Attention Mechanisms and Transformers

Convolutional Encoder-Decoder

- Drawbacks of **Recurrent Neural Networks**:
  - **Sequential** computation prohibits **parallelization**;
  - Long-term dependencies are **hard to learn**;
- A possible solution is to replace the **RNN encoder** with a **hierarchical 1D CNN - Convolutional Encoder-Decoder**;
  - Encoder is parallelizable, but decoder still requires sequential computation - the model is still **auto.regressive**.

Self-Attention and Transformer Networks

- We want NNs that **automatically weight** the **relevant** parts of the input, so we use **attention mechanisms**;
  - Performance gain;
  - None or few parameters;
  - Fast - parallelizable;
  - Tool for interpreting predictions.

Attention Mechanism: Recap

1. We have a **query vector** q and **input keys (vectors)** $H = [h_1, h_2, \ldots, h_L]^T$;
   - Input vectors appear in two roles: **keys** and **values**;
     - **Keys** are used to **compute the attention scores**;
     - **Values** are used to **compute the weighted average**;
2. We compute **affinity scores** $s_1, s_2, \ldots, s_L$ between q and h_i; there are several ways of comparing q and h_i:

- **Additive attention**: $s_i = w^T tanh(Ah_i + Bq)$;
- **Biliner attention**: $s_i = q^T U h_i$;
- **Dot product attention**: $s_i = q^T h_i$; but queries and keys must have the same size;

3. We convert these scores to **probabilities**: $p = softmax(s)$;
4. We use this to output a representation as a **weighted average**: $c = H^T p = \sum_{i=1}^{L} p_i h_i$.

Self-Attention

- **Self-attention** is a **special case** of **attention**;
- At each position, the encoder attends to the other positions in the encoder itself - same for the decoder;
- Self-attention for a sequence of length L:
  - **Query vectors**: $Q = [q_1, q_2, \dots, q_L]^T$;
  - **Key vectors**: $K = [k_1, k_2, \dots, k_L]^T$;
  - **Value vectors**: $V = [v_1, v_2, \dots, v_L]^T$;

  1. Compute **affinity scores** $S = QK^T$;
  2. Convert these scores to **probabilities**: $P = softmax(S)$;
  3. Output the weighted average of the value: $Z = PV$.

Transformer

- **Transformer** is a **neural network architecture** that uses **self-attention** in the encoder instead of RNNs/CNNs;
- Each word state attends to all the other words;
- Each self-attention is followed by a feed-forward transformation;
- Do several layers of this;
- Do the same for the decoder, attending only to already generated words.

Multi-Head Attention

- **Multi-head attention** is a **variant of self-attention** that allows the model to **jointly attend to information from different representation subspaces** at different positions;
- Define h **attention heads**;
- Apply attention in multiple channels, concatenate the outputs and pipe through linear layer: $MultiHead(X) = Concat(Z_1, \ldots, Z_h)W^O$, where $Z_i = Attention(XW_i^Q, XW_i^K, XW_i^V)$;

Positional Encoding

- **Positional encoding** is a **technique** used to **inject information about the relative or absolute position of tokens in a sequence**;