Report

Problem 1: Attention pattern visualizations

Let's look at this cross-attention layer for the decoder and compare what each of the heads of the decoder does.

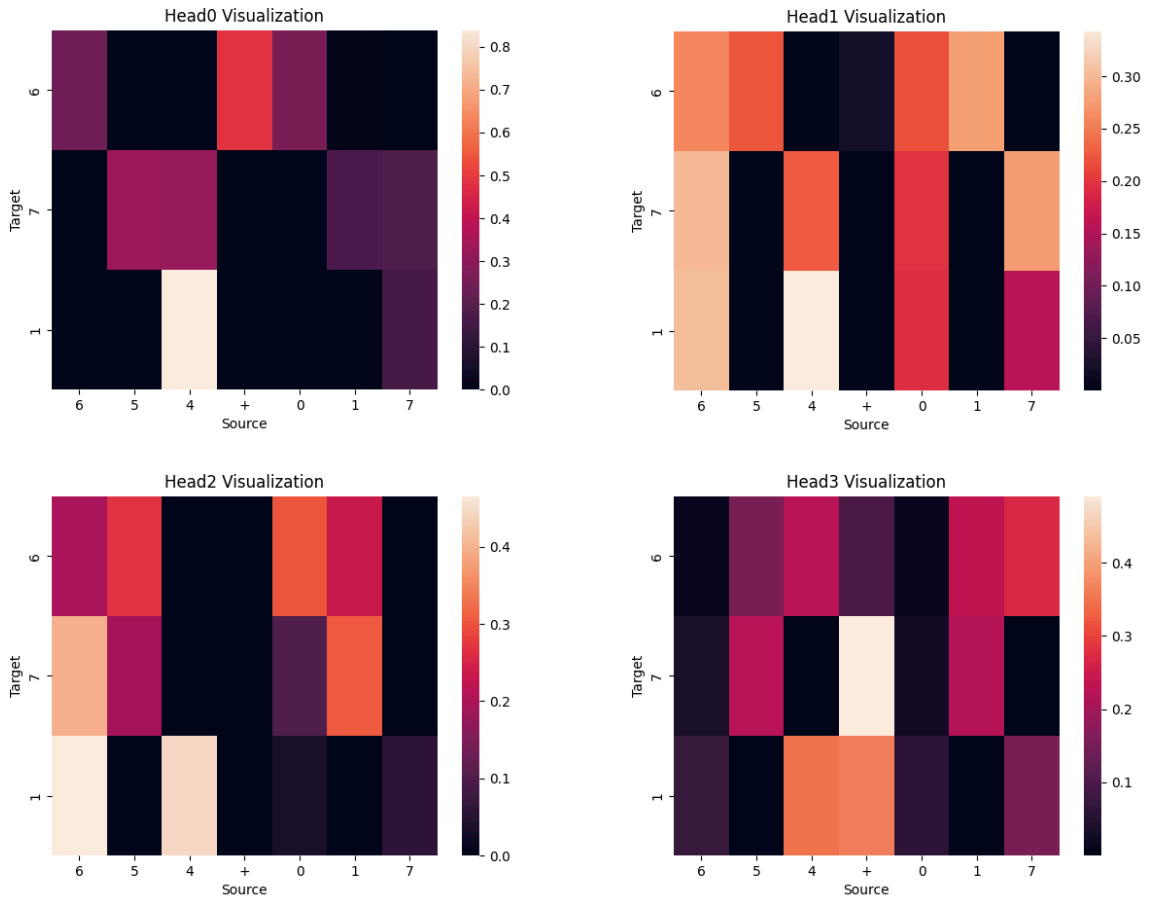

Figure 1: Comparison of 4 Different Heads

Figure 1 shows the focus of heads inside the decoder. The color on the heatmap represents the amount of attention each digit receives, and the lighter the color, the more attention in this case. Head 0 focuses most on the rightmost digits and how they affect the tens digit. It handles carry propagation somewhat, but is more focused on the effect of the addition on the corresponding digit. Head 1 has very diverse attention on almost all digits, with an interest in how the hundreds digit of the source affects the target digits. Head 2 is similar to head 0 in the sense that the diagonal patterns in the attention heatmap demonstrate it understands how the digits in the source affect the target. It also seems to be more interested in the first number in the addition,

presumably due to the fact that the first number is much larger. Head 3 is mainly focused on the + sign operator, and there is more attention towards higher-order digits. It is likely that Head 3 is also used for understanding carry propagation.

