# Lecture 14: Transformers

CS4787/5777 — Principles of Large-Scale ML Systems

Last time, we talked about a bunch of variants of neural networks.

- *Residual neural networks* include feedback connections in which the outputs of the model are fed back into itself.
- *Convolutional neural networks* restrict some of the linear transformations $W_l$ to be members of some subset of linear transformations, typically convolutions with some filter.
- *Recurrent neural networks* (RNN) repeat the same layers to process a sequence.

For sequence modeling, the classic way to do it was with RNNs.

| Warmup: up to a big-$\mathcal{O}$ analysis, how many flops are needed for this simple MLP-based RNN? |
|---|
| |

**Problems with RNN:**

- vanishing gradient
- exploding gradient
- long range dependencies
- difficult to compose with *parallelism*

**A dumb fix.**

- For each input token $i$, embed it to a vector $x_i$
  - by looking up in an embedding matrix
  - how large does this embedding matrix need to be?
- Pass $x_i$ into a multi-layer perceptron (MLP) yielding $y_i = \mathrm{MLP}(x_i; \mathcal{W})$, where $\mathcal{W}$ denotes all the weights of the MLP.
- Somehow aggregate all the outputs $y_i$, e.g. by averaging them $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$
- Then pass the result $\bar{y}$ into some linear model, e.g. multinomial logistic regression

| Up to a big-$\mathcal{O}$ analysis, how many flops are needed for this simple MLP-based model? |
|---|
| |

Does this approach "fix" the problems with RNN? What is the problem with this approach?

**Transformers.** Designed to process sequential data, but can generalize to any sort of structured data. Fixes the "problems" with RNNs by no longer unrolling the sequence length as depth of the network.

Represents an example as a matrix in $\mathbb{R}^{n \times d}$ where $n$ is the *sequence length* (a.k.a. $n$ "tokens") and $d$ is the *representation dimension*. Most characteristic layer: *attention layer* (more formally, "Scaled Dot-Product Attention"). Given input activation matrices $Q \in \mathbb{R}^{n \times d_k}$ (the "query" matrix), $K \in \mathbb{R}^{n \times d_k}$ (the "key" matrix), and $V \in \mathbb{R}^{n \times d_v}$ (the "value" matrix), the attention layer outputs

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

where this softmax applies along the rows of the matrix (i.e. each row of $\mathrm{softmax}(\cdot)$ sums to 1). You can think of this as a "soft" or "weighted" lookup. This formulation lets every token (every sequence element) look up into every other one: if we want to restrict this, we can use an *attention mask* $M \in \mathbb{R}^{n \times n}$, usually with elements in $\{-\infty, 0\}$, and set

$$\mathrm{MaskedAttention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + M\right)V.$$

This "zeros out" the entries of $\mathrm{softmax}(\cdot)$ for which $M_{ij} = -\infty$.

Multiple attention layers are combined together to form a *multi-head attention layer*. Such a layer with $h$ "heads" takes as input tensors $Q \in \mathbb{R}^{n \times h \times d_k}$, $K \in \mathbb{R}^{n \times h \times d_k}$, and $V \in \mathbb{R}^{n \times h \times d_v}$, and outputs a tensor of size $(n \times h \times d_v)$ such that

$$\mathrm{MultiHeadAttention}(Q, K, V)_{:,i,:} = \mathrm{MaskedAttention}(Q_{:,i,:}, K_{:,i,:}, V_{:,i,:});$$

that is, it's just $h$ attention layers running in parallel along the head dimension.

A typical multi-head attention block with representation dimension $d$ and number of heads $h$ (where $h$ evenly divides $d$) has $d_k = d_v = d/h$ and is parameterized by four matrices: $W_K \in \mathbb{R}^{d \times d}$, $W_Q \in \mathbb{R}^{d \times d}$, $W_V \in \mathbb{R}^{d \times d}$ and $W_O \in \mathbb{R}^{d \times d}$. Given input $X \in \mathbb{R}^{n \times d}$, it outputs

$$\mathrm{MultiHeadAttention}(XW_Q^T, XW_K^T, XW_V^T)W_O^T$$

where here we reshape $\mathrm{MultiHeadAttention}$ to operate on matrices like $Q \in \mathbb{R}^{n \times hd_k}$ rather than on tensors.

| Let's draw a diagram of a transformer block. Up to a big-$\mathcal{O}$ analysis, how many flops are needed? |
|---|
| |

How does this cost compare to the RNN?

**DEMO.**