Attention and Transformer Architectures

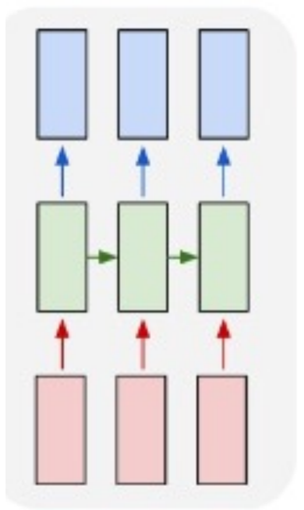
David I. Inouye

PURDUE UNIVERSITY | Elmore Family School of Electrical and Computer Engineering

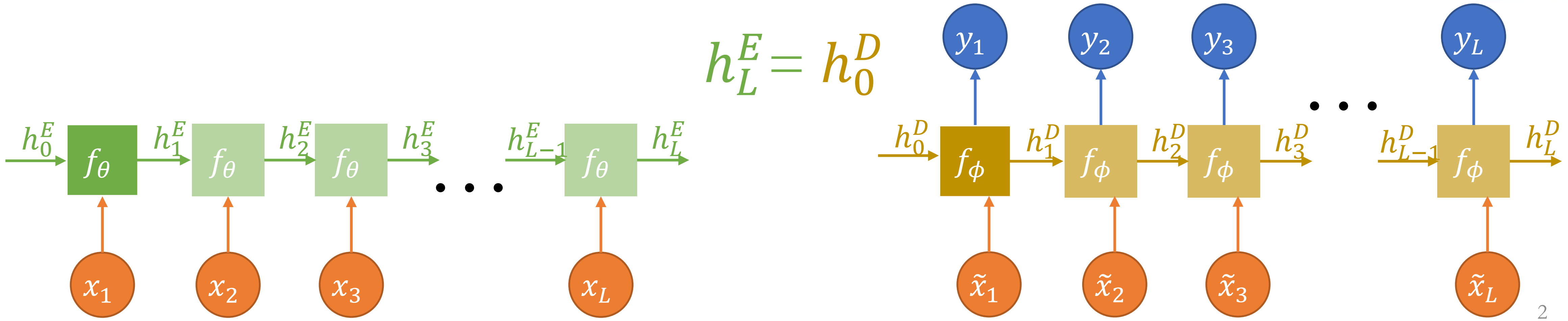
Standard RNNs struggle for **<u>sequence-to-sequence</u>** tasks because of limited hidden state capacity

- Example: Translation between French and English
- Could we use a one-to-one input/output RNN?
  - Problem: Input sequence could have different length.
  - Problem: The order of words is not the same in French and English.
- More common to use autoencoder structure with 2 RNNs.
  - Problem: Challenging to encode entire sentence in hidden state.

many to many

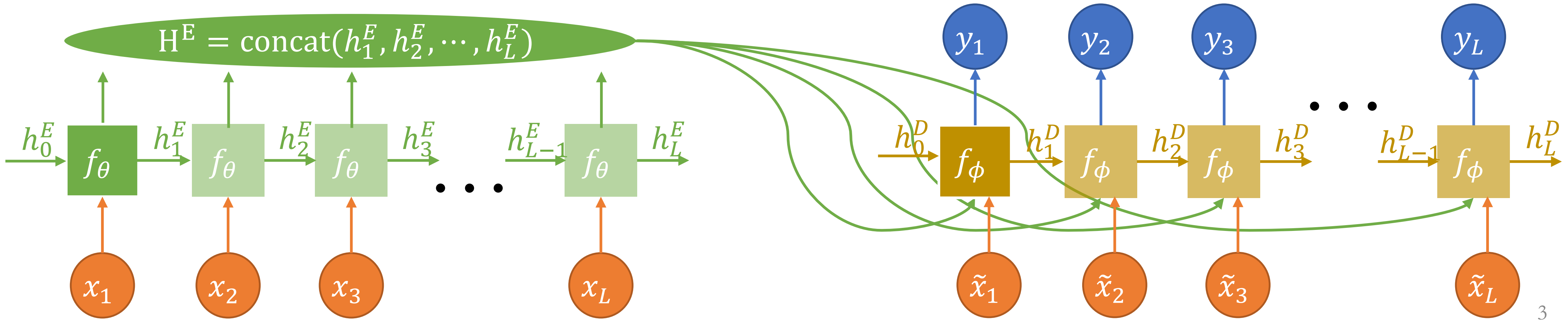


$h_L^E = h_0^D$



Attention is a model architecture that enables the decoder to efficiently use all encoder outputs

- Attention overcomes some of the challenges of RNN-based translation
- Attention allows long-range dependencies and avoids a completely sequential view of the input and output

