Hands-on TensorFlow: Handwritten Digit Recognition

Fatmi Firdaous

March 15, 2025

1 Introduction

This report presents the development of a fully connected neural network for handwritten digit recognition using the MNIST dataset.. The model is trained using TensorFlow and evaluated based on its accuracy and loss metrics.

2 Mnist DataSet Overview

The MNIST dataset is a benchmark dataset for handwritten digit recognition tasks. It contains a total of 70,000 images, with 60,000 designated for training and 10,000 for testing. Each image is a 28x28 grayscale representation of a digit between 0 and 9.

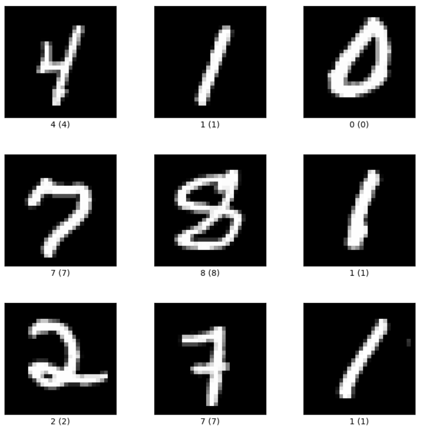


Figure 1: Mnist Dataset

3 Methodology

3.1 Data Preprocessing

The MNIST dataset is loaded using `tf.keras.datasets.mnist`. It is then split into training and testing sets.

Normalization

The pixel values of the images are scaled from the original 8-bit (0-255) range to a normalized range of [0,1] by dividing each pixel by 255.

One-hot encoding of label

The labels are converted to one-hot encoded to facilitate multi-class classification.

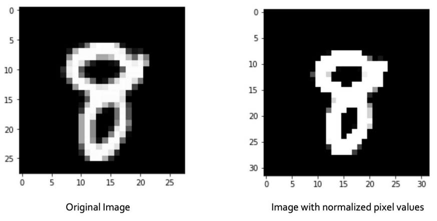


Figure 2: Digit after normalisation

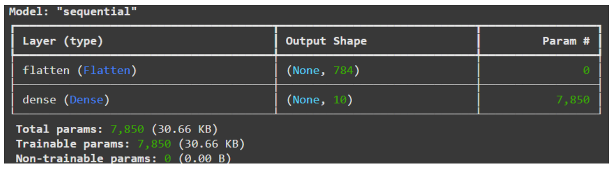
4 Modeling and Evaluation

4.1 Single Hidden Layer

he initial model is a one-layer neural network consisting of a fully connected (dense) layer with a softmax activation function. The input layer flattens the 28x28 image into a 784-dimensional vector.

- Input: Flattened 28x28 grayscale image (784 values)
- Output: 10 neurons (softmax activation for digit classification)

The model architecture is summarized using `model.summary()` to display the number of trainable parameters.



```
Model: "sequential"
┌──────────────────────────────┬──────────────────────┬─────────────┐
│ Layer (type)                 │ Output Shape         │     Param # │
├──────────────────────────────┼──────────────────────┼─────────────┤
│ flatten (Flatten)            │ (None, 784)          │           0 │
├──────────────────────────────┼──────────────────────┼─────────────┤
│ dense (Dense)                │ (None, 10)           │       7,850 │
└──────────────────────────────┴──────────────────────┴─────────────┘
 Total params: 7,850 (30.66 KB)
 Trainable params: 7,850 (30.66 KB)
 Non-trainable params: 0 (0.00 B)
```

Figure 3: Model's summary

Model Training and Evaluation

The model is trained using Stochastic Gradient Descent (SGD) with the following hyperparameters:

- Learning rate: 0.01
- Mini-batch size: 32
- Number of epochs: 30
- Validation set size: 25 percent of the training set

```
optimizer =
    optimizers.SGD(learning_rate=0.01)
model.compile(optimizer=optimizer,
loss='categorical_crossentropy',
metrics=['accuracy'])
```

Loss and Accuracy Evolution

The training loss and accuracy, along with the validation loss and accuracy, are plotted over epochs. The plots indicate whether the model is learning effectively and if overfitting occurs.

