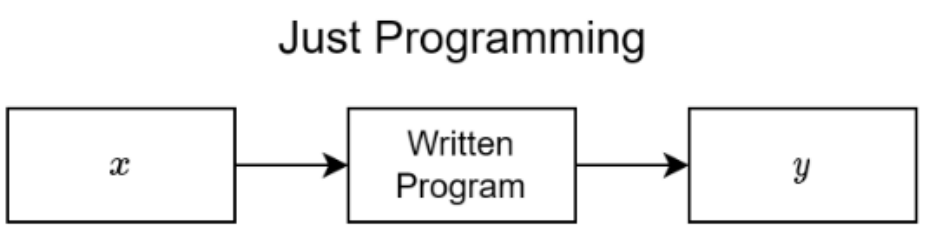
Module 1 - Session 1: Introduction to PyTorch and Neural Networks

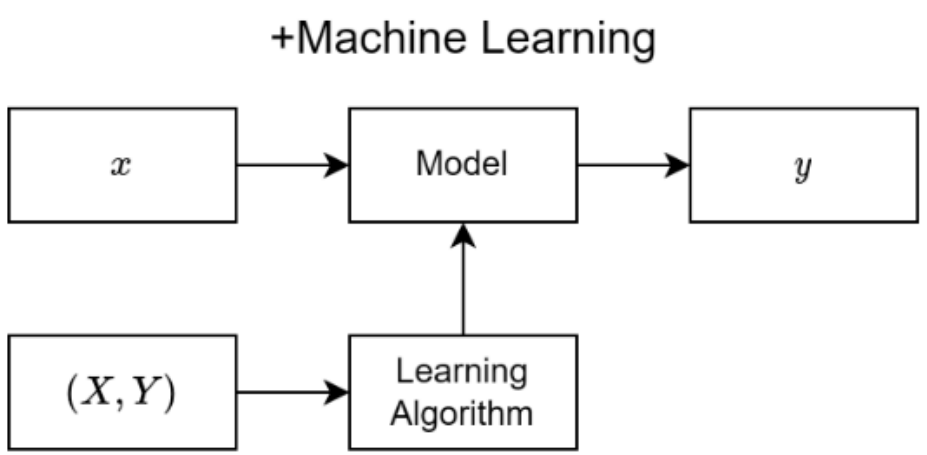
Module 1 Overview

What will we learn?

- Distinguish Deep Learning from Machine Learning
- What are neurons and how do they learn?
- Why learn the PyTorch framework?
- The ML Pipeline
- Activation Functions
- Tensors (PyTorch’s data structures)

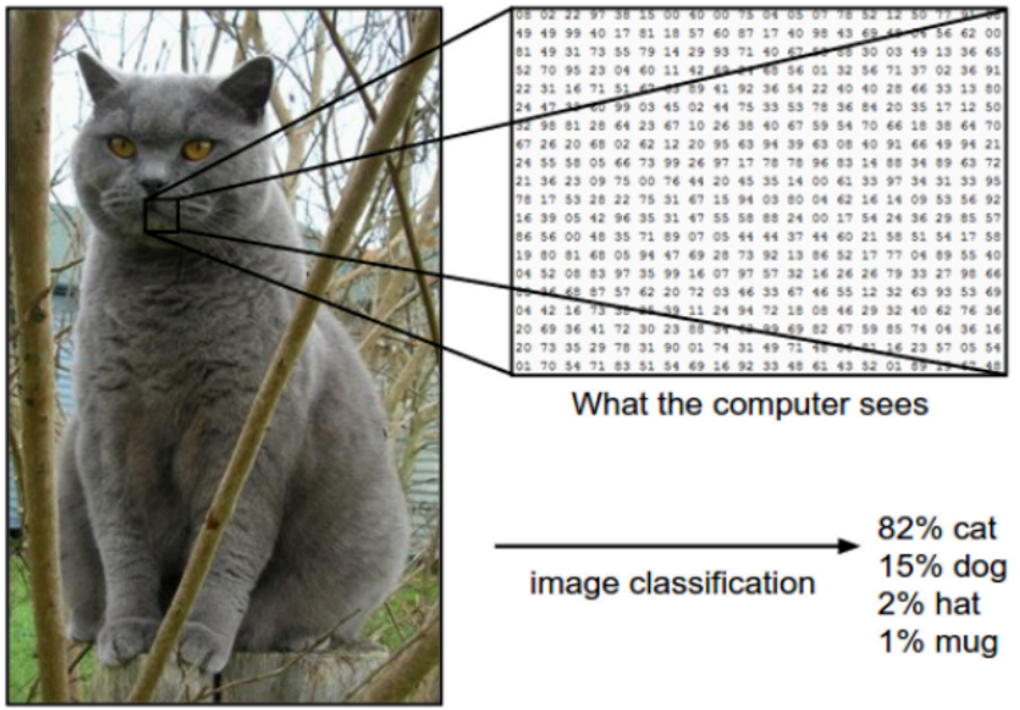
Machine Learning vs Traditional Programming





