MIDDLE EAST TECHNICAL UNIVERSITY

SEMESTER I EXAMINATION 2024-2025

CENG 403 – Deep Learning - Self-Attention & Transformers (Professor-Based) - ANSWERED

January 2025 TIME ALLOWED: 3 HOURS

INSTRUCTIONS TO CANDIDATES

1. This examination paper contains **SIX (6)** questions and comprises **EIGHT (8)** printed pages.
2. Answer all questions. The marks for each question are indicated at the beginning of each question.
3. Answer each question beginning on a **FRESH** page of the answer book.
4. This **IS NOT an OPEN BOOK** exam.
5. Show all mathematical derivations clearly with proper notation.
6. Draw clear diagrams with proper labels where requested.
7. Explain the intuition behind mechanisms where asked.

Question 1. Vanilla Self-Attention Mechanism (20 marks)
Based on Week 14a lecture content on basic self-attention.

(a) Explain the motivation behind self-attention. Why did the professor suggest moving away from sequential processing in RNNs to parallel processing? (5 marks)

Answer: Self-attention enables parallel processing of sequences, overcoming RNN's sequential bottleneck and long-term dependency issues.

Motivation for Self-Attention:

1. Sequential Processing Bottleneck in RNNs:

- RNNs process sequences step-by-step: $h_t = f(h_{t-1}, x_t)$
- Cannot proceed to time step $t + 1$ until t is complete
- For long sequences, this creates computational bottlenecks
- Makes parallel processing impossible during training and inference

2. Long-term Dependency Problem:

- Information from early time steps must flow through many hidden states
- Gradient vanishing makes learning long-range dependencies difficult
- Important early information may be lost or diluted

3. Parallel Processing Advantage:

- Self-attention can process all positions simultaneously
- Each position directly attends to all other positions
- No sequential dependency - can leverage modern parallel hardware
- Dramatically speeds up training and inference

4. Direct Information Access:

- Any position can directly access information from any other position
- No information bottleneck through intermediate hidden states
- Better modeling of long-range dependencies

(b) Consider a sequence of 3 word embeddings E_0, E_1, E_2. Using the vanilla self-attention approach described in the lecture, show step-by-step how to compute the updated embedding E_0'. Include: (10 marks)

- Similarity computation using dot product
- Softmax normalization
- Weighted combination

Answer: Step-by-step vanilla self-attention computation for updating E_0.

Given: Embeddings $E_0, E_1, E_2 \in \mathbb{R}^d$

Step 1: Similarity Computation using Dot Product Compare E_0 with all embeddings (including itself):

$$s_{0,0} = E_0 \cdot E_0 = E_0^T E_0 \tag{1}$$
$$s_{0,1} = E_0 \cdot E_1 = E_0^T E_1 \tag{2}$$
$$s_{0,2} = E_0 \cdot E_2 = E_0^T E_2 \tag{3}$$

Step 2: Softmax Normalization Normalize similarity scores to get attention weights:

$$\alpha_{0,0} = \frac{\exp(s_{0,0})}{\exp(s_{0,0}) + \exp(s_{0,1}) + \exp(s_{0,2})} \tag{4}$$
$$\alpha_{0,1} = \frac{\exp(s_{0,1})}{\exp(s_{0,0}) + \exp(s_{0,1}) + \exp(s_{0,2})} \tag{5}$$
$$\alpha_{0,2} = \frac{\exp(s_{0,2})}{\exp(s_{0,0}) + \exp(s_{0,1}) + \exp(s_{0,2})} \tag{6}$$

Note: $\alpha_{0,0} + \alpha_{0,1} + \alpha_{0,2} = 1$

Step 3: Weighted Combination Compute updated embedding as weighted sum:

$$E_0' = \alpha_{0,0} \cdot E_0 + \alpha_{0,1} \cdot E_1 + \alpha_{0,2} \cdot E_2$$

Intuition:

- Higher similarity $\rightarrow$ higher attention weight

- E'_0 is a weighted mixture of all embeddings
- Weights determined by how relevant each embedding is to E_0
- Self-attention allows E_0 to gather information from the entire sequence

(c) The professor mentioned that "different time steps can be processed in parallel." Explain what this means and why it's advantageous over RNN processing. (5 marks)

Answer: Parallel processing means all sequence positions can be computed simultaneously, unlike RNNs which require sequential computation.

