Fine-Tuning Basics

Page 1: Introduction to Fine-Tuning

Fine-tuning is a crucial concept in modern Machine Learning and Natural Language Processing (NLP). It refers to the process of taking a pre-trained model and continuing its training on a smaller, task-specific dataset. Instead of training a model from scratch, fine-tuning allows us to reuse knowledge learned from large-scale datasets.

Pre-trained models have already learned general patterns such as grammar, syntax, and semantic relationships. Fine-tuning adapts this general knowledge to solve specific tasks like sentiment analysis, text classification, or question answering.

Page 2: Why Fine-Tuning is Important

Training deep learning models from scratch requires massive datasets, high computational power, and time. Fine-tuning addresses these challenges by:

- Reducing training time
- Lowering computational cost
- Improving performance on domain-specific tasks
- Requiring less labeled data

Fine-tuning is widely used in real-world applications such as chatbots, recommendation systems, and enterprise NLP solutions.

Page 3: Pre-Training vs Fine-Tuning

Pre-Training

- Model is trained on very large datasets (e.g., Wikipedia, Common Crawl)
- Learns general language representations
- Computationally expensive

Fine-Tuning

- Model is trained on task-specific data
- Learns task-oriented patterns
- Computationally efficient

Aspect	Pre-Training	Fine-Tuning
Data Size	Very Large	Small/Medium
Cost	High	Low
Purpose	General Learning	Task Adaptation

Page 4: Fine-Tuning Workflow

The general fine-tuning workflow includes:

1. Selecting a pre-trained model
2. Preparing labeled dataset
3. Tokenizing input data
4. Adding task-specific layers
5. Training with low learning rate
6. Evaluating model performance

This workflow ensures minimal disruption to learned representations while adapting the model effectively.

Page 5: Types of Fine-Tuning

1. Full Fine-Tuning

- Updates all model parameters
- High accuracy
- Requires more computation

2. Partial Fine-Tuning

- Freezes lower layers
- Trains only top layers
- Suitable for small datasets

3. Parameter-Efficient Fine-Tuning (PEFT)

- Uses adapters or LoRA
- Minimal parameter updates
- Ideal for large language models

Page 6: Hyperparameters and Optimization

Important hyperparameters in fine-tuning:

- Learning Rate (usually very small: 2e-5)
- Batch Size (8, 16, 32)

- Epochs (2–5)
- Optimizer (AdamW)
- Loss Function (CrossEntropy, MSE)

Careful tuning avoids overfitting and catastrophic forgetting.

Page 9: Common Use Cases of Fine-Tuning

Fine-tuning is widely applied in:

- Sentiment Analysis
- Text Classification
- Named Entity Recognition
- Question Answering
- Chatbots
- Domain-specific NLP (medical, legal, finance)

Each use case benefits from domain adaptation and improved accuracy.

Page 10: Challenges and Best Practices

Challenges

- Overfitting on small datasets
- Catastrophic forgetting
- High GPU memory usage

Best Practices

- Use small learning rates
- Freeze layers initially
- Apply early stopping
- Validate with unseen data

Conclusion

Fine-tuning is a powerful technique that bridges the gap between generic pre-trained models and task-specific requirements. By carefully applying fine-tuning strategies, developers can build highly accurate and efficient machine learning systems with minimal resources.