# Tugas 13 Praktikum Mandiri

**SYAHRI GHIFARI MAULIDI 0110222217**

[1] Teknik Informatika, STT Terpadu Nurul Fikri, Depok

## 1. Import Library dan Membaca Dataset

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.utils import to_categorical
from sklearn.metrics import confusion_matrix
```

```
train_df = pd.read_csv('/content/sample_data/mnist_train_small.csv')
test_df  = pd.read_csv('/content/sample_data/mnist_test.csv')
```

Pertama, dilakukan proses import library yang akan digunakan sepanjang eksperimen.Library seperti NumPy digunakan untuk pengolahan data numerik, sedangkan Matplotlib dan Seaborn digunakan untuk visualisasi hasil model, seperti grafik dan confusion matrix. Selain itu, library dari TensorFlow Keras digunakan untuk membangun dan melatih model jaringan saraf MLP, sementara Scikit-learn dimanfaatkan untuk menghitung confusion matrix sebagai bagian dari evaluasi performa klasifikasi multi-kelas.

### 1.1 **Preprocessing Data**

```
[10]
✓ 0s
(X_train, y_train), (X_test, y_test) = mnist.load_data()

print(X_train.shape)
print(X_test.shape)

(60000, 28, 28)
(10000, 28, 28)
```

Merupakan tahap penting karena di sinilah data dipersiapkan sebelum digunakan oleh model. Langkah pertama adalah reshape data, yaitu mengubah citra dari bentuk 28×28 menjadi vektor satu dimensi berukuran 784. Langkah ini dilakukan karena MLP hanya dapat menerima input berupa vektor, bukan matriks dua dimensi. Selanjutnya, dilakukan normalisasi data dengan membagi nilai piksel dengan 255. Proses ini membuat nilai piksel berada pada rentang 0 sampai 1, sehingga proses training menjadi lebih stabil dan cepat. Terakhir, label kelas diubah menjadi format one-hot encoding. Teknik ini memungkinkan model memahami bahwa klasifikasi yang dilakukan adalah klasifikasi multi-kelas dengan 10 kategori digit.

### 1.2 Arsitektur Model MLP dan Compile Model

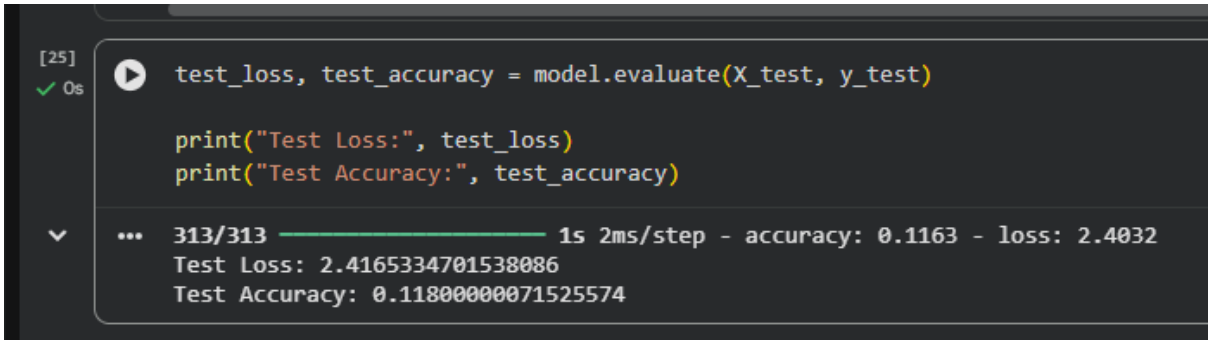
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical

(X_train, y_train), (X_test, y_test) = mnist.load_data()

X_train = X_train.reshape(60000, 784).astype('float32') / 255.0
X_test  = X_test.reshape(10000, 784).astype('float32') / 255.0

y_train = to_categorical(y_train, 10)
y_test  = to_categorical(y_test, 10)


model = Sequential()
model.add(Dense(128, activation='relu', input_shape=(784,)))
model.add(Dense(64, activation='relu'))
model.add(Dense(10, activation='softmax'))

model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

test_loss, test_accuracy = model.evaluate(X_test, y_test)
print("Test Accuracy:", test_accuracy)
```

```
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/dense.py:93: U
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.1163 - loss: 2.4032
Test Accuracy: 0.11800000071525574
```

