Handwritten Digit Recognition

Ananda Biswas

June 12, 2025

Contents

1 Objective

Develop a deep learning model using the **MNIST dataset of handwritten digits** to accurately classify digits (0–9) from 28×28 grayscale images.

This project aims to:

- **Design and train a neural network** to achieve high classification accuracy.
- **Evaluate the model** using metrics like accuracy, precision on the test set.
- **Visualize predictions** to interpret results and identify common misclassifications.

2 Necessary Libraries

[1]:
```
import numpy as np
import pandas as pd
import tensorflow as tf

from matplotlib import pyplot as plt

from matplotlib import image as img

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.activations import linear, sigmoid, relu

from tensorflow.keras.losses import SparseCategoricalCrossentropy

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.regularizers import l2

from tensorflow.nn import softmax

from sklearn.model_selection import train_test_split
```

3 Loading Data

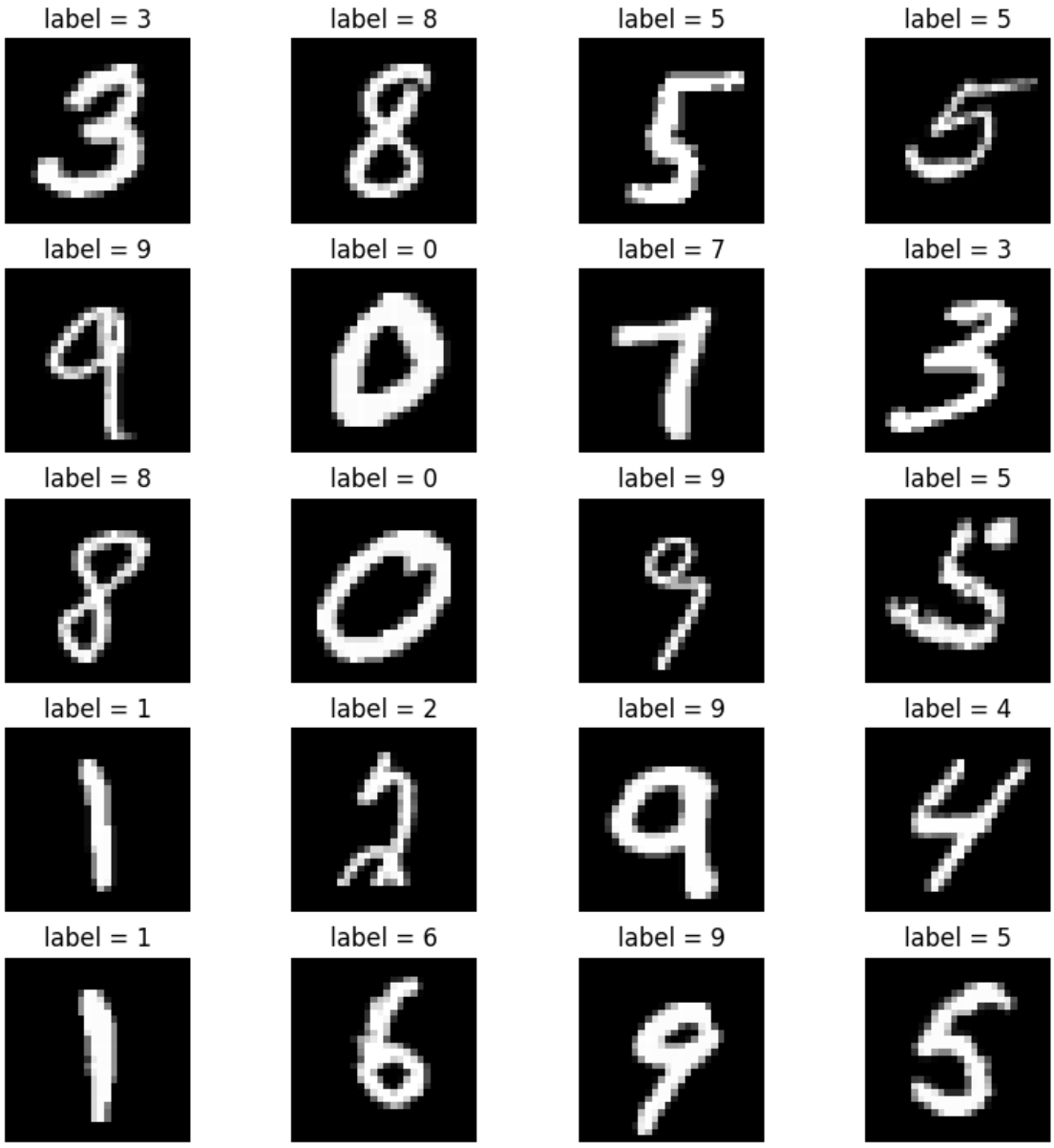
[2]:
```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
```

