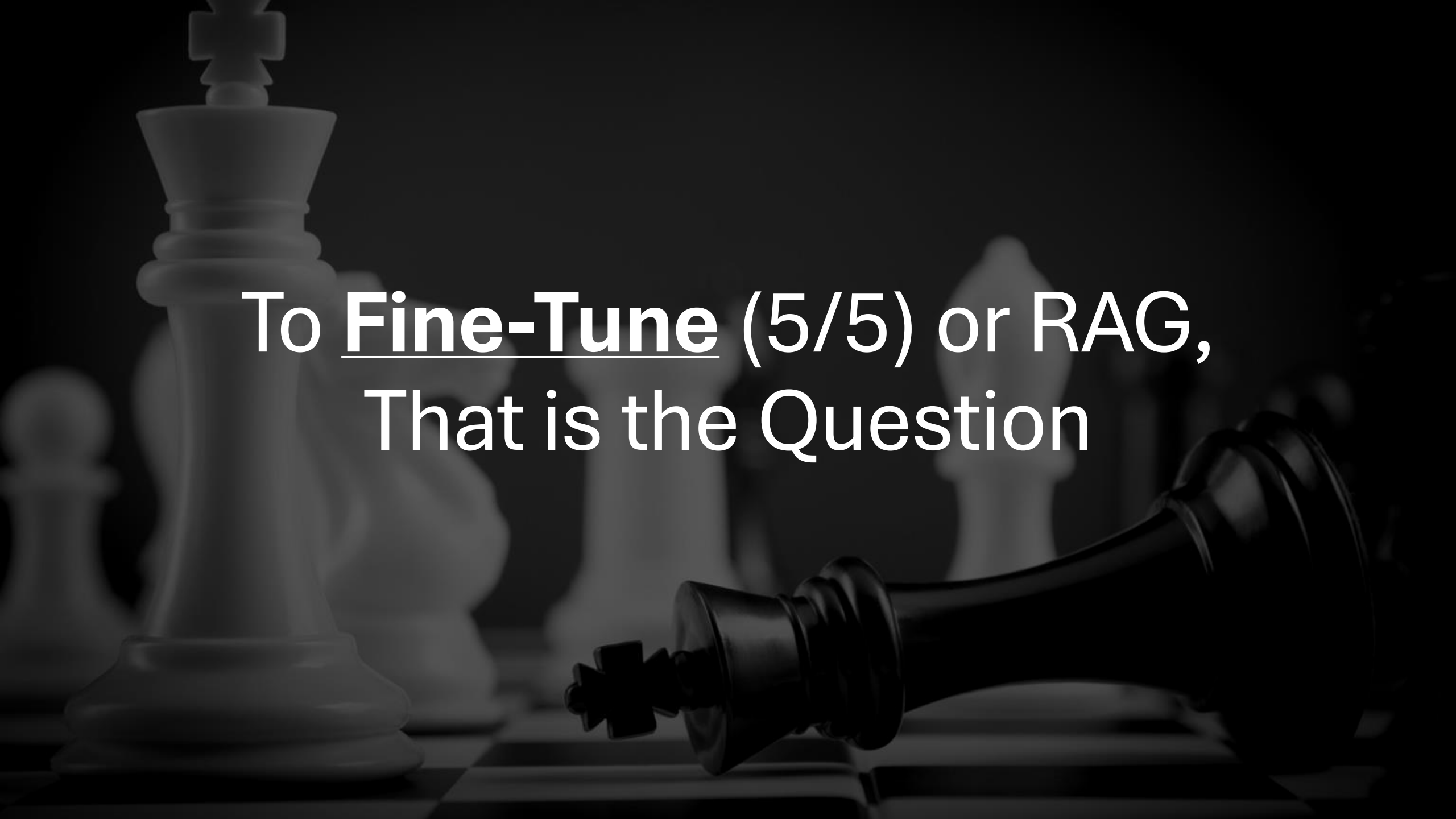




Topics in LLMs 2

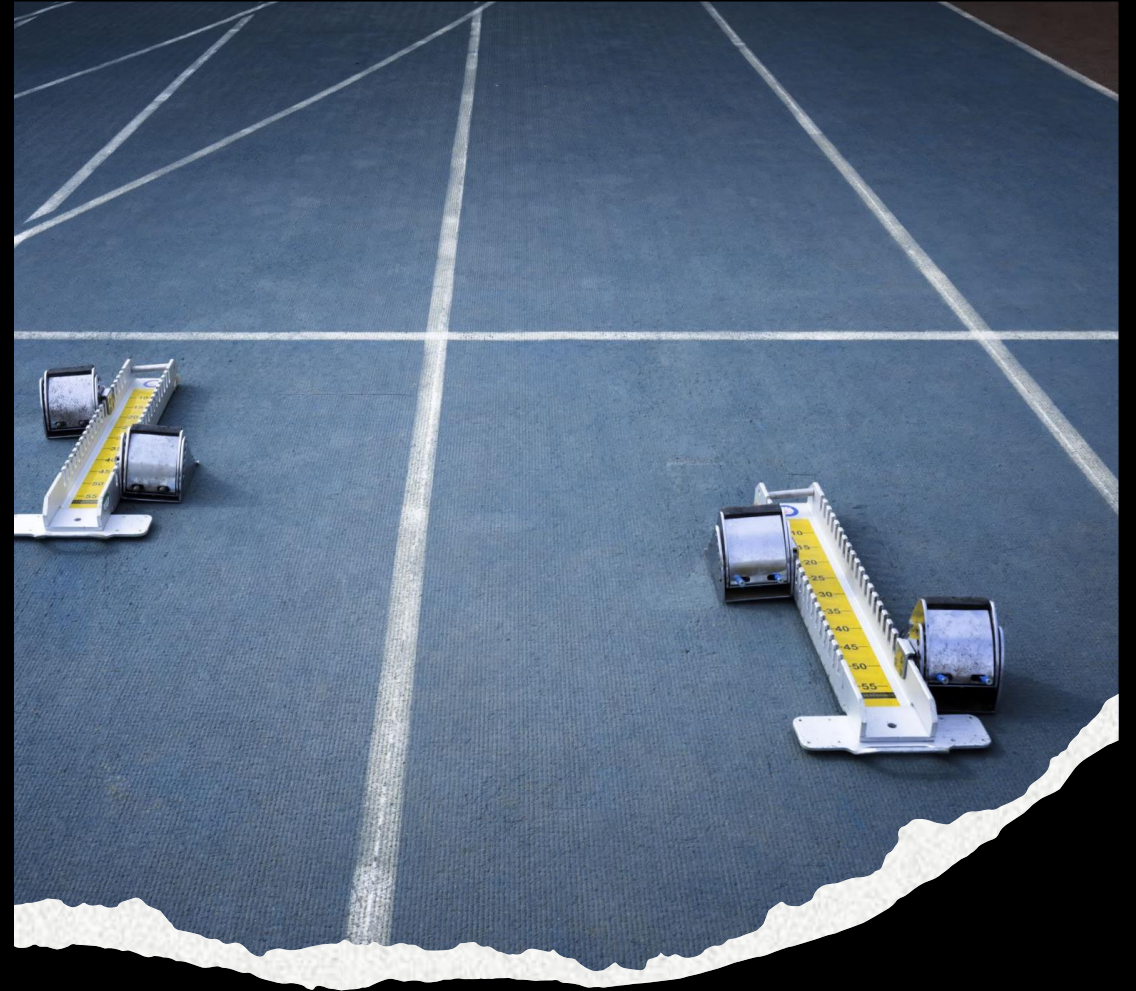
RAG and Fine-Tuning

Joel Kowalewski, PhD



To Fine-Tune (5/5) or RAG,
That is the Question

- **Fine-Tuning** an LLM or (generative model) refers to the use of a specialized training set to adjust the model's parameters (**weights and biases**)
- This involves dividing the training set into **batches** (small chunks) of data, then sending each batch into the model multiple times (**Iterations/Epochs**).
- After each batch, the model adjusts its parameters to improve its predictive success.
- In the case of LLMs, this success typically refers to performance on a MLM or (**Masked Language Modeling**) task. Here, the model “masks” or hides the next token, makes a prediction, and then compares its prediction to the actual token (this evaluation, the success, is quantified by a **Loss Function**)

