Summary:

single self attn:

$Z = \text{softmax}(QK^T) \cdot V$ $\quad Q = XW^Q$

$\quad K = XW^K$

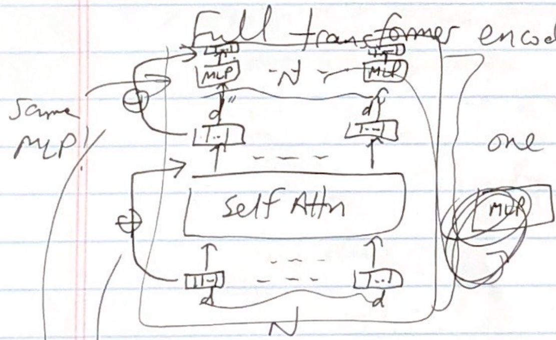
mult-head attn: $(z_1, \ldots z_h) W^O = Z$ $\quad V = XW^V$

- can check: perm equivariant! interchanging order of rows of X will result in same output w/ output rows interchanged.

- "MLP w/ input-dependent weights"

$$Z \sim \big(\underbrace{XW^Q(W^K)^T}_{\text{this is like X-dep weight}} X\big) \cdot XW^V$$

Full transformer encoder architecture



one encoder block

N → N

can stack!

(original "Attn is all you need" had 6 encoder blocks)

+ residual/skip connections!

The encoder produces a transformed embedding that takes into acct entire input w/ all its inter relations

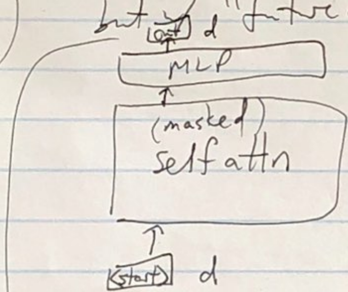
Transformer Decoder

To generate ~~text~~, need the decoder architecture

(original transformer papers had encoder-decoder pair w/ add'l en-dec att'n layer...)

↓

Same structure as encoder but w/ "future masking"



→ output is vector z w/ same dim as original embedding.

~~interpreted as "score" for~~

Multiply into vocabulary/dictionary embedding matrix (~50k for English)

$$\underbrace{Z^T \cdot D}_{N_{vocab}\text{ dim'l vector}} \leftarrow d\Big\{\overbrace{\left(\,|\;|\;|\;\cdots\;|\,\right)}^{N_{vocab}}$$

N_{vocab} dim'l vector

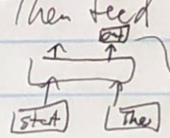
↳ interpret as score for each word in vocab

choose word w/ highest score, or top-k etc.

↓

e.g. "The"

Then feed back into decoder



→ e.g. "Thing" | use ~~emb~~ embedding of last word (which knows about entire preceding sequence) to predict next word

---

encoders useful for learning embeddings

↓

Classification / categorization

sentiment analysis

⋮

BERT example

"Bidirectional Encoder Representations from Transformers"

Google (2018)

QK^T

- Future masking: don't want attention scores to depend on future words

$$\text{softmax}\left(\text{Mask}: \begin{array}{c|cccc} & \langle\text{start}\rangle & I & \text{am} & \text{fine} \\ \hline \langle\text{start}\rangle & \cdot & -\infty & -\infty & -\infty \\ I & \cdot & \cdot & -\infty & -\infty \\ \text{am} & \cdot & \cdot & \cdot & -\infty \\ \text{fine} & \cdot & \cdot & \cdot & \cdot \end{array} \cdot QK^T \right)$$

- This is an example of "autoregressive model"

  learning $P(x_1, \ldots x_n) = P(x_1)\,P(x_2|x_1)\,P(x_3|x_1,x_2) \cdots P(x_n|x_1, \ldots x_{n-1})$

  - GPT is an example of this decoder only model

    Generative Pre-trained Transformer OpenAI (2018)

    GPT $\longrightarrow$ BERT $\longrightarrow$ GPT2 $\longrightarrow$ . . .

    now GPT-family (decoder only) are state of the art!

  - Can also give "prompt" $\rightarrow$ just an initial sequence instead of <start>

  - prompt can be diff language $\longrightarrow$ translation!

- Both BERT & GPT use self-supervised pre-training to learn attn model

  BERT: "masked language modeling"

  GPT: "next word prediction"

  using huge amount of data & giant model

  "foundation model" "backbone"

  fine tuning on "downstream" tasks (eg translation, classification, . . .)

