MODULE 4 • CS24351

Module 4: Deep Learning for NLP, Recurrent Architectures & Transformers

Topics 35 to 45 • RNNs, BPTT, LSTMs, Attention, Transformers, BERT & GPT Architectures

Module 4: Deep Learning for NLP, Recurrent Architectures & Transformers

📌 **Syllabus & Pedagogical Blueprint (Topics 35 to 45)**

Recurrent Neural Networks (RNN) • Backpropagation Through Time (BPTT) • Mathematical Derivations of Vanishing & Exploding Gradients • Gradient Clipping • Long Short-Term Memory (LSTM) Internal Gate Mechanics (Forget, Input, Candidate Cell, Output Gates, Constant Error Carousel) • Gated Recurrent Units (GRU) • Bidirectional Encoders (BiLSTM) • Sequence-to-Sequence (Seq2Seq) Architecture & Information Bottleneck • Bahdanau Additive Attention vs. Luong Multiplicative Attention • The Transformer Architecture (Vaswani et al. 2017) • Scaled Dot-Product Self-Attention • Multi-Head Attention (MHA) • Sinusoidal Positional Encoding Formulations • Encoder & Decoder LayerNorm and Residual Connections • Pretrained Language Models: BERT (MLM, NSP, Bidirectionality), GPT (Autoregressive Decoder), RoBERTa, T5, BART • 15 Solved University Examination Questions.

Topic 35: Recurrent Neural Networks (RNN) & The Vanishing Gradient Problem

An Elman RNN processes sequential inputs $\mathbf{x}_1, \dots, \mathbf{x}_T$ by updating a continuous hidden state $\mathbf{h}_t$:

$$\mathbf{h_t} = \tanh(\mathbf{W_{hh}}\mathbf{h_{t-1}} + \mathbf{W_{xh}}\mathbf{x_t} + \mathbf{b_h}) \qquad \hat{\mathbf{y}}_\mathbf{t} = \mathrm{softmax}(\mathbf{W_{hy}}\mathbf{h_t} + \mathbf{b_y})$$

⚠️ **Mathematical Proof of Vanishing Gradients during BPTT**

When backpropagating loss $\mathcal{L}_T$ back to step t, the gradient chain rule involves repeated matrix multiplications:

$$\frac{\partial \mathcal{L}_\mathbf{T}}{\partial \mathbf{h_t}} = \frac{\partial \mathcal{L}_\mathbf{T}}{\partial \mathbf{h_T}} \prod_{\mathbf{k=t+1}}^{\mathbf{T}} \frac{\partial \mathbf{h_k}}{\partial \mathbf{h_{k-1}}} = \frac{\partial \mathcal{L}_\mathbf{T}}{\partial \mathbf{h_T}} \prod_{\mathbf{k=t+1}}^{\mathbf{T}} \mathbf{W_{hh}^T} \mathrm{diag}(\mathbf{1} - \tanh^{\mathbf{2}}(\mathbf{z_k}))$$

Because $\tanh'(z) = 1 - \tanh^2(z) \leq 1.0$, if the maximum eigenvalue $\lambda_{\max}(\mathbf{W}_{hh}) < 1$, the product $\prod_{k=t+1}^{T} \mathbf{W}_{hh}^T \to 0$ exponentially as sequence length $T - t > 10$, destroying long-range dependencies! If $\lambda_{\max} > 1$, gradients explode ($\to \infty$).

Topic 36: Long Short-Term Memory (LSTM) Architecture

Hochreiter & Schmidhuber (1997) introduced the **LSTM**, which introduces an explicit linear memory cell state $\mathbf{C}_t$ governed by three multiplicative gates:

🏛️ Complete LSTM Mathematical Forward Equations

1. **Forget Gate ($\mathbf{f}_t$):** Decides what information to discard from prior cell state $\mathbf{C}_{t-1}$:

$$\mathbf{f_t} = \sigma(\mathbf{W_f}[\mathbf{h_{t-1}}, \mathbf{x_t}] + \mathbf{b_f})$$

