Implementasi Artificial Neural Network Menggunakan Multi-Layer Perceptron untuk Klasifikasi Angka Tulisan Tangan pada Dataset MNIST

Nama : Muhamad Aditia
Nim : 0110224213
Link Github : https://github.com/muhammadaditia433/MachineLeraning
Email : 0110224213.student.nurulfikri.ac.id

Abstract.

Artificial Neural Network (ANN) merupakan salah satu metode Machine Learning yang terinspirasi dari cara kerja jaringan saraf manusia. Pada praktikum ini dilakukan implementasi ANN dengan arsitektur Multi-Layer Perceptron (MLP) untuk melakukan klasifikasi angka tulisan tangan menggunakan dataset MNIST. Dataset MNIST berisi citra grayscale berukuran 28×28 piksel dengan label angka 0 sampai 9 sehingga termasuk dalam permasalahan klasifikasi multi-kelas. Proses yang dilakukan meliputi preprocessing data, pembangunan model MLP, pelatihan model, serta evaluasi performa menggunakan data uji. Hasil percobaan menunjukkan bahwa model MLP mampu mengenali pola angka tulisan tangan dengan tingkat akurasi yang cukup baik, sehingga ANN efektif digunakan untuk klasifikasi data citra.

Pendahuluan

Machine Learning merupakan salah satu cabang dari kecerdasan buatan yang memungkinkan sistem komputer belajar dari data tanpa harus diprogram secara eksplisit. Salah satu metode yang sering digunakan adalah Artificial Neural Network (ANN), yang meniru cara kerja jaringan saraf biologis.

Pada praktikum sebelumnya, ANN umumnya diterapkan pada dataset tabular seperti Titanic. Oleh karena itu, pada praktikum ini digunakan dataset MNIST yang berbentuk citra untuk memperluas pemahaman mengenai penerapan ANN pada jenis data yang berbeda. Dataset MNIST terdiri dari gambar angka tulisan tangan sehingga membutuhkan proses preprocessing khusus sebelum dapat digunakan dalam pelatihan model.

Landasan Teori

1. Artificial Neural Network (ANN)

Artificial Neural Network adalah model komputasi yang terdiri dari neuron-neuron buatan yang saling terhubung. ANN mampu mempelajari pola kompleks melalui proses pelatihan menggunakan data.

2. Multi-Layer Perceptron (MLP)

Multi-Layer Perceptron merupakan jenis ANN yang terdiri dari input layer, satu atau lebih hidden layer, dan output layer. MLP sering digunakan untuk masalah klasifikasi dan regresi.

3. Dataset MNIST

MNIST adalah dataset standar dalam bidang Machine Learning yang berisi 70.000 gambar angka tulisan tangan (0–9). Dataset ini sering digunakan untuk menguji performa algoritma klasifikasi pada data citra.

Metodologi Penelitian

Tahapan yang dilakukan dalam praktikum ini adalah sebagai berikut:

1. Mengambil dataset MNIST menggunakan library keras.
2. Melakukan preprocessing data berupa normalisasi dan encoding label.
3. Membangun model multi-layer perceptron
4. Melatih model menggunakan data training
5. Melakukan evaluasi performa model menggunakan data testing

Penjelasan Output Kodenya

1. Import Library

```
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,
Flatten from tensorflow.keras.utils import
to_categorical
```

Pada bagian awal program, dilakukan pemanggilan beberapa library yang dibutuhkan. Library NumPy digunakan untuk membantu proses pengolahan data dalam bentuk array. Matplotlib digunakan untuk menampilkan data, seperti contoh gambar dari dataset MNIST dan grafik hasil pelatihan model.

