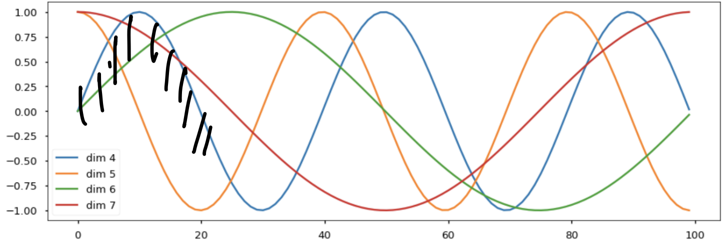
Positional Encoding: Use Sinusoids

Use Sinusoids of varying periods:

$$p_i = \begin{pmatrix} \sin(i/10000^{2*1/d}) \\ \cos(i/10000^{2*1/d}) \\ \vdots \\ \sin(i/10000^{2*\frac{d}{2}/d}) \\ \cos(i/10000^{2*\frac{d}{2}/d}) \end{pmatrix}$$

512 token
dim

Pos^n					
1	0	1	0	1	0 1
2	0.4	0.9	0.05	0.95	



multi-head attention

single-head?

grammatical structure

morphology

$Q^{(1)}, K^{(1)}, V^{(1)}$

attn head 1

Semantics / topics

(query)

query,

key,

value

$e_1 \qquad e_2 \qquad \ldots \qquad e_n$

$x_1 \qquad x_2 \qquad \ldots \qquad x_n$

$\sum w_{ij} e_j$

$w_{ij} = e_i \cdot e_j$

$E \quad V \times d$

Position representations through learned embeddings

- We can posit a new parameter matrix $P \in R^{N \times d}$, where N is the maximum length of any sequence that the model can process
- These position representations are then added to the word embeddings

$$\tilde{x}_i = x_i + P_i$$

[BERT]

A quick detail about the vocabulary

Let's take a look at the assumptions we've made about a language's vocabulary.

We assume a fixed vocab of tens of thousands of words, built from the training set.
All *novel* words seen at test time are mapped to a single UNK.

	word		vocab mapping	embedding
Common words	hat	→	pizza (index)	
	learn	→	tasty (index)	
Variations	taaaaasty	→	UNK (index)	
misspellings	laern	→	UNK (index)	
novel items	Transformerify	→	UNK (index)	

nationality

The byte-pair encoding algorithm

BPE

- Start with a vocabulary containing only characters and an “end of word” symbol.
- Using a corpus of text, find the most common adjacent characters “a,b”; add “ab” as a subword
- Replace instances of the character pair with the new subword ; repeat until desired vocab

A word segmentation algorithm:

- Start with a vocabulary of characters
- Most frequent ngram pairs ↦ a new ngram

ow 7
es 9
st 9

Dictionary

5	l o w
2	l o w e r
6	n e w e s t
3	w i d e s t

Vocabulary

l, o, w, e, r, n, ~~w~~, s, t, i, d

Start with all characters in vocab

Byte Pair Encoding

A word segmentation algorithm:

- Start with a vocabulary of characters
- Most frequent ngram pairs ↦ a new ngram

8

Dictionary

5	l o w
2	l o w e r
6	n e w **es** t
3	w i d **es** t

Vocabulary

l, o, w, e, r, n, w, s, t, i, d, **es**

Add a pair (e, s) with freq 9

es t

A word segmentation algorithm:

- Start with a vocabulary of characters
- Most frequent ngram pairs ↦ a new ngram

Dictionary

5	l o w
2	l o w e r
6	n e w **est**
3	w i d **est**

Vocabulary

l, o, w, e, r, n, w, s, t, i, d, es, **est**

Add a pair (es, t) with freq 9

Byte Pair Encoding

- A word segmentation algorithm:
  - Start with a vocabulary of characters
  - Most frequent ngram pairs $\mapsto$ a new ngram

