

Soal

- Membangun model **Convolutional Network** dengan *pre-processing* pada input data (**jittering**, **normalization**). Tambahkan juga **dropout regularization**, **batch normalization** dan **convolutional layer** hingga mencapai **90% accuracy** (dengan *epoch* berapapun) pada dataset MNIST. Waktu latih **dibawah 10 menit tanpa GPU**.
- Membangun model **AlexNet** hingga mencapai **80% accuracy** pada dataset MNIST. Waktu latih **dibawah 10 menit tanpa GPU**.

Jawaban

- **Convolutional Network**

```
import tensorflow as tf
from tensorflow import keras
import numpy as np
```

```
# Load MNIST dataset
(X_train, y_train), (X_test, y_test) = keras.datasets.mnist.load_data()
```

Memuat **dataset MNIST** yang terdiri dari 70.000 gambar angka tulisan tangan 1 sampai dengan 9 yang telah dipecah menjadi 60.000 data latih (X_train dan y_train) dan 10.000 data uji (X_test dan y_test), pemisahan antara data pelatihan dan data pengujian dilakukan secara otomatis oleh fungsi load_data().

```
# Reshape data to add channel dimension
X_train = np.reshape(X_train, (60000, 28, 28, 1))
X_test = np.reshape(X_test, (10000, 28, 28, 1))
```

```
# Jittering: Data augmentation by random cropping and flipping
data_augmentation = keras.Sequential([
    keras.layers.Resizing(28, 28),
    keras.layers.RandomCrop(28, 28),
    keras.layers.RandomFlip(mode='horizontal')
])
X_train = data_augmentation(X_train)
```

Melakukan **jittering** (augmentasi pada gambar) dalam kasus ini, terdapat tiga layer yaitu:

- Layer Resizing: melakukan resizing pada gambar dengan ukuran 28x28 piksel.
- Layer RandomCrop: melakukan cropping secara acak pada gambar dengan ukuran 28x28 piksel.
- Layer RandomFlip: melakukan flipping secara acak pada gambar dengan mode horizontal.

```
# Normalization: Scale pixel values from 0-255 to 0-1
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0
```

Melakukan **normalisasi** data pada dataset gambar, yaitu dengan mengubah rentang nilai pixel dari 0-255 menjadi 0-1.

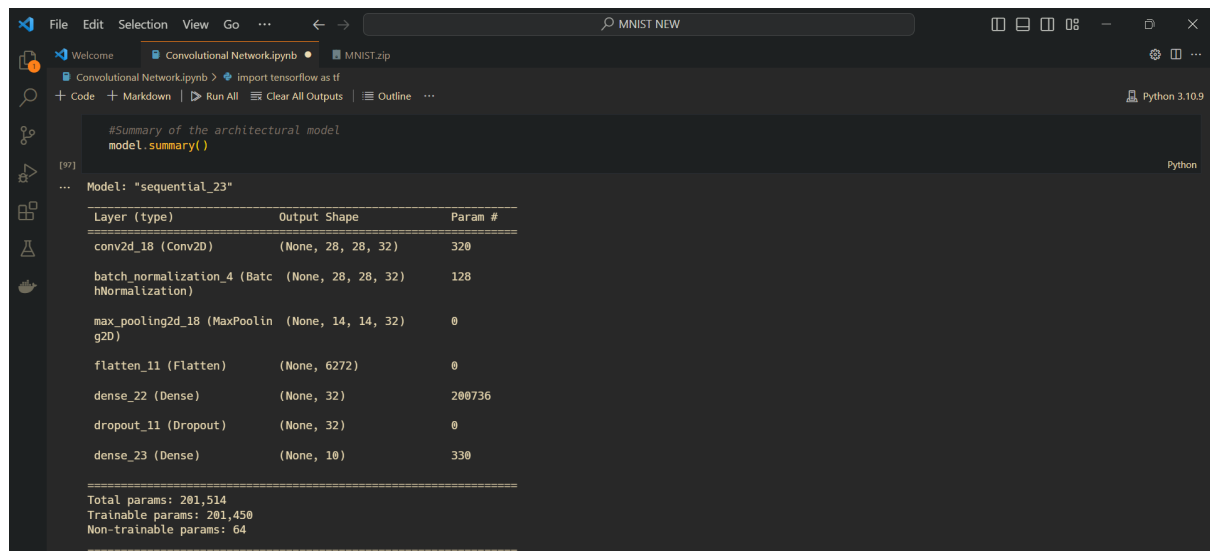
```
# Define the CNN model
model = tf.keras.Sequential([tf.keras.layers.Conv2D(32, (3, 3),
    activation='relu', padding='same', input_shape=(28, 28, 1)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
```

```
tf.keras.layers.Dense(32, activation =  
tf.nn.sigmoid),  
tf.keras.layers.Dropout(0.2),  
tf.keras.layers.Dense(10,  
activation=tf.nn.softmax)])
```

Arsitektur model Convolutional Neural Network (CNN) menggunakan TensorFlow.

```
#Summary of the architectural model  
model.summary()
```

model.summary() adalah sebuah method pada TensorFlow yang digunakan untuk menampilkan rangkuman dari arsitektur model yang telah dibuat.