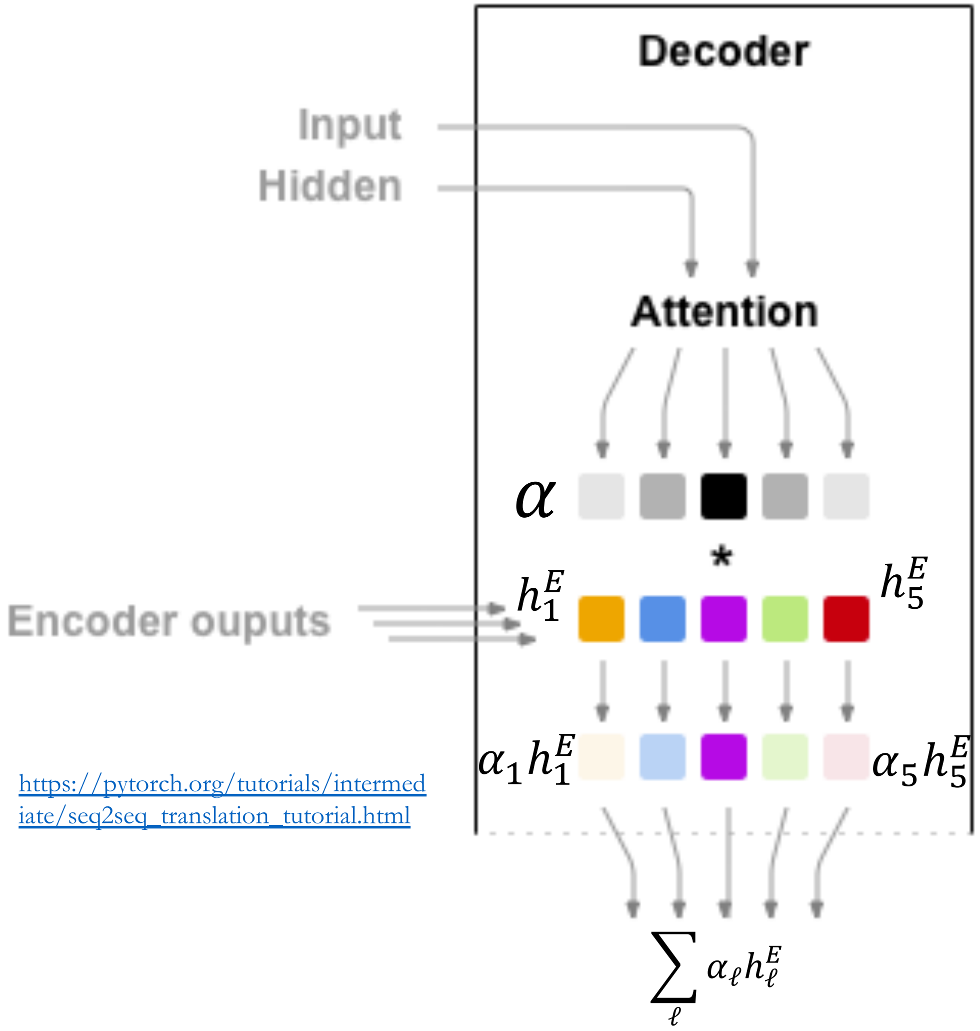


Based on the input and hidden state, **attention** determines the weights for adding the encoder outputs

- Informally, the attention mechanism determines which encoder outputs the network should "focus" on (α for attention)

- The weights are normalized to sum to one via softmax, $\sum_\ell \alpha_\ell = 1$
  - Akin to the intuitive idea of "limited attention"
  - In practice, I conjecture this normalization is critical for its training stability.

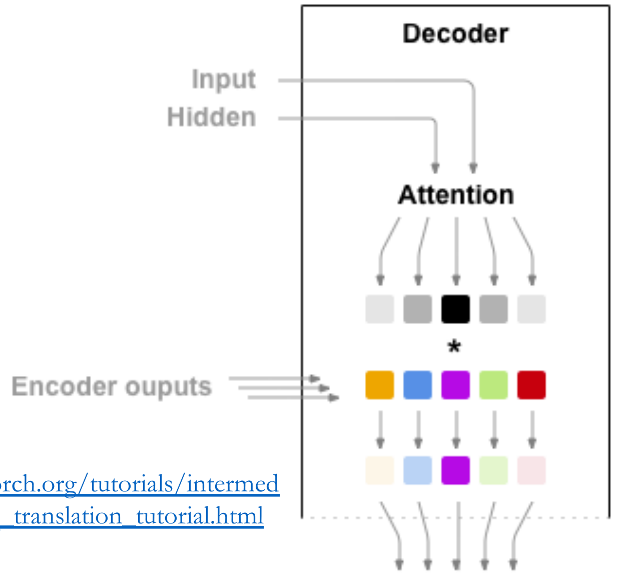
- The output of attention is a weighted sum of the inputs based on the computed attention weights $\text{out} = \sum_\ell \alpha_\ell h_\ell^E$