Parallel Processing in Self-Attention:

What it means:

- All updated embeddings E'_0, E'_1, E'_2 can be computed simultaneously
- No dependency between computations for different positions
- Each position's computation is independent of others' completion

Implementation:

- Use matrix operations for entire sequence at once
- Compute all similarity scores in parallel: $S = EE^T$
- Apply softmax to all rows simultaneously
- Compute all weighted combinations in parallel

Advantages over RNN Processing:

1. Speed:

- RNN: $O(T)$ sequential steps for sequence length T
- Self-attention: $O(1)$ parallel steps
- Massive speedup on modern GPUs with many cores

2. Hardware Utilization:

- RNNs underutilize parallel hardware (GPUs)

- Self-attention fully exploits matrix multiplication units
- Better memory bandwidth utilization

3. Training Efficiency:

- Can process entire batches of sequences in parallel
- Faster convergence due to better gradient flow
- More stable training dynamics

Question 2. Query-Key-Value Self-Attention (25 marks)

Based on the professor's explanation of extending vanilla self-attention.

(a) The professor introduced Query, Key, and Value functions as "parametric functions" to increase network capacity. Explain the intuition behind each of these three components: (8 marks)

- What does the Query represent conceptually?
- What does the Key represent conceptually?
- What does the Value represent conceptually?

Answer: Query-Key-Value mechanism provides learnable projections that enable more expressive attention computations than raw embeddings.

Query (Q) - "What am I looking for?"

- Represents the information need of the current position
- A learned transformation of the embedding: $Q_i = W_Q E_i$
- Encodes what kind of information position i wants to gather
- Example: In "The cat sat on the mat", query for "sat" might look for subject and object information

Key (K) - "What do I have to offer?"

- Represents the content/information available at each position
- A learned transformation: $K_j = W_K E_j$
- Encodes what type of information position j can provide
- Used to compute similarity with queries
- Example: Key for "cat" might represent subject-related information

Value (V) - "What information do I actually provide?"

- Represents the actual information content to be aggregated
- A learned transformation: $V_j = W_V E_j$
- The information that gets mixed based on attention weights
- Can be different from the key representation

- Example: Value for "cat" might contain semantic features about the animal

Why Separate Q, K, V?

- Allows different aspects of embeddings for matching vs. content
- Increases model expressivity compared to using raw embeddings
- Enables learning what to look for vs. what to provide
- Provides more flexible attention patterns

(b) Given word embeddings E_0 and E_1, and weight matrices W_Q, W_K, and W_V, write the mathematical equations for computing: (8 marks)

- Query vectors: $Q_0 =?$, $Q_1 =?$
- Key vectors: $K_0 =?$, $K_1 =?$
- Value vectors: $V_0 =?$, $V_1 =?$

Answer: Linear transformations applied to embeddings using learned weight matrices.

Query Vectors:

$$Q_0 = W_Q E_0 \tag{7}$$
$$Q_1 = W_Q E_1 \tag{8}$$

Key Vectors:

$$K_0 = W_K E_0 \tag{9}$$
$$K_1 = W_K E_1 \tag{10}$$

Value Vectors:

$$V_0 = W_V E_0 \tag{11}$$
$$V_1 = W_V E_1 \tag{12}$$

Matrix Dimensions:

- $E_i \in \mathbb{R}^{d_{model}}$ (embedding dimension)

- $W_Q, W_K, W_V \in \mathbb{R}^{d_k \times d_{model}}$ (projection matrices)
- $Q_i, K_i, V_i \in \mathbb{R}^{d_k}$ (projected vectors)
- Typically $d_k = d_{model}$ for single-head attention

Key Points:

- Same weight matrices used for all positions (parameter sharing)
- Different projections allow specialization of representations
- Learnable parameters optimize during training

(c) The professor mentioned "we use the same weight for each word." Explain what this means and why it's important for the self-attention mechanism. (4 marks)

Answer: Parameter sharing means the same W_Q, W_K, W_V matrices are applied to all sequence positions, enabling position-invariant learning.

