BAGIAN 3 SOURCE CODE

Pre Precessing 4. Ipynb

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import os

data = "F:/Open CV/Learn1/Train3"
crop_path="F:/Open CV/Learn1/Train3"

for dirname, _, filenames in os.walk('F:/Open CV/Learn1/Train3/'):
    for filename in filenames:

        img_a = cv2.imread(os.path.join(dirname,filename))
        image_scale =
cv2.resize(img_a,(70,50),interpolation=cv2.INTER_AREA)
        dst = cv2.fastNlMeansDenoisingColored(image_scale,None,6,6,7,21)
        gray = cv2.cvtColor(dst,cv2.COLOR_BGR2GRAY)
#           _,threshold = cv2.threshold(img_a, 200, 50, cv2.THRESH_BINARY)
        edge = cv2.Canny(gray, 400, 600)
        contours, hierarchy =cv2.findContours(edge,
cv2.RETR_LIST,cv2.CHAIN_APPROX_NONE)

        def x_cord_contour(contours):
            #Returns the X cordinate for the contour centroid
            if cv2.contourArea(contours) >= True:
                M = cv2.moments(contours)
                return (int(M['m10']/M['m00']))
            else:
                pass


        contours_left_to_right = sorted(contours, key=x_cord_contour, reverse =
True)

        for (i,c) in enumerate(contours_left_to_right):
            cv2.drawContours(image_scale, [c], 0, (0,0,0), -1)
            B = cv2.moments(c)
            cx = int(B['m10'] / B['m00'])
            cy = int(B['m01'] / B['m00'])
#             cv2.imshow('6 - Left to Right Contour', image_scale)
#             cv2.waitKey(0)
            (x, y, w, h) = cv2.boundingRect(c)
```

```
                                # Let's now crop each contour and save these images
            cropped_contour = image_scale[y:y+5 + h+5, x:x + w+5]
#               (T_value,threshold) = cv2.threshold(cropped_contour, 200, 300,
cv2.THRESH_BINARY)
            imgResized = cv2.resize(cropped_contour, (28,28))

            image_name = "%s/manipulate_image2_%s.jpg"%(crop_path,filename)
            print (image_name)
            cv2.imwrite(image_name, imgResized)

#               print(imgResized.shape)


#           print(os.path.join(dirname, filename))
```

Pre Processing 5.Ipynb

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import os

data = "F:/Open CV/Learn1/Train4"
crop_path="F:/Open CV/Learn1/Train4"

for dirname, _, filenames in os.walk('F:/Open CV/Learn1/Train4/'):
    for filename in filenames:

        img_a = cv2.imread(os.path.join(dirname,filename),0)
        _,threshold =
cv2.threshold(img_a,100,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)

        image_name = "%s/manipulatethreshold%s"%(crop_path,filename)
        print (image_name)
        cv2.imwrite(image_name,threshold)
```

Model2.Ipynb

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from PIL import Image
import sys
import array
import cv2
import tensorflow as tf
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D,
MaxPooling2D,BatchNormalization
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import LearningRateScheduler

from tensorflow.compat.v1 import ConProto
from tensorflow.compat.v1 import InteractiveSession

import os

con = ConProto()
con.gpu_options.allow_growth = True
session = InteractiveSession(con=con)

for dirname, _, filenames in os.walk('F:\Open CV\Learn1\Train5'):
    for filename in filenames:
        print(os.path.join(dirname, filename))


columnNames = list()
columnNames.append('label')
for i in range(784):
    pixel = 'pixel'
    pixel += str(i)
    columnNames.append(pixel)
train_data = pd.DataFrame(columns = columnNames)
i=0
count=1
for dirname, _, filenames in os.walk('F:/Open CV/Learn1/Train5/train/'):
    for filename in filenames:
        image_name = os.path.join(dirname, filename)
        #print(os.path.join(dirname, filename))
        label = dirname[dirname.rindex('/')+1:]
        #print(label)
        img = Image.open(image_name)
        rawData = img.load()
```

