Module 7 Summary: Deep Learning

Data Science with Python – BSc Course

What We Covered

Module 7: Deep Learning – Four Lessons

Lesson	Topic	Key Insight
L33	Perceptron	Single neuron, linear boundary
L34	MLPs & Activations	Hidden layers enable non-linearity
L35	Backpropagation	How networks learn via gradient descent
L36	Overfitting Prevention	Dropout, early stopping, regularization

You now understand how neural networks are built, trained, and regularized.

From single perceptrons to robust multi-layer networks

L33: Perceptron – The Building Block

The Foundation of Neural Networks

- Perceptron: weighted sum + activation = binary output
- Only learns linearly separable patterns (AND, OR but not XOR)
- Step function: original activation, not differentiable
- Sigmoid enables gradient-based learning (smooth, differentiable)

Key Limitation: Single perceptron = linear classifier. Cannot solve XOR problem (Minsky & Papert 1969).
Solution: Add hidden layers (multi-layer perceptrons).

Memory: Perceptron = linear boundary. XOR needs hidden layers.

L34: MLPs & Activations – Non-Linear Power

Overcoming Linear Limitations

- Hidden layers enable non-linear decision boundaries
- ReLU $(x) = \max(0, x)$ – modern default for hidden layers (no vanishing gradient)
- Output activations: sigmoid (binary), softmax (multiclass), linear (regression)
- Universal approximation: MLP with 1 hidden layer can approximate any continuous function

Keras Pattern: Sequential API – define layers $\rightarrow$ compile $\rightarrow$ fit $\rightarrow$ evaluate
Architecture: Start simple (1-2 hidden layers, 64-128 neurons), add depth for complex patterns

Memory: ReLU = max(0,x). Hidden layers = non-linear power. Keras = easy MLPs.

L35: Backpropagation – How Networks Learn

Training Neural Networks

- Forward pass: input flows through layers to produce prediction
- Backward pass: error flows backward, computes gradients via chain rule
- Gradient descent: $w := w - \eta \frac{\partial L}{\partial w}$ (move opposite to gradient)
- Learning rate η – most important hyperparameter (typical: 0.001)

Diagnosis: Monitor train/val loss curves. Train $\downarrow$, val $\downarrow$ = good. Train $\downarrow$, val $\uparrow$ = overfitting.
Optimizers: Adam is modern default (handles learning rate automatically)

Memory: Backprop = chain rule. Learning rate = step size. Watch val loss.

L36: Overfitting Prevention – Robust Models

Ensuring Generalization

- Dropout: randomly zero neurons during training (typical: 0.2-0.5)
- Early stopping: halt when validation loss stops improving (patience=10-50)
- L2 regularization: penalize large weights (lambda=0.001 typical)
- Always monitor train vs validation loss gap

Financial Data: Overfitting is common – use dropout + early stopping + small networks
Callbacks: EarlyStopping, ModelCheckpoint, ReduceLROnPlateau

Memory: Dropout = random zeros. Early stopping = stop at best val. Watch the gap.

Key Concepts Mastered

What You Can Now Do:

- Build neural networks with appropriate activation functions
- Train networks using backpropagation and gradient descent
- Diagnose overfitting from learning curves
- Apply regularization techniques (dropout, early stopping, L2)
- Implement complete neural network pipelines in Keras

Critical Skills:

- Choosing activations: ReLU (hidden), sigmoid (binary), softmax (multiclass)
- Reading loss curves: identify underfitting, overfitting, convergence
- Regularization: combine dropout + early stopping for robust models

You now understand how modern AI systems learn from data

Finance Applications

Where Deep Learning Shines in Finance

- **Market direction prediction:** Non-linear relationships between momentum, volatility, volume
- **Regime detection:** Classify bull/bear/sideways markets from multiple signals
- **Volatility forecasting:** Capture complex dependencies in market data
- **Credit scoring:** Non-linear interactions that logistic regression misses

Real-World Examples:

- JPMorgan's LOXM: optimal trade execution via deep learning
- Citadel & Two Sigma: neural networks for alpha generation
- Zest AI: fair lending with deep learning

Neural networks capture complexity that traditional models miss

Tools & Code Patterns

Essential Keras Workflow:

1. **Build:** `Sequential([Dense(64, 'relu'), Dropout(0.3), Dense(32, 'relu'), Dense(1, 'sigmoid')])`
2. **Compile:** `model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])`
3. **Train:** `model.fit(X_train, y_train, epochs=100, validation_split=0.2, callbacks=[EarlyStopping(patience=20)])`
4. **Evaluate:** `model.evaluate(X_test, y_test)`

Key Functions:

- sklearn: `Perceptron`, `MLPClassifier`, `StandardScaler`
- Keras: `Sequential`, `Dense`, `Dropout`, `EarlyStopping`, `ModelCheckpoint`
- Activations: `relu`, `sigmoid`, `softmax`, `linear`

Master this pattern and you can build any neural network

What's Next: Module 8 – NLP & Text

From Numbers to Language

Neural networks process numbers — stock prices, balance sheets. But finance is full of text: 300,000+ financial news articles daily, earnings calls, analyst reports, SEC filings.

Module 8: Natural Language Processing

- L37: Text Preprocessing – tokenization, stemming, cleaning
- L38: BOW & TF-IDF – converting text to features
- L39: Word Embeddings – capturing semantic meaning
- L40: Sentiment Analysis – extracting sentiment from financial text

The deep learning foundations you mastered here are the building blocks for language understanding.

Numbers tell part of the story — text tells the rest