Machine Learning (CE 40717)

Fall 2024

Ali Sharifi-Zarchi

CE Department
Sharif University of Technology

November 30, 2024

Language Moldels

- Definition of a Language Model (LM)
  - A machine learning model designed to predict and generate plausible language by analyzing text patterns.
  - Operates by learning from large amounts of text data to understand linguistic structures, context, and vocabulary.
    - Common example: Autocomplete in text typing

Language Moldels

- Purpose of Language Models
  - Estimate the probability of a word (token) or sequence of words within a longer text sequence.
  - Aim to understand and predict context in sentences.

Language Moldels

Example of Language Model Prediction

- When I hear rain on my roof, I _______ in my kitchen.
  Potential predictions with probabilities:
  - "cook soup" - 9.4%
  - "warm up a kettle" - 5.2%
  - "cower" - 3.6%
  - "nap" - 2.5%
  - "relax" - 2.2%

Large Language Moldels

- What are Large Language Models (LLMs)?
  - Large Language Models (LLMs) are powerful AI systems trained on vast text data, designed to understand and generate human language at scale.
  - LLMs can process entire sentences, paragraphs, and documents, capturing complex language patterns and context.
- Size and Parameters of LLMs
  - "Large" refers to models with a high number of parameters, which are weights learned during training.
    - Examples include BERT (110 million parameters) and PaLM 2 (up to 340 billion parameters).

Pre-training and adaptation

- Pre-training
    - Models are initially trained on massive datasets to learn general language patterns, grammar, and common knowledge.
    - The pre-trained model is further trained on a smaller, task-specific dataset to specialize in a particular application (e.g., sentiment analysis, translation).
- Adaptation
    - The pre-trained model is further trained on a smaller, task-specific dataset to specialize in a particular application (e.g., sentiment analysis, translation).
    - Fine-tuning tailors the model's responses and predictions to perform effectively on the target task.

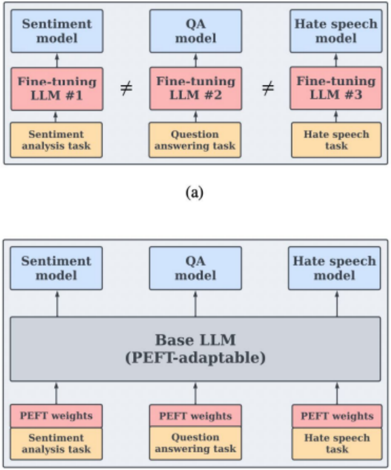
Pre-training and adaptation

Figure 1: Overview of Model Training: Pre-training on large, unlabeled data builds foundational knowledge, while fine-tuning on smaller, labeled datasets adapts the model for specific tasks.

Parameter-Efficient Fine-Tuning (PEFT)

- **Targeted Adaptation:** PEFT involves fine-tuning a small subset of model parameters, allowing the pretrained model to adapt to specific tasks without retraining the entire model.
- **Efficiency Gains:** By preserving the majority of the pretrained model's structure, PEFT significantly reduces training time, memory usage, and computational cost.
- **Scalability:** PEFT enables large language models to be applied effectively to a wide range of tasks, making it feasible to use large models in resource-constrained environments.

- Fine-tuning an LLM for a specific downstream task
    - (a) illustrates vanilla fine-tuning, which requires updating the entire model, resulting in a new model for each task.
    - (b) PEFT instead learns a small subset of model parameters for each task with a fixed base LLM. The same base model can be re-used during inference for different tasks.