ML vs Traditional Programming

The Challenge: Unstructured Data

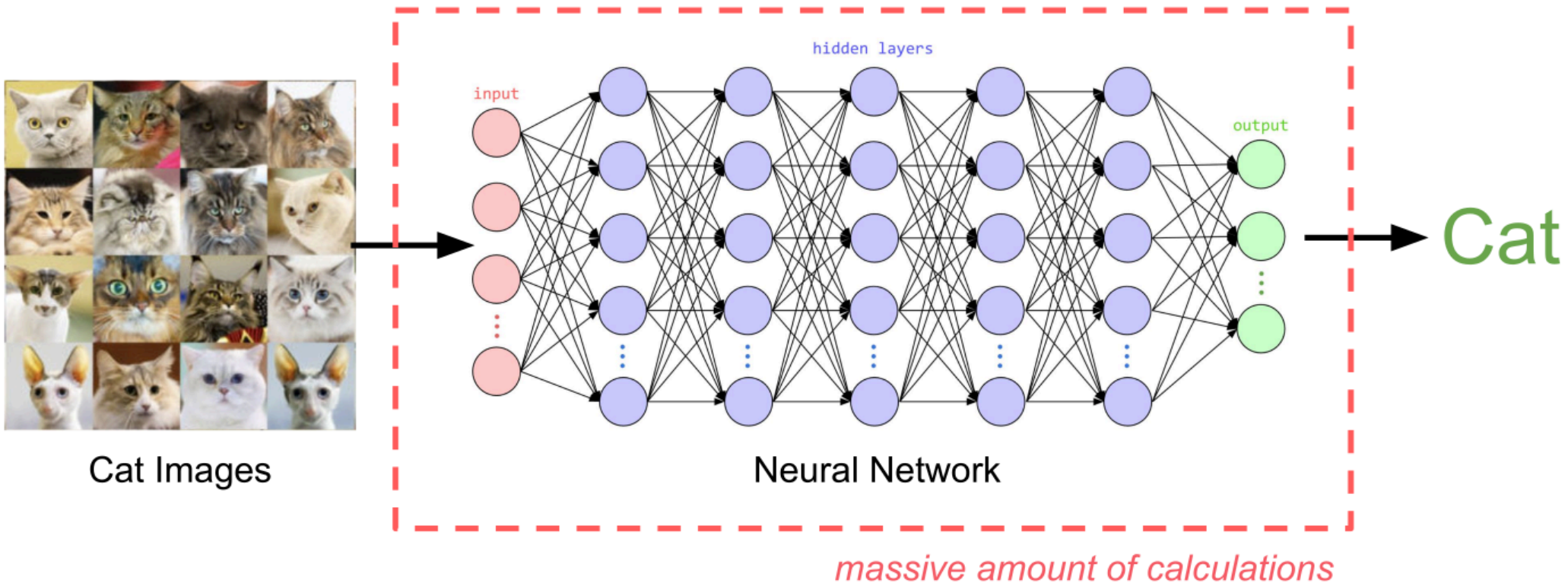


[Image credit: Andrej Karpathy]