Dictionary

5	**lo** w
2	**lo** w e r
6	n e w est
3	w i d est

Vocabulary

l, o, w, e, r, n, w, s, t, i, d, es, est, **lo**

Add a pair (l, o) with freq 7

z ab cd

m

z xyz

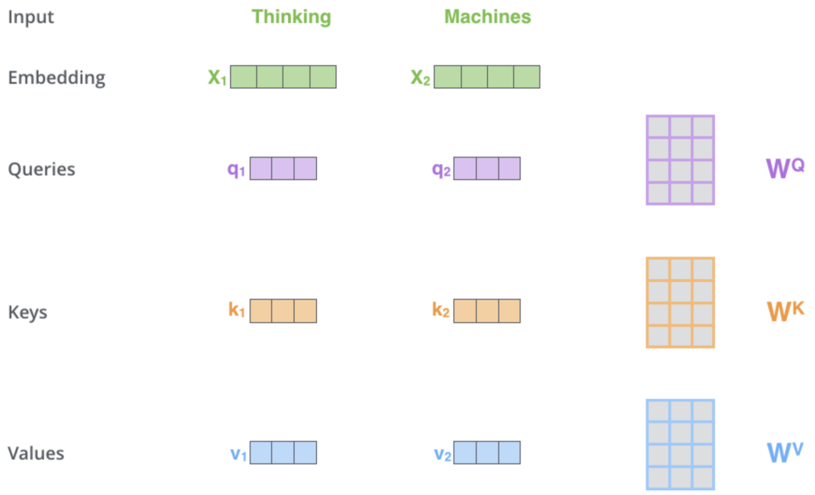
Byte Pair Encoding

32K BPE

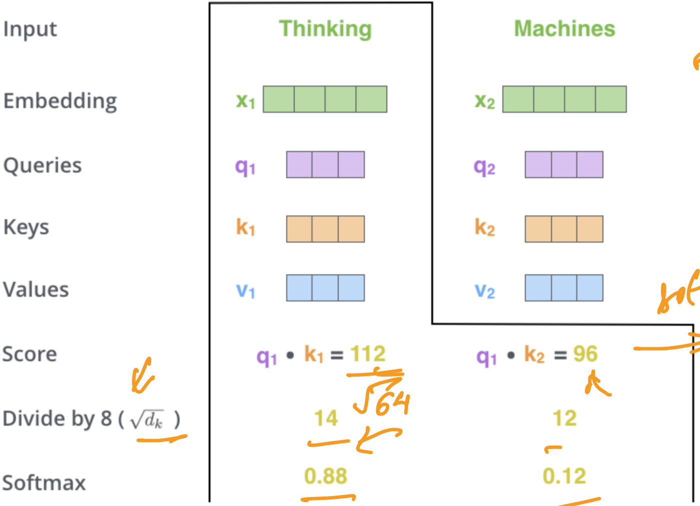
When do we stop

When the desired vocabulary size is met

More Details: Self-attention



Scaled Dot Product for Attention



$\sum q_i k_i$

d_k

softmax

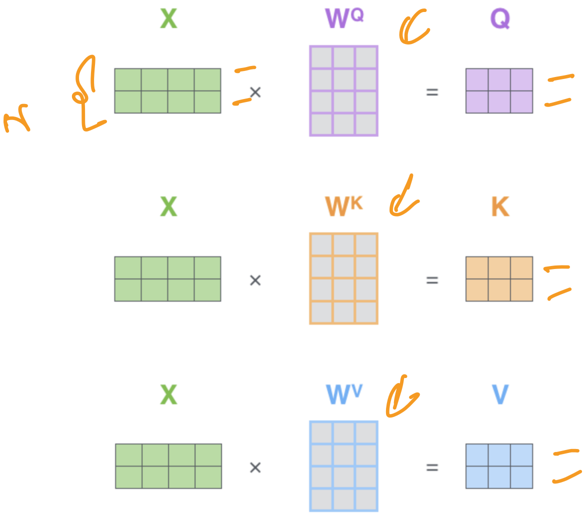
Why is scaling required?

Scaled dot-product

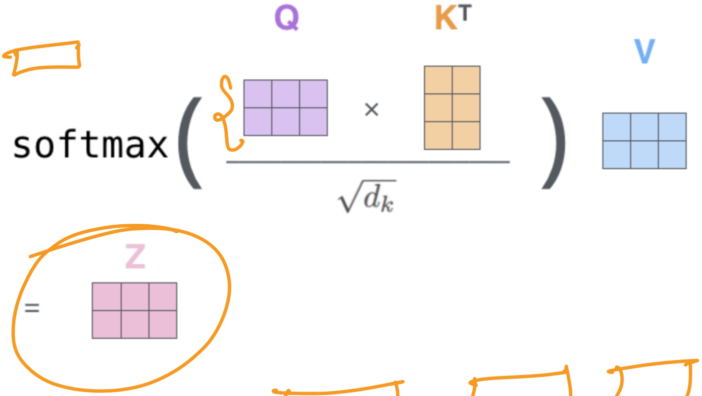
- Assume that the components of q and k are independent random variables with mean 0 and variance 1.
- Then their dot product $q \cdot k = \sum_{i=1}^{d_k} q_i k_i$ has mean 0 and variance d_k.
- Hence the scaling ensures that the resultant dot product has mean 0 and variance 1.

