MIDDLE EAST TECHNICAL UNIVERSITY

SEMESTER I EXAMINATION 2024-2025

CENG 403 – Deep Learning - Large Language Models & Vision Transformers (University Sources) - ANSWERED

Question 1. Large Language Model Pre-training Strategies (28 marks)

Based on foundational LLM research and university courses on language modeling.

(a) Compare and contrast autoregressive language modeling (GPT-style) versus masked language modeling (BERT-style) pre-training objectives: (12 marks)

Answer: Autoregressive models predict next tokens sequentially, while masked models predict hidden tokens bidirectionally, each with distinct advantages for different tasks.

Autoregressive Language Modeling (GPT-style):

Mathematical Formulation:

$$L_{AR} = -\sum_{t=1}^{T} \log P(x_t | x_{<t}; \theta) = -\sum_{t=1}^{T} \log P(x_t | x_1, ..., x_{t-1}; \theta)$$

Architecture Requirements:

- Causal masking prevents future token access
- Decoder-only transformer architecture
- Unidirectional attention patterns
- Sequential generation capability built-in

Masked Language Modeling (BERT-style):

Mathematical Formulation:

$$L_{MLM} = -\sum_{i \in \mathcal{M}} \log P(x_i | x_{\setminus \mathcal{M}}; \theta)$$

where $\mathcal{M}$ is the set of masked positions.

Architecture Requirements:

- Bidirectional attention (no causal masking)
- Encoder-only transformer architecture
- Random masking strategy (15% of tokens)

MASK token handling during training

Comparative Analysis:

Advantages of Autoregressive:

- Natural generation capability
- No train-test discrepancy
- Excellent for text generation tasks
- Scales well to very large models
- Supports variable-length generation

Advantages of Masked LM:

- Bidirectional context utilization
- Better sentence-level understanding
- Superior performance on classification tasks
- More efficient training (parallel prediction)
- Better representation learning for understanding tasks

Computational Complexity:

- **Autoregressive:** $O(T^2)$ for generation, $O(T)$ per token during training
- **Masked LM:** $O(T)$ for understanding, parallel training across all positions

Task Suitability:

- **Generation tasks:** Autoregressive excels (story writing, code generation)
- **Understanding tasks:** Masked LM excels (classification, NER, QA)

- **Few-shot learning:** Autoregressive shows better in-context learning

(b) Design a hybrid pre-training strategy that combines both autoregressive and masked language modeling. Explain: (10 marks)

Answer: Hybrid strategy uses unified architecture with task-specific attention masks, enabling both generation and understanding capabilities in a single model.

Hybrid Architecture Design:

Unified Transformer Architecture:

- Standard transformer with flexible attention masking
- Task-specific masking patterns during training
- Shared parameters across both objectives
- Prefix-based task identification

Training Procedure:

1. Alternating Objectives:

for each training batch **do**
  Sample task $\sim$ Bernoulli($p = 0.5$)
  if task == "autoregressive" **then**
    Apply causal attention mask
    Compute $L_{AR} = -\sum_{t=1}^{T} \log P(x_t|x_{<t})$
  else
    Apply bidirectional attention with masking
    Compute $L_{MLM} = -\sum_{i \in \mathcal{M}} \log P(x_i|x_{\setminus \mathcal{M}})$
  end if
  Update parameters with respective loss
end for

2. Combined Loss Function:

$$L_{hybrid} = \alpha L_{AR} + (1 - \alpha) L_{MLM}$$

where α balances the objectives.

Implementation Details:

Attention Mask Management:

- Dynamic masking based on task identifier
- Efficient implementation using attention bias
- Memory-efficient mask computation

Task Identification:

- Special tokens: [AR] for autoregressive, [MLM] for masked
- Position-dependent masking patterns
- Gradient isolation for task-specific layers if needed

Expected Benefits:

1. Unified Capabilities:

- Single model for both generation and understanding
- Reduced computational overhead compared to separate models
- Shared representations benefit both tasks

2. Improved Transfer Learning:

- Better downstream task performance
- More robust representations
- Versatile fine-tuning capabilities

3. Training Efficiency:

- Shared computational resources
- Reduced model maintenance overhead
- Better parameter utilization

Challenges and Solutions:

- **Training instability:** Use separate learning rates for each objective
- **Task interference:** Implement gradual curriculum learning
- **Memory overhead:** Use gradient checkpointing and mixed precision

(c) Analyze the scaling laws for language models. Given a computational budget C, derive the optimal allocation between model parameters N, dataset size D, and training compute, considering the relationship: (6 marks)

$$L(N, D) = A + \frac{B}{N^\alpha} + \frac{C}{D^\beta}$$

Answer: Optimal scaling requires balanced growth of parameters and data, with specific allocation ratios determined by the scaling exponents α and β.

