Introduction to Transformers

Sukanya Nath
Data Science Lab (DSL)
University of Bern

Fall 2023

Language Models

Estimating the likelihood of the next word given previous words

For example,

$$P(The\ sea\ is\ red) = \mathrm{P}(\mathrm{The, sea, is, red})$$
$$= \mathrm{P}(The)\mathrm{P}(sea|The)\mathrm{P}(is|The\ sea)\mathrm{P}(\mathrm{red}|\mathrm{The\ sea\ is})$$
$$\approx \mathrm{P}(The)\mathrm{P}(sea|The)\mathrm{P}(is|\ sea)\mathrm{P}(\mathrm{red}|\mathrm{is})^{*}$$

*Approximation based on Markov's assumptions

1. https://web.stanford.edu/~jurafsky/slp3/3.pdf

Language Models

Estimating the likelihood of the next word given previous words

For example,

$$P(The\ sea\ is\ red) = \mathrm{P}(\mathrm{The}, \mathrm{sea}, \mathrm{is}, \mathrm{red})$$
$$= \mathrm{P}(The)\mathrm{P}(sea|The)\mathrm{P}(is|The\ sea)\mathrm{P}(\mathrm{red}|\mathrm{The\ sea\ is})$$
$$\approx \mathrm{P}(The)\mathrm{P}(sea|The)\mathrm{P}(is|\ sea)\mathrm{P}(\mathrm{red}|\mathrm{is})^{*}$$

What could be some drawbacks?

*Approximation based on Markov's assumptions
1. https://web.stanford.edu/~jurafsky/slp3/3.pdf

Recurrent Neural Networks (RNNs)

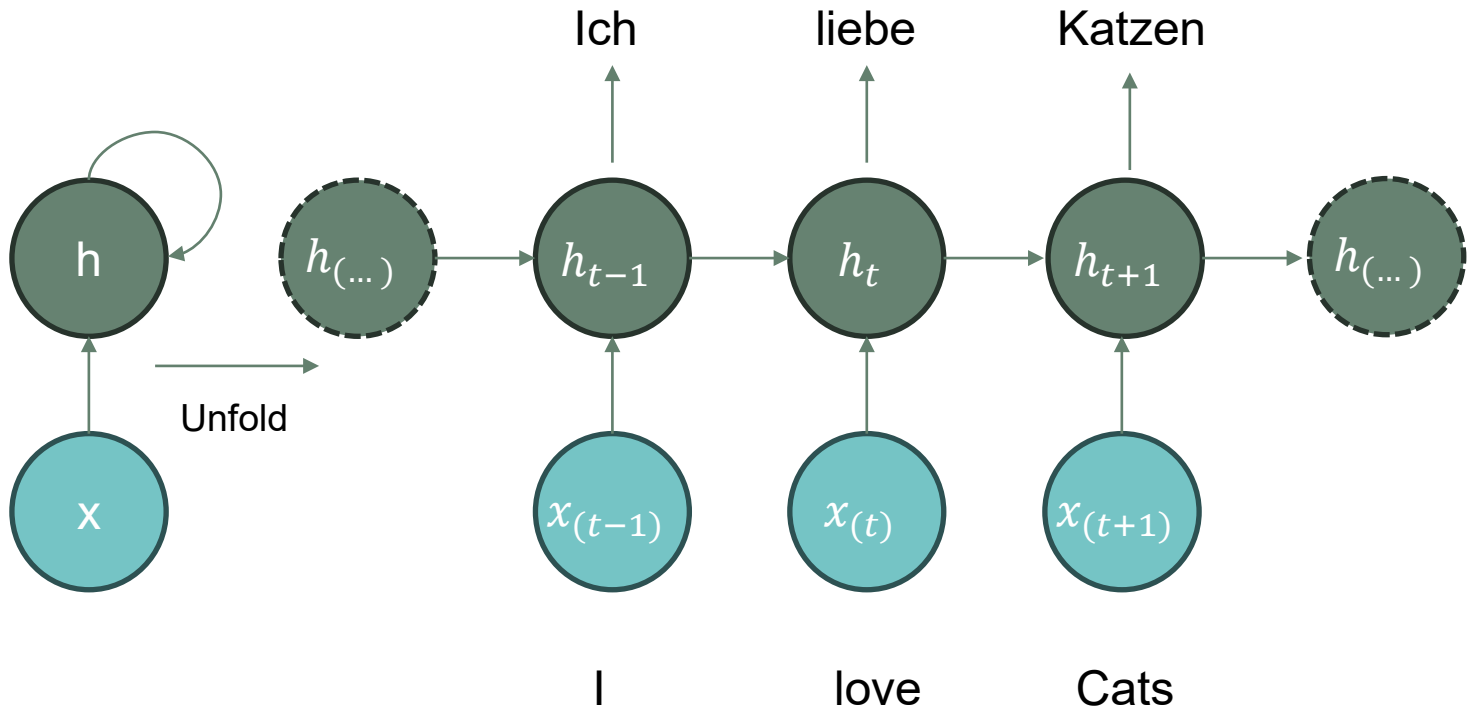
Introduce a latent variable h_t state such that, $y_t = g(h_t)$

where,

$h_t = f(h_{t-1}, x_t)$

x_t is the input at time t

y_t is the output at time t



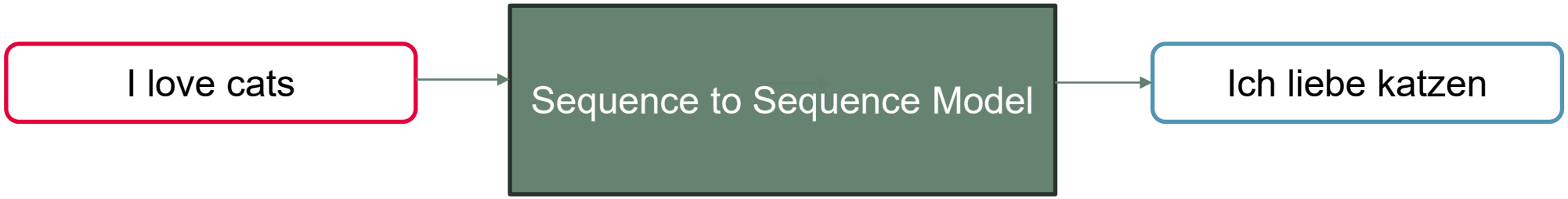
Shortcomings of RNNs

1. Sequential processing word by word (computation for time step t cannot be done until the computation for the time step t-1 has been completed slowing down training and inference).
2. Vanishing or exploding gradients during backpropagation through time.
3. Difficulty in accessing information from a long time ago.

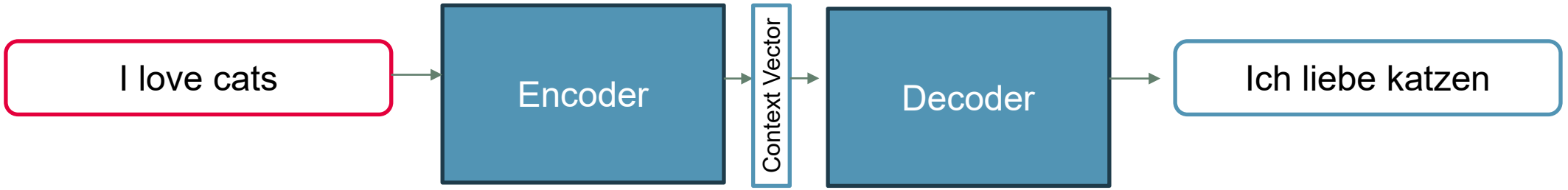
Attempted resolutions

1. Gradient clipping – numerically stable but erroneous gradient updates.
2. Control the gradient information flow using gates (LSTMs, GRUs)
   - Sequential nature prevents parallelization and limits batch processing.
   - Unidirectional processing.
3. Bidirectional RNNs (BiLSTMs) process the information in forward and backward passes producing outputs which are finally concatenated to create the final output
   - Slower than LSTMs.
   - Lack of parallelism.
   - Difficulty in capturing long-term dependencies and lack of directional information.

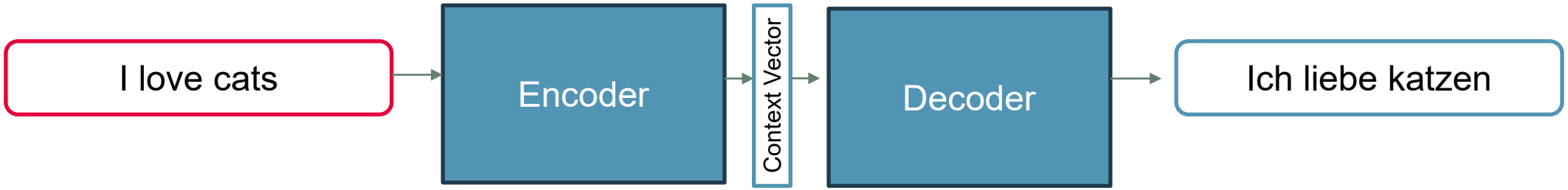
Sequence to Sequence Models



Sequence to Sequence Models



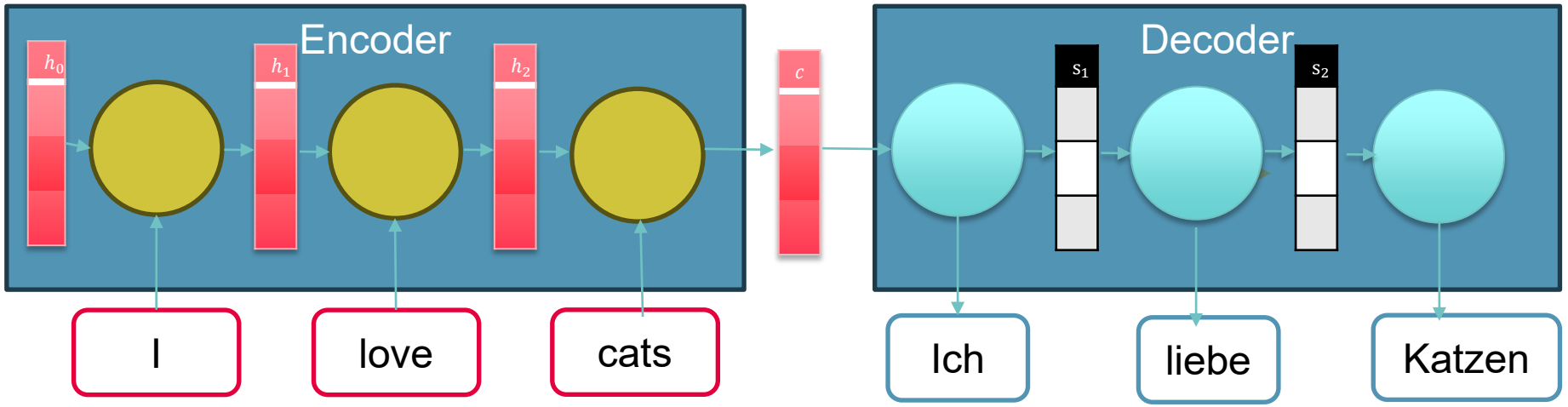
Sequence to Sequence Models



Separating encoding from decoding has the following advantages

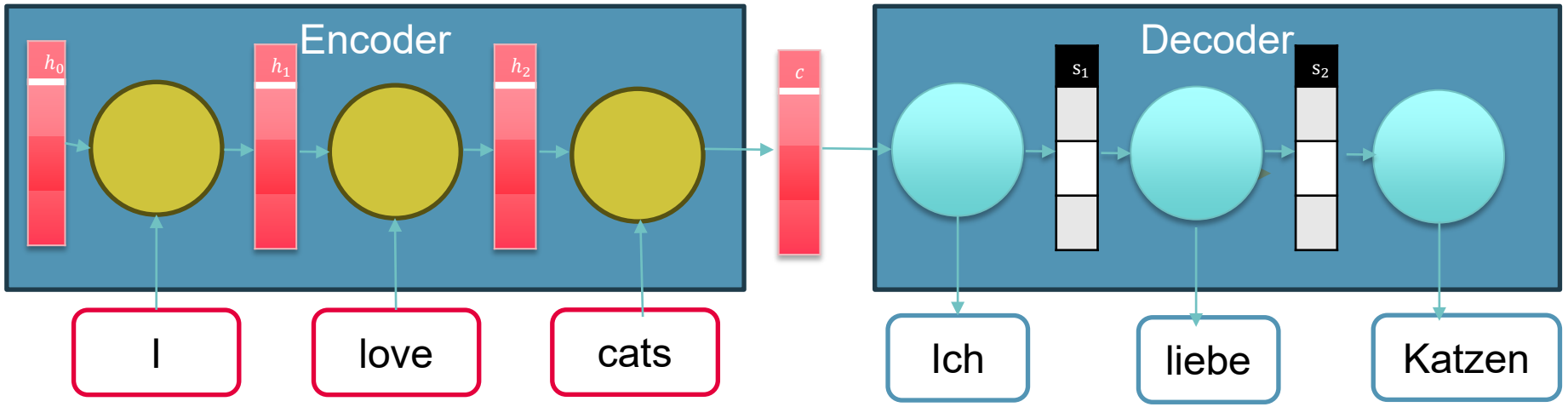
1. The decoder now has access to the full context before beginning decoding.
2. Parallel encoding and decoding is now possible for different sequences.