Dataset MNIST diambil langsung melalui Keras sehingga tidak perlu mengunduh data secara manual. Model neural network dibuat menggunakan arsitektur Sequential, yang menyusun layer secara berurutan. Layer Flatten digunakan untuk mengubah data gambar menjadi bentuk satu dimensi, sedangkan layer Dense digunakan sebagai layer utama pada model Multi-Layer Perceptron. Selain itu, fungsi to_categorical digunakan untuk mengubah label angka menjadi bentuk one-hot encoding agar dapat diproses oleh model klasifikasi. 2. Pengambilan Dataset MNIST

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()

print("Shape data training:", x_train.shape)
print("Shape data testing :", x_test.shape)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ──────────────────── 1s 0us/step
Shape data training: (60000, 28, 28)
Shape data testing : (10000, 28, 28)
```

Pada bagian ini, dataset MNIST dimuat menggunakan fungsi mnist.load_data(). Fungsi tersebut secara otomatis mengambil dan membagi dataset menjadi dua bagian, yaitu data training dan data testing. Data training digunakan untuk melatih model, sedangkan data testing digunakan untuk menguji performa model setelah proses pelatihan selesai.

Perintah print digunakan untuk menampilkan ukuran (shape) dari masing-masing data. Hasil yang ditampilkan menunjukkan bahwa data training berjumlah 60.000 gambar dan data testing berjumlah 10.000 gambar. Setiap gambar memiliki ukuran 28×28 piksel, yang merepresentasikan citra angka tulisan tangan.

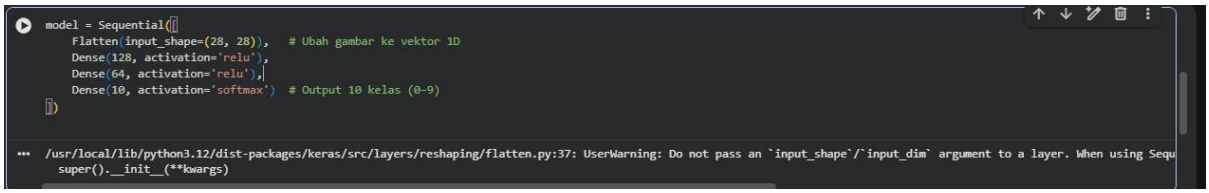
3. Preprocessing Data

```
# Normalisasi (0-255 → 0-1)
x_train = x_train / 255.0
x_test = x_test / 255.0

