import csv

import numpy as np

import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from os import getcwd

def get\_data(filename):

with open(filename) as training\_file:

# Your code starts here

reader = csv.reader(training\_file, delimiter=',')

imgs = []

labels = []

next(reader, None)

for row in reader:

label = row[0]

data = row[1:]

img = np.array(data).reshape((28, 28))

imgs.append(img)

labels.append(label)

images = np.array(imgs).astype(float)

labels = np.array(labels).astype(float)

# Your code ends here

return images, labels

path\_sign\_mnist\_train = f"{getcwd()}/../tmp2/sign\_mnist\_train.csv"

path\_sign\_mnist\_test = f"{getcwd()}/../tmp2/sign\_mnist\_test.csv"

training\_images, training\_labels = get\_data(path\_sign\_mnist\_train)

testing\_images, testing\_labels = get\_data(path\_sign\_mnist\_test)

# Keep these

print(training\_images.shape)

print(training\_labels.shape)

print(testing\_images.shape)

print(testing\_labels.shape)

# Their output should be:

# (27455, 28, 28)

# (27455,)

# (7172, 28, 28)

# (7172,)

training\_images = np.expand\_dims(training\_images, axis=3) # Your Code Here

testing\_images = np.expand\_dims(testing\_images, axis=3) # Your Code Here

# Create an ImageDataGenerator and do Image Augmentation

train\_datagen = ImageDataGenerator(

rescale=1. / 255,

rotation\_range=40,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True,

fill\_mode='nearest'

# Your Code Here

)

validation\_datagen = ImageDataGenerator(

rescale=1 / 255

# Your Code Here

)

# Keep These

print(training\_images.shape)

print(testing\_images.shape)

# Their output should be:

# (27455, 28, 28, 1)

# (7172, 28, 28, 1)

# Define the model

# Use no more than 2 Conv2D and 2 MaxPooling2D

model = tf.keras.models.Sequential([

tf.keras.layers.Conv2D(32, (3,3), activation='relu', input\_shape=(28, 28, 1)),

tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Conv2D(32, (3,3), activation='relu'),

tf.keras.layers.MaxPooling2D(2,2),

tf.keras.layers.Flatten(),

tf.keras.layers.Dense(512, activation='relu'),

tf.keras.layers.Dense(26, activation='softmax')

# Your Code Here

])

# Compile Model.

model.compile(optimizer='adam',

loss='sparse\_categorical\_crossentropy',

metrics=['accuracy']

# Your Code Here

)

train\_gen = train\_datagen.flow(

training\_images,

training\_labels,

batch\_size=10

)

val\_gen = validation\_datagen.flow(

testing\_images,

testing\_labels,

batch\_size=10

)

# Train the Model

history = model.fit\_generator(train\_gen,

epochs=2,

validation\_data=val\_gen

# Your Code Here (set 'epochs' = 2)

)

model.evaluate(testing\_images, testing\_labels, verbose=0)

# Plot the chart for accuracy and loss on both training and validation

%matplotlib inline

import matplotlib.pyplot as plt

acc = history.history['accuracy'] # Your Code Here

val\_acc = history.history['val\_accuracy'] # Your Code Here

loss = history.history['loss'] # Your Code Here

val\_loss = history.history['val\_loss'] # Your Code Here

epochs = range(len(acc))

plt.plot(epochs, acc, 'r', label='Training accuracy')

plt.plot(epochs, val\_acc, 'b', label='Validation accuracy')

plt.title('Training and validation accuracy')

plt.legend()

plt.figure()

plt.plot(epochs, loss, 'r', label='Training Loss')

plt.plot(epochs, val\_loss, 'b', label='Validation Loss')

plt.title('Training and validation loss')

plt.legend()

plt.show()