Assignment 2: attention mechanism in transformer

- Learning outcome:

  1. learning the basic of training a deep neural network, including dataset prepare, training script and evaluation script

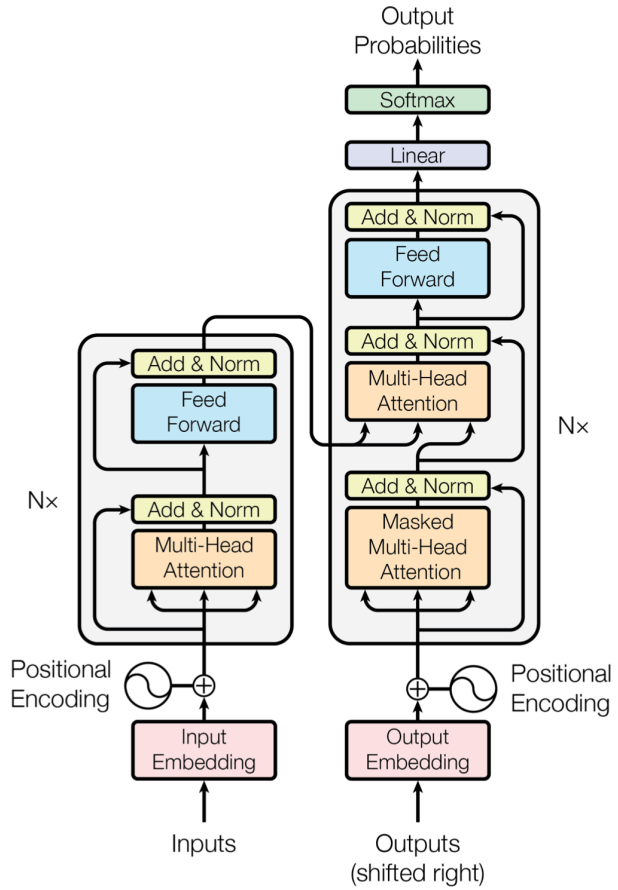
  2. how attention mechanism works

  3. write your own transformer (attention, position encoding, multi-head attention)