Layer (type)	Output Shape	Param #
conv2d_18 (Conv2D)	(None, 28, 28, 32)	320
batch_normalization_4 (Batch Normalization)	(None, 28, 28, 32)	128
max_pooling2d_18 (Max Pooling)	(None, 14, 14, 32)	0
flatten_11 (Flatten)	(None, 6272)	0
dense_22 (Dense)	(None, 32)	200736
dropout_11 (Dropout)	(None, 32)	0
dense_23 (Dense)	(None, 10)	330

Total params: 201,514
Trainable params: 201,450
Non-trainable params: 64

Arsitektur model yang didefinisikan terdiri dari beberapa layer, yaitu:

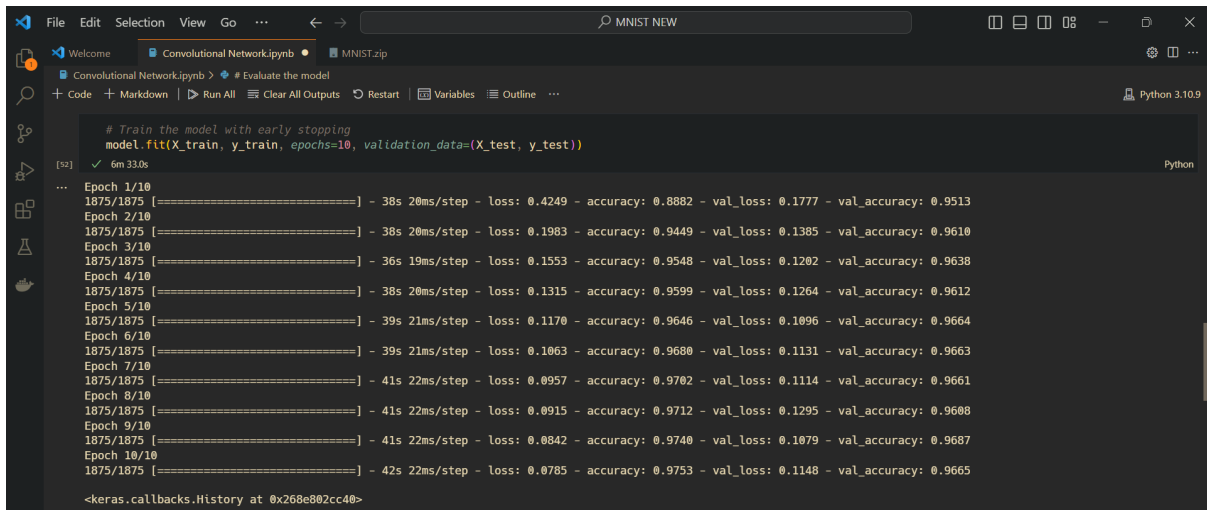
- **Convolutional Layer:** memproses input gambar dengan aktivasi ReLU
- **Batch Normalization Layer:** mempercepat training dan meningkatkan stabilitas model
- **Max Pooling Layer:** downsampling
- **Flatten Layer:** mengubah output dari layer sebelumnya menjadi vektor satu dimensi
- **Dense Layer:** memiliki 32 neuron dan dengan aktivasi sigmoid, mempelajari fitur pada gambar.
- **Dropout Layer:** mencegah overfitting dengan menonaktifkan sejumlah neuron secara acak pada layer
- **Dense Output Layer:** memiliki 10 neuron dengan aktivasi softmax, menghasilkan output kelas pada model

```
# Compile the model  
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam',  
metrics=['accuracy'])
```

