# Simple MNIST Neural Network

## Assignment Report

### Under the guidance of:

### Prof. Dr.-Ing. Milan Gnjatovi´c

Presented by:

Rudradeep Palit

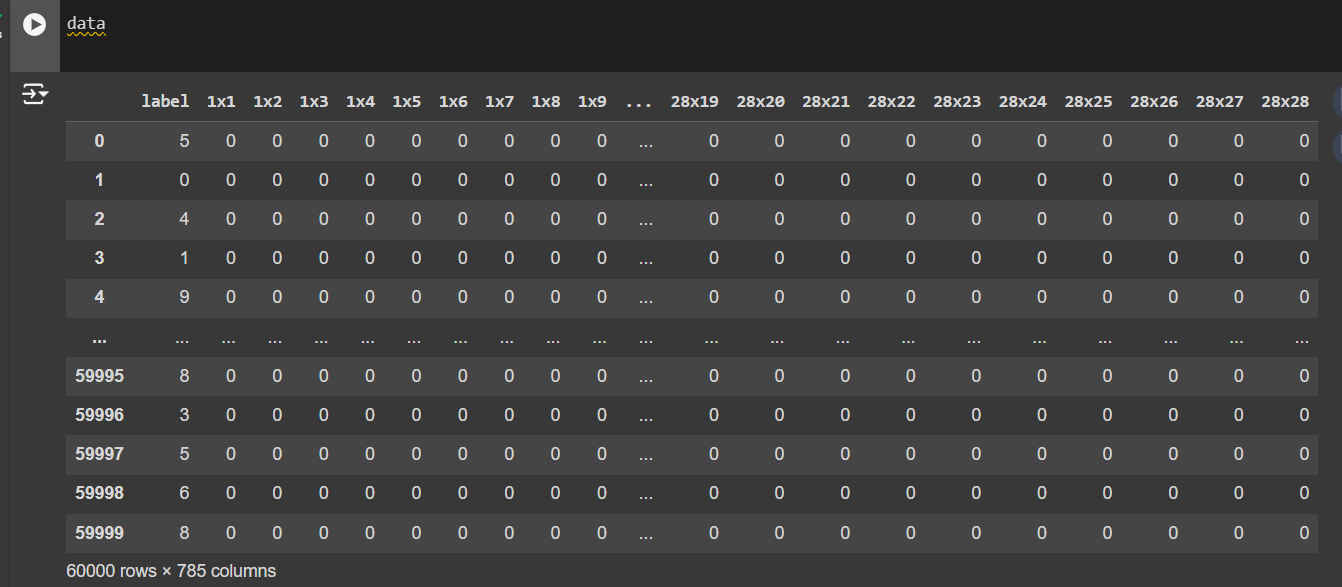
11038108

## 1. Introduction

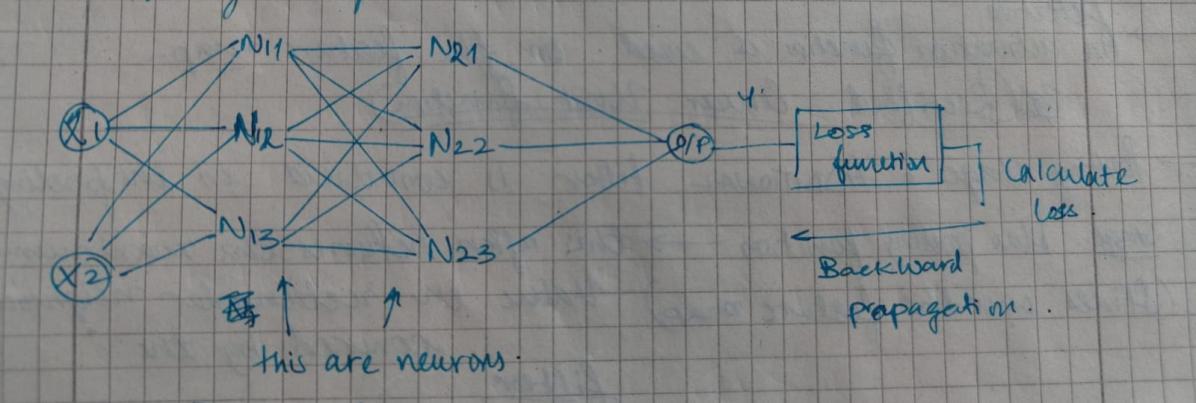
This project involves building a simple neural network from scratch to classify handwritten digits from the MNIST dataset using only Numpy, without leveraging any deep learning frameworks like TensorFlow or Keras. The goal is to understand the fundamental building blocks of a neural network and implement them manually.