What "Same Weight" Means:

- Single set of weight matrices W_Q, W_K, W_V for entire sequence
- No position-specific parameters
- All embeddings transformed using identical linear layers

Why This is Important:

1. Generalization:

- Model learns general transformation functions
- Works for sequences of any length
- No overfitting to specific positions

2. Parameter Efficiency:

- $O(d^2)$ parameters instead of $O(T \cdot d^2)$ for sequence length T
- Scales well to long sequences
- Fewer parameters to learn

3. Position Invariance:

- Same word gets same Q, K, V regardless of position

- Attention patterns emerge from content, not position
- Positional information added separately via positional encoding

(d) Derive the complete scaled dot-product attention formula as presented in the lecture, including the scaling factor $\frac{1}{\sqrt{d}}$. Explain why this scaling is necessary. (5 marks)

Answer: Scaled dot-product attention with normalization to prevent saturation in softmax function.

Complete Scaled Dot-Product Attention Formula:

For updating position i:

$$\text{Attention}(Q_i, K, V) = \sum_{j=1}^{n} \alpha_{ij} V_j \tag{13}$$

Where attention weights are:

$$\alpha_{ij} = \frac{\exp\left(\frac{Q_i \cdot K_j}{\sqrt{d_k}}\right)}{\sum_{k=1}^{n} \exp\left(\frac{Q_i \cdot K_k}{\sqrt{d_k}}\right)} \tag{14}$$

Matrix Form:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

Why Scaling by $\frac{1}{\sqrt{d_k}}$ is Necessary:

1. Dot Product Magnitude Growth:

- For random vectors $q, k \in \mathbb{R}^{d_k}$, $\mathbb{E}[q \cdot k] = 0$
- But $\text{Var}[q \cdot k] = d_k$ (assuming unit variance elements)
- Standard deviation grows as $\sqrt{d_k}$
- Large d_k leads to very large dot products

2. Softmax Saturation:

- Large dot products push softmax into saturation regions

- Gradients become very small (vanishing gradient)
- Attention becomes too peaked (one-hot like)
- Training becomes unstable

3. Scaling Effect:

- Dividing by $\sqrt{d_k}$ normalizes variance back to 1
- Keeps dot products in reasonable range
- Maintains useful gradients for learning
- Allows softer, more distributed attention patterns

Question 3. Multi-Head Attention (20 marks)
Based on the professor's explanation of multiple attention heads.

(a) The professor stated: "with one scaled dot product attention we are looking at just one potential interpretation of a word." Explain why multiple attention heads are needed and what they accomplish. (6 marks)

Answer: Multiple attention heads capture different types of relationships and interpretations simultaneously, enabling richer representation learning.

Limitations of Single Attention Head:

- Single attention pattern per position
- One interpretation/relationship at a time
- Limited capacity to capture diverse linguistic phenomena
- May miss important secondary relationships

Why Multiple Heads are Needed:

1. Multiple Relationships:

- Words participate in different types of relationships simultaneously
- Example: "bank" could relate to "river" (location) and "money" (finance)
- Each head can specialize in different relationship types

2. Different Linguistic Patterns:

- Syntactic relationships (subject-verb, modifier-noun)
- Semantic relationships (synonymy, co-occurrence)
- Long-range vs. short-range dependencies
- Different levels of abstraction

3. Representation Richness:

- Ensemble of attention patterns
- More robust feature extraction
- Better disambiguation of ambiguous words

- Improved model capacity without excessive parameters

What Multiple Heads Accomplish:

- Parallel processing of different interpretations
- Specialization of attention patterns
- More expressive final representations
- Better handling of complex linguistic phenomena

(b) Describe the multi-head attention process as explained in the lecture: (10 marks)

- How are the h different heads created?
- Why are projections to "lower dimensional spaces" used?
- How are the outputs combined?

Answer: Multi-head attention creates h parallel attention mechanisms with lower-dimensional projections, then concatenates and projects the results.

