Page No.
Date

Week 4: Transformers

1) Transformer Network Intuition

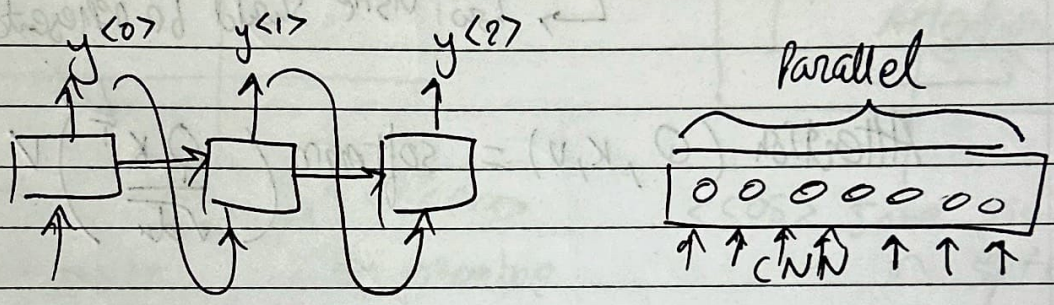
sequential models:

RNN → GRU → LSTM

increased complexity

- Attention + CNN
  - → self-Attention
  - → Multi-head attention (for loop over self attention)

$A^{<1>}$ $A^{<2>}$ $A^{<3>}$ $A^{<4>}$ $A^{<5>}$



2) Self-Attention

Jane visite l'Afrique en Septembre

$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ $x^{<4>}$ $x^{<5>}$

$A(q, K, v)$ = attention-based vector representation of a word (calculate for each word $A^{<1>}, \ldots, A^{<5>}$)

To get $A^{<3>}$, it will look at the surrounding words to get the content of l'Afrique i.e. whether it is a holiday destination or historical representation or the second largest continent in the world.

RNN attention

$$\alpha^{<t,t'>} = \frac{\exp(e^{<t,t'>})}{\sum_{t'=1}^{T_x} \exp(e^{<t,t'>})}$$

Transformers Attention

$$A(q,K,V) = \sum_i \frac{\exp(q \cdot k^{<i>})}{\sum_j \exp(q \cdot k^{<j>})} v^{<i>}$$

for l'Afrique → $q^{<3>}, k^{<3>}, v^{<3>}$

(query key value pairs)

eg:

(whats happening in l'Afrique? (visite))

$q^{<3>} = W^Q \cdot x^{<3>}$ → asks a question (vectorized version)