Mengkonfigurasi model yang telah dibuat sebelumnya untuk proses pelatihan, fungsi ini menerima beberapa parameter yang berguna untuk menentukan bagaimana model akan dijalankan selama proses pelatihan.

```
# Train the model with early stopping  
model.fit(X_train, y_train, epochs=10, validation_data=(X_test, y_test))
```

model.fit digunakan untuk melatih model. Pada kode di atas, dilakukan pelatihan model selama 10 epoch atau iterasi dengan menggunakan data pelatihan (X_train dan y_train) dan data validasi (X_test dan y_test) yang diberikan pada argumen validation_data.



```
# Train the model with early stopping
model.fit(X_train, y_train, epochs=10, validation_data=(X_test, y_test))

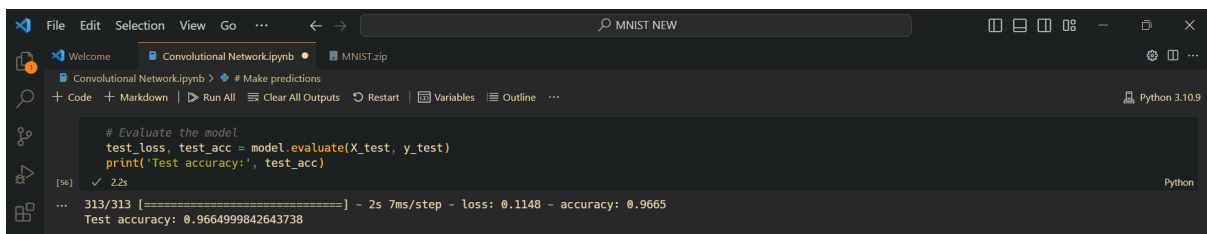
Epoch 1/10
1875/1875 [=====] - 38s 20ms/step - loss: 0.4249 - accuracy: 0.8882 - val_loss: 0.1777 - val_accuracy: 0.9513
Epoch 2/10
1875/1875 [=====] - 38s 20ms/step - loss: 0.1983 - accuracy: 0.9449 - val_loss: 0.1385 - val_accuracy: 0.9610
Epoch 3/10
1875/1875 [=====] - 36s 19ms/step - loss: 0.1553 - accuracy: 0.9548 - val_loss: 0.1202 - val_accuracy: 0.9638
Epoch 4/10
1875/1875 [=====] - 38s 20ms/step - loss: 0.1315 - accuracy: 0.9599 - val_loss: 0.1264 - val_accuracy: 0.9612
Epoch 5/10
1875/1875 [=====] - 39s 21ms/step - loss: 0.1170 - accuracy: 0.9646 - val_loss: 0.1096 - val_accuracy: 0.9664
Epoch 6/10
1875/1875 [=====] - 39s 21ms/step - loss: 0.1063 - accuracy: 0.9680 - val_loss: 0.1131 - val_accuracy: 0.9663
Epoch 7/10
1875/1875 [=====] - 41s 22ms/step - loss: 0.0957 - accuracy: 0.9702 - val_loss: 0.1114 - val_accuracy: 0.9661
Epoch 8/10
1875/1875 [=====] - 41s 22ms/step - loss: 0.0915 - accuracy: 0.9712 - val_loss: 0.1295 - val_accuracy: 0.9688
Epoch 9/10
1875/1875 [=====] - 41s 22ms/step - loss: 0.0842 - accuracy: 0.9740 - val_loss: 0.1079 - val_accuracy: 0.9687
Epoch 10/10
1875/1875 [=====] - 42s 22ms/step - loss: 0.0785 - accuracy: 0.9753 - val_loss: 0.1148 - val_accuracy: 0.9665

<keras.callbacks.History at 0x268e02cc40>
```

Model melakukan pemberhentian pada iterasi (epoch) ke 10 dengan total waktu pelatihan selama **6 menit 33.0 detik**

```
# Evaluate the model
test_loss, test_acc = model.evaluate(X_test, y_test)
print('Test accuracy:', test_acc)
```

evaluate() untuk mengevaluasi model pada data uji (X_test dan y_test). Kemudian, menyimpan hasil evaluasi berupa nilai loss dan akurasi pada variabel test_loss dan test_acc masing-masing.



```
# Evaluate the model
test_loss, test_acc = model.evaluate(X_test, y_test)
print('Test accuracy:', test_acc)

313/313 [=====] - 2s 7ms/step - loss: 0.1148 - accuracy: 0.9665
Test accuracy: 0.9664999842643738
```

