Fine-tuning LLMs

Shaw Talebi

What is Fine-tuning?

Taking a pre-trained model and training at least one model parameter

GPT-3

ChatGPT
(i.e. GPT-3.5-turbo)

[1], [2], [3], [4]

What is Fine-tuning?

Taking a pre-trained model and training at least one model parameter

Base Model

(davinci (GPT-3))

Prompt: Tell me how to fine-tune a model

Completion:
How can I control the complexity of a model?
How do I know when my model is done?
How do I test a model?
How do I define a task for my model?
How do I scale a model?
How do I make a probabilistic model?
How do I start using a model once it's complete?
How do I cope with missing data?
How do I make my modeling efforts sustainable?

Fine-tuned Model

(text-davinci-003)

Prompt: Tell me how to fine-tune a model

Completion:
Fine-tuning a model involves adjusting the parameters of a pre-trained model in order to make it better suited for a given task. There are generally three steps to fine-tuning a model:

1. Select a base model - Select a pre-trained deep learning model that has been trained on a large dataset.
2. Adjust parameters - Adjust parameters of the pre-trained model to better suit the desired task. This may include changing the number of layers, adjusting learning rate, adding regularization, or tweaking the optimizer.
3. Train the model - Train the new model on the desired dataset. The amount of data and the amount of training required will depend on the task and the model.

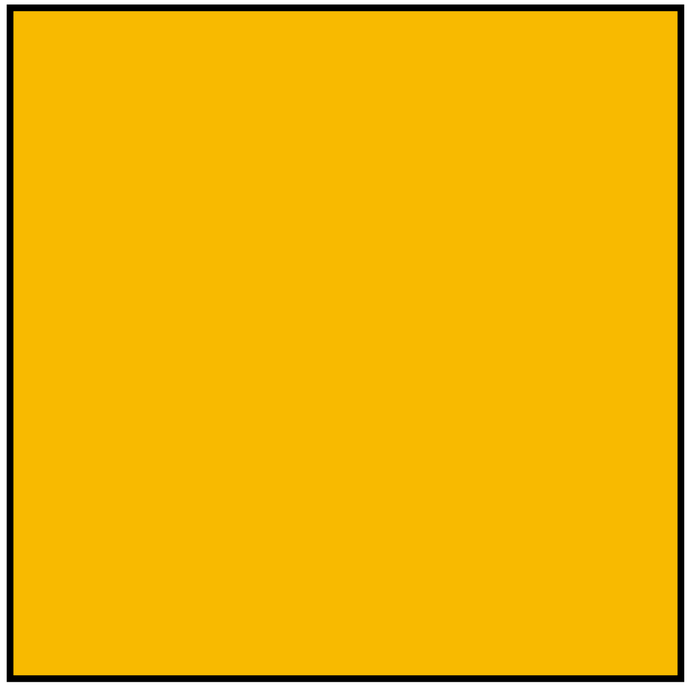
[1], [2], [3], [4]

Why Fine-tune.

A smaller (fine-tuned) model can outperform a larger base model

InstructGPT (1.3B)

>



GPT-3 (175B)

[4]

3 Ways to Fine-tune

Training Corpus

Listen to your

heart.

1) Self-supervised

Input	Output

Input: *Who was the 35th President of the United States?*

Output: *John F. Kennedy*

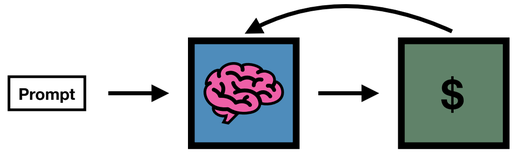
```
"""Please answer the following question.

Q: {Question}

A: {Answer}"""
```

2) Supervised

i. Supervised FT

ii. Train Reward Model

iii. RL with PPO



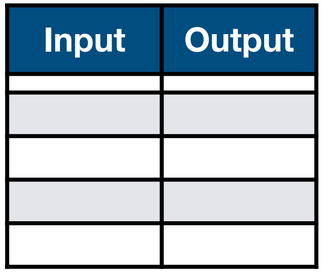
3) Reinforcement Learning

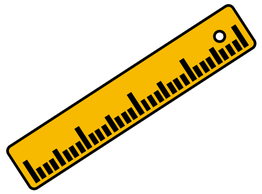
More details in the blog

[1] [3] [4]

Supervised Fine-tuning (in 5 Steps)

1. Choose fine-tuning task
2. Prepare training dataset
3. Choose a base model
4. Fine-tune model via supervised learning
5. Evaluate model performance