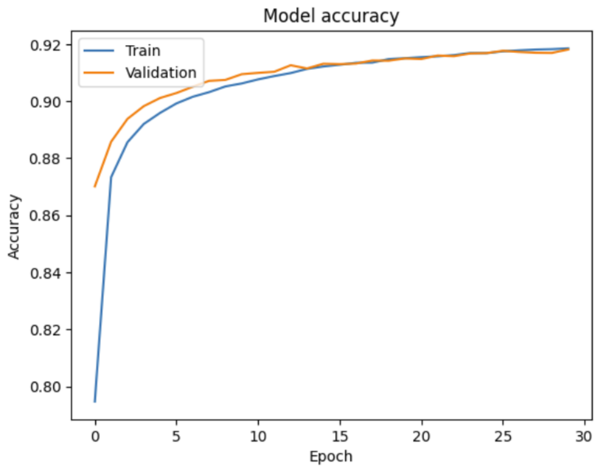


Figure 4: Training and validation accuracy trends.

Key points:

- The training accuracy starts lower and gradually improves, converging with the validation accuracy.
- The validation accuracy is slightly higher than the training accuracy in some instances, suggesting good generalization.
- Both accuracies stabilize at around 92%, indicating that the model has achieved a high level of performance.

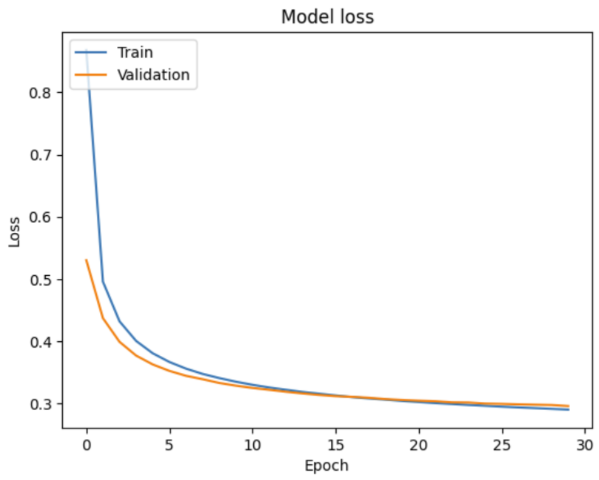


Figure 5: Training and validation loss over epochs.

Key points:

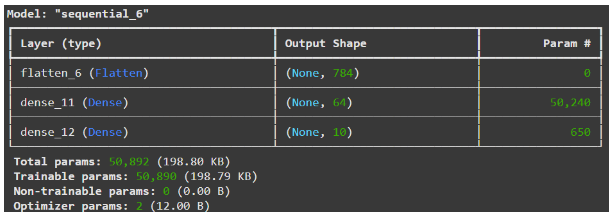
- The training loss and validation loss both decrease over time.
- The validation loss closely follows the training loss, indicating that the model is generalizing well.
- There is no sign of overfitting, as both losses stabilize and converge towards the same value.

4.2 2 Hidden Layer

To enhance performance, a hidden layer with 64 neurons and ReLU activation is added:

- Input layer: Flattened 28x28 grayscale image (784 values)
- Hidden layer: 64 neurons (ReLU activation)
- Output layer: 10 neurons (softmax activation)

The model is retrained, and its accuracy is compared to the previous architecture. The architecture is summarized using `model.summary()` to display the number of trainable parameters.



```
Model: "sequential_6"
┌─────────────────────────────┬──────────────────────┬─────────────┐
│ Layer (type)                │ Output Shape         │     Param # │
├─────────────────────────────┼──────────────────────┼─────────────┤
│ flatten_6 (Flatten)         │ (None, 784)          │           0 │
├─────────────────────────────┼──────────────────────┼─────────────┤
│ dense_11 (Dense)            │ (None, 64)           │      50,240 │
├─────────────────────────────┼──────────────────────┼─────────────┤
│ dense_12 (Dense)            │ (None, 10)           │         650 │
└─────────────────────────────┴──────────────────────┴─────────────┘
 Total params: 50,892 (198.80 KB)
 Trainable params: 50,890 (198.79 KB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 2 (12.00 B)
```

