LAMPIRAN

```
#tahap download file google drive ke server collab
!pip install -U -q PyDrive
import os
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

# 1. Authenticate and create the PyDrive client.
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)

# choose a local (colab) directory to store the data.
local_download_path = os.path.expanduser('~/dataset')
try:
  os.makedirs(local_download_path)
except: pass
# 2. Auto-iterate using the query syntax
#    https://developers.google.com/drive/v2/web/search-parameters
file_list = drive.ListFile(
    {'q': "'1-SI64twIlJHFoh9wyHeBpIUUewUbziA0' in parents"}).GetList()

for f in file_list:
  # 3. Create & download by id.
```

```
print('title: %s, id: %s' % (f['title'], f['id']))
fname = os.path.join(local_download_path, f['title'])
print('downloading to {}'.format(fname))
f_ = drive.CreateFile({'id': f['id']})
f_.GetContentFile(fname)

#tahap extract file ke collab server
import os
import zipfile
#create folder
PATH = os.path.expanduser('~/dataset')
# os.chdir(PATH)
# !ls
try:
  os.makedirs(PATH + '/data/baru')
except: pass
zip_ref = zipfile.ZipFile(PATH + '/DATA FULL OF DATA.zip', 'r')
zip_ref.extractall(PATH + '/data/baru')
zip_ref.close()
print("done")
os.chdir(PATH + '/data/baru')
!ls
 # Import libraries
import os,cv2
import numpy as np
import matplotlib.pyplot as plt
from sklearn.utils import shuffle
from sklearn.cross_validation import train_test_split
from keras import backend as K
```

```
K.set_image_dim_ordering('th')

from keras.utils import np_utils
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation, Flatten
from keras.layers.convolutional import Convolution2D, MaxPooling2D
from keras.optimizers import SGD,RMSprop,adam
from keras.models import load_model
from keras.layers import Dense, Activation, Dropout, Flatten,Conv2D, MaxPooling2D
from keras.layers.normalization import BatchNormalization
PATH = os.path.expanduser('~/dataset/data/baru/')

# Define data path
data_path = PATH
data_dir_list = sorted(os.listdir(data_path))

img_rows=28
img_cols=28
num_channel=1
num_epoch=50

# Define the number of classes
num_classes = 62
img_data_list=[]
for dataset in data_dir_list:
        img_list=os.listdir(data_path+'/'+ dataset)
        print ('Loaded the images of dataset-'+'{}\n'.format(dataset))
        for img in img_list:
```

```
            input_img=cv2.imread(data_path + '/'+ dataset + '/'+ img )#membaca semua image
            input_img=cv2.cvtColor(input_img, cv2.COLOR_BGR2GRAY)#buat jdi grayscale
            input_img_resize=cv2.resize(input_img,(28,28))#mengubah image gambar jadi 28*28
            img_data_list.append(input_img_resize)#menambah huruf lanjutan contoh ada ABC diappand D jdi ABCD

img_data = np.array(img_data_list)#mengubah jdi list/array
img_data = img_data.astype('float64')#mengubah fromat float jdi float64
img_data /= 255#dibagi dengan 255
print (img_data.shape)
if num_channel==1:
    if K.image_dim_ordering()=='th':
            img_data= np.expand_dims(img_data, axis=1)
            print (img_data.shape)
    else:
            img_data= np.expand_dims(img_data, axis=4)
            print (img_data.shape)

else:
    if K.image_dim_ordering()=='th':
            img_data=np.rollaxis(img_data,3,1)
            print (img_data.shape)
#%%
USE_SKLEARN_PREPROCESSING=False

if USE_SKLEARN_PREPROCESSING:
```

```
# using sklearn for preprocessing
from sklearn import preprocessing
def image_to_feature_vector(image, size=(28, 28)):