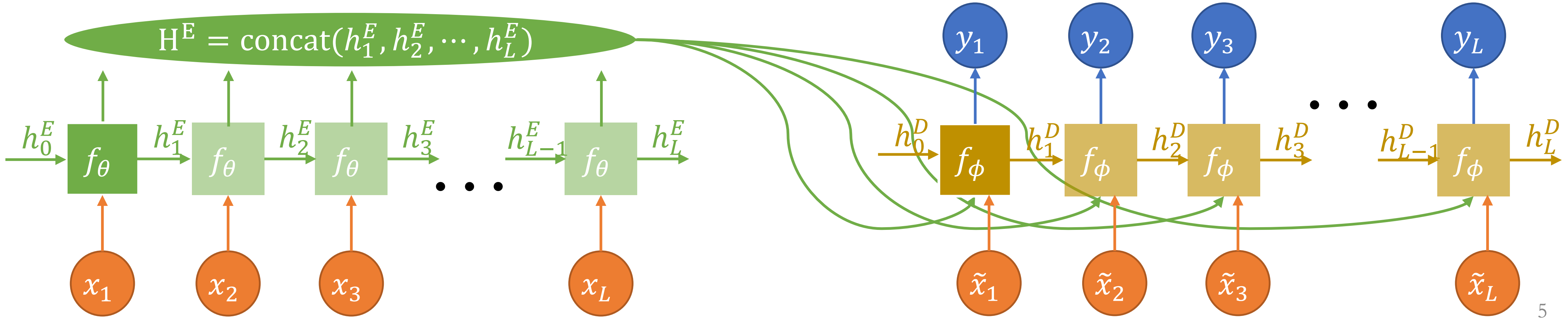
https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html

Attention can be represented by standard linear layers and softmax functions

- $H^E := [h_1^E, h_2^E, \cdots, h_L^E] \in \mathbb{R}^{k \times L_{\max}}$ (hidden state from encoder)
- $h'_{t-1} = [h_{t-1}^D, \tilde{x}_t]$ (concatenate)
- $\alpha_t = \sigma(W_a h'_{t-1} + b_a) \in \mathbb{R}^{L_{\max}}$ (softmax for attention weights)
- $c_t = H^E \alpha_t$ (take weighted average of encoder outputs)
- $z_t = \mathrm{ReLU}(W_c[\tilde{x}_t, c_t] + b_c)$ (incorporate input and context)
- $y_t, h_t^D = f_\theta(z_t, h_{t-1}^D)$ (standard RNN)