# One-hot encoding label
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
```

Sebelum data digunakan untuk melatih model, dilakukan proses preprocessing terlebih dahulu. Nilai piksel pada gambar MNIST awalnya berada pada rentang 0 sampai 255. Oleh karena itu, dilakukan normalisasi dengan cara membagi seluruh nilai piksel dengan 255 agar berada pada rentang 0 sampai 1. Proses ini bertujuan agar model lebih mudah dalam melakukan perhitungan dan proses pelatihan menjadi lebih stabil.

Selanjutnya, label data yang berupa angka 0 sampai 9 diubah ke dalam bentuk one-hot encoding. Perubahan ini dilakukan karena model yang digunakan merupakan klasifikasi multi-kelas, sehingga label perlu direpresentasikan dalam bentuk vektor agar dapat diproses dengan baik oleh neural network

4.Pembuatan Model Neural Network

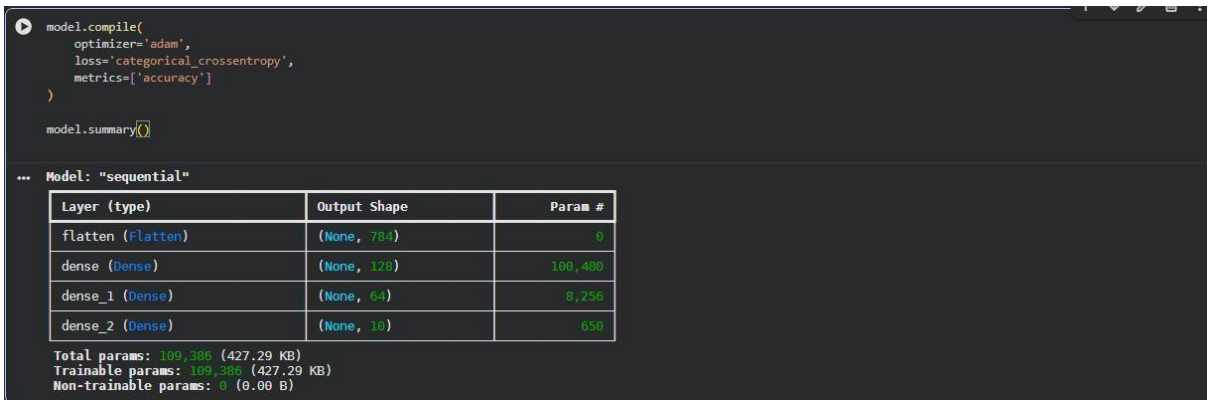


Pada tahap ini dibuat model Artificial Neural Network menggunakan arsitektur Multi-Layer Perceptron (MLP) dengan pendekatan Sequential. Layer pertama yang digunakan adalah Flatten, yang berfungsi untuk mengubah data gambar berukuran 28×28 piksel menjadi vektor satu dimensi agar dapat diproses oleh layer berikutnya.

Setelah itu ditambahkan dua hidden layer bertipe Dense. Hidden layer pertama terdiri dari 128 neuron dan hidden layer kedua terdiri dari 64 neuron. Keduanya menggunakan fungsi aktivasi ReLU, yang membantu model dalam mempelajari pola non-linear pada data serta mempercepat proses pelatihan.

Pada bagian akhir, digunakan output layer dengan 10 neuron yang merepresentasikan kelas angka 0 sampai 9. Fungsi aktivasi Softmax digunakan pada output layer agar model menghasilkan probabilitas untuk setiap kelas, sehingga kelas dengan nilai probabilitas tertinggi dapat dipilih sebagai hasil prediksi.

5.Proses Compile Model

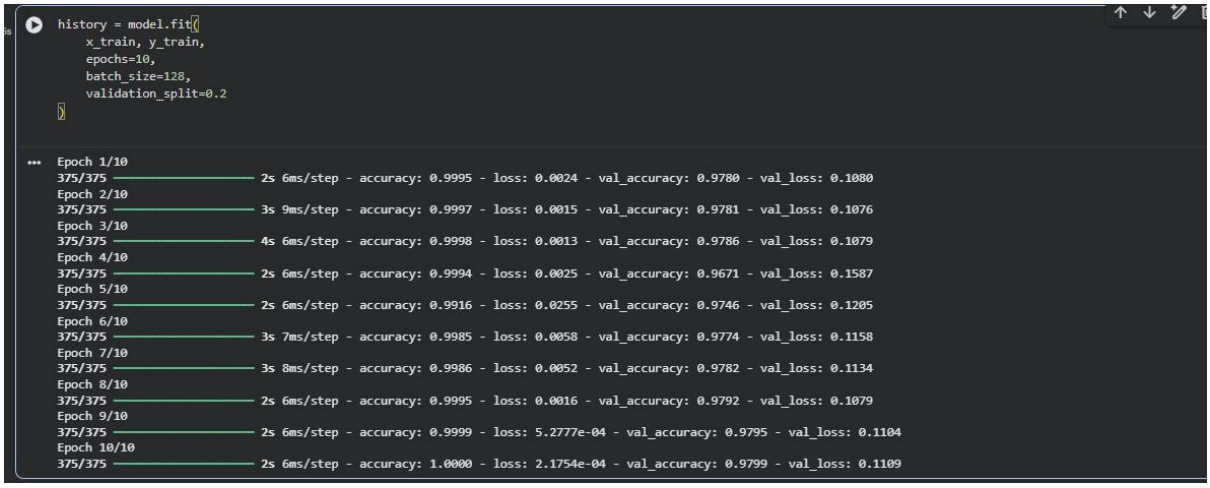


Setelah model neural network dibuat, langkah selanjutnya adalah melakukan proses compile model. Pada tahap ini ditentukan metode optimasi, fungsi loss, dan metrik evaluasi yang akan digunakan selama proses pelatihan.

Optimizer Adam dipilih karena mampu menyesuaikan learning rate secara otomatis sehingga proses pelatihan menjadi lebih cepat dan stabil. Fungsi loss yang digunakan adalah categorical crossentropy, karena permasalahan yang dihadapi merupakan klasifikasi multikelas dengan label dalam bentuk one-hot encoding. Untuk mengevaluasi performa model, digunakan metrik accuracy agar dapat diketahui tingkat ketepatan prediksi model.

Perintah model.summary() digunakan untuk menampilkan ringkasan arsitektur model, termasuk jumlah layer, jumlah neuron pada setiap layer, serta total parameter yang digunakan.

6.Proses Training
Model

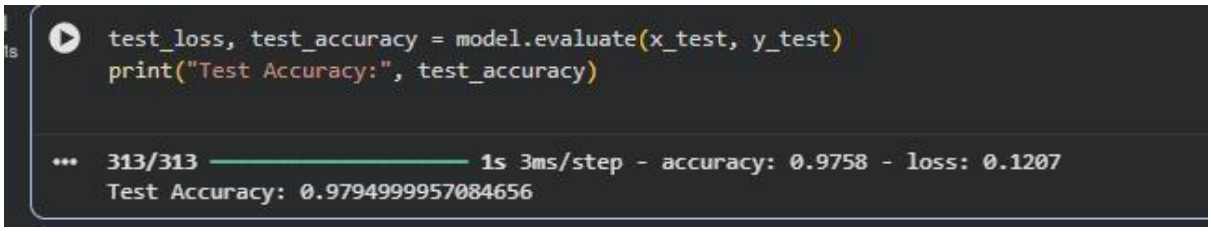


