Neural Network Programming Fundamentals

Learn the essentials of implementing neural networks effectively

Objective

Goal

Learn the **fundamentals** of neural network programming

Important Techniques

Implement neural networks effectively with:

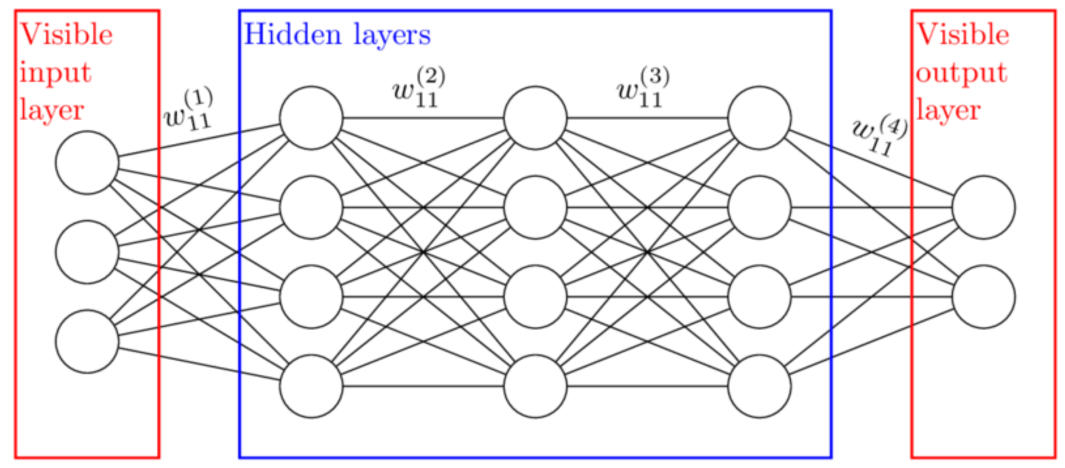
Vectorization

Avoid for-loops → Process entire training set at once

Two-Phase Computation

- Forward Propagation
- Backward Propagation

Simplify concepts using **Logistic Regression** as an introductory model



Binary Classification Example

Task

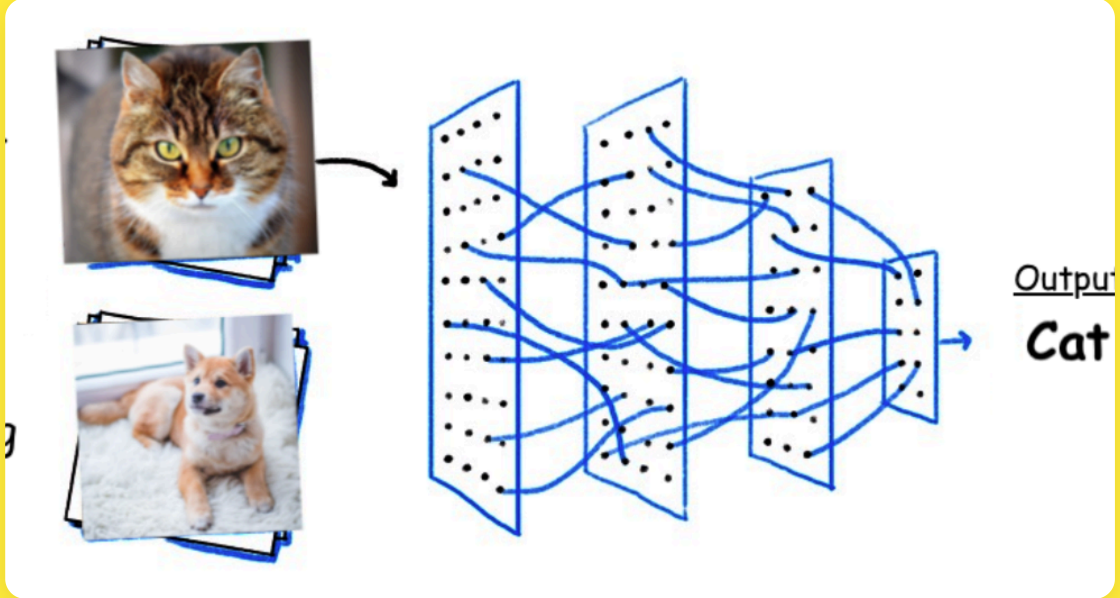
Classify an image as either **cat (1)** or **not cat (0)**

Input (x)

A digital image represented as pixel values

Output (y)

A value of **0** or **1** representing the class



Cat

1

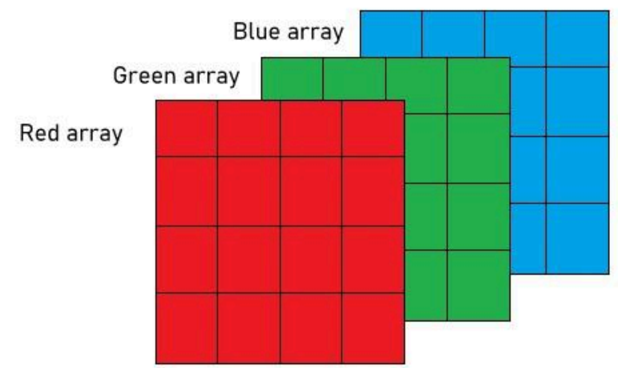
Not Cat

0

Representing an Image in a Computer

Image Storage

An image is stored as **three matrices**, each representing a color channel:



Arrrays stacked over each other
to form a Digital Image.