[3]:
```
print(f"Train images : {x_train.shape}")
print(f"Train labels : {y_train.shape}")

print(f"Test images : {x_test.shape}")
print(f"Test labels : {y_test.shape}")
```

```
Train images : (60000, 28, 28)
Train labels : (60000,)
Test images : (10000, 28, 28)
Test labels : (10000,)
```

4 Hovering over the data

[4]:
```
# visualizing any random 20 training example

fig, axes = plt.subplots(5, 4, figsize=(8,8))

fig.tight_layout(pad=0.15) #[left, bottom, right, top]

m = x_train.shape[0]

for i,ax in enumerate(axes.flat) :

    random_index = np.random.randint(m)

    an_example = x_train[random_index]

    ax.imshow(an_example, cmap = 'gray')

    ax.set_title(f"label = {y_train[random_index]}")

    ax.set_axis_off()

plt.show()
```



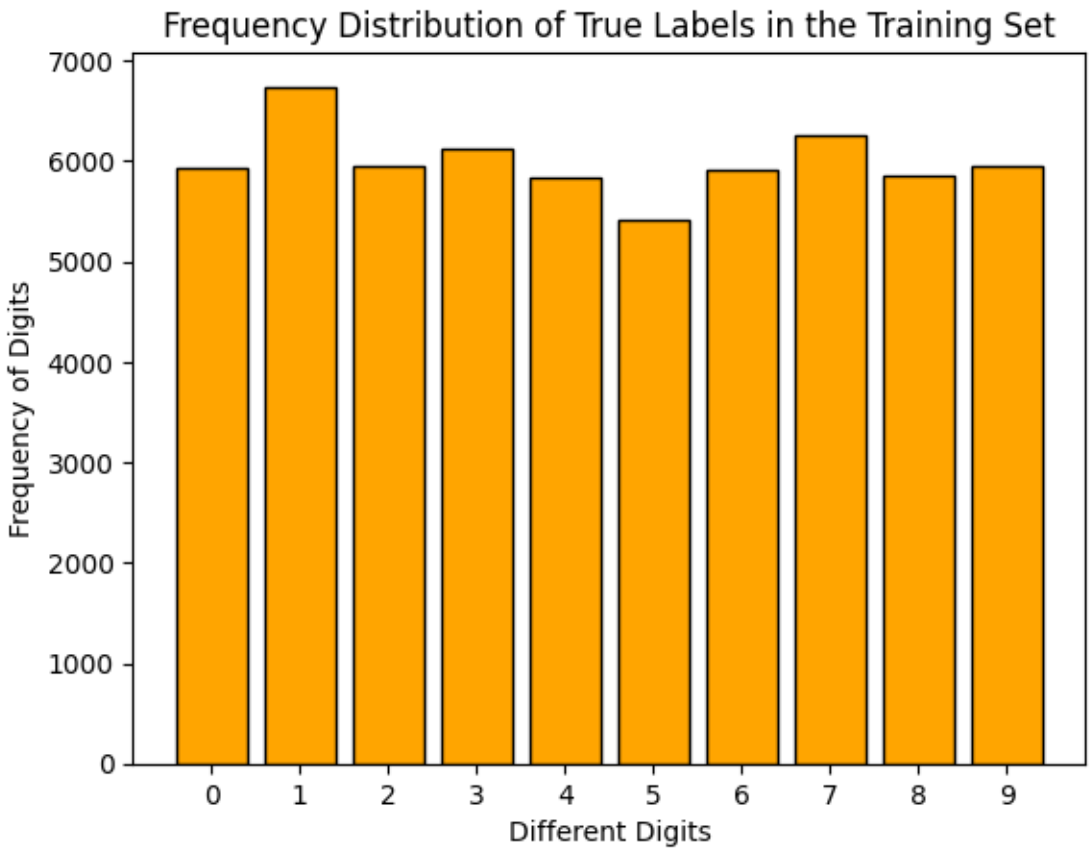
[5]:
```
# we shall also have a look how many times the digits 0-9 are present in the␣
 ↪training data

digits, frequency = np.unique(y_train, return_counts=True)

dictionary_1 = {
    'label' : digits,
    'frequency in the training data' : frequency
}
```

```
dataframe_1 = pd.DataFrame(dictionary_1)

dataframe_1
```

