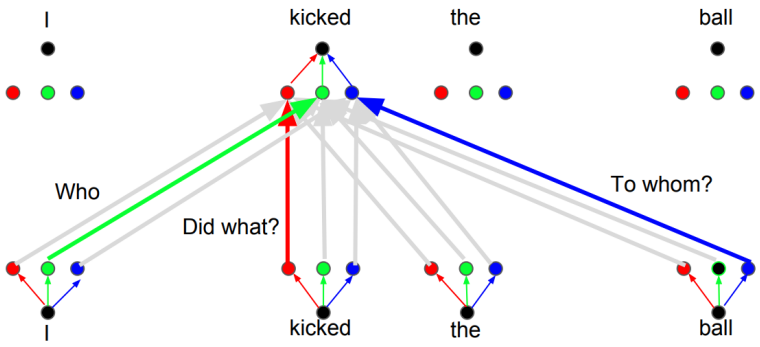
- A query vector $\mathbf{q}$, a key vector $\mathbf{k}$ and a value vector $\mathbf{v}$ are created from each input vector. These vectors are of length 64.
- e.g., $\mathbf{q}_1 = \mathbf{x}_1 \mathbf{W}^Q$. The generating matrices $\mathbf{W}$ can be trained.

Transformer – Self-Attention Layer



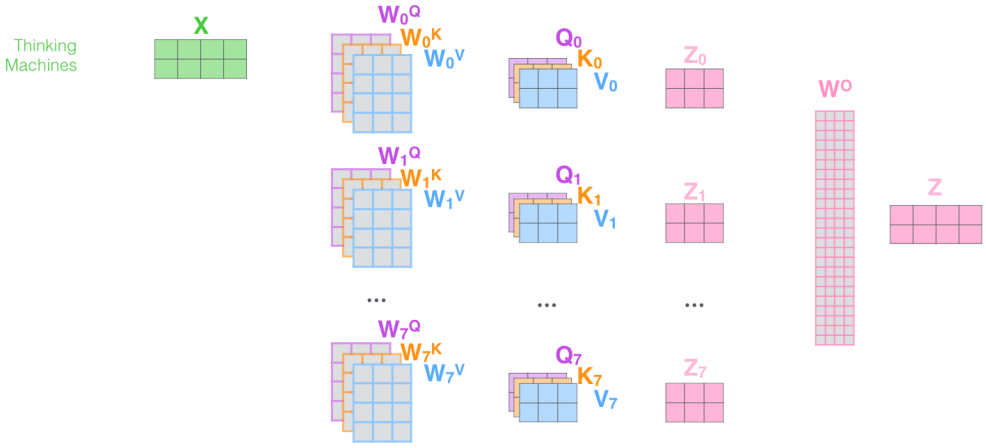
- Score is a dot product of query and key vectors.
- Division by 8 for more stable gradients. $d_k = 64$ is the vector length.
- Sum is the weighted value vectors. It is the self-attention layer output vector of "Thinking".
- Matrix calculation by packing the embedding vectors into matrix $\mathbf{X}$.

Transformer – Multihead Attention



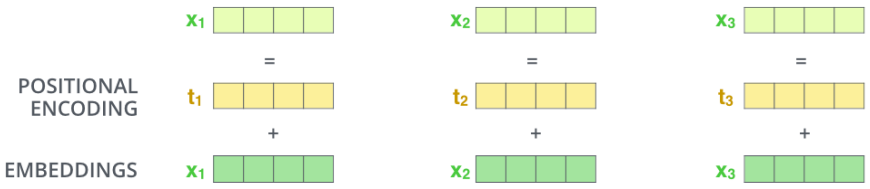
- Multihead attention – When translating a word, you may pay different attention to each word based on the type of question being asked.

Transformer – Multihead Attention



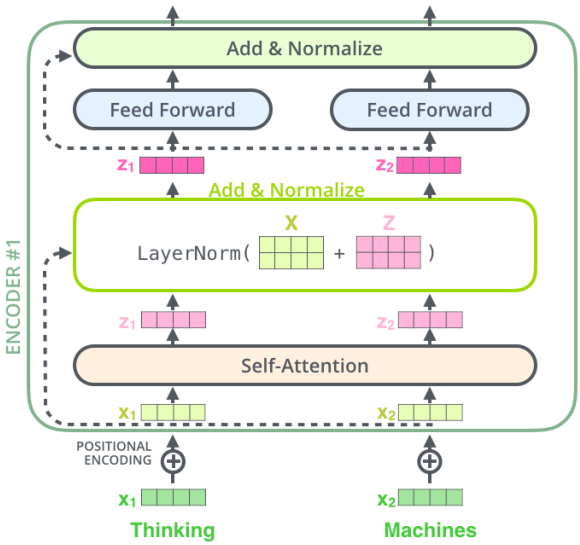
- Multihead attention – Split into 8 heads.
- Concatenate the resulting $\mathbf{Z}_i$ matrices, $i = 0, 1, \ldots, 7$, then multiply with weight matrix $\mathbf{W}^O$ to produce the output.