$\sqrt{d_k}$

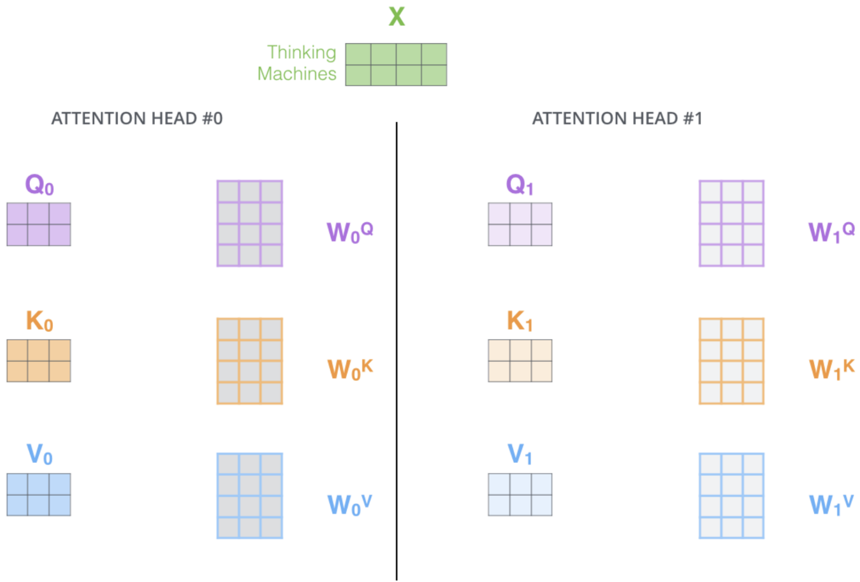
Matrix Calculation of Self Attention



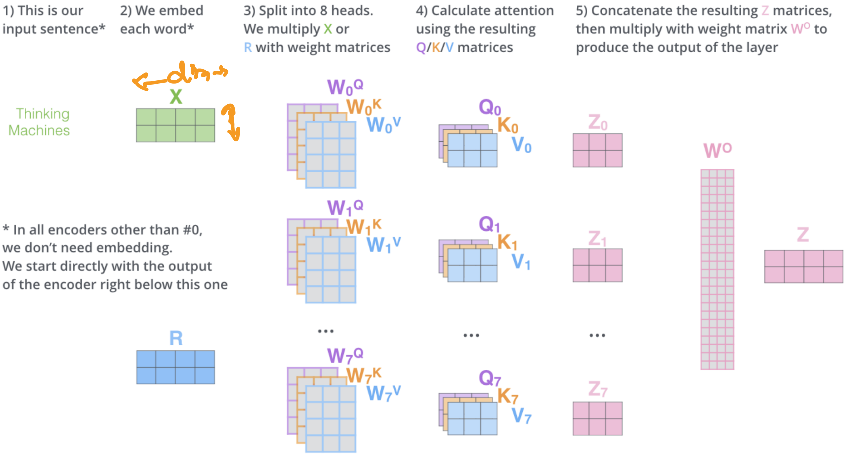
Matrix Calculation of Self Attention



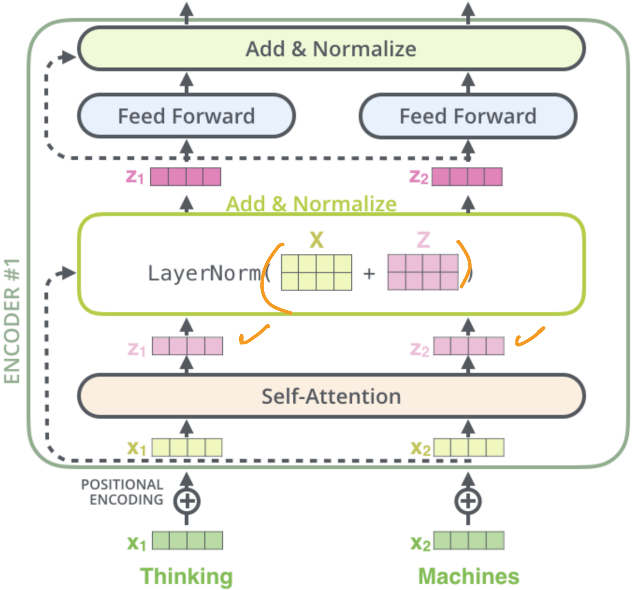
Multi-headed Attention



Multi-headed Self Attention Block



Encoder Block



What is Layer Normalization?