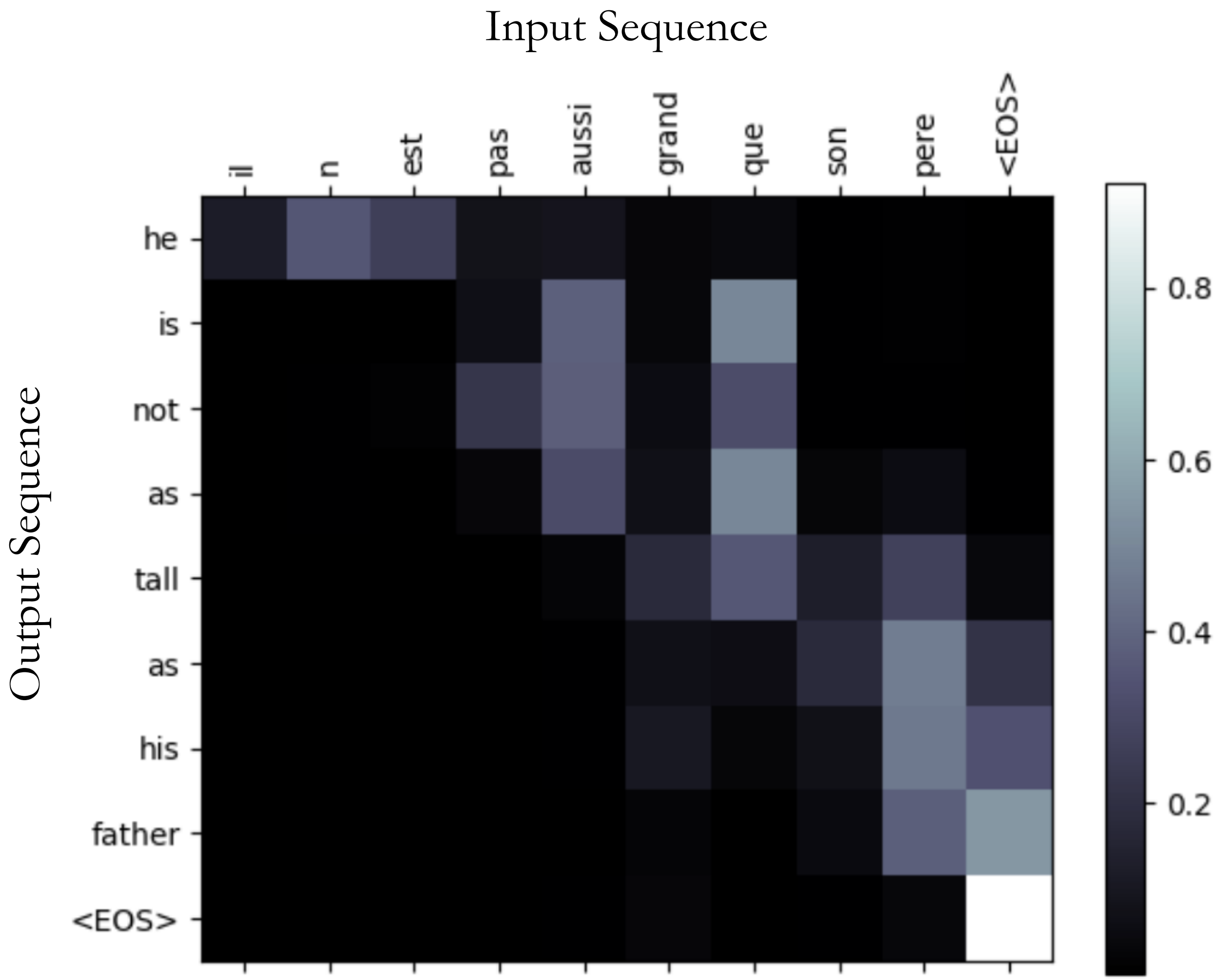


https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html



Because attention is a probability vector, it can enable some interpretation of the model

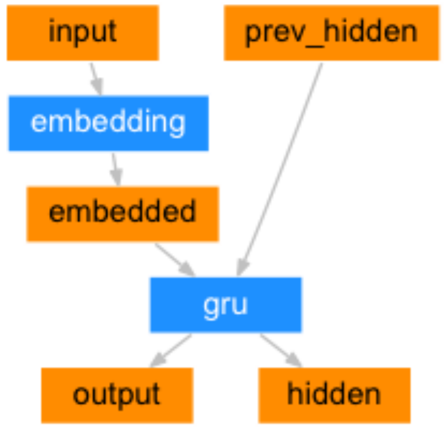
- A visualization of the attention map can reveal interpretable model structure
  - Notice the correspondence between input and output words
- Caution: This is an **abstract view** of the model
  - It should be interpreted with care as many details are hidden
  - It does not answer "why" but rather "what" the model is doing

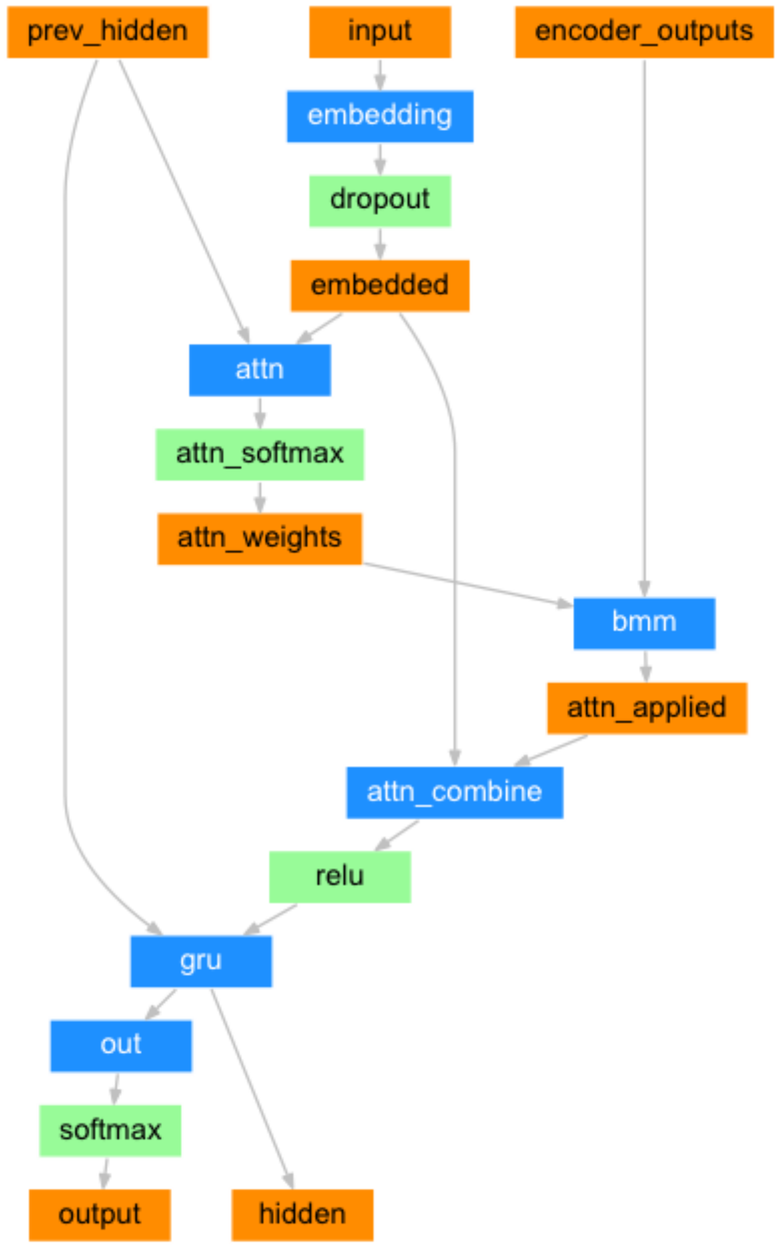


https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html

Demo of seq-2-seq task for French to English translation

- Encoder is simple RNN
- Decoder includes the attention mechanism





Attention can be generalized to many other contexts beyond translation

- Image captioning: Which pixels of the image should be focused on for generating the caption.
- Text prediction: Which previous inputs should be focused on for predicting the next word.
- Summarization: Which words in the document should be focused on for generating the next word in the summary.
- …