Head Ablation Study:
The result of the head ablation study is shown in Figure 2.

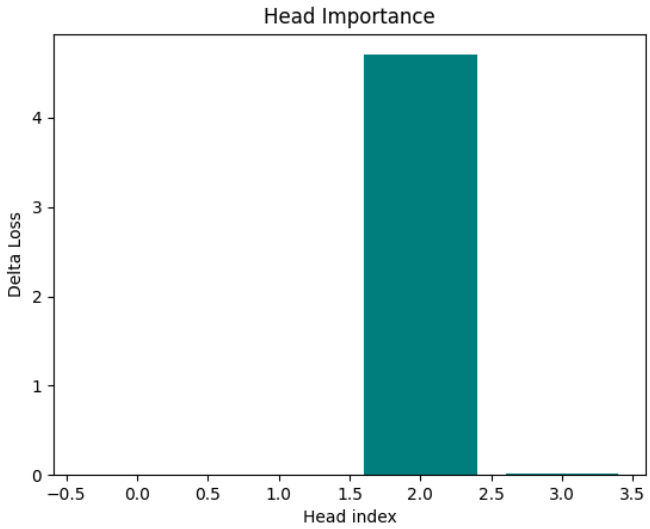

Figure 2: Head Ablation Study

From the result, it is clear that head 2 is the most important head by a large margin. Looking at Figure 1, it makes sense as Head 2 captures the main addition logic. Head 3 is the second most important head, and that is the head that presumably does carry propagation. Both Head 0 and 1 seem not to have much effect on the model, and their heatmaps also reflect that the learning from those attentions should be limited.

Percentage of heads that can be pruned with minimal accuracy loss:
Based on the head ablation study, if we choose to only remove Heads 0 and 1 from the example, then the accuracy loss would be minimal. That can be achieved by setting a threshold that head removal that results in $\Delta\text{Loss} < 1\%$ are removable, and applying this threshold to all heads.

Discussion on Carry Propagation:
Figure 1 shows that Head 0 and Head 3 most likely specialize in carry propagation. Specifically, if there are attention weights on a lower-order source digit and a higher-order target digit, then it is “carrying” the result over, which is how humans do it. Since each head learns a very specific sub-function in a model like this, without Head 3 or 0, the effect of carry propagation might be ignored completely, therefore highlighting the importance of the heads.

Problem 2: Extrapolation Curve

Figure 3 is the extrapolation curve comparing the 3 encoding methods. The blue line is the sinusoidal method, and the orange and green lines are the learned and the none method, respectively.

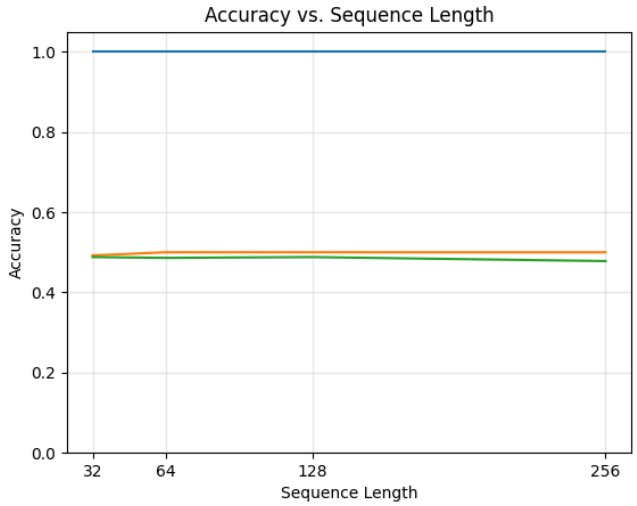

Figure 3: Extrapolation Curve of the 3 Encoding Methods

The results are exactly as expected. The sinusoidal curve was able to generalize its learning to sequence lengths that are higher than the training length, whereas the learned and none curve stayed at around 50% accuracy. Since we know that the None curve is the baseline model that just random guesses, the Learned curve does slightly better than that, so some information is preserved from the embeddings.