Encoder Decoder RNN Models



1. Encoder encodes the inputs step by step into a context vector
2. Context vector is processed by a decoder to produce outputs and decoder hidden states.

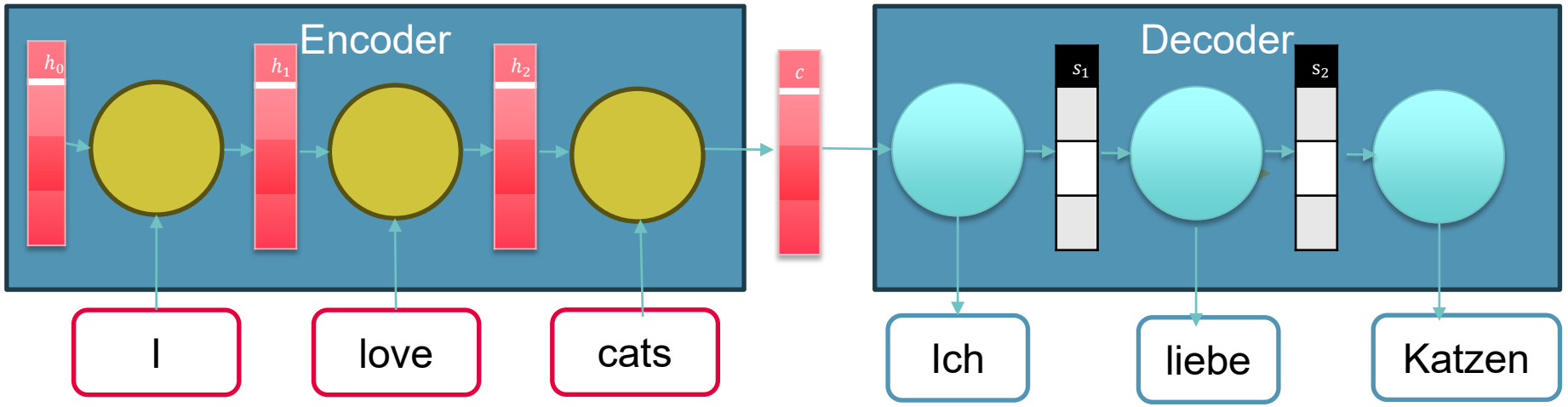
Encoder Decoder RNN Models



Context Vector, $c = q(\{h_1, \ldots, h_T\})$,

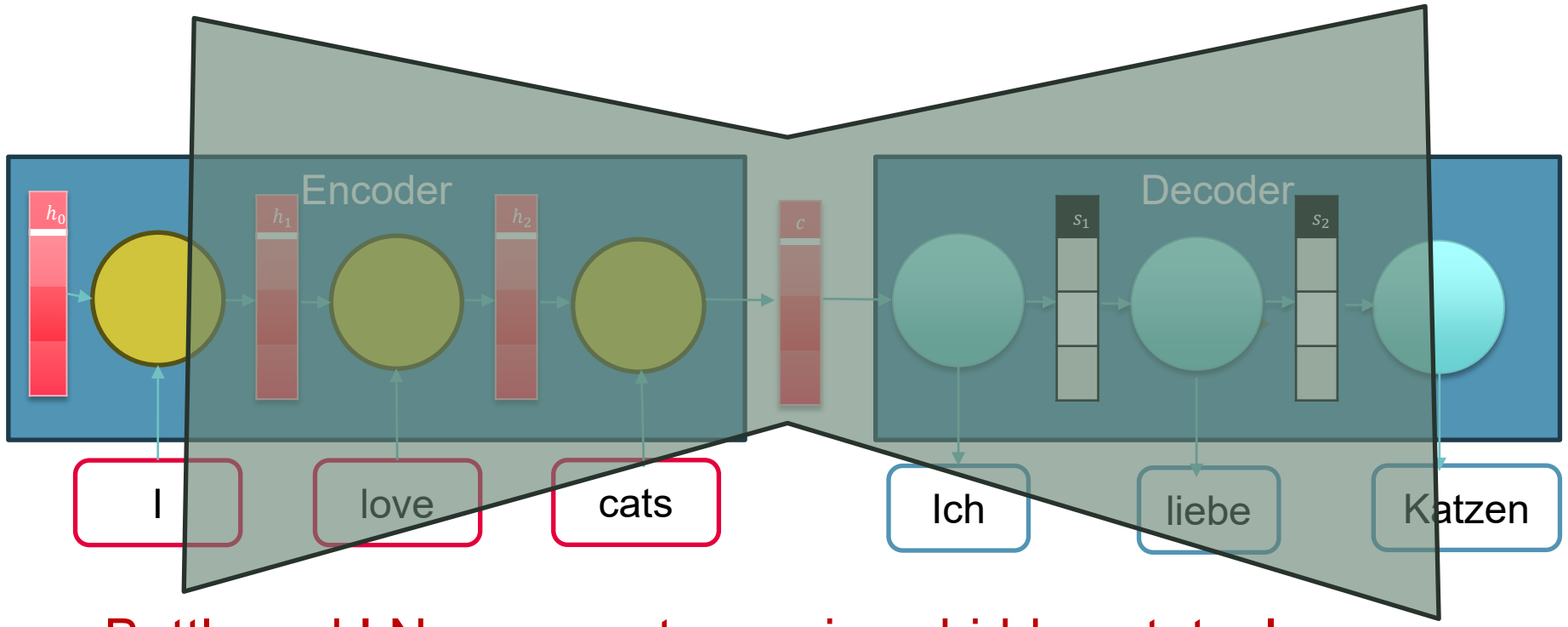
where q is a non-linear function

Encoder Decoder RNN Models



The problem?

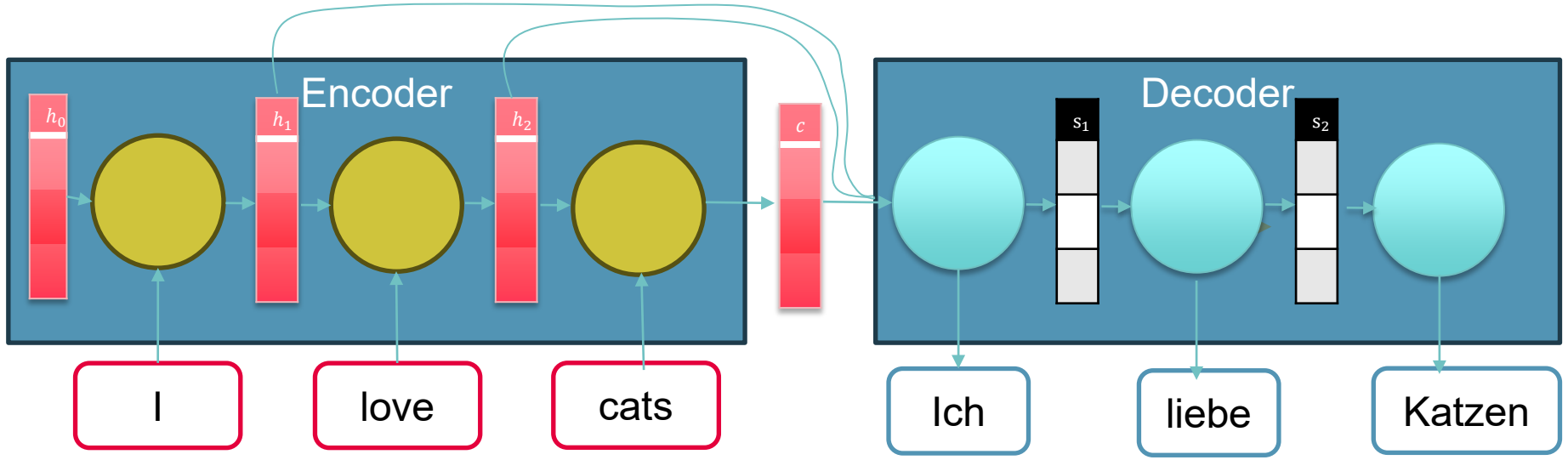
Encoder Decoder RNN Models



Bottleneck! No access to previous hidden states!

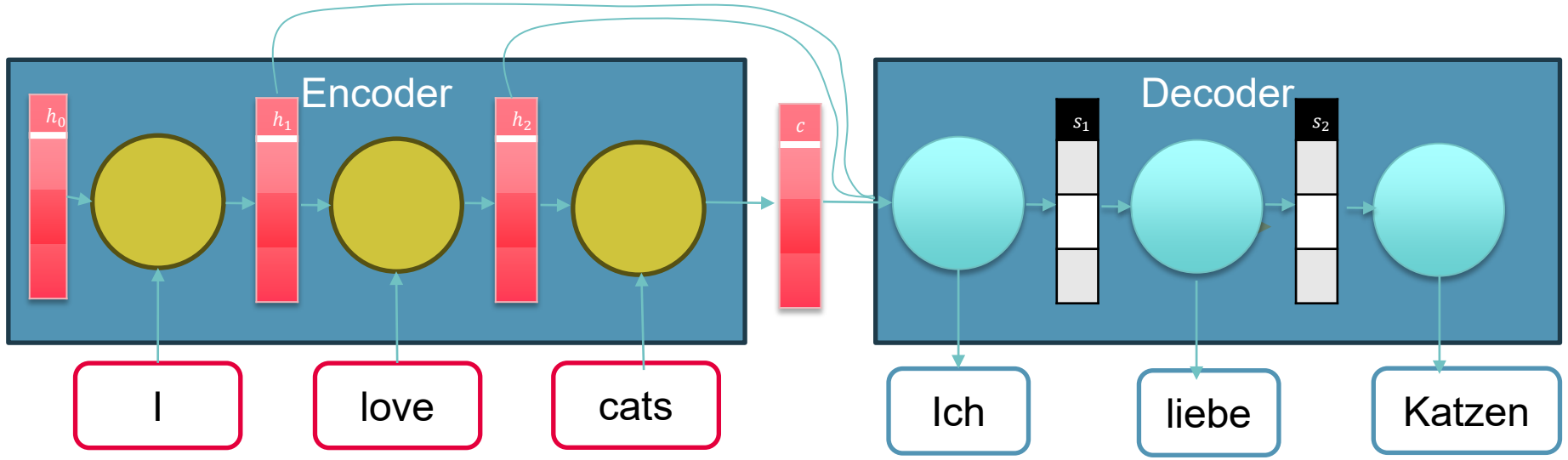
- Information has to be compressed into a fixed length vector leading to loss of information
- Especially information found early in the sequence tends to be “forgotten” after the entire sequence is processed.

Sequence to Sequence RNN Models



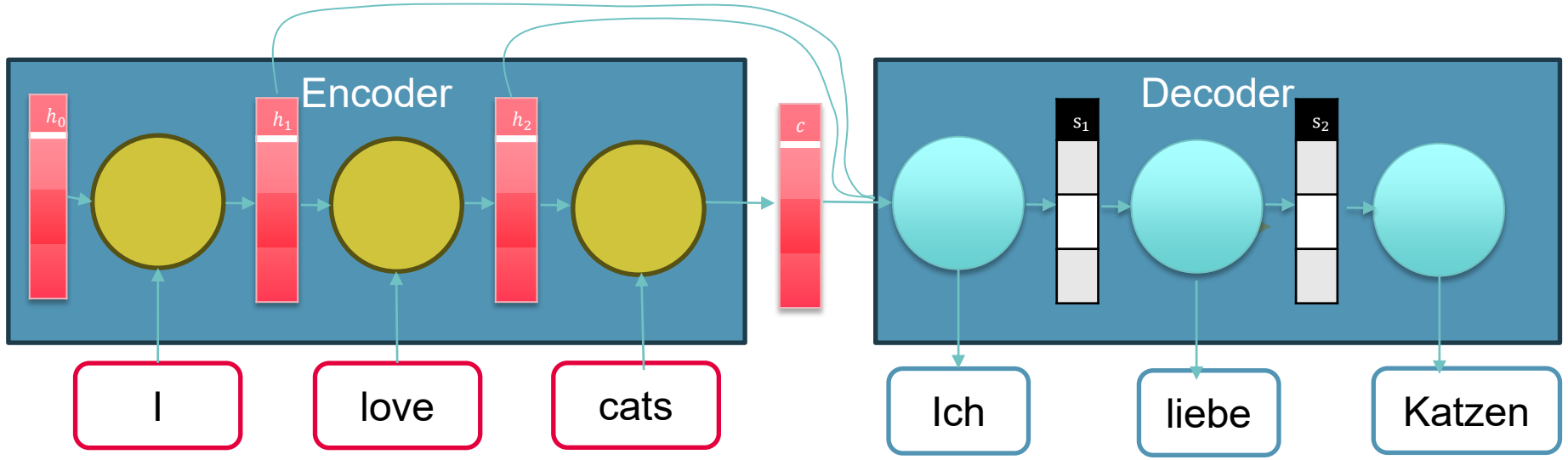
How about we give the decoder the access to all the encoder hidden states?

