CSC290/420 Machine Learning Systems for Efficient AI
Training - III (Finetuning)

Sreepathi Pai

November 3, 2025

URCS

Outline

ML Training before Finetuning

Full Fine-tuning

Adapters

Prefix Tuning

LoRa

Newer Techniques

Outline

ML Training so far (in the course)

- Start with random weights
- For each training example, compute output (forward mode)
- Calculate and back-propagate loss
- Repeat until convergence

Task-specific training

- Do different tasks require different models?
  - Summarization
  - Natural language generation
  - Translation
  - Question answering
  - etc.
- Until about a decade ago, thinking was yes.

Costs

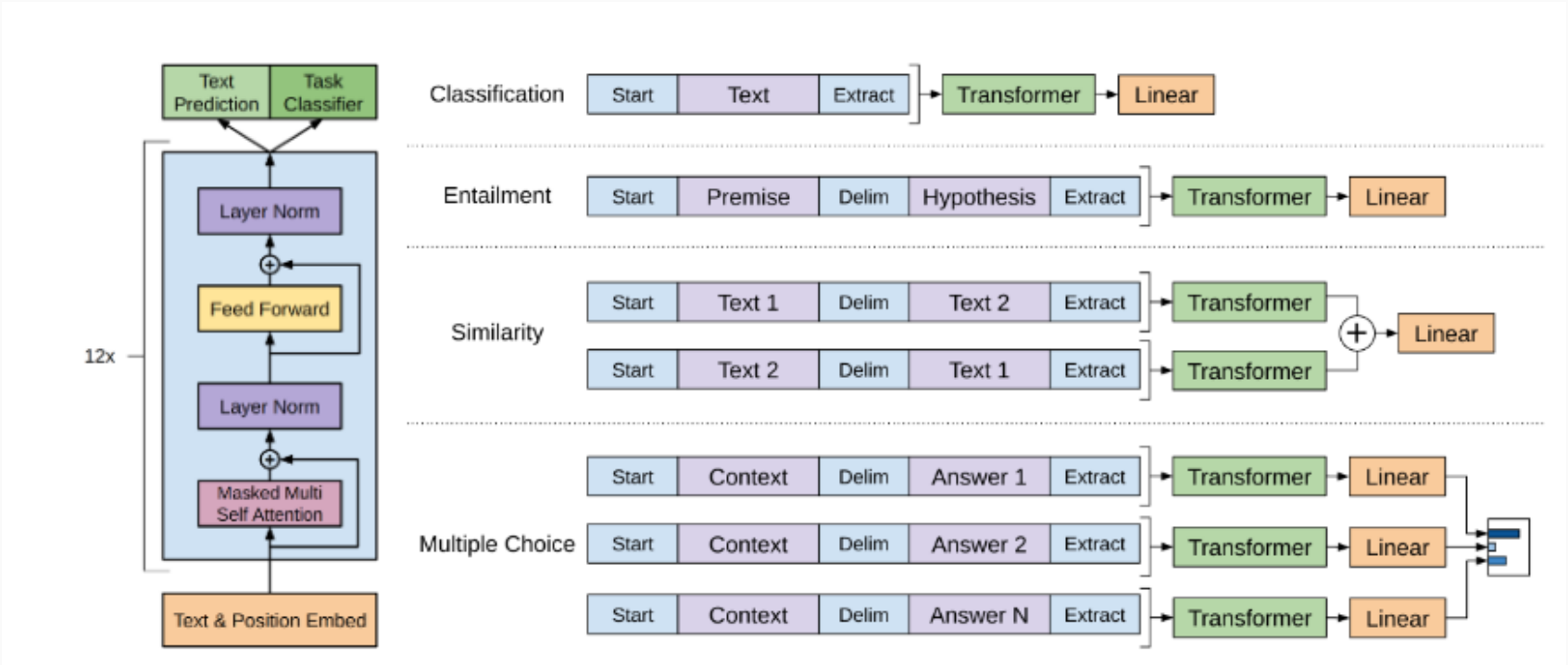
- Training a model for a task from scratch is expensive
- Deployment/Inference requires entire new set of task-specific weights

Outline

GPT

- GPT (the original) broke up training into two phases
  - Pre-training: General training with large dataset
  - Fine-tuning: Task-specific training

Radford et al., 2018, Improving Language Understanding by Generative Pre-Training



Pre-Training and Fine-tuning

- Pre-train for a general task
  - Once per model
  - Resembles usual training
- (Full) Fine-tuning
  - Start training with *pre-trained weights*
  - Once per task
  - (sometimes also called model tuning)