        # resize the image to a fixed size, then flatten the image into
        # a list of raw pixel intensities
        return cv2.resize(image, size).flatten()
img_data_list=[]
for dataset in data_dir_list:
        img_list=os.listdir(data_path+'/'+ dataset)
        print ('Loaded the images of dataset-'+'{}\n'.format(dataset))
        for img in img_list:
                input_img=cv2.imread(data_path + '/'+ dataset + '/'+ img )
                input_img=cv2.cvtColor(input_img,
cv2.COLOR_BGR2GRAY)
        input_img_flatten=image_to_feature_vector(input_img,(28,28))
                img_data_list.append(input_img_flatten)
img_data = np.array(img_data_list)
img_data = img_data.astype('float64')
print (img_data.shape)
img_data_scaled = preprocessing.scale(img_data)
print (img_data_scaled.shape)
print (np.mean(img_data_scaled))
print (np.std(img_data_scaled))
print (img_data_scaled.mean(axis=0))
print (img_data_scaled.std(axis=0))
if K.image_dim_ordering()=='th':
img_data_scaled=img_data_scaled.reshape(img_data.shape[0],num_channel,i
mg_rows,img_cols)
        print (img_data_scaled.shape)
```

```
        else:
        img_data_scaled=img_data_scaled.reshape(img_data.shape[0],img_rows,img
  _cols,num_channel)
                print (img_data_scaled.shape)
        if K.image_dim_ordering()=='th':
        img_data_scaled=img_data_scaled.reshape(img_data.shape[0],num_channel,i
  mg_rows,img_cols)
                print (img_data_scaled.shape)
        else:
        img_data_scaled=img_data_scaled.reshape(img_data.shape[0],img_rows,img
_cols,num_channel)
                print (img_data_scaled.shape)
  if USE_SKLEARN_PREPROCESSING:
        img_data=img_data_scaled
  #%%
  # Assigning Labels
  #TOTAL KELAS
  num_classes = 62
  num_of_samples = img_data.shape[0]
  labels = np.ones((num_of_samples,),dtype='int64')

  #PEMBERIAN LABEL
  labels[0:6015]=0      #0
  labels[6015:10646]=1        #1
  labels[10646:16660]=2       #2
  labels[16660:22668]=3       #3
  labels[22668:28682]=4       #4
  labels[28682:34689]=5       #5
```

```
labels[34689:40704]=6       #6
labels[40704:46711]=7       #7
labels[46711:52727]=8       #8
labels[52727:58740]=9       #9
labels[58740:63256]=10      #A
labels[63256:67272]=11      #B
labels[67272:73248]=12      #C
labels[73248:77264]=13      #D
labels[77264:81780]=14      #E
labels[81780:87795]=15      #F
labels[87795:90311]=16      #G
labels[90311:93327]=17      #H
labels[93327:96707]=18      #I
labels[96707:100706]=19     #J
labels[100706:103622]=20    #K
labels[103622:108404]=21    #L
labels[108404:114418]=22    #M
labels[114418:120431]=23    #N
labels[120431:126407]=24    #O
labels[126407:132422]=25    #P
labels[132422:135437]=26    #Q
labels[135437:140453]=27    #R
labels[140453:146463]=28    #S
labels[146463:152470]=29    #T
labels[152470:158474]=30    #U
labels[158474:163280]=31    #V
labels[163280:168196]=32    #W
labels[168196:171312]=33    #X
labels[171312:176226]=34    #Y
```

labels[176226:179242]=35 #Z
labels[179242:185254]=36 #aa
labels[185254:190668]=37 #bb
labels[190668:194184]=38 #cc
labels[194184:200197]=39 #dd
labels[200197:206213]=40 #ee
labels[206213:208629]=41 #ff
labels[208629:213118]=42 #gg
labels[213118:219129]=43 #hh
labels[219129:221103]=44 #ii
labels[221103:223566]=45 #jj
labels[223566:226682]=46 #kk
labels[226682:230665]=47 #ll
labels[230665:233481]=48 #mm
labels[233481:238492]=49 #nn
labels[238492:241208]=50 #oo
labels[241208:243624]=51 #pp
labels[243624:247140]=52 #qq
