Summary:

- Single self-attention:
  \[ Z = \text{softmax}(QK^T)V \]
  \[ Q = XW^Q \]
  \[ K = XW^K \]
  \[ V = XW^V \]

- Multi-head attention:
  \[ (Q_1, Q_2, \ldots, Q_h) \cdot W^Q = Z \]
  \[ (K_1, K_2, \ldots, K_h) \cdot W^K = V \]
  \[ (V_1, V_2, \ldots, V_h) \cdot W^V = \]

- Can check: perm equivariant! Interchanging order of rows of \( X \) will result in same output. Output rows interchanged.

- "MLP with input-dependent weights"
  \[ Z \sim (XW^Q) (W^K_X) \cdot XW^V \]

  This is like \( X \)-dependent weight

---

Full transformer encoder architecture

- Same MLP
- One encoder block
- Self-attention
- \( N \rightarrow N \)

- Can stack!

The encoder produces a transformed embedding that takes its input and its relations into account.
**Transformer Decoder**

To generate words, need the decoder architecture. (original transformer, encoder-decoder pair, add encoder att layer...)

Same structure as encoder but with "future masking".

- MLP
- (masked) self-att

** encoders useful for learning embeddings**

- classification/categorization
- sentiment analysis

**BERT example**

- Bidirectional encoder
- Representation for Transformers
- Google (2018)

Output is vector of same dimension as original embedding, interpreted as score for each word in vocab. Pick word w/ highest score or top-k etc.

E.g., "The"

Then feed back into decoder:

- "is"
- "thing"
- use embedding of last word (which knows about past or proceed sequence) to predict next word.
• AD Future mask: don't want attention scores to depend on future words

\[ \text{softmax(mask: I am from QKT)} \]

This is an example of "auto-regressive model"

\[ P(x_1, ..., x_n) = P(x_1) P(x_2|x_1) P(x_3|x_1, x_2) ... P(x_n|x_1, ..., x_{n-1}) \]

• GPT is an example of this decoder-only model

Generative Pre-trained Transformer OpenAI (2018)

GPT → BERT → GPT2 → ...

New GPT—fairly (decoder only) are state of the art!

• Can also give "prompt" → just an initial sequence instead of \langle \text{start} \rangle

• "prompt can codefiff language → translation!"

• Both BERT & GPT use self-training to learn attm model

"masked language model" ← "next word prediction"

→ using huge amount of data & great model

due to "downstream" (e.g., translation, classification)

"foundation model" "backbone" tasks
Examples of applications of transformers

Qu, Li, Qin. 2020. 03772 "Particle Transformer for Jet Tagging"

- Introduced permutation equiv. arch. for jet tagging based on transformers
- Also introduced new dataset "JetClass" for training it
  (provok 10M jets e-10 types x 10M each)
  - 4 vec
    - jetID (clus hadron, neutral hadron, e, mu, nu)
    - jetClass (is not true, reco)
    - displacement
  - particle features & pairwise interaction features N\times h
    \[
    \text{particle} \rightarrow \left[ \text{particle} \right] \rightarrow \left[ \text{pair} \right] \rightarrow X_2
    \]
    \[
    \text{softmax} (A_{ij} U) V
    \]
    - add interaction with other
    - add physics to learn with
  - "class att" (don’t completely understand this)
  - extract classifer output — standard trick in vision transformer
    - randomly initialized token
    - accumulates info from event, from attention to other tokens
  \[
  \begin{align*}
  Q &= W_0 \text{class} \\
  K &= W_k \text{class} \times w \\
  V &= W_v \text{class} \times w
  \end{align*}
  \]
Class atts (from vision transformer bit)

\[
\begin{align*}
\text{X} &\rightarrow \text{MAA} \rightarrow \text{MAA} \rightarrow \text{softmax} \\
\text{x-class} &\rightarrow \text{MAA} \rightarrow \text{softmax}
\end{align*}
\]

starts off random but accumulates info

- Part achieved SOTA results on all jet fusing tasks!
- Also, transformer benefited more from pretraining than GNN!

- Astro example: "ASTROMER transform-based encoder for rep of light curves" (Cortese-Oliva et al. 2020. 05. 01. 07)
  - Self-supervised, like NLP etc. unlike Part
  - Positional encoding (not perm equiv)
  - Masked light curve encoder-decoder
  - Data: R-band light curves of MACHO survey (Gaia, ATLAS, LMC)
  - 50k labeled variable stars for fit & eval
  - 500 OGLE II, 36k labeled ATLAS 42k
Although transformers have "taken" over, poor approaches to sequence modeling still useful to have in toolbox (as in ASTROMAP example) - RNN, LSTM, GRU.

Ref: Stanford CS 224n "cheat sheet" for RNNs

Brief review of RNN, LSTM, GRU architectures:

- Very Sequential, not at all parallel.
- Not feed forward (like FF with # layers)

```
\text{output} \rightarrow y_0, y_1, y_2, \ldots
\text{Sequence} \ x_0, x_1, x_2, \ldots
```

\text{"Hidden state" depends on prev hidden state}
\begin{cases}
  a_t = g_t(W_{ha} a_{t-1} + W_{hx} x_t + b_a) \quad \text{current input} \\
  y_t = g_t(W_{ya} a_t + b_y) \quad \text{depends on current hidden state}
\end{cases}

Repeat for each element of sequence

produces transformed sequence + hidden state that encodes full sequence
Can also use as $1 \rightarrow \text{many}$

$X_1 \rightarrow Y_1 \rightarrow Y_2 \rightarrow \ldots \quad \text{(music, text, sen)}$

$\text{many} \rightarrow 1 \quad \text{just keep } Y_1 \quad \text{classification}$

$\text{many} \rightarrow \text{many} \quad \text{name entity recog (classify each token according to type)}$

$\text{esp named entities (person, org, ..)}$

$\text{many} \rightarrow \text{many} \quad \text{at translation}$

$x_i \rightarrow x_{i+1}$

$\text{also: saves time, size, special}$

- Varnied RNNs struggle w/ vanishing/explosion gradients — long sequence input makes it bad

- introduce “gated” RNNs (analog of residual/layer connections)

Two popular examples: Gated Recurrent Unit (GRU)

Long Short-Term Memory (LSTM)

GRU LSTM (special case, LSTM more general)

Hidden + cell state

Local + short-term info

Long global info + long-term