$k^{<3>} = W^K \cdot x^{<3>}$ → figures out most relevent answer to q

$v^{<3>} = W^V \cdot x^{<3>}$ → how visite should be represented in A

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

3) Multi-Head Attention

h = #heads

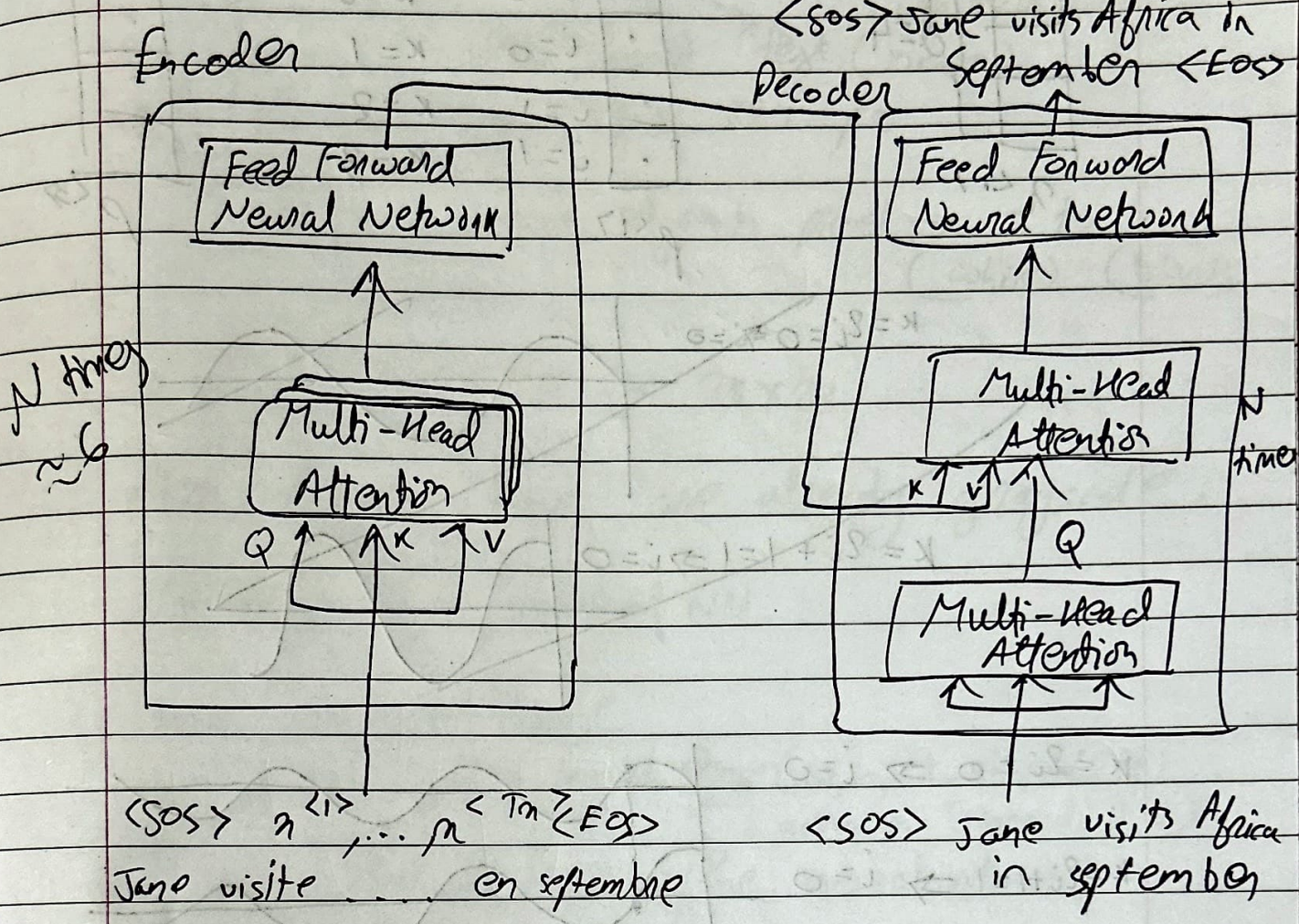
All the heads are computed in parallel instead of sequentially.

$$\text{head}_i = \text{Attention}(W_i^Q Q, W_i^K K, W_i^V, V)$$

$$\text{MultiHead}(Q,K,V) = \text{concat}(\text{head}_1, \text{head}_2, \ldots \text{head}_h) \cdot W_o$$

Page No.
Date

4) Transformer Network



Both encoder and decoder blocks run/iterate N times. Each of the output word generated is feed again into the decoder as input.

(1) Positional encoding

Equations: $PE_{(pos, 2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right)$

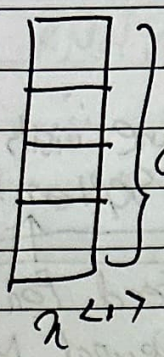
$PE_{(pos, 2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right)$