Our seq-2-seq attention is a special case of this more general attention mechanism

- The output of attention is a weighted average of the values

$$A(q, K, V) = \sum_i \alpha_i v_i = \sum_i \left(\frac{\exp\left(e(q, k_i)\right)}{\sum_j \exp\left(e(q, k_j)\right)} \right) v_i$$

  - q is the **query** input, K is the key matrix, V is the value matrix
  - α_i is the **attention weight** for the i-th value
  - $e(q, k_i)$ is the **attention score** (pre-softmax) based on the i-th key
    - Function of the query q and the i-th key
    - Intuitively, like a soft/approximate dictionary lookup table
      (i.e., high value if the query matches the key and low value if the query does not match key)

Our seq-2-seq attention is a special case of this more general attention mechanism

- Generalized attention equation

$$A(q, K, V) = \sum_i \alpha_i v_i = \sum_i \left(\frac{\exp\big(e(q, k_i)\big)}{\sum_j \exp\Big(e\big(q, k_j\big)\Big)} \right) v_i$$

- Query was hidden state with input

$$q = h'_{t-1} = [h^D_{t-1}, \tilde{x}_t]$$

- Attention score function was linear where $K = (W, b)$

$$e(q, k_i) = w_i^T q + b_i = w_i^T h'_{t-1} + b_i$$

- Values was encoder values

$$V = H^E$$

- (Attention eqs from before)
- $H^E \coloneqq [h^E_1, h^E_2, \cdots, h^E_L] \in \mathbb{R}^{k \times L_{\max}}$
  (hidden state from encoder)
- $h'_{t-1} = [h^D_{t-1}, \tilde{x}_t]$ (concatenate)
- $\alpha_t = \sigma(W_a h'_{t-1} + b_a) \in \mathbb{R}^{L_{\max}}$
  (softmax for attention weights)
- $c_t = H^E \alpha_t$
  (take weighted average of encoder outputs)

A **self-attention** module computes the queries, keys and values using the input sequence itself

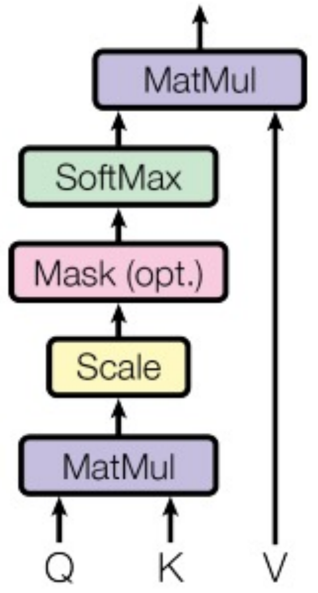
- $X = [x_1, x_2, \dots, x_L]^T \in \mathbb{R}^{L \times D_{in}}$
- Self attention to compute Q, K and V
  - $Q = XW_Q \in \mathbb{R}^{L \times H}$ (ignoring bias term for simplicity)
  - $K = XW_K \in \mathbb{R}^{L \times H}$
  - $V = XW_V \in \mathbb{R}^{L \times D_{out}}$
- Dot product attention scores

$$A(Q, K, V) = \sigma(QK^T)V$$

  - $QK^T \in \mathbb{R}^{L \times L}$ are the **attention scores**
  - The softmax σ is taken over the first dimension
- A single output for a single query has a simple form:

$$A(q, K, V) = \sigma([q^T k_1, \dots, q^T k_L])V = \sum_{\ell} \sigma_\ell([q^T k_1, \dots, q^T k_L])v_\ell$$

Scaled Dot-Product Attention



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*.

Self-attention has quadratic form inside the softmax and a linear form outside

- We can expand the attention mechanism as just a function of the sequence X
- $A_{self}(X) = \sigma(QK^T)V$
- $= \sigma\big(XW_Q(XW_K)^T\big)(XW_V)$
- $= \sigma\big(XW_Q W_K^T X^T\big)(XW_V)$
- Do not be afraid of attention, it's mostly just matrix multiplications :-)

Self-attention is a **permutation-equivariant** neural network module

- What is the key difference between a set and a sequence?
- In sets, the order of the elements doesn't matter!
- The input-output relationship of transformers behave more like sets than a sequences.
- Formally, self-attention is **equivariant** to the input order of the sequence
    - $A_{self}(PX) = PA_{self}(X)$ where P is a permutation matrix that permutes L rows
- Initial example
    - $X = [1,2,3,4]^T$
    - $A_{self}(X) = [6,7,8,9]^T$
- Permuted input produces the same output but permuted
    - $X' = PX = [4,3,2,1]^T$
    - $A_{self}(X') = [9,8,7,6]^T = PA_{self}(X)$