[5]:

	label	frequency in the training data
0	0	5923
1	1	6742
2	2	5958
3	3	6131
4	4	5842
5	5	5421
6	6	5918
7	7	6265
8	8	5851
9	9	5949

[6]:
```
# also a barplot of the training data

plt.bar(digits, frequency, color = 'orange', edgecolor = 'black')

plt.title("Frequency Distribution of True Labels in the Training Set")
plt.xlabel("Different Digits")
plt.ylabel("Frequency of Digits")
plt.xticks(digits, [x for x in range(10)])

plt.show()
```



5 Building a Neural Network Model

Our model will have **7 layers**, consisting of 1 input layer, 5 hidden layers and 1 output layer. We shall also employ **L2 regularization**.

The **input layer** will have **784 neurones.**

1st hidden layer will have **120 neurones** with **relu** as the activation function.

2nd hidden layer will have **90 neurones** with **relu** as the activation function.

3rd hidden layer will have **80 neurones** with **relu** as the activation function.

4th hidden layer will have **50 neurones** with **relu** as the activation function.

5th hidden layer will have **30 neurones** with **relu** as the activation function.

The **output layer** will have **10 neurones** with **softmax** function as the activation function.

[7]:
```
model_1 = Sequential(
    [
        tf.keras.Input(shape=(784,)),
        Dense(units=120, activation='relu', kernel_regularizer=l2(0.01)),
```

```
        Dense(units=90, activation='relu', kernel_regularizer=l2(0.01)),
        Dense(units=80, activation='relu', kernel_regularizer=l2(0.01)),
        Dense(units=50, activation='relu', kernel_regularizer=l2(0.01)),
        Dense(units=30, activation='relu', kernel_regularizer=l2(0.01)),
        Dense(units=10, activation='softmax')
    ])
```

