Bang for Your Buck

Parameter-Efficient Fine Tuning on a Budget

Robert Mueller
Taylor Hamilton

Berkeley
UNIVERSITY OF CALIFORNIA

Agenda

1 Introduction

2 Experimental Details

3 Results & Discussion

4 Future Work

Background

Decoder-only LLMs are

- Fine-tunable: they can be adapted to a huge variety of NLU tasks
- Getting big: 100s of billions of parameters

Parameter-Efficient Fine Tuning can significantly reduce tuning time, but it can be time- and cost-prohibitive to experiment with different PEFT methods and hyperparameters

Question

What can PEFT accomplish with $1?

Our task

Explore the ability of different PEFT methods, with different hyperparameters, to fine tune an LLM for an NLU classification task on a $1 budget

Our hypothesis

PEFT will demonstrate a modest effect – just a bit better than a baseline of guessing the most common class

One Dollar of Tuning

Specs

- 8 vcpu, 30GB RAM (n1-standard-8)
- NVidia v100 GPU
- 250GB Balanced Persistent Disk

Assumptions

- Google Cloud
- On-demand Pricing
- us-central1 region

~ 20 minutes

Fine Tuning Task: SuperGLUE boolq

Structure
Passage
yes/no Question
Answer (Label)

Example
Passage: "I like all vegetables."
Question: "Do I like carrots"
Label: "Yes"

Stats
12697 records
74-13-13 train-val-test
61% 'Yes'

Convert to Prompt

Using information in the passage, answer the following question: <question>? ### Passage: <passage> ### Answer:

Baselines

Base Model
Open Pre-Trained Transformers (OPT)
From Facebook

Num Parameters
125 million
350 million
2.7 billion

Un-tuned Accuracy
.06%

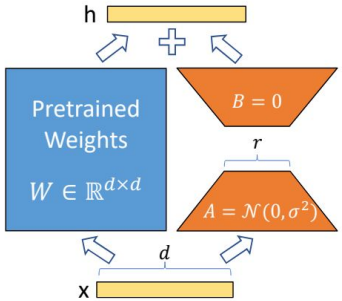
All-yes Accuracy
61%

$1 Fine-Tuned Accuracy
0%

PEFT Methods

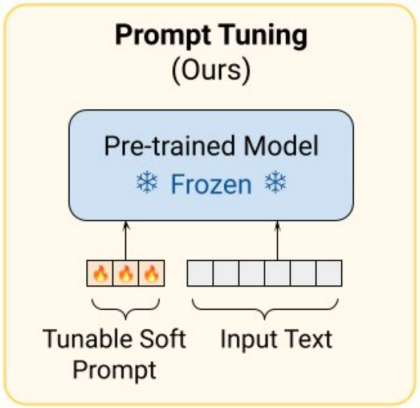
LoRA: Low Rank Adaptation of Large Language Models

inject trainable rank-decomposition matrices into the transformer architecture



Prompt Tuning

Prepend trainable prompts



P-Tuning

Augment word-based prompts with trainable prompts

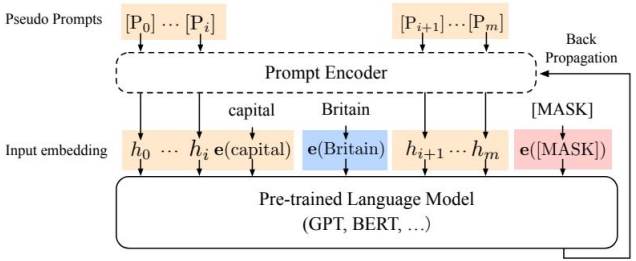


Image credits

1. LoRA paper
2. Jeff Dean (twitter)
3. P-Tuning paper

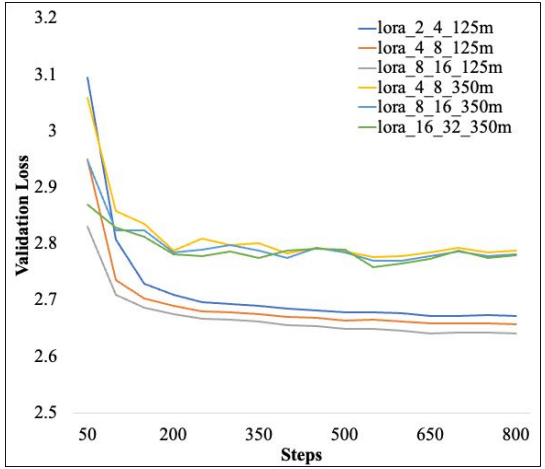
Raw Results

Models		Accuracy	Balanced Accuracy	Training Steps	Tuning Cost
Baselines	opt-125m	0.06%	0.05%	-	-
	all-yes-base	**61.28%**	**50.00%**	-	-
	full-fine-tune	0.00%	0.00%	2300	$0.97
OPT 125M Models	lora_2_4_125m	52.29%	42.69%	800	$0.93
	lora_4_8_125m	56.33%	48.17%	800	$0.93
	lora_8_16_125m	**60.61%**	**49.57%**	800	$0.92
	prompt_2_125m	0.73%	0.60%	900	$0.98
	prompt_48_125m	42.20%	38.68%	850	$0.95
	prompt_96_125m	25.02%	22.07%	850	$0.96
	prompt_192_125m	59.57%	49.16%	850	$1.05
	p-tune_48_64_125m	0.12%	0.13%	850	$0.93
	p-tune_2_128_125m	0.37%	0.30%	800	$0.87
	p-tune_10_128_125m	0.61%	0.56%	800	$0.86
OPT 350M Models	lora_4_8_350m	29.66%	24.70%	800	$0.91
	lora_8_16_350m	53.76%	45.17%	800	$0.91
	lora_16_32_350m	44.95%	39.53%	800	$0.91

Why Doesn't LoRA do better on the 350m model?

Our take:

- It's not undertraining ➡
- It's trying harder to actually learn something ⬇️



Models	Accuracy	Balanced Accuracy	'Yes' Predictions	'No' Predictions	Other Predictions
lora_4_8_125m	56.33%	48.17%	93%	4%	3%
lora_4_8_350m	29.66%	24.70%	35%	17%	48%

So how good is $1 of PEFT? (on boolq on opt-125m)

Not good.
Honestly,
pretty bad.

Where can we go from here?

Observation	Opportunity
We weren't undertraining	Newer, bigger models
Some highly questionable results (like full fine tuning's 0% accuracy)	Average multiple training runs
It's hard even to overcome the hurdle of determining the classes	Smarter initial weights

Thank you