Encoder Decoder RNN Models



Higher complexity….where should we pay attention?

Sequence to Sequence RNN Models



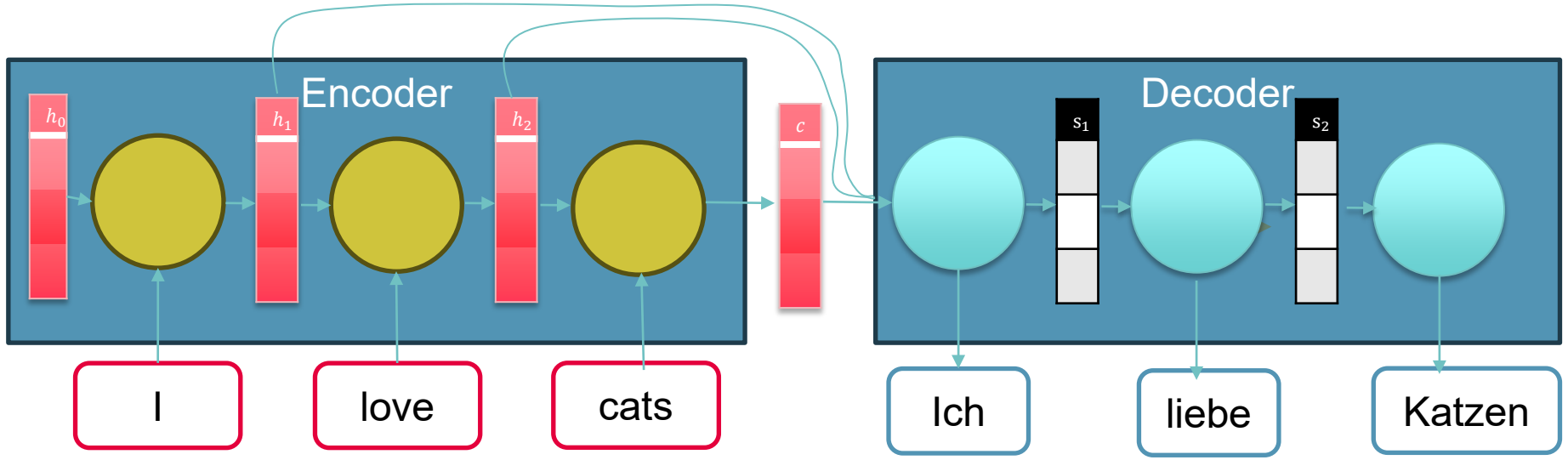
Instead of one context vector, we give a series of context vectors computed from the hidden states, where $\alpha_{t,i}$ is the corresponding weight (alignment score) for hidden state h_i

Examples of Alignment Scoring Functions

$$\alpha_{t,i} = \begin{cases} s_t^T . h_i, Luong\ 2015 \\ s_t^T . h_i / \sqrt{n} \\ s_t^T . W_a . h_i, Luong\ 2015 \\ v_a^T \tanh(W_a [s_t; h_i])\ Bahdanau\ 2014 \end{cases}$$

Adapted from Luong, M. T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025.

Sequence to Sequence RNN Models



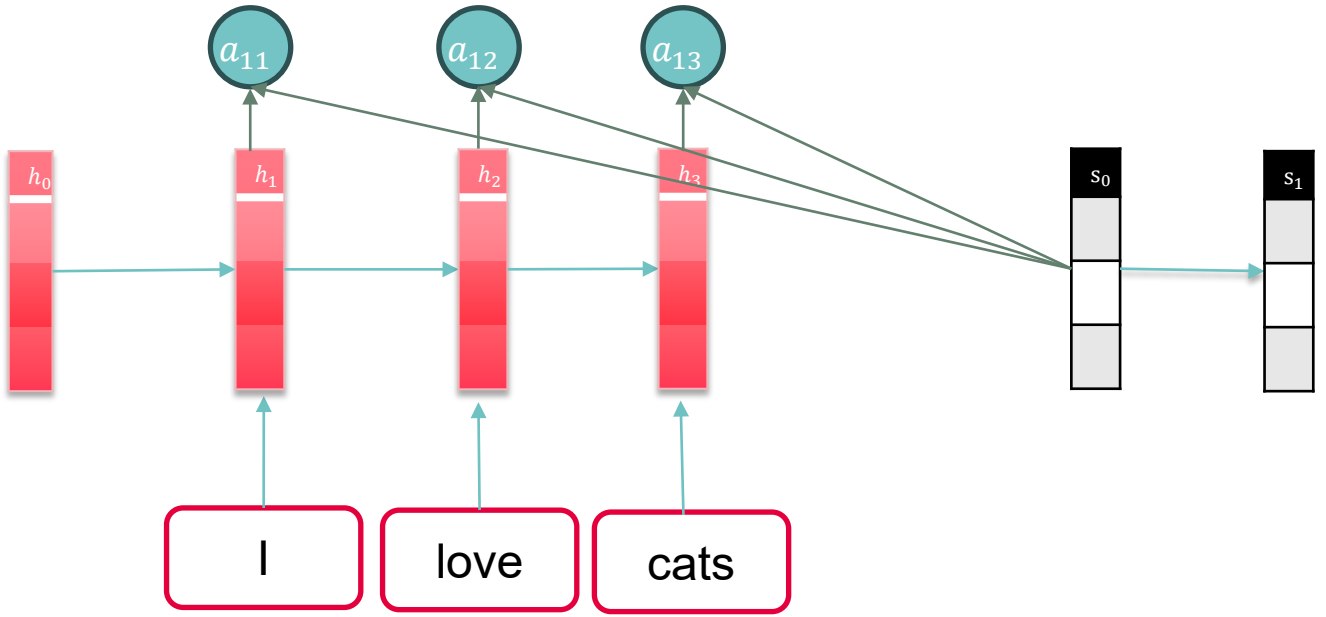
Decoder hidden state $s_t = f(s_{t-1}, y_{t-1}, c_t)$

Where, y_{t-1} represents the earlier predicted words

Encoder Decoder RNN with Attention

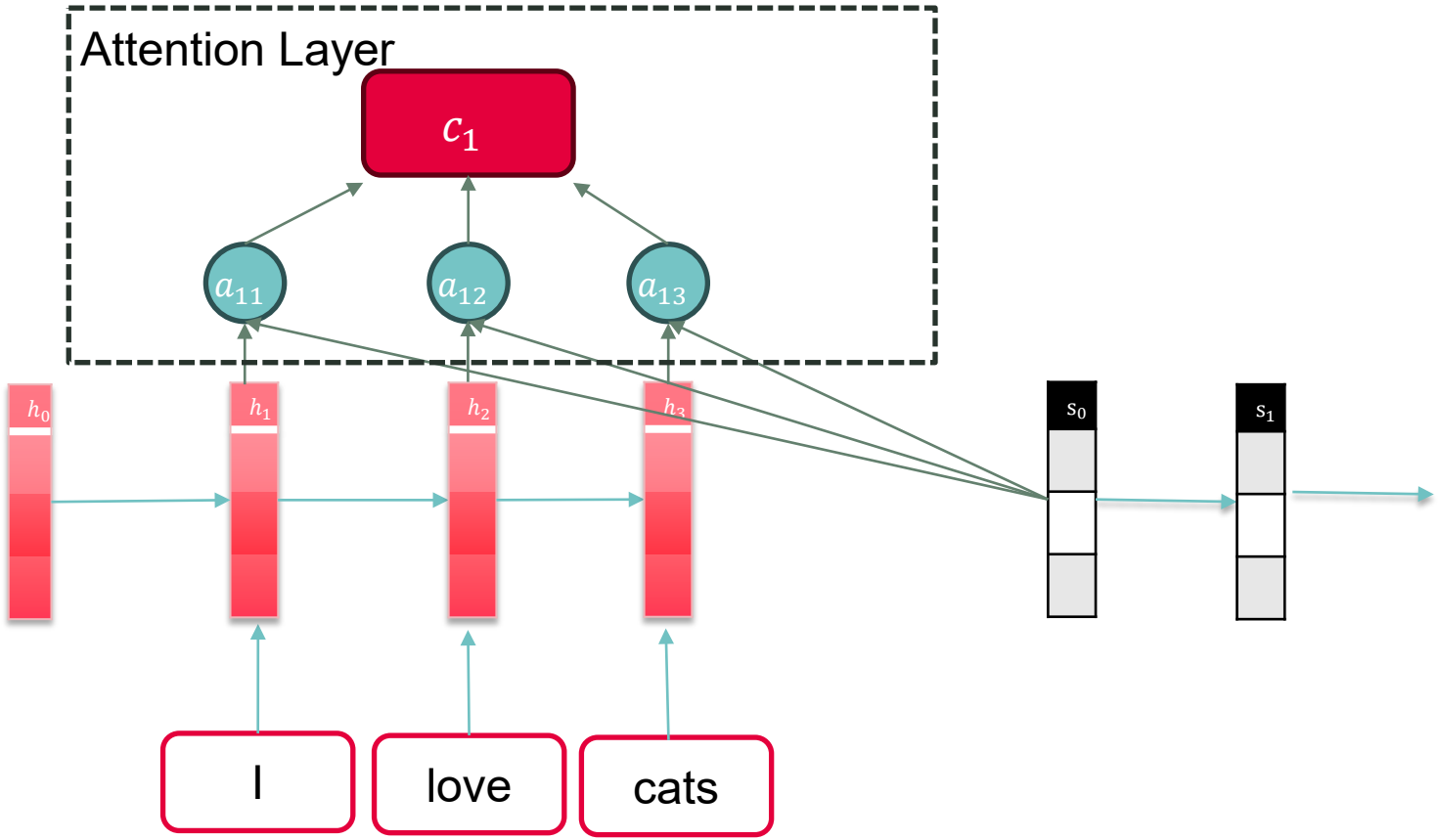
Attention weights $a_{t,i} = \frac{\exp(\alpha_{t,i})}{\sum_{k=1}^{T} \exp(\alpha_{t,k})}$ where,

