Nama : Rizky Bagus Paramadanï

NIM : 226150100111019

Kelas : Machine Learnig – A

Tugas

- Build a convolutional network with pre-processing on the input data (jittering, normalization). Also add dropout regularization, batch normalization, and at least one additional convolutional layer which achieves at least 90% test accuracy (for any training epoch) on MNIST dataset. Your part 1 network should train under 10 minutes, without GPUs.
- Fine-tune AlexNet to achieve at least 80% test accuracy on the MNIST dataset. Your network should train under 10 minutes, without GPUs.

Jawab :

1. Model CNN
   a. Source Code

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import mnist
from tensorflow.keras.layers import Conv2D,
MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.layers import
BatchNormalization
import time
import os
```

```
# Load MNIST
def load_dataset():
  (x_train, y_train), (x_test, y_test) =
mnist.load_data()
  x_train = x_train.reshape(-1, 28, 28, 1)
  x_test = x_test.reshape(-1, 28, 28, 1)
  y_train = to_categorical(y_train)
  y_test = to_categorical(y_test)
  return (x_train, y_train,x_test,y_test)

x_train, y_train,x_test,y_test = load_dataset()
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
input_shape = (28, 28, 1)
```

```
# Data augmentation with ImageDataGenerator
datagen = ImageDataGenerator(rotation_range=10,
zoom_range=0.1, width_shift_range=0.1,
height_shift_range=0.1)
datagen.fit(x_train)
start = time.time()

# Define CNN model
model = Sequential([
    Conv2D(32, kernel_size=(3, 3), activation='relu',
input_shape=input_shape),
    BatchNormalization(),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(64, kernel_size=(3, 3),
activation='relu'),
    BatchNormalization(),
```

```
    MaxPooling2D(pool_size=(2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])

# Compile model
opt = SGD(learning_rate=0.01, momentum=0.9)
model.compile(optimizer=opt,
loss='categorical_crossentropy',
metrics=['accuracy'])
# Train model with data augmentation
history = model.fit(datagen.flow(x_train, y_train,
batch_size=32),
                                steps_per_epoch=len(x_t
rain) // 32, epochs=10, validation_data=(x_test,
y_test))
end = time.time()
total = end-start
test_loss, test_acc = model.evaluate(x_test, y_test)
```

Source code diatas merupakan model dari CNN menggunakan dataset MNIST dan dilakukan augmentasi menggunakan ImageDataGenerator yang disediakan oleh Keras. Data augmentasi menggunakan ImageDataGenerator menggunakan parameter augmentasi seperti rotasi, zoom, dan pergeseran yang selanjutnya mendefinisikan model CNN. Model CNN terdiri dari dua layer Conv2D dengan aktivasi ReLu, BatchNormalization, MaxPooling2D, dan dua Dense layer dengan masing-masing menggunakan aktivasi ReLu dan Softmax. Model dilatih dengan menggunakan generator data yang telah dibuat sebelumnya, dengan ukuran batch 32 dan 10 epoch.

```
print('Test loss:', test_loss)
print('Test accuracy:', test_acc)
print(f"Finsih Traine {total/60} minute")
os.environ['CUDA_VISIBLE_DEVICES'] = ''
if tf.test.gpu_device_name():
```

```
    print('GPU found')
else:
    print("No GPU found")
```

```
predictions = model.predict(x_test)
print(np.argmax(np.round(predictions[0])))
plt.imshow(x_test[0].reshape(28, 28), cmap =
plt.cm.binary)
plt.show()
```

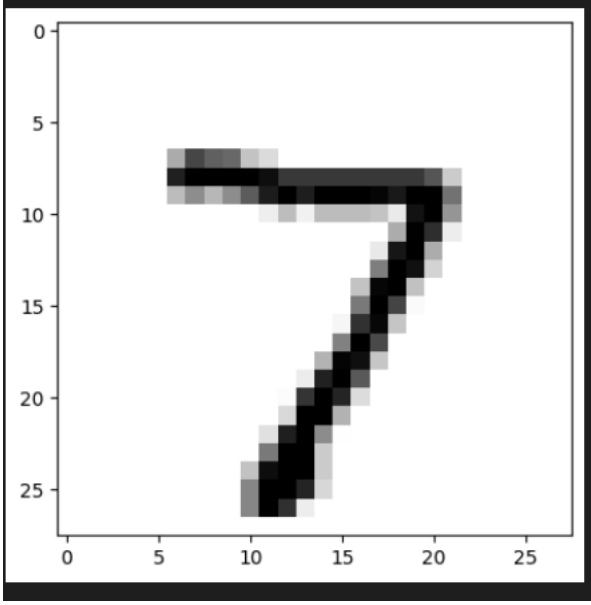
b. Hasil