2. **Input Gate ($\mathbf{i}_t$) & Candidate Cell ($\tilde{\mathbf{C}}_t$):** Decides which new values to write into memory:

$$\mathbf{i_t} = \sigma(\mathbf{W_i}[\mathbf{h_{t-1}}, \mathbf{x_t}] + \mathbf{b_i}) \qquad \tilde{\mathbf{C}}_\mathbf{t} = \tanh(\mathbf{W_c}[\mathbf{h_{t-1}}, \mathbf{x_t}] + \mathbf{b_c})$$

3. **Cell State Update ($\mathbf{C}_t$):** Linear aggregation (Constant Error Carousel):

$$\mathbf{C_t} = \mathbf{f_t} \odot \mathbf{C_{t-1}} + \mathbf{i_t} \odot \tilde{\mathbf{C}}_\mathbf{t}$$

4. **Output Gate ($\mathbf{o}_t$) & Hidden State ($\mathbf{h}_t$):** Filters cell state to produce output:

$$\mathbf{o_t} = \sigma(\mathbf{W_o}[\mathbf{h_{t-1}}, \mathbf{x_t}] + \mathbf{b_o}) \qquad \mathbf{h_t} = \mathbf{o_t} \odot \tanh(\mathbf{C_t})$$

Why LSTM Prevents Vanishing Gradients: The derivative $\frac{\partial \mathbf{C}_t}{\partial \mathbf{C}_{t-1}} = \mathbf{f}_t$. If the forget gate is saturated at $\mathbf{f}_t \approx 1$, error gradients flow backward through time with zero exponential decay!

Topic 37: The Transformer Architecture (Vaswani et al. 2017)

The **Transformer** completely replaces recurrence with pure attention mechanisms, enabling massive parallelization across all tokens during training.

1. Scaled Dot-Product Attention

Given Query ($\mathbf{Q}$), Key ($\mathbf{K}$), and Value ($\mathbf{V}$) matrices projected from input embeddings:

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

Scaling Factor $\frac{1}{\sqrt{d_k}}$: For large projection dimensions d_k, the dot products grow large in magnitude, pushing the softmax function into regions with vanishingly small gradients. Scaling by $\sqrt{d_k}$ stabilizes variance to unit magnitude.

2. Multi-Head Attention (MHA)

Allows the model to jointly attend to information from different representation subspaces at different positions:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_\mathbf{1}, \ldots, \text{head}_\mathbf{h})\mathbf{W^O}$$

$$\text{where head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q,\ \mathbf{K}\mathbf{W}_i^K,\ \mathbf{V}\mathbf{W}_i^V)$$

3. Sinusoidal Positional Encodings

Because self-attention is permutation-invariant, positional information must be injected via deterministic sinusoids:

$$\mathbf{PE}_{(\mathbf{pos},\mathbf{2i})} = \sin\left(\frac{\mathbf{pos}}{\mathbf{10000^{2i/d_{\text{model}}}}}\right) \qquad \mathbf{PE}_{(\mathbf{pos},\mathbf{2i+1})} = \cos\left(\frac{\mathbf{pos}}{\mathbf{10000^{2i/d_{\text{model}}}}}\right)$$

Topic 38: Pretrained Transformer Models — BERT vs. GPT

Dimension	BERT (Devlin et al. 2018)	GPT Series (Radford et al. 2018–2023)
Architecture	**Encoder-Only:** Full bidirectional self-attention across all positions.	**Decoder-Only:** Causal masked self-attention (tokens attend only to prior positions).
Pretraining Objective	Masked Language Model (MLM 15%) + Next Sentence Prediction (NSP).	Autoregressive Next-Token Prediction: $\max_\theta \sum \log P(x_i \mid x_{
Ideal Tasks	NLU: Text classification, NER, extractive QA, sentence similarity.	NLG: Open-ended generation, creative writing, dialogue, in-context zero/few-shot reasoning.

Topic 39: Master University Examination Question Bank (Module 4)

Q1. Explain the Masked Multi-Head Attention in Transformer Decoders and why causal masking is required. (8 Marks)

During autoregressive text generation, the model predicts token y_t conditioning strictly on preceding tokens y_{t}, a **Causal Mask Matrix** $\mathbf{M}$ (with upper triangular entries set to $-\infty$) is added to attention logits before softmax:

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

Because $\exp(-\infty) = 0$, attention weights to all future positions become strictly zero!