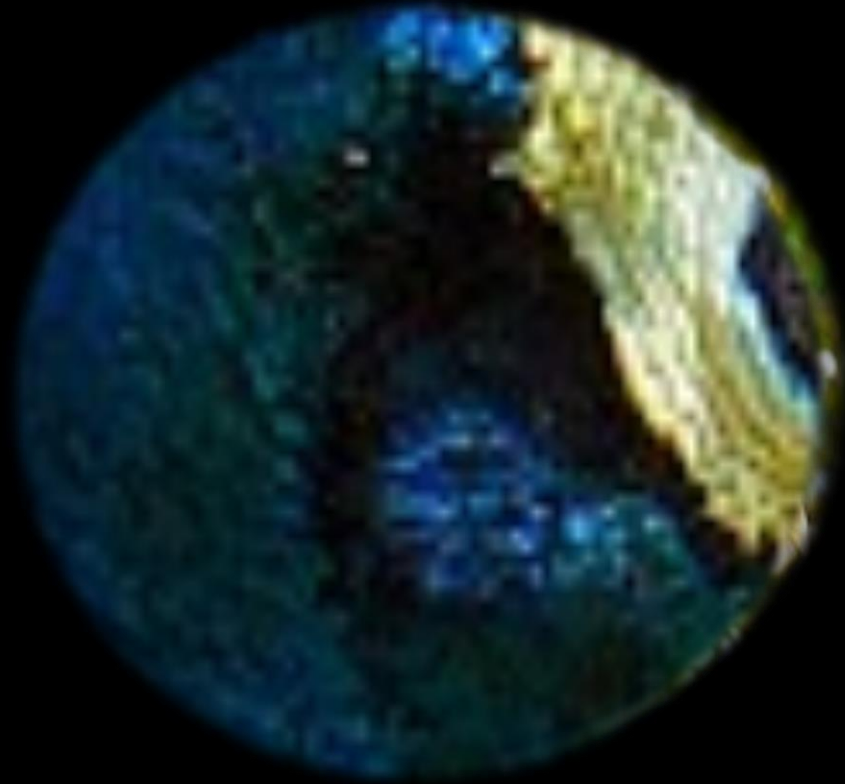


Understanding the Fine-Tuning Process

Epoch 1

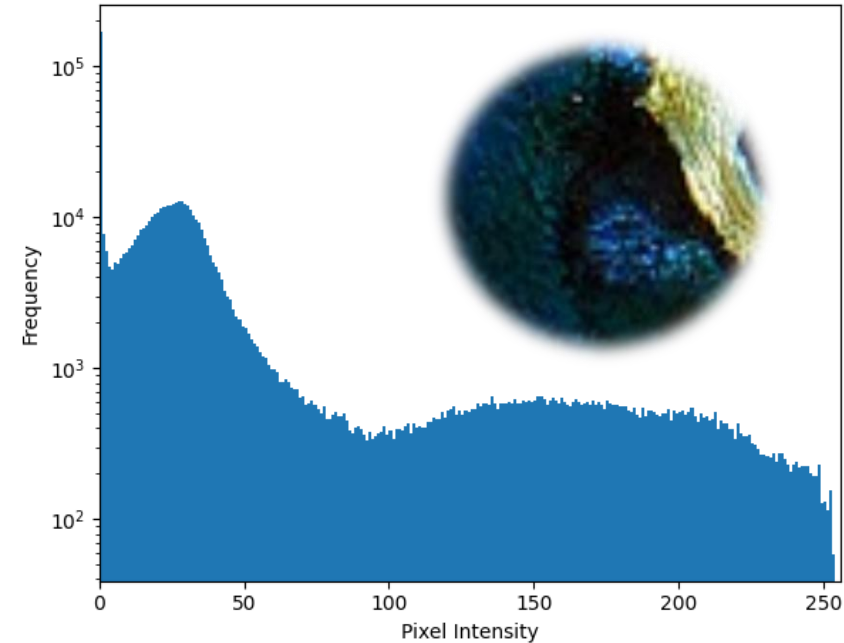
Epoch 2

Epoch 3

