1. Introduction

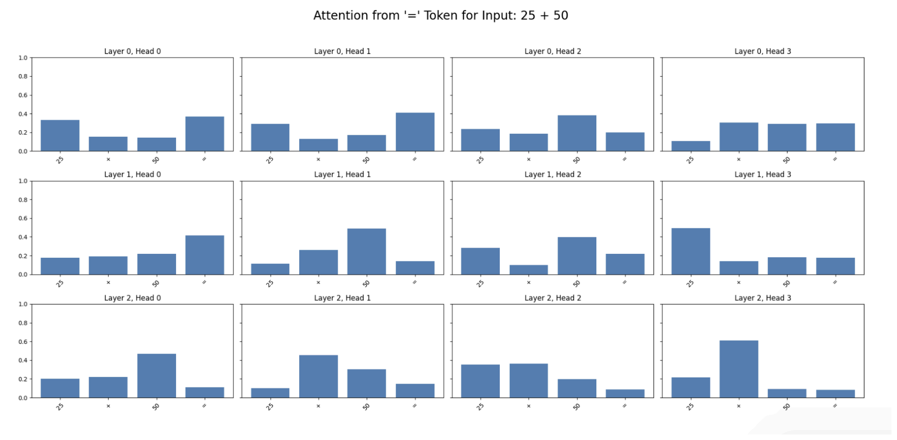
The goal of this project was to help me build proficiency in mechanistic interpretability through a foundational project reverse engineering the algorithm learned by a neural network on a simple task, that also gives cool insights on what is really going on within transformer models. I trained a 3-layer transformer model to perform modular addition (specifically, two-number modular addition mod 113. After 50 epochs, the model successfully converged and achieved 100% accuracy on a held-out test set. This report analyzes the model's attention patterns to understand the internal algorithm it learned.

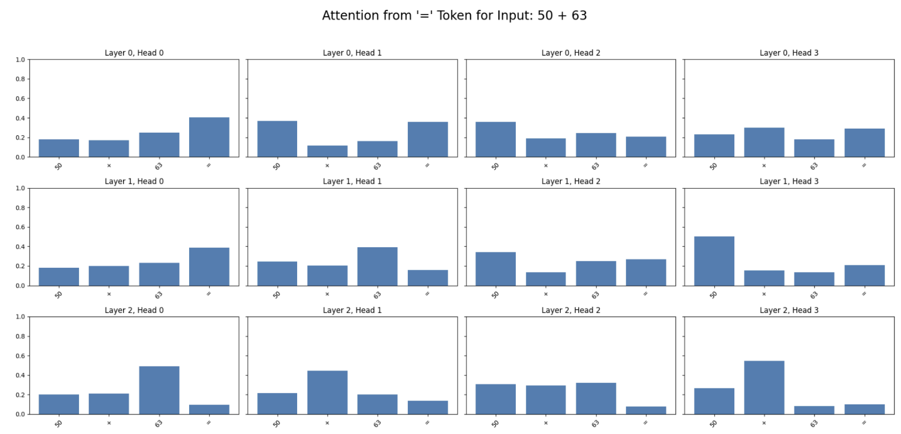
2. Methodology

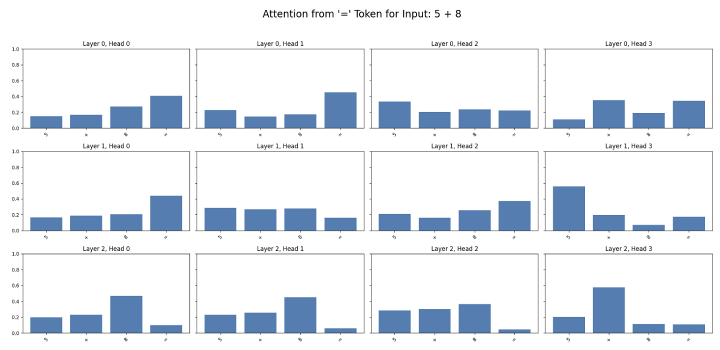
I analyzed the model using the transformer_lens library. The model takes in the tokens “a”, “+,” “b”, and “=,” where a and b are some integers in [0, 113). I focused on the attention patterns originating from the final token (=). I ran the model on several key input pairs (like 5+8, 90+90, and 50+0) and generated summary plots of attention for all 12 heads on matplotlib.

