(1) Wh | Google | IBM Cl | Model | about:l | about:l | IBM-Pr | Welcor | IBM | IBM-Pr | AI-B9-

eu-de.dataplatform.cloud.ibm.com/analytics/notebooks/v2/38b9b3de-0eef-4e4c-8560-551f52caf440/view?projectid=d9a66470-61f0-47f6-b46...

IBM Watson Studio | Search in your workspaces | Buy | Maharaja S's Account | Frankfurt | MS

Projects / A Gesture-based sterile browsin... / Model Building

In [1]:
```
from keras.preprocessing.image import ImageDataGenerator
```

In [6]:
```
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='JyZHr10lN5oaoubH3rck89IXO8KIpzjkRVVNMmXEAl0s',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.eu.cloud-object-storage.appdomain.cloud')

bucket = 'agesturebasedsterilebrowsingofrad-donotdelete-pr-himk3rbb4iwae7'
object_key = 'dataset.zip'

streaming_body_1 = cos_client.get_object(Bucket=bucket, Key=object_key)['Body']

# Your data file was loaded into a botocore.response.StreamingBody object.
# Please read the documentation of ibm_boto3 and pandas to learn more about the possibilities to load the data.
# ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/
# pandas documentation: http://pandas.pydata.org/
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3
```

TapScanner 11-1....pdf | Model Building....ipynb | Model Building.ipynb | dataset (1).zip | Show All

IBM Watson Studio | Search in your workspaces | Buy | Maharaja S's Account | Frankfurt

Projects / A Gesture-based sterile browsin... / Model Building

```
# Please read the documentation of ibm_boto3 and pandas to learn more about the possibilities to load the data.
# ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/
# pandas documentation: http://pandas.pydata.org/
```

In [7]:
```
from io import BytesIO
import zipfile
unzip = zipfile.ZipFile(BytesIO(streaming_body_1.read()),'r')
file_paths = unzip.namelist()
for path in file_paths:
    unzip.extract(path)
```

In [8]:
```
ls
```

```
dataset/
```

In [9]:
```
import numpy as np
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense,Flatten
from tensorflow.keras.layers import Conv2D,MaxPooling2D
from keras.preprocessing.image import ImageDataGenerator
```

In [10]:
```
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
```

In [11]:
```
test_datagen=ImageDataGenerator(rescale=1./255)
```

In [13]:
```
x_train=train_datagen.flow_from_directory(r'dataset/train',
                                          target_size=(64,64),
                                          batch_size=5,
                                          color_mode='grayscale',
                                          class_mode='categorical')
```

TapScanner 11-1....pdf | Model Building....ipynb | Model Building.ipynb | dataset (1).zip | Show All

```
                                                color_mode='grayscale',
                                                class_mode='categorical')
```

```
Found 594 images belonging to 6 classes.
```

In [14]:
```
x_test = test_datagen.flow_from_directory(r'dataset/test',
                                          target_size=(64, 64),
                                          batch_size=5,
                                          color_mode='grayscale',
                                          class_mode='categorical')
```

```
Found 30 images belonging to 6 classes.
```

In [15]:
```
print(x_train.class_indices)
```

```
{'0': 0, '1': 1, '2': 2, '3': 3, '4': 4, '5': 5}
```

In [17]:
```
model=Sequential()
```

In [19]:
```
model.add(Conv2D(32,(3,3),input_shape=(64,64,1),activation='relu'))
```

In [20]:
```
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(32,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(units=128,activation='relu'))
model.add(Dense(units=6,activation='softmax'))
```

In [21]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
```

In [21]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        320

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 29, 29, 32)        9248

 max_pooling2d_1 (MaxPooling  (None, 14, 14, 32)       0
 2D)

 flatten (Flatten)           (None, 6272)              0

 dense (Dense)               (None, 128)               802944

 dense_1 (Dense)             (None, 6)                 774

=================================================================
Total params: 813,286
Trainable params: 813,286
Non-trainable params: 0
_________________________________________________________________
```

In [22]:
```
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
```

In [23]:
```
model.fit_generator(
    generator=x_train,steps_per_epoch=len(x_train),
    epochs=20,validation_data=x_test,validation_steps=len(x_test))
```