Accuracy at lengths 32, 64, 128, 256:

Table 1 shows the result of the numerical results at different lengths for the 3 encoding methods.

	Sinusoidal	Learned	None
Length 32	1.0	0.492	0.488
Length 64	1.0	0.5	0.486
Length 128	1.0	0.5	0.488
Length 256	1.0	0.5	0.478

Table 1: Accuracy Table of Length vs. Encoding Methods

Mathematical Explanation:
Intuitively speaking, the sinosoidal model utlizes a formula for the positional encoding, shown below:

PE(pos, 2i) = sin(pos / 10000^(2i/d_model))
PE(pos, 2i+1) = cos(pos / 10000^(2i/d_model))

There are no limits on length on the actual formula, so if the mode learned the pattern of the sequence(strictly increasing in this case), it can then apply the learned pattern to index that didnt exist during training easily, thus the extrapolation.
Alternatively, the learned model are learning only from examples it is able to see during training, and adjust the embedding at the indexes accordingly to the sequence. So when a longer sequence is inputted, it will not have any information beyond what it has seen in training, hence the random guessing.

Positional Embedding Visualization of Learned Encoding:
Figure 4 is the heatmap for the first 64 embeddings of the learned encoding method, and the colors represents values ranging from -0.1 to 0.1

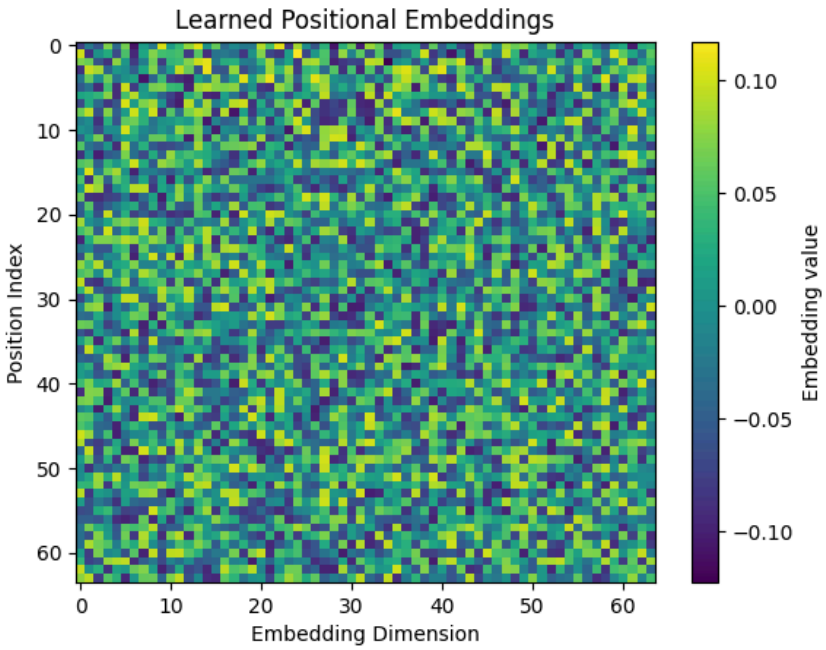

Figure 4: Learned Positional Embeddings for a Length 64 Sequence

From the figure, the embeddings are highly random with no patterns or transitions between the embeddings, so it is apparent that the model was not able to learn any absolute positional

information. This also explains why the learned model forms random guesses that has 50% accuracy when applied to a sequence length longer than what it learned.