Simple proof of **equivariance** property

- $A_{self}(X) = \sigma\left(XW_Q W_K^T X^T\right)(XW_V)$
- $A_{self}(PX) = \sigma\left((PX)W_Q W_K^T (PX)^T\right)\left((PX)W_V\right)$
- $= \sigma\left(P\left(XW_Q W_K^T X^T\right)P^T\right)PXW_V$
- $= P\sigma\left(XW_Q W_K^T X^T P^T\right)PXW_V$
  (permuting rows and then softmax is equivalent to softmax and then permuting rows)
- $= P\sigma\left(XW_Q W_K^T X^T\right)P^T PXW_V$
  (permuting scores and then softmax is equivalent to softmax and then permuting outputs)
- $= P\sigma\left(XW_Q W_K^T X^T\right)XW_V$
- $= PA_{self}(X)$

Masked attention forces attention weights on future values to be 0

- For generating the output sequence, we cannot let the current word depend on future words, it should only depend on past words
- To enforce this, we add a mask to the attention scores before applying the softmax
- The mask has $-\infty$ where the value should be 0
- This ensures that decoder outputs only depend on previous words/outputs

Multi-headed attention combines multiple attention mechanisms via concatenation

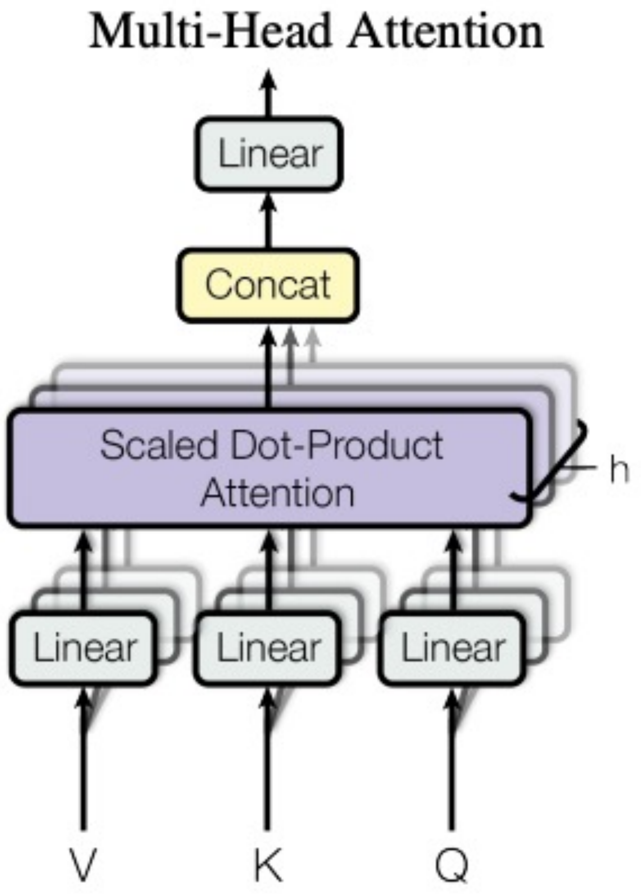
- Suppose we have H attention modules with an output dimension of D_{head}

$$A_1(X), A_2(X), \cdots, A_H(X)$$

- Multiheaded attention simply concatenates the output of each attention and applies a linear function

