

## EDGE INTELLIGENCE LAB ASSIGNMENT 2

SWETHA K

25MML0008

### Task 1:

Analysing MNIST dataset applying Artificial Neural Network, saving the model and finding the weight of the model

The screenshot shows a Jupyter Notebook interface with several code cells and output sections. The notebook title is "25MML0008\_EdgeIntelligence.ipynb".

- Jar 1: Data - MNIST DATASET**

```
import tensorflow as tf
from tensorflow.keras import layers, models
import numpy as np
```

mnist = tf.keras.datasets.mnist
(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

Downloading data from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz>
11490434/11490434 0s 0us/step
- Jar 2: Task - Classification of handwritten digits**

```
x_train, x_test = x_train / 255.0, x_test / 255.0
```
- Jar 3: Model - Simple Feed forward Neural Network**

(one flatten layer to convert the input to 1D, Dense layer to learn the features, Dropout to minimize overfitting and final Dense layer for output)

```
model = models.Sequential([
    layers.Flatten(input_shape=(28, 28)),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(10, activation='softmax')
])
```
- Jar 4: Loss - Sparse Categorical Crossentropy**
- Jar 5: Learning - ADAM**

```
model.compile(optimizer='adam',
              loss= 'sparse_categorical_crossentropy',
              metrics=['accuracy'])
```
- Jar 6: Accuracy - Achieved accuracy of 97.78%**

```
print("training the neural network...")
model.fit(x_train, y_train, epochs=5)

Training the neural network...
Epoch 1/5
1875/1875 9s 4ms/step - accuracy: 0.8579 - loss: 0.4913
Epoch 2/5
1875/1875 9s 4ms/step - accuracy: 0.9534 - loss: 0.1548
Epoch 3/5
1875/1875 8s 4ms/step - accuracy: 0.9662 - loss: 0.1099
Epoch 4/5
1875/1875 9s 5ms/step - accuracy: 0.9723 - loss: 0.0880
Epoch 5/5
1875/1875 7s 4ms/step - accuracy: 0.9769 - loss: 0.0737
<keras.src.callbacks.History at 0xb1ab0e2d50>
```
- ```
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print(f'\nTest accuracy: {test_acc*100:.2f}%')

313/313 - 1s - 2ms/step - accuracy: 0.9778 - loss: 0.0737

Test accuracy: 97.78%
```

```

[1] predictions = model.predict(x_test)
print(f"Prediction for first test image: {np.argmax(predictions[0])}")
print(f"Actual label: {y_test[0]}")

313/313  is 2ms/step
Prediction for first test image: 7
Actual label: 7

Saving the model

[1] import pickle
weights = model.get_weights()
with open('mnist_weights.pkl', 'wb') as f:
    pickle.dump(weights, f)

```

### Weight of the model

```

[1] import os

# Get the size in bytes
file_size_bytes = os.path.getsize('mnist_weights.pkl')

# Convert bytes to KB
file_size_kb = file_size_bytes / 1024

print(f"File Size: {file_size_kb:.2f} KB")

File Size: 397.80 KB

Load the model

[1] with open('mnist_weights.pkl', 'rb') as f:
    loaded_weights = pickle.load(f)
    model.set_weights(loaded_weights)

```

### Using the pickled model to get the accuracy from test data

```

[1] # Load MNIST test data (ensure it's normalized to 0-1)
_, _, (x_test, y_test) = tf.keras.datasets.mnist.load_data()
x_test = x_test / 255.0

# Evaluate
loss, accuracy = model.evaluate(x_test, y_test, verbose=0)

print(f"Test Loss: {loss:.4f}")
print(f"Test Accuracy: {accuracy * 100:.2f}%")

Test Loss: 0.0737
Test Accuracy: 97.78%

[1] model.summary()

```

```

[1] model.summary()

Model: "sequential_1"

```

| Layer (type)        | Output Shape | Param # |
|---------------------|--------------|---------|
| flatten_1 (Flatten) | (None, 784)  | 0       |
| dense_2 (Dense)     | (None, 128)  | 100,400 |
| dropout_1 (Dropout) | (None, 128)  | 0       |
| dense_3 (Dense)     | (None, 10)   | 1,200   |

```

Total params: 100,312 (1.16 MB)
Trainable params: 101,770 (397.54 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 203,542 (795.09 KB)

```

Findings: Since the model is using ANN (Simple Neural Network) the number of parameters of the model are lesser compared to a CNN model, thus the weight of the model is 397.54KB.

## Task 2:

1. Creating an account on Edge Impulse
2. Data acquisition
3. Choosing data connect -> Scan QR through phone
4. Selecting labels and uploading dataset
5. Splitting dataset into train and test.

The screenshot shows the Edge Impulse Studio interface. On the left, there's a sidebar with navigation links: Dashboard, Devices, Data acquisition, Experiments, EON Tuner, Impulse design (with Create Impulse and Live classification options), and an Upgrade Plan section. The main area is titled "Dataset" and shows a summary: "DATA COLLECTED 27 items" with a pie chart icon, and "TRAIN / TEST SPLIT 81% / 19%" with a donut chart icon. Below this, there's a table titled "Dataset" with columns for SAMPLE NAME, LABEL, and ADDED. The table lists several entries, including "Chair.6ebbaaos" (Chair, Yesterday, 17...), "Chair.6ebbau3l" (Chair, Yesterday, 17...), "Chair.6ebbag96" (Chair, Yesterday, 17...), "Chair.6ebb9qm2" (Chair, Yesterday, 17...), "Chair.6ebb7fvp" (Monitor, Yesterday, 17...), and "Keyboard.6ebb5dm1" (Keyboard, Yesterday, 17...). To the right, there are sections for "Collect data" (with a note to "Connect a device") and "RAW DATA" (with a button to "Click on a sample to load..."). At the top right, it says "Target: Cortex-M4F 80MHz".