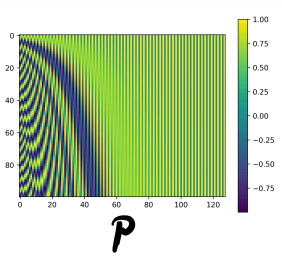
Transformers

Initialization:
Input is a set (or sequence) of tokens from a finite dictionary.
E.g. Text: She works in the white house.

z_1 z_2 z_3 z_4 z_5 z_6 z_7

Map tokens to word vectors. $z_i \in \mathbb{R}^d$
These vectors are learned



P

$+P(1)$ $+P(2)$ Add positional embedding

$z = [z_1 \cdots z_n]$ $\quad z \leftarrow z + P$ $\quad P_{k,2i} = \sin\left(\frac{k}{n^{2i/d}}\right)$ $\quad P_{k,2i+1} = \cos\left(\frac{k}{n^{2i/d}}\right)$

The vector z_i distills the "meaning" of the i^{th} word. Similar words have similar embeddings. e.g. "white" $\simeq$ "blue"

Issue: The vector z_5 of "white" should change as it is followed by "house". Words modify each other's meaning.

(Self-)Attention: High level: Each vector becomes a weighted average of its nearest neighbors and itself.

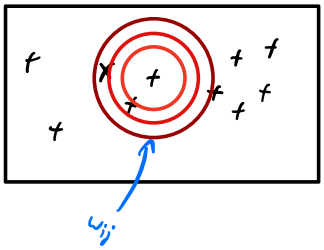
Subtleties:

z_i

$\vec{q}_i := Q z_i$ $\quad \vec{k}_i := K z_i$ $\quad \vec{v}_i := V v_i$

Query
Key
Value

Three linear projection matrices (learned parameters)



Similarity measure between z_i & z_j: $s_{ij} = \frac{q_i^T k_j}{\sqrt{d}}$

Weight of z_j as neighbor for z_i: $w_{ij} = \frac{e^{s_{ij}}}{\sum_{k=1}^{n} e^{s_{ik}}}$ ⇐ softmax function

Self-attention: $\vec{z}_i \leftarrow \sum_{j=1}^{n} w_{ij} \vec{v}_j$

Skip connection

$$z' \leftarrow V z \, \underbrace{\text{softmax}\left(z^T K Q^T z\right)}_{W^T \in \mathbb{R}^{n \times n}} + z$$

Feed-Forward Neural Net:

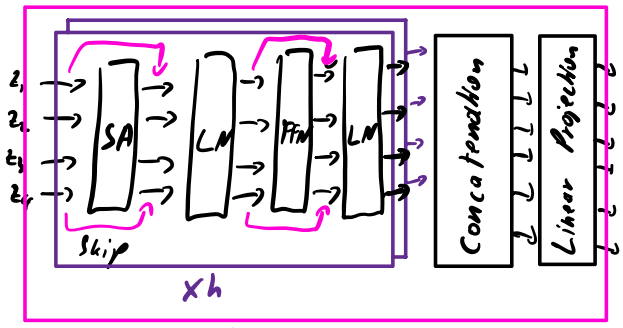
After the self-attention, each vector is processed by a FFN (=MLP).

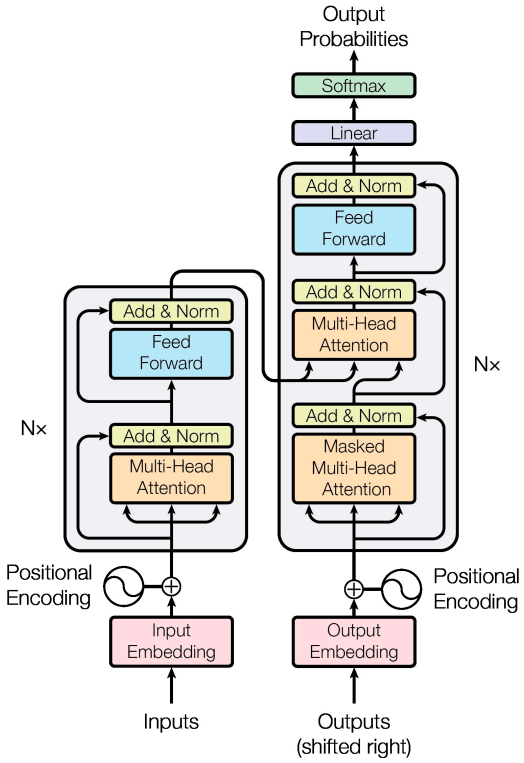
MLP Skip-connection

$z_i'' \leftarrow V\sigma(Uz_i) + z_i'$

ReLU

Transformer Layer:





SA: Self-Attention
LN: Layer Normalization
FFN: Feed Forward Neural Net (=MLP)

LLM: Predict the next word:

I like to eat

Input

context

output: (probability)

bananas 0.1
cherries 0.05
fruit 0.03
cake 0.02

K-class classification problem

↑ number of possible tokens.

Encoder turns context into word vectors.

Decoder predicts the next word. After a word is predicted it is inserted into the decoder as next input.

Sampling: The decoder outputs a probability vector. Top-p sampling samples words up to a probability mass of p.

Eg. p = 0.18 only sample (bananas, cherries, fruit)

Emergent abilities: Large Langue Models learn to reason, do elementary math,