Data Mining & Machine Learning

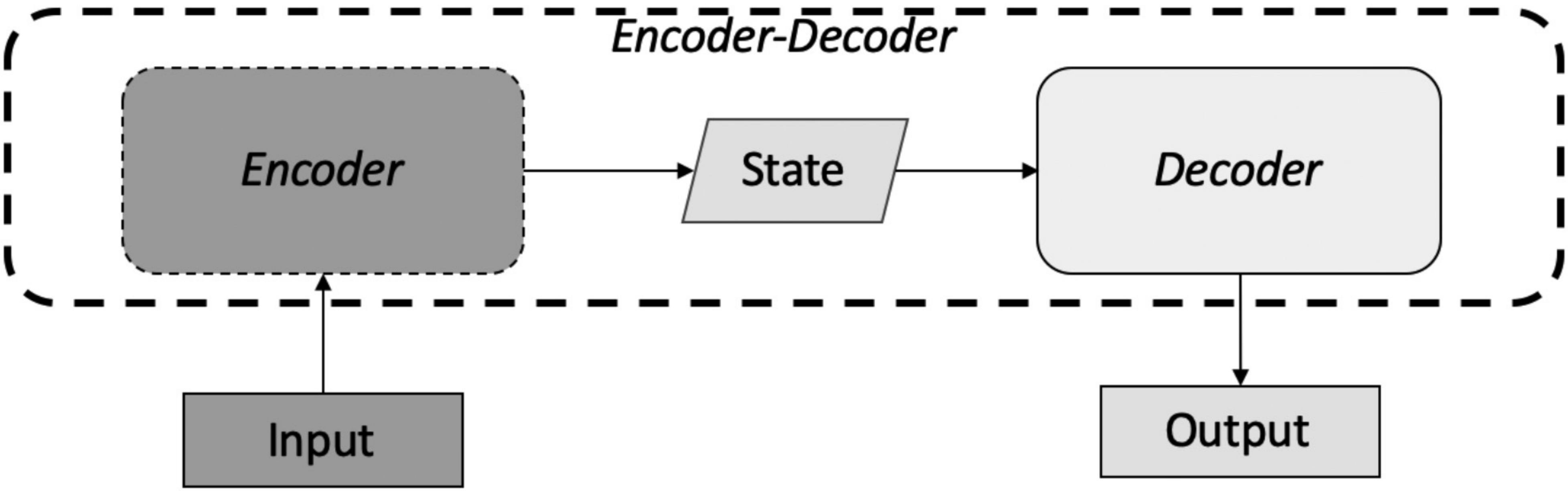
CS37300
Purdue University

Oct 30, 2023

Transformers