## 2. Dataset

The MNIST dataset consists of 70,000 grayscale images of handwritten digits, with 60,000 images for training and 10,000 images for testing. Each image is 28x28 pixels, flattened into a 784-dimensional vector for processing.



## Network Architecture



The implemented neural network has the following architecture:

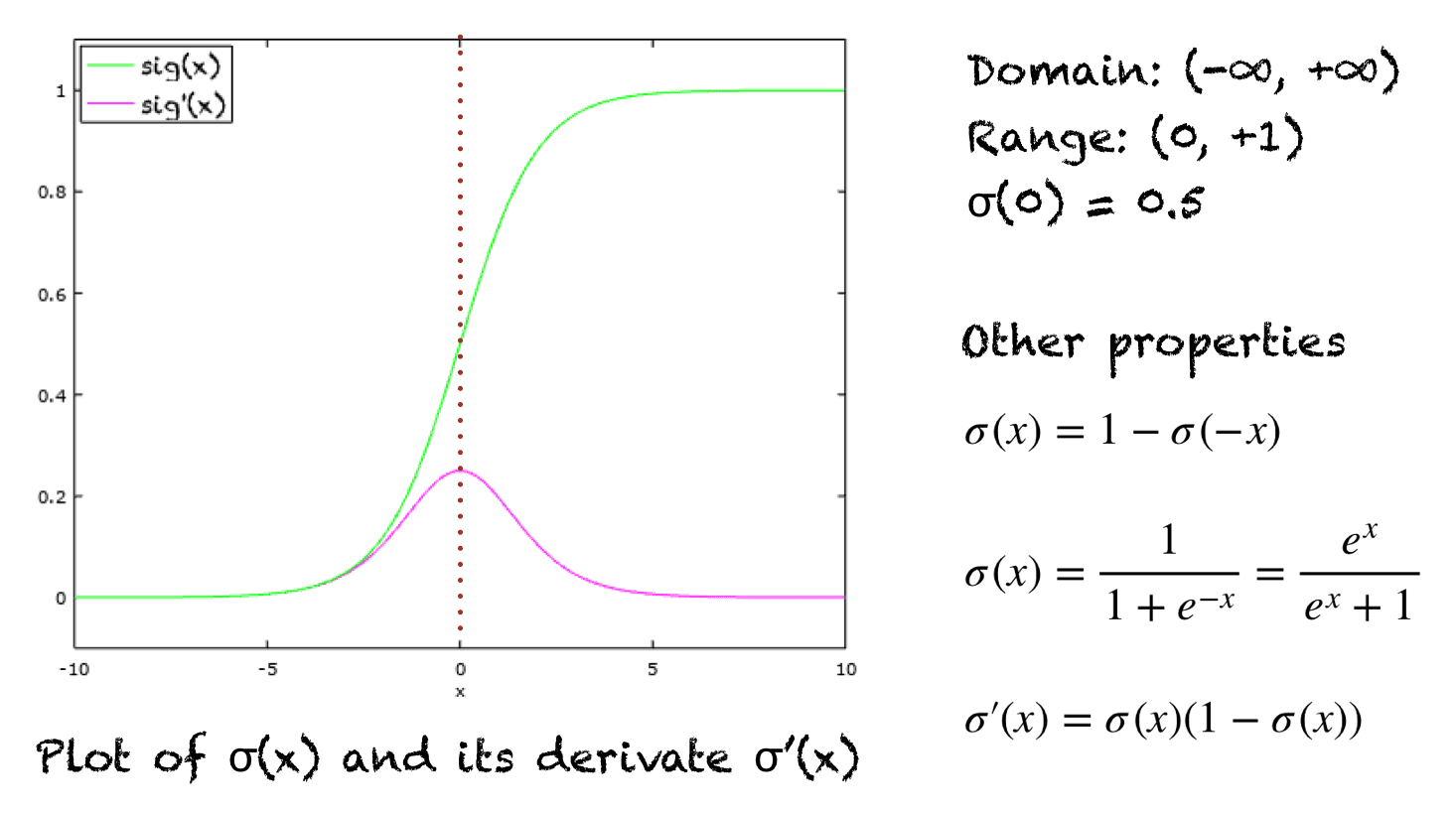
Input Layer: 784 neurons (one for each pixel)