```
history = model.fit(
    x_train, y_train,
    epochs=10,
    batch_size=128,
    validation_split=0.2
)

Epoch 1/10
375/375 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.9995 - loss: 0.0024 - val_accuracy: 0.9780 - val_loss: 0.1080
Epoch 2/10
375/375 ━━━━━━━━━━━━━━━━━━━━ 3s 9ms/step - accuracy: 0.9997 - loss: 0.0015 - val_accuracy: 0.9781 - val_loss: 0.1076
Epoch 3/10
375/375 ━━━━━━━━━━━━━━━━━━━━ 4s 6ms/step - accuracy: 0.9998 - loss: 0.0013 - val_accuracy: 0.9786 - val_loss: 0.1079
Epoch 4/10
375/375 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.9994 - loss: 0.0025 - val_accuracy: 0.9671 - val_loss: 0.1587
Epoch 5/10
375/375 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.9916 - loss: 0.0255 - val_accuracy: 0.9746 - val_loss: 0.1205
Epoch 6/10
375/375 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - accuracy: 0.9985 - loss: 0.0058 - val_accuracy: 0.9774 - val_loss: 0.1158
Epoch 7/10
375/375 ━━━━━━━━━━━━━━━━━━━━ 3s 8ms/step - accuracy: 0.9986 - loss: 0.0052 - val_accuracy: 0.9782 - val_loss: 0.1134
Epoch 8/10
375/375 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.9995 - loss: 0.0016 - val_accuracy: 0.9792 - val_loss: 0.1079
Epoch 9/10
375/375 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.9999 - loss: 5.2777e-04 - val_accuracy: 0.9795 - val_loss: 0.1104
Epoch 10/10
375/375 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 1.0000 - loss: 2.1754e-04 - val_accuracy: 0.9799 - val_loss: 0.1109
```

Pada tahap ini dilakukan proses pelatihan model menggunakan data training. Model dilatih selama 10 epoch, di mana satu epoch menunjukkan satu kali proses pembelajaran pada seluruh data training. Jumlah batch size yang digunakan adalah 128, yang berarti data training dibagi menjadi beberapa batch kecil agar proses pelatihan lebih efisien.

Selain itu, digunakan validation split sebesar 20%, sehingga sebagian data training dipisahkan sebagai data validasi. Data validasi ini digunakan untuk memantau performa model selama proses pelatihan dan membantu melihat apakah model mengalami overfitting atau tidak. Hasil pelatihan, seperti nilai loss dan accuracy pada data training dan validasi, disimpan dalam variabel history.

7.Evaluasi
Model



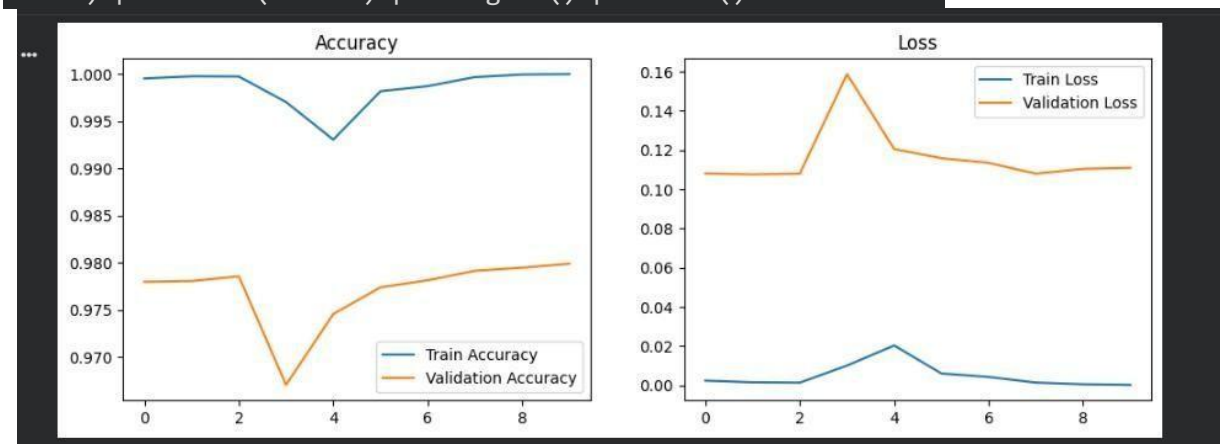
```
test_loss, test_accuracy = model.evaluate(x_test, y_test)
print("Test Accuracy:", test_accuracy)

