import csv

import numpy as np

import tensorflow as tf

from sklearn.model\_selection import train\_test\_split

RANDOM\_SEED = 42 import pandas as pd

# Specify each path

dataset = 'model/keypoint\_classifier/keypoint.csv'  
model\_save\_path = 'model/keypoint\_classifier/keypoint\_classifier\_10\_signs.keras'  
tflite\_save\_path = 'model/keypoint\_classifier/keypoint\_classifier.tflite'

# Set number of classes

NUM\_CLASSES = 10

# Dataset reading

X\_dataset = np.loadtxt(dataset, delimiter=',', dtype='float32', usecols=list(range(1, (21 \* 2) + 1)))  
data\_set = pd.read\_csv(dataset)  
data\_set

0 0.0 0.0.1 -0.19696969696969696 -0.12626262626262627 \  
0 0 0.0 0.0 -0.203046 -0.121827   
1 0 0.0 0.0 -0.203046 -0.121827   
2 0 0.0 0.0 -0.202020 -0.126263   
3 0 0.0 0.0 -0.195980 -0.130653   
4 0 0.0 0.0 -0.196970 -0.131313   
... .. ... ... ... ...   
3361 1 0.0 0.0 0.101351 -0.070946   
3362 1 0.0 0.0 0.109215 -0.071672   
3363 1 0.0 0.0 0.115108 -0.046763   
3364 1 0.0 0.0 0.117216 -0.043956   
3365 1 0.0 0.0 0.126866 -0.044776   
  
 -0.30303030303030304 -0.2878787878787879 -0.3282828282828283 \  
0 -0.309645 -0.284264 -0.335025   
1 -0.304569 -0.284264 -0.329949   
2 -0.303030 -0.287879 -0.328283   
3 -0.301508 -0.291457 -0.321608   
4 -0.297980 -0.292929 -0.323232   
... ... ... ...   
3361 0.077703 -0.185811 -0.114865   
3362 0.088737 -0.180887 -0.098976   
3363 0.100719 -0.143885 -0.104317   
3364 0.102564 -0.135531 -0.109890   
3365 0.108209 -0.123134 -0.100746   
  
 -0.4494949494949495 -0.29797979797979796 ... 0.005050505050505051 \  
0 -0.446701 -0.299492 ... 0.005076   
1 -0.451777 -0.299492 ... 0.010152   
2 -0.454545 -0.297980 ... 0.010101   
3 -0.457286 -0.296482 ... 0.010050   
4 -0.454545 -0.292929 ... 0.015152   
... ... ... ... ...   
3361 -0.206081 -0.270270 ... -0.195946   
3362 -0.191126 -0.255973 ... -0.177474   
3363 -0.140288 -0.269784 ... -0.179856   
3364 -0.113553 -0.293040 ... -0.194139   
3365 -0.089552 -0.283582 ... -0.208955   
  
 -0.9292929292929293 0.15656565656565657 -0.4494949494949495.1 \  
0 -0.928934 0.152284 -0.451777   
1 -0.928934 0.157360 -0.446701   
2 -0.929293 0.156566 -0.449495   
3 -0.929648 0.155779 -0.452261   
4 -0.929293 0.156566 -0.449495   
... ... ... ...   
3361 -0.219595 -0.256757 -0.334459   
3362 -0.197952 -0.266212 -0.307167   
3363 -0.165468 -0.291367 -0.312950   
3364 -0.120879 -0.322344 -0.300366   
3365 -0.100746 -0.328358 -0.268657   
  
 0.15656565656565657.1 -0.601010101010101 0.1414141414141414 \  
0 0.157360 -0.604061 0.142132   
1 0.162437 -0.598985 0.147208   
2 0.161616 -0.601010 0.146465   
3 0.160804 -0.603015 0.145729   
4 0.161616 -0.601010 0.141414   
... ... ... ...   
3361 -0.341216 -0.354730 -0.283784   
3362 -0.348123 -0.313993 -0.276451   
3363 -0.352518 -0.241007 -0.273381   
3364 -0.388278 -0.227106 -0.300366   
3365 -0.406716 -0.194030 -0.317164   
  
 -0.6919191919191919 0.11616161616161616 -0.7878787878787878   
0 -0.695431 0.116751 -0.786802   
1 -0.690355 0.126904 -0.786802   
2 -0.691919 0.121212 -0.787879   
3 -0.698492 0.120603 -0.788945   
4 -0.696970 0.121212 -0.787879   
... ... ... ...   
3361 -0.260135 -0.212838 -0.192568   
3362 -0.215017 -0.201365 -0.160410   
3363 -0.133094 -0.187050 -0.079137   
3364 -0.117216 -0.208791 -0.062271   
3365 -0.085821 -0.212687 -0.033582   
  
[3366 rows x 43 columns]

y\_dataset = np.loadtxt(dataset, delimiter=',', dtype='int32', usecols=(0))

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_dataset, y\_dataset, train\_size=0.75)

# Model building

model = tf.keras.models.Sequential([  
 tf.keras.layers.Input((21 \* 2, )),  
 tf.keras.layers.Dropout(0.2),  
 tf.keras.layers.Dense(20, activation='relu'),  
 tf.keras.layers.Dropout(0.4),  
 tf.keras.layers.Dense(10, activation='relu'),  
 tf.keras.layers.Dense(NUM\_CLASSES, activation='softmax')  
])

model.summary() # tf.keras.utils.plot\_model(model, show\_shapes=True)