Hidden Layer: 128 neurons

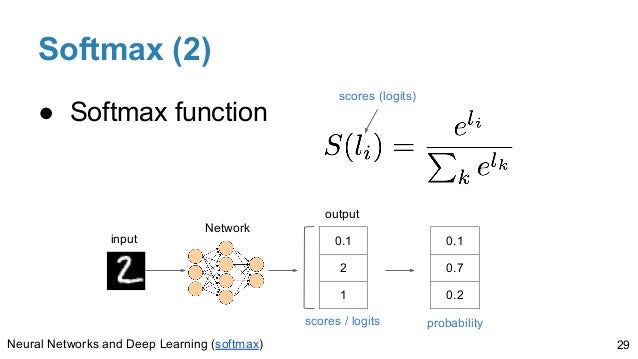
Output Layer: 10 neurons (one for each digit from 0 to 9)

## 4. Activation Functions

Sigmoid Function: Used in the hidden layer to introduce non-linearity.

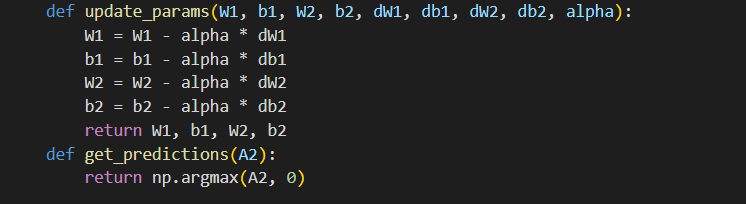


Softmax Function: Applied to the output layer to convert the network’s outputs into probability distributions.



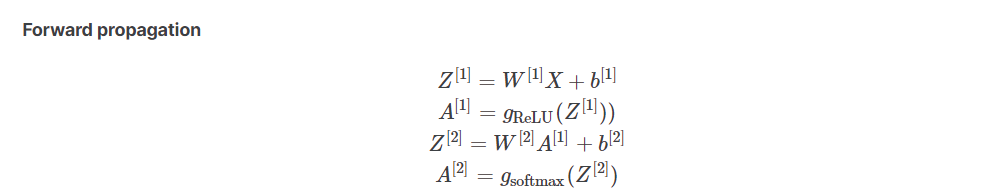
## 5. Loss Function

Cross-Entropy Loss: Measures the difference between the predicted probabilities and the actual labels.



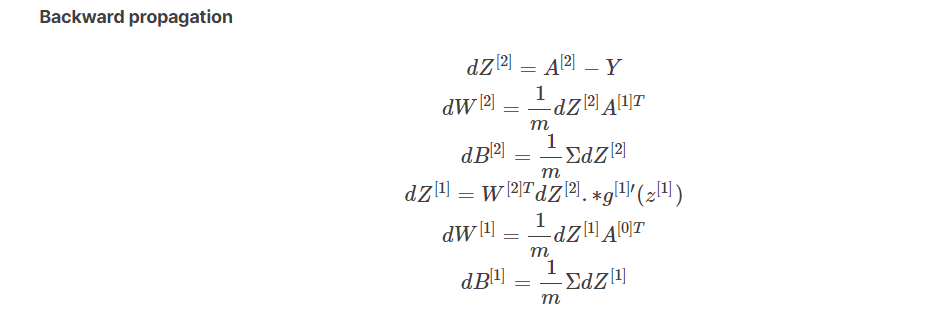
## 6. Forward Propagation

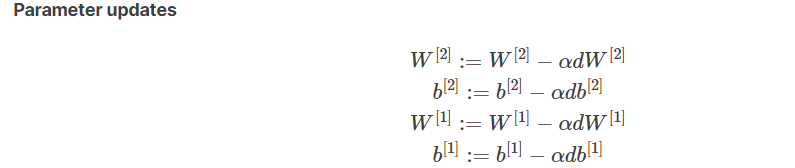
Forward propagation involves computing the activations of each layer using the weights and biases:



7. Backward Propagation

Backward propagation involves computing the gradients of the loss with respect to weights and biases, and updating them using gradient descent:





## 8. Training Process

The network is trained for a specified number of epochs, using mini-batch gradient descent:

**Epochs**: Number of times the entire dataset is passed through the network.

**Batch Size:** Number of samples processed before updating the weights.

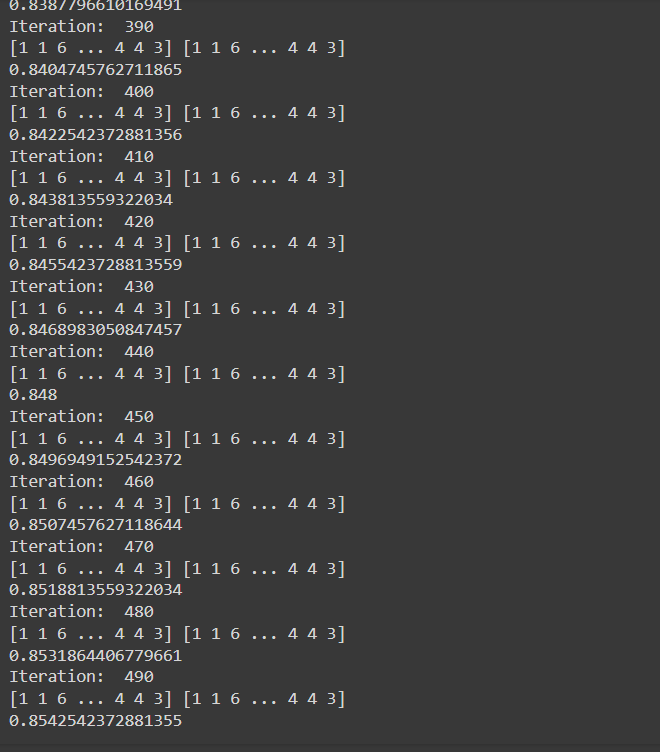
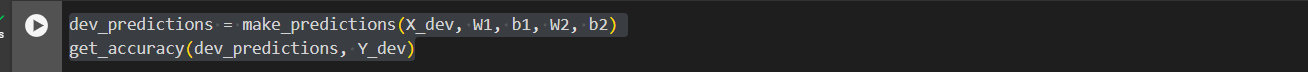
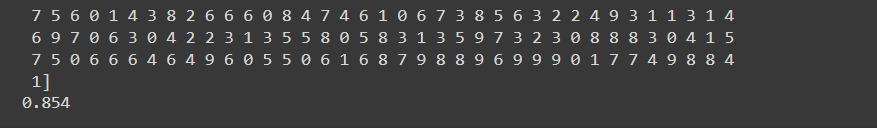


fig .Iterations

## 9. Results

The model achieves satisfactory accuracy on the test set, demonstrating the ability to classify handwritten digits.





## 10. Conclusion

Implementing a neural network from scratch provides valuable insights into the inner workings of neural networks, including forward and backward propagation, activation functions, and gradient descent. We have achieved an accuracy of 85%, which we can further improve by :

To improve the accuracy of the neural network for the MNIST digit classification task, you can consider the following strategies:

1. Increase Model Complexity

Add More Layers: Introducing additional hidden layers can help the model learn more complex representations.

Deeper Forward Propagation

2. Use Different Activation Functions

ReLU:Include Leaky ReLU, which allows a small gradient when the unit is not active.

3. Regularization

Dropout: Prevents overfitting by randomly setting a fraction of input units to 0 at each update during training time.

4. Data Augmentation

Augment Data: Increase the size and variability of your training set by applying transformations such as rotations, shifts, and zooms.

5. Hyperparameter Tuning

Learning Rate: Adjust the learning rate for optimal performance.

Batch Size and Epochs: Experiment with different batch sizes and the number of epochs.

6. Batch Normalization

Normalize Inputs: Helps to accelerate training and stabilize the learning process.

Reference:

* <https://datascience.stackexchange.com/questions/57005/why-there-is-no-exact-picture-of-softmax-activation-function>
* <https://www.datacamp.com/tutorial/mastering-backpropagation>
* <https://github.com/Ashutosh27ind/neuralNetworkMNSITdataFromScratch>