Sequence-to-sequence models

- Encoder-decoder architectures

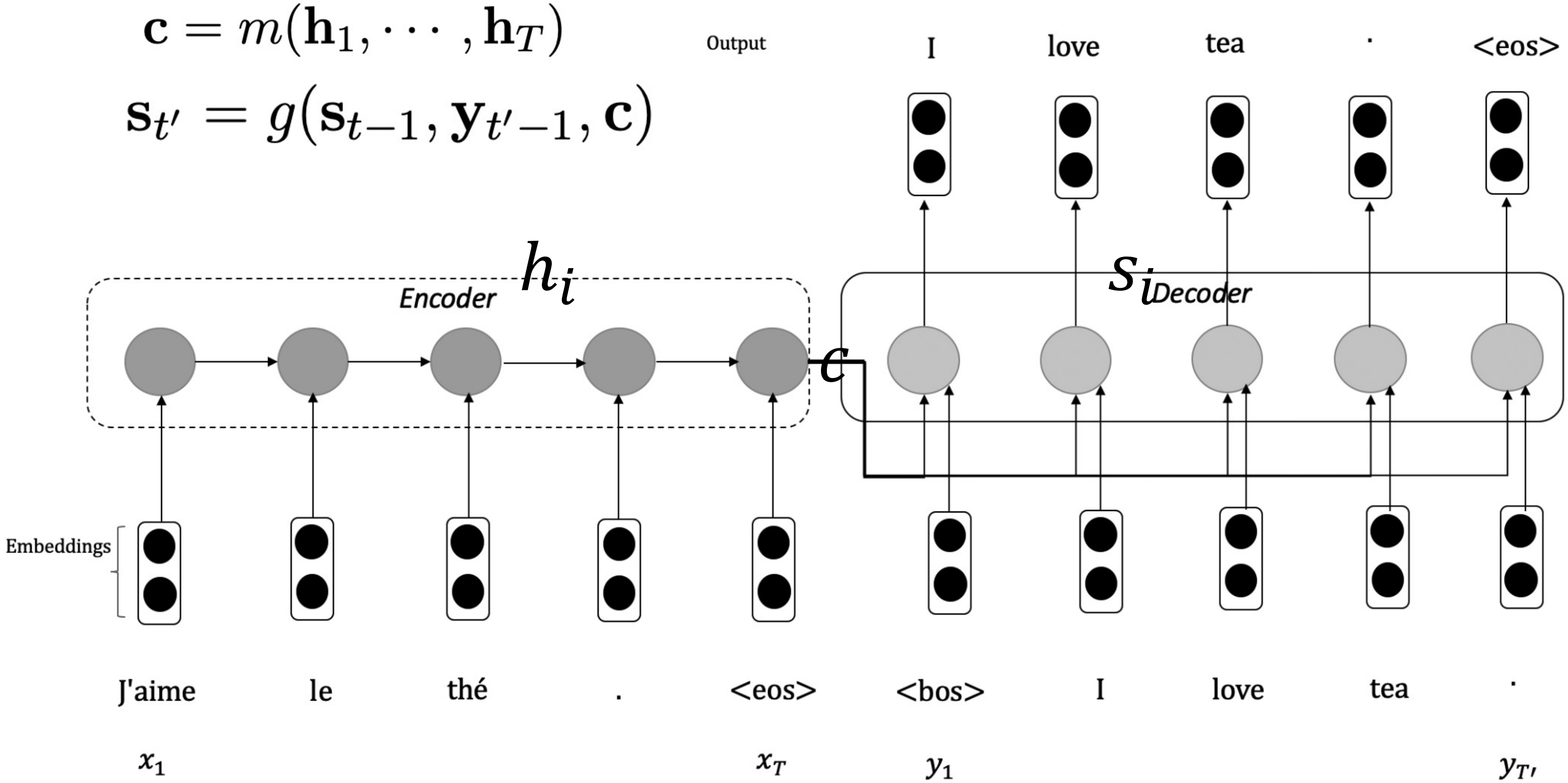


Combine two RNNs to build seq2seq models?