Figure 6: Model's accuracy

after the whole previous process the result obtained

Loss and Accuracy Evolution

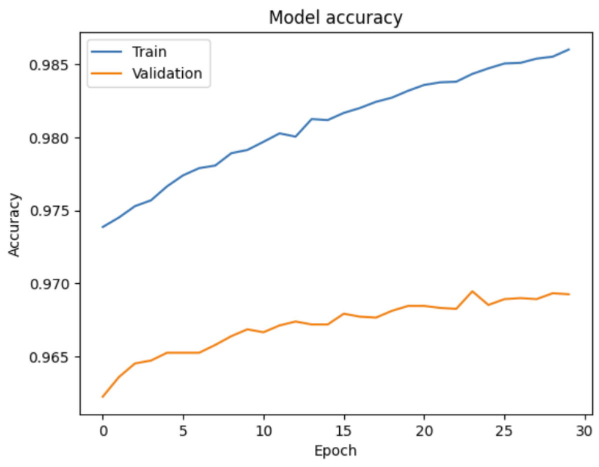


Figure 7: Training and validation accuracy trends.

Key observations:

- Training accuracy steadily improves from about 0.974 to 0.986.
- Validation accuracy increases from about 0.962 to 0.969.

- The widening gap between training and validation accuracy (especially after epoch 15) confirms the overfitting observed in the loss plot.

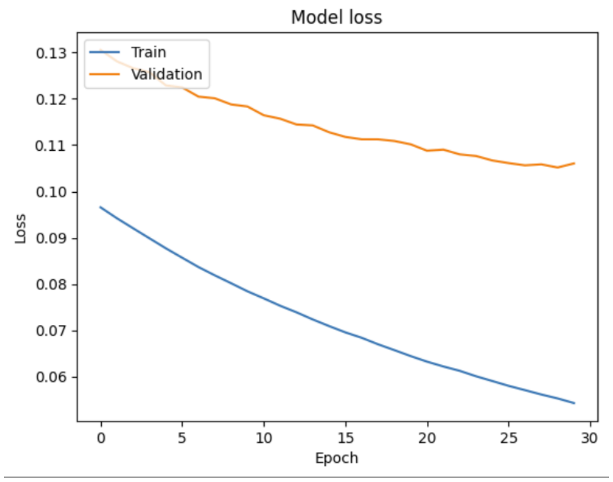


Figure 8: Training and validation loss trends.

Key observations:

- Both training and validation loss decrease consistently over time, which is positive.
- Training loss starts around 0.097 and drops to about 0.055.
- Validation loss starts higher at about 0.122 and decreases to about 0.105.
- The persistent gap between training and validation loss indicates some overfitting.

Error Analysis

Some digits are misclassified by the model. These misclassified images, along with their predicted labels and probabilities, are visualized. Potential reasons for misclassification include:

- Poorly written digits resembling another class
- Insufficient training data for specific digit styles
- Overlapping feature representations between digits

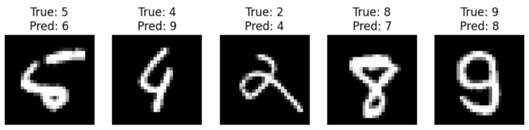


Figure 9: misclassified digits

5 Conclusion

Overall, these results(both for one hidden layer and 2 layers) demonstrate that the model is effectively learning from the data and achieving high accuracy in recognizing handwritten digits. The validation accuracy is a particularly important metric, as it indicates how well the model will perform on new, unseen data.