IS584 - Mini Assignment

1. Transformers require PositionalEncoding because, unlike RNNs, they lack inherent mechanisms to capture token order due to their parallelized self-attention structure. To enable sequence modeling, positional embeddings are added to token embeddings. While learned embeddings can suffice for fixed-length inputs, they fail to generalize beyond seen positions, making them unsuitable for long-sequence extrapolation. Sinusoidal encodings, proposed by Vaswani et al. (2017), address this by generating continuous position vectors using sine and cosine functions of varying frequencies, enabling both extrapolation and interpolation. These encodings preserve relative distances and support mathematical operations like translation (shifting) and alignment across positions. This approach allows transformers to model positional relationships without memorization, ensuring robust performance across varying input lengths and maintaining coherence in tasks such as translation or summarization.

2. In the Transformer architecture, Query, Key, and Value vectors are generated by applying learned linear transformations to input embeddings—augmented with positional encodings to retain sequence order. In self-attention, all three come from the same input; in cross-attention, queries come from the decoder and keys/values from the encoder. These projections split the input into lower-dimensional subspaces, allowing each attention head to focus on different aspects of the sequence. For example, with an embedding size of 8 and 4 heads, each head operates on 2 dimensions. The attention mechanism computes dot products between queries and keys, scales them for stability, applies softmax to get attention weights, and uses these to compute a weighted sum of values. Outputs from all heads are concatenated and passed through a final linear layer, enabling the model to capture diverse and complex relationships in parallel.

3. The self-attention mask in the decoder is used to enforce causal self-attention, ensuring that the model does not attend to future tokens during sequence generation. Since the decoder generates tokens one at a time, each position in the sequence should only be influenced by preceding positions. The mask achieves this by assigning large negative values (effectively negative infinity) to positions representing future tokens in the attention score matrix. When passed through the softmax function, these values become zero, preventing any contribution from those future tokens. This mechanism is critical for preserving autoregressive generation, aligning training with inference behavior. It also supports teacher forcing by allowing the model to consider all previous ground-truth tokens while blocking access to those not yet predicted. Without this mask, the model would incorrectly learn from future context, leading to information leakage and unreliable predictions.

4. Split heads are used in the attention mechanism to enable the model to jointly attend to information from multiple representation subspaces. Instead of computing a single attention function over the full embedding space, the multi-head attention mechanism divides the input into smaller subspaces (or "heads") and performs attention separately in each. Each head learns a different set of Query, Key, and Value projections, allowing it to focus on different types of relationships, such as syntactic, semantic, or positional patterns within the sequence. After computing attention independently across all heads, their outputs are concatenated and linearly transformed to produce the final representation. This parallel structure increases the model's expressiveness and allows it to capture diverse dependencies across the input sequence that a single-head attention mechanism might miss. It also supports better generalization by enabling specialization across heads without increasing computational complexity significantly.

5. Self-attention and cross-attention are both fundamental mechanisms in transformer architectures, but they serve different roles based on the source of their inputs. In self-attention, the queries, keys, and values are all derived from the same input sequence. This allows the model to compute dependencies between elements within that sequence, capturing contextual relationships and enabling the representation of each token to reflect the influence of all other tokens. It is used in both the encoder and decoder stacks for modeling intra-sequence dependencies. In contrast, cross-attention appears only in the decoder and is responsible for linking the decoder's output sequence with the encoder's output. Here, the queries come from the decoder, while the keys and values come from the encoder output, allowing the decoder to condition each generated token on the entire input sequence. This mechanism enables inter-sequence alignment, making it essential for tasks

like machine translation and text summarization where output generation is based on an external input. Together, self-attention enables the model to understand context within a sequence, while cross-attention enables the integration of context from a different sequence.

6. In the vanilla Transformer architecture, the cross-attention mask is applied within the decoder's cross-attention sublayer, specifically before the softmax computation in the scaled dot-product attention mechanism. This sublayer is responsible for allowing the decoder to attend to the encoder's outputs. If certain parts of the encoder output should not be attended to, such as padding tokens or masked input regions, a cross-attention mask is applied to the attention scores. This is done by setting the corresponding positions in the attention score matrix to a large negative value (e.g., negative infinity), ensuring that they have zero influence after softmax normalization. This mechanism ensures that the decoder focuses only on meaningful parts of the encoder output while ignoring irrelevant or padded positions.

