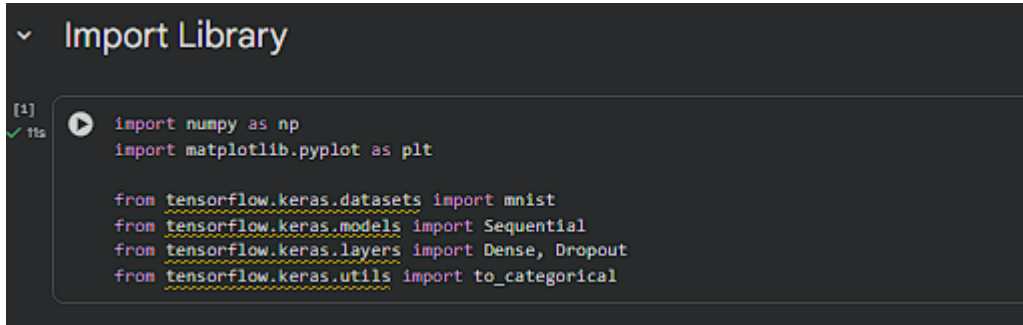


Nama: Noer Muhammad Ayub

NIM: 01102220142

Tugas: Praktikum 13

## 1. import Library



```
[1] import numpy as np
import matplotlib.pyplot as plt

from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.utils import to_categorical
```

Kode tersebut digunakan untuk menyiapkan seluruh pustaka yang diperlukan dalam proses pembuatan dan pelatihan model *neural network* untuk klasifikasi data citra. Library NumPy diimpor untuk melakukan pengolahan data numerik dan manipulasi array, sedangkan Matplotlib digunakan untuk keperluan visualisasi data, seperti menampilkan contoh gambar atau grafik hasil pelatihan model.

Dataset MNIST diimpor dari modul `tensorflow.keras.datasets` sebagai sumber data utama yang berisi citra angka tulisan tangan. Selanjutnya, kelas `Sequential` dari Keras digunakan untuk membangun arsitektur model jaringan saraf secara berurutan, di mana setiap layer ditambahkan satu per satu. Layer `Dense` berfungsi sebagai lapisan *fully connected* yang memproses fitur input, sementara `Dropout` digunakan untuk mengurangi risiko *overfitting* dengan menonaktifkan sebagian neuron secara acak selama proses pelatihan.

Terakhir, fungsi `to_categorical` diimpor untuk mengubah label kelas numerik menjadi format *one-hot encoding*, yang diperlukan dalam proses pelatihan model klasifikasi dengan lebih dari dua kelas. Secara keseluruhan, kode ini merupakan tahap awal persiapan sebelum dilakukan pemuatan data, *preprocessing*, perancangan model, dan pelatihan jaringan saraf.

## 2. Load Dataset



```
[2] (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
```

Baris kode tersebut digunakan untuk memuat dataset MNIST dan sekaligus membaginya menjadi data latih dan data uji. Fungsi `mnist.load_data()` secara otomatis mengambil dataset MNIST yang berisi citra angka tulisan tangan berukuran  $28 \times 28$ .

piksel. Hasil pemanggilan fungsi ini berupa dua pasangan data, yaitu data pelatihan dan data pengujian.

Variabel `X_train` dan `y_train` menyimpan data citra dan label yang digunakan untuk melatih model, sedangkan `X_test` dan `y_test` menyimpan data citra dan label yang digunakan untuk menguji performa model setelah pelatihan selesai. Pemisahan ini penting agar evaluasi model dapat dilakukan secara objektif dan tidak bias terhadap data yang sudah dipelajari sebelumnya.

### 3. Visualisasi Data



Kode tersebut digunakan untuk **menampilkan satu contoh citra dari dataset MNIST** beserta labelnya. Fungsi `plt.imshow(X_train[0], cmap='gray')` menampilkan gambar pertama pada data pelatihan dalam bentuk citra dua dimensi dengan skala warna abu-abu, sesuai dengan karakteristik citra MNIST. Baris `plt.title(f'Label: {y_train[0]}')` menambahkan judul pada gambar yang menunjukkan label angka sebenarnya dari citra tersebut. Selanjutnya, `plt.axis('off')` digunakan untuk menyembunyikan sumbu koordinat agar tampilan gambar lebih bersih dan fokus pada bentuk angka yang ditampilkan.

### 4. Preprocessing Data

```
Preprocessing Data

[4]
✔ Os
X_train = X_train.reshape(60000, 784)
X_test = X_test.reshape(10000, 784)

[5]
✔ Os
#Normalisasi Data
X_train = X_train / 255.0
X_test = X_test / 255.0

[5]
# One-Hot Encoding Label
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
```

Bagian **preprocessing data** ini bertujuan menyiapkan data MNIST agar dapat diproses oleh model neural network. Data citra yang semula berbentuk matriks  $28 \times 28$  diubah menjadi vektor satu dimensi berukuran 784 melalui proses *reshape*, karena layer *Dense* hanya menerima input berbentuk vektor. Selanjutnya, nilai piksel dinormalisasi dengan membagi 255 agar berada pada rentang 0–1 sehingga proses pelatihan lebih stabil. Terakhir, label diubah ke format *one-hot encoding* untuk mendukung klasifikasi multikelas (0–9).