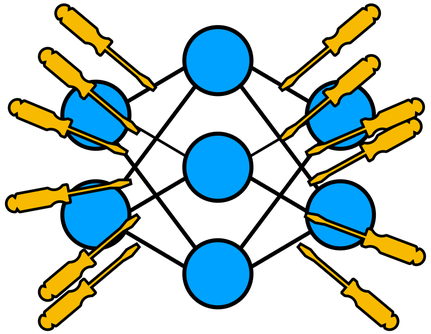




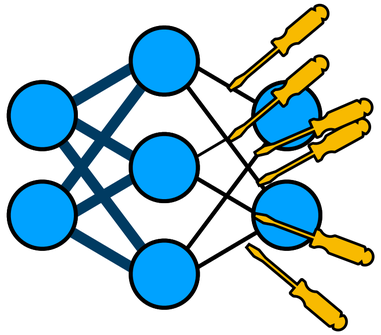
 [1]

3 Options for Parameter Training

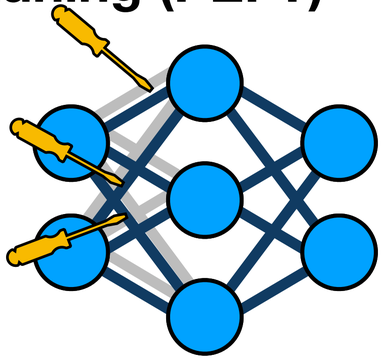
1) Retrain all parameters



2) Transfer Learning



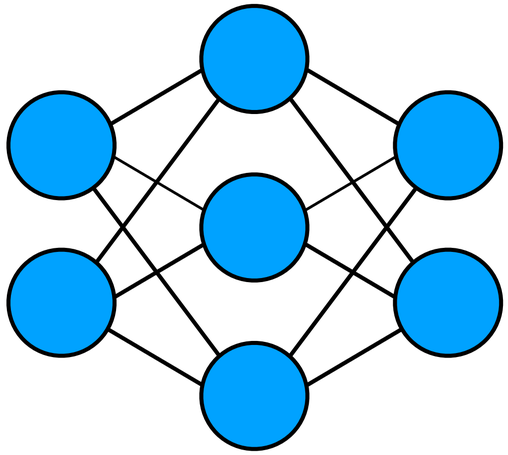
3) Parameter Efficient Fine-tuning (PEFT)



[5][6]

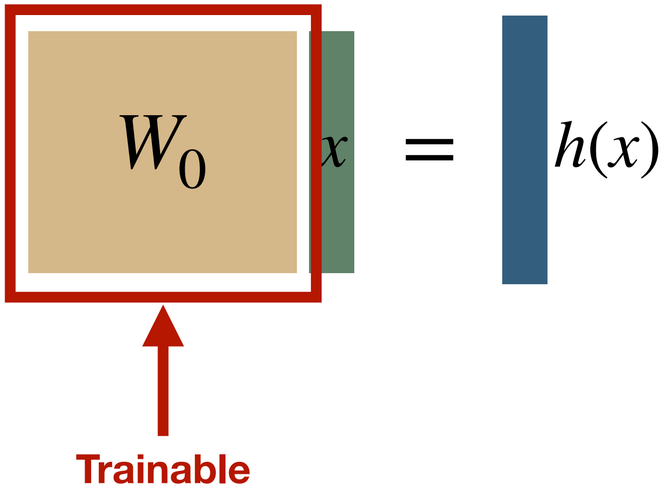
Low-Rank Adaptation (LoRA)

Fine-tunes model by adding new trainable parameters



$x \rightarrow h(x)$

$$h(x) = W_0 x$$



$$W_0 \in R^{d \times k}$$
$$x \in R^{k \times 1}$$
$$h(x) \in R^{d \times 1}$$

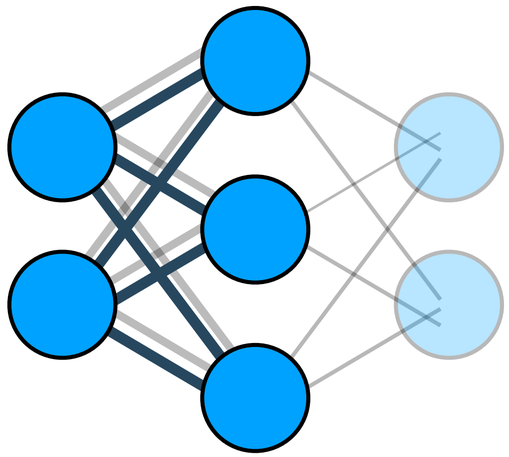
$$d = 1{,}000$$
$$k = 1{,}000$$
$$\Longrightarrow$$
$$d \times k = \textbf{1,000,000}$$
trainable parameters

 [6]