```
        #print(rawData)
        data = []
        data.append(label)
        #28 itu dimensionnya, 784 itu 28x28
        for y in range(28):
            for x in range(28):
                data.append(rawData[x,y])
        k = 0
        train_data.loc[i] = [data[k] for k in range(785)]
        i = i+1
    print(count)
    count = count + 1


train_data.head()


columnNames = list()
columnNames.append('label')
for i in range(784):
    pixel = 'pixel'
    pixel += str(i)
    columnNames.append(pixel)
test_data = pd.DataFrame(columns = columnNames)
i=0
count=1
for dirname, _, filenames in os.walk('F:/Open CV/Learn1/Train5/test/'):
    for filename in filenames:
        image_name = os.path.join(dirname, filename)
        #print(os.path.join(dirname, filename))
        label = dirname[dirname.rindex('/')+1:]
        #print(label)
        img = Image.open(image_name)
        rawData = img.load()
        #print(rawData)
        data = []
        data.append(label)
        for y in range(28):
            for x in range(28):
                data.append(rawData[x,y])
        k = 0
        test_data.loc[i] = [data[k] for k in range(785)]
        i = i+1
    print(count)
    count = count + 1


test_data.to_csv("test.csv",index=False)
```

```
train_data.to_csv('train.csv',index=False)
train = train_data
test = test_data

Y_train = train["label"]
Y_train = Y_train.astype(int)
Y_train = Y_train - 1

Y_test = test["label"]
Y_test = Y_test.astype(int)
Y_test = Y_test - 1

X_train = train.drop(labels = ["label"],axis = 1)
X_train = X_train / 255.0
X_test = test.drop(labels = ["label"],axis = 1)
X_test = X_test / 255.0
X_train = X_train.values.reshape(-1,28,28,1)
X_test = X_test.values.reshape(-1,28,28,1)
Y_train = to_categorical(Y_train, num_classes = 6)
Y_test = to_categorical(Y_test, num_classes = 6)

datagen = ImageDataGenerator(
        rotation_range=10,
        zoom_range = 0.10,
        width_shift_range=0.1,
        height_shift_range=0.1)

model = Sequential()
model.add(Conv2D(32, kernel_size = 3, activation='relu', input_shape = (28, 28,
1)))
model.add(BatchNormalization())
model.add(Conv2D(32, kernel_size = 3, activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(32, kernel_size = 5, strides=2, padding='same',
activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.4))

model.add(Conv2D(64, kernel_size = 3, activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(64, kernel_size = 3, activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(64, kernel_size = 5, strides=2, padding='same',
activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.4))

model.add(Conv2D(128, kernel_size = 4, activation='relu'))
```

```
model.add(BatchNormalization())
model.add(Flatten())
model.add(Dropout(0.4))
model.add(Dense(6, activation='softmax'))

# COMPILE WITH ADAM OPTIMIZER AND CROSS ENTROPY COST
model.compile(optimizer="adam", loss="categorical_crossentropy",
metrics=["accuracy"])

# Y_train = np.array.reshape(-1,28,28,1)


X_train2, X_test, Y_train2, Y_test = train_test_split(X_train, Y_train, test_size =
0.3)
annealer = LearningRateScheduler(lambda x: 1e-3 * 0.95 ** x)

history = model.fit_generator(datagen.flow(X_train2,Y_train2, batch_size=64),
                              epochs = 30, steps_per_epoch = X_train2.shape[0]//64,
                              validation_data=datagen.flow(X_test,Y_test))

model.save('64x3-CNN.model')
model.summary()

# import matplotlib.pyplot as plt
# plt.plot(history.history['accuracy'], label='accuracy')
# plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
# plt.xlabel('Epoch')
# plt.ylabel('Accuracy')
# plt.ylim([0.5, 1])
# plt.legend(loc='lower right')

scores = model.evaluate(X_test, Y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))

test_y = np.argmax(model.predict(X_test),axis =1)

rows = 5
cols = 10

f = plt.ure(size=(2*cols,2*rows))

for i in range(rows*cols):
    f.add_subplot(rows,cols,i+1)

    plt.imshow(X_test.reshape([-1,28,28,1]),cmap="Blues")

    plt.axis("off")
    plt.title(str(test_y[i]))
```