6 Model Fitting

[8]:
```
model_1.compile(optimizer = Adam(learning_rate = 0.001),
                loss = SparseCategoricalCrossentropy())

model_1.fit(x_train.reshape(60000, 784), y_train, epochs=50)
```

```
Epoch 1/50
1875/1875 [==============================] - 14s 6ms/step - loss: 2.8314
Epoch 2/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.8056
Epoch 3/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.4625
Epoch 4/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.3971
Epoch 5/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.3764
Epoch 6/50
1875/1875 [==============================] - 10s 5ms/step - loss: 0.3585
Epoch 7/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.3436
Epoch 8/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.3418
Epoch 9/50
1875/1875 [==============================] - 11s 6ms/step - loss: 0.3300
Epoch 10/50
1875/1875 [==============================] - 11s 6ms/step - loss: 0.3259
Epoch 11/50
1875/1875 [==============================] - 10s 5ms/step - loss: 0.3265
Epoch 12/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.3190
Epoch 13/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.3135
Epoch 14/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.3133
Epoch 15/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.3075
Epoch 16/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.3096
Epoch 17/50
```

```
1875/1875 [==============================] - 9s 5ms/step - loss: 0.3086
Epoch 18/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.3017
Epoch 19/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.3030
Epoch 20/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.3012
Epoch 21/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.3011
Epoch 22/50
1875/1875 [==============================] - 10s 5ms/step - loss: 0.3015
Epoch 23/50
1875/1875 [==============================] - 10s 5ms/step - loss: 0.2960
Epoch 24/50
1875/1875 [==============================] - 10s 5ms/step - loss: 0.2955
Epoch 25/50
1875/1875 [==============================] - 10s 5ms/step - loss: 0.2977
Epoch 26/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2902
Epoch 27/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2960
Epoch 28/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2931
Epoch 29/50
1875/1875 [==============================] - 10s 5ms/step - loss: 0.2903
Epoch 30/50
1875/1875 [==============================] - 10s 5ms/step - loss: 0.2894
Epoch 31/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2888
Epoch 32/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2889
Epoch 33/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2913
Epoch 34/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2911
Epoch 35/50
1875/1875 [==============================] - 10s 5ms/step - loss: 0.2866
Epoch 36/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2898
Epoch 37/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2902
Epoch 38/50
1875/1875 [==============================] - 10s 5ms/step - loss: 0.2846
Epoch 39/50
1875/1875 [==============================] - 10s 6ms/step - loss: 0.2854
Epoch 40/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2872
Epoch 41/50
```

```
1875/1875 [==============================] - 11s 6ms/step - loss: 0.2854
Epoch 42/50
1875/1875 [==============================] - 10s 5ms/step - loss: 0.2847
Epoch 43/50
1875/1875 [==============================] - 10s 5ms/step - loss: 0.2854
Epoch 44/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2828
Epoch 45/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2870
Epoch 46/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2818
Epoch 47/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2820
Epoch 48/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2834
Epoch 49/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2814
Epoch 50/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2839
```

[8]: `<keras.callbacks.History at 0x1f1dd716990>`

7 Accuracy on Test Set

[9]:
```python
def accuracy(yhat, true) :

    p = len(true)

    correct = 0

    for i in range(p) :
        if (yhat[i] == true[i]) :
            correct += 1

    percentage = (correct / p) * 100

    return percentage
```

