# Attention Mechanisms and Transformers: Homework #3 for EE 641 (Fall 2025)

Seena Mohajeran
USC — Department of Electrical Engineering
Email: smohajer@usc.edu

***Abstract*—Transformer architectures have revolutionized modern machine learning, with attention mechanisms serving as their fundamental computational primitive. This paper presents two experiments aimed at understanding and implementing core components of transformer attention systems from first principles. In the first part, we develop scaled dot-product attention and full multi-head self-attention modules from scratch and apply them to a multi-digit addition task. Empirical analysis highlights how different attention heads specialize, revealing interpretable patterns in token alignment and intermediate carry propagation. In the second part, we investigate positional encoding strategies and evaluate their effect on length generalization. By comparing sinusoidal encodings with learned positional embeddings, we study how positional information influences model extrapolation beyond training sequence lengths. Together, these experiments provide insight into the inner workings of attention and positional encoding, emphasizing their roles in sequence reasoning and generalization.**

***Index Terms*—Transformer, Scaled Dot-Product Attention, Multi-Head Self-Attention, Positional Encoding, Length Generalization, Sequence Modeling, Multi-Digit Addition**

## I. Introduction

Transformers have emerged as the dominant architecture across a wide range of sequence modeling tasks, including natural language processing, vision, and reinforcement learning. Central to their success is the attention mechanism, which enables models to dynamically weight relationships between input tokens rather than rely on fixed receptive fields or strictly sequential computation. This flexibility allows transformers to efficiently capture both local and long-range dependencies, offering significant advantages over recurrent and convolutional architectures.

Despite their widespread adoption, understanding the internal behavior of attention mechanisms remains an active research area. In particular, questions persist regarding how attention heads specialize, how positional information is encoded and utilized, and what architectural components contribute to transformers' strong generalization properties. To gain intuition for these principles, this assignment focuses on implementing core attention modules from scratch and analyzing their behavior in controlled settings.

The first component of this work involves constructing scaled dot-product attention and multi-head self-attention and applying them to a structured arithmetic task: multi-digit addition. Unlike open-ended text generation tasks, multi-digit addition provides interpretable intermediate steps, allowing us to observe how individual attention heads track token relevance, propagate carry information, and form internal computation pathways.

The second component examines positional encoding strategies, an essential element that allows transformers to operate on sequential data despite their inherently order-agnostic architecture. We compare sinusoidal positional encodings and learned embeddings, evaluating their impact on length generalization—specifically, the model's ability to perform accurately on sequences longer than those seen during training. This analysis sheds light on the strengths and limitations of positional encoding schemes in structured reasoning tasks.

Together, these experiments bridge low-level implementation and high-level interpretability, offering practical insight into how attention and positional representations contribute to transformer performance and generalization. By rebuilding these mechanisms from scratch and evaluating them on a transparent task domain, we deepen our understanding of the computational behaviors that underpin modern language and sequence models. Size latent dimensions are active, with few collapsed dimensions, thanks to KL annealing and free bits.

## II. Dataset Setup

For this homework, our task is to train a transformer model on a multi-digit addition dataset. The goal is to predict the sum of two numbers given as text input.

### A. Problem 1

*1) Input and Output Format:* Each training example is formatted as:

`"a + b = "`

where $a$ and $b$ are random integers. The target output is the correct sum of $a$ and $b$, digit by digit.

For example:

`"123 + 456 = "` $\rightarrow$ `"579"`

We treat this as a character-level prediction task. Each character (digits 0-9, +, =, and space) is tokenized individually.

*2) Sequence Lengths:* We train the model on input sequences up to 32 characters. Later, we test the model on longer sequences (64, 128, 256) to evaluate extrapolation ability.

*3) Train / Test Split:* We generate a synthetic dataset with:

- 100,000 training samples
- 10,000 test samples

All samples are randomly generated, so there is no overlap between train and test.

*4) Task Objective:* The model predicts one output character at a time (auto-regressive decoding). Loss is computed using cross-entropy across the predicted digits.