Scaling Law Analysis:

Given Relationship:

$$L(N, D) = A + \frac{B}{N^\alpha} + \frac{C}{D^\beta}$$

Where:

- L: Test loss (performance metric)
- N: Number of model parameters
- D: Dataset size (number of tokens)
- A, B, C, α, β: Empirically determined constants

Computational Budget Constraint: Training compute scales as: Compute $\propto N \cdot D$

Given fixed budget: $N \cdot D = \text{constant} = K$

Optimization Problem: Minimize $L(N, D)$ subject to $N \cdot D = K$

Using constraint: $D = K/N$

$$L(N) = A + \frac{B}{N^\alpha} + \frac{C}{(K/N)^\beta} = A + \frac{B}{N^\alpha} + \frac{C \cdot N^\beta}{K^\beta}$$

First-Order Condition:

$$\frac{dL}{dN} = -\alpha \frac{B}{N^{\alpha+1}} + \beta \frac{C}{K^\beta} N^{\beta-1} = 0$$

Solving for Optimal N:

$$\alpha \frac{B}{N^{\alpha+1}} = \beta \frac{C}{K^{\beta}} N^{\beta-1}$$

$$\alpha B K^{\beta} = \beta C N^{\alpha+\beta}$$

$$N^* = \left(\frac{\alpha B K^{\beta}}{\beta C} \right)^{\frac{1}{\alpha+\beta}}$$

Optimal Dataset Size:

$$D^* = \frac{K}{N^*} = \left(\frac{\beta C K^{\alpha}}{\alpha B} \right)^{\frac{1}{\alpha+\beta}}$$

Key Insights:

1. Balanced Scaling:

- Optimal allocation depends on ratio $\frac{\alpha}{\beta}$
- If $\alpha > \beta$: Invest more in model size
- If $\beta > \alpha$: Invest more in data size

2. Empirical Values:

- Typical values: $\alpha \approx 0.076$, $\beta \approx 0.095$
- Suggests slightly more benefit from data scaling
- Optimal ratio: $N^* : D^* \approx 1 : 20$ tokens per parameter

3. Practical Implications:

- Large models require proportionally large datasets
- Computational budget should be balanced between model and data
- Deviating from optimal allocation leads to suboptimal performance

MIDDLE EAST TECHNICAL UNIVERSITY

Question 2. Advanced Training Techniques for Large Language Models (30 marks)

Based on recent advances in LLM training methodologies.

(a) Implement a complete Reinforcement Learning from Human Feedback (RLHF) training pipeline: (15 marks)

Answer: Complete RLHF pipeline with supervised fine-tuning, reward model training, and PPO optimization with KL regularization.

Phase 1: Supervised Fine-Tuning (SFT)

function SupervisedFineTuning(base_model, demonstrations)
// Load pre-trained model
$\pi^{SFT} \leftarrow$ base_model

// Prepare demonstration data
$\mathcal{D}_{demo} \leftarrow \{(x_i, y_i)\}$ where y_i are human demonstrations

// Standard supervised training
for epoch in training_epochs **do**
  for batch in $\mathcal{D}_{demo}$ **do**
    $L_{SFT} \leftarrow -\mathbb{E}_{(x,y)\sim batch}[\log \pi^{SFT}(y|x)]$
    Update π^{SFT} using gradient descent on L_{SFT}
  end for
end for
return π^{SFT}

Phase 2: Reward Model Training

function TrainRewardModel(π^{SFT}, preference_data)
// Initialize reward model
$r_\phi \leftarrow$ RewardModel(base_architecture)

// Prepare pairwise preference data
$\mathcal{D}_{pref} \leftarrow \{(x, y_w, y_l)\}$ where $y_w \succ y_l$

for epoch in reward_training_epochs **do**
  for batch in $\mathcal{D}_{pref}$ **do**
    // Compute reward scores
    for (x, y_w, y_l) in batch **do**
      $r_w \leftarrow r_\phi(x, y_w)$

$r_l \leftarrow r_\phi(x, y_l)$
end for
// Bradley-Terry loss
$L_{reward} \leftarrow -\mathbb{E}[\log \sigma(r_w - r_l)]$
Update r_ϕ using gradient descent on L_{reward}
end for
end for
return r_ϕ

Phase 3: PPO Training with KL Regularization

function PPOTraining(π^{SFT}, r_ϕ, prompts)
// Initialize policy and reference model
$\pi_\theta \leftarrow \pi^{SFT}$ (trainable copy)
$\pi_{ref} \leftarrow \pi^{SFT}$ (frozen reference)

for iteration in ppo_iterations **do**
  // Sample rollouts
  $\mathcal{B} \leftarrow \{\}$
  for prompt x in prompts_batch **do**
    $y \sim \pi_\theta(\cdot|x)$ // Generate response
    $r \leftarrow r_\phi(x, y)$ // Get reward
    kl $\leftarrow \log \pi_\theta(y|x) - \log \pi_{ref}(y|x)$
    advantage $\leftarrow r - \beta \cdot$ kl // KL penalty
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{(x, y, r, \text{advantage})\}$
  end for

  // PPO update
  for ppo_epoch in range(K) **do**
    for minibatch in $\mathcal{B}$ **do**
      // Compute probability ratios
      ratio $\leftarrow \frac{\pi_\theta(y|x)}{\pi_{old}(y|x)}$
      // Clipped objective
      clip_ratio $\leftarrow$ clip(ratio, $1 - \epsilon, 1 + \epsilon$)
      $L_{clip} \leftarrow \min(\text{ratio} \cdot \text{advantage}, \text{clip\_ratio} \cdot \text{advantage})$
      // Combined loss with KL penalty
      $L_{total} \leftarrow -L_{clip} + \beta \cdot \text{KL}(\pi_\theta || \pi_{ref})$
      Update π_θ using gradient descent
    end for

29: **end for**
30: **end for**
31: **return** π_θ

Key Implementation Details:

KL Divergence Regularization:

$$\text{KL}(\pi_\theta || \pi_{ref}) = \mathbb{E}_{x,y}[\log \pi_\theta(y|x) - \log \pi_{ref}(y|x)]$$

Hyperparameter Settings:

- KL penalty coefficient: $\beta = 0.02$
- PPO clip ratio: $\epsilon = 0.2$
- PPO epochs per iteration: $K = 4$
- Advantage normalization for stability