labels[247140:253128]=53 #rr
labels[253128:255744]=54 #ss
labels[255744:261754]=55 #tt
labels[261754:264570]=56 #uu
labels[264570:267786]=57 #vv
labels[267786:270901]=58 #ww
labels[270901:274317]=59 #xx
labels[274317:277230]=60 #yy
labels[277230:280372]=61 #zz

```
# convert class labels to on-hot encoding
Y = np_utils.to_categorical(labels, num_classes)
#Shuffle the dataset
x,y = shuffle(img_data,Y, random_state=2)
# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=2)
# Defining the model
input_shape=img_data[0].shape
model = Sequential()
model.add(Convolution2D(32, 5,5,border_mode='same',input_shape=input_shape))
model.add(Activation('relu'))
model.add(Convolution2D(32, 5, 5))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))
model.add(Convolution2D(64, 5, 5))
model.add(Activation('relu'))
#model.add(Convolution2D(64, 5, 5))
#model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes))
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy',
optimizer='adam',metrics=["accuracy"])
```

```
# Viewing model_configuration
model.summary()
model.get_config()
model.layers[0].get_config()
model.layers[0].input_shape
model.layers[0].output_shape
model.layers[0].get_weights()
np.shape(model.layers[0].get_weights()[0])
model.layers[0].trainable
# Training
hist = model.fit(X_train, y_train, batch_size=16, nb_epoch=num_epoch, verbose=1,
validation_data=(X_test, y_test))
# visualizing losses and accuracy
train_loss=hist.history['loss']
val_loss=hist.history['val_loss']
train_acc=hist.history['acc']
val_acc=hist.history['val_acc']
xc=range(num_epoch)
#menampilkan grafik loss
plt.figure(1,figsize=(7,5))
plt.plot(xc,train_loss)
plt.plot(xc,val_loss)
plt.xlabel('num of Epochs')
plt.ylabel('loss')
plt.title('train_loss vs val_loss')
plt.grid(True)
plt.legend(['train','val'])
plt.style.use(['classic'])
```

```
#menampilkan grafik akurasi
plt.figure(2,figsize=(7,5))
plt.plot(xc,train_acc)
plt.plot(xc,val_acc)
plt.xlabel('num of Epochs')
plt.ylabel('accuracy')
plt.title('train_acc vs val_acc')
plt.grid(True)
plt.legend(['train','val'],loc=4)
plt.style.use(['classic'])
#SIMPAN KE FILE MODEL HDF5
#%%
# Saving and loading model and weights
from keras.models import model_from_json
from keras.models import load_model
try:
  os.makedirs(os.path.expanduser('~/dataset/data/data1'))
except: pass
lokasi = os.path.expanduser('~/dataset/data/data1')
# serialize model to JSON
model_json = model.to_json()
with open(lokasi + "/model.json", "w") as json_file:
    json_file.write(model_json)
# serialize weights to HDF5
model.save_weights(lokasi + "/model.h5")
print("Saved model to disk")
# load json and create model
json_file = open(lokasi + '/model.json', 'r')
```

```
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
# load weights into new model
loaded_model.load_weights(lokasi + "/model.h5")
print("Loaded model from disk")
model.save(lokasi + '/model2.hdf5')
loaded_model=load_model(lokasi + '/model2.hdf5')

# Install the PyDrive wrapper & import libraries.
# This only needs to be done once in a notebook.
!pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials
# Authenticate and create the PyDrive client.
# This only needs to be done once in a notebook.
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)

# Create & upload a file.
uploaded = drive.CreateFile({'title': 'model2.hdf5'})
uploaded.SetContentFile(lokasi + '/model2.hdf5')
uploaded.Upload()
print('Uploaded file with ID {}'.format(uploaded.get('id')))
```