Transformers & Attention Mechanism

- Used in SOTA NLP like ChatGPT
- used in pretty much everything now
- permutation-equivariant architecture!
- actually a type of GNN
- like an MLP w/ input-dependent weights

$$X \to w(x) \cdot X$$

→ much more powerful in principle!

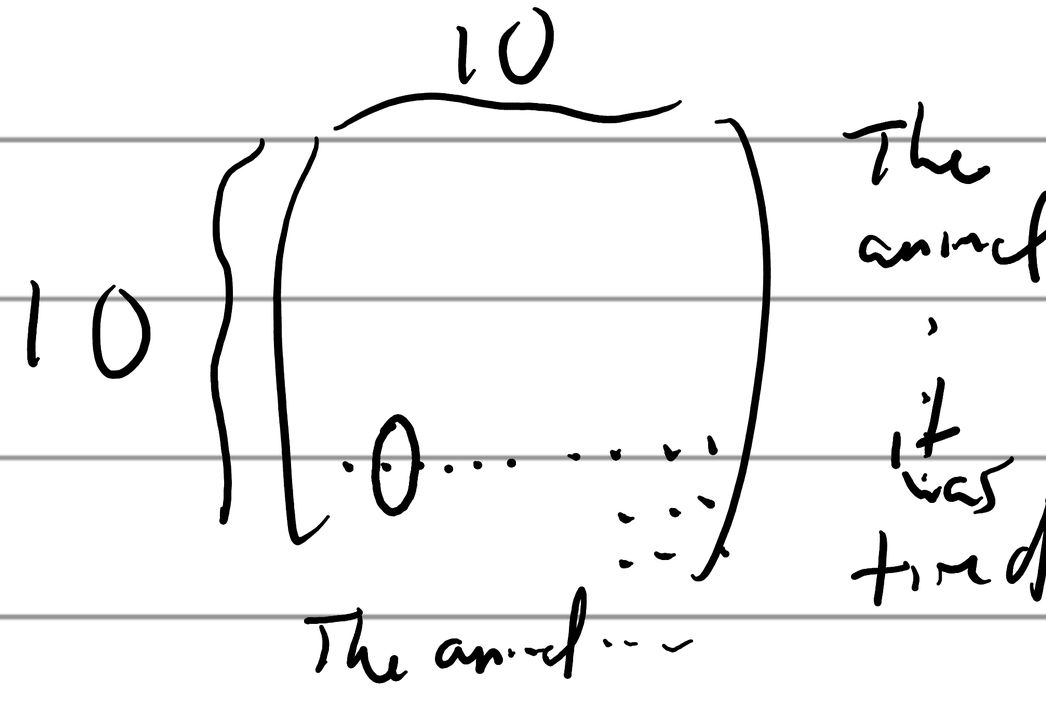
- Key idea: (self) attention mechanism
- Key idea: learning embeddings
  → learn relations btw elements of sequence
  relations expressed by dot product in embedding space

NLP example:

"The animal didn't cross the street because it was tired."

- What does "it" refer to?
  Clearly "animal" not "street"!
- Earlier NLP models had trouble w/ this (trouble learning long-term relations)

- Attention matrix



each row list of probs sum to 1

$p_{i1}, \dots p_{i10}$

$\sum p_i = 1$

→ Maybe v.t p_{72} and p_{77} large, others small

The Attention Mechanism (Bahdanau 2014) (Vaswani et al 2017)

Want to map input sequence to embedding space

$R^d \longrightarrow R^{d'}$

so that relations are captured by dot product

$y_i \cdot z_j \qquad y, z \in R^{d'}$

want probabilities, use softmax

$p_{ij} = a(y_i \cdot z_j)$ "attention matrix"

if y, z come from same input → "self-attention"

→ Simplest map: linear

$X \longrightarrow X W^Q = Q$ "query" $\leftarrow d \times d'$ matrix

$X \longrightarrow X W^K = K$ "key"

We can apply attention matrix

to input sequence to produce a new sequence

$$\begin{pmatrix} \sum_{i=1}^{N} \rho_{1i} \, \vec{X}_i \\ \vdots \\ \sum_{i=1}^{N} \rho_{Ni} \, \vec{X}_i \end{pmatrix}$$

if we work in embedding space

can learn another map

$$X \rightarrow X W^V = V \quad \text{"values"}$$

$\Rightarrow$ Transformer architecture

sequence $\longrightarrow$ sequence mapping

Recap:

Given sequence of vectors $\in \mathbb{R}^d$

$$X = \begin{pmatrix} \vec{x}_1 \text{ —} \\ \vec{x}_2 \text{ —} \\ \vdots \\ \vec{x}_N \text{ —} \end{pmatrix} \in \mathbb{R}^{N \times d}$$

$\underbrace{\qquad\qquad}_{d}$

Learn 3 maps to embedding space $\mathbb{R}^{d'}$

$$\left.\begin{array}{ll} \text{Query} & Q = X \cdot W^Q \\ \text{Key} & K = X \cdot W^K \\ \text{Value} & V = X \cdot W^V \end{array}\right\} \; d \times d' \text{ matrices (learned weights)}$$

self attn matrix:

$$\boxed{P = a(Q \cdot K^T)} \qquad N \times N \text{ matrix}$$

Transformer output:

$$\boxed{X \rightarrow Y = P \cdot V} \in \mathbb{R}^{N \times d'}$$

★ Note: permutation equivariant!

$$X = \begin{pmatrix} \vec{x}_1 \\ \vdots \\ \vec{x}_N \end{pmatrix} \in \mathbb{R}^d \longrightarrow Y = \begin{pmatrix} \vec{y}_1 \\ \vdots \\ \vec{y}_N \end{pmatrix} \in \mathbb{R}^{d'}$$

interchange $\vec{x}_i \leftrightarrow \vec{x}_j \iff$ interchange $\vec{y}_i \leftrightarrow \vec{y}_j$

can check w/ indices

$$Y_{i\alpha} = a\left(X_{ia} W^Q_{a\beta} (W^K)^T_{\beta b} (X^T)_{bj} \right) X_{jc} W^V_{c\alpha}$$

perm acts here

sum perm inv't

In fact can argue for this architecture as simplest non-linear permutation-equiv. seq → seq map

"Multi-headed Attn"

in example "it" also relates to "tired"

how to capture that?

could overload style attn matrix

but more powerful to learn another one!

- $W^{Q,K,V}_{r=1,\cdots N_h}$ analogous to multiple filters in CNN!

- each gives $N \times d'$ embedded sequence Z_r

- concatenate $(Z_1 \cdots Z_{N_h})$ $N \times (d' \cdot N_h)$

- fuse it all together w/ final matrices $W^{Q,K,V}$ $(d' \cdot N_h) \times d''$

$$Z_{mt} = Z^{Q,K,V} W^{Q,K,V}$$

"Positional Encoding"

- in NLP & other sequences, position does carry info!

- add back in positional info w/ positional encoding

$$\vec{x}_i \rightarrow \vec{x}_i + \vec{pos}(i)$$

common examples

$\sin(\omega_a i)$

$\cos(\omega_a i)$

- transformer still permutation invariant wrt original input sequence

---

"Tokenization"

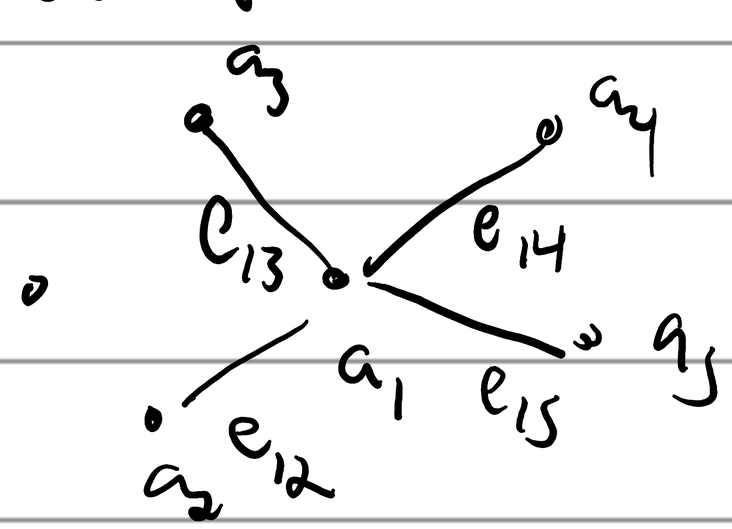
In NLP need to map words to vectors ("tokens")

many tokenization algorithms

often using ML - pretrain task

Transformers as GNNs

Recall GNN



MLPs

$a_i' = \phi^{node}(a_i, \rho(e'_{j_1 i}, \ldots, e'_{j_n i}))$

$e_{ij}' = \phi^{edge}(a_i, a_j, e_{ij})$

Transformer

$X_i' = \sum_j \rho_{ij} X_j$ ← this is a_j; this is like e_{ij}

↳ $\mathrm{attn}(X_i, X_j)$

~ Fully connected graph

nodes are X_i

edges are $\mathrm{Attn}(X_i, X_j)$ } multiply not usually considered vis a vis MLPs