Berdasarkan potongan kode dan hasil eksekusi yang ditampilkan, model Multilayer Perceptron (MLP) dibangun untuk melakukan klasifikasi multi-kelas pada dataset MNIST yang terdiri dari 10 kelas angka tulisan tangan. Dataset MNIST terlebih dahulu dimuat dan dilakukan preprocessing dengan mengubah citra berukuran 28×28 piksel menjadi vektor satu dimensi

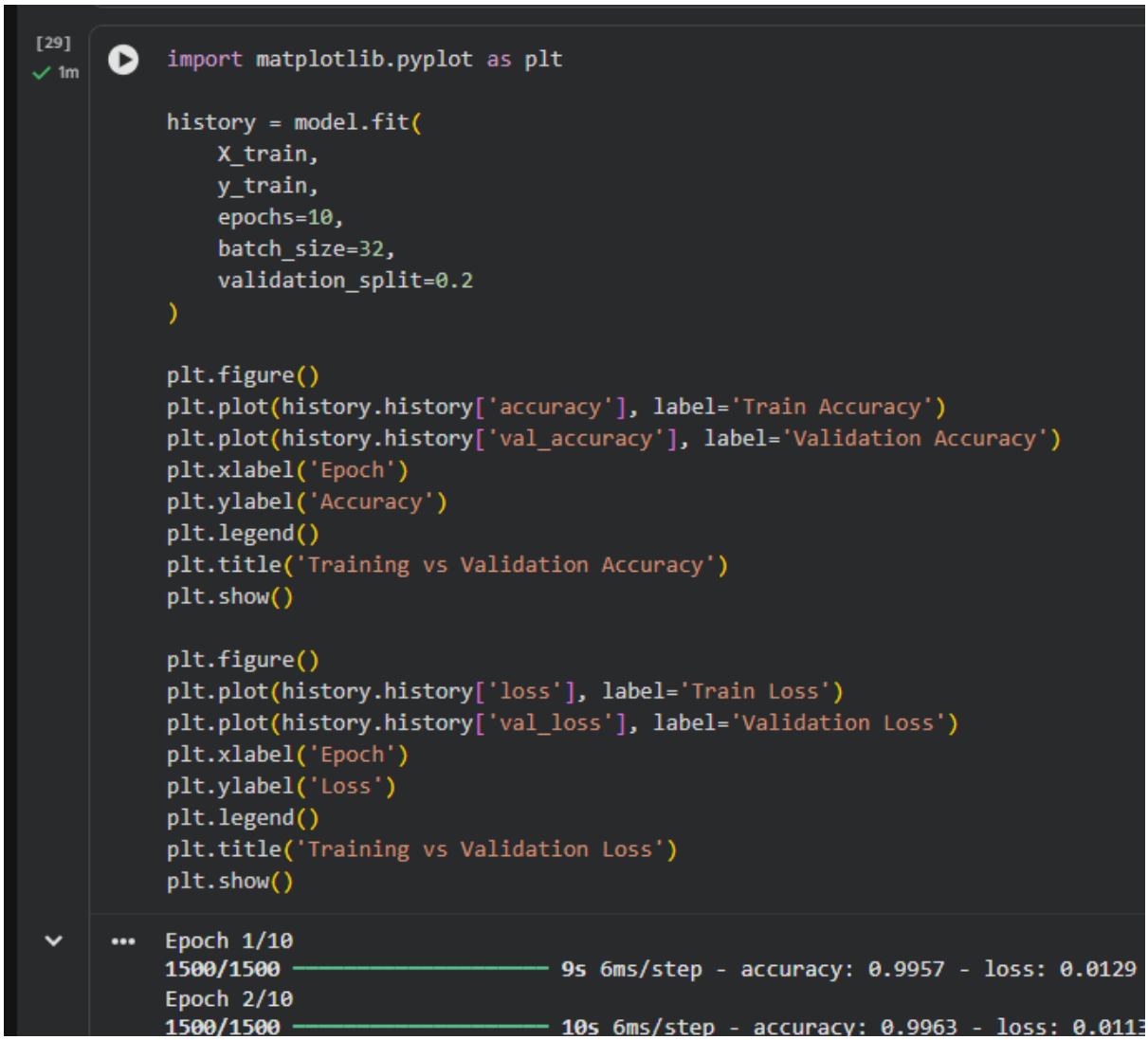
sepanjang 784 serta melakukan normalisasi nilai piksel ke rentang 0 sampai 1. Selain itu, label kelas diubah ke dalam bentuk one-hot encoding agar sesuai dengan skema klasifikasi multi-kelas yang digunakan oleh model. Model MLP yang digunakan terdiri dari dua hidden layer dengan masing-masing 128 dan 64 neuron yang menggunakan fungsi aktivasi ReLU, serta satu output layer dengan 10 neuron dan fungsi aktivasi Softmax. Arsitektur ini sebenarnya sudah cukup representatif untuk menangkap pola dasar pada data MNIST. Model kemudian dikompilasi menggunakan optimizer Adam dan fungsi loss categorical crossentropy, dengan metrik evaluasi berupa akurasi, yang merupakan konfigurasi umum dan sesuai untuk permasalahan klasifikasi multi-kelas.

