Instruction Fine-tuning using Alpaca
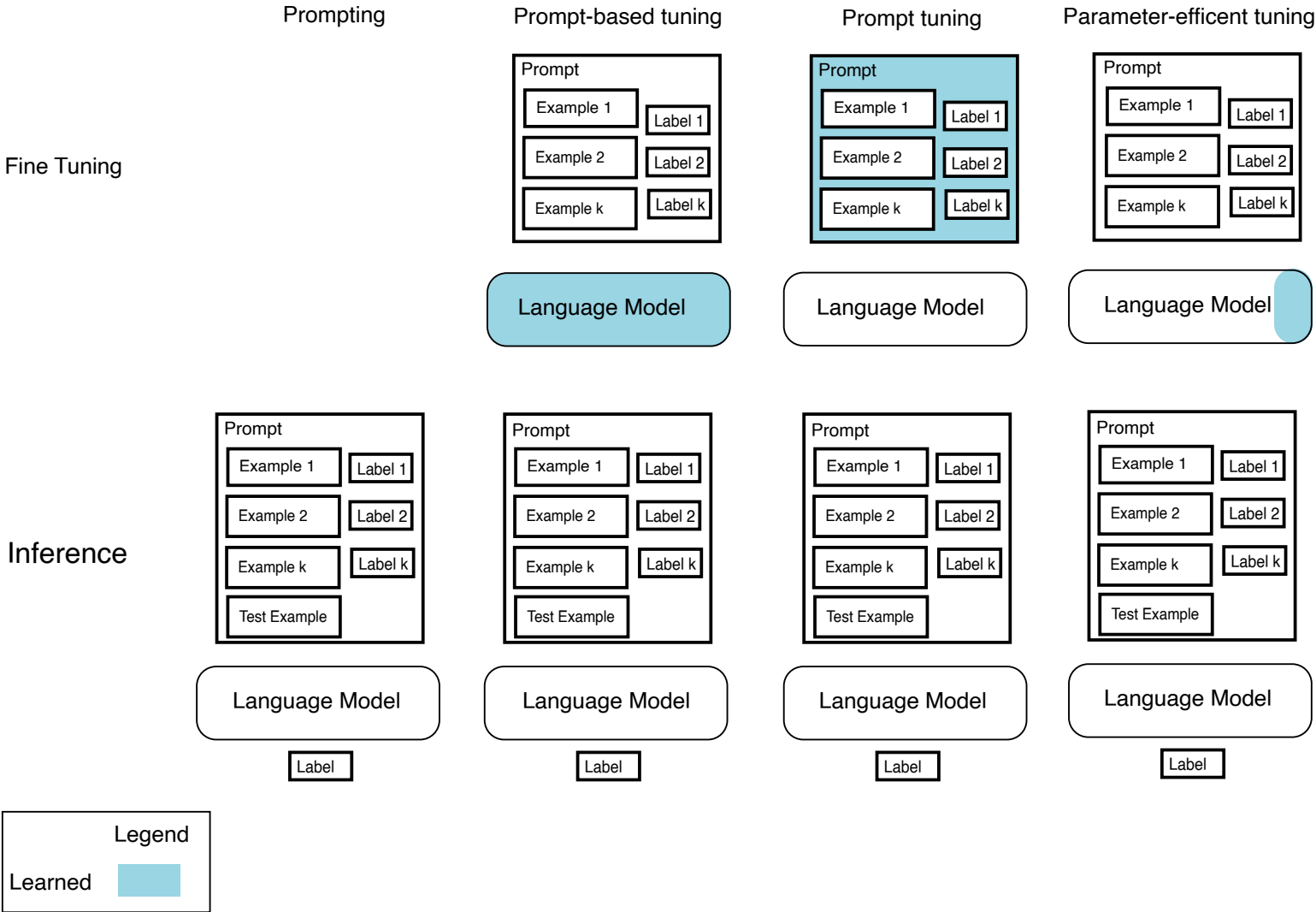# Best Practices and Learned Lessons