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t)$$

$$\mathbf{c} = m(\mathbf{h}_1, \cdots, \mathbf{h}_T)$$

$$\mathbf{s}_{t'} = g(\mathbf{s}_{t-1}, \mathbf{y}_{t'-1}, \mathbf{c})$$

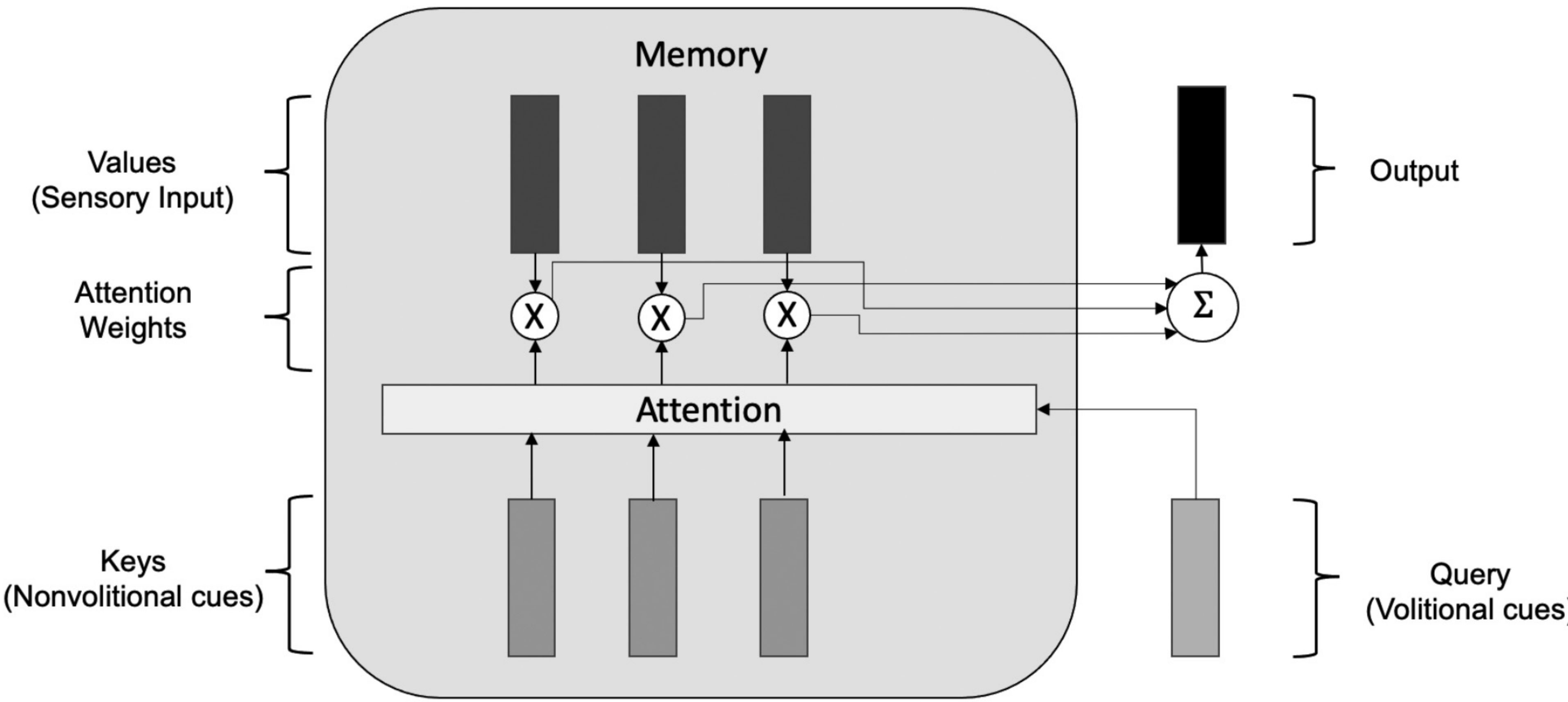


Limitations of RNNs for seq2seq

- A single context vector linking two RNNs may compress too much (loss of information)
- Could potentially link hidden vars of encoder to hidden vars of decoder
- Generally for RNNs:
    - Exploding/vanishing gradients
    - Recurrence relationships make it difficult to parallelize