313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9758 - loss: 0.1207
Test Accuracy: 0.9794999957084656
```

Setelah proses pelatihan selesai, model dievaluasi menggunakan data testing yang belum pernah dilihat sebelumnya oleh model. Evaluasi ini dilakukan untuk mengetahui seberapa baik model dalam melakukan prediksi pada data baru.

Fungsi model.evaluate() digunakan untuk menghitung nilai loss dan accuracy pada data testing. Nilai accuracy kemudian ditampilkan untuk menunjukkan tingkat ketepatan model dalam mengklasifikasikan angka tulisan tangan pada dataset MNIST.

8.Visualisasi Hasil Training

```
plt.figure(figsize=(12,4))
# Accuracy plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label='Train
Accuracy') plt.plot(history.history['val_accuracy'],
label='Validation Accuracy') plt.title('Accuracy')
plt.legend() # Loss plt.subplot(1,2,2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation
Loss') plt.title('Loss') plt.legend() plt.show()
```

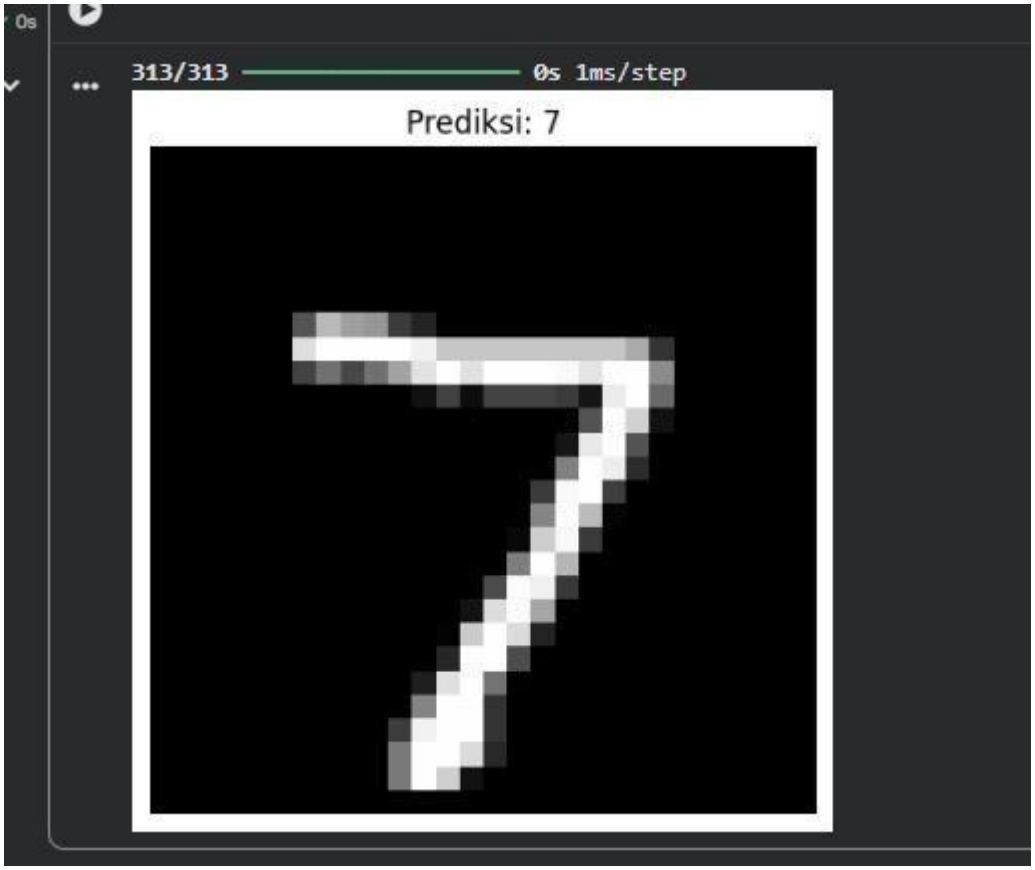


Pada bagian ini dilakukan visualisasi hasil pelatihan model menggunakan grafik. Grafik pertama menampilkan akurasi model pada data training dan data validasi di setiap epoch. Dari grafik ini dapat dilihat bagaimana kemampuan model dalam mempelajari data serta kestabilan performa selama proses pelatihan.

Grafik kedua menampilkan nilai loss pada data training dan validasi. Grafik loss digunakan untuk melihat apakah nilai kesalahan model semakin menurun seiring bertambahnya epoch. Dengan membandingkan grafik training dan validasi, dapat diketahui apakah model mengalami overfitting atau masih mampu melakukan generalisasi dengan baik.

9.Prediksi Model