Yamen Ajjour

Motivation

Large language models are
few shot and multitask learners

Outlines
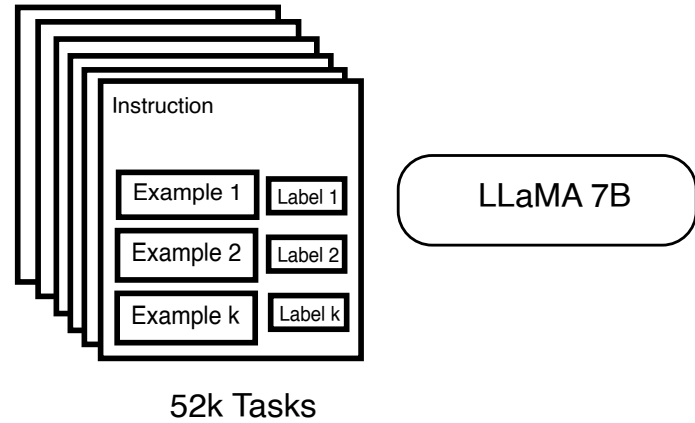
- Prompt-based learning methods
- Instruction fine-tuning
- LoRa: Low-Rank Adaptation of Large Language Models
- Use Case: Instruction fine-tuning on Sentiment Analysis using Alpaca
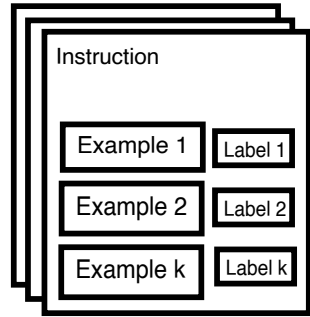- Lessons Learned

Prompt-based learning

	Prompting	Prompt-based tuning	Prompt tuning	Parameter-efficent tuning

Fine Tuning

Prompt-based tuning:
Prompt
Example 1 — Label 1
Example 2 — Label 2
Example k — Label k
Language Model

Prompt tuning:
Prompt
Example 1 — Label 1
Example 2 — Label 2
Example k — Label k
Language Model

Parameter-efficent tuning:
Prompt
Example 1 — Label 1
Example 2 — Label 2
Example k — Label k
Language Model

Inference

Prompting:
Prompt
Example 1 — Label 1
Example 2 — Label 2
Example k — Label k
Test Example
Language Model
Label

Prompt-based tuning:
Prompt
Example 1 — Label 1
Example 2 — Label 2
Example k — Label k
Test Example
Language Model
Label

Prompt tuning:
Prompt
Example 1 — Label 1
Example 2 — Label 2
Example k — Label k
Test Example
Language Model
Label

Parameter-efficent tuning:
Prompt
Example 1 — Label 1
Example 2 — Label 2
Example k — Label k
Test Example
Language Model
Label

Legend
Learned

Instruction Fine-tuning

Prompt fine-tune

Instruction

Example 1	Label 1
Example 2	Label 2
Example k	Label k

LLaMA 7B

52k Tasks

Instruction Fine-tuning

Prompting

Prompt fine-tune

Instruction

Example 1	Label 1
Example 2	Label 2
Example k	Label k

175 Tasks

Generate

GPT3

$\Longrightarrow$

Instruction

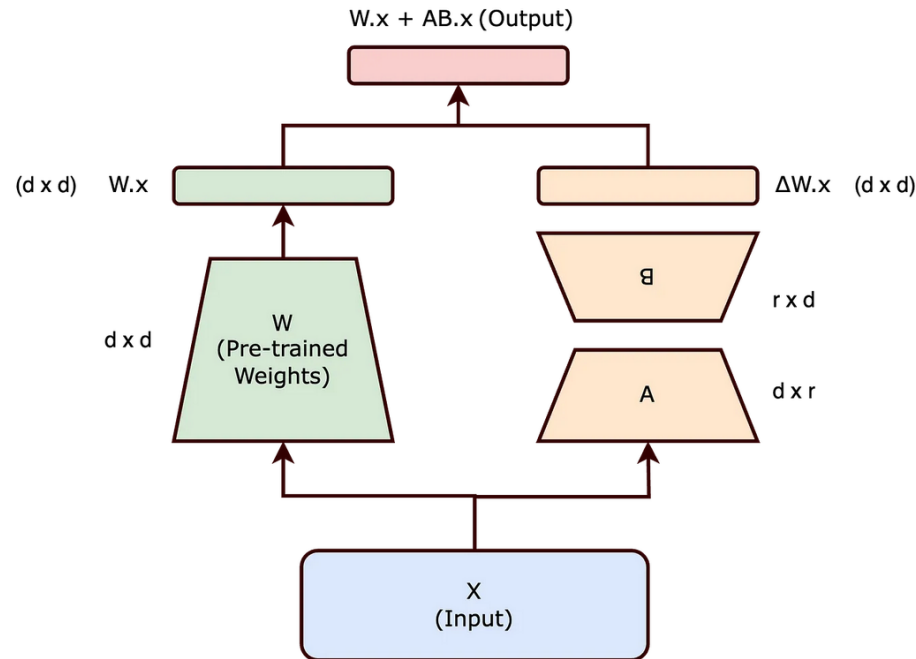| Example 1 | Label 1 |
| Example 2 | Label 2 |
| Example k | Label k |
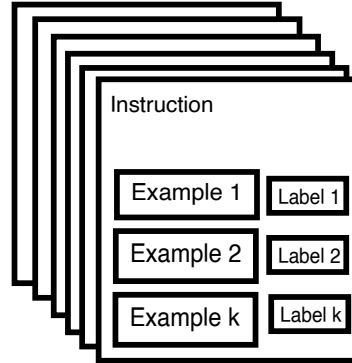
52k Tasks

Alpaca 7B

LoRa: Low-Rank Adaptation of Large Language Models

$$h = W_0 x + \Delta W x$$
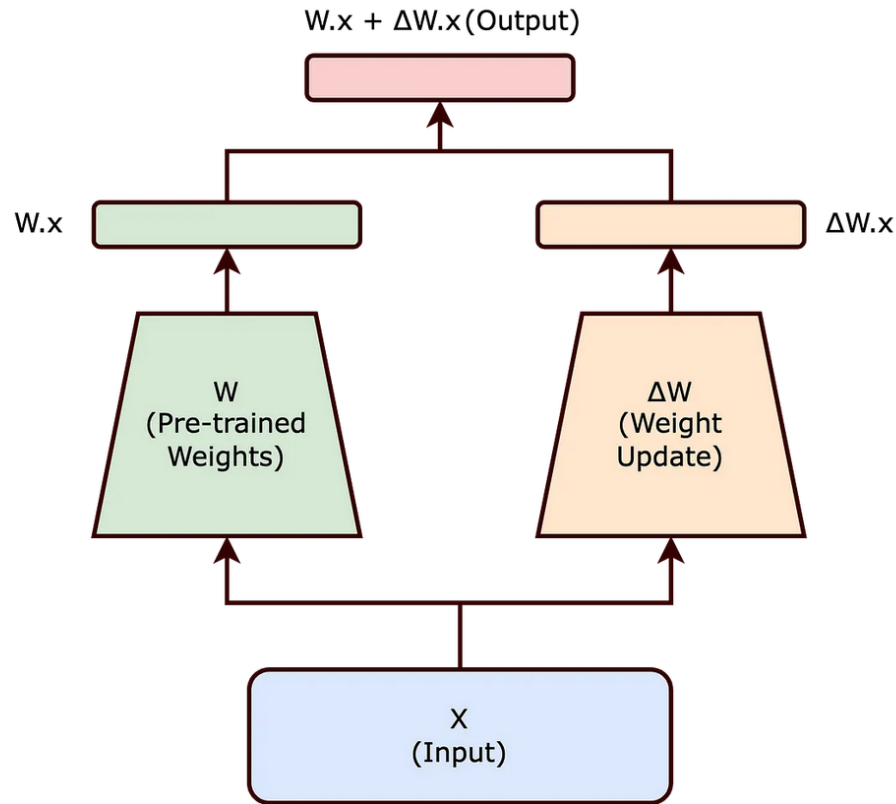
LoRa: Low-Rank Adaptation of Large Language Models

$$h = W_0 x + BA x$$

Implementation

- Model: wxjiao/alpaca-7b
- Github: https://github.com/tloen/alpaca-lora/

Hardware

- Nvidia A100 with 80GB
- 8-cores CPU with access to 128 GB RAM.

Details

- Float16 precision
- Early stopping on the training loss
- Linear Learning rate scheduler
- Adam Optimizer

Sentiment Analysis

Develop a classifier to categorize the sentiment of a review into positive or negative

Experimental Setup

Amazon Polarity Dataset

split	Reviews	Positive	Negative
Training	32,000	16,022	15,978
Validation	100	54	46
Test	4,000	2,007	1,993

Data Sample

Review 1: I have been programming in C++ for two years on the college level and found this book to be good at building a foundation knowledge of the language that can be expanded on with other book. I am on day 15 and alread have a good start on a bussiness program.

Sentiment: Positive

Prompting

Instruction

Classify the following review into two categories: 1) positive, and 2) negative based on its content, given the following examples.

Examples

```
Example 1
Example 2
```

Input
I have been programming in C++ for two years on the college level and found this book to be good at building a foundation knowledge of the language that can be expanded on with other book. I am on day 15 and alread have a good start on a bussiness program.

Response
Positive

Instruction fine-tuning

Instruction

Classify the following review into two categories: 1) positive, and 2) negative based on its content.

Input
I have been programming in C++ for two years on the college level and found this book to be good at building a foundation knowledge of the language that can be expanded on with other book. I am on day 15 and alread have a good start on a bussiness program.

Response
Positive

Sentiment Classification Results with Alpaca-7b

Approach	Shots	Accuracy
SVM		0.83
prompting	16	0.88
Instruction-fine-tuning	16	0.88

Lessons Learned

- ❏ Instruction fine tuning is faster than prompting
- ❏ Set multiple seeds and average over them
- ❏ Use the Alpaca template format for prompt engineering
- ❏ Balance the label distribution in the few shot as well
- ❏ Use sampling methods to choose your few shots
- ❏ Explicitly state the labels in the prompt and define them