LAPORAN IMPLEMENTASI PROGRAM ARTIFICIAL NEURAL NETWORK BACKPROPAGATION UNTUK MNIST HANDWRITING DATASET

disusun oleh :

Kelompok 7

1. Rafif Rabbani (G6401211018)
2. Farhan Nurohman (G6401211028)
3. Lutfiah Nursabiliyanti (G6401211041)
4. Lailatul Shakdiyah (G6401211050)
5. Muhammad Ghifar Azka N. (G6401211108)

DEPARTEMEN ILMU KOMPUTER
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
IPB UNIVERSITY

2024

PENDAHULUAN

Neural Network adalah metode komputasi menggunakan neuron yang diorganisasikan sebagai jaringan yang saling berhubungan untuk perhitungan non-linear dasar sehingga mirip dengan jaringan saraf manusia. Neural Network dibangun untuk memecahkan suatu permasalahan tertentu seperti pengenalan pola atau klasifikasi dengan proses pembelajaran/pelatihan. Metode pembelajaran/pelatihan terbagi menjadi tiga kelompok. Pertama, *supervised learning* yaitu metode pelatihan yang belajar dari dataset yang sudah memiliki label. Kedua, *unsupervised learning* yaitu metode pelatihan yang belajar dari dataset yang belum memiliki label. Ketiga, *hybrid learning* yaitu metode pelatihan yang menggabungkan elemen-elemen *supervised learning* dan *unsupervised learning.*

Backpropagation merupakan salah satu teknik pelatihan *supervised learning* yang paling banyak digunakan untuk pengenalan pola-pola yang kompleks. Arsitektur jaringan *backpropagation* terdiri dari tiga lapisan, lapisan masukan (*input layer*), lapisan tersembunyi (*hidden layer*) yang terdiri dari banyak lapisan (*multilayer*), dan lapisan keluaran(*output layer*). Tahap pelatihan (*training*) dilakukan dengan cara melakukan perubahan bobot. Jika pelatihan sudah selesai, dilakukan proses penyelesaian masalah dengan fase pengujian (*testing*).

METODE

Tahapan umumnya adalah sebagai berikut:

1. Load data MNIST dan memisahkan antara data train dan data testing.
2. Dari data train, pisahkan lagi jadi data train dan validation.
3. Melakukan metode min-max scaller pada setiap data.
4. Membuat model Neural Network menggunakan scikit learn.
5. Melihat akurasi model dan Confusion Matrix.
6. Membuat model Neural Network menggunakan Keras.
7. Melihat akurasi model dan Confusion Matrix.

Arsitektur Neural Network yang dibuat adalah sebagai berikut:

1. Jumlah node pada input layer: 64.
2. Jumlah layer pada hidden layer: 2.
3. Jumlah node pada hidden layer pertama: 16.
4. Jumlah node pada hidden layer kedua: 16.
5. Jumlah node pada output layer: 10.

PEMBAHASAN

Menggunakan Library ScikitLearn

Pertama-tama, data MNIST handwriting dimuat menggunakan fungsi load_digits() dari modul sklearn.datasets. Data digits berisi nilai-nilai *pixel* gambar 8x8 serta label yang menunjukkan gambar itu merupakan angka berapa. Terdapat 10 angka yang akan dipelajari (target), yakni angka 0-9. Data nilai *pixel* terdapat pada digits.data, sedangkan data angka

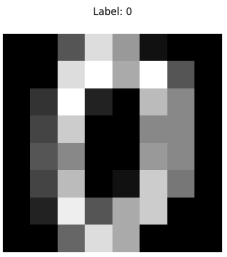
target terdapat pada digits.target. Sementara itu, digits.image menyimpan *value* yang sama dengan digits.data, hanya saja dalam bentuk yang dapat ditampilkan seperti pada output kode di bawah. Di sini, kita menampilkan sampel paling pertama yang kebetulan memiliki label "0".

```
from sklearn.datasets import load_digits
import matplotlib.pyplot as plt

digits = load_digits()
x = digits.data
y = digits.target

img_label = 0

plt.gray()
plt.matshow(digits.images[img_label])
plt.title('Label: {}'.format(y[img_label]))
plt.axis('off')
plt.show()
```



Berikutnya, kita akan membagi *dataset* menjadi data untuk *training*, validasi, dan *testing*. Pertama-tama kita bagi data menjadi 70% *training*, 30% *testing*. Berikutnya, dari data *training*, kita ambil lagi 30%-nya sebagai data validasi. Data validasi ini berguna supaya kita bisa memprediksi akurasi dari model dan melakukan *hyperparameter tuning* tanpa "menyentuh" data testing sebenarnya.

Pada *dataset* digits, terdapat 1797 jumlah sampel. Untuk variabel "x", ukuran datanya memiliki angka "64" yang mewakili jumlah *pixel* gambar 8x8.