Attention mechanism

- Focus on selective information



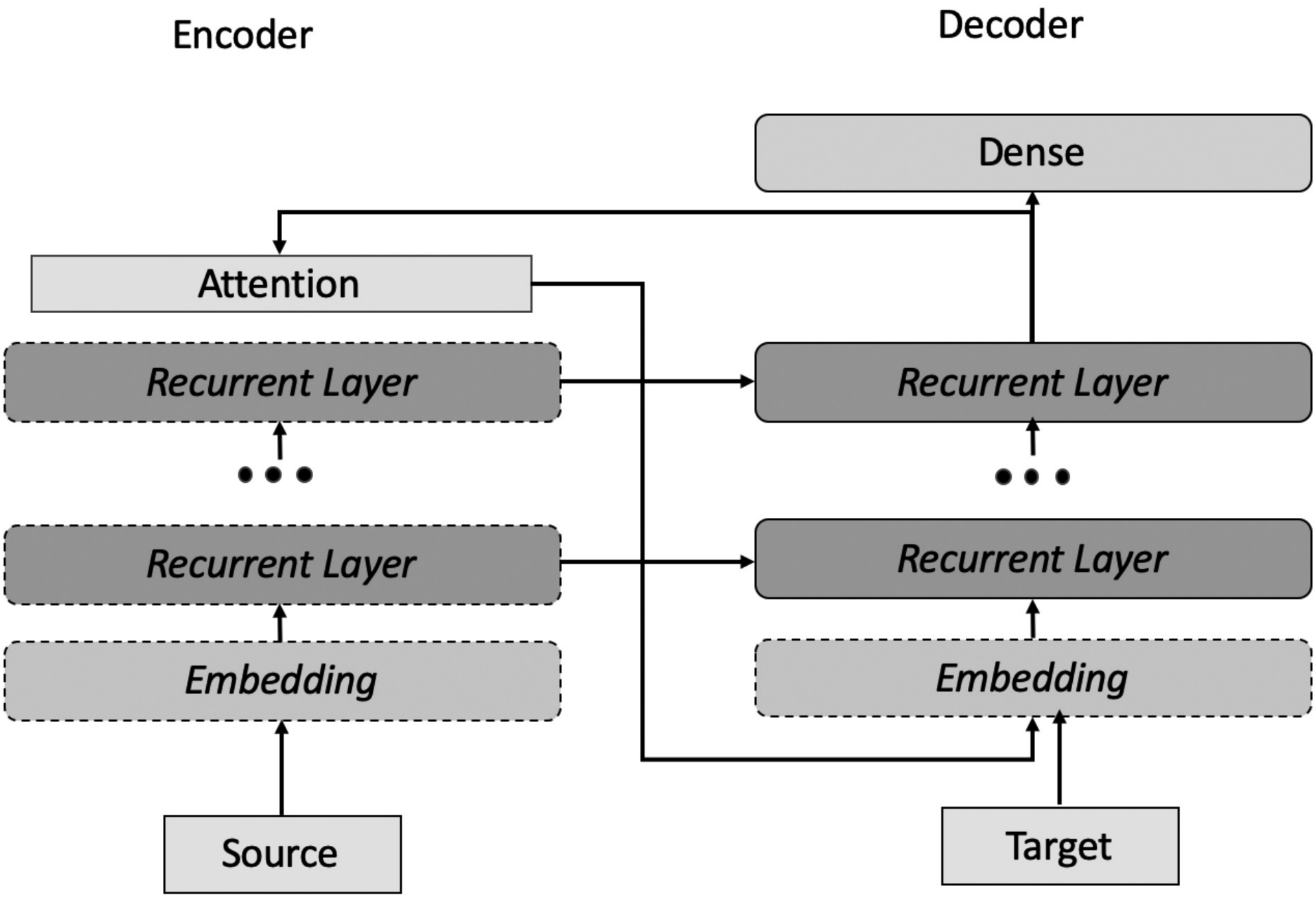
Attention mechanism contd

- Memory: Stores key-value pairs (k_i , v_i)
- For a new query q, calculate its similarity with stores keys:
  - $b_i = sim(q, k_i)$. Normalize so that **b** is a probability distribution
- Based on similarity, form a "weighted value output"
  - $o = \sum_i b_i v_i$

Similarity function

- Used for calculating similarity in score and query
- Any suitable kernel function
  - Dot product
  - Mahalanobis similarity
  - Latent variable based

Encoder-decoder with attention



Implementation of attention

- Several Recurrent layers
- keys,values extracted from encoder states
- Query extracted from decoder’s state at time/step t-1
- Context output from attention used for next state t of the decoder
- Context variable here is a weighted average from several keys/values extracted from the encoder states

Transformers

- Combine CNNs (parallelize computations) and RNNs (capture sequences)
- Main components:
  - Inputs/Outputs tokenized and positionally embedded
    - Orders could be handled in RNNs, but here the feature vectors encode the relative positions
  - Masked multi-head attention
  - [contd later]

Self-attention

- Model dependencies amongst in the input embeddings

- Obtain key,value,query triplets from the same input embedding x_i by learning projection matrices W_k, W_v, W_q

- The query vector $q_i = W_q x_i$, is combined with all other key vectors in the input to generate the output for its own self $\sum q_i k_i^T$

- The outputs are then generated by combining normalized weighted self-query with the value, where d_k is the size of the key vector

$$attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{d_k}}\right)\mathbf{V}$$

Multi-head attention

- Combine several individual attentions learnt above

- Analogous to different convolution filters in CNNs

$$head_i = attention(\mathbf{W}_q{}^i\mathbf{Q}, \mathbf{W}_k{}^i\mathbf{K}, \mathbf{W}_v{}^i\mathbf{V})$$

$$multihead(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathbf{W}_O\, \mathrm{concat}(head_1, \ldots, head_h)$$

- Masked attention to remove the influence of future tokens for the current token in learning

