# A Convolutional Encoder Model for Neural Machine Translation

Presented by: Dushyanta Dhyani, Pravar Mahajan (The Ohio State University)
Paper By: Jonas Gehring, Michael Auli, David Grangier, Yann N. Dauphin (Facebook AI Research)

## Neural Machine Translation

End to end deep learning based machine translation model.

Current State of the Art uses an encoder-decoder model.

**Encoder:** Creates a hidden representation (encoding) of the input sentence.

**Decoder:** Converts the hidden representation (decoding) into a sentence in target language.

Encoder and Decoder are generally Recurrent Neural Networks. Current state of the art models use separate Bi-LSTMs for the encoder and decoder components

**Drawback:** Slow, since recurrent nets are not easily parallelizable.

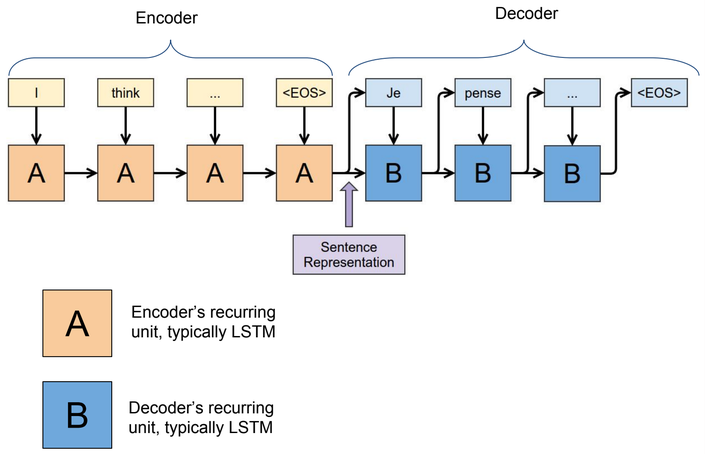



Image source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

## Convolutional Encoder

Convolution operations are fast and easy parallelizable.

**Challenges:**

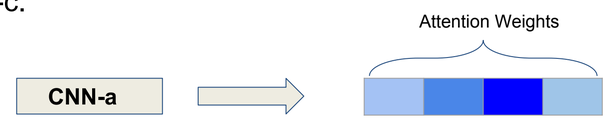
- Convolution does not fit naturally into sequence modeling task due to their spatial nature.
- Word ordering is lost.

**Solution:**

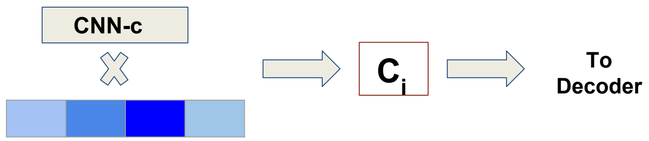
- Include position embedding.

**Encoder Architecture**

- Two Set of Convolution Layers :
  - cnn-a - Responsible for encoding the sentence and is used to generate attention weights that are applied to cnn-c.



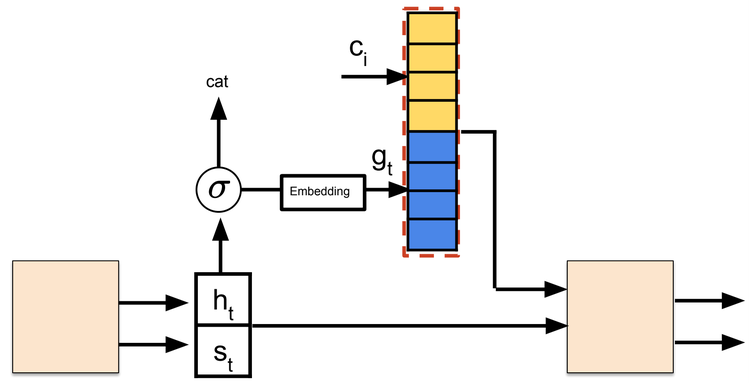

  - cnn-c - Responsible for generating the conditional input $c_i$ to the decoder.




- Both CNN-a/CNN-c have:
  - Input Representation as Word Embeddings + Position Embeddings
  - Fixed Kernel width and fixed number of convolution layers (which are different for CNN-a and CNN-c)
  - Contains Residual connections bypassing convolution layers to ease learning for deep networks.

**Decoder Architecture**

- Recurrent Neural Network (GRU/LSTM/RNN etc)
- Generates attention weights using CNN-a and current hidden state $h_t$.
- Applies the attention weights over CNN-c to get input context $c_i$
- Input to the next step of the recurrent network is concatenation of input step $c_i$ and embedding $g_t$ of the current output $y_t$




## The Complete Architecture

Tying the encoder and decoder together, the whole model looks like this:

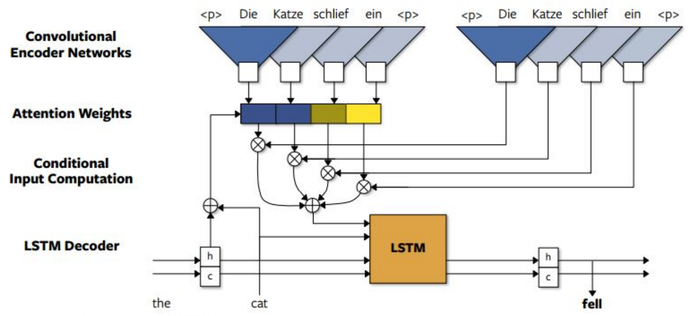



Image source: https://arxiv.org/pdf/1611.02344.pdf