Attention Transformers

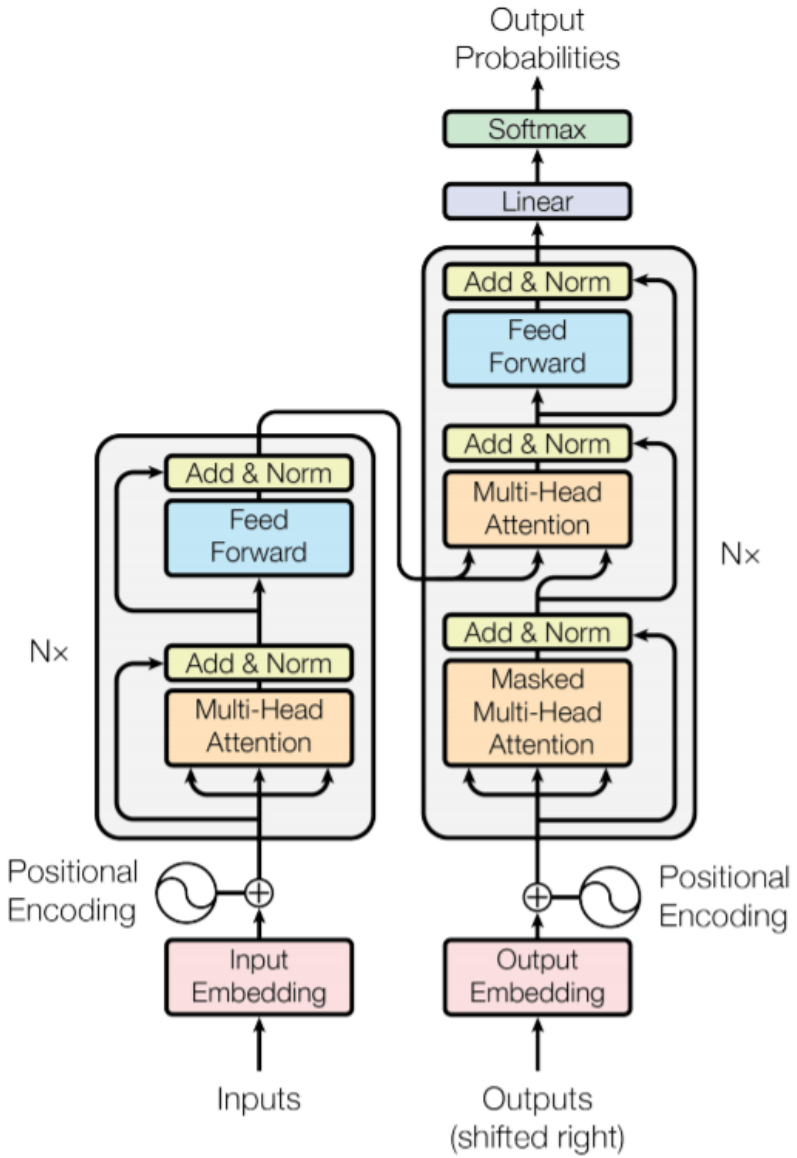
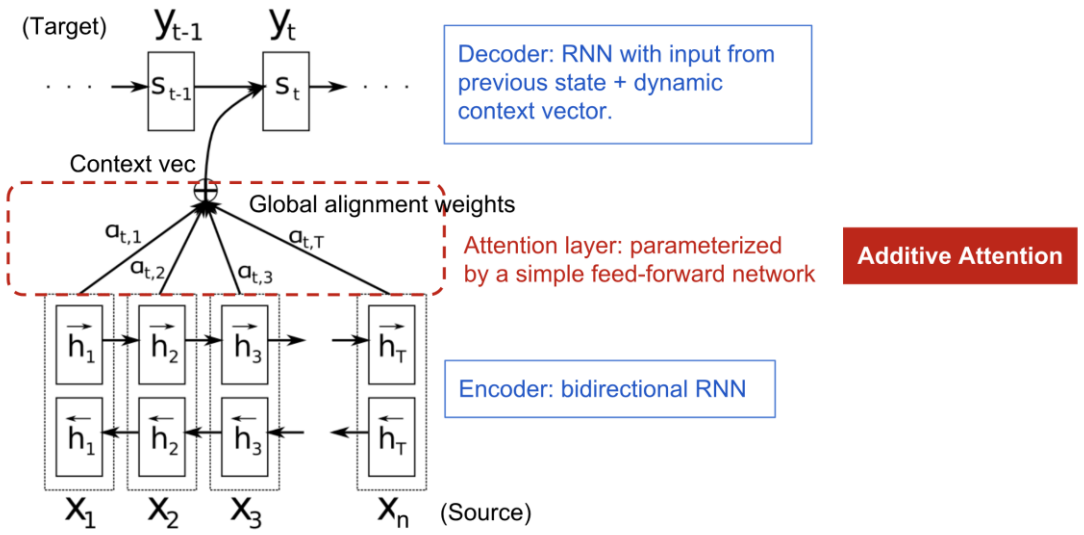