```
1875/1875 [==============================] - 40s 21ms/step - loss: 0.2066 - accuracy: 0.9369 - val_loss: 0.0553 - val_accuracy: 0.9823
Epoch 2/10
1875/1875 [==============================] - 39s 21ms/step - loss: 0.0869 - accuracy: 0.9739 - val_loss: 0.0400 - val_accuracy: 0.9879
Epoch 3/10
1875/1875 [==============================] - 40s 21ms/step - loss: 0.0733 - accuracy: 0.9778 - val_loss: 0.0358 - val_accuracy: 0.9882
Epoch 4/10
1875/1875 [==============================] - 41s 22ms/step - loss: 0.0574 - accuracy: 0.9830 - val_loss: 0.0359 - val_accuracy: 0.9893
Epoch 5/10
1875/1875 [==============================] - 40s 21ms/step - loss: 0.0499 - accuracy: 0.9851 - val_loss: 0.0512 - val_accuracy: 0.9824
Epoch 6/10
1875/1875 [==============================] - 40s 22ms/step - loss: 0.0490 - accuracy: 0.9853 - val_loss: 0.0277 - val_accuracy: 0.9914
Epoch 7/10
1875/1875 [==============================] - 41s 22ms/step - loss: 0.0427 - accuracy: 0.9873 - val_loss: 0.0301 - val_accuracy: 0.9898
Epoch 8/10
1875/1875 [==============================] - 41s 22ms/step - loss: 0.0401 - accuracy: 0.9873 - val_loss: 0.0335 - val_accuracy: 0.9902
Epoch 9/10
1875/1875 [==============================] - 40s 21ms/step - loss: 0.0372 - accuracy: 0.9887 - val_loss: 0.0351 - val_accuracy: 0.9902
Epoch 10/10
1875/1875 [==============================] - 40s 21ms/step - loss: 0.0341 - accuracy: 0.9893 - val_loss: 0.0273 - val_accuracy: 0.9918
313/313 [==============================] - 2s 6ms/step - loss: 0.0273 - accuracy: 0.9918
```

```
Test loss: 0.027275424450635 91
Test accuracy: 0.9918000102043152
Finsih Traine 6.717317668596904 minute
No GPU found
```

Setelah model dilatih menggunakan dataset MNIST dan dievalusi mengasilkan nilai Test Loss sebesar 0.027 yang menunjukkan bahwa model memiliki peforma yang baik dalam meprediksi label pada data uji. Nilai Test Accuracy sebesar 99,18% menunjukkan bahwa model dapat mengklasifikasikan data uji dengan baik. Dengan waktu untuk melatih model selama 6,7 menit tanpa menggunakan GPU.

```
313/313 [==============================] - 2s 5ms/step
7
```



2. AleXNet + Fine Tune

a. Source Code

```
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras import datasets, layers,
models, losses
import numpy as np
import matplotlib.pyplot as plt
import os
```

```
(x_train, y_train), (x_test, y_test)=tf.keras.datasets.mnist.load_dat
a()
x_train = tf.pad(x_train, [[0, 0], [2,2], [2,2]])/255
x_test = tf.pad(x_test, [[0, 0], [2,2], [2,2]])/255
x_train = tf.expand_dims(x_train, axis=3, name=None)
x_test = tf.expand_dims(x_test, axis=3, name=None)
x_train = tf.repeat(x_train, 3, axis=3)
x_test = tf.repeat(x_test, 3, axis=3)
x_val = x_train[-2000:,:,:,:]
y_val = y_train[-2000:]
x_train = x_train[:-2000,:,:,:]
y_train = y_train[:-2000]
```

```
(x_train, y_train), (x_test, y_test)=tf.keras.datasets.mnist.load_dat
a()
```

```
import time
start = time.time()
model = models.Sequential()
model.add(layers.experimental.preprocessing.Resizing(224, 224, interp
olation="bilinear", input_shape=x_train.shape[1:]))

model.add(layers.Conv2D(96, 11, strides=4, padding='same'))
model.add(layers.Lambda(tf.nn.local_response_normalization))
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D(3, strides=2))

model.add(layers.Conv2D(256, 5, strides=4, padding='same'))
model.add(layers.Lambda(tf.nn.local_response_normalization))
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D(3, strides=2))

model.add(layers.Conv2D(384, 3, strides=4, padding='same'))
model.add(layers.Activation('relu'))

model.add(layers.Conv2D(384, 3, strides=4, padding='same'))
model.add(layers.Activation('relu'))

model.add(layers.Conv2D(256, 3, strides=4, padding='same'))
model.add(layers.Activation('relu'))

model.add(layers.Flatten())
model.add(layers.Dense(4096, activation='relu'))
model.add(layers.Dropout(0.5))

model.add(layers.Dense(4096, activation='relu'))
model.add(layers.Dropout(0.5))

model.add(layers.Dense(10, activation='softmax'))


for layer in model.layers[:-1]:
    layer.trainable = False

model.compile(optimizer='adam', loss=losses.sparse_categorical_crosse
ntropy, metrics=['accuracy'])

history = model.fit(x_train, y_train, batch_size=64, epochs=3, valida
tion_data=(x_val, y_val))
```