Red

R	R	R
R	R	R
R	R	R

Green

G	G	G
G	G	G
G	G	G

Blue

B	B	B
B	B	B
B	B	B

Feature Vector

The image is converted into a **long vector** by unrolling all pixel values

Example: 64×64 Pixel Image

3 matrices of 64×64 pixels

$$64 \times 64 \times 3 = 12{,}288$$

Number of input features (nx)

Preparing the Training Data

Single Training Example

A training example consists of a pair **(x, y)**:

x
Input vector
(image features)

→

y
Output label
(0 or 1)

Training Set

Contains **m examples**:

$(x^1, y^1), (x^2, y^2), \ldots, (x^m, y^m)$

Training Set Visualization

Example 1
(x^1, y^1)

Example 2
(x^2, y^2)

Example 3
(x^3, y^3)

...

Example m
(x^m, y^m)

Key Point

Each example consists of an **input vector** and its corresponding **output label**

Representing Data in Matrices

Matrix Organization

To simplify programming and perform **efficient computations**:

Input Matrix X

x_1^1	x_1^2	...	x_1^m
x_2^1	x_2^2	...	x_2^m
...	...	...	...
$x_{n_x}^1$	$x_{n_x}^2$	...	$x_{n_x}^m$

Dimensions: $(n_x \times m)$

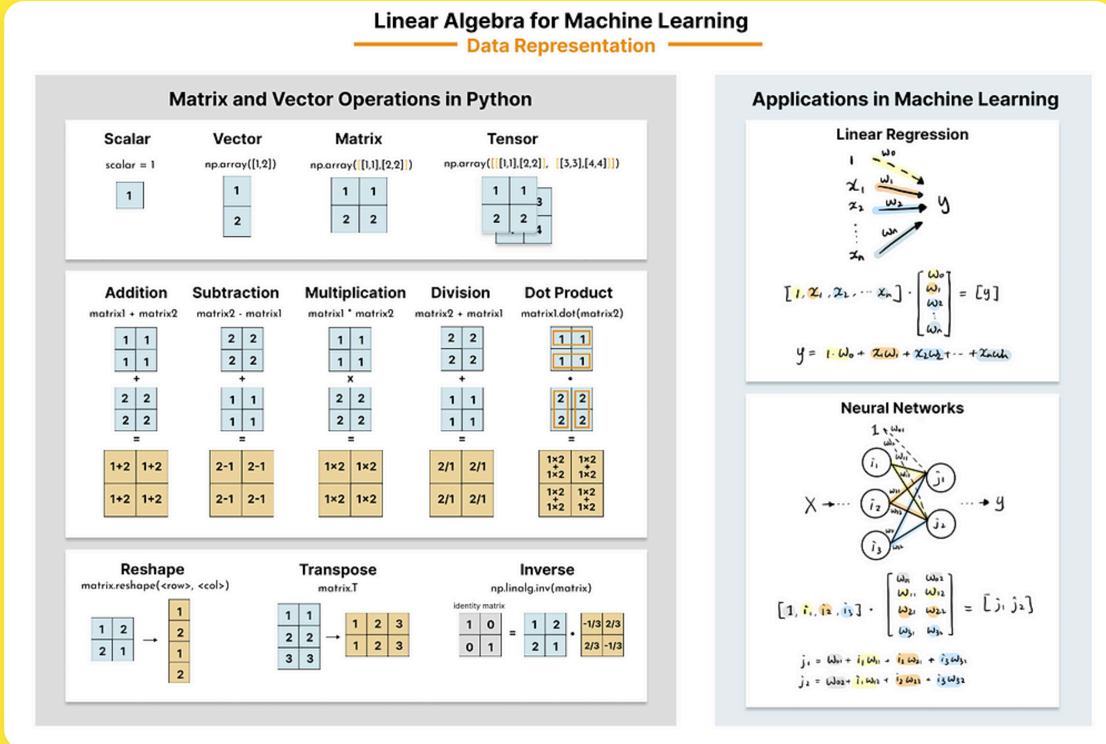
Each column = one training example

Output Matrix Y

y^1	y^2	...	y^m

Dimensions: $(1 \times m)$

Each column = label corresponding to example in X



X Matrix

n_x rows = features
m columns = examples

Y Matrix

1 row = labels
m columns = examples

Benefits

- Faster computation
- Clearer code
- Vectorized operations
- Extends to neural networks

Standard Notation Used

m

Number of Training Examples

Total examples in the dataset

n_x

Number of Input Features

e.g., number of pixels in the image

X

Input Matrix

Size: $(n_x \times m)$

Y

Output Matrix

Size: $(1 \times m)$

Notation Pattern

Single Example		Multiple Examples
(x, y)	→	(X, Y)

$$(x^1, y^1), (x^2, y^2), \ldots, (x^m, y^m) \rightarrow (X, Y)$$

Extension to Neural Networks

This representation pattern is also used when building **multi-layer neural networks**, where:

- Each layer has its own matrices
- Dimensions follow consistent patterns
- Vectorized operations apply across layers

Key Insight

Understanding this notation is **fundamental** to implementing neural networks efficiently