Q2. Compare Gated Recurrent Unit (GRU) with LSTM in terms of parameters and gate architecture. (6 Marks)

- **GRU (Cho et al. 2014):** Merges cell state and hidden state into a single hidden vector $\mathbf{h}_t$. Contains only 2 gates: *Reset Gate* ($\mathbf{r}_t$) and *Update Gate* ($\mathbf{z}_t$). Features $\approx 25\%$ fewer parameters than LSTM, training faster with comparable performance on smaller datasets.
- **LSTM:** Maintains separate cell state $\mathbf{C}_t$ and hidden state $\mathbf{h}_t$ controlled by 3 distinct gates (f_t, i_t, o_t).

Topic 40: Attention Mechanisms — Additive (Bahdanau) vs. Multiplicative (Luong)

Attention Type	Alignment Score Function $\text{score}(\mathbf{s}_i, \mathbf{h}_j)$	Computational Efficiency
Bahdanau Additive Attention	$\text{score}(\mathbf{s}_i, \mathbf{h}_j) = \mathbf{v}_a^T \tanh(\mathbf{W}_a\mathbf{s}_i + \mathbf{U}_a\mathbf{h}_j)$	Requires single hidden layer feed-forward network; higher parameter count.
Luong Multiplicative (General)	$\text{score}(\mathbf{s}_i, \mathbf{h}_j) = \mathbf{s}_i^T\mathbf{W}_a\mathbf{h}_j$	Matrix multiplication; highly optimized on GPU tensor cores.
Luong Dot Product	$\text{score}(\mathbf{s}_i, \mathbf{h}_j) = \mathbf{s}_i^T\mathbf{h}_j$	Zero parameters; requires decoder and encoder hidden states to have identical dimensions.

Topic 41: Pretrained Foundation Transformer Taxonomy

Model Family	Architectural Paradigm	Pretraining Objective	Typical Downstream Use-Cases
BERT / RoBERTa / DeBERTa	**Encoder-Only** (Bidirectional self-attention)	Masked Language Modeling (MLM 15%) + Disentangled Attention	Classification, Token Tagging (NER), Extractive QA (SQuAD), Sentence Embeddings.
GPT Series (1-4) / LLaMA / Mistral	**Decoder-Only** (Causal autoregressive masked attention)	Next-Token Autoregressive Prediction $\sum \log P(x_t \mid x_{	Open-ended text generation, In-context zero/few-shot reasoning, Conversational AI.
T5 / BART	**Encoder-Decoder** (Full cross-attention)	Span Corruption / Denoising Autoencoding sequence-to-sequence	Abstractive text summarization, Machine translation, Text re-writing.

Topic 45: Advanced Deep Transformer Architectures & University Exam Problem Bank (Part II)

🏛️ Solved Numerical 12: Multi-Head Attention Dimensions & Tensor Computation

Given Transformer model dimension $d_{\text{model}} = 512$, number of attention heads $h = 8$, sequence length $T = 64$:

- Head projection dimension: $\mathbf{d_k} = \mathbf{d_v} = \frac{\mathbf{d_{model}}}{\mathbf{h}} = \frac{\mathbf{512}}{\mathbf{8}} = \mathbf{64}$.
- Input matrix $\mathbf{X} \in \mathbb{R}^{64\times512}$.
- Weight projection matrices per head: $\mathbf{W}_i^Q \in \mathbb{R}^{512\times64}$, $\mathbf{W}_i^K \in \mathbb{R}^{512\times64}$, $\mathbf{W}_i^V \in \mathbb{R}^{512\times64}$.
- Raw attention logits: $\mathbf{Q}_i\mathbf{K}_i^T \in \mathbb{R}^{64\times64}$.
- Scaled dot product attention per head: $\text{head}_i = \text{softmax}\left(\frac{\mathbf{Q}_i\mathbf{K}_i^T}{\sqrt{64}}\right)\mathbf{V}_i \in \mathbb{R}^{64\times64} \times \mathbb{R}^{64\times64} = \mathbb{R}^{64\times64}$.
- Concatenated output across all 8 heads: $\text{Concat}(\text{head}_1, \ldots, \text{head}_8) \in \mathbb{R}^{64\times512}$.
- Final linear projection: $\mathbf{W}^O \in \mathbb{R}^{512\times512} \implies \text{Output Tensor } \mathbf{Y} \in \mathbb{R}^{\mathbf{64\times512}}$.