High-Dimensional Data

- **224 × 224 image = 50,176 pixels**
- **1080p image = 2,073,600 pixels**

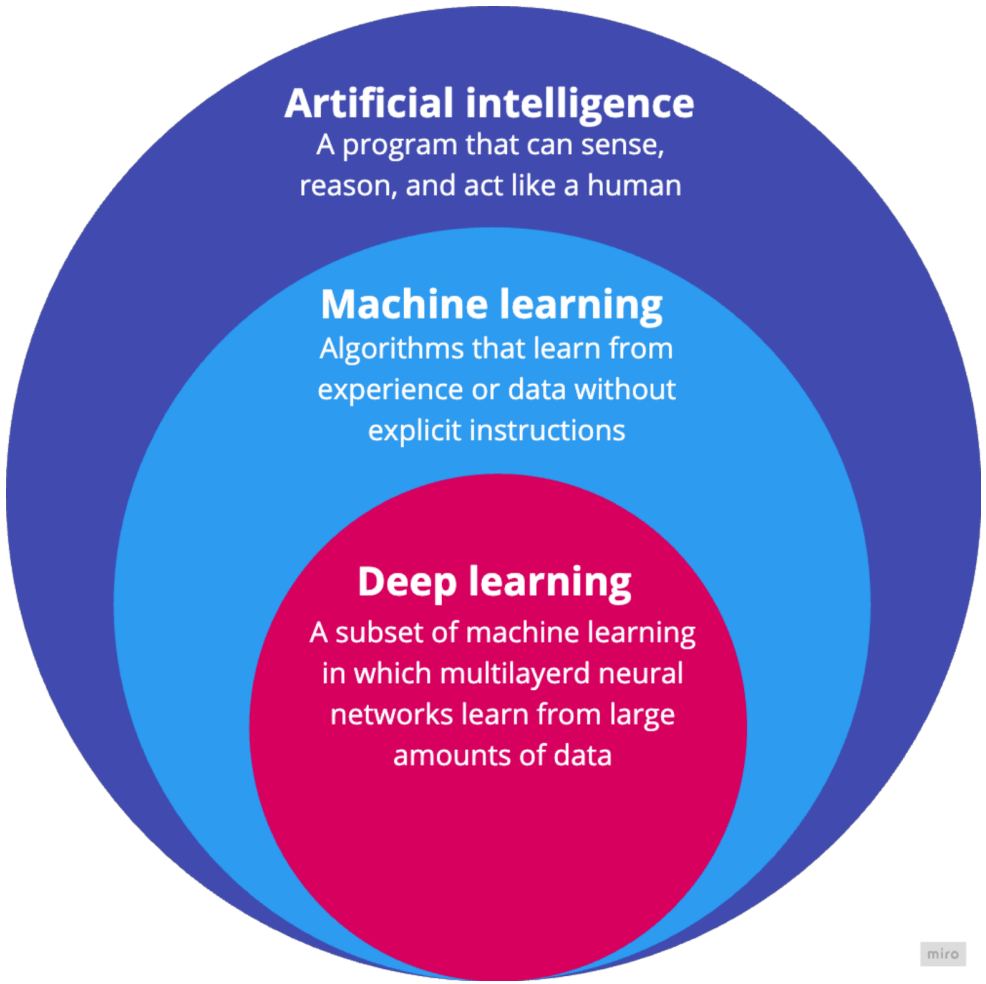
Neural Networks: Universal Approximators



Neural Network Processing

Representation Learning from raw data

Deep Learning vs Machine Learning



AI, ML, DL Venn Diagram

ML vs DL

Characteristic	Machine Learning (ML)	Deep Learning (DL)
Performance Envelope	Competitive on structured/tabular data	Dominant in unstructured domains (vision, speech, NLP)
Data Regime	Performs well with small to medium datasets	Typically requires large-scale datasets to generalize well
Computational Cost	Often CPU-friendly; faster training	GPU/TPU-dependent; computationally expensive
Model Class	Linear models, Decision Trees, Random Forests, Gradient Boosting	Multi-layer Neural Networks, CNNs, RNNs, Transformers
Feature Engineering	Relies heavily on manual feature design	Learns hierarchical representations directly