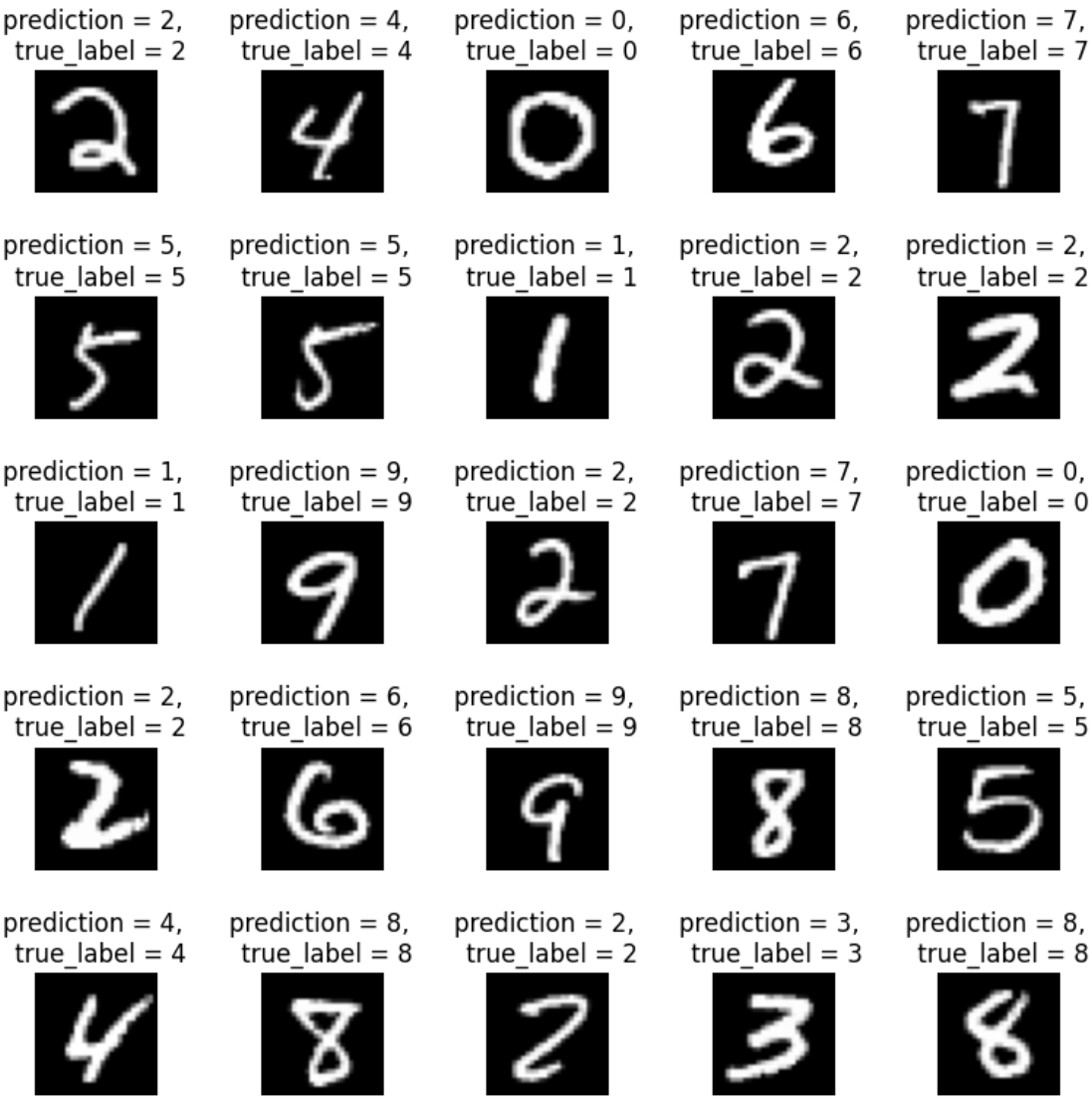
[10]:
```python
a = x_test.shape[0]

output_on_test_set = model_1.predict(x_test.reshape(10000, 784))

yhat = np.zeros(a)

for i in range(a) :
    yhat[i] = np.argmax(output_on_test_set[i])
```

```
percentage = accuracy(yhat, y_test)

print(f"Accuracy of the model on test set = {percentage}%")
```

```
313/313 [==============================] - 1s 3ms/step
Accuracy of the model on test set = 96.05%
```

8 Predictions

[11]:
```
fig, axes = plt.subplots(5, 5, figsize = (8, 8))

plt.tight_layout(pad=2)

for i, ax in enumerate(axes.flat) :

    random_index = np.random.randint(x_test.shape[0])

    model_output = model_1.predict(x_test[random_index].reshape(1, 784),␣
 ↪verbose=0) # verbose = 0 is for omitting the progress

    yhat = np.argmax(model_output)

    ax.imshow(x_test[random_index], cmap = 'gray')

    ax.axis('off')

    ax.set_title(f"prediction = {yhat}, \n true_label = {y_test[random_index]}")

plt.show()
```


prediction = 2, true_label = 2
prediction = 4, true_label = 4
prediction = 0, true_label = 0
prediction = 6, true_label = 6
prediction = 7, true_label = 7
prediction = 5, true_label = 5
prediction = 5, true_label = 5
prediction = 1, true_label = 1
prediction = 2, true_label = 2
prediction = 2, true_label = 2
prediction = 1, true_label = 1
prediction = 9, true_label = 9
prediction = 2, true_label = 2
prediction = 7, true_label = 7
prediction = 0, true_label = 0
prediction = 2, true_label = 2
prediction = 6, true_label = 6
prediction = 9, true_label = 9
prediction = 8, true_label = 8
prediction = 5, true_label = 5
prediction = 4, true_label = 4
prediction = 8, true_label = 8
prediction = 2, true_label = 2
prediction = 3, true_label = 3
prediction = 8, true_label = 8