```
end = time.time()
total = end-start
```

Source code diatas merupakan model AlexNet yang digunakan untuk klasikasi citra dengan dataset MNIST yang memiliki lima layer konvolusi yang memiliki ouput layer dari 10 neuron dan menggunakan fungsi aktifasi softmax, fine tuning dilakukan dengan cara mengatur agar semua layer kecuali layer ouput tidak terpengaruh oleh proses pelatihan.

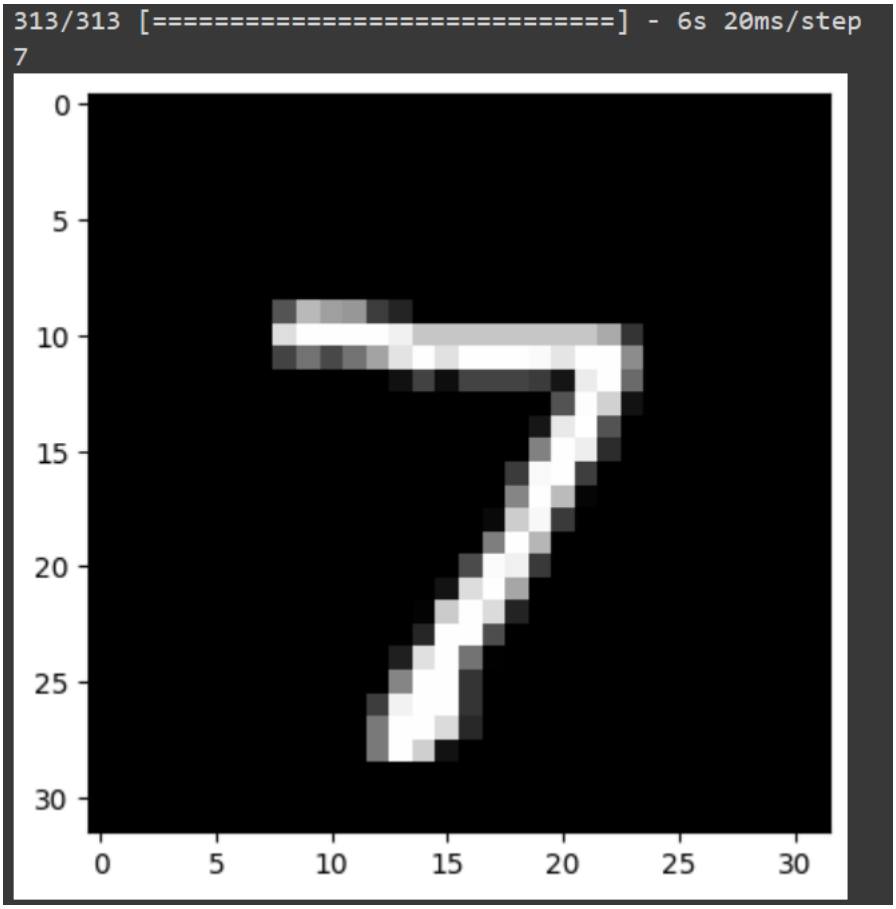
```
test_loss, test_acc = model.evaluate(x_test, y_test)
print('Test loss:', test_loss)
print('Test accuracy:', test_acc)
print(f"Finsih Traine {total/60} minute")
os.environ['CUDA_VISIBLE_DEVICES'] = ''
if tf.test.gpu_device_name():
    print('GPU found')
else:
    print("No GPU found")
```

```
predictions = model.predict(x_test)
print(np.argmax(np.round(predictions[0])))
plt.imshow(x_test[0], cmap = plt.cm.binary)
plt.show()
```

b. Hasil

```
Epoch 1/3
907/907 [==============================] - 108s 117ms/step - loss: 0.4336 - accuracy: 0.8475 - val_loss: 0.0784 - val_accuracy: 0.9795
Epoch 2/3
907/907 [==============================] - 104s 115ms/step - loss: 0.0926 - accuracy: 0.9760 - val_loss: 0.0679 - val_accuracy: 0.9810
Epoch 3/3
907/907 [==============================] - 104s 115ms/step - loss: 0.0735 - accuracy: 0.9815 - val_loss: 0.0423 - val_accuracy: 0.9870
```

```
313/313 [==============================] - 7s 22ms/step - loss: 0.0782 - accuracy: 0.9814
Test loss: 0.07819024473428726
Test accuracy: 0.9814000129699707
Finsih Traine 5.386426516373953 minute
No GPU found
```



Hasil dari pelatihan model ini mendapatkan nilai akurasi 98% dengan waktu traine 5,4 menit.