Figure 1: The Transformer - model architecture.

Attention

- Computes attention weights to reweight the input to get a context vector
  - Use a separate neural network to compute weights

$$\text{score}(s_t, h_i) = v_a^T \tanh(W_a[s_t; h_i])$$
$$a_{ti} = \text{softmax}\left(\text{score}(s_{t-1}, h_i)\right)$$
$$c_t = \sum_i a_{ti} h_i$$



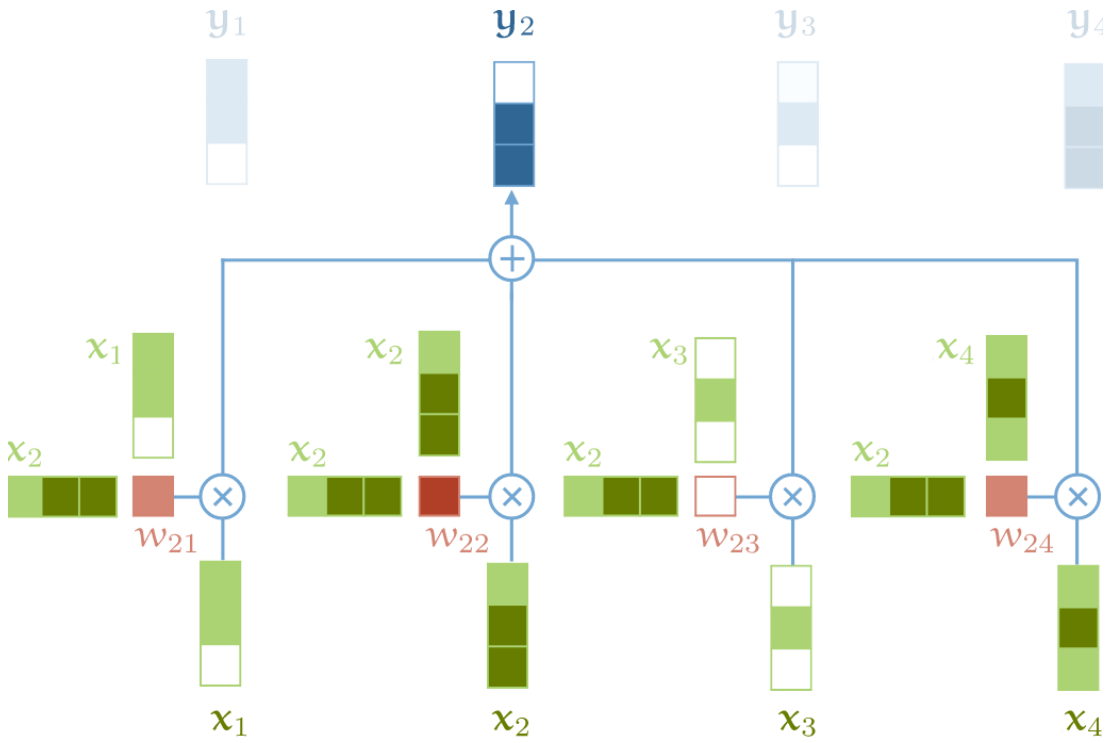
Self-attention

- Attention as dot product between input tokens

$$y_i = \sum_j w_{ij} x_j$$

$$w_{ij} = \frac{\exp(x_i x_j)}{\sum_k \exp(x_i x_k)}$$

- Re-computes inputs based on context
- Note: no parameters
  - w_{ij} is computed per input sequence
- Effectively, an operation on 'sets'
- After shuffling same activations