Characteristic	Machine Learning (ML)	Deep Learning (DL)
	informed by domain knowledge	from raw data
Interpretability	Many models are interpretable (linear models, trees)	Largely opaque; “black box” nature requiring post-hoc explainability

Many Deep Learning Applications

Vision:

- LandingAI
- Ultralytics

Language:

- OpenAI
- Gemini

Data Annotation

Models must be fine-tuned on annotated data

Start Simple: The Delivery Problem

Scenario: Local delivery company, 30-minute promise

New order: 7 miles away

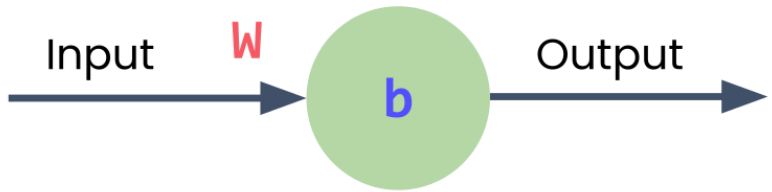
Question: Can you get there in under 30 minutes?

Historical Delivery Data

Distance (miles)	Time (minutes)
5.0	22.2
6.0	25.6
7.0	?

Can you see the pattern?

A Single Neuron = A Linear Equation



Mathematical Unit

`output = W * input + b`

Single Neuron

$$y = Wx + b$$

- **W** = weight (slope)
- **b** = bias (y-intercept)
- **x** = input (distance)
- **y** = output (predicted time)

How Does a Neuron Learn?

1. Start with random weight and bias
2. Make predictions
3. Measure error (how far off?)
4. Use calculus to find adjustment direction
5. Take small step toward better values
6. Repeat hundreds/thousands of times