$\alpha_{t,i}$ is the alignment score

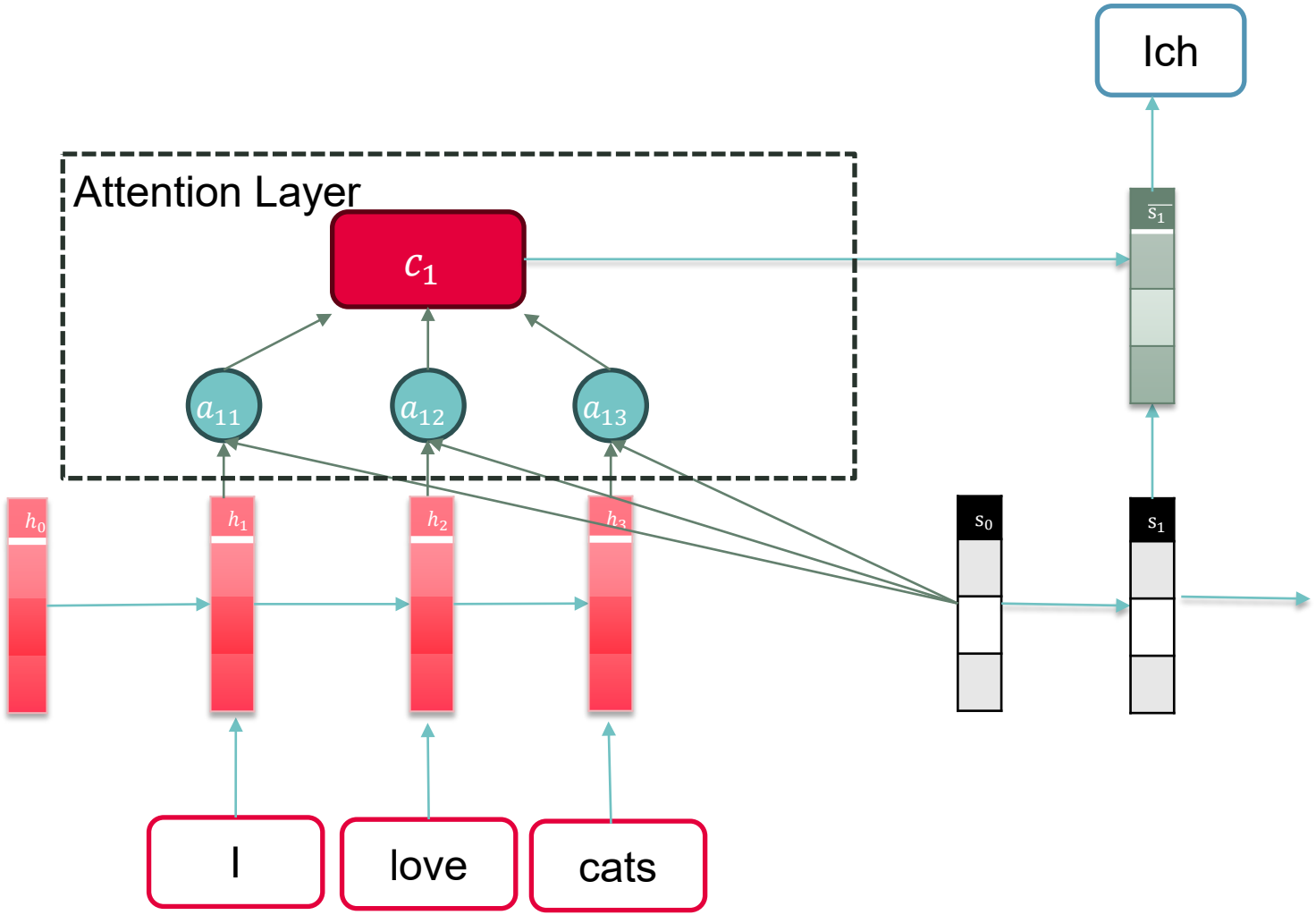


Encoder Decoder RNN with Attention

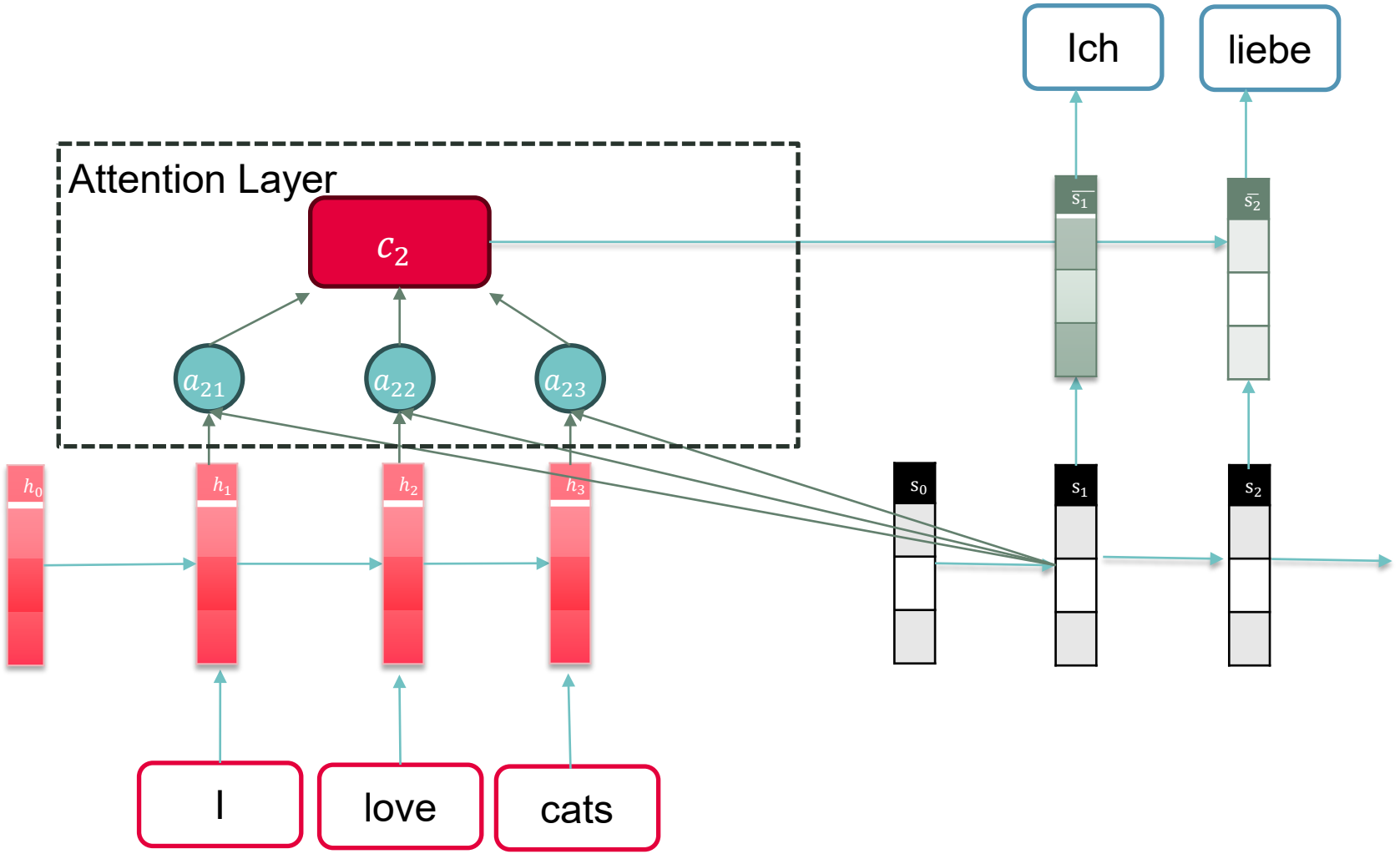
Context Vector $c_t = \sum_{i=1}^{T} a_{t,i} \cdot h_i$



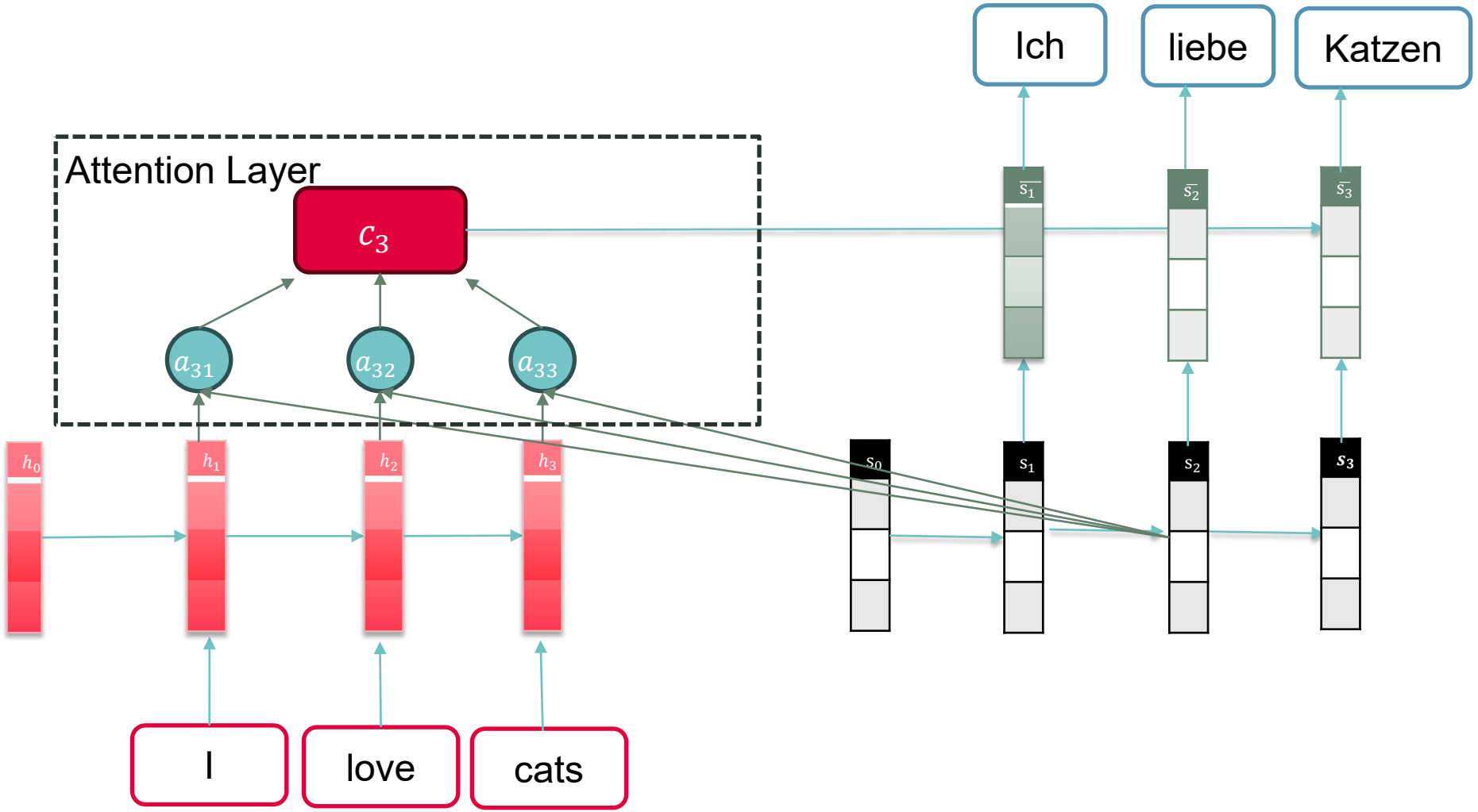
Encoder Decoder RNN with Attention



Encoder Decoder RNN with Attention



Encoder Decoder RNN with Attention

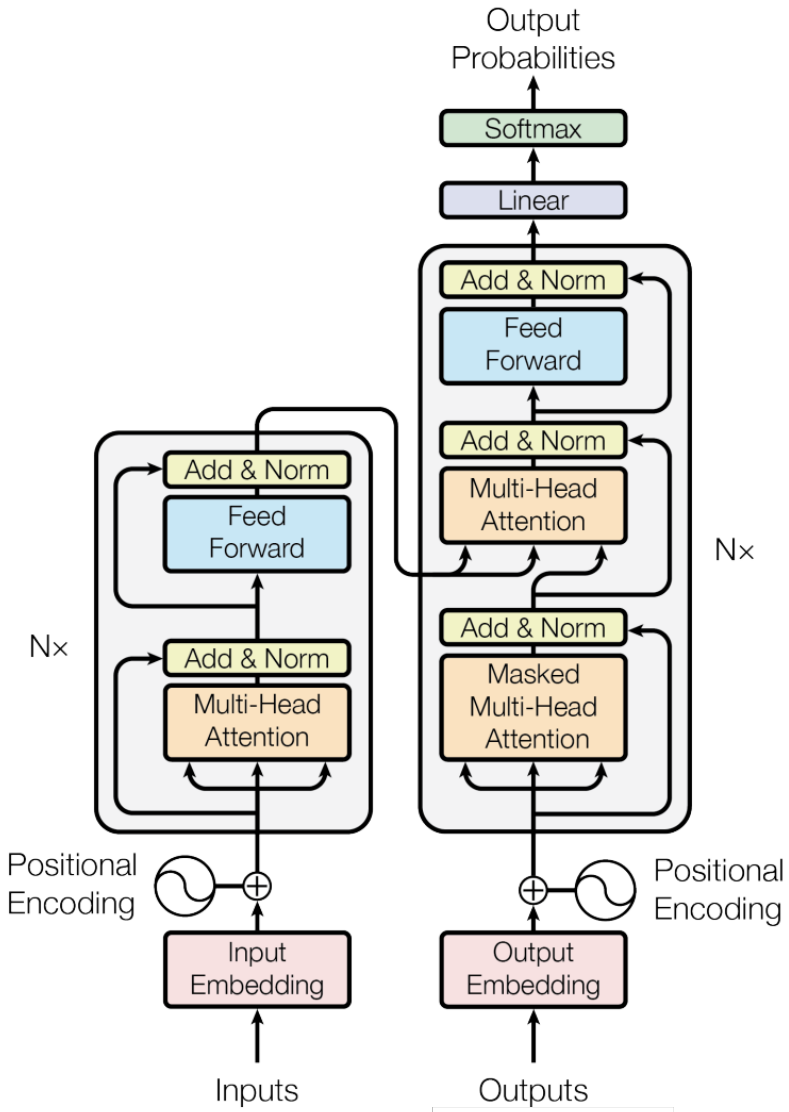


Takeaway

1. Encoder – Decoder architecture helps to produce variable length outputs (n->m).
2. An attention mechanism allows the modelling of dependencies without regard for the distance in either input or output sequences.
3. However, sequential nature of a RNN prevents parallelization within training examples. This becomes critical at longer sequence lengths as memory constraints limit batching across examples.