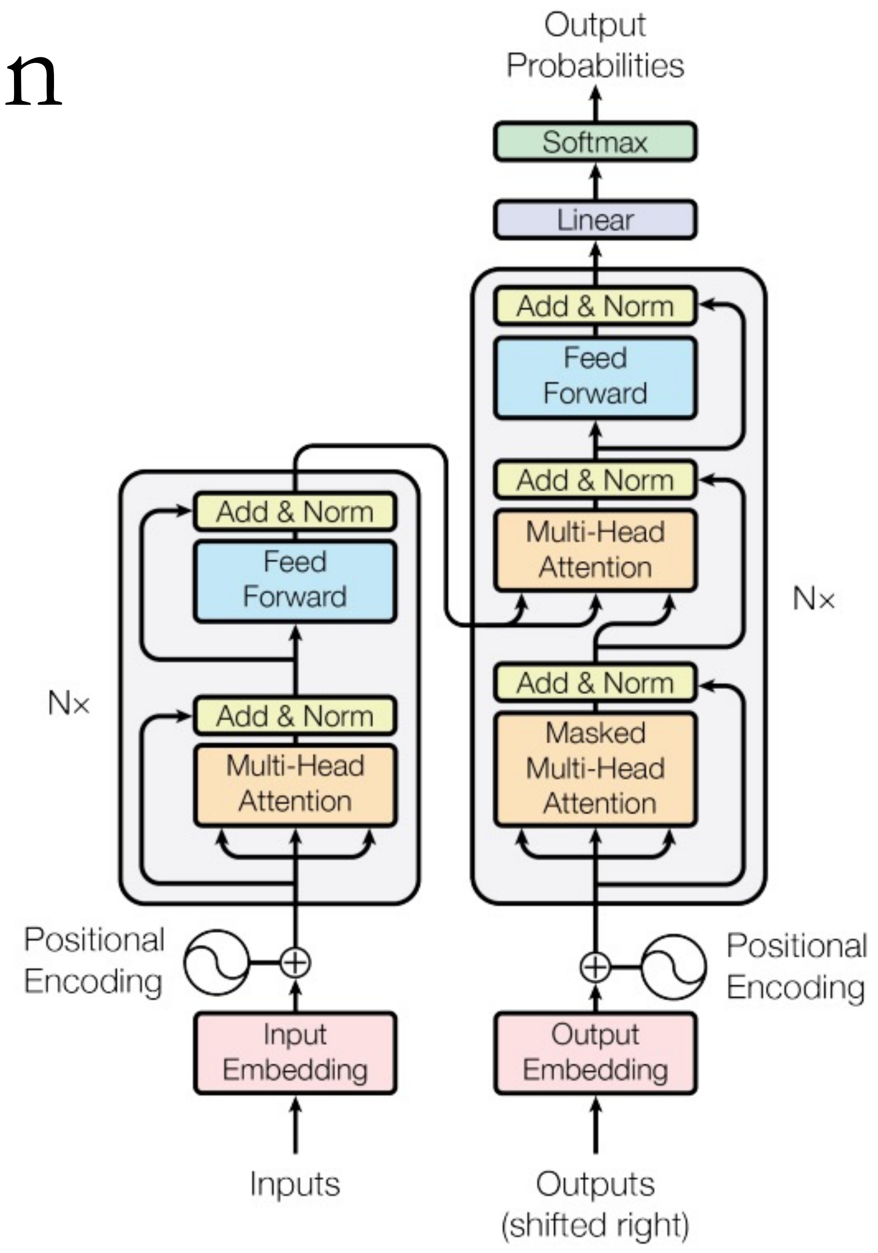
$$A_{multi-head}(X) = [A_1(X), A_2(X), \cdots, A_H(X)]\mathrm{W}_\mathrm{H}$$

  - The concatenated dimension is $D_{head} \cdot H$
  - $W_H \in \mathbb{R}^{(D_{head} \cdot H) \times D_{out}}$



Transformers uses only attention instead of RNN structure

- No RNN structure, parallel
- **Positional encoding (next slide)**
- Multi-headed **scaled** attention
    - Masked version ensures current output does not depend on "future" outputs
    - Cross attention and self-attention
- Includes feed forward layers that operate on each input independently
    - Think of applying a simple NN to a batch of samples (where each sample corresponds to one element of the sequence)



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*.

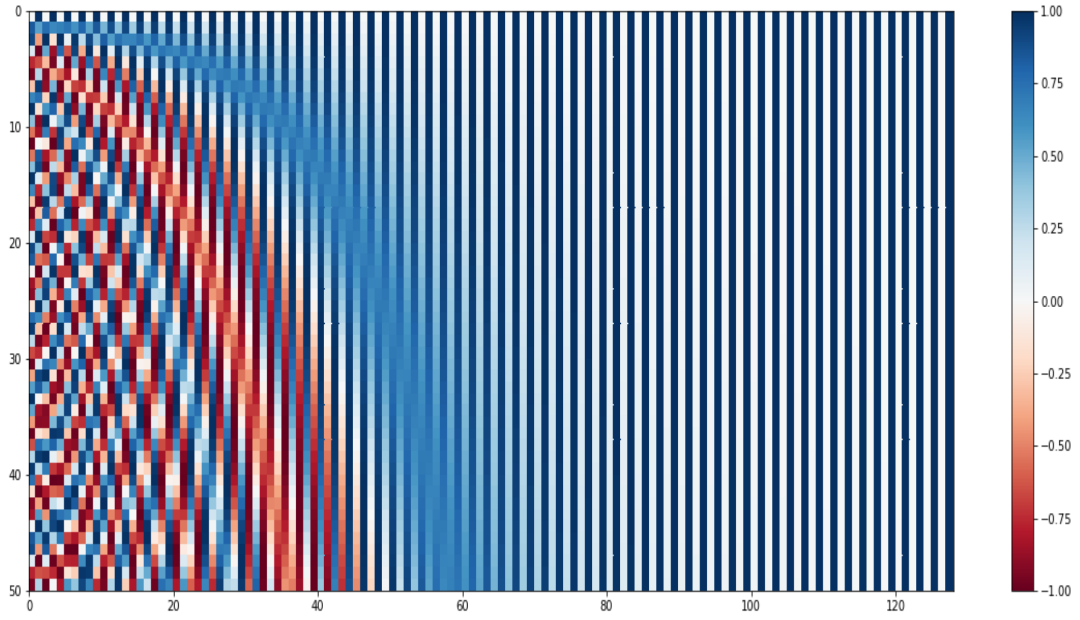
Without positional encoding, the model would be unable to leverage word position information