### 2.1 Evaluasi Model




digunakan untuk mengevaluasi model menggunakan data testing yang belum pernah dilihat selama training. Hasil evaluasi berupa nilai loss dan accuracy menunjukkan performa akhir model. Akurasi yang tinggi menandakan bahwa model mampu mengklasifikasikan digit dengan benar, sementara nilai loss yang rendah menunjukkan kesalahan prediksi yang relatif kecil. Tahap ini penting untuk menilai kemampuan generalisasi model terhadap data baru.

## 2.2 Training Model dan Visualisasi Akurasi dan Loss



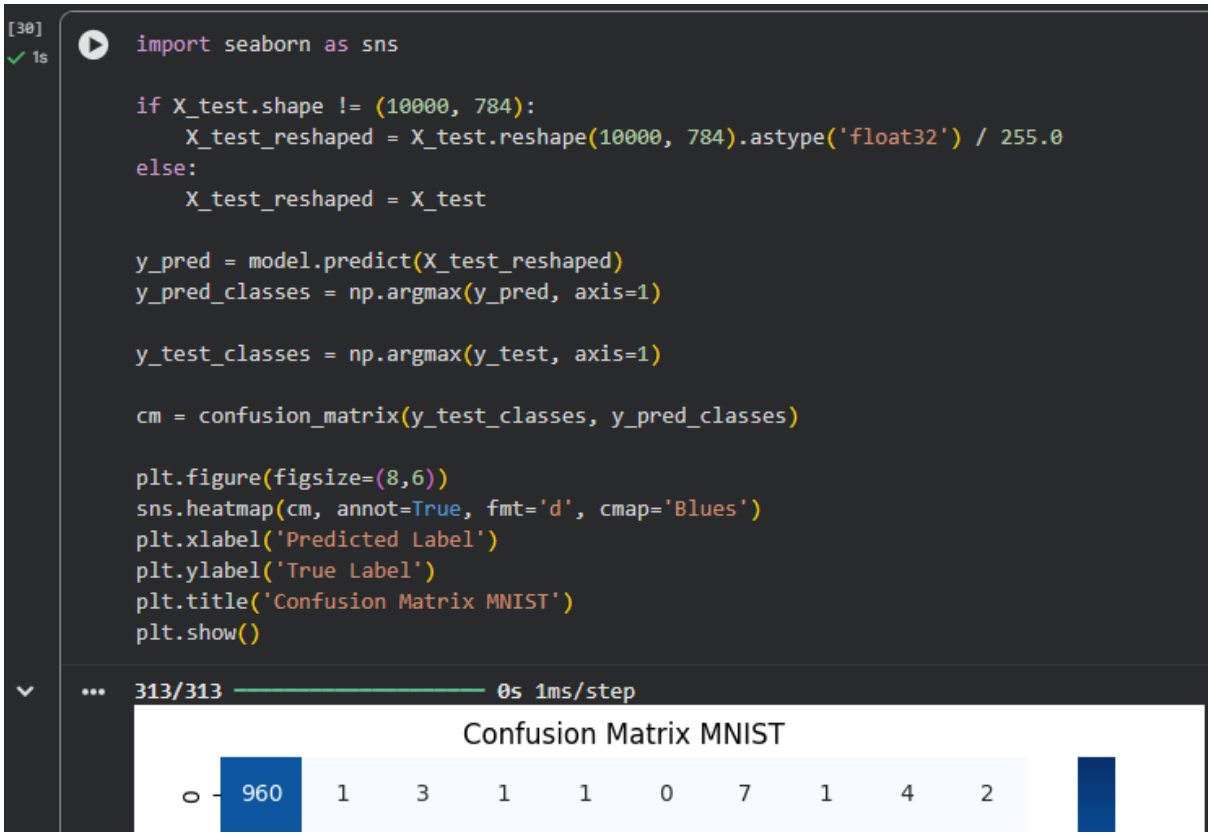
```
[29]
✓ 1m
import matplotlib.pyplot as plt

history = model.fit(
    X_train,
    y_train,
    epochs=10,
    batch_size=32,
    validation_split=0.2
)

plt.figure()
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Training vs Validation Accuracy')
plt.show()

plt.figure()
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.title('Training vs Validation Loss')
plt.show()

...  Epoch 1/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 9s 6ms/step - accuracy: 0.9957 - loss: 0.0129
Epoch 2/10
1500/1500 ━━━━━━━━━━━━━━━━━━━━ 10s 6ms/step - accuracy: 0.9963 - loss: 0.011
```

Proses training model dimulai. Model dilatih menggunakan data training selama beberapa epoch, dengan sebagian data digunakan sebagai validation set. Selama training, model secara bertahap memperbarui bobot jaringan