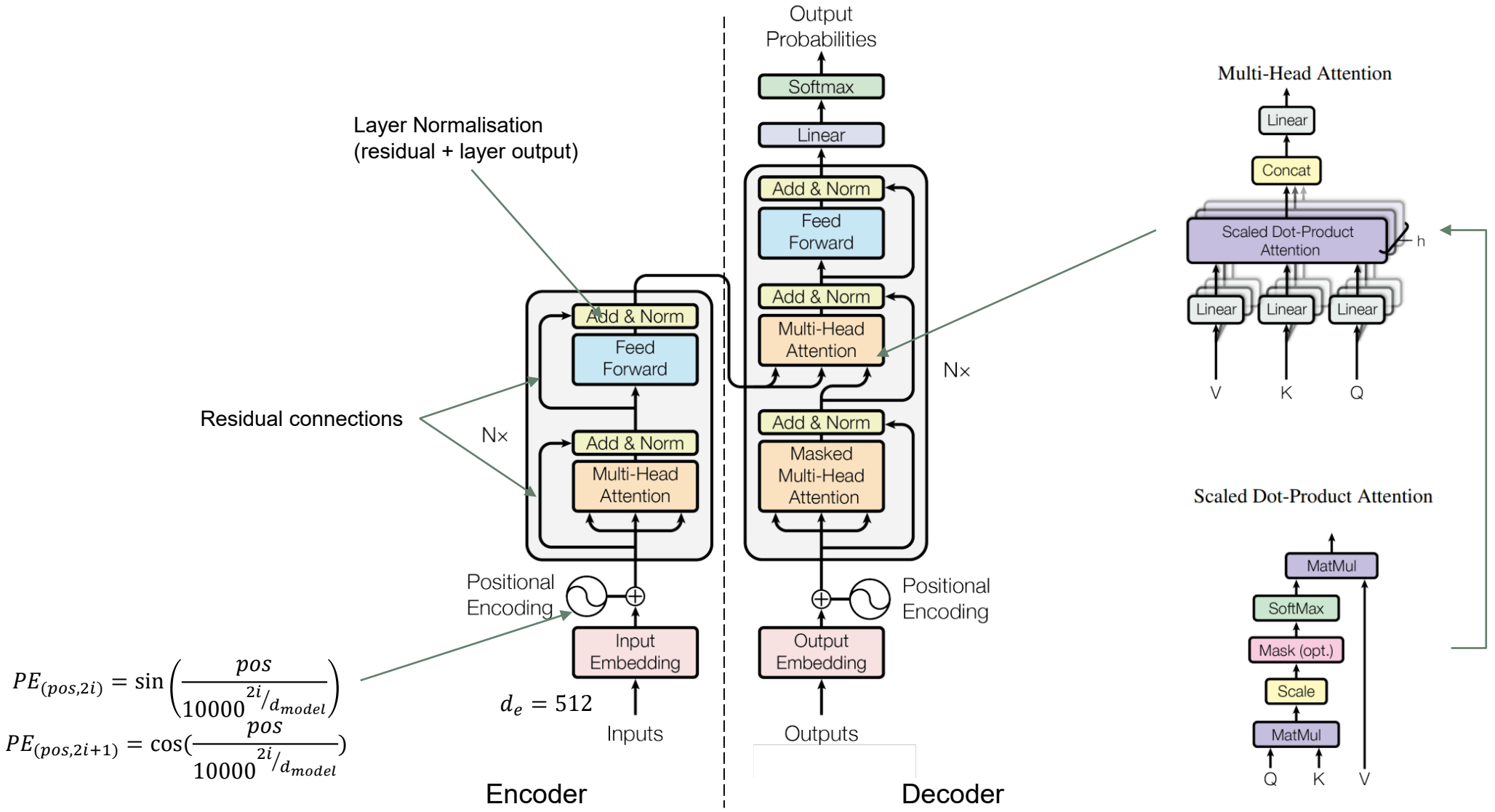
Transformer Model

How about a model that avoids recurrence and simply relies on simply attention mechanism?



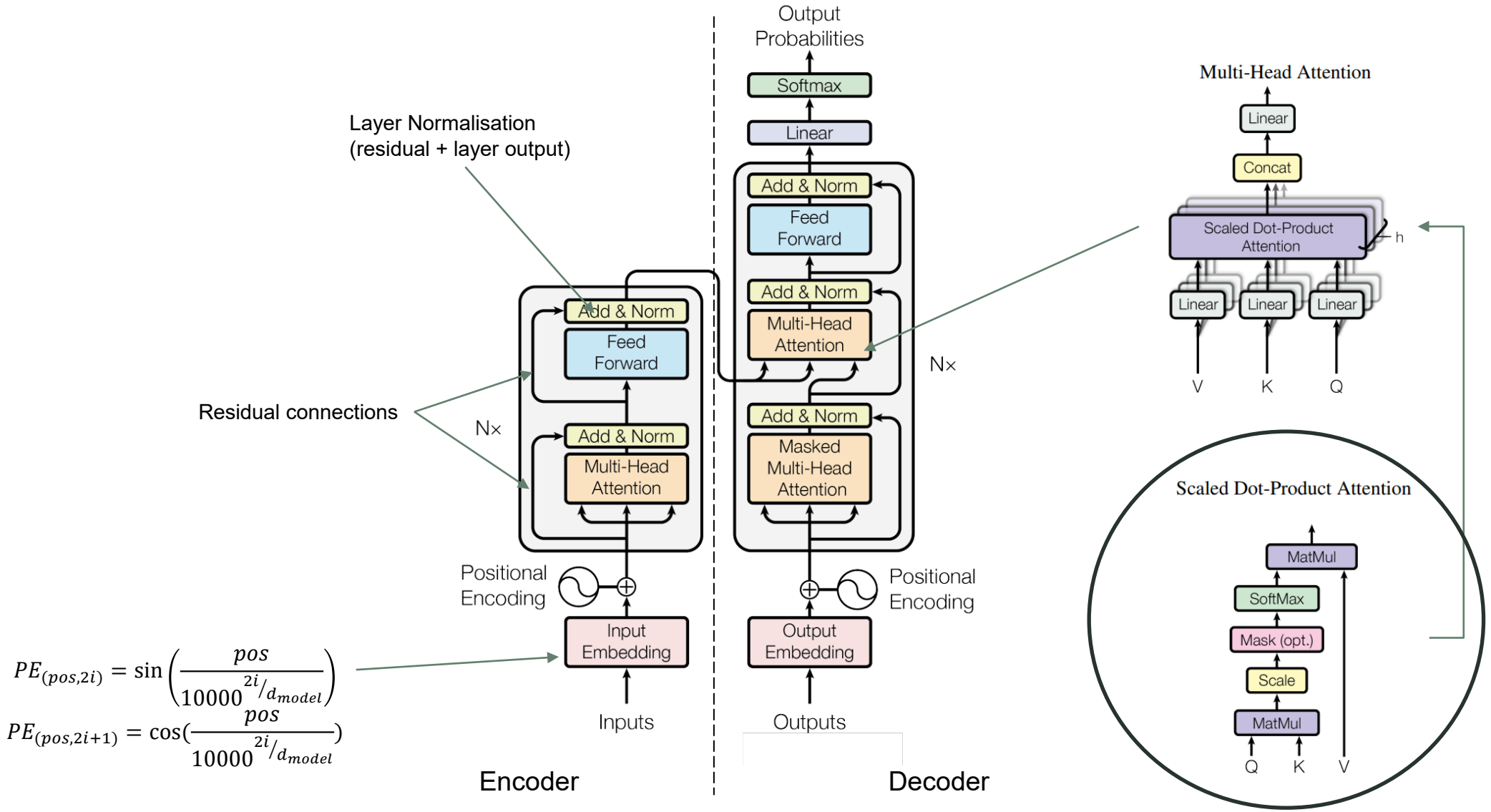
Picture Credits: Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

Transformer Model



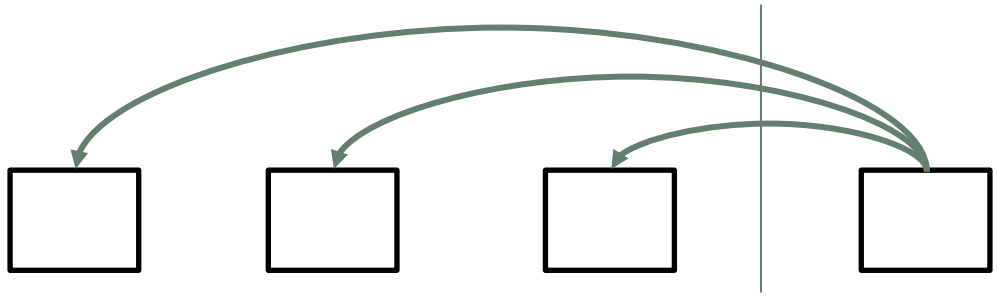
Picture Credits: Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