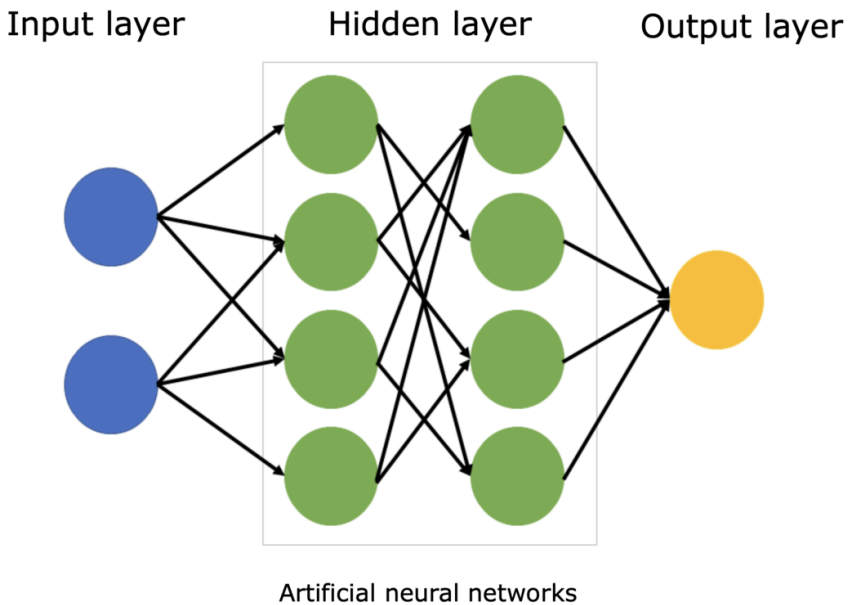
Multiple Inputs

Single input: `distance` → `delivery_time`

Multiple inputs: `distance` + `time_of_day` + `weather` → `delivery_time`

$$y = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$

Layers and Networks

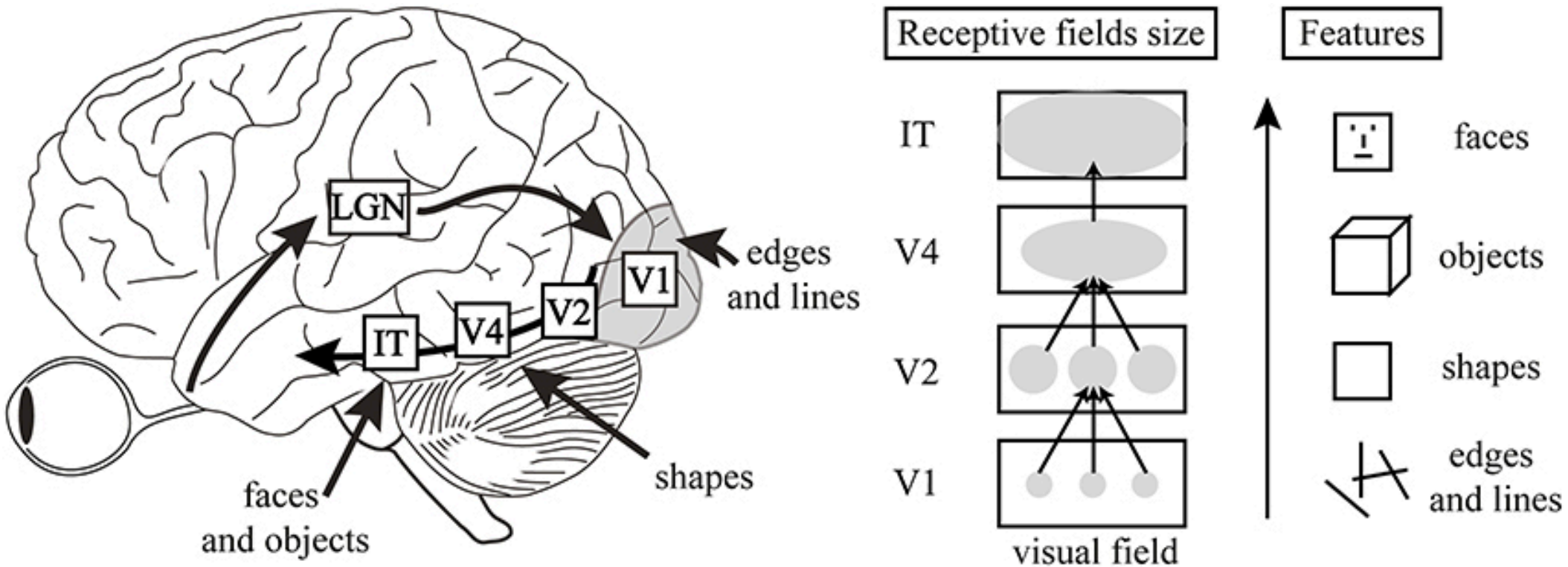


Neural Network Structure

- **Layer:** group of neurons with same inputs
- **Hidden layers:** layers between input and output
- **Network:** connected layers

Analogy: Visual Cortex

Hierarchical Feature Extraction: From Retinal Input to Complex Object Recognition in the Ventral Stream.



Visual Cortex High-Level Features

PyTorch

> PyTorch is an open-source deep learning framework designed to accelerate the path from research prototyping to production deployment. Originally developed by Meta's AI Research lab (FAIR) and released in 2016, it is now governed by the PyTorch Foundation under the Linux Foundation.

PyTorch Logo

Why PyTorch?

```python
import torch
a = torch.tensor([2.0])
b = torch.tensor([3.0])
result = a + b
print(result)  # tensor([5.])
```

Simple. Pythonic. Powerful.

The Problem with Early Frameworks

Static Computational Graphs

- Define everything upfront
- Compile before running
- No flexibility to change
- Cryptic error messages
- No standard Python debugging

PyTorch’s Solution

Dynamic Computation

- Write clean Pythonic code
- Use normal loops and if statements
- Change anything, anytime
- Error messages point to your code
- Standard Python debugging

What’s Next?

- The 6-stage ML pipeline
- Building a neural network in PyTorch
- Training your first model

Click here to go to Session 2: The ML Pipeline and Building Your First Model