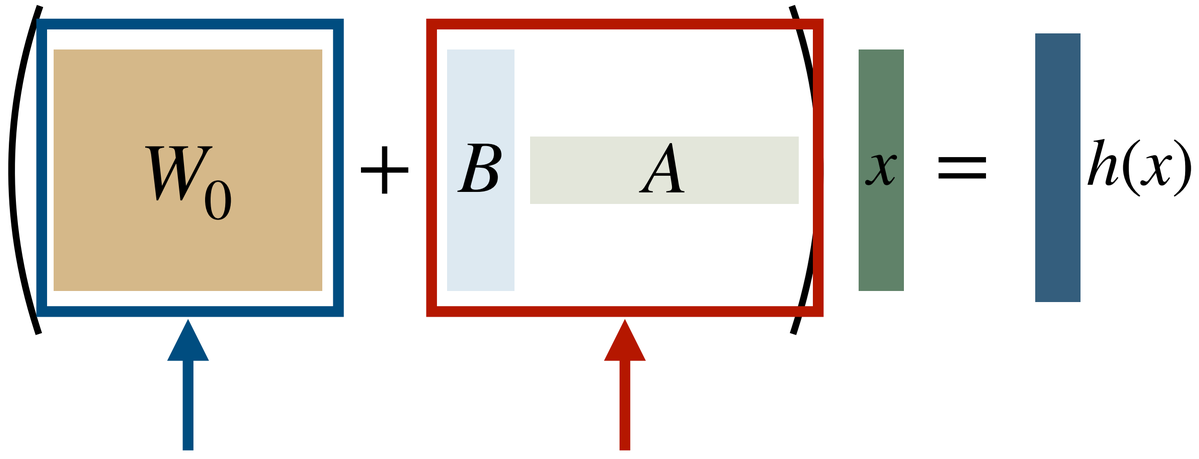
Low-Rank Adaptation (LoRA)

Fine-tunes model by adding new trainable parameters

$$h(x) = W_0 x + \Delta W x \qquad \Delta W = BA$$
$$= W_0 x + BAx$$



$x \rightarrow h(x)$



$$W_0, \Delta W \in R^{d \times k}$$
$$B \in R^{d \times r}$$
$$A \in R^{r \times k}$$
$$h(x) \in R^{d \times 1}$$

$d = 1{,}000$
$k = 1{,}000$ $\Longrightarrow$ $(d \times r) + (r \times k)$ = **4,000 trainable parameters**
$r = 2$

 [6]

Example Code: Fine-tuning an LLM w/ LoRA

Imports

```
from datasets import load_dataset, DatasetDict, Dataset

from transformers import (
    AutoTokenizer,
    AutoConfig,
    AutoModelForSequenceClassification,
    DataCollatorWithPadding,
    TrainingArguments,
    Trainer)

from peft import PeftModel, PeftConfig, get_peft_model, LoraConfig
import evaluate
import torch
import numpy as np
```


https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs

Example Code: Fine-tuning an LLM w/ LoRA

Base model

```
model_checkpoint = 'distilbert-base-uncased'

# define label maps
id2label = {0: "Negative", 1: "Positive"}
label2id = {"Negative":0, "Positive":1}

# generate classification model from model_checkpoint
model = AutoModelForSequenceClassification.from_pretrained(
    model_checkpoint, num_labels=2, id2label=id2label, label2id=label2id)
```


https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs

Example Code: Fine-tuning an LLM w/ LoRA

Load data

```
# load dataset
dataset = load_dataset("shawhin/imdb-truncated")
dataset

# dataset =
# DatasetDict({
#     train: Dataset({
#         features: ['label', 'text'],
#         num_rows: 1000
#     })
#     validation: Dataset({
#         features: ['label', 'text'],
#         num_rows: 1000
#     })
# })
```


https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs

Example Code: Fine-tuning an LLM w/ LoRA

https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs

Preprocess data

```
# create tokenizer
tokenizer = AutoTokenizer.from_pretrained(model_checkpoint, add_prefix_space=True)
```

```
# create tokenize function
def tokenize_function(examples):
    # extract text
    text = examples["text"]

    #tokenize and truncate text
    tokenizer.truncation_side = "left"
    tokenized_inputs = tokenizer(
        text,
        return_tensors="np",
        truncation=True,
        max_length=512
    )

    return tokenized_inputs

# add pad token if none exists
if tokenizer.pad_token is None:
    tokenizer.add_special_tokens({'pad_token': '[PAD]'})
    model.resize_token_embeddings(len(tokenizer))

# tokenize training and validation datasets
tokenized_dataset = dataset.map(tokenize_function, batched=True)
tokenized_dataset
```

```
# tokenized_dataset =
# DatasetDict({
#     train: Dataset({
#         features: ['label', 'text', 'input_ids', 'attention_mask'],
#         num_rows: 1000
#     })
#     validation: Dataset({
#         features: ['label', 'text', 'input_ids', 'attention_mask'],
#         num_rows: 1000
#     })
# })
```