## 5. Pembuatan Model

```
Pembuatan Model Multi-Layer Perceptron (MLP)

101 model = Sequential([
102     Dense(128, activation='relu', input_shape=(784,)),
103     Dropout(0.2),
104     Dense(64, activation='relu'),
105     Dense(10, activation='softmax')
106 ])

-- /usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an 'input_shape' / 'input_dim' argument to a layer. When using Sequential models, prefer using an 'Input(shape)' object as the first layer in the model instead.
super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Kode tersebut digunakan untuk membangun arsitektur model neural network dengan pendekatan *Sequential*, di mana layer disusun secara berurutan. Layer pertama *Dense(128, activation='relu', input\_shape=(784,))* berfungsi sebagai *hidden layer* yang menerima input berupa vektor berukuran 784 dan menggunakan fungsi aktivasi ReLU untuk menangkap pola nonlinier pada data. Layer *Dropout(0.2)* ditambahkan untuk mengurangi *overfitting* dengan menonaktifkan 20% neuron secara acak selama proses pelatihan. Selanjutnya, layer *Dense(64, activation='relu')* berperan sebagai *hidden layer* tambahan untuk memperdalam representasi fitur. Terakhir, layer *Dense(10, activation='softmax')* digunakan sebagai *output layer* yang menghasilkan probabilitas untuk 10 kelas digit MNIST.

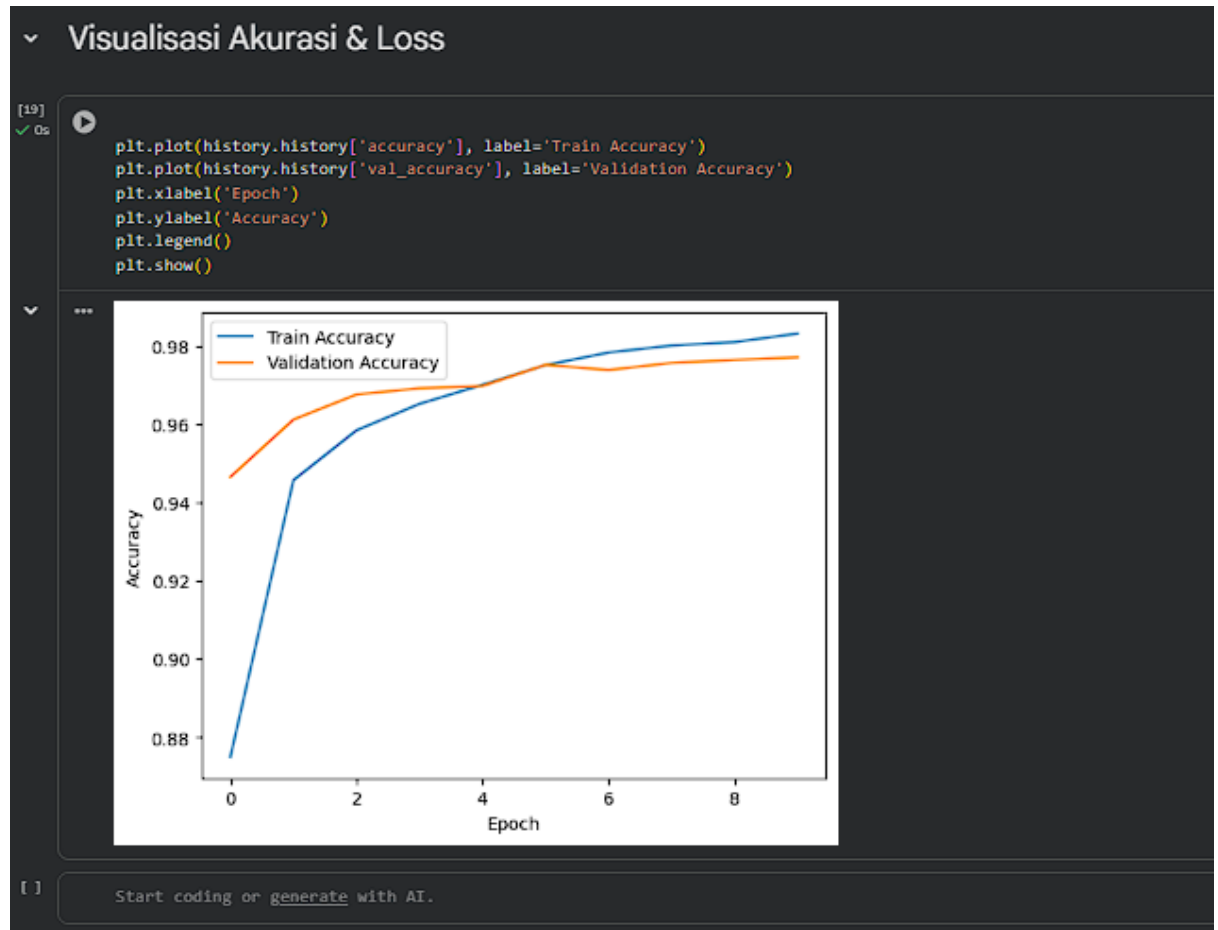
## 6. Compile Model