http://peterbloem.nl/blog/transformers

Queries, keys and values

- Add some flexibility by linear transformations

$$q_i = W_q x_i, k_i = W_k x_i, v_i = W_v x_i$$

- Adding some controllable parameters to self-attention

$$w'_{ij} = q_i^T k_j$$

$$w_{ij} = \frac{\exp(w'_{ij})}{\sum_k \exp(w'_{ik})}$$

$$y_i = \sum_j w_{ij} v_j$$

Scaling the product

- The softmax is sensitive to large input
- Normalize by number of dimensions

$$w'_{ij} = \frac{q_i^T k_j}{\sqrt{d}}$$

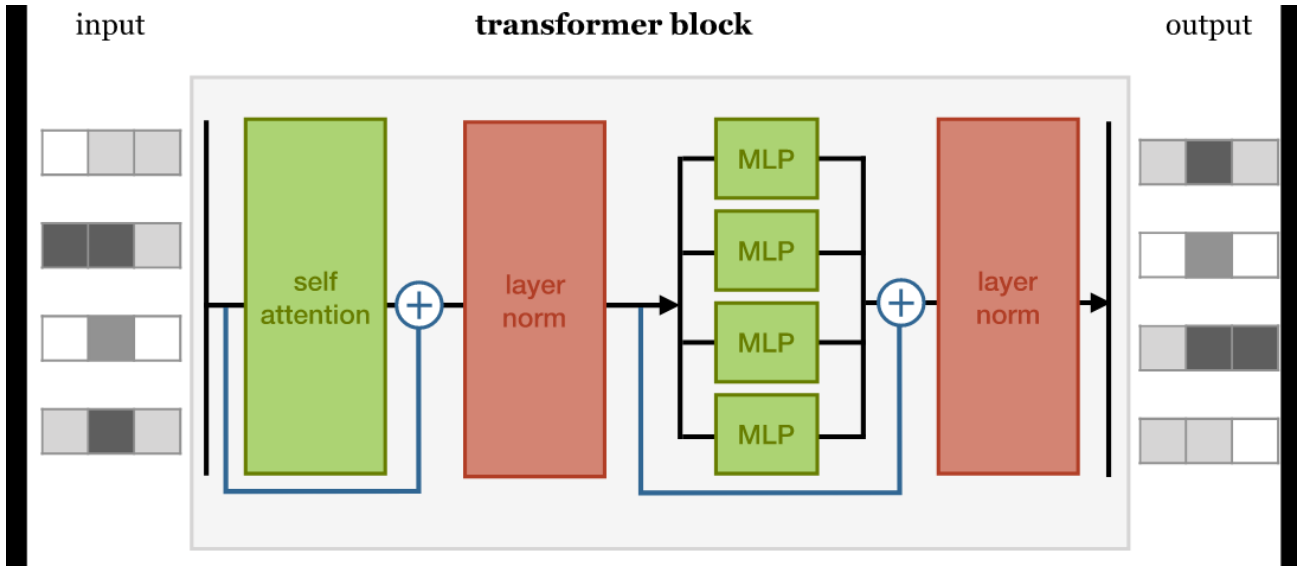
- $c\sqrt{d}$ is approximately the norm of a d-dimensional vector with all values c

Multi-head attention

- We can replace weight vectors with weight matrices
- Each matrix row is a “separate attention”
- Good if there are many interpretations of a word or a token
- One attention per interpretation (intuitively)