Transformer Model

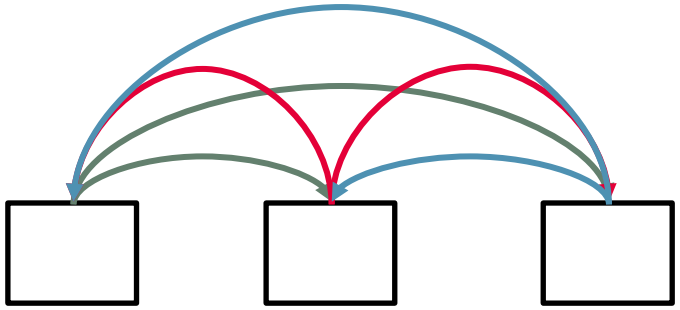


Picture Credits: Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

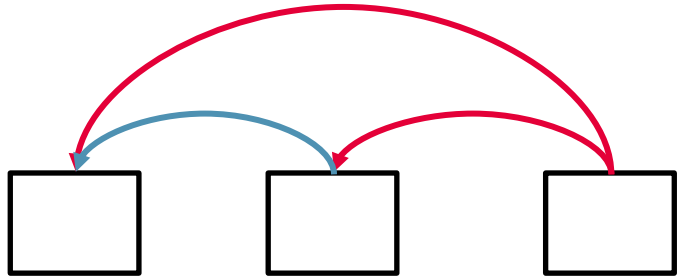
Three ways of attention



Encoder Decoder attention



Encoder Self-Attention

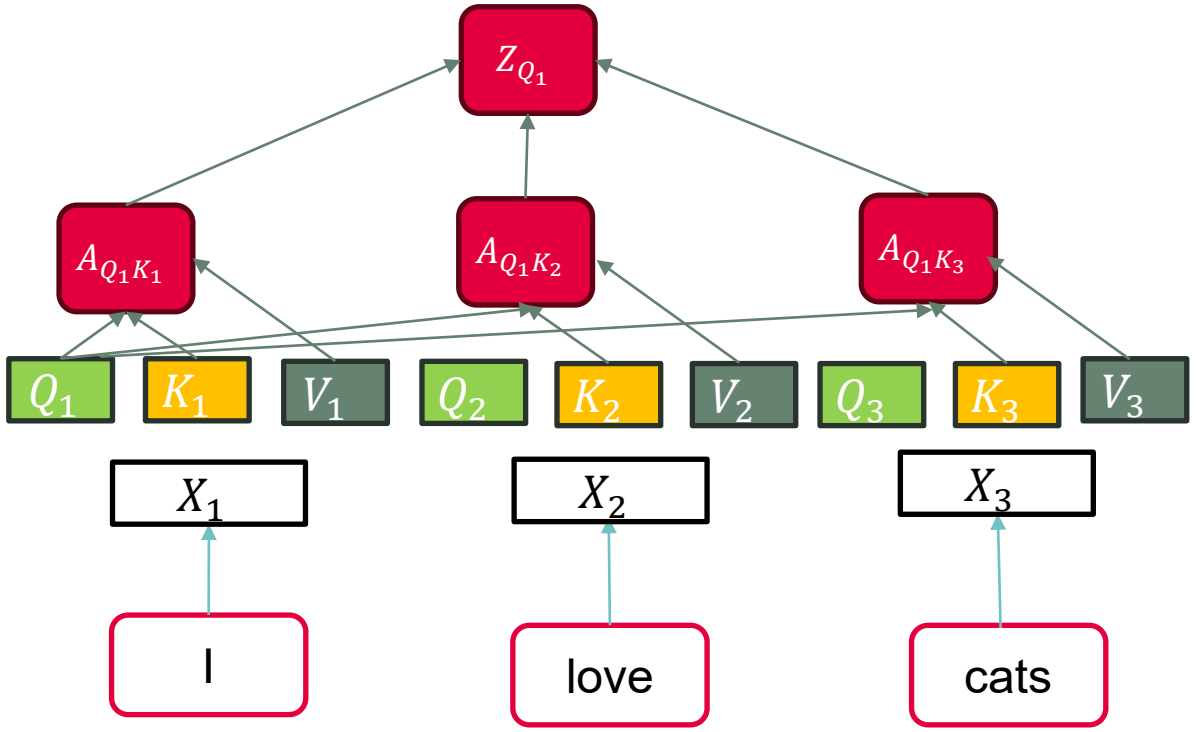


Masked Decoder Self-Attention

Self Attention

Scaled Dot-Product Attention score $Z_{Q_i} = \sum_{i=1}^{T} A_{ij}$

Where $A_{ij} = softmax\left(\frac{Q_i K_j^T}{\sqrt{d_k}}\right) V_j$

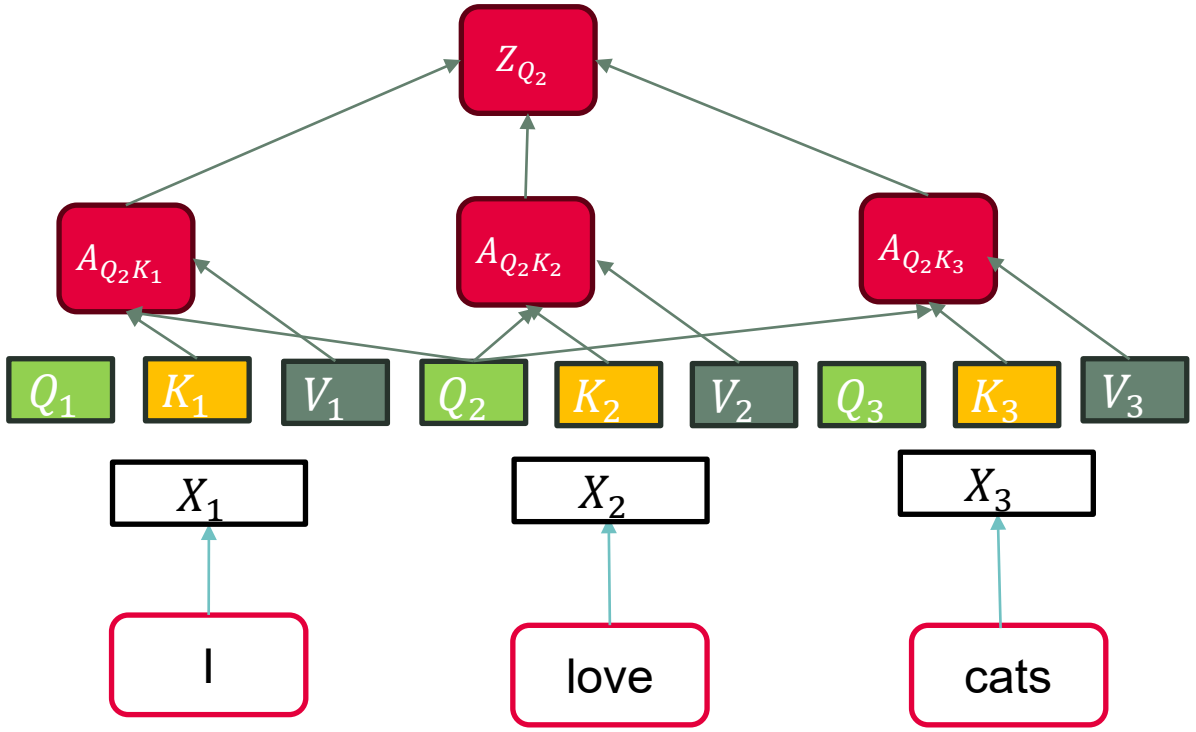


$where\ Q_i = W_q X_i\ (Query),$
$K_i = W_k X_i\ (Key),$
$V_i = W_v X_i\ (Value),$

Self Attention

Scaled Dot-Product Attention score $Z_{Q_i} = \sum_{i=1}^{T} A_{ij}$

Where $A_{ij} = softmax\left(\frac{Q_i K_j^T}{\sqrt{d_k}}\right) V_j$

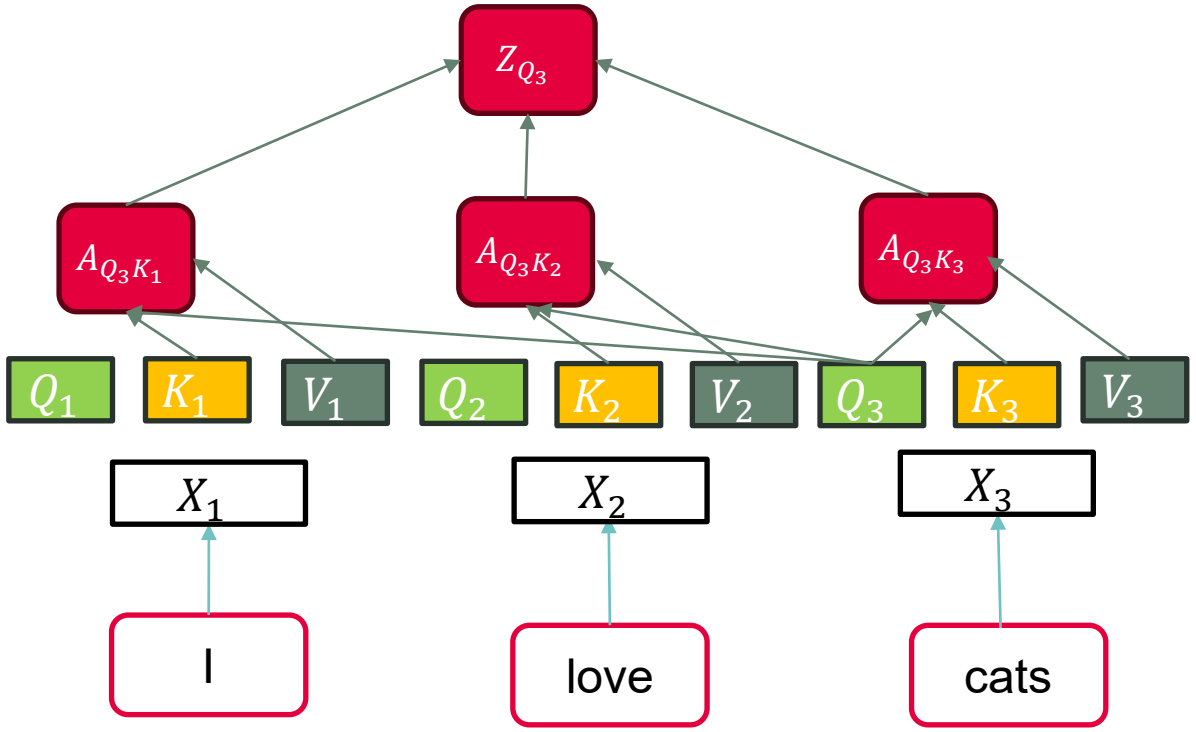


$where\ Q_i = W_q X_i\ (Query),$
$K_i = W_k X_i\ (Key),$
$V_i = W_v X_i\ (Value),$

Self Attention

Scaled Dot-Product Attention score $Z_{Q_i} = \sum_{i=1}^{T} A_{ij}$

Where $A_{ij} = softmax\left(\frac{Q_i K_j^T}{\sqrt{d_k}}\right) V_j$



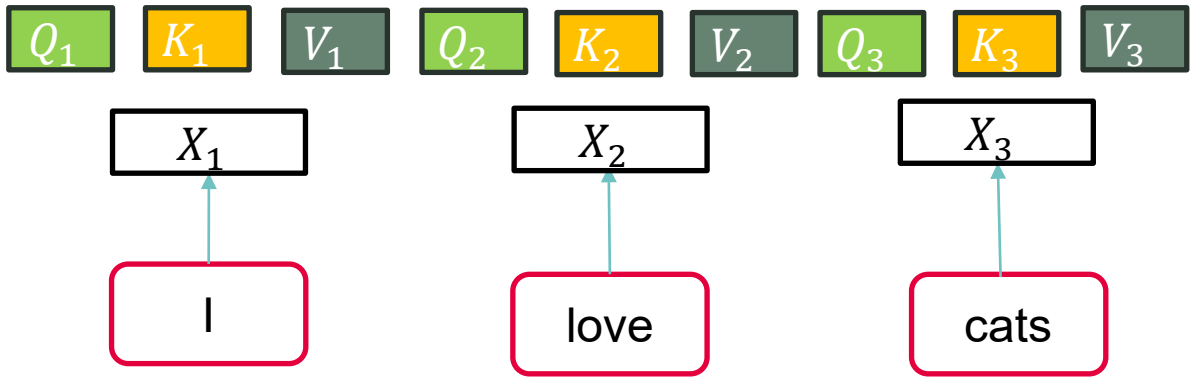
$$where\ Q_i = W_q X_i\ (Query),$$
$$K_i = W_k X_i\ (Key),$$
$$V_i = W_v X_i\ (Value),$$

Self Attention

Scaled Dot-Product Attention score $Z_{Q_i} = \sum_{i=1}^{T} A_{ij}$

Where $A_{ij} = softmax\left(\frac{Q_i K_j^T}{\sqrt{d_k}}\right) V_j$

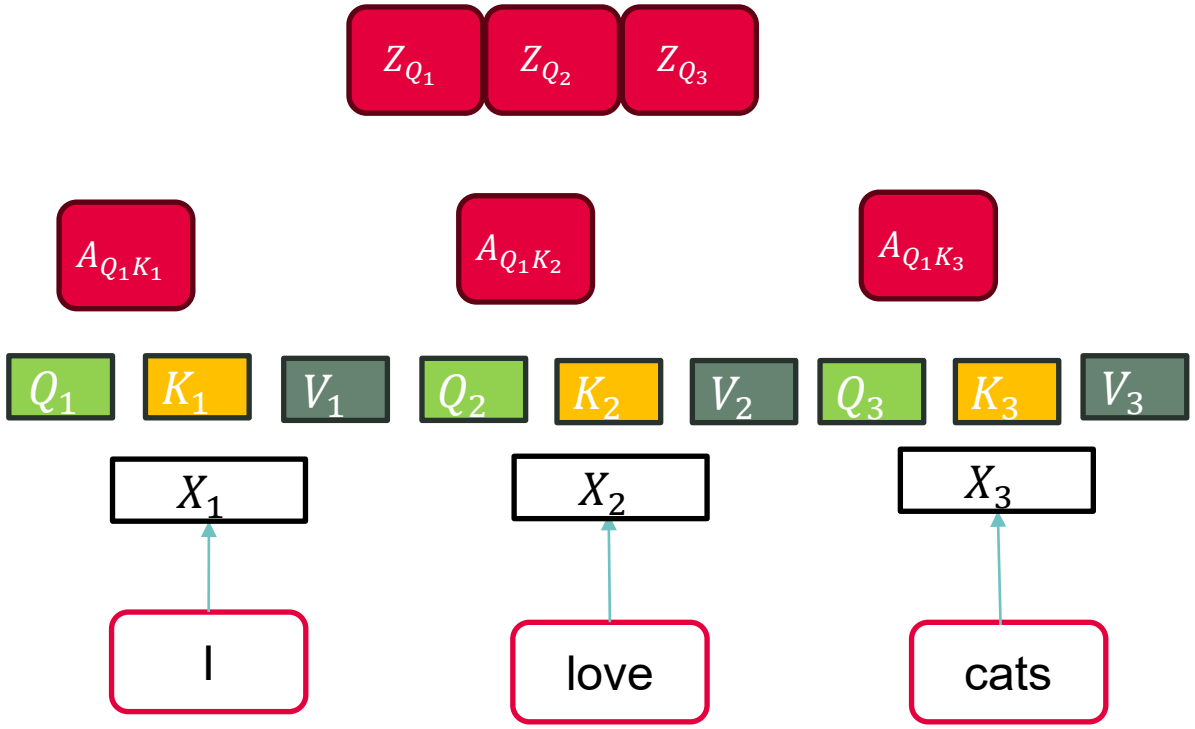
$where\ Q_i = W_q X_i\ (Query),$
$K_i = W_k X_i\ (Key),$
$V_i = W_v X_i\ (Value),$



Self Attention

Scaled Dot-Product Attention score $Z_{Q_i} = \sum_{i=1}^{T} A_{ij}$

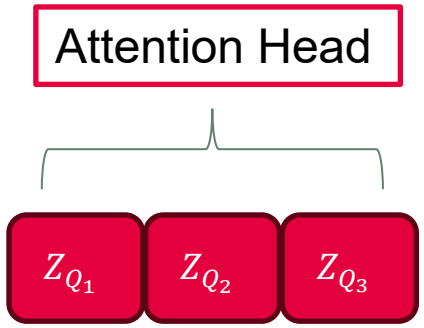
Where $A_{ij} = softmax\left(\frac{Q_i K_j^T}{\sqrt{d_k}}\right) V_j$



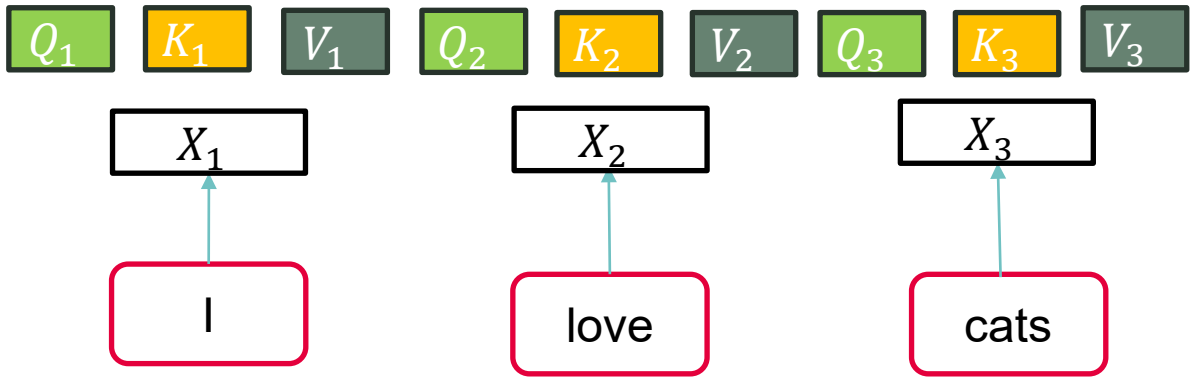
$where\ Q_i = W_q X_i\ (Query),$
$K_i = W_k X_i\ (Key),$
$V_i = W_v X_i\ (Value),$

Note these values can be calculated in parallel !