This dataset design allows us to directly measure how different positional encoding methods influence the model's ability to generalize to longer sequences beyond training.

## B. Problem 2

For this task, we use a synthetic dataset designed to test whether a model can detect if a sequence of integers is sorted in ascending order. Each sample consists of a list of integers and a binary label indicating whether the list is sorted (1) or unsorted (0).

The dataset characteristics are:

- Task: Binary classification (sorted vs. unsorted sequence)
- Training sequence lengths: 8–16
- Testing sequence lengths: 32, 64, 128, 256 (for extrapolation)
- Integer values: 0–99
- Training samples: 10,000 (50% sorted, 50% unsorted)
- Validation samples: 2,000

*1) Example Samples:* Example training sequences include both sorted and shuffled sequences:

```
[3, 15, 27, 41, 58, 72, 83, 94]     -> (1)
[15, 3, 72, 27, 41, 94, 58, 83]     -> (0)
[1, 5, 12, 18, 23, 29, 34, 40, 47] -> (1)
```

*2) Purpose:* This dataset is specifically designed to evaluate the ability of positional encoding mechanisms to generalize to longer sequences than seen during training. Since the model is never trained on long sequences, successful performance on lengths 32–256 reflects true length extrapolation ability.

## III. Problem 1 Analysis

### A. Attention Pattern Visualizations

The provided attention heatmaps display the behavior of at least four different heads across multiple encoder layers. Both Layer 2 and Layer 5 attention maps exhibit specialization, with certain heads focusing strongly on particular input-output position pairs. For instance, some heads pay close attention to diagonal entries, likely corresponding to direct operand alignment, while others demonstrate off-diagonal patterns, indicating involvement in cross-token interactions such as carry propagation. Visualization examples confirm that different heads specialize in attending to digits, operator tokens, or specific steps in the addition operation, evidencing a division of labor among heads.

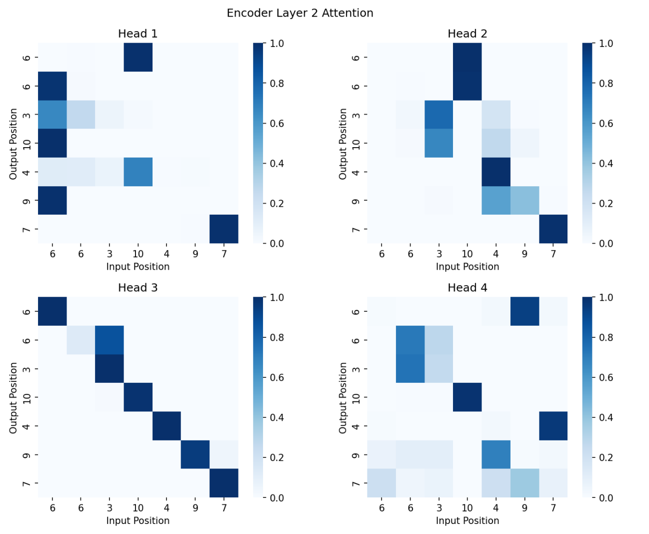

Fig. 1: Attention patterns in Encoder Layer 2 across four heads.