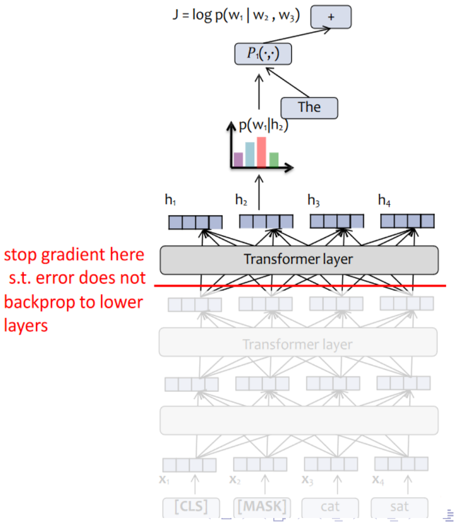


(a)

(b)

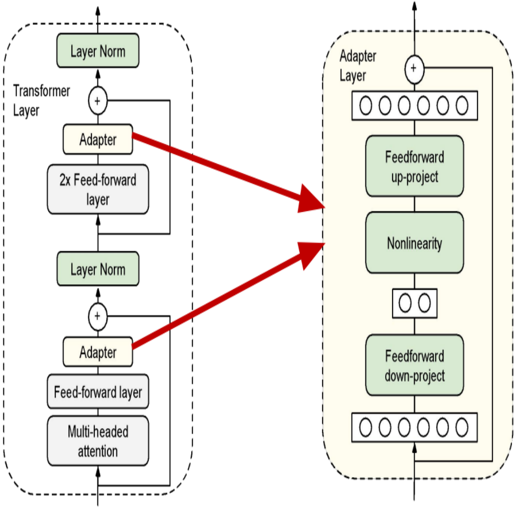
Fine-Tuning the Top Layers Only

- **PEFT Baseline Efficiency:** Freeze parameters except top K layers, reducing computation and memory usage.
- **Flexible Application:** Can be applied to most deep neural networks for efficient fine-tuning.



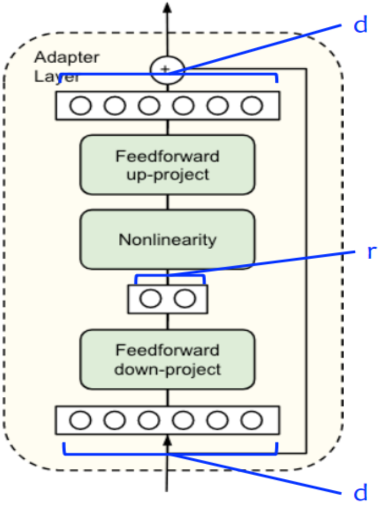
Adapters

- **Adapters:** New modules inserted between layers of a pre-trained model, with the original model weights fixed.
- **Training Efficiency:** Only adapter modules are tuned, initialized to ensure their output resembles that of the original model.



Adapters

- **Adapter Design:** Feedforward layer with one hidden layer and a residual connection.
- **Bottleneck Architecture:** Input and output dimensions are d, with a reduced dimension r in the middle.
- **Initialization:** Near-identity initialization with a skip connection and near-zero weights.



Compacters

- **Compacters:** are an extension of adapters which aim to make the technique even more efficient.
- Adapters are standard fully connected layers.
  - Linear project to lower dimension followed by nonlinearity, followed by projection back up to original dimension.
  - $y = \boldsymbol{W_2} GELU(\boldsymbol{W_1 x} + \boldsymbol{b_1}) + \boldsymbol{b_2}$
- The compacter replaces the fully connected layer with a parameterized hypercomplex multiplication layer.
  - Each $\boldsymbol{W}$ is learned as a sum of n Kronecker products.
  - n is a user-specified hyperparameter.
- Compacters reduce the number of parameters in the adapter layer to $\frac{1}{n}$ without harming the performance.

Compacters

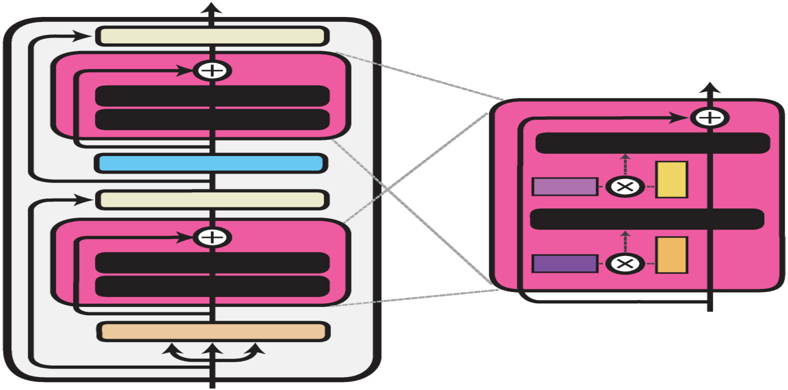


Figure 2: The compacter replaces the linear down and up projection of the bottleneck adapter with a phm layer. The phm layer obtains its weights by computing the kronecker product of two smaller matrices.

BitFit

- **Bias Fine-Tuning:** Only fine-tune the bias terms and final classification layer, reducing the number of parameters updated (<1% of the model).
- **Implementation:** Use a custom optimizer to fine-tune only the bias parameters by selecting them from the model's named parameters, but this approach may fail with large models.
- Fail when model size is large

- Recall the equations for multi-head attention

$$Q^{m,\ell}(x) = W_q^{m,\ell} x + b_q^{m,\ell}$$

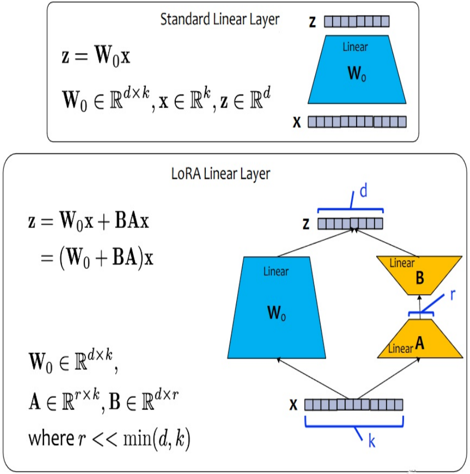
$$K^{m,\ell}(x) = W_k^{m,\ell} x + b_k^{m,\ell}$$

$$V^{m,\ell}(x) = W_v^{m,\ell} x + b_v^{m,\ell}$$

  - ℓ is the layer index
  - m is the attention head index
  - Only the bias terms are updated.

LoRA

- **Key Idea:** Keep the original pretrained parameters $\boldsymbol{W_0}$ fixed and learn an additive modification $\Delta \boldsymbol{W}$ via low-rank decomposition $\Delta W = BA$, where $\boldsymbol{BA}$ has rank r, much smaller than k and d.
- **LoRA Linear Layer:** The modification $\Delta W = BA$ is added to the original linear transformation, where $\boldsymbol{A}$ and $\boldsymbol{B}$ have much smaller dimensions than the original weight matrix $\boldsymbol{W_0}$.



LoRA

Initialization

- We initialize the trainable parameters:

$$A_{ij} \sim \mathcal{N}(0, \sigma^2) \quad \forall i, j$$

- $B = 0$
- Thus, at the start of fine-tuning, the parameters have their pretrained values:

$$\Delta W = BA = 0$$

$$W_0 + BA = W_0$$

LoRA

Hot Swapping Parameters

- $\boldsymbol{W_0}$ and $\boldsymbol{BA}$ have the same dimension, so we can "swap" the LoRA parameters in and out of a Standard Linear Layer.
- To get a Standard Linear Layer with parameters $\boldsymbol{W}$ that includes our LoRA fine-tuning:

$$W \leftarrow W_0 + BA$$

- To remove the LoRA fine-tuning from that Standard Linear Layer:

$$W \leftarrow W - BA = W_0$$

- If we do LoRA training on two tasks such that the parameters $\boldsymbol{B'A'}$ are for task 1 and $\boldsymbol{B''A''}$ are for task 2, then we can swap back and forth between them.

LoRA

- **LoRA Linear Layers:** LoRA linear layers can replace every linear layer in the Transformer model.
- **Original Paper Focus:** The original LoRA paper specifically applies LoRA only to the attention weights (Q, K, V).
- **Mathematical Formulation:**

$$Q = LoRALinear(X; W_q, A_q, B_q)$$

$$K = LoRALinear(X; W_k, A_k, B_k)$$

$$V = LoRALinear(X; W_v, A_v, B_v)$$

QLoRA

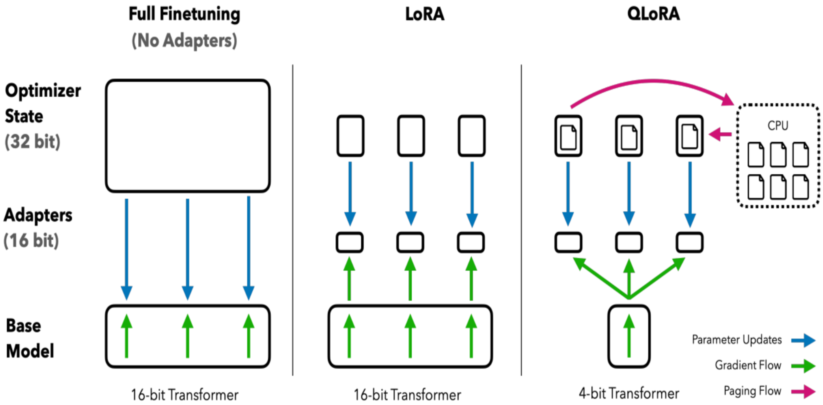
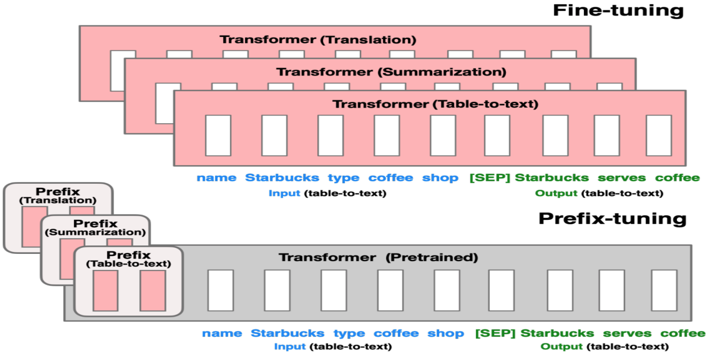


Figure 3: Different finetuning methods and their memory requirements. QLORA improves over LoRA by quantizing the transformer model to 4-bit precision and using paged optimizers to handle memory spikes.

Prefix Tuning

- Freeze all pretrained parameters and prepend trainable prefix tokens to the input and hidden activations.
- The prefix is processed by the model like real words, allowing each batch element to run a different tuned model during inference.

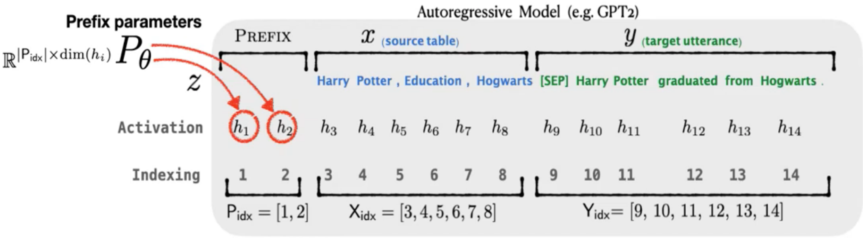


Prefix Tuning

$$h_i = \begin{cases} P_\theta[i,:], & \text{if } i \in \mathsf{P}_{\text{idx}}, \\ \text{LM}_\phi(z_i, h_{<i}), & \text{otherwise.} \end{cases}$$

$$\max_\theta \log p_{\phi,\theta}(y \mid x) = \sum_{i \in \mathsf{Y}_{\text{idx}}} \log p_{\phi,\theta}(z_i \mid h_{<i})$$

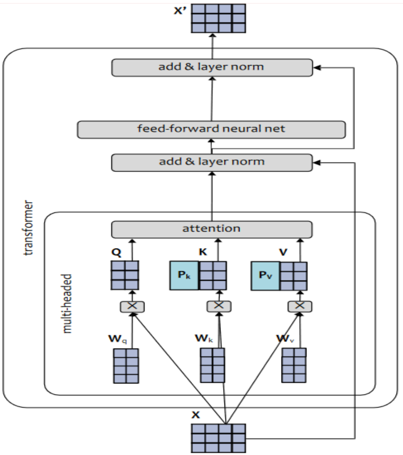
freeze LM parameters ϕ
update prefix parameters θ



Prefix Tuning

Prefix Tuning with Multi-Head Attention

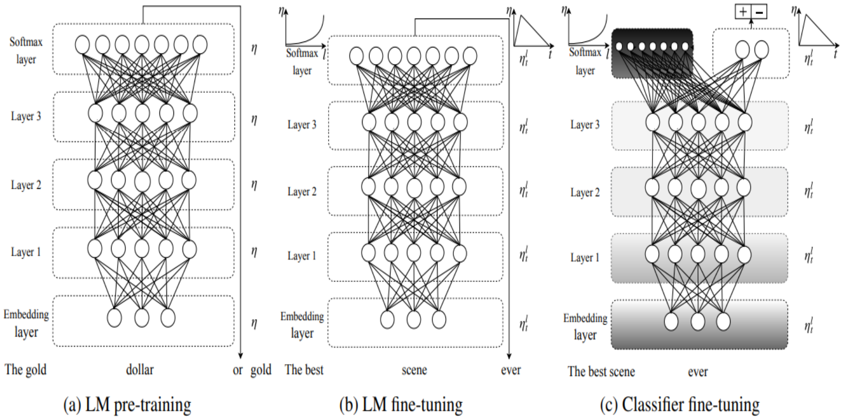
- The model uses prefix tokens (P_k and P_v) along with the attention matrices (Q, K, V) to guide the attention mechanism in the multi-headed transformer.
- The prefix tokens are processed and combined with the query, key, and value matrices (W_q, W_k, W_v) to form the final attention mechanism in the transformer architecture.



Universal Language Model Fine-Tuning Principle (ULMFit)

- **ULMFiT Overview:** Universal Language Model Fine-tuning (ULMFiT) is a breakthrough in NLP that allows pre-trained language models to be adapted to various tasks with minimal data and improved performance.
- **Key Concepts**
  - **Transfer Learning:** Utilizes knowledge from one task to improve performance on related tasks.
  - **Language Model Pre-training:** The model is pre-trained on large text corpora to understand general language structure before task-specific fine-tuning.
  - **Discriminative Fine-Tuning:** Different layers of the model are fine-tuned at varying rates to prevent catastrophic forgetting and improve task adaptation.
  - **Gradual Unfreezing:** Fine-tuning begins with the top layers, gradually unfreezing earlier layers to retain general language knowledge.
  - **Slanted Triangular Learning Rates:** Adjusts learning rate by first increasing it and then slowly decreasing, optimizing fine-tuning.

Universal Language Model Fine-Tuning Principle (ULMFit)



ULMFiT consists of three stages: a) The LM is trained on a general-domain corpus to capture general features of the language in different layers. b) The full LM is fine-tuned on target task data using discriminative fine-tuning ('Discr') and slanted triangular learning rates (STLR) to learn task-specific features. c) The classifier is fine-tuned on the target task using gradual unfreezing, 'Discr', and STLR to preserve low-level representations and adapt high-level ones (shaded: unfreezing stages; black: frozen).

Universal Language Model Fine-Tuning Principle (ULMFit)

- **Mathematical Concepts**
  - **Neural Networks:** ULMFiT uses Long Short-Term Memory (LSTM) networks to process text sequences.
  - **Embeddings:** Words are represented as vectors in a high-dimensional space, capturing semantic relationships.
  - **Gradient Descent:** Optimization method used to minimize errors by adjusting model parameters.
  - **Learning Rate:** The learning rate is dynamically adjusted during training to optimize learning speed and accuracy.

Contribution

- **These slides were prepared with contributions from:**
  - Amirhossein Akbari

Any Questions?