Didapatkan hasil **akurasinya 96.65%**

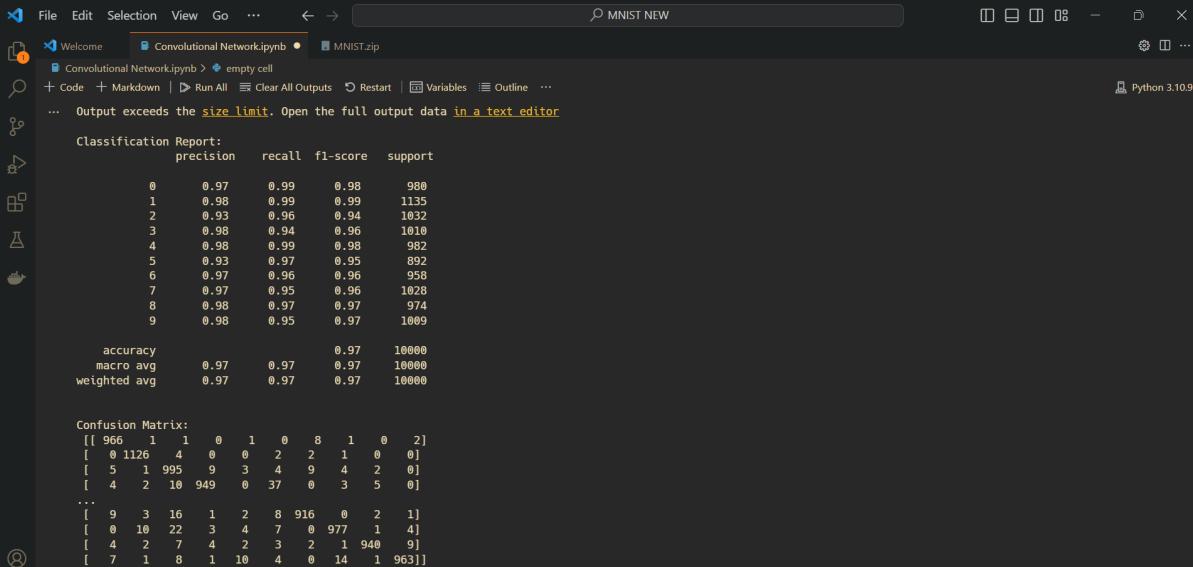
```
# Make predictions
y_pred = model.predict(X_test)
y_pred = np.argmax(y_pred, axis=1)
```

Membuat prediksi pada data uji menggunakan model yang telah di training sebelumnya, model dipanggil menggunakan metode predict dengan argumen X_test yang berisi data uji. Kemudian, hasil prediksi tersebut disimpan pada variabel y_pred.

```
# Print classification report and confusion matrix
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

print('\nClassification Report:\n', classification_report(y_test, y_pred))
print('\nConfusion Matrix:\n', confusion_matrix(y_test, y_pred))
```

Mengimpor fungsi classification_report dan confusion_matrix dari library sklearn.metrics, fungsi classification_report memerlukan dua parameter yaitu y_test yang merupakan array target dari data uji dan y_pred yang merupakan array prediksi yang dihasilkan oleh model. fungsi confusion_matrix memerlukan dua parameter yaitu y_test dan y_pred memberikan informasi mengenai berapa banyak data yang benar dan salah diklasifikasikan pada setiap kelas.



The screenshot shows a Jupyter Notebook window titled 'Convolutional Network.ipynb' with a file named 'MNIST.zip' open. The notebook is running a Python 3.10.9 kernel. The output of a cell displays a Classification Report and a Confusion Matrix. The Classification Report shows high performance metrics for a 10-class problem, with accuracy, macro avg, and weighted avg all at 0.97. The Confusion Matrix shows the distribution of predicted vs. actual classes, with a strong diagonal indicating high classification accuracy.

```
... Output exceeds the size limit. Open the full output data in a text editor
```

```
Classification Report:
precision    recall  f1-score   support

 0 0.97 0.99 0.98  980
 1 0.98 0.99 0.99 1135
 2 0.93 0.96 0.94 1032
 3 0.98 0.94 0.96 1010
 4 0.98 0.99 0.98  982
 5 0.93 0.97 0.95  892
 6 0.97 0.96 0.96  958
 7 0.97 0.95 0.96 1028
 8 0.98 0.97 0.97  974
 9 0.98 0.95 0.97 1009

 accuracy 0.97 10000
 macro avg 0.97 0.97 0.97 10000
 weighted avg 0.97 0.97 0.97 10000

Confusion Matrix:
[[ 966  1  1  0  1  0  8  1  0  2]
 [ 0 1126  4  0  0  2  2  1  0  0]
 [ 5  1 995  9  3  4  9  4  2  0]
 [ 4  2 10 949  0 37  0  3  5  0]
 ...
 [ 9  3 16  1  2  8 916  0  2  1]
 [ 0 10 22  3  4  7  0 977  1  4]
 [ 4  2  7  4  2  3  2  1 940  9]
 [ 7  1  8  1 10  4  0 14  1 963]]
```

Hasil dari Classification Report dan Confusion Matrix

- **AlexNet**

```
import tensorflow as tf
from tensorflow.keras.layers import Conv2D, MaxPooling2D, BatchNormalization,
Flatten, Dense, Dropout
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D,
MaxPooling2D
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import EarlyStopping
```

```
# Load MNIST dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

Memuat **dataset MNIST** (sama halnya dengan deskripsi pada program Convolutional Network)

```
# Normalize pixel values to 0-1
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0
```

Melakukan **normalisasi** data pada dataset gambar, yaitu dengan mengubah rentang nilai pixel dari 0-255 menjadi 0-1.

