A Deep Learning Approach for Handwritten Digit Classification Using the MNIST Dataset

Maitreya Ganu , IISER Thiruvananthapuram

13 th December 2024

Abstract

The MNIST (Modified National Institute of Standards and Technology) dataset is one of the most widely used datasets in the field of machine learning and computer vision. It contains 70,000 grayscale im- ages of handwritten digits (0-9), with 60,000 images used for training and 10,000 for testing. This project explores the application of deep learning techniques, specifically a neural network, to classify handwritten digits from the MNIST dataset. The model is trained on the training set, validated on a subset of the data, and evaluated on the test set to assess its classification accuracy. The results demonstrate the effectiveness of deep learning in achieving high accuracy for digit classification, and the performance is compared with traditional machine learning algorithms.

1 Introduction:

In this project , I have used Deep Neural Networks (DNNs) to classify the hand-written digits in the MNIST data set and was able to achieve an accuracy of 99.96%. The primary goal of this project is to explore the potential of deep learning in solving basic image classification tasks and to understand how neural networks can be trained, validated, and tested for optimal performance. I have used heatmaps and graphs to ensure a good readability to the reader . The following libraries were used in this project : tensorflow , numpy , matplotlib and seaborn.

2 Data Preprocessing:

In the MNIST dataset, the pixel values of images range from 0 to 255, where 0 corresponds to black and 255 corresponds to white, with intermediate values representing shades of gray. To prepare the data for training a machine learning model, it is common practice to normalize the pixel values, scaling them from the range $[0, 255]$ to $[0, 1]$.

2.1 Why Normalize?

Normalization is an essential preprocessing step in machine learning, especially when using neural networks. The key benefits of normalization include:

- Neural networks generally perform better when input features are within a smaller range, such as $[0, 1]$.
- Normalization accelerates convergence during training by preventing gradients from becoming too large or too small during backpropagation.
- It ensures that all input features (in this case, pixel values) are on the same scale, avoiding any one feature from dominating the learning process.

2.2 Effect of Normalization

By normalizing the pixel values, the model can learn more effectively and efficiently. This scaling ensures that the pixel values are uniformly distributed within a range that is easier for the model to process, leading to faster training times and improved performance.

3 Model Architecture and Training:

In this section, we define and train a neural network using the Keras Sequential API. The model consists of three layers:

- **Flatten layer:** The input images are 28x28 pixels in size. The Flatten layer is used to convert the 2D array of pixels into a 1D array of 784 elements (28×28), making it suitable for input into fully connected layers.
- **Dense layer (Hidden layer):** This fully connected layer has 100 neurons and uses the `ReLU` (Rectified Linear Unit) activation function. ReLU is commonly used for hidden layers as it introduces non-linearity and helps with gradient propagation during training.
- **Dense layer (Output layer):** The final output layer contains 10 neurons, corresponding to the 10 classes of handwritten digits (0-9). The Sigmoid activation function is applied here, which outputs values between

0 and 1, though softmax is generally more appropriate for multi-class classification problems. In this case, sigmoid will treat each class independently.

The model is compiled with the adam optimizer, a popular optimization algorithm that adapts learning rates based on training progress. The loss function used is sparse categorical cross entropy, suitable for classification tasks where the target labels are integers instead of one-hot encoded vectors. The accuracy metric is used to track the model's performance during training. The model is then trained on the training data for 100 epochs and the accuracy and the loss was recorded during each epoch . The confusion matrix is visualized as a heat map which allows a good interpretation of the final output of the DNN.

