Faysal Aziz

Neural Networks with Python

From Fundamentals to Modern Deep Learning

FA-0338 · three-day

Agentic & Generative AI · Software Development · DevOps, Cloud & Infrastructure · Data & Analytics

---

Introduction

Build the neural-network foundations behind vision, sequence modelling, transformers, and modern AI.

Neural networks have become a core engineering technology behind image recognition, forecasting, language processing, recommendation systems, speech applications, and generative AI. Although modern frameworks make it possible to assemble sophisticated models quickly, effective practitioners still need to understand what happens beneath the framework: how information moves through a network, how errors are measured, how gradients update parameters, and why one architecture is better suited to a particular type of data than another.

This three-day course develops that understanding using Python and PyTorch. It begins with the mechanics of feed-forward neural networks and progresses into convolutional neural networks for image data, recurrent neural networks for sequential information, and transformer-based architectures used in modern large language models. The emphasis remains practical throughout: building models, preparing data, training networks, evaluating results, and understanding the architectural decisions that affect performance.

The course also introduces the modern pretrained-model workflow. Current PyTorch 2.x development continues to improve neural-network execution, attention mechanisms, hardware acceleration, and large-model training capabilities, while Hugging Face Transformers provides a widely used ecosystem for pretrained text, vision, audio, and multimodal models. Parameter-efficient techniques such as LoRA are also increasingly important because they allow large pretrained models to be adapted without retraining every model parameter.

The instructor has **over 30 years of industry experience** and will deliver the programme using real industry-demanded concepts, workflows, terminology, and development practices rather than presenting neural networks as a purely academic subject.

Learning outcomes

By the end of this course, participants should be able to:

- Explain the structure and behaviour of feed-forward neural networks.
- Understand forward propagation, loss functions, backpropagation, and gradient descent.
- Build and train neural networks using Python and PyTorch.
- Select suitable activation functions, loss functions, and optimizers.
- Prepare datasets for neural-network training.
- Identify overfitting, underfitting, and common training problems.
- Build convolutional neural networks for image-related problems.
- Understand convolution, pooling, filters, feature maps, and transfer learning.
- Explain recurrent neural networks and their use with sequential data.
- Understand LSTM and GRU architectures.
- Explain attention and transformer architecture.
- Understand the fundamental architecture and operation of large language models.
- Work with pretrained transformer models.
- Understand embeddings, tokenization, fine-tuning, LoRA, and basic RAG concepts.
- Select appropriate neural-network architectures for common problem types.

Prerequisites

Participants must already be comfortable with:

- Intermediate Python programming
- Functions, classes, modules, packages, and debugging
- NumPy arrays and matrix operations
- Basic algebra, vectors, matrices, derivatives, and chain rule
- Basic probability and statistics
- Classification, regression, training/testing, and evaluation metrics
- Jupyter Notebook or an equivalent Python environment
- Installing packages and basic command-line operations

This course is not suitable for participants who are new to Python, mathematics, or machine learning.

Training outline

1. Neural Networks and Deep Learning Foundations

1. Neural networks within machine learning
2. Neurons, weights, biases, and layers
3. Inputs and outputs
4. Parameters and hyperparameters
5. Feed-forward neural networks
6. Classification and regression networks
7. Training versus inference

2. Mathematical Foundations for Neural Networks

1. Vectors, matrices, and tensors
2. Matrix multiplication
3. Weighted sums
4. Linear transformations
5. Derivatives and partial derivatives
6. Chain rule
7. Gradients

3. Forward Propagation and Activation Functions

1. Forward propagation
2. Hidden layers
3. Output layers
4. Sigmoid
5. Tanh
6. ReLU
7. Leaky ReLU
8. Softmax

4. Loss Functions and Backpropagation

1. Mean squared error
2. Binary cross-entropy

3. Multiclass cross-entropy
4. Computational graphs
5. Gradient calculation
6. Backpropagation
7. Parameter updates

5. Gradient-Based Optimization

1. Gradient descent
2. Stochastic gradient descent
3. Mini-batch training
4. Momentum
5. Adam
6. Learning rates
7. Learning-rate scheduling

6. PyTorch Fundamentals

1. Tensor creation and manipulation
2. Tensor shapes and data types
3. Tensor operations
4. CPU and GPU devices
5. Automatic differentiation
6. Gradient tracking
7. Neural-network modules