Model: "sequential"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓  
┃ Layer (type) ┃ Output Shape ┃ Param # ┃  
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩  
│ dropout (Dropout) │ (None, 42) │ 0 │  
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤  
│ dense (Dense) │ (None, 20) │ 860 │  
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤  
│ dropout\_1 (Dropout) │ (None, 20) │ 0 │  
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤  
│ dense\_1 (Dense) │ (None, 10) │ 210 │  
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤  
│ dense\_2 (Dense) │ (None, 10) │ 110 │  
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘

Total params: 1,180 (4.61 KB)

Trainable params: 1,180 (4.61 KB)

Non-trainable params: 0 (0.00 B)

# Model checkpoint callback  
cp\_callback = tf.keras.callbacks.ModelCheckpoint(  
 model\_save\_path, verbose=1, save\_weights\_only=False)  
# Callback for early stopping  
es\_callback = tf.keras.callbacks.EarlyStopping(patience=20, verbose=1)

# Model compilation  
model.compile(  
 optimizer='adam',  
 loss='sparse\_categorical\_crossentropy',  
 metrics=['accuracy']  
)

# Model training

model.fit( X\_train, y\_train, epochs=1000, batch\_size=128, validation\_data=(X\_test, y\_test), callbacks=[cp\_callback, es\_callback] )

# Model evaluation  
val\_loss, val\_acc = model.evaluate(X\_test, y\_test, batch\_size=128)

7/7 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9626 - loss: 0.2090

# Loading the saved model  
model = tf.keras.models.load\_model(model\_save\_path)

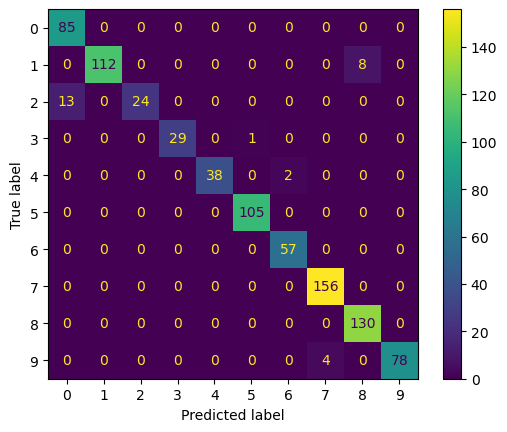
# Inference test  
predict\_result = model.predict(np.array([X\_test[0]]))  
print(np.squeeze(predict\_result))  
print(np.argmax(np.squeeze(predict\_result)))

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step  
[1.5422294e-04 6.3131541e-02 3.4740573e-05 5.3249194e-05 2.8047030e-05  
 7.1899231e-06 2.4718815e-05 6.7471917e-04 9.3580472e-01 8.6876913e-05]  
8

# Confusion matrix

import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt  
from sklearn.metrics import confusion\_matrix, classification\_report, ConfusionMatrixDisplay  
  
def print\_confusion\_matrix(y\_true, y\_pred, report=True):  
 labels = sorted(list(set(y\_true)))  
 cmx\_data = confusion\_matrix(y\_true, y\_pred, labels=labels)  
 cm\_display = ConfusionMatrixDisplay(confusion\_matrix = cmx\_data, display\_labels = labels)  
 cm\_display.plot()  
   
 if report:  
 print('Classification Report')  
 print(classification\_report(y\_test, y\_pred))  
  
Y\_pred = model.predict(X\_test)  
y\_pred = np.argmax(Y\_pred, axis=1)  
print\_confusion\_matrix(y\_test, y\_pred)

27/27 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step   
Classification Report  
 precision recall f1-score support  
  
 0 0.87 1.00 0.93 85  
 1 1.00 0.93 0.97 120  
 2 1.00 0.65 0.79 37  
 3 1.00 0.97 0.98 30  
 4 1.00 0.95 0.97 40  
 5 0.99 1.00 1.00 105  
 6 0.97 1.00 0.98 57  
 7 0.97 1.00 0.99 156  
 8 0.94 1.00 0.97 130  
 9 1.00 0.95 0.97 82  
  
 accuracy 0.97 842  
 macro avg 0.97 0.94 0.95 842  
weighted avg 0.97 0.97 0.97 842



# Convert to model for Tensorflow-Lite

# Save as a model dedicated to inference  
model.save(model\_save\_path, include\_optimizer=False)

# Transform model (quantization)  
converter = tf.lite.TFLiteConverter.from\_keras\_model(model)  
converter.optimizations = [tf.lite.Optimize.DEFAULT]  
tflite\_quantized\_model = converter.convert()  
open(tflite\_save\_path, 'wb').write(tflite\_quantized\_model)

# Inference test

interpreter = tf.lite.Interpreter(model\_path=tflite\_save\_path)  
interpreter.allocate\_tensors()

# Get I / O tensor  
input\_details = interpreter.get\_input\_details()  
output\_details = interpreter.get\_output\_details()

interpreter.set\_tensor(input\_details[0]['index'], np.array([X\_test[0]]))

%%time  
# Inference implementation  
interpreter.invoke()  
tflite\_results = interpreter.get\_tensor(output\_details[0]['index'])

print(np.squeeze(tflite\_results))  
print(np.argmax(np.squeeze(tflite\_results)))