saraf untuk meminimalkan nilai loss dan meningkatkan akurasi. Output dari cell ini memperlihatkan perkembangan nilai accuracy dan loss pada setiap epoch, yang menjadi indikator seberapa baik model belajar dari data.

Hasil training divisualisasikan dalam bentuk grafik. Grafik akurasi menunjukkan perbandingan antara akurasi training dan validation pada setiap epoch. Jika kedua grafik meningkat dan tidak berjauhan, hal ini menandakan bahwa model belajar dengan baik tanpa mengalami overfitting. Grafik loss

menunjukkan penurunan nilai kesalahan seiring bertambahnya epoch, yang mengindikasikan proses pembelajaran berjalan secara stabil dan efektif. Visualisasi ini membantu manusia memahami perilaku model secara lebih intuitif dibandingkan hanya melihat angka.

### 2.3 Confusion Matrix




Menganalisis performa model secara lebih mendalam menggunakan confusion matrix. Confusion matrix memperlihatkan hubungan antara label sebenarnya dan label hasil prediksi. Prediksi yang benar akan terlihat pada diagonal utama, sementara kesalahan prediksi berada di luar diagonal. Dari confusion matrix, dapat dilihat bahwa sebagian besar digit berhasil diklasifikasikan dengan benar. Namun, masih terdapat beberapa kesalahan pada digit yang memiliki bentuk mirip, seperti 4 dengan 9 atau 3 dengan 5. Hal ini menunjukkan keterbatasan MLP dalam menangkap informasi spasial citra.