Deep learning is a subset of machine learning (itself a subset of AI) that automatically learns hierarchical features, unlike classical ML which requires hand-engineering.

Motivation: deep learning scales better, handles complexity, and recent advances in data, compute, and algorithms make it practical now.

Core unit: the **perceptron** → inputs × weights + bias → passed through nonlinear activation.

Common activations: sigmoid (0–1 output), tanh (-1–1), ReLU (fast and effective).

Nonlinearity is crucial; without it, even deep networks collapse to a single linear transformation.

Networks are built by stacking perceptrons:

- Multi-output perceptron for multiple predictions.
- Sequentially stacked layers form hidden layers and deep neural networks.
- PyTorch examples show manual and built-in implementations.

To train networks, define a **loss function** to quantify error:

- Cross-entropy for classification, mean squared error for regression.
- Minimize empirical loss $J(W)$ across the dataset.

Optimization uses gradient descent:

- Randomly initialize weights.
- Compute gradients (via backpropagation using chain rule).
- Update weights in opposite direction of gradient with step size (learning rate).
- Too small learning rate → slow/stuck; too large → unstable/divergence. Adaptive methods can adjust automatically.

Mini-batch gradient descent balances efficiency and stability (better than full dataset or single sample updates).

Overfitting vs. generalization:

- Goal is to learn patterns that generalize to new data.
- Regularization helps prevent overfitting.
- Techniques: **dropout** (randomly deactivate neurons during training) and **early stopping** (halt training when validation error stops improving).

Overall: lecture introduced motivation, perceptrons, activation functions, network construction, loss functions, optimization, and practical considerations for training deep neural networks.