import tensorflow

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Conv2D, Flatten

# Configuration

img\_width, img\_height = 28, 28

input\_shape = (img\_width, img\_height, 1)

batch\_size = 1000

no\_epochs = 25

no\_classes = 10

validation\_split = 0.2

verbosity = 1

# Load data

def load\_data():

return tensorflow.keras.datasets.mnist.load\_data(path="mnist.npz")

# Model creation

def create\_model():

model = Sequential()

model.add(Conv2D(4, kernel\_size=(3, 3), activation='relu', input\_shape=input\_shape))

model.add(Conv2D(8, kernel\_size=(3, 3), activation='relu'))

model.add(Conv2D(12, kernel\_size=(3, 3), activation='relu'))

model.add(Flatten())

model.add(Dense(256, activation='relu'))

model.add(Dense(no\_classes, activation='softmax'))

return model

# Model compilation

def compile\_model(model):

model.compile(loss=tensorflow.keras.losses.sparse\_categorical\_crossentropy,

optimizer=tensorflow.keras.optimizers.Adam(),

metrics=['accuracy'])

return model

# Model training

def train\_model(model, X\_train, y\_train):

model.fit(X\_train, y\_train,

batch\_size=batch\_size,

epochs=no\_epochs,

verbose=verbosity,

shuffle=True,

validation\_split=validation\_split)

return model

# Model testing

def test\_model(model, X\_test, y\_test):

score = model.evaluate(X\_test, y\_test, verbose=0)

print(f'Test loss: {score[0]} / Test accuracy: {score[1]}')

return model

# Load data

(X\_train, y\_train), (X\_test, y\_test) = load\_data()

# Normalize data

(X\_train, X\_test) = (X\_train / 255.0, X\_test / 255.0)

# Reshape data

(X\_train, X\_test) = (

X\_train.reshape(X\_train.shape[0], X\_train.shape[1], X\_train.shape[2], 1),

X\_test.reshape(X\_test.shape[0], X\_test.shape[1], X\_test.shape[2], 1),

)

# Create and train the model

model = create\_model()

model = compile\_model(model)

model = train\_model(model, X\_train, y\_train)

model = test\_model(model, X\_test, y\_test)