```
179 model.compile(
180     optimizer='adam',
181     loss=categorical_crossentropy,
182     metrics=['accuracy'])
183
184 0
185
186 #Pembuatan Model Multi-Layer Perceptron (MLP)
187 model = Sequential([
188     Dense(128, activation='relu', input_shape=(784,)),
189     Dropout(0.2),
190     Dense(64, activation='relu'),
191     Dense(10, activation='softmax')
192 ])
193
194 #Compile Model
195 model.compile(
196     optimizer='adam',
197     loss=categorical_crossentropy,
198     metrics=['accuracy'])
199
200 #Training Model
201 history = model.fit(
202     x_train,
203     y_train,
204     epochs=10,
205     batch_size=128,
206     validation_split=0.2
207 )
208
209 -- /usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an 'input_shape' / 'input_dim' argument to a layer. When using Sequential models, prefer using an 'Input(shape)' object as the first layer in the model instead.
210 super().__init__(activity_regularizer=activity_regularizer, **kwargs)
211
212 Epoch 1/10: 3s/10ms/step - accuracy: 0.7729 - loss: 0.7558 - val_accuracy: 0.9465 - val_loss: 0.1816
213 100%|#####| 10/10 [10s]
214 Epoch 2/10: 3s/10ms/step - accuracy: 0.9419 - loss: 0.1983 - val_accuracy: 0.9632 - val_loss: 0.1341
215 100%|#####| 10/10 [10s]
216 Epoch 3/10: 2s/7ms/step - accuracy: 0.9678 - loss: 0.1423 - val_accuracy: 0.9676 - val_loss: 0.1093
217 100%|#####| 10/10 [10s]
218 Epoch 4/10: 3s/7ms/step - accuracy: 0.9647 - loss: 0.1137 - val_accuracy: 0.9692 - val_loss: 0.1015
219 100%|#####| 10/10 [10s]
220 Epoch 5/10: 4s/10ms/step - accuracy: 0.9783 - loss: 0.0957 - val_accuracy: 0.9687 - val_loss: 0.1010
221 100%|#####| 10/10 [10s]
222 Epoch 6/10: 3s/7ms/step - accuracy: 0.9756 - loss: 0.0816 - val_accuracy: 0.9752 - val_loss: 0.0861
223 100%|#####| 10/10 [10s]
224 Epoch 7/10: 3s/7ms/step - accuracy: 0.9787 - loss: 0.0694 - val_accuracy: 0.9738 - val_loss: 0.0883
225 100%|#####| 10/10 [10s]
226 Epoch 8/10: 3s/7ms/step - accuracy: 0.9797 - loss: 0.0641 - val_accuracy: 0.9757 - val_loss: 0.0810
227 100%|#####| 10/10 [10s]
228 Epoch 9/10: 3s/8ms/step - accuracy: 0.9815 - loss: 0.0543 - val_accuracy: 0.9764 - val_loss: 0.0812
229 100%|#####| 10/10 [10s]
230 Epoch 10/10: 3s/7ms/step - accuracy: 0.9837 - loss: 0.0483 - val_accuracy: 0.9771 - val_loss: 0.0806
231 100%|#####| 10/10 [10s]
232
233 #Evaluate Model
234 # Ensure y_test is one-hot encoded for evaluation
235 y_test = to_categorical(y_test, 10)
236 test_loss, test_accuracy = model.evaluate(x_test, y_test)
237 print("Test Accuracy:", test_accuracy)
238
239 100%|#####| 10/10 [10s]
240 Test Accuracy: 0.975200077703884
```

Bagian compile model digunakan untuk mengonfigurasi model sebelum proses pelatihan dimulai. Pada kode `model.compile()`, optimizer Adam dipilih untuk mengatur proses pembaruan bobot karena mampu beradaptasi secara efisien terhadap laju pembelajaran. Fungsi `loss categorical_crossentropy` digunakan karena model melakukan klasifikasi multikelas dengan label *one-hot encoding*. Sementara itu, metrik

accuracy ditambahkan untuk mengukur tingkat ketepatan prediksi model selama proses training dan evaluasi.

## 7. Visualisasi Akuran dan Loss



Kode tersebut digunakan untuk memvisualisasikan performa model selama proses pelatihan. Nilai `history.history['accuracy']` diplot untuk menunjukkan perubahan akurasi data latih pada setiap epoch, sedangkan `history.history['val_accuracy']` menampilkan akurasi data validasi. Sumbu X diberi label *Epoch* yang merepresentasikan jumlah iterasi pelatihan, dan sumbu Y menunjukkan nilai *Accuracy*. Legenda ditambahkan agar perbedaan antara akurasi pelatihan dan validasi mudah dibedakan, lalu grafik ditampilkan menggunakan `plt.show()`.