Transformer – Positional Encoding



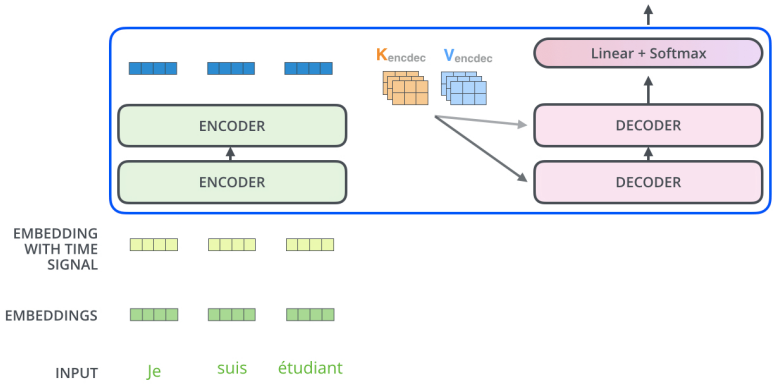
- A positional encoding vector is added to each input embedding to account for the order of the words in the input sequence.
- These vectors follow a specific pattern that the model learns.
- These added values to the embedding provide distance information between the embedding vectors when they are projected into Q/K/V vectors.

Transformer – Residuals



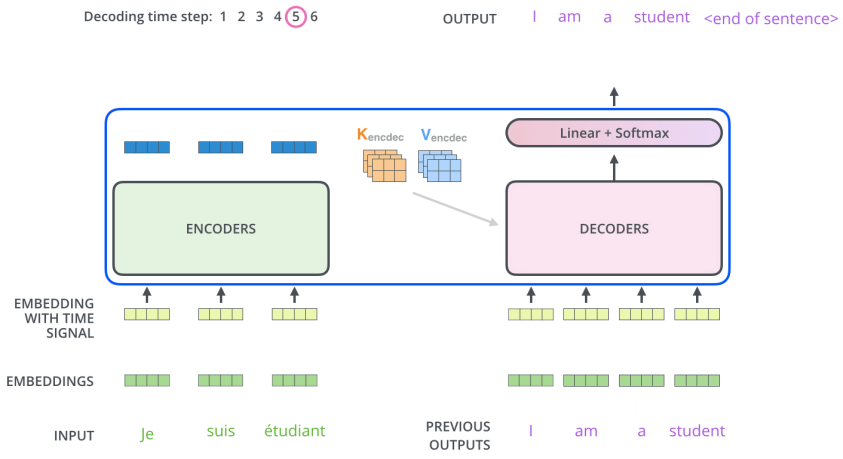
- There is a residual connection followed by a layer-normalization step.

Transformer – Decoder



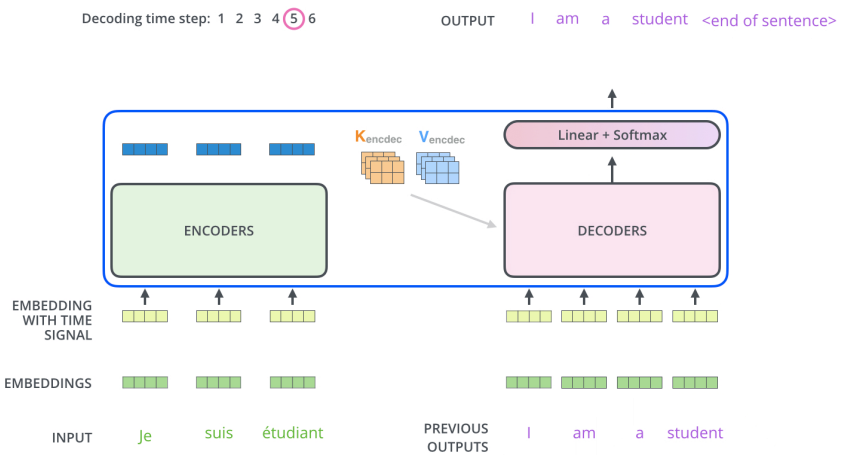
- The output of the top encoder is transformed into a set of attention vectors $\mathbf{K}$ and $\mathbf{V}$.
- These vectors are used by each decoder in the "encoder-decoder attention" layer which helps the decoder focus on appropriate places in the input sequence.