Q3. Detail the RoBERTa (Robustly Optimized BERT Approach) Modifications over Original BERT. (8 Marks)

Liu et al. (2019) demonstrated that original BERT was severely undertrained, introducing 4 critical modifications that established new state-of-the-art results across GLUE, SQuAD, and RACE:

1. **Removed Next Sentence Prediction (NSP):** Showing that NSP hurts downstream task performance when inputs are continuous sequences of full documents.
2. **Dynamic Masking:** Generating a novel 15% random mask pattern every time a sequence is fed to the model across training epochs (original BERT used static masking fixed at data pre-processing).
3. **Larger Batch Sizes & Learning Rates:** Scaling batch size from 256 to 8,000 sequences with AdamW optimization.
4. **Byte-Level BPE:** Using a 50,000-token byte-level BPE tokenizer without unicode preprocessing.

Topic 45.3: Master University Exam Problem Bank (Part III)

🏛️ Solved Problem 13: Transformer Layer Normalization vs Batch Normalization

In Computer Vision, Batch Normalization normalizes across batch dimension N for each channel. In NLP, sentence lengths vary widely and mini-batches contain padded tokens:

- **Layer Normalization (Ba et al. 2016):** Normalizes across the *hidden feature dimensions* d_{model} independently for each individual token:

$$\mu_i = \frac{1}{d}\sum_{j=1}^{d} x_{ij} \qquad \sigma_i^2 = \frac{1}{d}\sum_{j=1}^{d}(x_{ij}-\mu_i)^2$$

$$\text{LayerNorm}(x_i) = \frac{x_i - \mu_i}{\sqrt{\sigma_i^2+\epsilon}} \odot \gamma + \beta$$

- *Advantage:* Completely independent of batch size and padding tokens; identical execution during training and autoregressive single-token inference!

Q4. Detail the KV-Cache (Key-Value Caching) Optimization for Fast Autoregressive LLM Inference. (8 Marks)

During autoregressive generation in decoder-only LLMs (GPT), generating token $t+1$ requires computing attention over all preceding tokens $1 \dots t$.
• **Naive Generation:** Re-computing Key and Value projections for all t past tokens at every step scales quadratically $O(T^2)$ in compute.
• **KV-Cache:** Stores past computed Key ($\mathbf{K}_{1..t}$) and Value ($\mathbf{V}_{1..t}$) tensor matrices in GPU RAM. At step $t+1$, the model computes Query, Key, and Value *only for the single newly generated token* ($\mathbf{q}_{t+1}, \mathbf{k}_{t+1}, \mathbf{v}_{t+1}$), appends $\mathbf{k}_{t+1}, \mathbf{v}_{t+1}$ to cache, and computes attention in $O(T)$ linear time!

Q5. Explain the T5 (Text-to-Text Transfer Transformer) Unified Framework. (8 Marks)

T5 (Raffel et al. 2020) unifies all NLP tasks into a single universal **Text-to-Text** sequence-to-sequence encoder-decoder architecture where both inputs and outputs are always text strings:
• **Classification:** `"cola sentence: The course is fine." -> "acceptable"`
• **Translation:** `"translate English to German: That is good. -> Das ist gut."`
• **Summarization:** `"summarize: [Article text] -> [Summary]"`
Pretrained using *Span Corruption* (masking random contiguous token spans with sentinel tokens `` and training the decoder to reconstruct the missing text).

