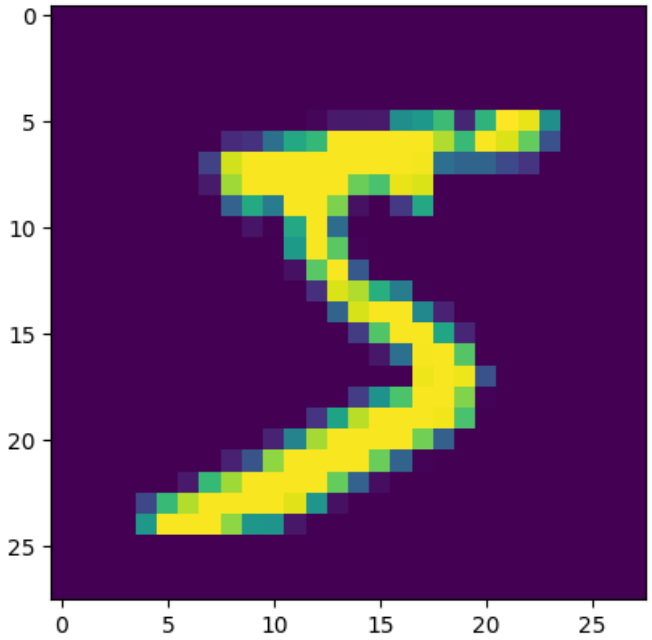
CNNs for Image Processing

Yashwant Bhosale

COEP Tech.

April 13, 2025

Handwritten Digit Recognition

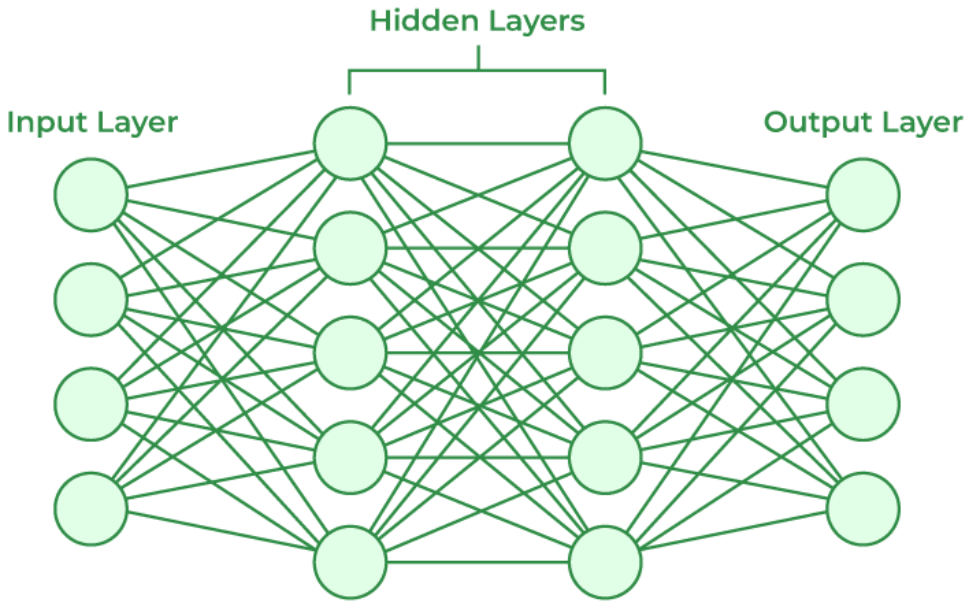


Have you ever wondered how computers can recognize handwritten digits? and given a program that recognizes handwritten digits, can it draw those?

How can humans do this so effortlessly?