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=.3)
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=.3)

print('x_train', x_train.shape)
print('x_val', x_val.shape)
print('x_test', x_test.shape)

print('y_train', y_train.shape)
print('y_val', y_val.shape)
print('y_test', y_test.shape)
```

```
x_train (1374, 64)
x_val (243, 64)
x_test (180, 64)
y_train (1374,)
y_val (243,)
y_test (180,)
```

Berikutnya, kita lakukan *training* terhadap data *training*. Jumlah *hidden layer* yang digunakan adalah 2 (16 node pada masing-masing *layer*), dan fungsi aktivasi yang digunakan adalah *relu*. Pada akhirnya, didapatkan akurasi saat *training* sebesar ~0.98.

```
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score

mlp = MLPClassifier(hidden_layer_sizes=(16,16, ), activation='relu', max_iter=1000,
epsilon=1e-08)

mlp.fit(x_train, y_train)
prediksi_val = mlp.predict(x_val)
acc_val = accuracy_score(y_val, prediksi_val)
print('Akurasi Validasi Training ANN:', acc_val)
```

```
Akurasi Validasi Training ANN: 0.9814814814814815
```

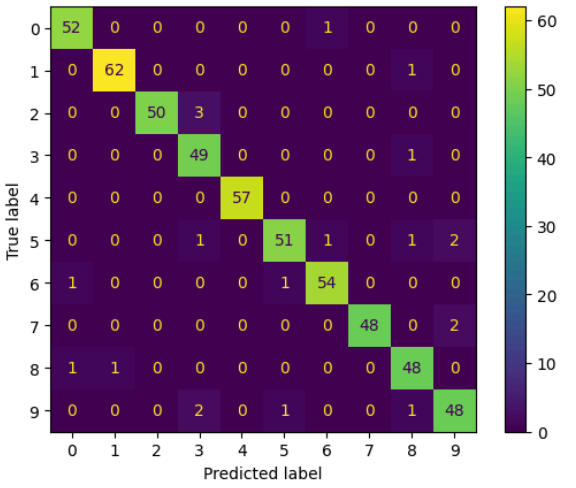
Pada tahap *testing*, kita mendapatkan akurasi sebesar ~0.96. Walau *error*-nya tergolong sedikit, melalui *confusion matrix* kita dapat melihat bahwa salah satu angka yang sulit dibedakan adalah “2” dan “3”, yang memiliki tiga *error*.

```
from sklearn.metrics import accuracy_score, confusion_matrix,
ConfusionMatrixDisplay

prediksi = mlp.predict(x_test)
accuracy = accuracy_score(y_test, prediksi)

conf_matrix = confusion_matrix(y_test, prediksi)
disp = ConfusionMatrixDisplay(conf_matrix)
disp.plot()
plt.show()
accuracy = accuracy_score(y_test, prediksi)
print('Akurasi Testing ANN:', accuracy)
```

```
Akurasi Testing ANN: 0.9611111111111111
```



Menggunakan Library Keras

Dataset yang akan digunakan sama persis dengan sebelumnya, yakni yang bersumber dari scikit-learn. Namun sebelum bisa membangun model ANN menggunakan Keras, kita

perlu mengubah kolom output (yang sebelumnya hanya satu, berisikan angka 0-9) menjadi 10, merepresentasikan One-Hot Encode. Sederhananya, yang tadinya "0" akan menjadi "1000000000", "1" akan menjadi "0100000000", seterusnya hingga "9" yang akan menjadi "0000000001".

```
from tensorflow.keras.utils import to_categorical
import numpy as np

y_train = to_categorical(y_train,10)
y_val = to_categorical(y_val,10)
y_test_2 = np.copy(y_test)
y_test = to_categorical(y_test,10)
```

Berikutnya, kita bangun model yang menggunakan *hyperparameter* sama seperti model sebelumnya.

```
from keras.models import Sequential
from keras.layers import Flatten, Dense

model = Sequential()
model.add(Flatten())
model.add(Dense(16,activation='relu'))
model.add(Dense(16,activation='relu'))
model.add(Dense(10,activation='softmax'))