So Transformers still go beyond standard GNNs

Summary:

single self attn:

$$Z = \text{softmax}(QK^T) \cdot V \qquad Q = XW^Q, \; K = XW^K, \; V = XW^V$$

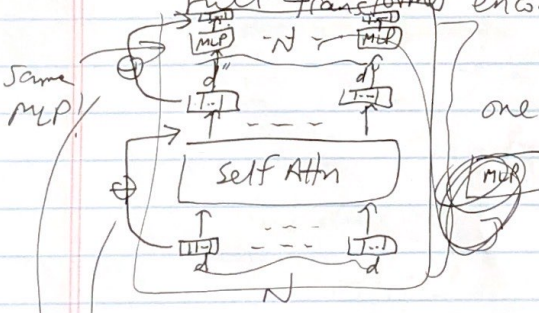
multi-head attn: $(Z_1 \cdots Z_h) W^O = Z$

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

- "MLP w/ input-dependent weights"

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

Full transformer encoder architecture



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 interrelations

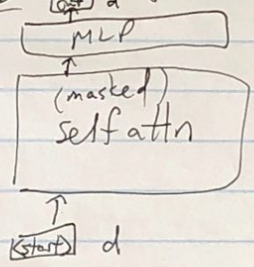
Transformer Decoder

To generate ~~[illegible]~~, need the decoder architecture

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

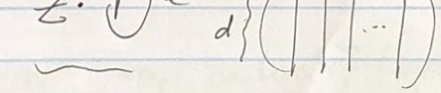
↓

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)

$z^T \cdot$  $\leftarrow d$

N_{vocab} dim'l vector

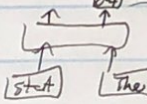
↳ interpret as score for each word in vocab

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

↓

e.g. "The"

Then feed back into decoder



use 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 masky: don't want attention scores to depend on future words

$$\text{softmax}\left(\text{mask}: \begin{matrix} & \text{(start)} & \text{I} & \text{am} & \text{fine} \\ \text{(start)} & \cdot & -\infty & -\infty & -\infty \\ \text{I} & \cdot & \cdot & -\infty & -\infty \\ \text{am} & \cdot & \cdot & \cdot & -\infty \\ \text{fine} & \cdot & \cdot & \cdot & \cdot \end{matrix} \cdot QK^T\right)$$

- This is an example of "autoregressive model"

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

  - GPT is an example of this decoder only model

    Generative Pre-trained Transformer OpenAI (2018)

    GPT → BERT → GPT2 → ...

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

  - Can also give "prompt" → just an initial sequence instead of <start>

  - prompt can be diff language → translation!

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

  - "masked language modeling"
  - "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

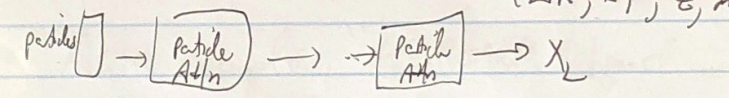
Qu, Li, Qian 2202.03772 "Particle Transformer for Jet Tagging"

- 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 × 10M each

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

- particle features & pairwise "interaction" features $N \times N \times d_x$ $(\Delta R, k_T, z, m^2)$



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

↳ add interaction info to attn
add physics to learned attn

- "class attn" — (don't completely understand this)

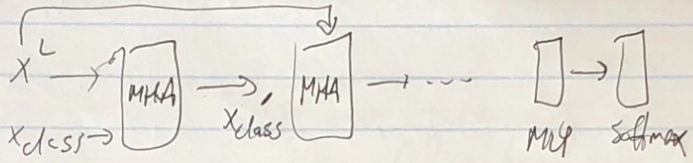
extract class. for output — standard trick in vision transformers

randomly initialized token

accumulates 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

- ParT achieved SOTA results on all jet tagging tasks!
- Also interesting: pretrain + fine train >> train from scratch
  on smaller dataset — on smaller dataset
- Also transformer benefited more from pretrain than GNN!

example of "foundation model"

- Astro example: "ASTROMER transf. based embddg 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 ParT
- positional encoding (not perm equiv)

→ masked light curve modeling
encoder-decoder

- Data R-band light curves of MACHO survey (Galactic Bulge & LMC)
  for pretrain — 1.5M light curves
- 20k labeled variable stars for f.t. & eval
  Also OGLE-III 360k labeled; ATLAS 422k