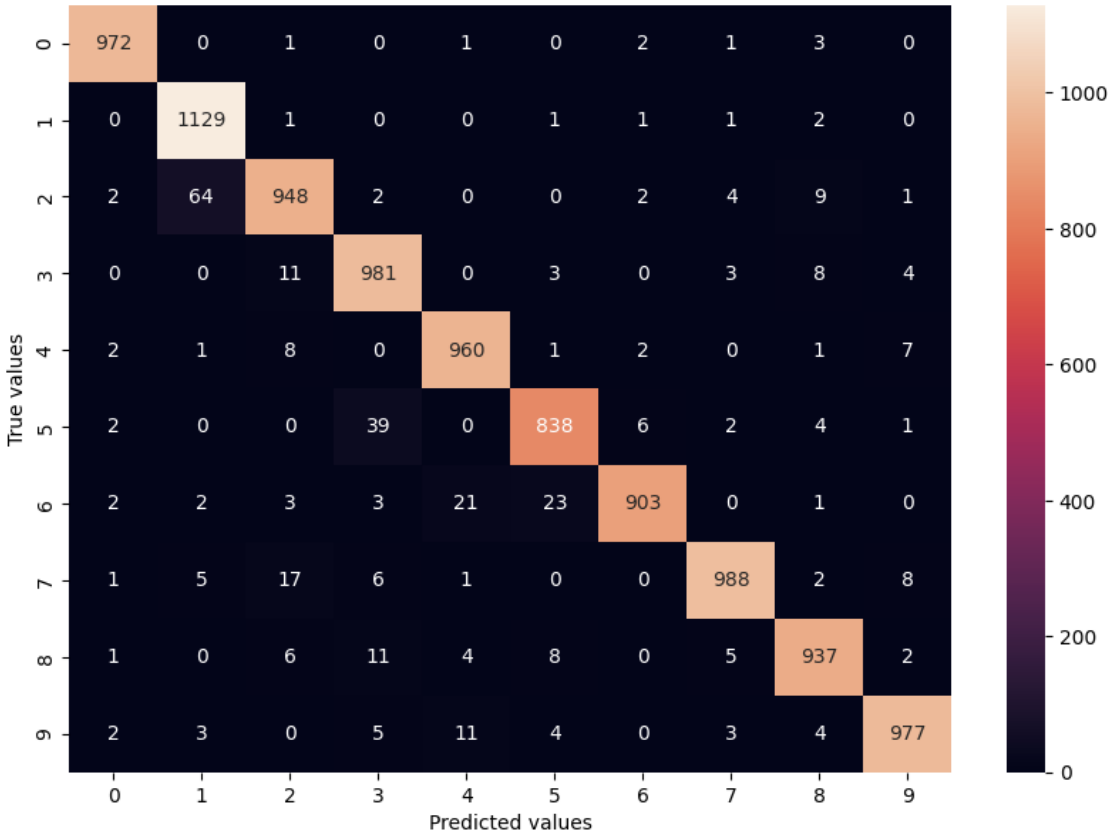

Figure 1: Confusion matrix visualized as a heat map

The following graph shows how the accuracy varies with the number of epochs . Since we had converted our data into the range $[0, 1]$, the accuracy is very close to 1.

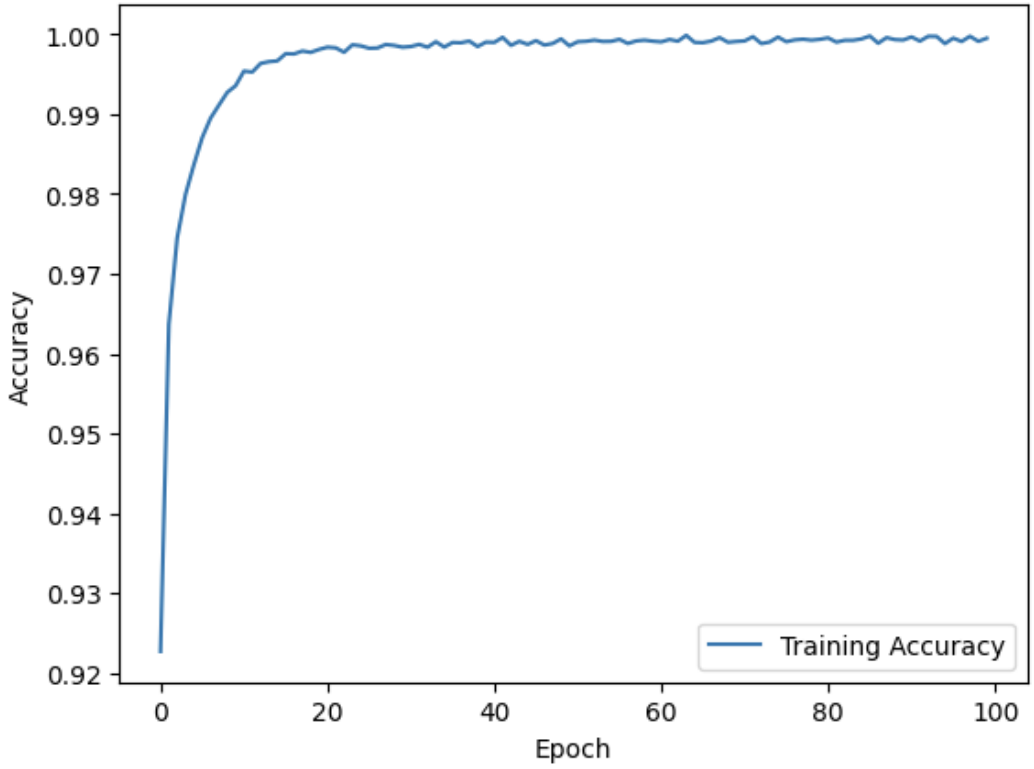

Figure 2: Variation of accuracy of the model with number of epochs

4 Conclusion:

In this project, we successfully applied a Deep Neural Network (DNN) to classify handwritten digits from the MNIST dataset. The normalization of the input data, scaling the pixel values to the range $[0, 1]$, helped improve the model's accuracy. As a result, the accuracy approached 1, indicating that the model was effectively learning to classify the digits. The confusion matrix was visualized as a heatmap, providing a clear interpretation of how the model's predictions compared to the true labels. Overall, the model performed well on the MNIST dataset, achieving high accuracy, and demonstrated the effectiveness of deep learning in image classification tasks.

5 Code and Data Availability:

- Access the code for this project :
  MNIST Project Code
- Access the data set used in this project by importing it through keras.