🏛️ Solved Problem 18: Scaled Dot-Product Attention Variance Stability Proof

Prove why the dot product $\mathbf{q} \cdot \mathbf{k} = \sum_{i=1}^{d_k} q_i k_i$ has variance d_k under independent zero-mean unit-variance components:

- Assume components $q_i, k_i \sim \mathcal{N}(0,1)$ are independent random variables with $\mathbb{E}[q_i] = 0, \text{Var}(q_i) = 1$.
- $\mathbb{E}[q_i k_i] = \mathbb{E}[q_i]\mathbb{E}[k_i] = 0$.
- $\text{Var}(q_i k_i) = \mathbb{E}[q_i^2 k_i^2] - (\mathbb{E}[q_i k_i])^2 = \mathbb{E}[q_i^2]\mathbb{E}[k_i^2] = (1)(1) = 1$.
- The sum of d_k independent random variables has variance:

$$\text{Var}(\mathbf{q}\cdot\mathbf{k}) = \sum_{i=1}^{d_k} \text{Var}(q_i k_i) = \sum_{i=1}^{d_k} 1 = d_k$$

- Dividing by standard deviation $\sqrt{d_k}$: $\text{Var}\left(\frac{\mathbf{q}\cdot\mathbf{k}}{\sqrt{d_k}}\right) = \frac{1}{d_k}\text{Var}(\mathbf{q}\cdot\mathbf{k}) = \frac{d_k}{d_k} = 1.0$.
- *Significance:* Stabilizes logits across arbitrary model dimensions ($d_k = 64, 128$), preventing softmax saturation!

Q6. Explain the Reinforcement Learning from Human Feedback (RLHF) Pipeline with PPO and DPO. (8 Marks)

To align generative foundation models with human helpfulness, honesty, and harmlessness (HHH):
1. **Supervised Fine-Tuning (SFT):** Fine-tune base LLM on high-quality instruction-response demonstrations.
2. **Reward Model (RM) Training:** Present human annotators with model responses (y_w, y_l) and train RM $r_\theta(x, y)$ via pairwise ranking loss: $\mathcal{L}_{\text{RM}} = -\log \sigma(r_\theta(x, y_w) - r_\theta(x, y_l))$.
3. **RL Optimization (PPO / DPO):** Optimize policy π_θ against reward model penalized by KL-divergence from SFT model: $\max_\theta \mathbb{E}[r_\theta(x, y) - \beta D_{\text{KL}}(\pi_\theta || \pi_{\text{SFT}})]$. Direct Preference Optimization (DPO - Rafailov et al. 2023) replaces PPO by deriving the exact closed-form optimal policy loss without training an explicit reward model!

🏛️ Solved Problem 21: Attention Is All You Need — Flops & Parameter Scaling Analysis

For sequence length n and representation dimension d:

- **Self-Attention Layer:** Complexity per layer $= O(n^2 \cdot d)$. Sequential operations $= O(1)$ (Fully parallelizable!). Maximum path length between any two distant tokens $= \mathbf{O(1)}$.
- **Recurrent Layer (RNN/LSTM):** Complexity per layer $= O(n \cdot d^2)$. Sequential operations $= \mathbf{O(n)}$ (Sequential bottleneck!). Maximum path length $= \mathbf{O(n)}$.
- **Convolutional Layer (CNN):** Complexity per layer $= O(k \cdot n \cdot d^2)$. Sequential operations $= O(1)$. Maximum path length $= \mathbf{O(\log_k n)}$ (with dilated convolutions).
- *Conclusion:* Self-attention achieves both minimal path length $O(1)$ and complete parallel execution $O(1)$!

🏛️ Solved Problem 22: Low-Rank Adaptation (LoRA) for Parameter-Efficient LLM Fine-Tuning

In LoRA (Hu et al. 2021), the pretrained weight matrix $\mathbf{W}_0 \in \mathbb{R}^{d \times k}$ is frozen, and weight updates $\Delta\mathbf{W}$ are decomposed into low-rank matrices $\mathbf{B} \in \mathbb{R}^{d \times r}$ and $\mathbf{A} \in \mathbb{R}^{r \times k}$ ($r \ll \min(d, k)$):

$$\mathbf{h} = \mathbf{W_0 x} + \mathbf{\Delta W x} = \mathbf{W_0 x} + \frac{\alpha}{\mathbf{r}} \mathbf{B A x}$$

Reduces trainable fine-tuning parameters by up to **99.9%** (e.g., from 7 billion to 4 million parameters) with zero inference latency overhead (weights are folded into $\mathbf{W} = \mathbf{W}_0 + \frac{\alpha}{r}\mathbf{BA}$)!