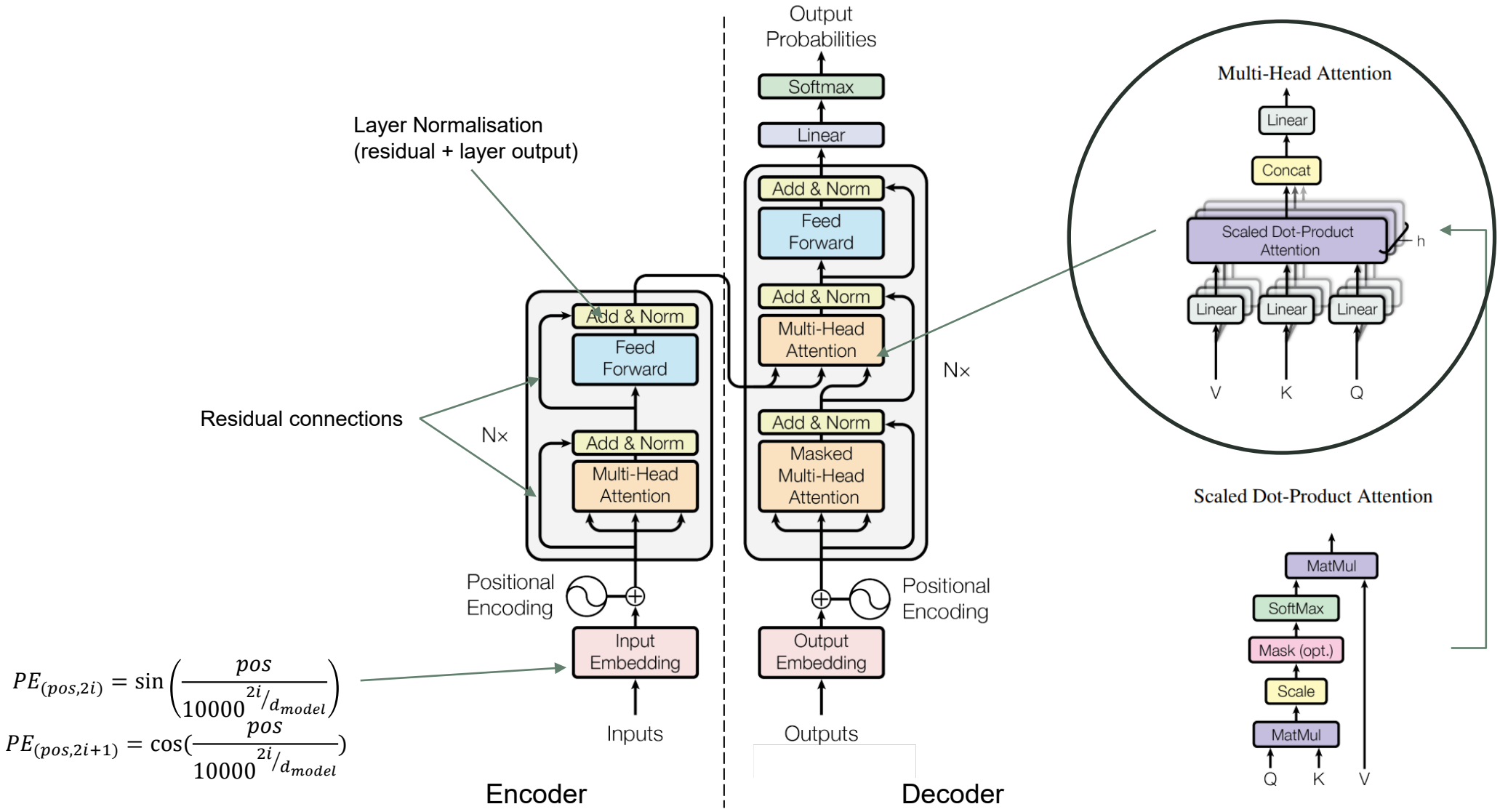
Self Attention



Attention Matrix $\mathrm{A}(\mathrm{Q}, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$



Transformer Model

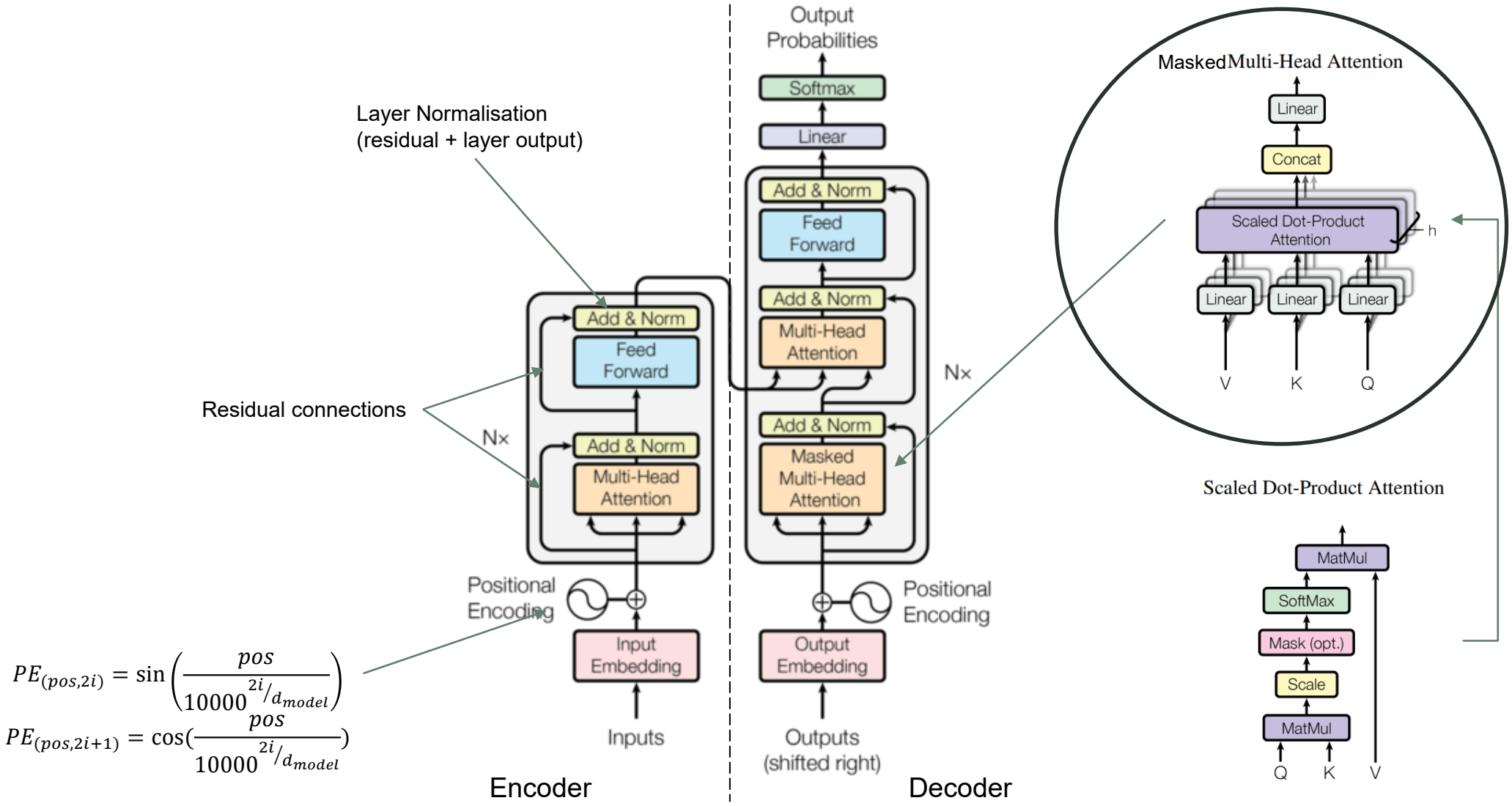


Picture Credits: Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

Multi-Head Self Attention

1. Apply self-attention multiple times in parallel and concatenate the results
   - 8 attention heads in the original transformer by Vaswani et al. 2017
2. For each self attention head, use different W_q, W_k, W_v
3. Helps to focus on different parts in the sequence.

Transformer Model



$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

Picture Credits: Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

Masked Multi Head Attention

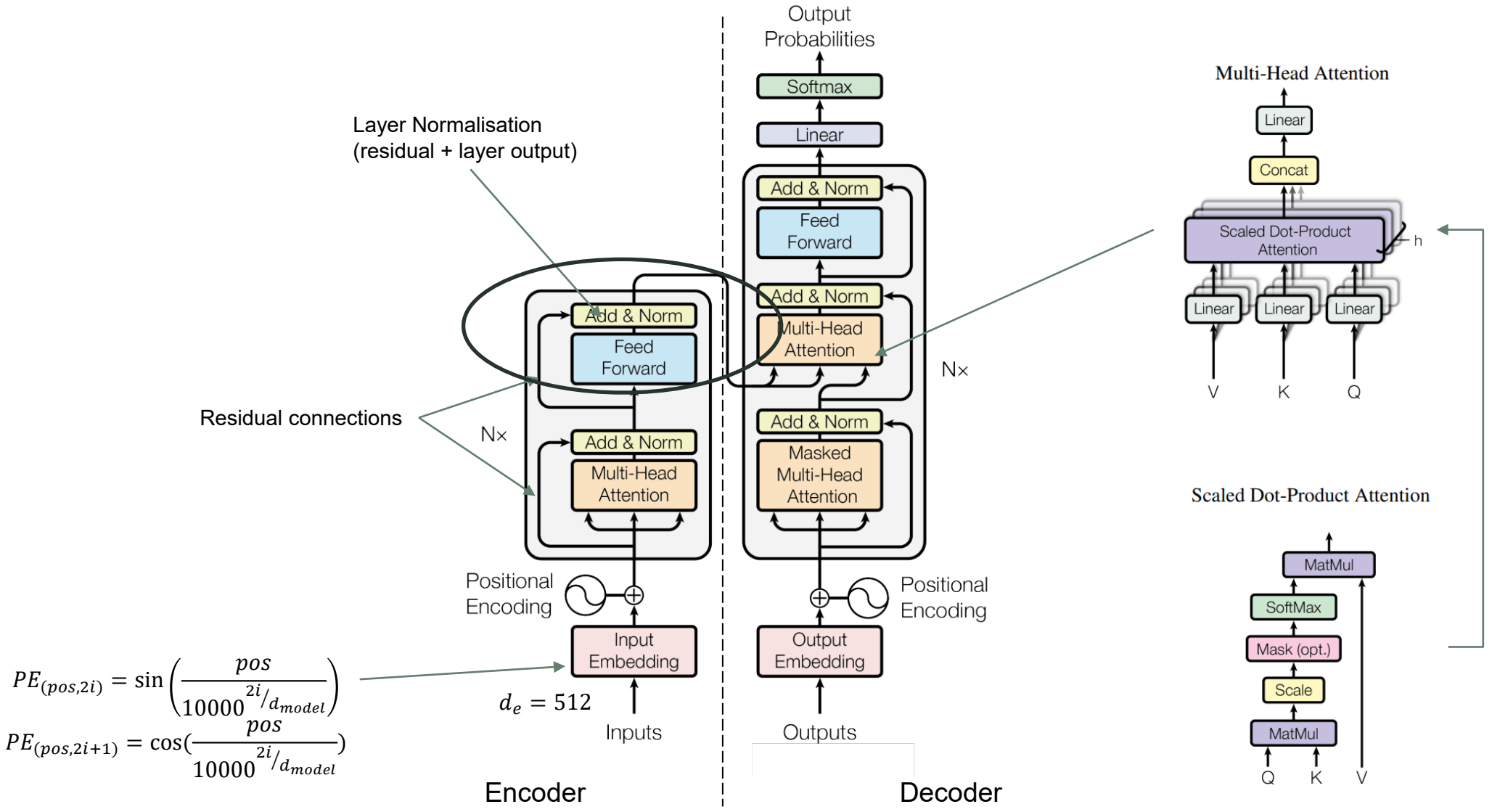
Masking of subsequent sequence elements (so that these values are not selected) and allow attending to positions up to and including the current position.