  4. training a transformer to caption an image

  5. visualize the results

- Requirement:
- Basic knowledge in Python, PyTorch, NumPy

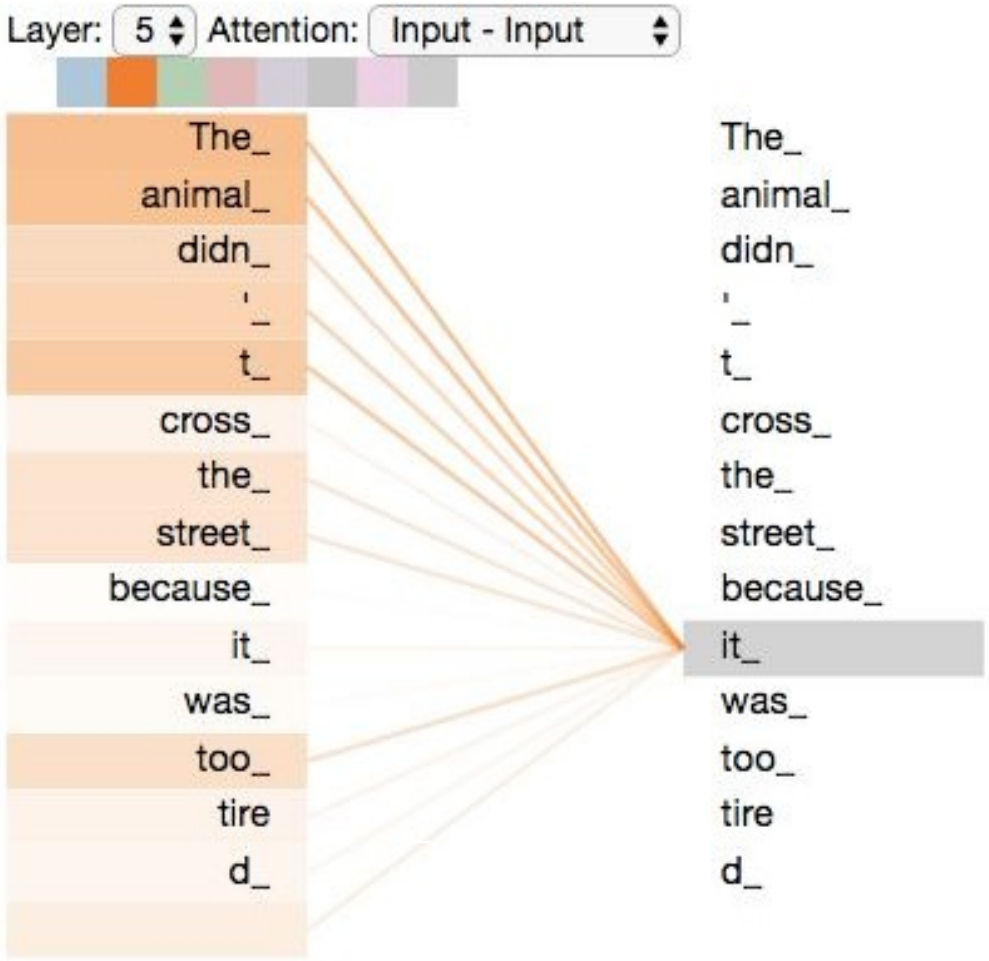


$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Vaswani, A. "Attention is all you need." *Advances in Neural Information Processing Systems* (2017).

Self-Attention

- Each element attends to every other element
- It is an attention mechanism relating different positions of a single sequence in order to compute a representation of the same sequence
- Each element becomes query, key, and value from the input embeddings by multiplying by a weight matrix



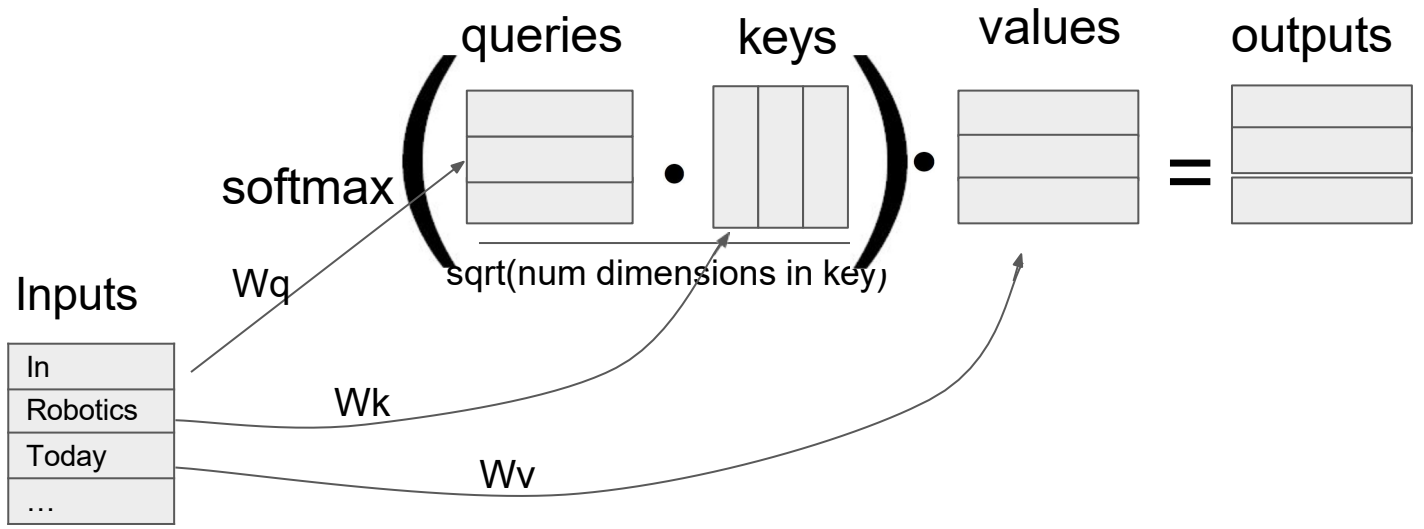
https://jalammar.github.io/illustrated-transformer/

Scaled Dot Product Attention

- Key algorithm in transformers
- Attention weights: how likely each query matches key (used for weighted sum of values)
- Dividing by d_k makes algorithm easier to train
  - prevents binary prediction of one 1 and a whole bunch of zeros
  - helps the gradients flow
- then can take loss, backpropagate, update the weights and values
- NOT stepping through a sequence like with a RNN

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

- **input consists of queries and keys of dimension dk, and values of dimension d**
- **Queries packed into matrix Q**
- **The keys and values are also packed together into matrices K and V .**