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

Selanjutnya, kita lakukan *training* pada model.

```
model.fit(x_train,y_train,epochs=100,batch_size=50,validation_data=(x_val,y_val))
```

```
Epoch 1/100
18/18 [==============================] - 2s 19ms/step - loss: 4.6939 - acc: 0.0808 - val_loss: 2.9008 - val_acc: 0.0741
Epoch 2/100
18/18 [==============================] - 0s 6ms/step - loss: 2.6471 - acc: 0.0739 - val_loss: 2.3337 - val_acc: 0.1323
Epoch 3/100
18/18 [==============================] - 0s 6ms/step - loss: 2.2856 - acc: 0.1513 - val_loss: 2.1534 - val_acc: 0.2143
Epoch 4/100
18/18 [==============================] - 0s 6ms/step - loss: 2.1431 - acc: 0.1923 - val_loss: 2.0339 - val_acc: 0.2302
Epoch 5/100
18/18 [==============================] - 0s 7ms/step - loss: 2.0323 - acc: 0.2412 - val_loss: 1.9270 - val_acc: 0.2698
Epoch 6/100
18/18 [==============================] - 0s 5ms/step - loss: 1.9018 - acc: 0.3299 - val_loss: 1.8002 - val_acc: 0.3624
Epoch 7/100
18/18 [==============================] - 0s 6ms/step - loss: 1.7686 - acc: 0.4039 - val_loss: 1.6744 - val_acc: 0.4577
Epoch 8/100
18/18 [==============================] - 0s 5ms/step - loss: 1.6459 - acc: 0.4505 - val_loss: 1.5644 - val_acc: 0.4868
Epoch 9/100
18/18 [==============================] - 0s 5ms/step - loss: 1.5385 - acc: 0.4699 - val_loss: 1.4501 - val_acc: 0.5159
Epoch 10/100
18/18 [==============================] - 0s 5ms/step - loss: 1.4351 - acc: 0.5040 - val_loss: 1.3483 - val_acc: 0.5635
Epoch 11/100
18/18 [==============================] - 0s 5ms/step - loss: 1.3387 - acc: 0.5427 - val_loss: 1.2536 - val_acc: 0.5794
Epoch 12/100
18/18 [==============================] - 0s 5ms/step - loss: 1.2287 - acc: 0.5870 - val_loss: 1.1075 - val_acc: 0.6270
Epoch 13/100
18/18 [==============================] - 0s 6ms/step - loss: 1.1066 - acc: 0.6132 - val_loss: 1.0120 - val_acc: 0.6640
Epoch 14/100
```

Pada akhirnya, didapatkan akurasi sebesar ~0.93.

```
loss, accuracy = model.evaluate(x_test, y_test)
print('Akurasi Testing ANN:', accuracy)
```

```
17/17 [==============================] - 0s 2ms/step - loss: 0.2209 - acc: 0.9315
Akurasi Testing ANN: 0.9314814805984497
```

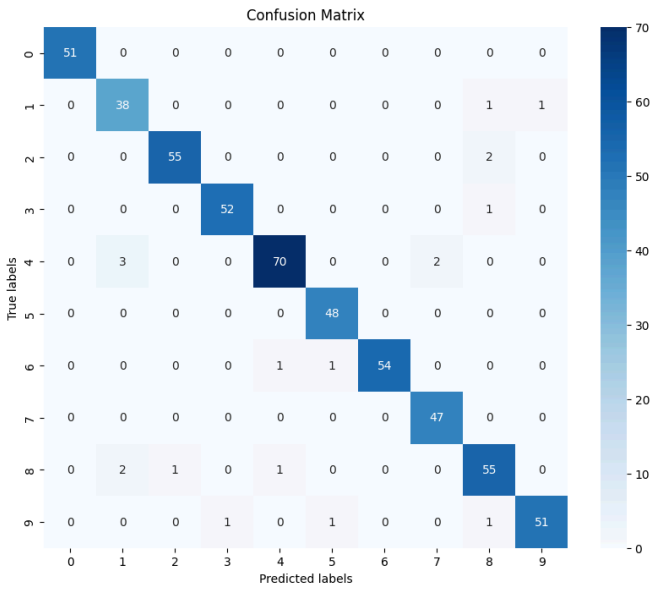
Apabila dibuat *confusion matrix*, hasilnya adalah sebagai berikut. Menurut matriks di bawah, salah satu angka yang sulit dibedakan adalah “1” dan “4” dengan tiga *error*.

```
import seaborn as sns

y_pred = model.predict(x_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_test_classes = np.argmax(y_test, axis=1)

conf_matrix = confusion_matrix(y_test_classes, y_pred_classes)

plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
xticklabels=np.arange(10), yticklabels=np.arange(10))
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()
```



KESIMPULAN

Telah dibuat dua model ANN untuk dataset MNIST Handwriting, satu menggunakan library scikit-learn dan satunya lagi menggunakan library Keras. Kedua model menunjukkan hasil yang cukup akurat sehingga dengan yakin dapat digunakan untuk memprediksi sampel-sampel *handwriting* lainnya. Model scikit-learn memiliki akurasi sebesar ~0,96 sedangkan model Keras memiliki akurasi sebesar ~0.93. Berdasarkan Confusion Matrix kedua model, beberapa angka yang termasuk sulit dibedakan adalah “2” dengan “3” dan “1” dengan “4”.

LAMPIRAN

Kode Program: mnist_kel7.ipynb