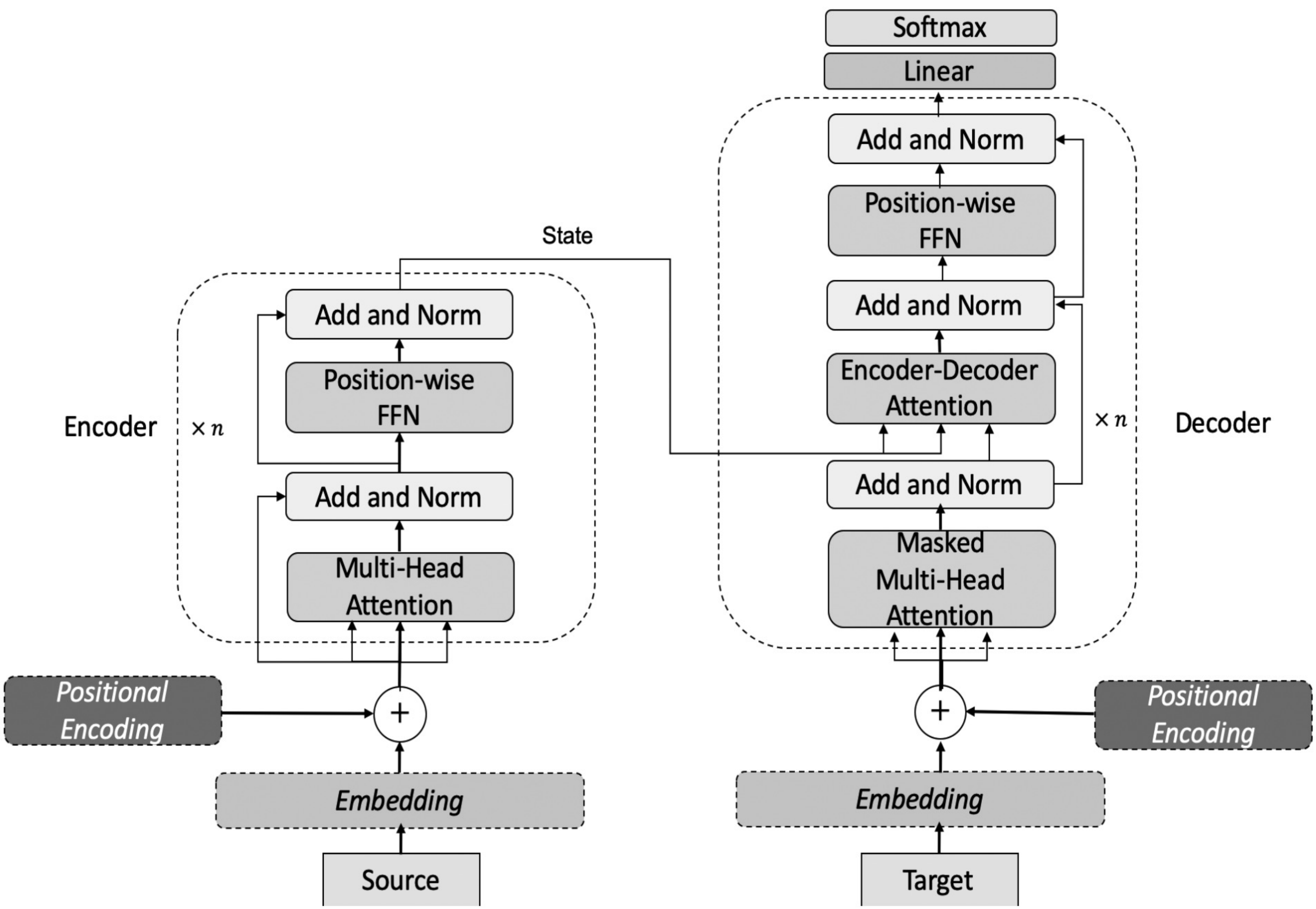
- Encoder only sees attention learning from inputs

- Decoder sees query vectors from target outputs as well output from the encoders

Transformer components (contd)

- Inputs/Outputs embeddings
- Masked multi-head attention
- Feed-forward networks after the attention mechanisms
- Residual (skip) connections
- Layer normalization – for faster convergence

Components of a transformer block



Optimization of NNs

- An accelerated variant of Batch SGD
  - Can parallelize gradient evaluations across multi-core architectures
  - Batch size has a regularizing effect
- Adaptive learning rate
- Challenges
  - Local minima, saddle points
  - Vanishing/Exploding gradients
  - Ill-conditioned Hessians