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

Masked Attention $\mathrm{A}(\mathrm{Q}, K, V) = softmax\left(\frac{QK^T + M}{\sqrt{d_k}}\right)V$

where M is a mask matrix

Transformer Model



Picture Credits: Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

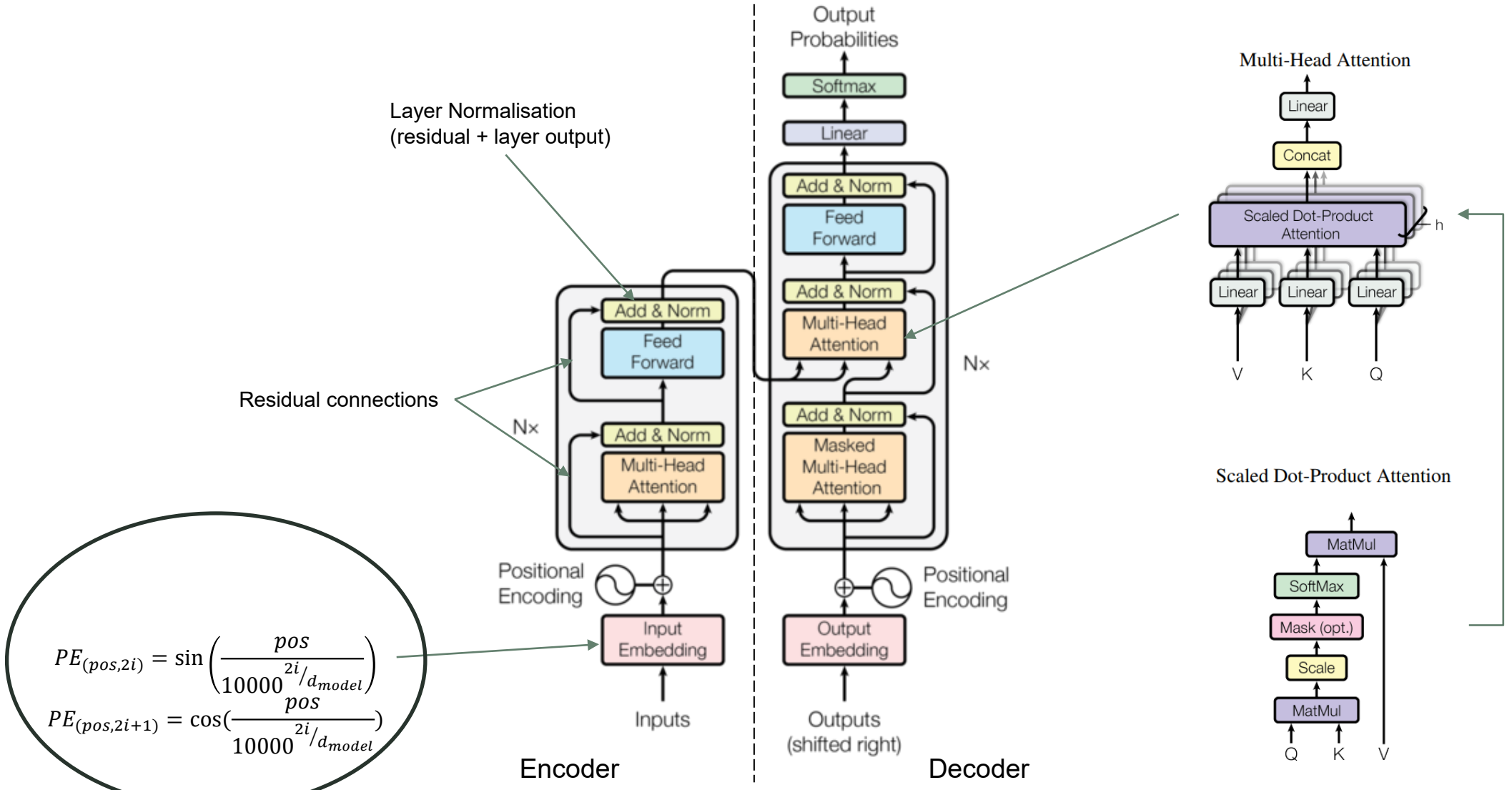
Point Wise Feed Forward Networks

1. In the absence of element-wise non-linearities, self-attention is simply performing a re-averaging of the value vector
2. Solution: A feedforward layer after each attention head

Residual Connections and Layer Normalisation

1. Difficult to train the parameters of a given layer because its input from the preceding layer keeps shifting
   - Solution: Normalization across layers (8 attention heads) to zero mean and standard deviation of one within each layer.
2. Residual connections provide another path for data to reach latter parts of the neural network by skipping some layers
   - Helps to preserve the ‘identity’ function, i.e., the original input

Transformer Model



Picture Credits: Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

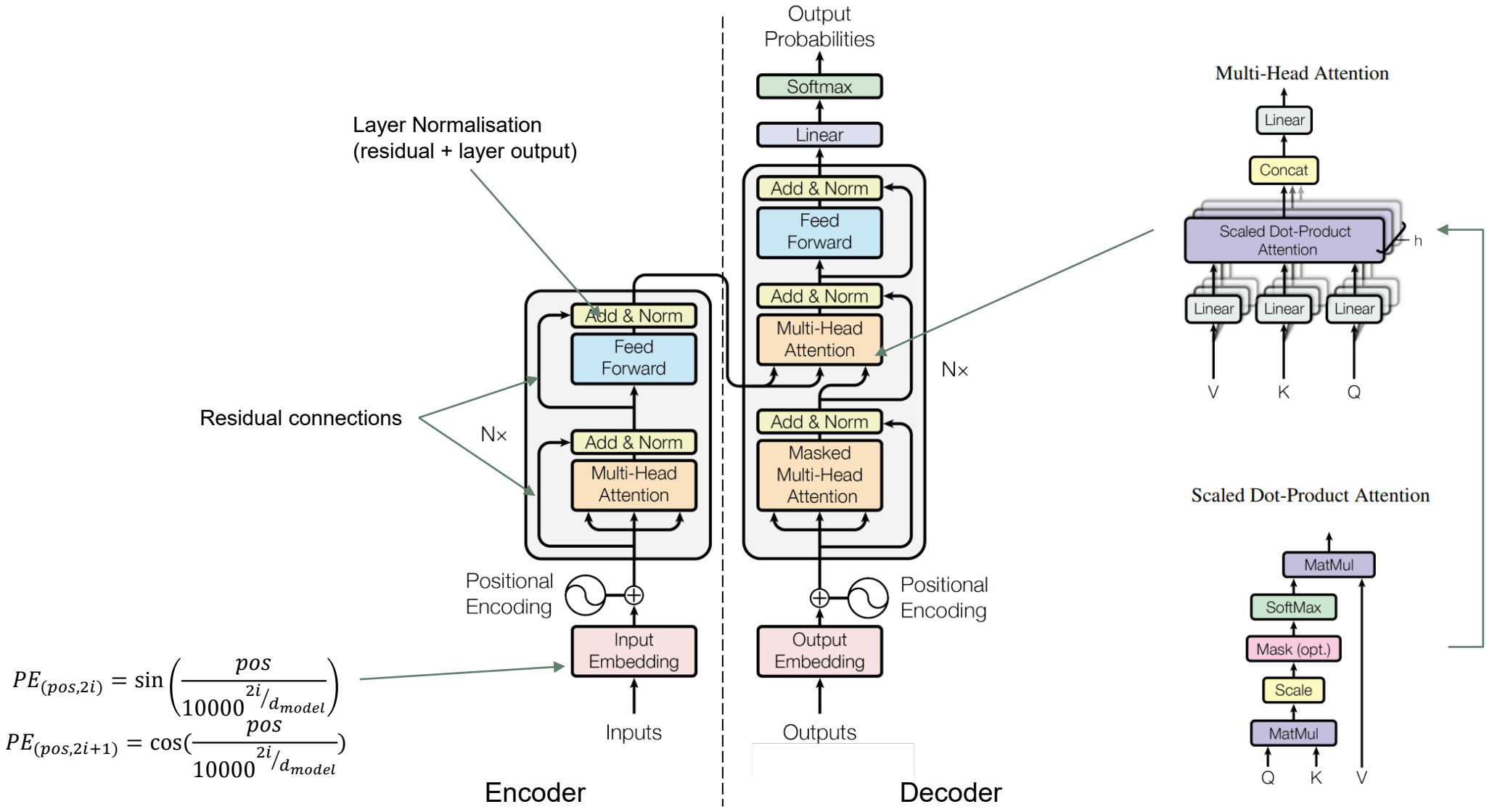
Positional Embeddings

Why do we need them?

1. Self-attention operation (Scaled dot-product and fully connected layer) are permutation invariant.
2. Encode the positional information such that
   - Large and small values can be managed.
   - Value is unique to a position and independent of sentence length.
   - Stays within a range (bounded) and varies non-linearly.
   - Sinusoidal positional encoding is a vector of small constants added to the embeddings

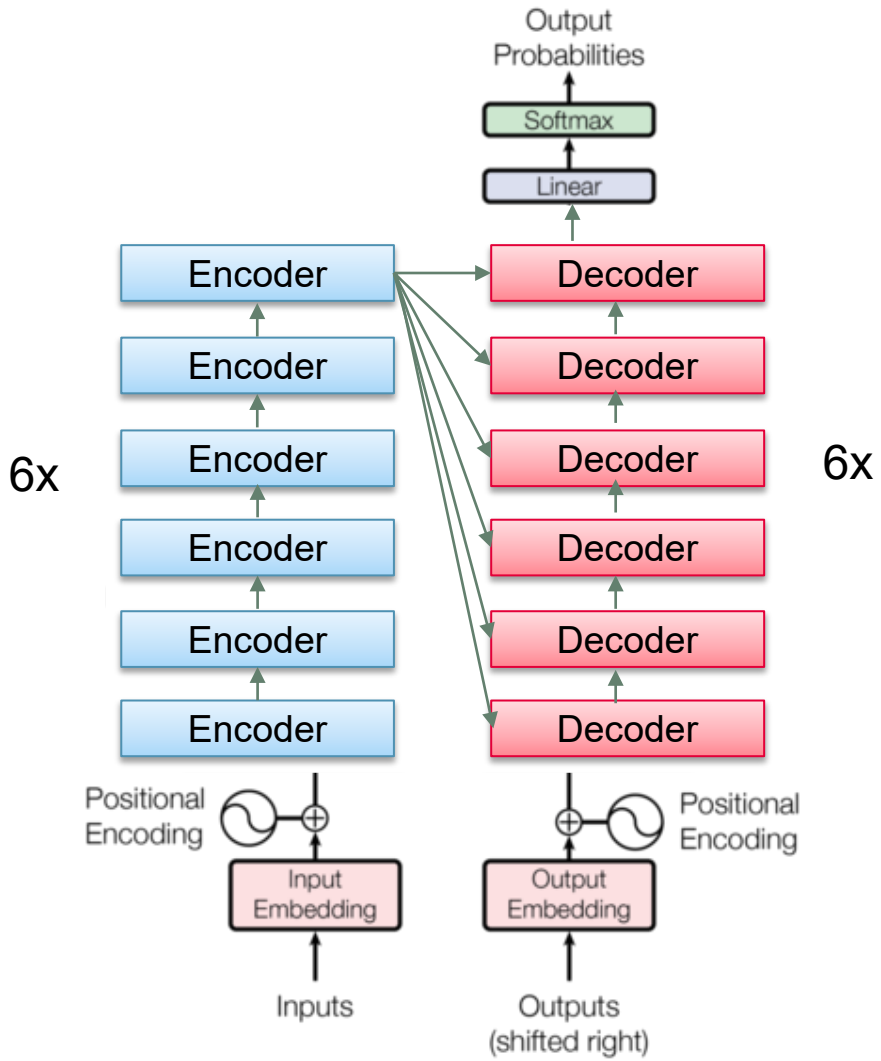
https://erdem.pl/2021/05/understanding-positional-encoding-in-transformers#positional-encoding-visualization

Transformer Model



Picture Credits: Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

Encode and Decoder Stacks



Takeaway

Why are Transformers so successful?

1. Self-attention mechanism helps to encode long-range dependencies.
2. Transformer models can be parallelized making them faster to process.
3. Transformer models are self-supervised and can learn useful representations even from unlabeled data.

Transformer Training Approach

1. Pre-training on large unlabeled datasets (Self-supervised learning)
2. Training for downstream-tasks on labeled data (supervised learning)

Limitations

Quadratic computation in self-attention (in recurrent models computation was linear)

- Suggested approaches include
  - Local attention instead of global.
  - Random interactions.

References

- The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time. (jalammar.github.io)
- 2021-CS109B/docs/lectures/lecture24/presentation/Lecture24_Attention.pdf at master · Harvard-IACS/2021-CS109B (github.com)
- The Transformer Family | Lil'Log (lilianweng.github.io)
- Introduction to Deep Learning (sebastianraschka.com)