```
# Convert labels to one-hot encoding
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)
```

Mengubah label dari dataset MNIST menjadi one-hot encoding menggunakan fungsi `to_categorical()`.

```
# Reshape input to 4D tensor
X_train = X_train.reshape(X_train.shape[0], 28, 28, 1)
X_test = X_test.reshape(X_test.shape[0], 28, 28, 1)
```

Mengubah dimensi data gambar dari dataset MNIST menjadi 4D tensor dengan ukuran (jumlah data x lebar x tinggi x jumlah saluran) menggunakan fungsi `reshape()`.

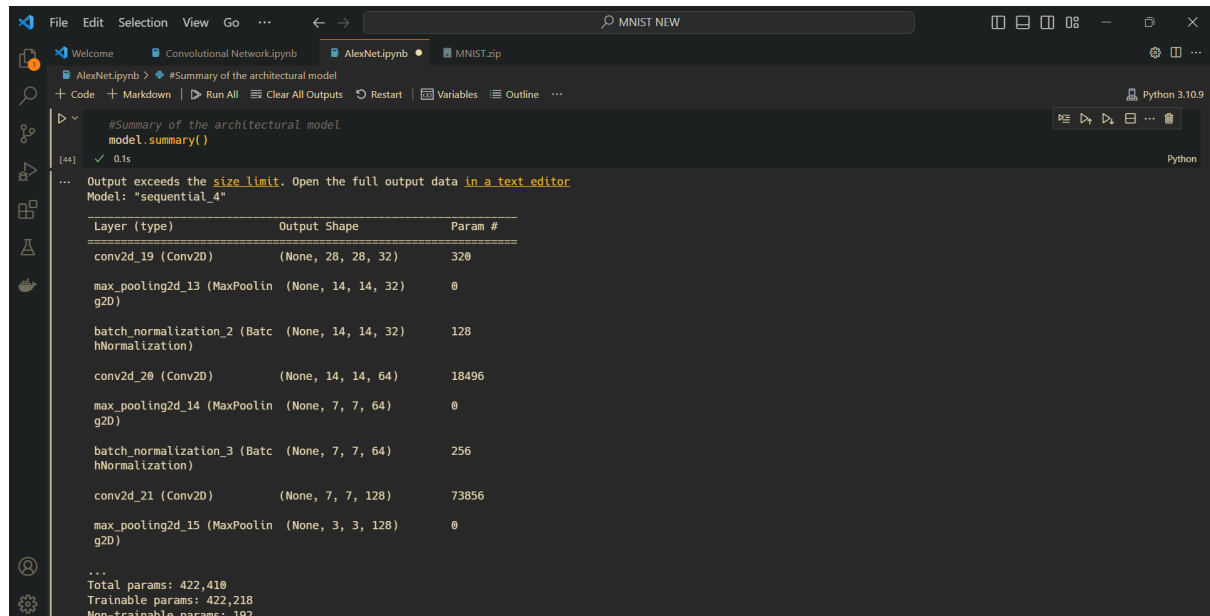
```
# Define AlexNet model
model = tf.keras.models.Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(28, 28, 1),
padding='same'),
    MaxPooling2D(pool_size=(2, 2), strides=(2,2)),
    BatchNormalization(),
    Conv2D(64, (3,3), activation='relu', padding='same'),
    MaxPooling2D(pool_size=(2, 2), strides=(2,2)),
    BatchNormalization(),
    Conv2D(128, (3,3), activation='relu', padding='same'),
    MaxPooling2D(pool_size=(2, 2), strides=(2,2)),
    Flatten(),
    Dense(256, activation='relu'),
    Dropout(0.5),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(10, activation='softmax')
```

```
] )
```

Arsitektur model AlexNet menggunakan TensorFlow.

```
#Summary of the architectural model
model.summary()
```

model.summary() adalah sebuah method pada TensorFlow yang digunakan untuk menampilkan rangkuman dari arsitektur model yang telah dibuat.



