
Part 1: Theory of Fine-Tuning

Concept Check (Multiple Choice Questions):

1. Correct Answer: B) It customizes the model for specific tasks or domains.

2. Correct Answer: B) When the model loses its generalization ability after excessive fine-tuning on a specific task.

Application Task:

1. Explanation of Transfer Learning (150–200 words):

Transfer learning is like teaching a professional athlete to excel in a new sport. For example, a basketball player already has strong physical fitness, hand-eye coordination, and teamwork skills. When transitioning to volleyball, they don't start from scratch; instead, they build on their existing skills and focus on learning volleyball-specific techniques like serving and spiking.

Similarly, in machine learning, transfer learning involves taking a pre-trained model (e.g., BERT or GPT) that has already learned general language patterns from vast datasets and fine-tuning it for a specific task (e.g., sentiment analysis or legal document classification). The model retains its general knowledge (e.g., grammar, syntax) and adapts to the new task by learning task-specific patterns (e.g., legal jargon or emotional tone). This approach saves time, computational resources, and data compared to training a model from scratch.

For instance, in healthcare, a pre-trained LLM can be fine-tuned to classify medical records into diagnostic categories. The model leverages its general language understanding and learns to recognize medical terms and patterns, making it highly effective for the task.

---

2. Example Dataset Structure:

Task: Sentiment Analysis on Movie Reviews

Dataset Structure:

```
text,label
"I loved the movie! The acting was fantastic.",positive
"The film was boring and poorly made.",negative
"An average movie with some good scenes.",neutral
```

Cleaning Steps:

1. Remove special characters and HTML tags.
2. Convert text to lowercase.
3. Remove stopwords (optional, depending on the task).
4. Balance the dataset to ensure equal representation of labels.

Part 2: Practical Fine-Tuning Session

Hands-On Coding Task:

1. Environment Setup:

```python
# Install required libraries
!pip install transformers datasets accelerate
```

```
# Verify GPU availability
import torch
print("GPU Available:", torch.cuda.is_available())
```

2. Preprocessing Data:

```python
from datasets import load_dataset
from transformers import AutoTokenizer

# Load the IMDB dataset
dataset = load_dataset("imdb")

# Load the tokenizer for distilbert-base-uncased
tokenizer = AutoTokenizer.from_pretrained("distilbert-base-uncased")

# Tokenize the dataset
def preprocess_function(examples):
    return tokenizer(examples["text"], truncation=True, padding="max_length", max_length=512)

tokenized_dataset = dataset.map(preprocess_function, batched=True)
```

3. Model Training:

```python
from transformers import AutoModelForSequenceClassification, TrainingArguments, Trainer
```

```
# Load the pre-trained model
model = AutoModelForSequenceClassification.from_pretrained("distilbert-base-uncased", num_labels=2)

# Define training arguments
training_args = TrainingArguments(
    output_dir="./results",
    evaluation_strategy="epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=3,
    weight_decay=0.01,
    save_strategy="epoch",
    logging_dir="./logs",
)

# Initialize the Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_dataset["train"],
    eval_dataset=tokenized_dataset["test"],
)

 Fine-tune the model
trainer.train()
```

4. Save and Evaluate:

python

Save the fine-tuned model

trainer.save_model("./fine-tuned-distilbert-imdb")

Evaluate the model

eval_results = trainer.evaluate()

print("Evaluation Results:", eval_results)

Reflection:

Key Challenges and Solutions:

1. Challenge: Limited GPU memory caused out-of-memory errors during training.

Solution: Reduced batch size (`per_device_train_batch_size`) and used gradient accumulation.

2. Challenge: Overfitting on the training dataset.

Solution: Added regularization techniques like weight decay and early stopping.

3. Challenge: Low accuracy on the test set.

Solution: Increased the number of epochs and fine-tuned the learning rate.

Suggestions for Improving Performance:

Hyperparameter Tuning: Experiment with different learning rates, batch sizes, and epochs.

Data Augmentation: Add more diverse examples to the dataset to improve generalization.

Advanced Techniques: Use techniques like learning rate scheduling or mixed-precision training.

Larger Model: If resources allow, fine-tune a larger model like `bert-base-uncased` for better performance.

By completing this assignment, you'll gain hands-on experience in fine-tuning LLMs and a deeper understanding of the theory behind it. Let me know if you need further clarification or assistance!