d => dimension of the word embedding and positional encoding

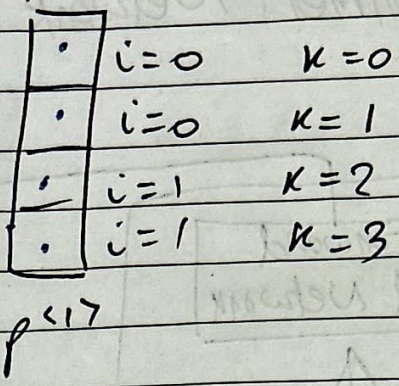
pos => position of the word

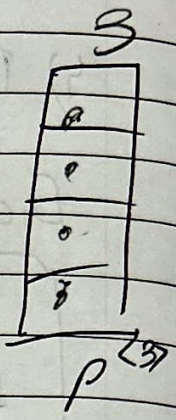
* k//2 = i

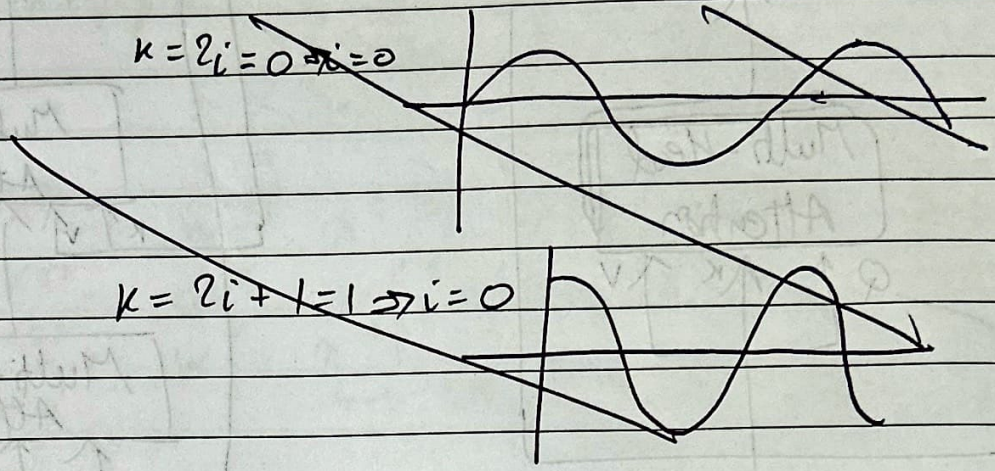
word embedding (vector)

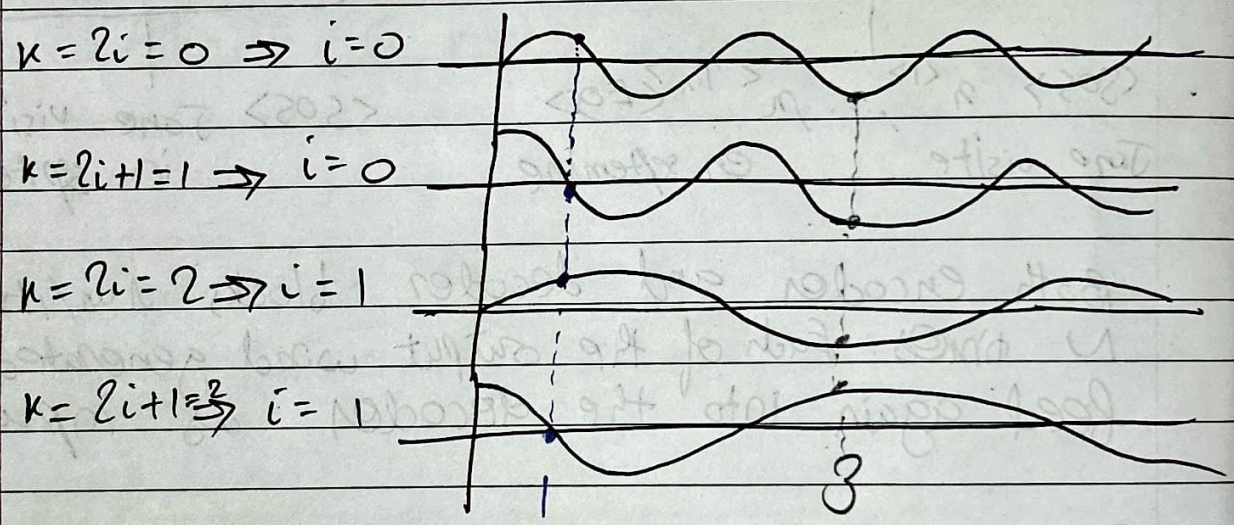


pos = 1









$p^{<3>}$ will be different than $p^{<1>}$

The transformer also uses batch Norm and Add Norm type of layers to speed up learning.