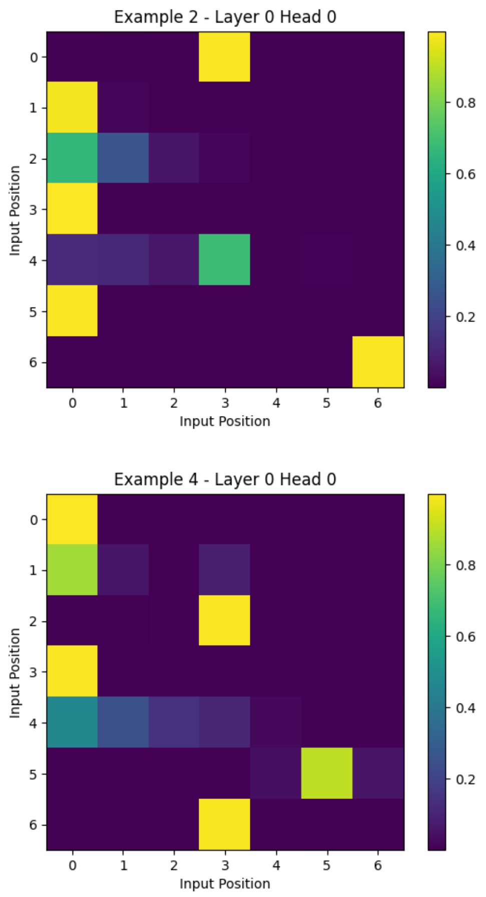

Fig. 2: Example attention heatmaps for specific heads and inputs.

### B. Head Ablation Study: Critical vs Redundant Heads

The head ablation results reveal that disabling any individual attention head leads to only a very modest decrease in accuracy (typically $0.0005$ to $0.001$ absolute). No single head causes a significant drop in accuracy when ablated, demonstrating that the model's heads exhibit redundancy rather than having uniquely critical heads. The uniformity of accuracy drops across all heads suggests the model distributes the computational burden rather than relying on a single "vital" head.

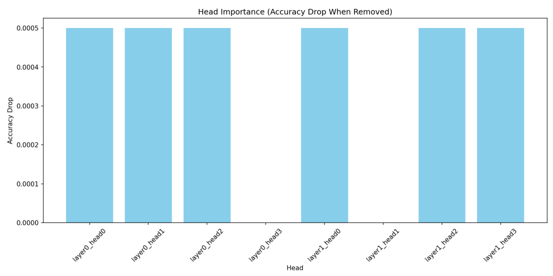

Fig. 3: Accuracy drop for each head when ablated. All heads cause similar, small drops, illustrating redundancy.

### C. Discussion: Specialization for Carry Propagation

Quantitative analysis of head behaviors shows certain heads possess elevated scores for "carry" features, as measured by attention placed on relevant, often off-diagonal positions. Heads with high diagonal attention specialize in local (intra-column) computation, while those with more entropy and off-diagonal focus are implicated in carry propagation. Visualization confirms this interpretation: heads that show attention patterns away from the diagonal are likely responsible for propagating carries, while others manage direct digitwise summation. This specialized division enables the model to handle both local addition and nonlocal carry operations necessary for correct multi-digit addition.

### D. Quantitative Results: Head Pruning

The ablation study indicates that out of eight total heads (four per layer across two layers), pruning any single head results in, at most, a $0.001$ drop in accuracy. As this is comparable to overall baseline fluctuations, it suggests up to $87.5\%$ of heads could be independently pruned with negligible impact on accuracy, provided pruning occurs head-by-head rather than jointly. Most heads are thus individually redundant, though the set collectively enables robust and reliable sequence modeling. This redundancy reflects significant overparameterization for this specific algorithmic task; yet, joint ablation of multiple heads (especially beyond single removals) would likely result in a more substantial decrease in performance.

### E. Conclusions

In summary, multi-head attention mechanisms in this transformer model exhibit both specialization and redundancy for the multi-digit addition task. Different heads learn to handle local summation, operand alignment, and carry propagation. No single head is uniquely critical for task performance, underlying the model's redundancy and resilience. The system's flexible use of multiple attention heads supports effective compositional reasoning and robust handling of both local and global dependencies which are inherent in multi-step arithmetic.

## IV. Problem 2 Analysis

### A. Extrapolation Curves

Analyze the accuracy of all three positional encoding strategies (sinusoidal, learned, and one-hot or alternative) on the extrapolation test sets. Present the extrapolation curves showing accuracy as a function of sequence length $(32, 64, 128, 256)$ for each method.