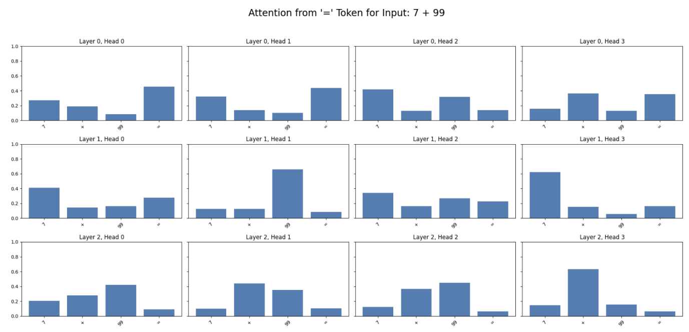
3. Plots:

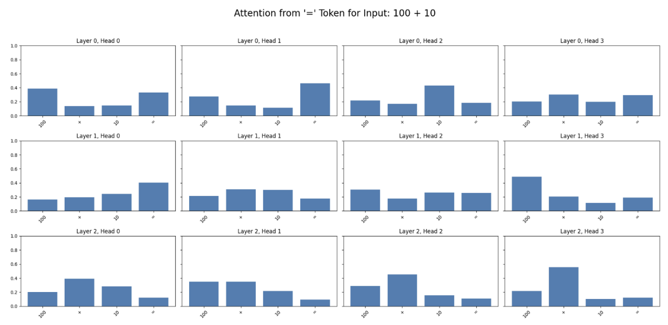
The plots generated are shown here:

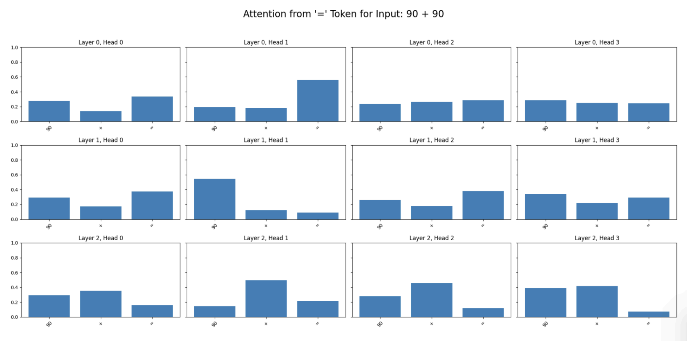












4: Key Findings: How My Transformer Learned to Add Numbers

- Each attention head has found its own niche, divvying up the work. The plots allow us to observe broad categories for the attention heads — heads that focus primarily on the first integer input a (such as layer 1 head 3 and layer 2 head 3), heads that focus primarily on the second integer input b (such as layer 0 head 3 and layer 2's heads 1 and 2), heads that spread across all the inputs (these are very common in layer 0, especially heads 2 and 3 in layer 0), and "self-heads" that mainly attend to the equals token "=."

- Some unsurprising results, but cool to observe: a majority of attention is focused on the tokens corresponding to the input numbers a and b, especially in layers 1 and 2 where the plots show sharp spikes in attention. Similarly, very little attention is focused on the token corresponding to the plus operator, demonstrating the model has learned that only addition is being performed in the training data and that only the two inputs a and b primarily matter in determining the output.
- Layers show a clear progression: layer 0's attention heads show more broadly distributed and noisy attention, as the model gathers information. In many heads in layer 0 especially, there is clearly strong self-attention to the "=" token itself, while the "+" token is largely ignored. Layer 1's attention and layer 2's attention is clearly more sharply focused and specialized, as the model no longer needs to gather information and instead has heads that often pay strong attention to one or two input tokens specifically. Layer 1 and layer 2 also show much more attention being focused on the "+" token than layer 0.
- To summarize: the progression the model has learned involves doing a broad pass in layer 0 to locate input tokens, and then using specialized heads to route the necessary information in layers 1 and 2: certain heads have the task of finding the input integers a or b specifically and passing their values up. Once a and b are isolated, they can be fed into the Multi-Layer Perceptron that has learned to perform the computation.