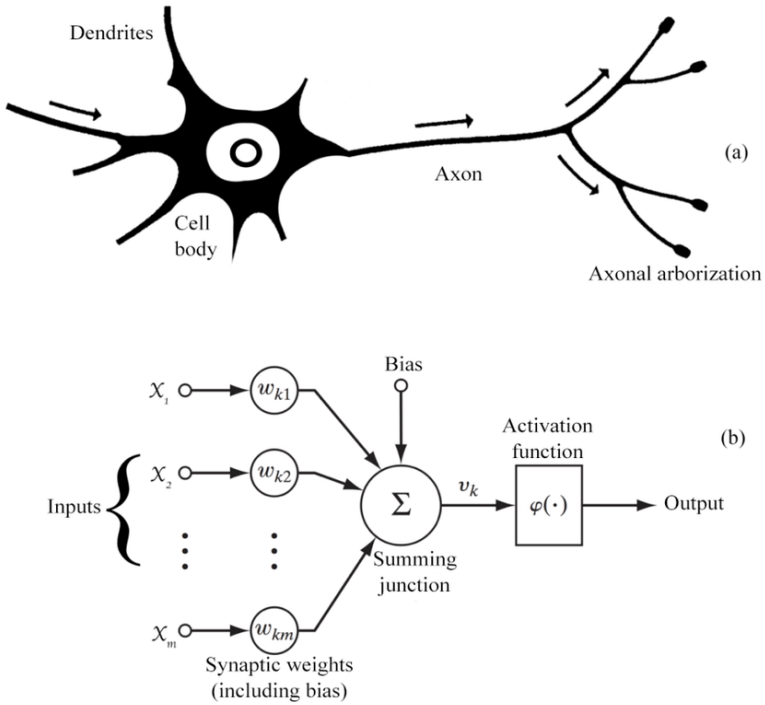
1. Our brains have evolved to be incredibly adept at visual pattern recognition,
2. Humans learn to recognize digits through exposure to a variety of handwriting samples over time, while NNs require large, labeled datasets to train and learn features automatically.
3. Humans can easily adapt to different handwriting styles, interpret ambiguous digits based on context, and recognize digits even if they are slightly distorted or rotated, whereas NNs might struggle with such variations unless specifically trained on diverse data.

But, what is NN?



NN means **Neural Networks**
As the name suggests there are **Neurons** in the neural network.
and there are **layers** of these **neurons**.
In a bird's eye view, there are 3 types of layers: **input layer**, **hidden layer** and **output layer**

Why does network of neurons behave intelligently?



Neural networks appear "intelligent" because they can learn and adapt to complex patterns in data by adjusting internal parameters **(weights and biases)** through a process called **backpropagation**, effectively mimicking the way the human brain learns by making connections between neurons based on experience, allowing them to make predictions and classifications on new data with a high degree of accuracy, especially when dealing with non-linear relationships between inputs and output.

Math behind Neural Networks

Linear Transformation

In a neural network, we have layers of neurons. Each neuron is connected to the neurons in the previous layer through weighted connections. Mathematically, this connection can be represented as:

$$Z = W \cdot X + b$$

- Z is the linear combination of inputs X, weights W, and bias b.
- W represents the weights associated with each connection.
- X is the input data
- b is the bias term

Math behind Neural Networks

Activation Function

The output of a neuron is usually passed through an activation function, which introduces non-linearity to the model. The most commonly used activation functions are:

- Sigmoid Function:

$$A = \frac{1}{(1 + e^{-Z})}$$

- ReLU (Rectified Linear Unit):

$$A = max(0, Z)$$

Math behind Neural Networks

Loss Function The loss function measures how well the model is performing. For classification tasks like ours, we often use cross-entropy loss

$$L(y, \hat{y}) = -\sum(y_i * log(\hat{y}_i))$$

- y is the actual label
- $\hat{y}$ is the predicted label

Math behind Neural Networks

The model learns by minimizing the loss function. Gradient descent is the optimization technique to update the model's parameters (weights and biases) iteratively.

$$W_{new} = W_{old} - \eta \cdot \frac{\partial L}{\partial W}$$

$$b_{new} = b_{old} - \eta \cdot \frac{\partial L}{\partial W}$$

here η is Learning Rate

Activation Function	Formula	Range
main-1.pdf	$\sigma(x) = \frac{1}{1+e^{-x}}$	(0, 1)
ReLU	$f(x) = \max(0, x)$	$[0, \infty]$
Tanh	$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	(-1, 1)