In [22]:
```
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
```

In [23]:
```
model.fit_generator(
    generator=x_train,steps_per_epoch=len(x_train),
    epochs=20,validation_data=x_test,validation_steps=len(x_test))
```

```
/tmp/wsuser/ipykernel_164/1772156357.py:1: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use
`Model.fit`, which supports generators.
  model.fit_generator(
Epoch 1/20
119/119 [==============================] - 6s 45ms/step - loss: 1.5038 - accuracy: 0.3872 - val_loss: 1.0099 - val_accuracy: 0.5667
Epoch 2/20
119/119 [==============================] - 5s 43ms/step - loss: 0.7806 - accuracy: 0.6869 - val_loss: 0.8101 - val_accuracy: 0.6000
Epoch 3/20
119/119 [==============================] - 5s 44ms/step - loss: 0.5088 - accuracy: 0.7997 - val_loss: 0.4645 - val_accuracy: 0.8000
Epoch 4/20
119/119 [==============================] - 5s 44ms/step - loss: 0.4513 - accuracy: 0.8401 - val_loss: 0.5337 - val_accuracy: 0.8000
Epoch 5/20
119/119 [==============================] - ETA: 0s - loss: 0.3116 - accuracy: 0.88 - 5s 42ms/step - loss: 0.3116 - accuracy: 0.8872 - val_loss:
0.6007 - val_accuracy: 0.7333
Epoch 6/20
119/119 [==============================] - 5s 42ms/step - loss: 0.2729 - accuracy: 0.9024 - val_loss: 0.2647 - val_accuracy: 0.8667
Epoch 7/20
119/119 [==============================] - 5s 43ms/step - loss: 0.2088 - accuracy: 0.9310 - val_loss: 0.3697 - val_accuracy: 0.9333
Epoch 8/20
119/119 [==============================] - 5s 42ms/step - loss: 0.1797 - accuracy: 0.9495 - val_loss: 0.2751 - val_accuracy: 0.9333
Epoch 9/20
119/119 [==============================] - 5s 43ms/step - loss: 0.1330 - accuracy: 0.9495 - val_loss: 0.2462 - val_accuracy: 0.9000
Epoch 10/20
119/119 [==============================] - 5s 42ms/step - loss: 0.1351 - accuracy: 0.9461 - val_loss: 0.2363 - val_accuracy: 0.9000
Epoch 11/20
119/119 [==============================] - 5s 42ms/step - loss: 0.1264 - accuracy: 0.9529 - val_loss: 0.2328 - val_accuracy: 0.8667
```

```
Epoch 13/20
119/119 [==============================] - 5s 43ms/step - loss: 0.0858 - accuracy: 0.9731 - val_loss: 0.2426 - val_accuracy: 0.9333
Epoch 14/20
119/119 [==============================] - 5s 42ms/step - loss: 0.1069 - accuracy: 0.9663 - val_loss: 0.1280 - val_accuracy: 0.9667
Epoch 15/20
119/119 [==============================] - 5s 42ms/step - loss: 0.0704 - accuracy: 0.9798 - val_loss: 0.4321 - val_accuracy: 0.9000
Epoch 16/20
119/119 [==============================] - 5s 44ms/step - loss: 0.1195 - accuracy: 0.9562 - val_loss: 0.1727 - val_accuracy: 0.9333
Epoch 17/20
119/119 [==============================] - 5s 41ms/step - loss: 0.0448 - accuracy: 0.9865 - val_loss: 0.1533 - val_accuracy: 0.9333
Epoch 18/20
119/119 [==============================] - 5s 42ms/step - loss: 0.0933 - accuracy: 0.9764 - val_loss: 0.0669 - val_accuracy: 0.9667
Epoch 19/20
119/119 [==============================] - 5s 43ms/step - loss: 0.0375 - accuracy: 0.9865 - val_loss: 0.0678 - val_accuracy: 0.9667
Epoch 20/20
119/119 [==============================] - 5s 42ms/step - loss: 0.0624 - accuracy: 0.9798 - val_loss: 0.1029 - val_accuracy: 0.9333
```

Out[23]:
```
<keras.callbacks.History at 0x7fb784beae50>
```

In [24]:
```
model.save('gesture.h5')
```

In [25]:
```
!tar -zcvf image-classification-model_new.tgz gesture.h5
```

```
gesture.h5
```

In [26]:
```
ls-1
```

```
dataset/
gesture.h5
image-classification-model_new.tgz
```

In [27]:
```
!pip install watson-machine-learning-client --upgrade
```

```
Collecting watson-machine-learning-client
  Downloading watson_machine_learning_client-1.0.391-py3-none-any.whl (538 kB)
```