Lec 5 Transformer

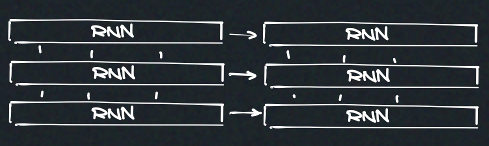
Transformer: self-attention or
self-attention + cross-attention

Query keys to get attn. scores. softmax to get weight

Attention is All You Need

seq2seq model entirely based on attention

Before:



Transformer:
- fast!
- everything matrix

Types:
- Encoder-decoder — T5, MBART
- Decoder-only — GPT, LLaMa

→ See Transformer diagrams

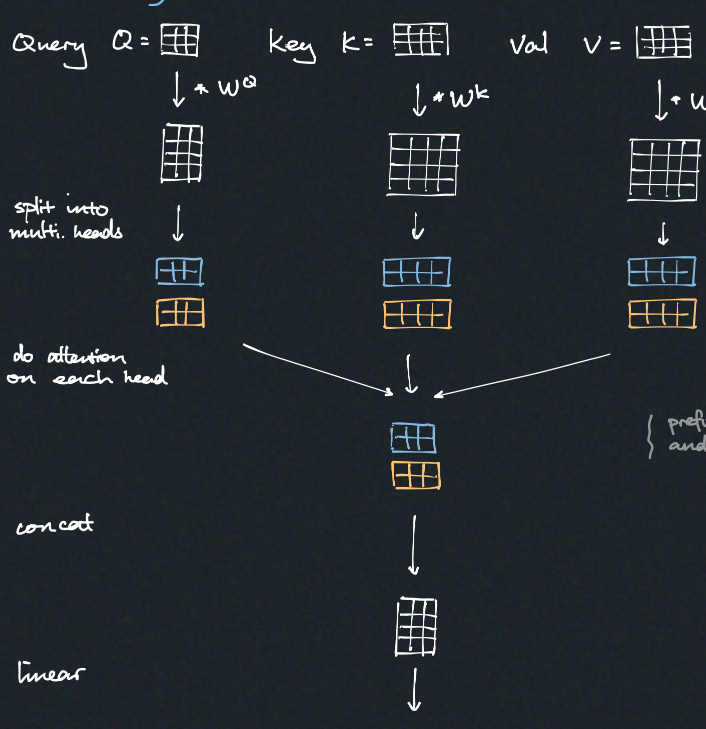
Input: split using subwords

▸ Multi-head attention — attend to different parts of sentence to disambiguate

attend multiple places for multiple functions

I run a small business
I run a mile
The robber made a run for it
The stocking had a run

$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_{1 \dots n})$
$\text{head}_i = \text{Attention}(W^Q Q_i, W^K K_i, W^V K_i)$



{ prefix LM: do both masked and unmasked attention

▸ Positional Emb — deal with attention ignoring order

A big dog and a big cat

$W_{big} + W_{pos2}$ $\quad$ $W_{big} + W_{pos6}$

→ Sinusoidal func $\quad p_t^{(i)} = f(t)^{(i)} = \begin{cases} \sin(w_k \cdot t) & i = 2k \\ \cos(w_k \cdot t) & i = 2k+1 \end{cases}$

$$w_k = \frac{1}{10000^{2k/d}}$$

motivation: dot prod of two embs higher if words closer

→ Or just learn the positional embedding
 └ simpler, but won't extrapolate to longer input

→ Abs vs. Rel encoding
 └ positional, and hope relative info is recovered
 explicitely encode relative location e.g. whether key is -5 of query

→ Rotary Positional Encoding ← used in LLaMa — dot prod of emb and rel position

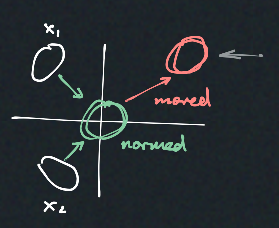
$f_q(x_m, m) \cdot f_k(x_n, n) = g(x_m, x_n, m-n)$

{ RoPE uses imaginary number stuff involved

Training stability

▸ Layer normalisation

$$\text{LayerNorm}(x; g, b) = \frac{\overset{\text{gain}}{g}}{\underset{\text{vec stddev}}{\sigma(x)}} \odot (x - \underset{\text{vec mean}}{\mu(x)}) + \overset{\text{bias}}{b}$$

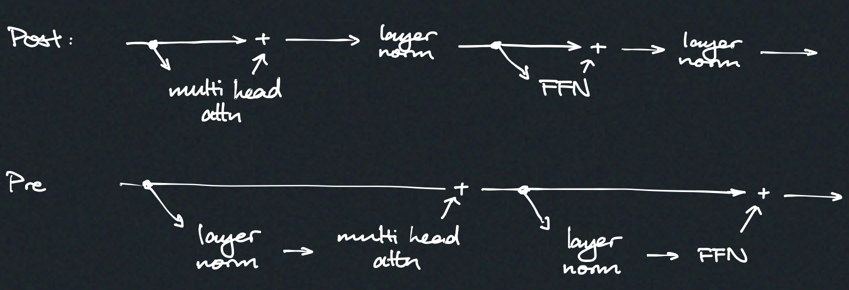


← afterward things go to similar space

▸ RMSNorm — simplified LayerNorm, no mean subtraction & no bias

▸ Residual Connections
from input to Add & Norm
→ attension learns to attend elsewhere, since self is free through residual

▸ Pre / post layer norm



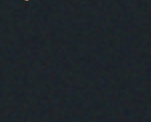
▸ Feedforward layer (FFN)

$$\text{FFN}(x; \underbrace{W_1, b_1, W_2, b_2}_{\text{sometimes just disable}}) = \underbrace{f}_{\text{non linearity}}(xW_1 + b_1)W_2 + b_2$$

Choosing f

- ReLU

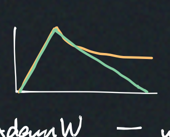
↙ LLaMa
- Swish
 └ SiLU



Model Optimisation

▸ Optimisers

- SGD
- Adam
- Vaswani+ 2017 lr schedule



- AdamW — weight decay to regularise Adam

▸ Low-precision Training

- 16 bit training
 └ IEEE half precision, 5-bit exponent
 └ bfloat16, 8-bit precision ← usually more stable

▸ Checkpoint & Restart

- Monitor gradient norm
- Check your code

Compairing Transformers

Original	LLaMa
post norm	pre norm
LayerNorm	RMSNorm
ReLU	SiLU
Sinusoidal	RoPE