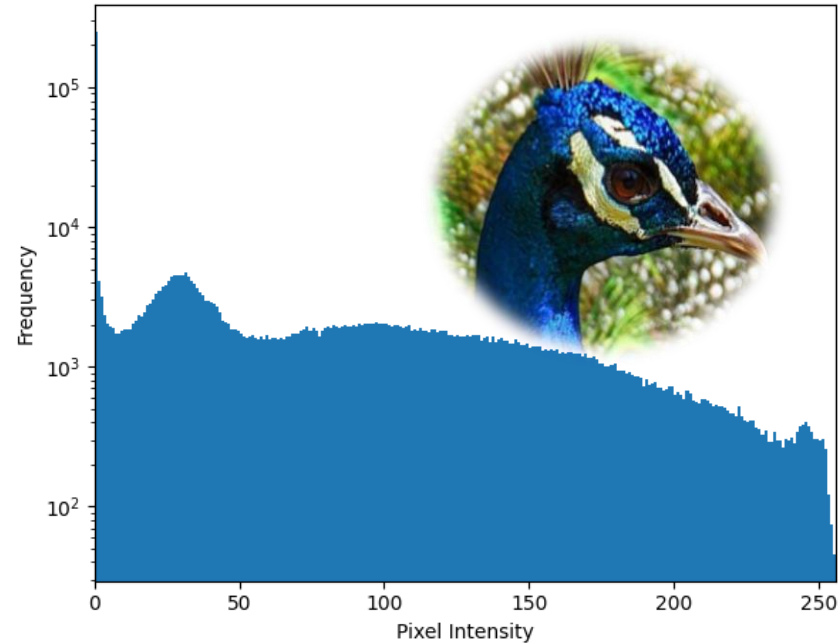


AI: Seeing the Mathematical Structure of Data

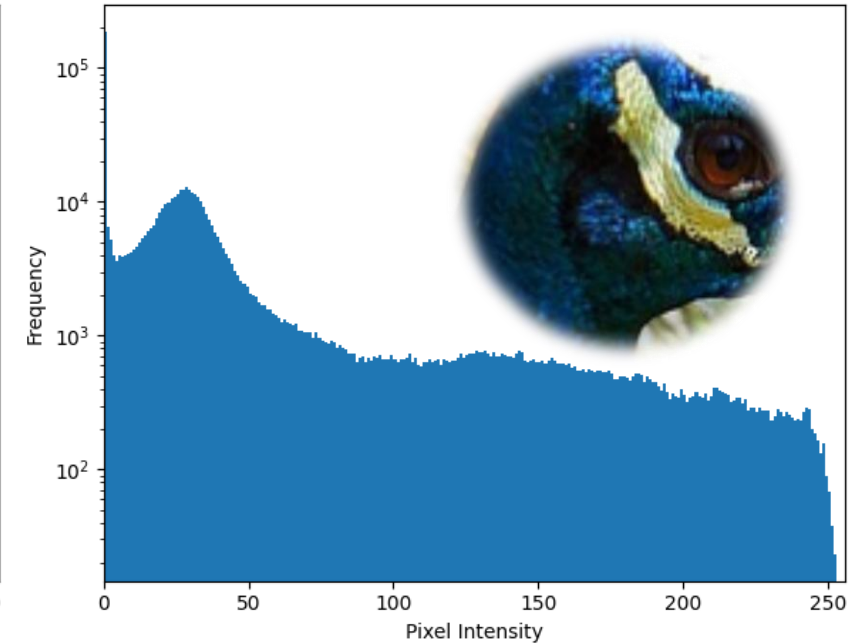
Histogram for Screenshot 2024-05-04 095727.png