Example Code: Fine-tuning an LLM w/ LoRA

```
# create data collator
data_collator = DataCollatorWithPadding(tokenizer=tokenizer)
```

Evaluation metrics

```
# import accuracy evaluation metric
accuracy = evaluate.load("accuracy")

# define an evaluation function to pass into trainer later
def compute_metrics(p):
    predictions, labels = p
    predictions = np.argmax(predictions, axis=1)

    return {"accuracy": accuracy.compute(predictions=predictions,
                                         references=labels)}
```



https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs

Example Code: Fine-tuning an LLM w/ LoRA

Untrained model performance

```
# define list of examples
text_list = ["It was good.", "Not a fan, don't recommed.",
"Better than the first one.", "This is not worth watching even once.",
"This one is a pass."]

print("Untrained model predictions:")
print("----------------------------")
for text in text_list:
    # tokenize text
    inputs = tokenizer.encode(text, return_tensors="pt")
    # compute logits
    logits = model(inputs).logits
    # convert logits to label
    predictions = torch.argmax(logits)

    print(text + " - " + id2label[predictions.tolist()])

# Output:
# Untrained model predictions:
# ----------------------------
# It was good. - Negative
# Not a fan, don't recommed. - Negative
# Better than the first one. - Negative
# This is not worth watching even once. - Negative
# This one is a pass. - Negative
```


https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs

Example Code: Fine-tuning an LLM w/ LoRA

Fine-tuning with LoRA

```
peft_config = LoraConfig(task_type="SEQ_CLS", # sequence classification
                         r=4, # intrinsic rank of trainable weight matrix
                         lora_alpha=32, # this is like a learning rate
                         lora_dropout=0.01, # probablity of dropout
                         target_modules = ['q_lin']) # we apply lora to query layer
```

```
model = get_peft_model(model, peft_config)
model.print_trainable_parameters()

# trainable params: 1,221,124 || all params: 67,584,004 || trainable%: 1.8068239934
```


https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs

Example Code: Fine-tuning an LLM w/ LoRA

Fine-tuning with LoRA

```
# hyperparameters
lr = 1e-3 # size of optimization step
batch_size = 4 # number of examples processed per optimziation step
num_epochs = 10 # number of times model runs through training data

# define training arguments
training_args = TrainingArguments(
    output_dir= model_checkpoint + "-lora-text-classification",
    learning_rate=lr,
    per_device_train_batch_size=batch_size,
    per_device_eval_batch_size=batch_size,
    num_train_epochs=num_epochs,
    weight_decay=0.01,
    evaluation_strategy="epoch",
    save_strategy="epoch",
    load_best_model_at_end=True,
)
```


https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs

Example Code: Fine-tuning an LLM w/ LoRA

Fine-tuning with LoRA

```
# creater trainer object
trainer = Trainer(
    model=model, # our peft model
    args=training_args, # hyperparameters
    train_dataset=tokenized_dataset["train"], # training data
    eval_dataset=tokenized_dataset["validation"], # validation data
    tokenizer=tokenizer, # define tokenizer
    data_collator=data_collator, # this will dynamically pad examples
    compute_metrics=compute_metrics, # evaluates model using compute_m
)

# train model
trainer.train()
```

Epoch	Training Loss	Validation Loss	Accuracy
1	No log	0.423429	{'accuracy': 0.876}
2	0.412400	0.551401	{'accuracy': 0.878}
3	0.412400	0.593060	{'accuracy': 0.899}
4	0.211600	0.640572	{'accuracy': 0.894}
5	0.211600	0.839775	{'accuracy': 0.891}
6	0.064300	0.930830	{'accuracy': 0.887}
7	0.064300	0.988515	{'accuracy': 0.889}
8	0.020000	1.007572	{'accuracy': 0.884}
9	0.020000	1.040836	{'accuracy': 0.888}
10	0.009500	1.034907	{'accuracy': 0.896}

https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs

Example Code: Fine-tuning an LLM w/ LoRA

Trained model performance

```
model.to('mps') # moving to mps for Mac (can alternatively do 'cpu')

print("Trained model predictions:")
print("--------------------------")
for text in text_list:
    inputs = tokenizer.encode(text, return_tensors="pt").to("mps") # moving to mps

    logits = model(inputs).logits
    predictions = torch.max(logits,1).indices

    print(text + " - " + id2label[predictions.tolist()[0]])

# Output:
# Trained model predictions:
# --------------------------
# It was good. - Positive
# Not a fan, don't recommed. - Negative
# Better than the first one. - Positive
# This is not worth watching even once. - Negative
# This one is a pass. - Positive # this one is tricky
```


https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs