# Digit Recognition Using Neural Networks and OpenCV

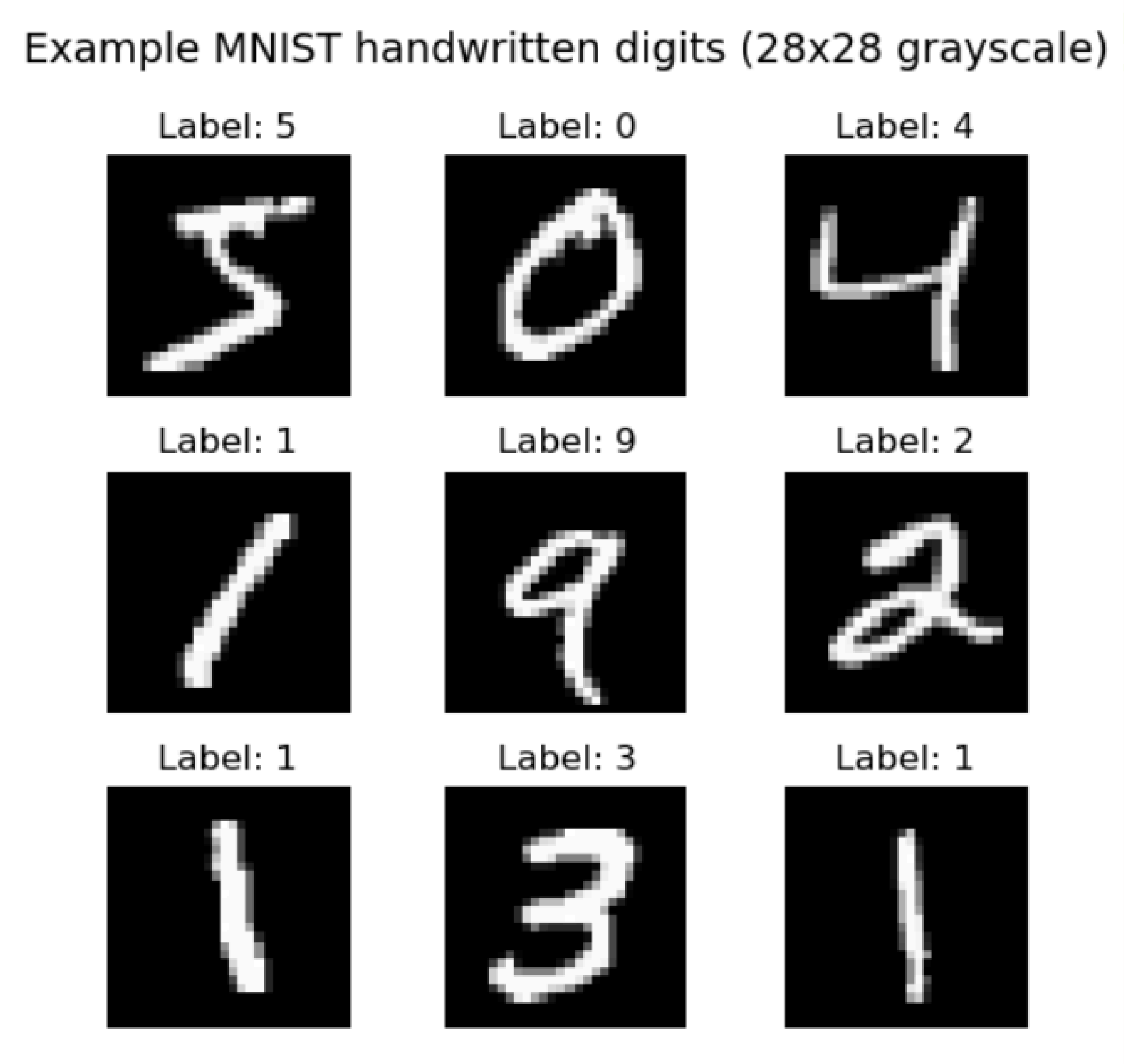
Java Implementation with MNIST Dataset

**Objective:**

- Automatically classify handwritten digits (0–9) using machine learning.

**Dataset Used:**

- MNIST Dataset
  - 60,000 training images
  - 10,000 test images
  - Each image: 28×28 grayscale
  - centered digit



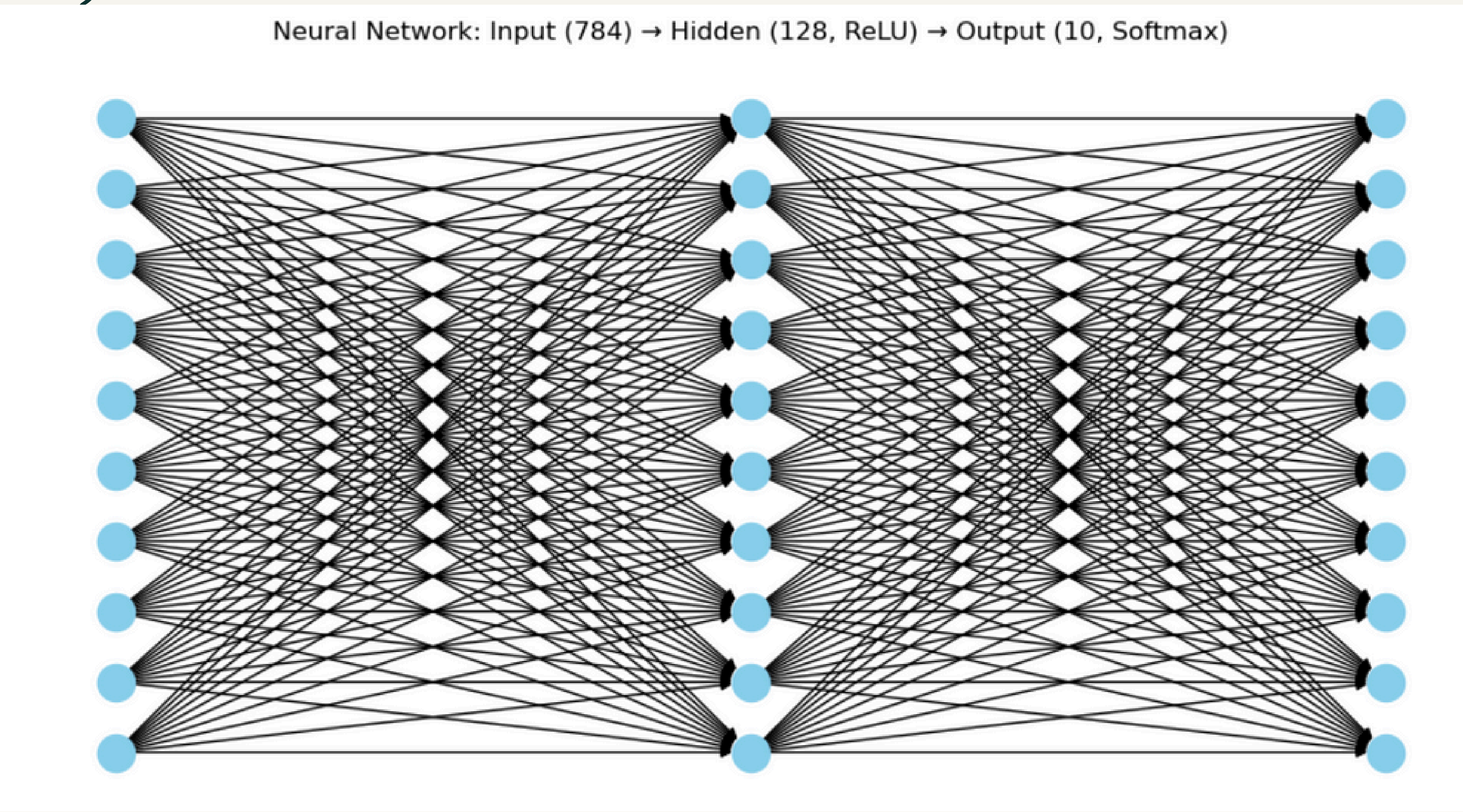
# Neural Network Basics

## Definitions:

- Neural Network: A layered computational model inspired by the human brain.
- Neuron: Basic computation unit that applies a weighted sum followed by an activation function.
- Activation Function: Introduces non-linearity (e.g., ReLU, Softmax).

## Used Architecture:

- Input layer: 784 neurons (28×28 pixels)
- Hidden layer: 128 neurons (ReLU)
- Output layer: 10 neurons (Softmax)



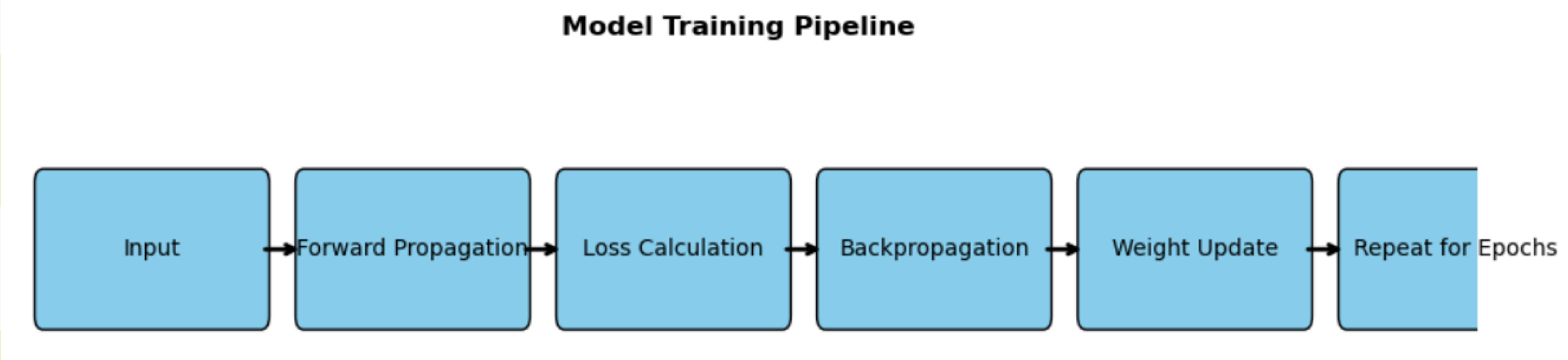
**Model Training Pipeline**

**Training Procedure:**

1. **Forward Propagation**
   - Compute activations layer by layer
2. **Loss Computation**
   - Cross-Entropy Loss:
3. **Backpropagation**
   - Compute gradients using chain rule
4. **Parameter Update**
   - Gradient Descent with learning rate α

**Batch Size: 32**

**Epochs: 10**



## MNIST Data Loader

File Format: IDX (binary format)

- train-images.idx3-ubyte: images
- train-labels.idx1-ubyte: labels

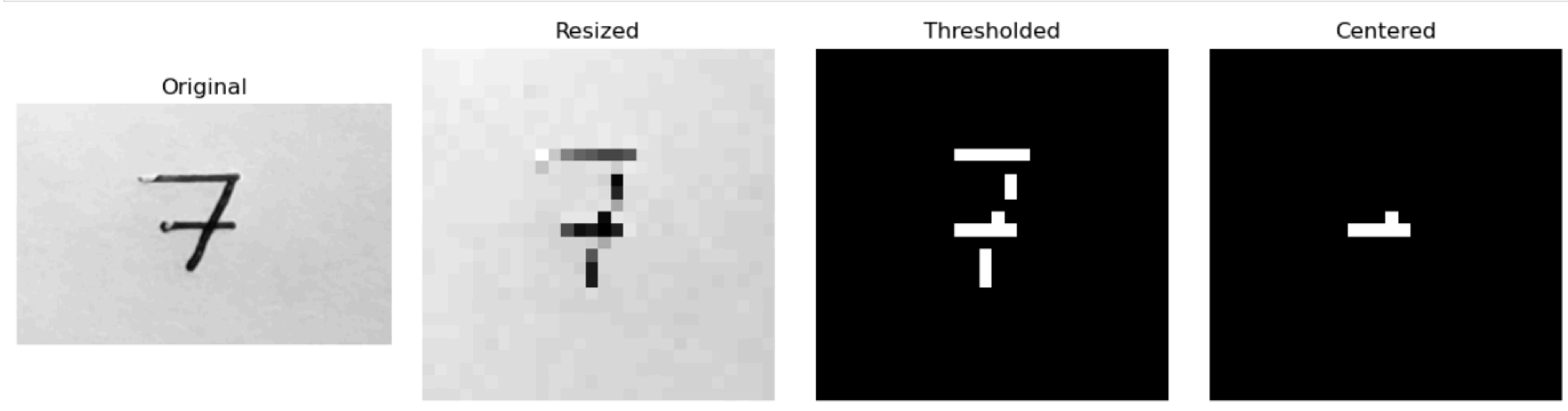
**Process:**

- Parse headers (magic number, number of items)
- Read bytes, normalize pixel values to [0, 1]
- Convert labels to one-hot encoding

# Image Preprocessing with OpenCV

## Steps:

1. Read Image – Grayscale
2. Resize – 28×28
3. Thresholding – Binarize image using Otsu's method
4. Contour Detection – Identify digit's bounding box
5. Centering – Digit centered in 28×28 canvas
6. Normalization – Pixel values ∈ [0, 1]
7. Flattening – Convert to 784-length vector

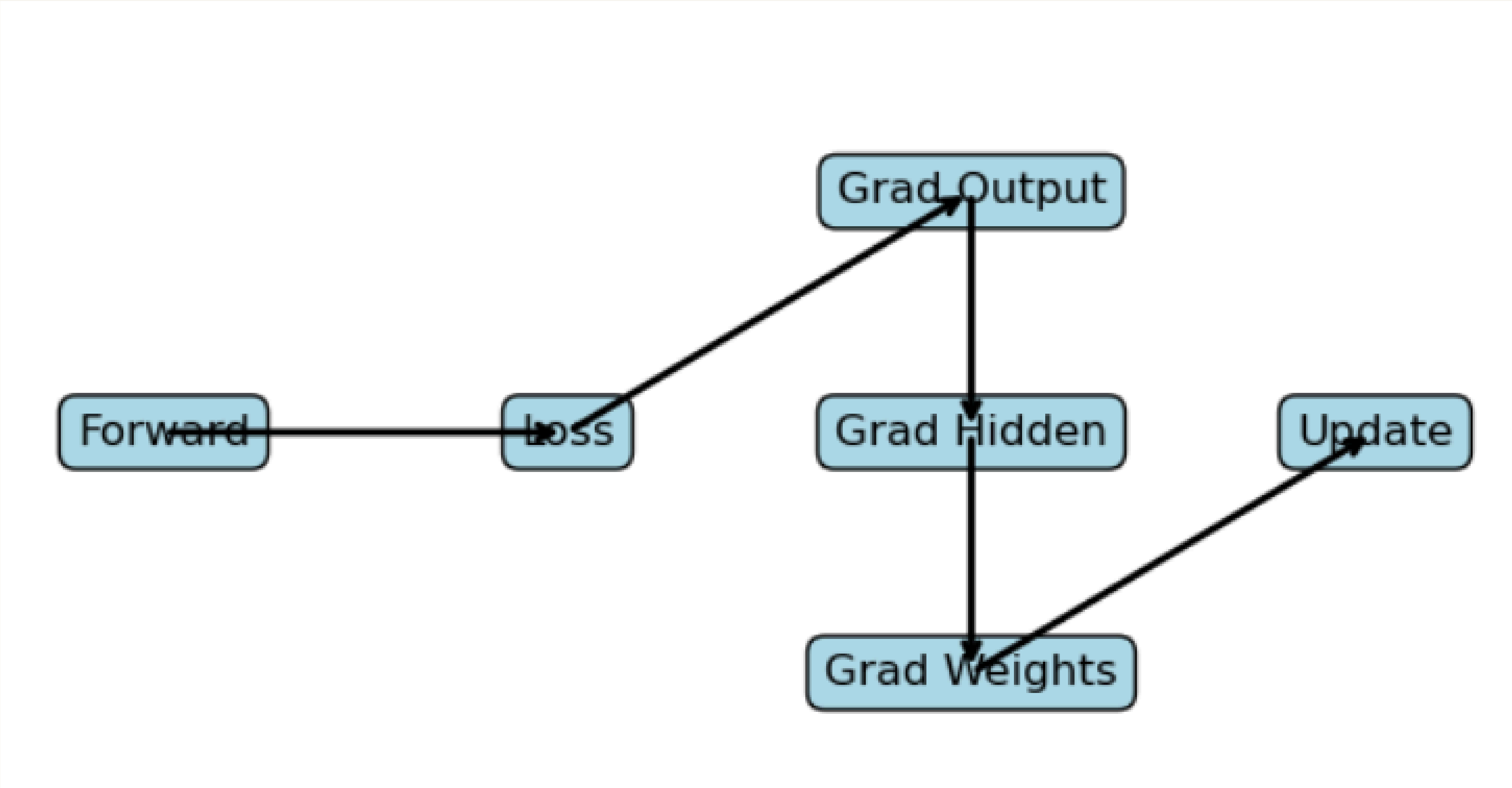


# Activation Function

$$\text{ReLU: } f(x) = \max(0, x)$$

$$\text{Softmax: } \sigma(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

# Backpropagation Logic



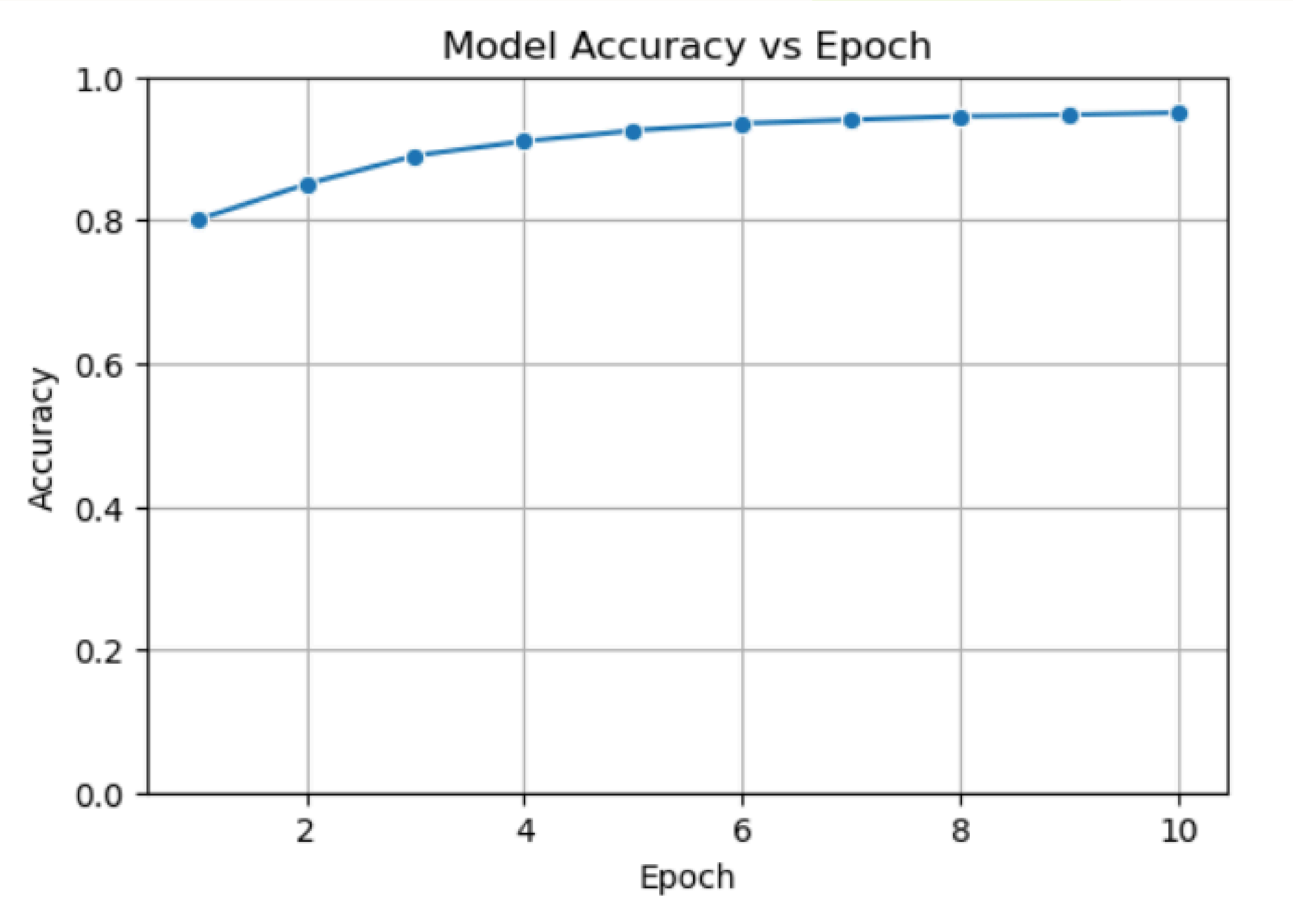
**Evaluation Metrics**

**Accuracy:**

- Accuracy = (Correct Predictions) / (Total Predictions)
- This metric measures how many of the total predictions made by the model were correct.

**Example Output:**

- Epoch 10/10 – Loss: 0.0723 – Accuracy: 97.85%



# Conclusion

- Developed a working digit recognizer  from scratch
- Implemented full training logic without ML libraries
- Preprocessed real-world images with OpenCV
- Achieved high accuracy on a standard benchmark


HOPE YOU ENJOYED MY PRESENTATION
THANK YOU FOR LISTENING