Examples of applications of transformers

Qu, Li, Qian 2202.03772 "Particle Transformer for Jet Tagging"[1]

- Introduced permutation equiv. arch. for jet tagging based on transformers
- Also introduced new dataset "JetClass" for training it

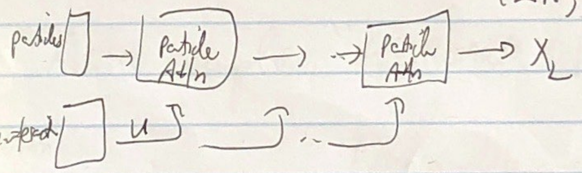
(previous datasets ~1M) 100M jets — 10 types x 10M each

- 4 vec
- particle ID (chg hadron, neutral hadron, e, μ, γ)
  ↳ not truth, reco!
- displacement

new to JetClass

- particle features & pairwise "interaction" features $N \times N \times f_X$

(ΔR, k_T, z, m^2)



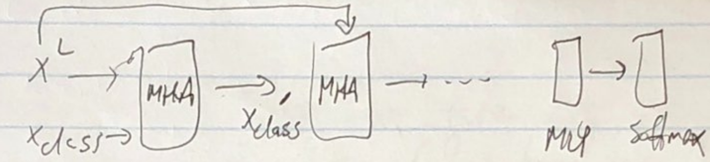
$$\text{softmax}\left(\frac{QK^T}{\sqrt{d}} + U\right) V$$

↳ add interaction mtx to attn
add physics to learned attn

- "class attn" — (don't completely understand this)
  extract classifier output — standard trick in "vision transformer"[3]
  randomly initialized token
  accumulates ~~attn~~ info from event from attention to other tokens

$$\begin{cases} Q = W_q x_{class} \\ K = W_k [x_{class}, x_L] \\ V = W_v [x_{class}, x_L] \end{cases}$$

Class attn (from vision transformer lit)



starts off random but accumulates info

- PartT achieved SOTA results on all jet tagging tasks!

example of "foundation model"

- Also interesting: pretraining + fine tuning on smaller dataset >> training from scratch on smaller dataset

- Also transformer benefited more from pretraining than GNN!

- Astro example: "ASTROMER transf. based embedding for repr of light curves" Donoso-Oliva et al 2205.01677

cf TimeModAttn example from Astro presentation — pretrained on sim

- self supervised, like NLP etc unlike PartT
- positional encoding (not perm equiv)

masked light curve modeling
encoder-decoder

- Data R-band light curves of MACHO survey (Galactic Bulge & LMC) for pretraining — 1.5M light curves

- 20k labeled variable stars for f.t. & eval
Also OGLE-III 360k labeled; ATLAS 422k

pretraining + finetuning
Also outperforms classifiers trained just on those datasets!

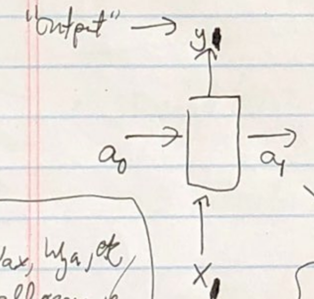
Although transformers have "taken" over, prior approaches to sequence modeling still useful to have in toolbox (as in ASTROMER example) — RNN, LSTM, GRU

Ref: Stanford CS-230 "cheatsheet" for RNNs

Brief overview of RNN, LSTM, GRU architectures

— very sequential, not at all perm equiv.

— not feed forward per se — or like FF w/ # layers depending on length of sequence



Sequence $x_0, x_1, x_2, \dots$

"hidden state"

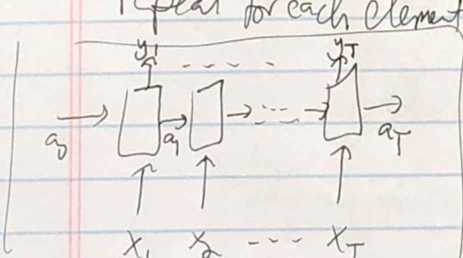
W_{aa}, W_{ax}, W_{ya}, etc. same for all sequence "weight sharing"

$$\begin{cases} a_1 = g_1(W_{aa} a_0 + W_{ax} x_1 + b_a) \\ y_1 = g_2(W_{ya} a_1 + b_y) \end{cases}$$

depends on prev hidden state + current input

depends on current hidden state

repeat for each element of sequence



produces transformed sequence + hidden state that encodes full sequence

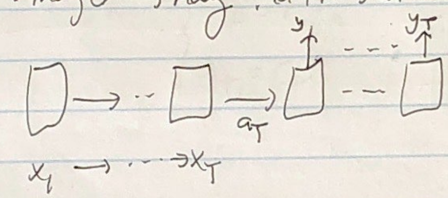
- Can also use as 1 → many

$x_1 \to y_1 \to y_2 \to \cdots$ (music, text gen)

- many → 1 : just keep y_T → classification

- many → many : name entity recog (classify each token according to type) esp named entities (person, corp., ...)

- many → many : translation



- vanilla RNNs struggle w/ vanishing/expl gradients — long sequences information is lost
  - also: slower to train since sequential

short term memory problem

→ introduce "gated" RNNs (analog of residual/skip connections?)

Two popular examples: Gated Rec. unit (GRU)

Long Short-Term Memory (LSTM)

GRU ⊂ LSTM (special case, LSTM more general)

hidden + cell state

local short term info

global info long term