Creating h Different Heads:

1. Separate Projection Matrices: For each head $i \in \{1, 2, ..., h\}$:

$$Q^{(i)} = XW_Q^{(i)} \tag{15}$$
$$K^{(i)} = XW_K^{(i)} \tag{16}$$
$$V^{(i)} = XW_V^{(i)} \tag{17}$$

2. Independent Attention Computation:

$$\text{head}_i = \text{Attention}(Q^{(i)}, K^{(i)}, V^{(i)}) = \text{softmax}\left(\frac{Q^{(i)}K^{(i)T}}{\sqrt{d_k}}\right)V^{(i)}$$

Lower Dimensional Projections:

Why Lower Dimensions?

- Original dimension: d_{model} (e.g., 512)
- Head dimension: $d_k = d_v = \frac{d_{model}}{h}$ (e.g., 64 for $h = 8$)

- Reduces computational cost per head
- Controls total parameter count

Parameter Efficiency:

- Total parameters: $h \times 3 \times d_{model} \times d_k = 3 \times d_{model}^2$
- Same as single large head with dimension d_{model}
- But provides h different attention patterns

Combining Outputs:

1. Concatenation:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, ..., \text{head}_h)W_O$$

2. Final Projection:

- Concatenated output: dimension $h \times d_k = d_{model}$
- Output projection: $W_O \in \mathbb{R}^{d_{model} \times d_{model}}$
- Mixes information from different heads
- Returns to original embedding dimension

Complete Process:

(a) Project embeddings to h sets of Q, K, V (lower dim)
(b) Compute h attention patterns in parallel
(c) Concatenate all head outputs
(d) Apply final linear projection
(e) Result: enriched representations with multiple perspectives

(c) The professor showed an attention visualization where the word "it" attends to relevant words. Explain how this demonstrates the disambiguation capability of attention mechanisms. (4 marks)

Answer: Attention visualizations show how ambiguous words like "it" learn to focus on contextually relevant words, resolving ambiguity through learned attention patterns.

Disambiguation Through Attention:

1. The Ambiguity Problem:

- "It" is a pronoun that can refer to different entities
- Without context, "it" has no specific meaning
- Traditional embeddings give fixed representation regardless of context

2. How Attention Resolves Ambiguity:

- Attention weights show where "it" is "looking"
- High attention to relevant nouns indicates referent resolution
- Different contexts lead to different attention patterns
- Example: "The animal crossed the street. It was scared." $\rightarrow$ high attention to "animal"

3. What the Visualization Demonstrates:

- Network learns meaningful semantic relationships
- Attention isn't random - it follows linguistic principles
- Different attention heads may capture different relationships
- Model discovers syntax and semantics without explicit supervision

4. Broader Implications:

- Contextual representations emerge naturally
- Same word gets different representations in different contexts
- Attention serves as interpretable mechanism for understanding model behavior
- Validates that model learns linguistically meaningful patterns

Question 4. Transformer Architecture (25 marks)
Based on the professor's detailed explanation of transformer blocks.

(a) The professor noted that self-attention "loses position information." Explain this problem and describe the positional encoding solution using trigonometric functions. (8 marks)

Answer: Self-attention is permutation invariant, losing word order information. Positional encoding adds position-specific signals to embeddings.

Position Information Loss Problem:

1. Permutation Invariance:

- Self-attention computes attention weights based on content similarity
- Same words in different orders produce identical representations
- Example: "cat chased dog" vs "dog chased cat" would get same embeddings
- Critical word order information is lost

2. Why Position Matters:

- Word order determines meaning in most languages
- Syntax depends on positional relationships
- Different positions may have different linguistic roles
- Temporal sequence is crucial for language understanding

Positional Encoding Solution:

1. Basic Idea:

- Add position-specific vectors to word embeddings
- Each position gets unique positional encoding
- Combined embedding: input = word_embedding+positional_encoding

2. Trigonometric Functions:

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

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

Where:

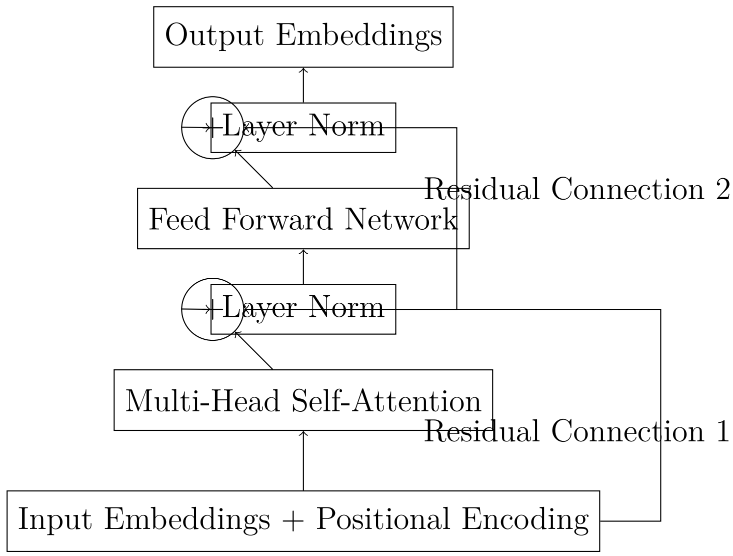
- pos = position in sequence (0, 1, 2, ...)
- i = dimension index (0, 1, 2, ..., $d_{model}/2 - 1$)
- Even dimensions use sine, odd dimensions use cosine

3. Properties of Trigonometric Encoding:

- Different frequencies for different dimensions
- Deterministic (no learnable parameters)
- Bounded values: $[-1, 1]$
- Can extrapolate to longer sequences than seen in training
- Enables learning of relative position relationships

(b) Draw and label a complete transformer encoder block as described in the lecture, including: (12 marks)

- Multi-head self-attention
- Skip connections (residual connections)
- Layer normalization
- Feed-forward network
- All input/output flows



Answer: Complete transformer encoder block with multi-head attention, residual connections, and layer normalization.

Component Descriptions:

1. Multi-Head Self-Attention:

- Processes input embeddings with multiple attention heads
- Captures different types of relationships
- Output has same dimension as input

2. Residual Connections (Skip Connections):

- Direct path from input to addition point
- Helps gradient flow during backpropagation
- Enables training of deeper networks
- Formula: output = sublayer(input) + input

3. Layer Normalization:

- Normalizes features across embedding dimension
- Stabilizes training dynamics
- Applied after residual addition

- Formula: $\text{LayerNorm}(x) = \gamma \frac{x-\mu}{\sigma} + \beta$

4. Feed-Forward Network:

- Two linear transformations with ReLU activation
- Applied position-wise (same network for each position)
- Increases then decreases dimensionality
- $\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$

Information Flow:

(a) Input embeddings + positional encoding
(b) Multi-head self-attention
(c) Add residual connection + layer normalization
(d) Feed-forward network
(e) Add residual connection + layer normalization
(f) Output embeddings (ready for next layer)

(c) Explain the purpose of skip connections and layer normalization in the transformer architecture as discussed in the lecture. (5 marks)

Answer: Skip connections enable gradient flow and layer normalization stabilizes training, together enabling deep transformer architectures.

Skip Connections (Residual Connections):

Purpose:

- Provide direct gradient flow path
- Enable identity mapping when needed
- Facilitate training of deep networks
- Prevent degradation problem

How They Work:

- $\text{output} = F(\text{input}) + \text{input}$
- Sublayer learns residual function F
- Easier to learn modifications than complete transformations

- Gradient flows directly through skip path

Layer Normalization:

Purpose:

- Stabilize training dynamics
- Reduce internal covariate shift
- Enable higher learning rates
- Improve convergence speed

How It Works:

- Normalizes across feature dimension for each position
- Zero mean, unit variance for each embedding
- Learnable scale (γ) and shift (β) parameters
- Applied after residual addition

Combined Benefits:

- Enables training of 12+ layer transformers
- Stable gradients throughout the network
- Faster convergence and better final performance
- Robust to hyperparameter choices

END OF PAPER