Basic Idea

Cut down on uninformative variation in hidden layer vectors by normalization

i/p to layer Norm

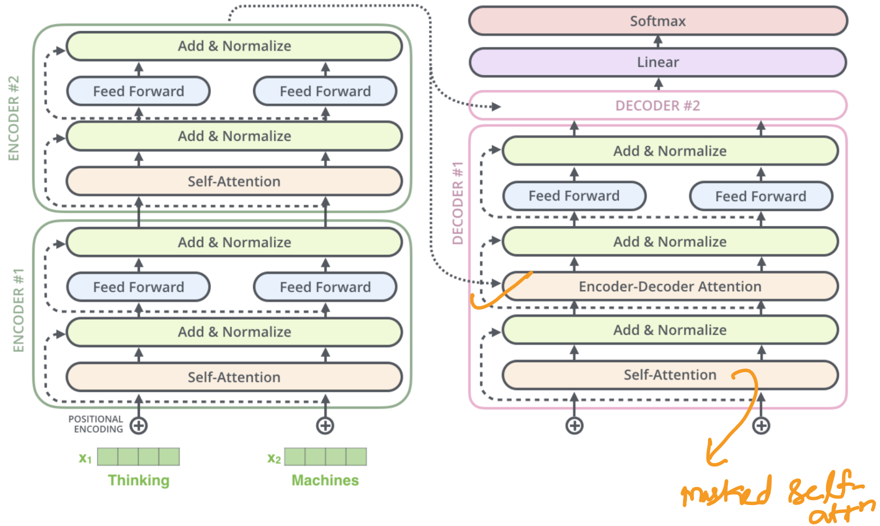
How does the normalization work

Let $x = \{x_1, \ldots, x_d\}$ be an individual word vector in the model.
Let the mean be $\mu = \frac{1}{d}\sum_{i=1}^{d} x_i$, $\mu \in R$
Let the variance be $\sigma = \sqrt{\frac{1}{d}\sum_{i=1}^{d}(x_i - \mu)^2}$, $\sigma \in R$
Let $\gamma \in R^d$ and $\beta \in R^d$ be learnable parameters (can omit)

$$LayerNorm = \gamma\left(\frac{x-\mu}{\sigma+\varepsilon}\right) + \beta$$

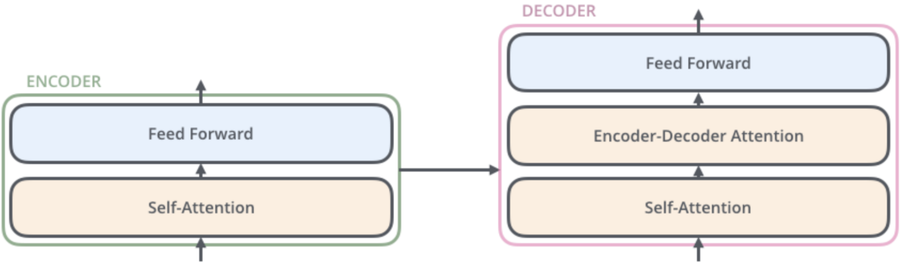
Original paper also used γ and β as learnable parameters, similar to BatchNorm, but later papers showed that those are not important (and are even harmful).

Transformer with 2 stacked encoders and decoders



How is the decoder different?

$x_1 \quad x_2 \quad x_3$

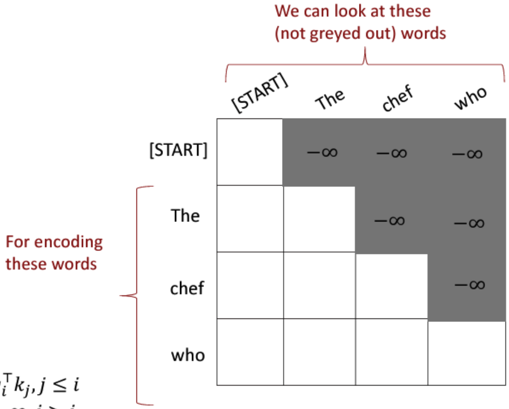


$x_1 \quad x_2 \quad x_3 \quad x_4$

Masking the future in self-attention

- To use self-attention in **decoders**, we need to ensure we can't peek at the future.
- At every timestep, we could change the set of **keys and queries** to include only past words. (Inefficient!)
- To enable parallelization, we **mask out attention** to future words by setting attention scores to $-\infty$.

$$e_{ij} = \begin{cases} q_i^\top k_j, j \leq i \\ -\infty, j > i \end{cases}$$



I went to the market

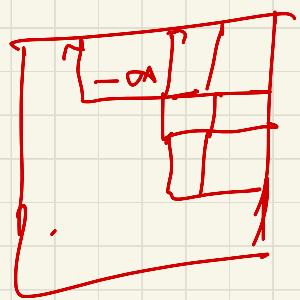
मैं बाजार गया

(1) hout

(2) hout

h I hwent —— –

I went to the market



<start> <मैं> <बाजार