```
pred = model.predict(x_test)
plt.imshow(x_test[0], cmap='gray')
plt.title(f"Prediksi: {np.argmax(pred[0])}")
plt.axis('off') plt.show()
```



Pada tahap ini dilakukan proses prediksi menggunakan model yang telah dilatih. Model digunakan untuk memprediksi label dari data testing. Hasil prediksi berupa nilai probabilitas untuk setiap kelas angka 0 sampai 9.

Selanjutnya ditampilkan salah satu contoh gambar dari data testing beserta hasil prediksinya. Angka dengan nilai probabilitas tertinggi dipilih sebagai hasil prediksi akhir. Visualisasi ini bertujuan untuk melihat secara langsung apakah model mampu mengenali angka tulisan tangan dengan benar.

Kesimpulan

Berdasarkan hasil praktikum yang telah dilakukan, dapat disimpulkan bahwa model Artificial Neural Network dengan arsitektur Multi-Layer Perceptron mampu melakukan klasifikasi angka tulisan tangan pada dataset MNIST dengan cukup baik. Proses preprocessing seperti normalisasi data dan one-hot encoding berpengaruh terhadap kestabilan dan performa model selama pelatihan.

Hasil evaluasi menunjukkan bahwa model dapat mengenali pola angka tulisan tangan dengan tingkat akurasi yang baik pada data testing. Selain itu, visualisasi grafik akurasi dan loss menunjukkan bahwa model mengalami peningkatan performa seiring bertambahnya epoch dan tidak menunjukkan gejala overfitting yang signifikan. Dengan demikian, penerapan

ANN menggunakan MLP dapat menjadi solusi yang efektif untuk permasalahan klasifikasi data citra sederhana seperti MNIST.

Referensi

1.LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). *Gradient-based learning applied to document recognition*. Proceedings of the IEEE.

2.Chollet, F. (2017). *Deep Learning with Python*. Manning Publications.

3.TensorFlow. (2024). *MNIST Dataset*. https://www.tensorflow.org/datasets/catalog/mnist

4.Kaggle. (2023). *Multi-Layer Perceptron – MNIST*. https://www.kaggle.com/code/fgiorgio/multi-layer-perceptron-mnist