- Remember that self-attention is **permutation-equivariant** operator
- Similarly, the feed-forward layers are just like operating on a batch of samples so they are also a **permutation-equivariant** operator
- Therefore, it would be challenging for the model to reason about absolute or relative word positions
- Adding positional encoding to the word representation overcomes this issue so that the order of words is embedded in the sequences

Without positional encoding, the model would be unable to leverage word position information

- Positional encoding has the same dimension D as the word embedding
- It alternates between sine and cosine with a geometric progression of wavelengths from 2π to $10{,}000 \cdot 2\pi$
  - $PE_{2i}(t) = \sin\left(\frac{t}{10{,}000^{2i/D}}\right)$
  - $PE_{2i+1}(t) = \cos\left(\frac{t}{10{,}000^{2i/D}}\right)$
- This enables relative and absolute positioning information to be encoded

Position in sentence (up to 50)

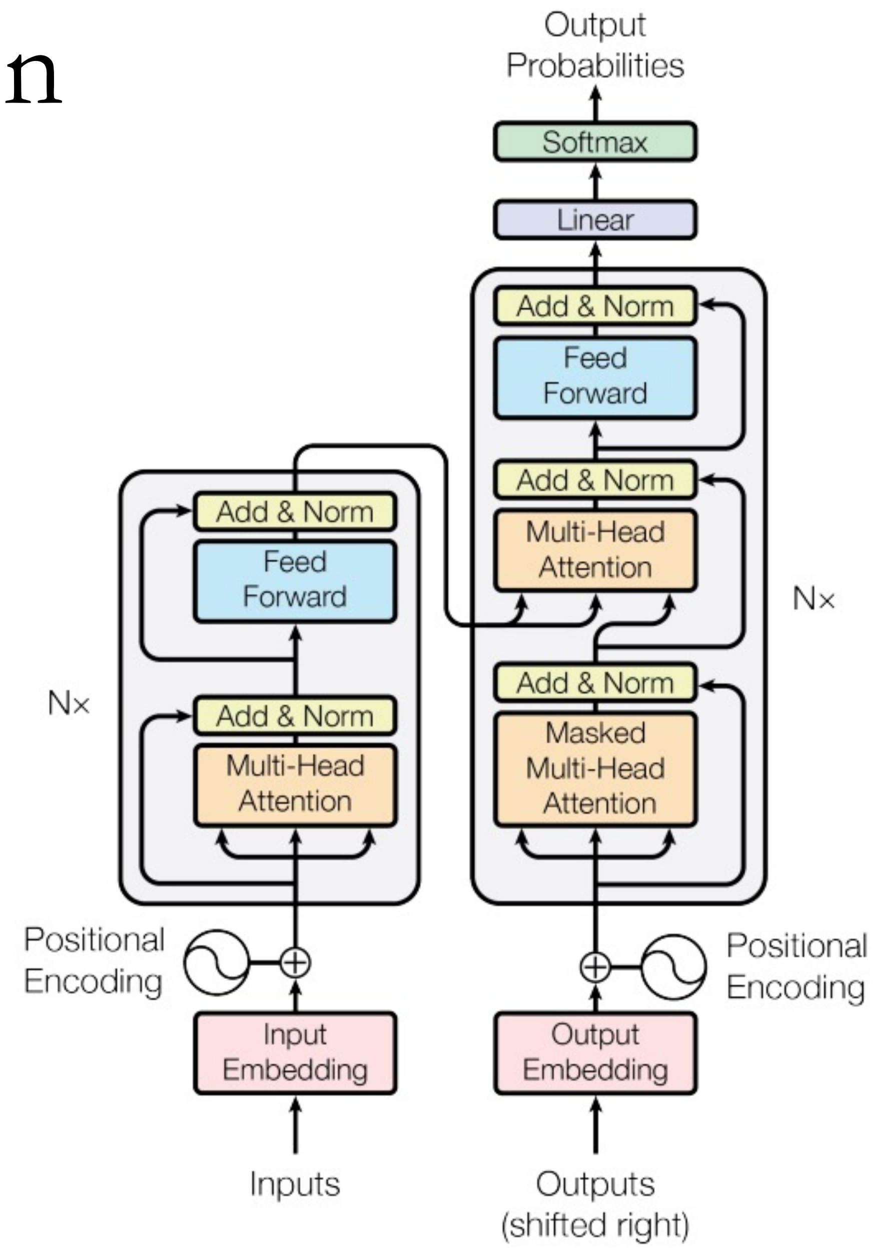


Dimension index at most D

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*.

Transformers uses only attention instead of RNN structure

- No RNN structure, parallel
- **Positional encoding**
- Multi-headed scaled attention
  - Masked version ensures current output does not depend on "future" outputs
  - Cross attention and self-attention
- Includes feed forward layers that operate on each input independently
  - Think of applying a simple NN to a batch of samples (where each sample corresponds to one element of the sequence)



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*.