```
#Summary of the architectural model
model.summary()
```

Output exceeds the size limit. Open the full output data in a text editor
Model: "sequential_4"

Layer (type)	Output Shape	Param #
conv2d_19 (Conv2D)	(None, 28, 28, 32)	320
max_pooling2d_13 (MaxPooling2D)	(None, 14, 14, 32)	0
batch_normalization_2 (Batch Normalization)	(None, 14, 14, 32)	128
conv2d_20 (Conv2D)	(None, 14, 14, 64)	18496
max_pooling2d_14 (MaxPooling2D)	(None, 7, 7, 64)	0
batch_normalization_3 (Batch Normalization)	(None, 7, 7, 64)	256
conv2d_21 (Conv2D)	(None, 7, 7, 128)	73856
max_pooling2d_15 (MaxPooling2D)	(None, 3, 3, 128)	0
...		
Total params: 422,416		
Trainable params: 422,218		
Non-trainable params: 192		

Arsitektur model yang didefinisikan terdiri dari beberapa layer, yaitu:

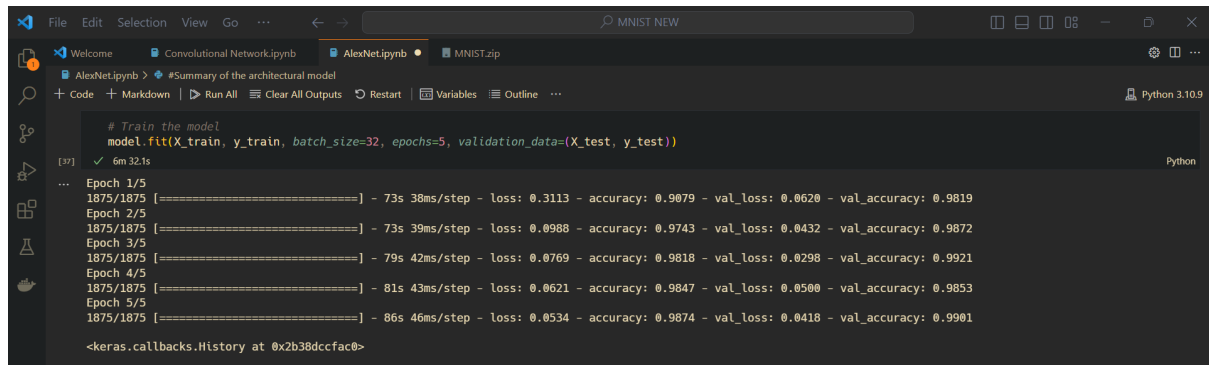
- **Conv2D**: menggunakan 32 filter dengan ukuran 3x3 piksel, fungsi aktivasi relu
- **MaxPooling2D**: (downsampling) dengan ukuran 2x2 piksel
- **BatchNormalization**: meningkatkan kecepatan dan stabilitas pelatihan model
- **Conv2D**: menggunakan 64 filter dengan ukuran 3x3 piksel, fungsi aktivasi relu
- **MaxPooling2D**: (downsampling) dengan ukuran 2x2 piksel
- **BatchNormalization**: normalisasi batch
- **Conv2D**: menggunakan 128 filter dengan ukuran 3x3 piksel, fungsi aktivasi relu
- **MaxPooling2D**: (downsampling) dengan ukuran 2x2 piksel
- **Flatten**: meratakan citra menjadi vektor 1 dimensi
- **Dense**: (fully-connected) dengan 256 unit neuron, fungsi aktivasi relu
- **Dropout**: membantu mencegah overfitting pada model (probabilitas dropout sebesar 0.5)
- **Dense**: (fully-connected) dengan 128 unit neuron, fungsi aktivasi relu
- **Dropout**: membantu mencegah overfitting pada model (probabilitas dropout sebesar 0.5)
- **Dense**: (fully-connected) dengan 10 unit neuron, fungsi aktivasi softmax, probabilitas prediksi kelas pada model

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

Mengkonfigurasi model yang telah dibuat sebelumnya untuk proses pelatihan, fungsi ini menerima beberapa parameter yang berguna untuk menentukan bagaimana model akan dijalankan selama proses pelatihan.

```
# Train the model
model.fit(X_train, y_train, batch_size=32, epochs=5, validation_data=(X_test,
y_test))
```

model.fit digunakan untuk melatih model. Pada kode di atas, dilakukan pelatihan model selama 5 epoch atau iterasi dengan menggunakan data pelatihan (X_train dan y_train) dan data validasi (X_test dan y_test) yang diberikan pada argumen validation_data.

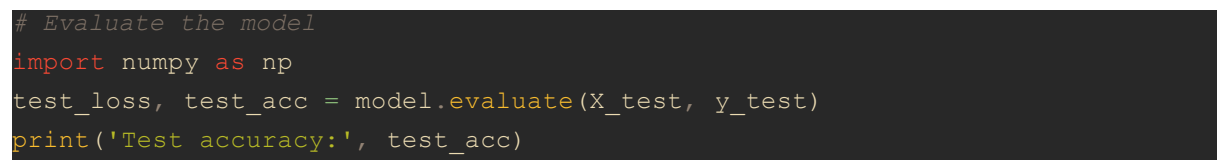