Histogram for Screenshot 2024-05-04 095744.png

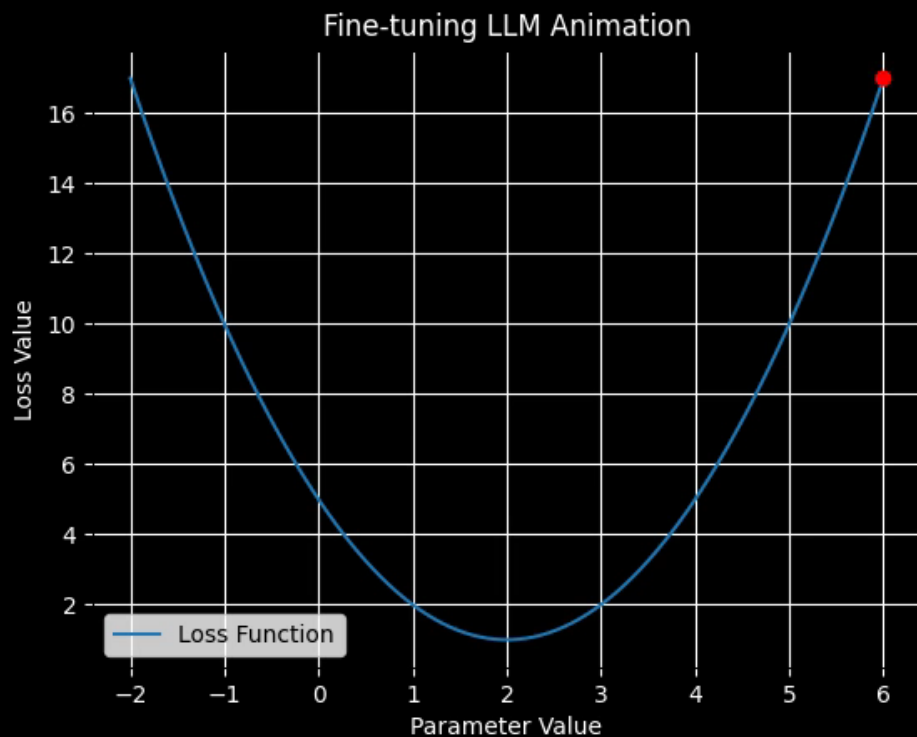


Histogram for Screenshot 2024-05-04 095755.png

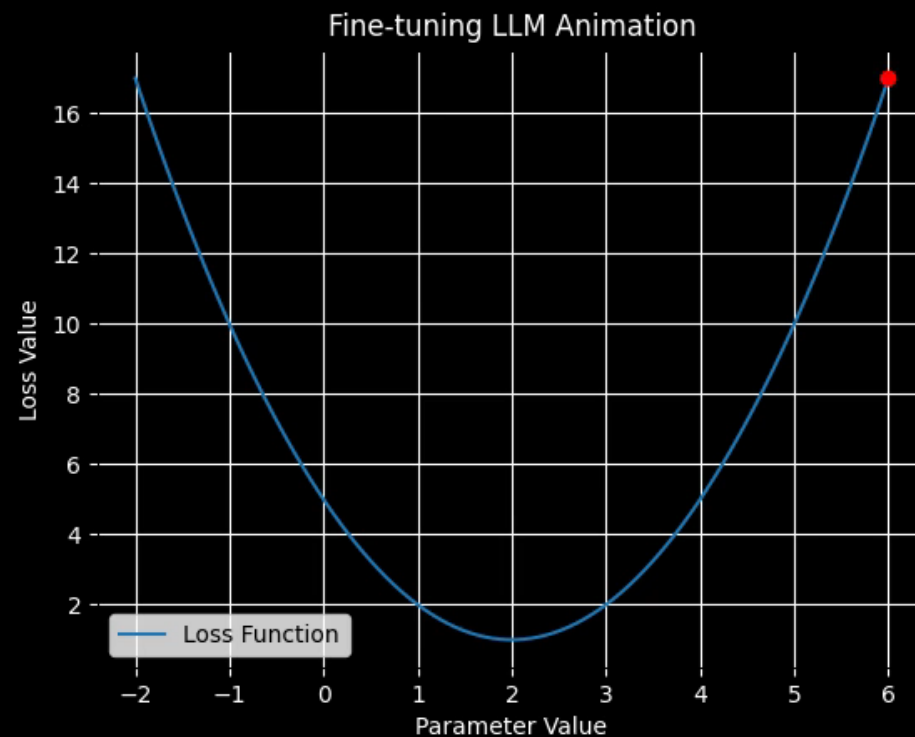


AI: Learning the Mathematical Structure of Data

Learning Rate = .1
50 Iterations/Epochs



Learning Rate = .01
50 Iterations/Epochs



Fine-Tuning LLMs Examples

The background of the slide is a close-up photograph of a wooden abacus. The wood is light-colored with a visible grain. There are several circular holes and a thick black line drawn on the surface. The numbers '17' and '18' are printed in black ink. A small, polished metal ball is resting on a wooden peg in the lower-left foreground.

```

from transformers import BertForSequenceClassification, BertTokenizer, Trainer, TrainingArguments
from datasets import load_dataset, load_metric

# Load dataset
dataset = load_dataset("glue", "mrpc")
metric = load_metric("glue", "mrpc", trust_remote_code=True)

# Load model and tokenizer
model = BertForSequenceClassification.from_pretrained("bert-base-uncased")
tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")

# Tokenize inputs
def tokenize_function(examples):
    return tokenizer(
        examples["sentence1"],
        examples["sentence2"],
        padding="max_length", # Pad to the maximum length
        truncation=True,
        max_length=128 # Adjust this based on data
    )

```

```

{'sentence1': 'Amrozi accused his brother , whom he
called " the witness " , of deliberately distorting his
evidence .', 'sentence2': 'Referring to him as only "
the witness " , Amrozi accused his brother of
deliberately distorting his evidence .', 'label': 1,
'idx': 0}

```

**DATA SAMPLE,
Post-Processing**


```

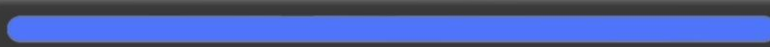
# Training arguments
training_args = TrainingArguments(
    output_dir="./results",
    evaluation_strategy="epoch",
    num_train_epochs=3,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    logging_dir="./logs",
)

# Trainer
def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = logits.argmax(axis=-1)
    return metric.compute(predictions=predictions, references=labels)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset["train"],
    eval_dataset=tokenized_dataset["validation"],
    compute_metrics=compute_metrics,
)

# Train
trainer.train()

```

 [1377/1377 05:14, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Accuracy	F1
1	No log	0.513653	0.784314	0.861635
2	0.574900	0.433456	0.818627	0.869718
3	0.429600	0.548696	0.835784	0.883478