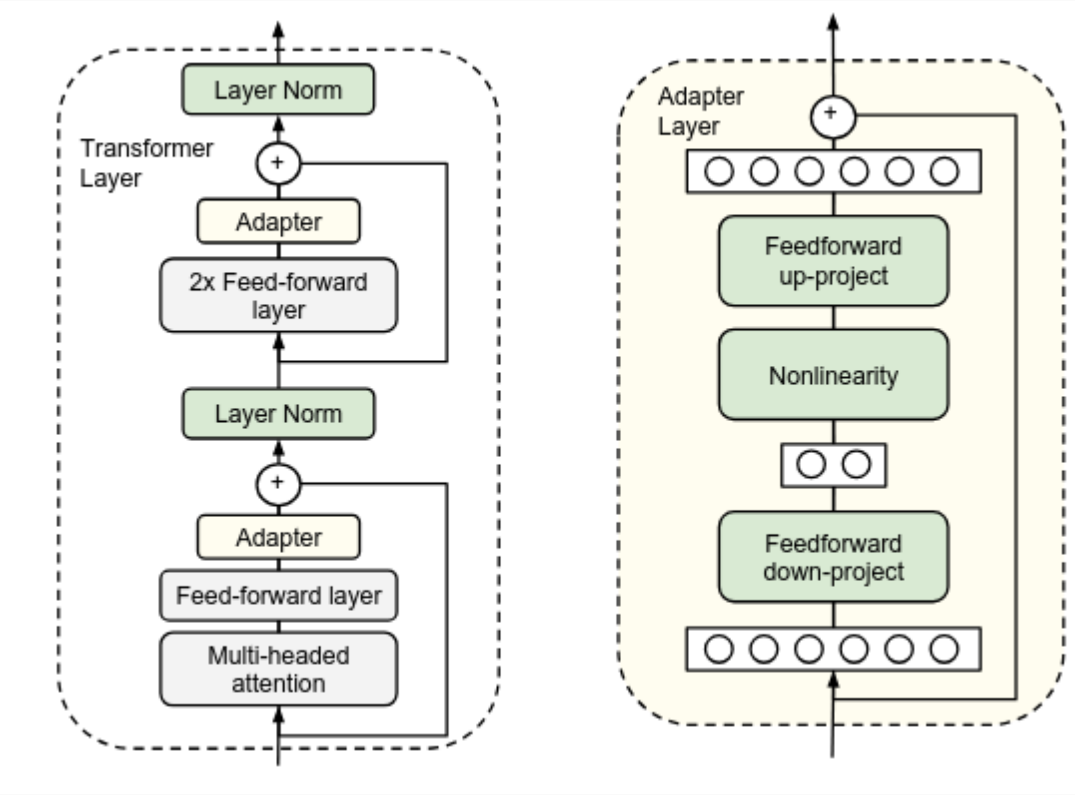
Costs

- A checkpoint is the parameters of a model
- For full-fine tuning, the checkpoint is the same size as the original model
- Alternatives:
  - Let tasks share some layers
  - I.e., task-specific training only modifies weights in some layers, not all

Outline

NLP Adapters

- Let $\psi_w(x)$ be the original function based on the weights w
- An adapter composes a function $\phi_{w,v}(x)$ where w is original weights
  - And v_0 is chosen so that the two functions are approximately equal
- During task-specific training only v is trained, not w



Houlsby et al., 2019, Parameter-Efficient Transfer Learning for NLP

Costs

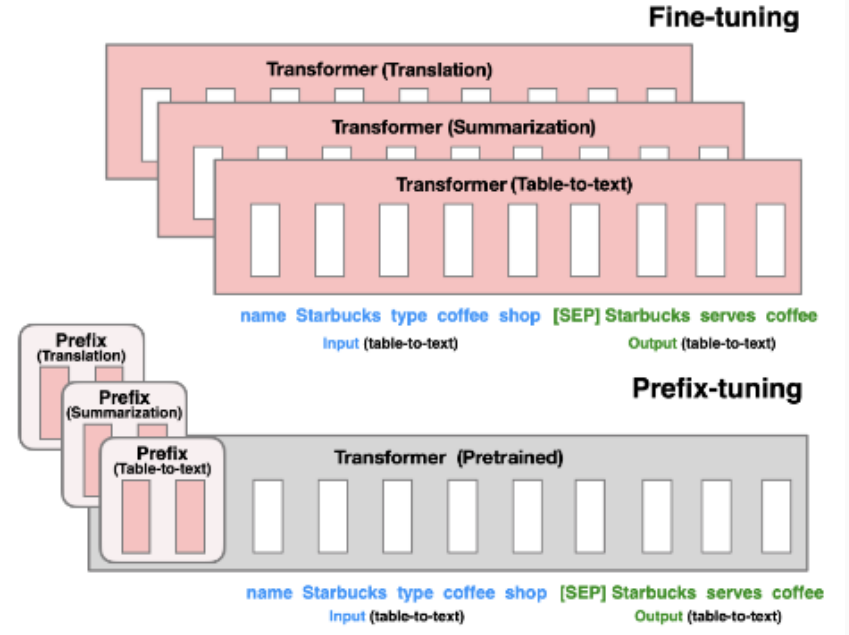
- $|v|$ is chosen to be much smaller than $|w|$
- So training is "cheap"
    - Much fewer parameters trained
- Inference requires adapter layers to run
    - Higher latency
- However, weights in adapter layers can be swapped out as needed for different task
- Uses about 30% more parameters than BERT

Outline

Prefix Tuning

- Add a set of "prefix" parameters before the input and encoding layers, P_θ
- During fine-tuning, train these prefix parameters

Li and Liang, Prefix-Tuning: Optimizing Continuous Prompts for Generation



Costs

- During training [finetuning], prefix is implemented as a MLP
  - produces a prefix P_θ
- During inference
  - load task-specific P_θ
  - appears as a longer context/prefix/prompt
- Uses about 0.1% task-specific parameters (based on GPT-2)

Prompt Engineering / Zero-shot

- GPT-3 and newer excel at "zero-shot" learning
- e.g., Summarization by simply suffixing "TLDR" to prompt.
- No retraining/fine-tuning required

Outline

Low-rank

- Add a LoRa module that contains two matrices A and B
- During original training, B is 0.
- During fine-tuning, A and B are trained
- Weight $h = W_0 x + BAx$
- Crucially, $A \in \mathbb{R}^{r \times k}$ and $B \in \mathbb{R}^{d \times r}$ with $r \ll \min(d, k)$

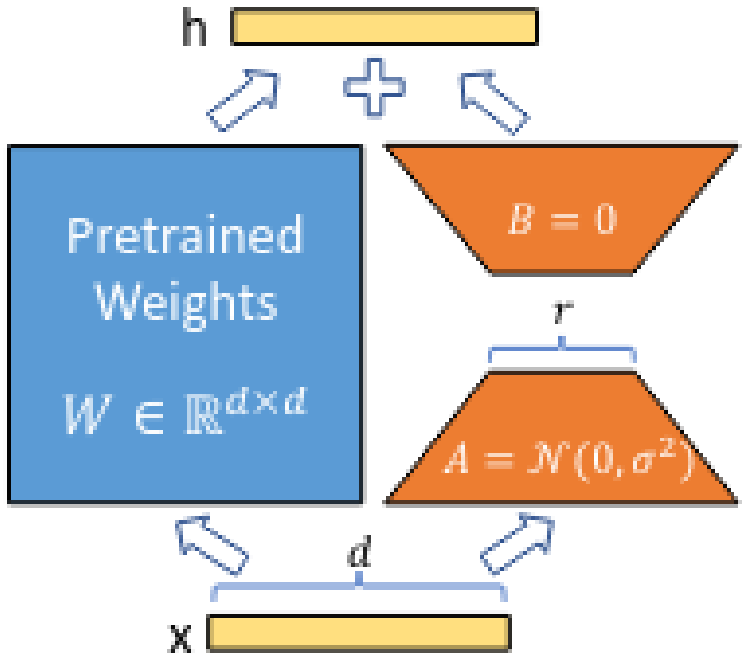


Figure 1: Our reparametrization. We only train A and B.

Hu et al., LoRA: Low-Rank Adaptation Of Large Language Models

Architecture

- During (original) training, the LoRa updates can be ignored
  - Multiplying with 0
- During fine-tuning with LoRa, only A and B are tuned to task
- Uses 0.1% of original parameters (35MB vs 350GB for GPT-3)

Deployment

- Inference uses different A and B for each task
- However, these weights don't need to be loaded separately
    - as in adapters
- Instead, the weights of the models W_0 can be combined with BA

Costs

- Trains less than 0.1% of model's parameters
- No extra cost during inference
- Alpaca 7B finetuned (full) cost about $600
- Alpaca 7B-LoRa used a single GPU and a few hours

Outline

Representation Tuning

- Techniques so far have been dubbed Parameter Efficient Finetuning
- Instead of changing parameters, change representations they encode
- Based on work that seeks to interpret LLMs
- Billed as Drop-in replacements for PEFT techniques
- Wu et al, ReFT: Representation Finetuning for Language Models