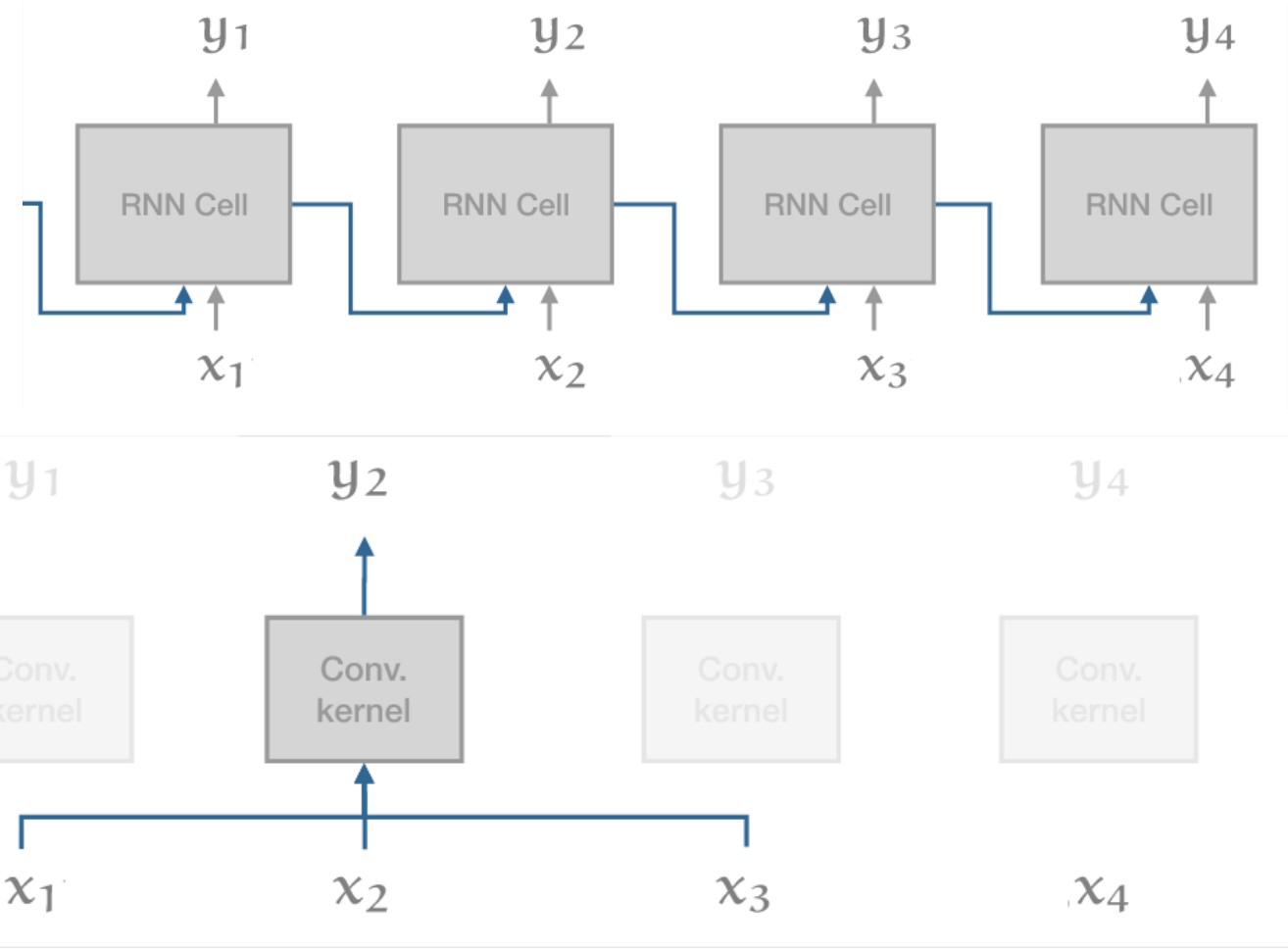
Transformers

- A stackable architecture with blocks containing self-attention and MLPs
- Using position embeddings to make transformer sensitive to input order



RNN vs transformers

- Parallelism
- Much better efficiency
- Still able to capture long-term



Challenges

- Quadratic cost to input length
  - Transformers XL: divide sequences hierarchically
  - Sparse transformers: self-attention on specific pairs
- Memory intense
  - Half-precision variables (except for loss)
  - Gradient accumulation
  - Gradient checkpointing