Transformer – Decoder



- The positional encoding (time signal) is added to the decoder input to indicate the position of each word.
- Each step in the decoding phase outputs an element of the sequence. The output of each step is fed to the bottom decoder in the next time step. The model produces the output words one at a time.

Transformer – Decoder

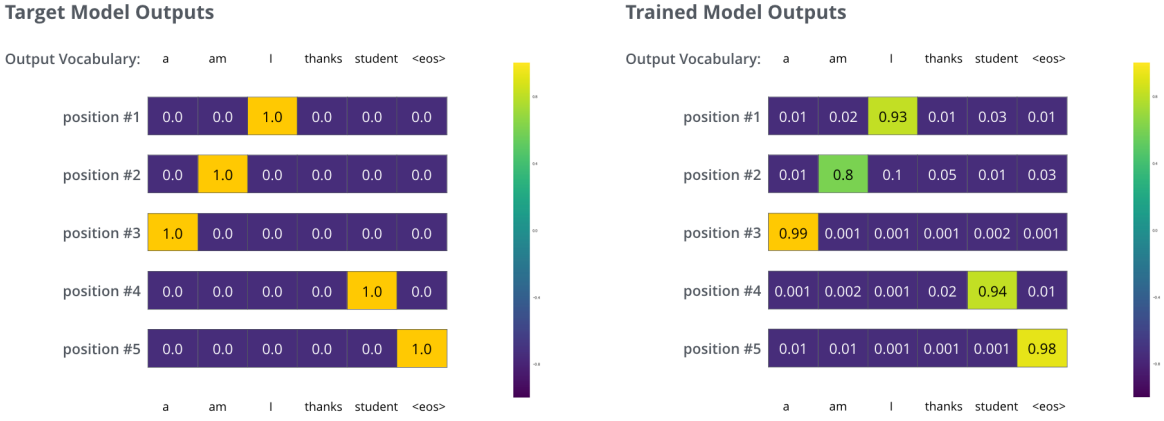


- The self-attention layer is only allowed to attend to earlier positions in the output sequence. Mask future positions with -inf.
- The "encoder-decoder attention" layer creates the Query matrix from the layer below it and the Key and Value matrices from the output of the encoder stack.

Transformer – Training

- The output layer is a fully-connected linear layer followed by a softmax layer. The softmax makes the output logits vector into a probability distribution. With one-hot encoding, the largest probability corresponds to the word in the output vocabulary.

- The loss function is the cross entropy between the true distribution (one-hot encoding) and the model distribution.



Pervasive Attention

- An alternative method of neural machine translation that is based on deep 2D CNN.
- Pervasive Attention – Attention-like properties are pervasive throughout the network structure.
- The CNN is over a 2D grid of the positions in source and target sequences.
- The convolutional filters are masked to prohibit accessing information derived from future tokens in the target sequence. (autoregressive model)

Maha Elbayad, Laurent Besacier, and Jakob Verbeek, "Pervasive Attention: 2D Convolutional Neural Networks for Sequence-to-Sequence Prediction," *arXiv*, 2018.

Pervasive Attention

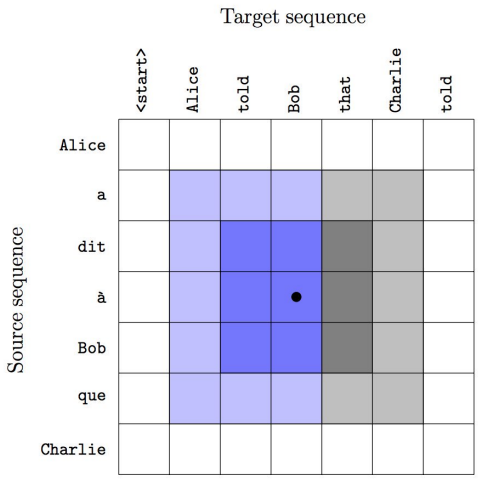


Figure: Using masked 3×3 convolutional filters. Blue ones are the receptive fields (after one and two layers), while grey ones are masked.

- Attention-like capabilities by construction:
  Every layer of the CNN computes features of the source tokens, based on the target sequence produced so far. These features are used to predict the next output token.
- The method learns deep feature hierarchies based on a stack of 2D convolutional layers, and it benefits from parallel computation during training.