7. In the cross-attention sublayer of the decoder in a Transformer model, the queries (Q) are generated from the decoder's previous layer output, while the keys (K) and values (V) are derived from the encoder's final output representations. This configuration enables the decoder to condition each of its outputs on the entire input sequence. Specifically, the decoder uses its current internal state to form a query and retrieves relevant information from the encoded source sequence via attention over the encoder's keys and values. This is crucial for tasks such as translation, where the decoder must align its generation with specific parts of the input. Thus, the decoder contributes the queries, while the encoder supplies the keys and values in the cross-attention mechanism.

8. In the decoder of the vanilla Transformer, outputs are shifted to the right during training to enable teacher forcing while preserving the autoregressive nature of sequence generation. This shift ensures that, at each time step, the model predicts the next token using only the ground truth tokens that precede it, rather than having access to the token it is supposed to predict. By prepending a start-of-sequence token and shifting the entire output one position to the right, the model learns to generate the sequence step-by-step, just as it would during inference. This mechanism aligns the input and output of the decoder temporally, enabling efficient parallel training while ensuring that the causal constraint (not attending to future tokens) is strictly maintained. Without this shift, the model would see the actual target token during prediction, resulting in information leakage and an unrealistic training signal.

9. To improve the training process in Parts 6, 9, and 12 of the Transformer architecture, where key components like multi-head attention, masking, and feedforward networks are implemented, we can apply a combination of optimization strategies and architectural enhancements. Learning rate scheduling with warm-up and decay improves stability, while gradient clipping helps control exploding gradients. Label smoothing reduces overconfidence and enhances generalization. Architecturally, using pre-layer normalization improves gradient flow, and dropout in attention and feedforward layers helps prevent overfitting. Increasing the number of attention heads or feedforward dimensions can enhance representation capacity. Additionally, techniques like parameter-efficient tuning (e.g., LoRA) and optimized attention implementations (e.g., FlashAttention) can improve both performance and training efficiency.

10. Different Transformer variants are suited to different tasks based on their input-output structure. An encoder-only Transformer (e.g., BERT) is designed for tasks that require understanding a single input sequence, such as text classification, named entity recognition, and sentence similarity, where full bidirectional context is beneficial. A decoder-only Transformer (e.g., GPT) is autoregressive and suited for language modeling, text generation, code completion, and dialogue systems, where the model predicts the next token based on past tokens. The encoder-decoder Transformer (e.g., T5, BART, original Transformer) is ideal for sequence-to-sequence tasks like machine translation, summarization, and question answering, where the output must be conditioned on a separate input sequence. Each architecture leverages attention differently to align with the nature of the task it is designed for.

References

Hedu AI by Batool Haider. (2020, December 8). *Visual guide to Transformer neural networks – (Episode 1) Position embeddings* [Video]. YouTube. https://www.youtube.com/watch?v=mMa2PmYJlCo

Hedu AI by Batool Haider. (2020, December 8). *Visual guide to Transformer neural networks – (Episode 2) Multi-head & self-attention* [Video]. YouTube. https://www.youtube.com/watch?v=gJ9kaJsE78k

Hedu AI by Batool Haider. (2021, February 2). *Visual guide to Transformer neural networks – (Episode 3) Decoder's masked attention* [Video]. YouTube. https://www.youtube.com/watch?v=gJ9kaJsE78k

Raschka, S. (2024, January 14). *Understanding and coding self-attention, multi-head attention, causal-attention, and cross-attention in LLMs*. Ahead of AI. https://magazine.sebastianraschka.com/p/understanding-and-coding-self-attention

Vaswani, A., et al. (2017). Attention Is All You Need. Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017), 5998-6008. https://arxiv.org/abs/1706.03762

Xiao, T., & Zhu, J. (2025). *Foundations of large language models*. arXiv. https://doi.org/10.48550/arXiv.2501.09223