Transformer Circuit Exercises

A Latex writeup of transformer-circuits.pub/2021/exercises

August 16, 2024

Exercises

Warm Up

- Describe the transformer architecture at a high level.
- Describe how an individual attention head works in detail, in terms of the matrices W_Q, W_K, W_V, and W_{out}. (The equations and code for an attention head are often written for all attention heads in a layer concatenated together at once. This implementation is more computationally efficient, but harder to reason about, so we'd like to describe a single attention head.)
- Attention heads move information from a subspace of the residual stream of one token to a different subspace in the residual stream of another. Which matrix controls the subspace that gets read, and which matrix controls the subspace written to? What does their product mean?
- Which tokens an attention head attends to is controlled by only two of the four matrices that define an attention head. Which two matrices are these?
- Attention heads can be written in terms of two matrices instead of four, $W_Q^T \cdot W_K$ and $W_{out} \cdot W_V$. In the previous two questions, you gave interpretations to these matrices. Now write out an attention head with only reference to them.
  - What is the rank of these matrices?
- You'd like to understand whether an attention head is reading in the output of a previous attention head. What does $W_V^2 \cdot W_{out}^1$ tell you about this? What do the singular values tell you?

Exercise 1 - Building a Simple Virtual Attention Head

Small transformers often have multiple attention heads which look at the previous token, but no attention heads which look at the token two previous. In this exercise, we'll see how two previous token heads can implement a small "virtual attention head" looking two tokens behind, without sacrificing a full attention head to the purpose.

Let's consider two attention heads, head 1 and head 2, which both attend to the previous token. Head 1 is in the first layer, head 2 is in the second layer. To make it easy to write out explicit matrices, we'll have the k, q, and v vectors of both heads be 4 dimensions and the residual stream be 16 dimensions.

(a) Write down W_V^1 and W_{out}^1 for head 1, such that the head copies dimensions 0-3 of its input to 8-11 in its output.

(b) Write down W_V^2 and W_{out}^2 for head 2, such that it copies 3 more dimensions of the previous token, and one dimension from two tokens ago (using a dimension written to by the previous head).

(c) Expand out $W_{net}^1 = W_{out}^1 \cdot W_V^1$ and $W_{net}^2 = W_{out}^2 \cdot W_V^2$. What do these matrices tell you?

(d) Expand out the following matrices: Two token copy: $W_{net}^2 \cdot W_{net}^1$. One token copy: $W_{net}^2 \cdot \mathrm{Id} + \mathrm{Id} \cdot W_{net}^1$.

- Observation: When we think of an attention head normally, they need to dedicate all their capacity to one task. In this case, the two heads dedicated 7/8ths of their capacity to one task and 1/8th to another.

Exercise 2 - Copying Text with an Induction Head (Pointer Arithmetic Version)

The simplest kind of in-context meta-learning that neural networks do is increasing the probability of sequences they've seen before in this context. This is done with an "induction head" that looks at what followed after last time we saw a token.

There are at least two algorithms for implementing induction heads. In this exercise, you'll build up the "pointer arithmetic" algorithm by hand.

(a) Let $u_0^{cont}, u_1^{cont}, \ldots, u_n^{cont}$ be the principal components of the content embedding. Write W_Q and W_K for an attention head (with 4 dimensional queries and keys) selecting tokens with similar content to the present token, including the present token itself.

(b) Let $u_0^{\cos}, u_0^{\sin}, u_1^{\cos}, u_1^{\sin}, \ldots$ be a basis describing the position embedding in terms of vectors that code for the sine and cosine embedding of the token position (e.g., $\lambda(\cos(\alpha_0 n_{tok}))$ with descending magnitudes. Write W_Q and W_K for an attention head (with 4 dimensional queries and keys) that self-selects the present token position.

(c) Using the position embedding basis described in (b), write W_Q and W_K for an attention head (with 4 dimensional queries and keys) that self-selects the *previous* token position. Hint: think about a 2D rotation matrix.

(d) Write W_Q and W_K for an attention head (with 8 dimensional queries and keys) selecting tokens with similar content to the present token, but disprefers attending to itself. Hint: refer to (b) and use extra 4 dimensions for keys and queries.

(e) Write down W_V and W_{out} for the attention head you described in (d), such that it extracts the largest 8 dimensions of the position embedding from the token it attends to, and writes them to the vectors $v_0, v_1, \ldots$.

(f) Write W_Q and W_K for an attention head which attends to the token after a previous copy of the present token. Hint: use the outputs of the head from (e) and the strategy you used in (c).

Exercise 3 - Copying Text with an Induction Head (Previous Token K-Composition Version)

Some positional encoding mechanisms, such as rotary attention, don't expose positional information to the W_V matmul. Transformers trained with these mechanisms can't use the strategy from (e) and (f) in the previous exercise to manipulate positional encoding vectors.

For these transformers, we've seen an alternate mechanism, where the first head copies information about the preceding token into a subspace, and the second head uses that subspace to construct queries and keys. Assuming the same positional encoding mechanism as above, write down $W_Q^1, W_K^1, W_V^1, W_O^1$ and W_Q^2 and W_K^2 for a pair of attention heads implementing this algorithm.