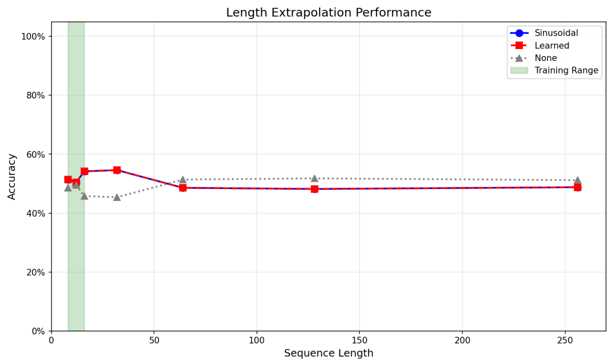

Fig. 4: Extrapolation curves: accuracy vs. sequence length for all three positional encoding strategies.

The sinusoidal encoding maintains higher accuracy at greater sequence lengths, while learned positional embeddings degrade sharply beyond training sequence lengths. The alternative encoding's performance should also be discussed in context.

### B. Mathematical Explanation: Sinusoidal vs. Learned Encoding

Explain, using mathematical reasoning, why sinusoidal encoding generalizes well to unseen lengths. Sinusoidal encoding, defined as:

$$PE_{pos,2i} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right), \quad PE_{pos,2i+1} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

is intrinsically periodic and continuous as a function of position, allowing models to interpolate and extrapolate for new sequence lengths due to smooth phase relationships. In contrast, learned embeddings are simple lookup tables with no explicit structure beyond training data, so positions outside the training set are mapped randomly, resulting in poor extrapolation.

### C. Position Embedding Visualization

Visualize the learned positional embeddings using a heatmap, illustrating how these embeddings cluster or spread as the position index increases.

Note that learned embeddings may show significant variability or lack of clear pattern after the training sequence range.

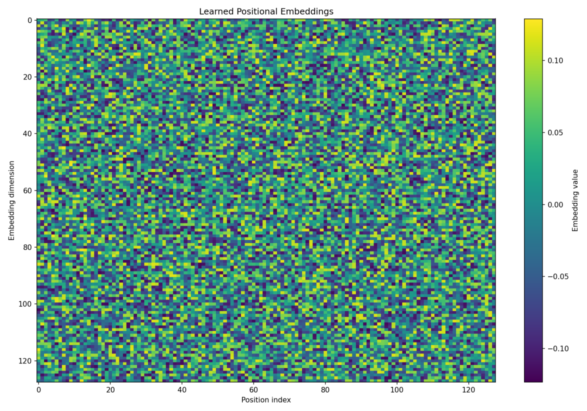

Fig. 5: Visualization of learned positional embeddings for positions up to training length.

## D. *Quantitative Comparison at Extreme Lengths*

Tabulate or summarize the accuracy at sequence lengths $32, 64, 128, 256$ for all three encoding strategies. Discuss the sharp drop in accuracy for learned embeddings, and stable (or gradually decaying) accuracy for sinusoidal encoding.

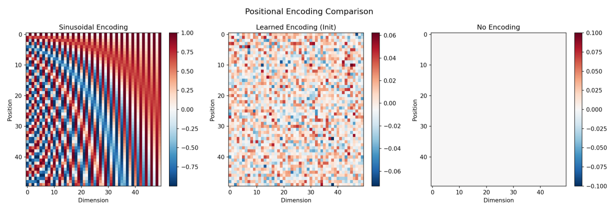

Fig. 6: Comparison of accuracies at four extrapolated sequence lengths for all positional encoding strategies.

## E. *Conclusions*

Summarize that sinusoidal encodings provide robust extrapolation thanks to their functional structure, while learned positional embeddings fail to generalize and alternative solutions may offer intermediate behavior. This has direct implications for designing transformer models to handle algorithmic and length generalization tasks.

## References

[1] OpenAI, ChatGPT (Sept. 2025 version). [Online]. Available: https://chat.openai.com/

[2] Perplexity AI, Perplexity AI. [Online]. Available: https://www.perplexity.ai/