Multi-Head Attention

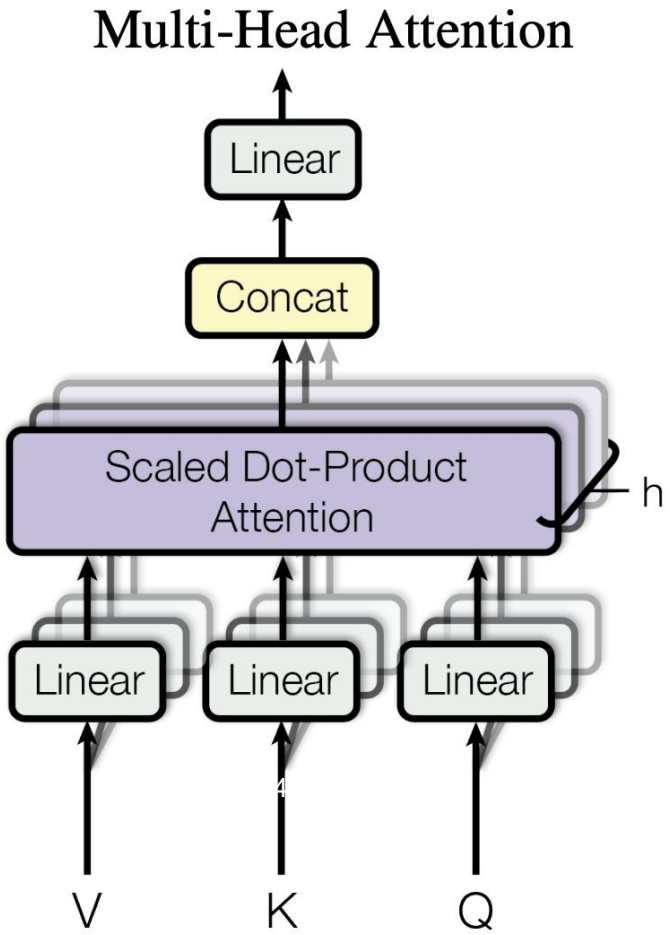
- Idea:
  a. Stack linear layers (weight matrices without biases) that are independent each for keys, queries, values.
  b. Concatenate output of attention heads to form (plus non-linearity) output layer
- Why?
  a. Allows for model to focus on different positions
  b. Gives attention layer multiple “representation subspaces”
  c. No longer need to oversaturate one attention mechanism

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$

$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

$h = 8$ parallel attention layers, or heads.

Learnable parameter matrices



Positional Encoding

- Need for information about the position and order of tokens in a sequence
- Positional Embedding: Vector that represents position of each token
  - Element wise addition the position embedding to the word embedding vector
  - Positional embedding be fixed or learned

"We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset k, PEpos+k can be represented as a linear function of PEpos."

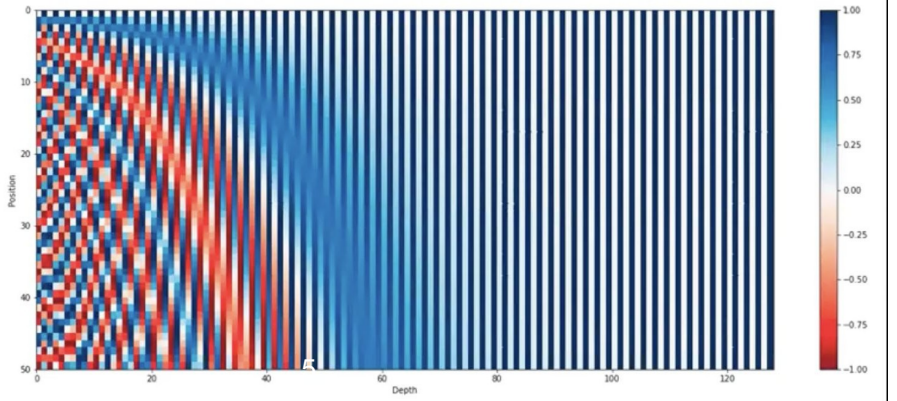
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

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

pos is the position and i is the dimension.

* Sine method:

$$\overrightarrow{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \vdots \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1}$$



For every sine-cosine pair corresponding to frequency ω_k, there is a linear transformation $M \in \mathbb{R}^{2\times 2}$ (independent of t) where the following equation holds:

=>

$$M . \begin{bmatrix} \sin(\omega_k . t) \\ \cos(\omega_k . t) \end{bmatrix} = \begin{bmatrix} \sin(\omega_k . (t+\phi)) \\ \cos(\omega_k . (t+\phi)) \end{bmatrix}$$

$$M_{\phi,k} = \begin{bmatrix} \cos(\omega_k . \phi) & \sin(\omega_k . \phi) \\ -\sin(\omega_k . \phi) & \cos(\omega_k . \phi) \end{bmatrix}$$

Note: positional embeddings are now learned

Full-Architecture