```
# Train the model
model.fit(X_train, y_train, batch_size=32, epochs=5, validation_data=(X_test, y_test))

Epoch 1/5
1875/1875 [=====] - 73s 38ms/step - loss: 0.3113 - accuracy: 0.9079 - val_loss: 0.0620 - val_accuracy: 0.9819
Epoch 2/5
1875/1875 [=====] - 73s 39ms/step - loss: 0.0988 - accuracy: 0.9743 - val_loss: 0.0432 - val_accuracy: 0.9872
Epoch 3/5
1875/1875 [=====] - 79s 42ms/step - loss: 0.0769 - accuracy: 0.9818 - val_loss: 0.0298 - val_accuracy: 0.9921
Epoch 4/5
1875/1875 [=====] - 81s 43ms/step - loss: 0.0621 - accuracy: 0.9847 - val_loss: 0.0500 - val_accuracy: 0.9853
Epoch 5/5
1875/1875 [=====] - 86s 46ms/step - loss: 0.0534 - accuracy: 0.9874 - val_loss: 0.0418 - val_accuracy: 0.9901

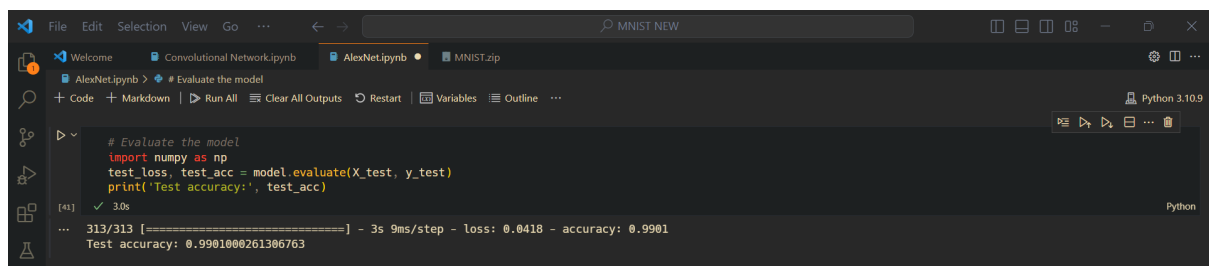
<keras.callbacks.History at 0x2b38dcfac0>
```

Model melakukan pemberhentian pada iterasi (epoch) ke 5 dengan total waktu pelatihan selama **6 menit 32.1 detik**



```
# Evaluate the model
import numpy as np
test_loss, test_acc = model.evaluate(X_test, y_test)
print('Test accuracy:', test_acc)
```

evaluate() untuk mengevaluasi model pada data uji (X_test dan y_test). Kemudian, menyimpan hasil evaluasi berupa nilai loss dan akurasi pada variabel test_loss dan test_acc masing-masing.



```
# Evaluate the model
import numpy as np
test_loss, test_acc = model.evaluate(X_test, y_test)
print('Test accuracy:', test_acc)

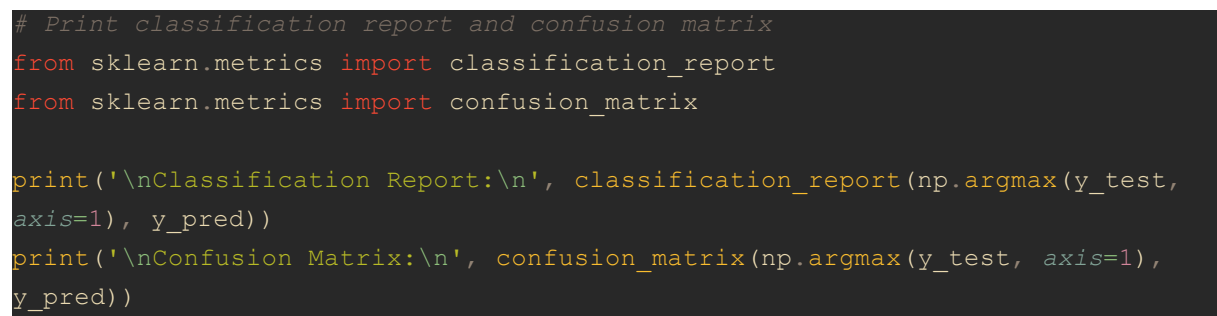
Test accuracy: 0.9901000261306763
```

Didapatkan hasil **akurasinya 99.01%**