7. Building Neural Networks with PyTorch

1. nn.Module
2. Linear layers
3. Activation layers
4. Model composition
5. Forward methods
6. Loss functions
7. Optimizers
8. Model parameters

8. Data Preparation and Model Training

1. Feature and target preparation
2. Feature scaling
3. Training, validation, and test datasets
4. PyTorch Dataset
5. DataLoader
6. Batch processing
7. Training loops
8. Validation loops

9. Improving Neural-Network Performance

1. Underfitting
2. Overfitting
3. Training and validation curves
4. Weight initialization
5. Dropout
6. Weight decay
7. Early stopping
8. Model capacity

10. Convolutional Neural Networks

1. CNN architecture
2. Image tensors
3. Convolution operations
4. Kernels and filters
5. Feature maps
6. Stride and padding
7. Pooling
8. Receptive fields

11. Building CNNs with PyTorch

1. Convolutional layers
2. Pooling layers

3. Activation layers
4. Flattening
5. Fully connected layers
6. CNN model construction
7. Image classification
8. CNN training workflow

12. CNN Training and Transfer Learning

1. Batch normalization
2. Data augmentation concepts
3. Pretrained CNN models
4. Feature extraction
5. Frozen layers
6. Fine-tuning
7. Transfer learning

13. Recurrent Neural Networks

1. Sequential data
2. Temporal dependencies
3. RNN architecture
4. Hidden states
5. Sequence inputs and outputs
6. Backpropagation through time
7. Vanishing gradients
8. Exploding gradients

14. LSTM and GRU Networks

1. LSTM architecture
2. Cell states
3. Input, forget, and output gates
4. GRU architecture
5. Reset and update gates
6. LSTM versus GRU
7. Bidirectional networks

15. Sequence Modelling with PyTorch

1. RNN layers
2. LSTM layers
3. GRU layers
4. Sequence dimensions
5. Hidden-state handling
6. Sequence classification
7. Time-series considerations

16. Attention Mechanisms

1. Limitations of recurrent processing
2. Attention concepts
3. Queries, keys, and values
4. Attention scores
5. Self-attention
6. Multi-head attention

17. Transformer Architecture

1. Token embeddings
2. Positional information
3. Self-attention layers
4. Feed-forward layers
5. Residual connections
6. Layer normalization
7. Encoder architecture
8. Decoder architecture

18. Large Language Model Fundamentals

1. Language modelling
2. Tokens and tokenization
3. Vocabulary
4. Embeddings
5. Context windows

6. Next-token prediction
7. Pretraining
8. Inference

19. Working with Pretrained Transformers

1. Hugging Face Transformers
2. Model repositories
3. Tokenizers
4. Model loading
5. Pipelines
6. Text generation
7. Generation parameters

20. Fine-Tuning and LLM Adaptation

1. Transfer learning
2. Fine-tuning
3. Instruction tuning concepts
4. Parameter-efficient fine-tuning
5. LoRA
6. Quantization concepts
7. Compute and memory considerations

21. Embeddings and Retrieval-Augmented Generation

1. Text embeddings
2. Vector representations
3. Semantic similarity
4. Vector retrieval
5. Retrieval-augmented generation
6. Context augmentation

22. Model Evaluation and Diagnostics

1. Classification metrics
2. Regression metrics

3. Confusion matrices
4. Training instability
5. Data leakage
6. Class imbalance
7. Generalization

23. Practical Neural-Network Operations

1. Model saving and loading
2. Training and evaluation modes
3. Reproducibility
4. GPU considerations
5. Batch-size considerations
6. Memory management
7. Inference workflows

24. Neural-Network Architecture Selection

1. Feed-forward networks
2. CNNs
3. RNNs
4. LSTMs and GRUs
5. Transformers
6. Large language models
7. Transfer learning
8. Problem-to-architecture mapping

Disclaimer

This course outline is provided as a structured guideline for the intended training programme. The trainer reserves the professional discretion to amend, reorder, expand, reduce, substitute, or omit individual topics where reasonably necessary to accommodate participant competency, available instructional time, technical limitations, changes in relevant technologies, or other instructional considerations. Such modifications may be made without prior notice where deemed appropriate to preserve the relevance and effectiveness of the training.

© 2026 Faysal Aziz · Kuala Lumpur