**No Code
OpenAI API**

OpenAI API

```
{"messages": [{"role": "user", "content": "What is the focus of Dr. Joel Kowalewski's research?"}, {"role": "assistant", "content": "Dr. Joel Kowalewski focuses on developing computational discovery pipelines based on AI and machine learning algorithms, primarily for managing agricultural pests and arthropod vectors, as well as drug discovery."}]]}
{"messages": [{"role": "user", "content": "What fellowship did Dr. Joel Kowalewski receive?"}, {"role": "assistant", "content": "Dr. Joel Kowalewski was awarded the Dean's Distinguished Fellowship in August 2015."}]]}
{"messages": [{"role": "user", "content": "What is one of Dr. Joel Kowalewski's notable publications?"}, {"role": "assistant", "content": "One of Dr. Joel Kowalewski's notable publications is 'Prolonged activation of CO2-sensitive neurons in mosquitoes,' co-authored with S. T. Chen and A. Ray, and published in Interface Focus."}]]}
{"messages": [{"role": "user", "content": "What is Dr. Joel Kowalewski's contribution to AI and drug discovery?"}, {"role": "assistant", "content": "Dr. Joel Kowalewski developed a machine learning drug discovery pipeline that identified several drug candidates for COVID-19 treatment and successfully predicted vapor pressures of small molecules for potential inhaled therapeutics."}]]}
{"messages": [{"role": "user", "content": "What is Dr. Joel Kowalewski's patent in the field of pesticides?"}, {"role": "assistant", "content": "Dr. Joel Kowalewski holds a patent for 'Pesticides and Insect Repellents,' registered under U.S. Patent Application No. 62/968,817."}]]}
{"messages": [{"role": "user", "content": "What are Dr. Joel Kowalewski's contributions to sensory perception?"}, {"role": "assistant", "content": "Dr. Joel Kowalewski contributed to the understanding of the neurophysiological basis of human sensory perception, developing AI algorithms to predict human olfactory perception and screen chemicals for odorant receptors."}]]}
```

Log in

Sign up

Welcome to the OpenAI developer platform

Start with the basics

Quickstart tutorial

Make your first Chat Completions API request

Prompt examples

Explore what OpenAI models can do with prompts

Build an assistant



Introduction Beta

Learn the basics of building an assistant



Assistants deep dive Beta

Explore how assistants work and important concepts

Explore the API

EXAMPLE COSTS

10,000 tokens

Training for 3 epochs = $10,000 \times 3 = 30,000$

Input cost: $30,000 \text{ tokens} / 1,000 * \$0.012 = \$0.36$

Output cost: $30,000 \text{ tokens} / 1,000 * \$0.016 = \$0.48$

Total cost: $\$0.36 + \$0.48 = \$0.84$

Create a fine-tuned model

Base Model

gpt-3.5-turbo-1106

⌵

Training data

Add a jsonl file to use for training.

☒ Upload new ☐ Select existing

DATA (jsonl)

📎+

Upload a file or drag and drop here
(.jsonl)

Validation data

Add a jsonl file to use for validation metrics.

☐ Upload new ☐ Select existing ☒ None

Summary Table: RAG (5/1/24) versus Fine-Tuning (Next time)

Feature	Fine-tuning LLM	RAG
Customization	Pros: Highly customizable to specific tasks and datasets. Cons: Requires extensive computational resources for training.	Pros: Adapts dynamically by retrieving from a knowledge base. Cons: Retrieval errors can propagate to generation errors.
Data Dependency	Pros: Performs well with large, diverse datasets. Cons: Performance depends heavily on the quality of training data.	Pros: Less dependent on specific training datasets. Cons: Reliant on the quality and relevance of external data.
Computational Cost	Pros: High upfront computational cost for training, but lower ongoing costs. Cons: High initial investment in resources.	Pros: Lower training costs since it uses pre-trained models. Cons: Higher query-time costs due to retrieval process.
Adaptability	Pros: Highly tailored once trained. Cons: Not easy to adapt to new data quickly; retraining required.	Pros: Can adapt to new information if the knowledge source is updated. Cons: May struggle with very recent or niche topics not in the database.
Inference Time	Pros: Faster performance at inference time. Cons: Eventually, retraining is needed since new data/information is generated at a high rate. This issue leads back to the computational cost dilemma; fine-tuning may require repeated investments in hardware or cloud compute services.	Pros: Can incorporate real-time data if connected to a live database. Cons: Slower response times due to retrieval step.
Scalability	Pros: Scales well once trained, with consistent performance. Cons: Limited by the cost and feasibility of retraining.	Pros: Can scale by expanding the external data source. Cons: Affected by the efficiency of the retrieval system.