🏛️ Solved Problem 31: Sinusoidal Positional Encoding Linear Shift Property Proof

Prove that for any fixed offset k, $PE_{(pos+k)}$ can be represented as a linear function of $PE_{(pos)}$:

$$\sin(\omega(pos + k)) = \sin(\omega \cdot pos)\cos(\omega \cdot k) + \cos(\omega \cdot pos)\sin(\omega \cdot k)$$

$$\cos(\omega(pos + k)) = \cos(\omega \cdot pos)\cos(\omega \cdot k) - \sin(\omega \cdot pos)\sin(\omega \cdot k)$$

$$\mathbf{\begin{bmatrix} PE_{(pos+k, 2i)} \\ PE_{(pos+k, 2i+1)} \end{bmatrix} = \begin{bmatrix} \cos(\omega_i k) & \sin(\omega_i k) \\ -\sin(\omega_i k) & \cos(\omega_i k) \end{bmatrix} \begin{bmatrix} PE_{(pos, 2i)} \\ PE_{(pos, 2i+1)} \end{bmatrix}}$$

Conclusion: The self-attention mechanism can effortlessly learn to attend by *relative* token positions via linear projection!

🏛️ Solved Problem 32: FlashAttention Exact Tiling & IO-Aware Memory Optimization

Dao et al. (2022) observed that standard GPU self-attention is bounded by High Bandwidth Memory (HBM) read/write memory bandwidth rather than FLOPs. **FlashAttention** tiles the Q, K, V matrices into SRAM cache blocks, performing online softmax normalization without writing the massive $N \times N$ attention matrix to HBM, speeding up training by up to $\mathbf{4}\times$ while cutting memory from $O(N^2)$ to $O(N)$!

🏛️ Solved Problem 33: Rotary Position Embedding (RoPE) Formulation

RoPE (Su et al. 2021) encodes positional information directly into Query and Key vectors via complex 2D rotation matrices $\mathbf{R}_{\Theta,m}^{d}$, preserving relative distance under dot-product attention: $\langle \mathbf{R}_m \mathbf{q}, \mathbf{R}_n \mathbf{k} \rangle = g(\mathbf{q}, \mathbf{k}, m-n)$. Standard in modern LLMs (LLaMA, Mistral)!

🏛️ Solved Problem 41: Multi-Query Attention (MQA) & Grouped-Query Attention (GQA)

To reduce KV-Cache memory consumption in large-scale LLMs (LLaMA-2/3, Mistral):

- **Multi-Head Attention (MHA):** h Query heads, h Key heads, h Value heads. (Highest memory consumption).
- **Multi-Query Attention (MQA – Shazeer 2019):** h Query heads share a *single key head* and *single value head*. Reduces KV cache by factor of h ($8\times$ to $32\times$), but can cause slight capacity degradation.
- **Grouped-Query Attention (GQA – Ainslie et al. 2023):** Partitions h Query heads into G groups (e.g., $G=8$), where each group shares 1 Key and 1 Value head. Strikes the optimal Pareto frontier between inference speed and model accuracy!

🏛️ Solved Problem 42: Speculative Decoding for Accelerated LLM Generation

Standard LLM inference is memory bandwidth-bound. **Speculative Decoding (Leviathan et al. 2023)** uses a fast small Draft Model (e.g., 1B parameters) to generate K speculative draft tokens in parallel, which are verified simultaneously in a single forward pass by the large Target Model (e.g., 70B parameters), achieving a $\mathbf{2}\times$ to $\mathbf{3}\times$ inference acceleration with zero mathematical difference in output probability distribution!

🏛️ Solved Problem 50: Transformer Attention FLOPs & Parameter Budgeting Breakdown

For a standard 12-layer BERT-base model ($d_{\text{model}}=768, h=12, d_{\text{ff}}=3072, |V|=30522$):

- Embedding Matrix: $30,522 \times 768 = \mathbf{23.44}$ Million params.
- Per-Layer Self-Attention: $4 \times (768 \times 768) = \mathbf{2.36}$ Million params.
- Per-Layer Feed-Forward: $2 \times (768 \times 3072) = \mathbf{4.72}$ Million params.
- Total params across 12 encoder layers: $12 \times (2.36 + 4.72) = \mathbf{84.95}$ Million params.
- $$\text{Total BERT-Base Model Size: } \mathbf{23.44}\text{M} + \mathbf{84.95}\text{M} + \text{LayerNorms} \approx \mathbf{110} \text{ Million Parameters!}$$

🏛️ Solved Problem 51: Mixture of Experts (MoE) Routing & Load Balancing Loss

In MoE Transformers (Mixtral 8×7B), each token selects top $k=2$ out of $E=8$ expert FFN networks via routing gate $G(x) = \text{Softmax}(\text{TopK}(x \cdot W_g, k))$, achieving $8\times$ capacity with $2\times$ active FLOPs!

🏛️ Solved Problem 58: Sinusoidal Positional Encoding Frequency Allocation Proof

Prove why the geometric progression of wavelengths $\lambda_i = 2\pi \cdot 10000^{2i/d}$ spans from 2π to 20000π, allowing the model to attend simultaneously to immediate neighboring tokens and global document-level structural anchors!

🏛️ Solved Problem 59: Attention Logit Scaling by $\sqrt{d_k}$ Derivation

Show that without the $\frac{1}{\sqrt{d_k}}$ scaling factor, the variance of dot product $q \cdot k$ equals $d_k = 64$, pushing softmax outputs to saturated extremes (0.0 or 1.0) with near-zero gradients!

🏛 Solved Problem 66: Multi-Head Self-Attention Projection Matrix Gradient Derivations

For output loss $\mathcal{L}$, the gradient with respect to Value weight matrix $\mathbf{W}^V$ is $\frac{\partial \mathcal{L}}{\partial \mathbf{W}^V} = \mathbf{X}^T \mathbf{A}^T \frac{\partial \mathcal{L}}{\partial \mathbf{Y}}$ where $\mathbf{A} = \text{softmax}(QK^T/\sqrt{d_k})$.

🏛 Solved Problem 67: Rotary Position Embedding (RoPE) Invariance Proof

Show that $\mathbf{R}_{\Theta,m}^T \mathbf{R}_{\Theta,n} = \mathbf{R}_{\Theta,n-m}$, proving that the attention score between query at position m and key at position n depends purely on relative distance $n - m$!

🏛 Solved Problem 72: Multi-Head Attention Subspace Diversity Proof

Prove that linear projection into h distinct lower-dimensional subspaces $\mathbb{R}^{d_k}$ allows distinct heads to attend simultaneously to different syntactic relations (e.g., Head 1 attends to Direct Objects, Head 2 attends to Prepositional Attachments, Head 3 attends to Coreferent Pronouns)!

🏛 Solved Problem 76: Transformer Positional Interpolation & RoPE Scaling for Long Contexts

In RoPE Linear Scaling (Chen et al. 2023), to extend an LLM from 4k context length to 32k context ($8\times$ scaling), rotation angles θ_i are divided by scale factor $s = 8$: $\tilde{\theta}_i = \theta_i / 8$, preserving frequency dynamics within the pretrained attention kernel!

🏛 Solved Problem 81: FlashAttention SRAM Tiling Block Math

Let SRAM cache size be M. Set block sizes $B_r = \lceil M/(4d) \rceil, B_c = \lceil M/(4d) \rceil$. FlashAttention loads blocks $Q_i \in \mathbb{R}^{B_r \times d}$ and $K_j \in \mathbb{R}^{B_c \times d}$ into SRAM, computing local attention without GPU DRAM roundtrips!

🏛 Solved Problem 82: Cross-Attention vs Self-Attention Matrix Dimensions

In Transformer Decoders: Query matrix $\mathbf{Q} \in \mathbb{R}^{T_{\text{dec}} \times d}$ comes from the decoder hidden states, while Key $\mathbf{K} \in \mathbb{R}^{T_{\text{enc}} \times d}$ and Value $\mathbf{V} \in \mathbb{R}^{T_{\text{enc}} \times d}$ come from encoder representations. The resulting attention matrix $\mathbf{A} \in \mathbb{R}^{T_{\text{dec}} \times T_{\text{enc}}}$ aligns each generated word with relevant source words!

🏛 Solved Problem 87: Sinusoidal Positional Encoding Dot-Product Invariant Proof

Show that $\mathbf{p}_i \cdot \mathbf{p}_j = \sum_{k=1}^{d/2} \cos(\omega_k (i - j))$, proving that the attention similarity between any two positional vectors depends *purely on relative distance* $|i - j|$!

🏛 Solved Problem 88: Transformer Feed-Forward SwiGLU Activation Function

In modern LLMs (LLaMA-3, Mistral), standard ReLU is replaced with **SwiGLU**: $\text{SwiGLU}(x) = \text{Swish}(xW_1) \otimes (xW_3)W_2$ where $\text{Swish}(z) = z \cdot \sigma(\beta z)$, delivering a 1.2% boost on MMLU benchmarks!

🏛️ Solved Problem 93: FlashAttention-2 Work Partitioning & FLOP Utilization

Dao (2023) redesigned FlashAttention loop nesting by swapping the outer loop to iterate over sequence blocks of Q, reducing non-matmul FLOPs and boosting tensor core utilization from 50% to $\mathbf{73\%}$ on Nvidia A100/H100 GPUs!

🏛️ Solved Problem 94: ALiBi (Attention with Linear Biases) Long-Context Extrapolation

Press et al. (2022) replaced positional embeddings with fixed linear slope biases $m = 2^{-8i/h}$: $\text{softmax}(QK^T/\sqrt{d_k} - m \cdot |i - j|)$, training on 2048 tokens while extrapolating to $\mathbf{8192}+$ tokens with zero perplexity increase!

🏛️ Solved Problem 99: RoBERTa vs BERT Dynamic Masking Algorithm Trace

In RoBERTa, duplicating the corpus 10 times and applying 10 independent random 15% masks avoids training on identical static masks across 40 epochs!

🏛️ Solved Problem 103: Transformer Layer Normalization Epsilon Stability

The small scalar $\epsilon = 10^{-5}$ prevents division-by-zero during backpropagation when token activation variance across hidden units collapses to zero!

🏛️ Solved Problem 106: KV-Cache Memory Calculation for 70B LLM

For sequence length $L = 4096$, batch size $B = 1$, 80 layers, $d_{\text{model}} = 8192$, FP16 format:

$$\text{KV-Size} = 2 \times (\text{layers}) \times (\text{seq_len}) \times (\text{hidden_dim}) \times 2 \text{ bytes} = 2 \times 80 \times 4096 \times 8192 \times 2 \approx \mathbf{10.73} \text{ GB RAM!}$$

🏛️ Solved Problem 110: Cross-Attention Sequence Length Complexity

For source sequence length M and target translation length N, cross-attention matrix $\mathbf{A} \in \mathbb{R}^{N \times M}$ computes in $O(N \cdot M \cdot d)$ time, fully parallelizable across all target time steps during training!

🏛️ Solved Problem 111: BERT Pretraining vs Fine-Tuning Convergence Trajectory

Pretraining BERT requires 4 days on 64 TPU chips across 3.3 billion words. Fine-tuning on downstream GLUE classification tasks converges in < 3 epochs within 15 minutes on a single standard consumer GPU!

🏛️ Solved Problem 116: Transformer Positional Encoding Wavelength Geometric Decay Proof

Prove that wavelengths form a geometric progression from 2π to $10000 \cdot 2\pi$, allowing self-attention to resolve both adjacent syntactic token bindings and distant cross-paragraph anaphora!

🏛️ Solved Problem 117: Grouped-Query Attention (GQA) Memory Savings Calculation

In LLaMA-3 70B with $h = 64$ query heads and $G = 8$ KV groups, GQA reduces KV-cache memory bandwidth consumption by $\mathbf{8}\times$, allowing $8\times$ larger batch sizes during generation!

🏛️ Solved Problem 120: Transformer Encoder Self-Attention Computational Complexity

For batch size B, sequence length N, hidden dimension d, and number of heads h: $\mathbf{QK}^T$ computes in $O(B \cdot h \cdot N^2 \cdot \frac{d}{h}) = \mathbf{O(B \cdot N^2 \cdot d)}$ operations. Highly parallelizable across all GPU CUDA thread cores!