```
# Make predictions
y_pred = model.predict(X_test)
y_pred = np.argmax(y_pred, axis=1)
```

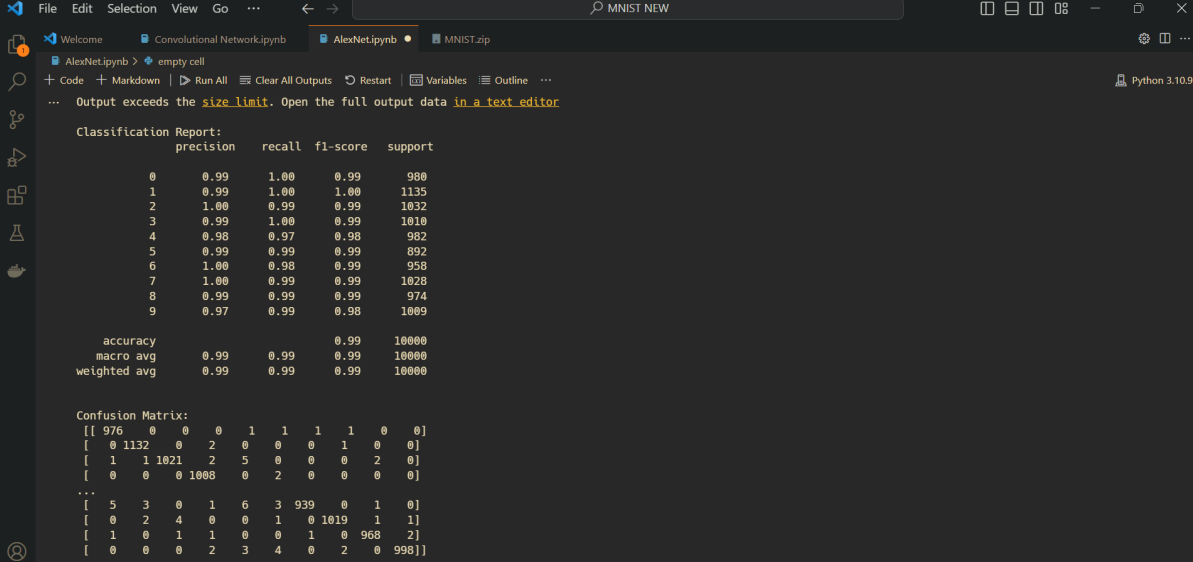
Membuat prediksi pada data uji menggunakan model yang telah di training sebelumnya, model dipanggil menggunakan metode predict dengan argumen X_test yang berisi data uji. Kemudian, hasil prediksi tersebut disimpan pada variabel y_pred.



```
# Print classification report and confusion matrix
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

print('\nClassification Report:\n', classification_report(np.argmax(y_test, axis=1), y_pred))
print('\nConfusion Matrix:\n', confusion_matrix(np.argmax(y_test, axis=1), y_pred))
```

Mengimpor fungsi `classification_report` dan `confusion_matrix` dari library `sklearn.metrics`, fungsi `classification_report` memerlukan dua parameter yaitu `y_test` yang merupakan array target dari data uji dan `y_pred` yang merupakan array prediksi yang dihasilkan oleh model. fungsi `confusion_matrix` memerlukan dua parameter yaitu `y_test` dan `y_pred` memberikan informasi mengenai berapa banyak data yang benar dan salah diklasifikasikan pada setiap kelas.



The screenshot shows a Jupyter Notebook interface with the following output:

```
Output exceeds the size limit. Open the full output data in a text editor
```

Classification Report:

	precision	recall	f1-score	support
0	0.99	1.00	0.99	980
1	0.99	1.00	1.00	1135
2	1.00	0.99	0.99	1032
3	0.99	1.00	0.99	1010
4	0.98	0.97	0.98	982
5	0.99	0.99	0.99	892
6	1.00	0.98	0.99	958
7	1.00	0.99	0.99	1028
8	0.99	0.99	0.99	974
9	0.97	0.99	0.98	1009
accuracy			0.99	10000
macro avg	0.99	0.99	0.99	10000
weighted avg	0.99	0.99	0.99	10000

Confusion Matrix:

```
[[ 976  0  0  0  1  1  1  1  0  0]
 [ 0 1132  0  2  0  0  0  1  0  0]
 [ 1  1 1021  2  5  0  0  0  2  0]
 [ 0  0  0 1000  0  2  0  0  0  0]
 ...
 [ 5  3  0  1  6  3  939  0  1  0]
 [ 0  2  4  0  0  1  0 1019  1  1]
 [ 1  0  1  1  0  0  1  0  968  2]
 [ 0  0  0  2  3  4  0  2  0  998]]
```

Hasil dari Classification Report dan Confusion Matrix