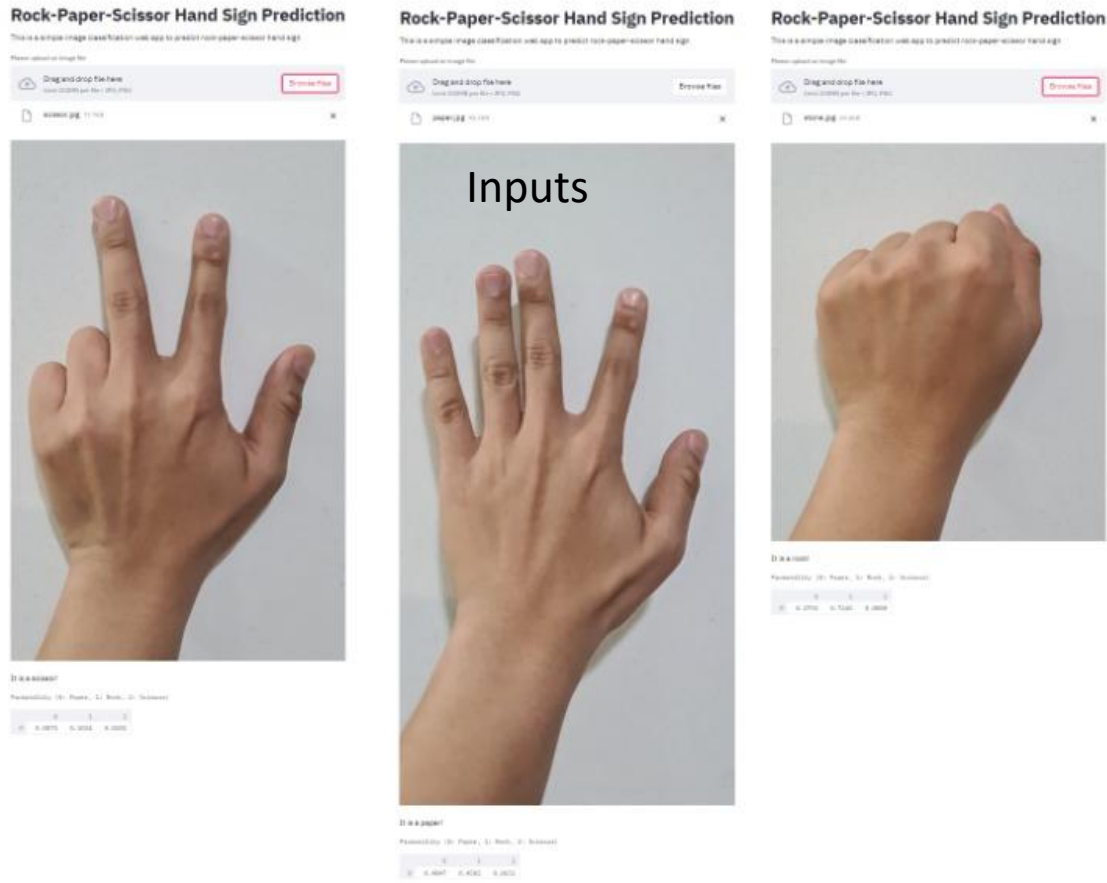


Exercise:

Rock-Paper-Scissor Image Classification



Predict <Using Streamlit UI>

```
Anaconda Prompt (anaconda3) - streamlit run rps_app.py
(base) D:\rps>
(base) D:\rps>python rps_app_model.py
2021-01-01 21:35:33.196500: W tensorflow/stream_executor/platform/default/dso
dLError: cudart64_110.dll not found
2021-01-01 21:35:33.196712: I tensorflow/stream_executor/cuda/cudart_stub.cc:
n your machine.
Found 2142 images belonging to 3 classes.
Found 378 images belonging to 3 classes.
Found 33 images belonging to 3 classes.
2021-01-01 21:35:36.010128: I tensorflow/compiler/jit/xla_cpu_device.cc:41] N
2021-01-01 21:35:36.020307: I tensorflow/stream_executor/platform/default/dso
2021-01-01 21:35:36.047411: I tensorflow/core/common_runtime/gpu/gpu_device.c
pciBusID: 0000:01:00.0 name: GeForce GTX 1070 computeCapability: 6.1
coreClock: 1.7216Hz coreCount: 15 deviceMemorySize: 8.006GiB deviceMemoryBandw
2021-01-01 21:35:36.049622: W tensorflow/stream_executor/platform/default/dso
dLError: cudart64_110.dll not found
2021-01-01 21:35:36.051578: W tensorflow/stream_executor/platform/default/dso
dLError: cublas64_11.dll not found
2021-01-01 21:35:36.053446: W tensorflow/stream_executor/platform/default/dso
dLError: cublaslt64_11.dll not found
2021-01-01 21:35:36.057836: I tensorflow/stream_executor/platform/default/dso
2021-01-01 21:35:36.059344: I tensorflow/stream_executor/platform/default/dso
2021-01-01 21:35:36.063930: I tensorflow/stream_executor/platform/default/dso
2021-01-01 21:35:36.066138: W tensorflow/stream_executor/platform/default/dso
dLError: cusparse64_11.dll not found
2021-01-01 21:35:36.068612: W tensorflow/stream_executor/platform/default/dso
dLError: cudnn64_8.dll not found
2021-01-01 21:35:36.068764: W tensorflow/core/common_runtime/gpu/gpu_device.c
using libraries mentioned above are installed properly if you would like to u
s for how to download and setup the required libraries for your platform.
Skipping registering GPU devices...
2021-01-01 21:35:36.069082: I tensorflow/core/platform/cpu_feature_guard.cc:14
check (library (cudnn64)) to use the following CPU instructions in performance
Python
Python Program
rps_app.py - D:\rps\python (3.8.8)
File Edit Format Run Options Window Help
import numpy as np
import streamlit as st
import tensorflow as tf
from PIL import Image, ImageOps

def import_and_predict(image_data, model):
    size = (75,75)
    image = ImageOps.fit(image_data, size, Image.ANTIALIAS)
    image = image.convert('RGB')
    image = np.asarray(image)
    image = image.astype(np.float32) / 255.0

    img_reshape = image[np.newaxis,...]

    prediction = model.predict(img_reshape)

    return prediction

model = tf.keras.models.load_model('D:/rps/my_model1.hdf5')

st.write("""
# Rock-Paper-Scissor Hand Sign Prediction
""")

st.write("This is a simple image classification web app to predict rock-paper-scissor hand sign")

file = st.file_uploader("Please upload an image file", type=["jpg", "png"])

if file is None:
    st.text("You haven't uploaded an image file")
else:
    image = Image.open(file)
    st.image(image, use_column_width=True)
    prediction = import_and_predict(image, model)

    if np.argmax(prediction) == 0:
        st.write("It is a paper!")
    elif np.argmax(prediction) == 1:
        st.write("It is a rock!")
    else:
        st.write("It is a scissor!")

    st.text("Probability (0: Paper, 1: Rock, 2: Scissor)")
    st.write(prediction)
```

Learn ML-AI with Rock-Paper-Scissor Classification

Following are the steps involved in creating a well-defined ML project:

- Understand and define the problem statement
- Getting Rock-Paper-Scissor datasets
- Create a Python file to load the dataset
- Build and train the machine learning model
- Save the model
- Predict the result

Learn ML-AI with Rock-Paper-Scissor Classification

Problem Statement: Predicting the user's hand gestures (Rock-Paper-Scissor) from the trained model

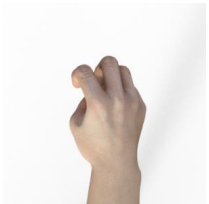


The model for the web app was trained with the rps dataset from Laurence Moroney. This is a synthetic dataset of computer generated human hands (left) with different colors that formed either rock, paper, or scissor.

Source: https://github.com/marcellusruben/rock_paper_scissor_web_app

Rock Paper Scissors Dataset (Overview)

- Rock Paper Scissors is a dataset containing 2,892 images of diverse hands in Rock/Paper/Scissors poses.
- Rock Paper Scissors contains images from a variety of different hands, from different races, ages and genders, posed into Rock / Paper or Scissors and labelled as such.



Rock



Paper



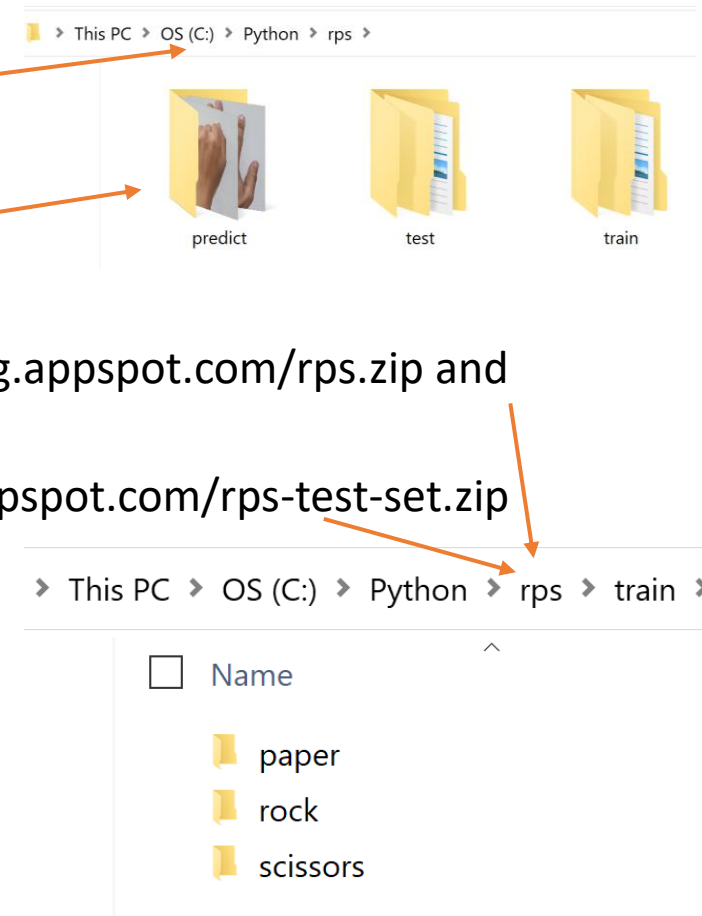
Scissor

[Credit: Rock Paper Scissors Dataset – Imoroney@ \(laurencemoroney.com\)](#)

- These images have all been generated using CGI techniques as an experiment in determining if a CGI-based dataset can be used for classification against real images.

Preparation of Dataset

1. Create a folder called “rps” in your laptop as “C:\Python\rps”.
2. Create sub folders called ‘train’, ‘test’ and ‘predict’ inside “C:\Python\rps”.
3. Download (alternative download the image datasets from blackboard)
 - the [training set](https://storage.googleapis.com/laurencemoroney-blog.appspot.com/rps.zip) at <https://storage.googleapis.com/laurencemoroney-blog.appspot.com/rps.zip> and unzip it into ‘train’ folder
 - the [test set](https://storage.googleapis.com/laurencemoroney-blog.appspot.com/rps-test-set.zip) at <https://storage.googleapis.com/laurencemoroney-blog.appspot.com/rps-test-set.zip> and unzip it into ‘test’ folder
4. Alternatively, you can download “rps.zip” file from Blackboard and unzip it into “C:\Python\rps”.
5. Launch the “IDLE” on windows search and click on the program to open.



Lab Experiment Setup

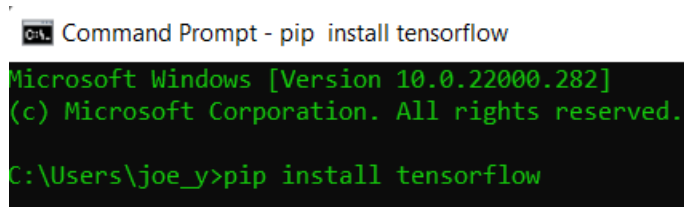
Requirements

- Python IDLE - 3.9.7 64bits (Recommended)

Python Package Installation

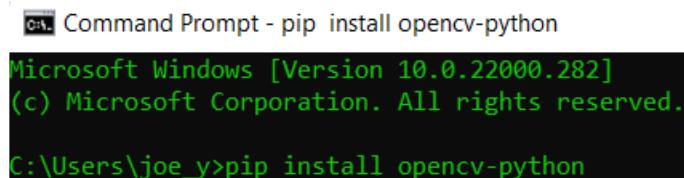
Open **Windows Command Prompt** as “Administrator”.

- Tensorflow (2.0 and above) → Type “pip install tensorflow” and install (est. 5 mins).



```
Command Prompt - pip install tensorflow
Microsoft Windows [Version 10.0.22000.282]
(c) Microsoft Corporation. All rights reserved.
C:\Users\joe_y>pip install tensorflow
```

- OpenCV → Type “pip install opencv-python” and install (est. 1-2 mins).



```
Command Prompt - pip install opencv-python
Microsoft Windows [Version 10.0.22000.282]
(c) Microsoft Corporation. All rights reserved.
C:\Users\joe_y>pip install opencv-python
```

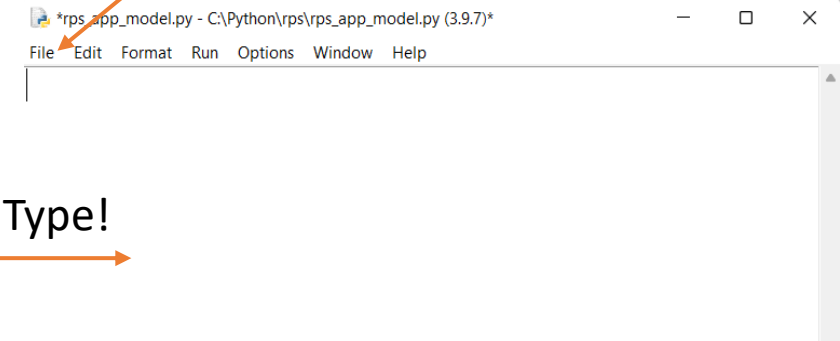
Getting started with Rock-Paper-Scissors Classification App

Importing libraries and getting iris datasets

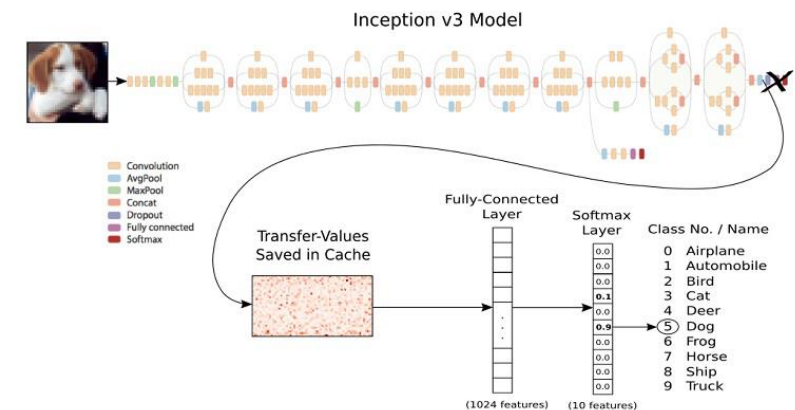
```
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.layers import Flatten, Dense, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
import tensorflow as tf
import os
```

Inception v3 is a widely-used image recognition model that has been shown to attain greater than 78.1% accuracy on the ImageNet dataset.

Open Python IDLE and create a new file.
Save the file as “C:\Python\rps\rps_app_model.py”.



Type!



Getting started with Rock-Paper-Scissors Classification App

```
def image_gen_w_aug(train_parent_directory, test_parent_directory):
```

```
    train_datagen = ImageDataGenerator(rescale=1/255,  
                                       rotation_range = 30,  
                                       zoom_range = 0.2,  
                                       width_shift_range=0.1,  
                                       height_shift_range=0.1,  
                                       validation_split = 0.15)
```

Image Augmentation

```
    test_datagen = ImageDataGenerator(rescale=1/255)
```

```
    train_generator = train_datagen.flow_from_directory(train_parent_directory,  
                                                       target_size = (75,75),  
                                                       batch_size = 214,  
                                                       class_mode = 'categorical',  
                                                       subset='training')
```

```
    val_generator = train_datagen.flow_from_directory(train_parent_directory,  
                                                     target_size = (75,75),  
                                                     batch_size = 37,  
                                                     class_mode = 'categorical',  
                                                     subset = 'validation')
```

```
    test_generator = test_datagen.flow_from_directory(test_parent_directory,  
                                                     target_size=(75,75),  
                                                     batch_size = 37,  
                                                     class_mode = 'categorical')
```

```
    return train_generator, val_generator, test_generator
```

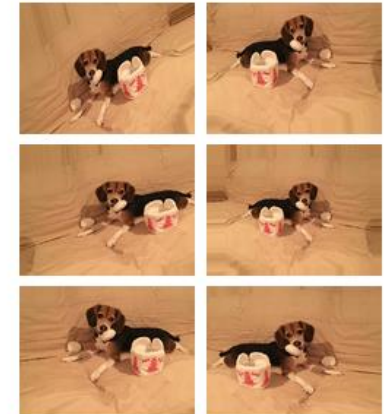
ImageDataGenerator (Keras)

https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

Input Image



Augmented Images



Getting started with Rock-Paper-Scissors Classification App

Dropout prevents overfitting and reduces training time!

Create `pre_trained_model` using InceptionV3 neural network with imagenet weight for image classification.

```
def model_output_for_TL (pre_trained_model, last_output):
```

```
    x = Flatten()(last_output)
```

```
    # Dense hidden layer
```

```
    x = Dense(512, activation='relu')(x)
```

```
    x = Dropout(0.2)(x)
```

```
    # Output neuron.
```

```
    x = Dense(3, activation='softmax')(x)
```

```
    model = Model(pre_trained_model.input, x)
```

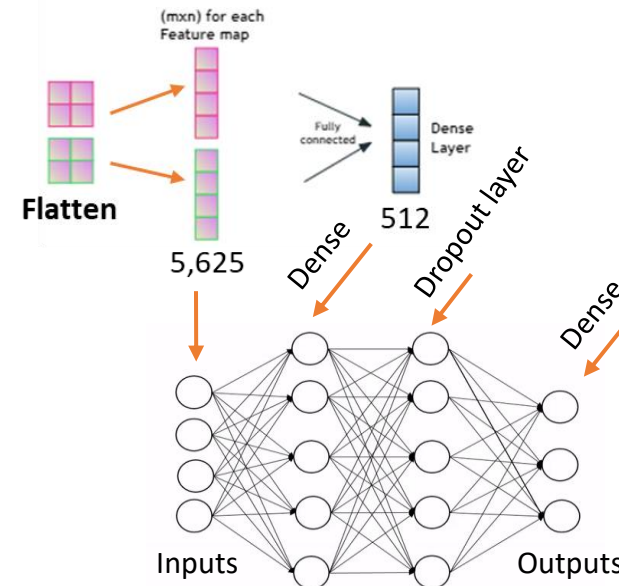
```
    return model
```

```
train_dir = os.path.join('C:/Python/rps/datasets/train/')
```

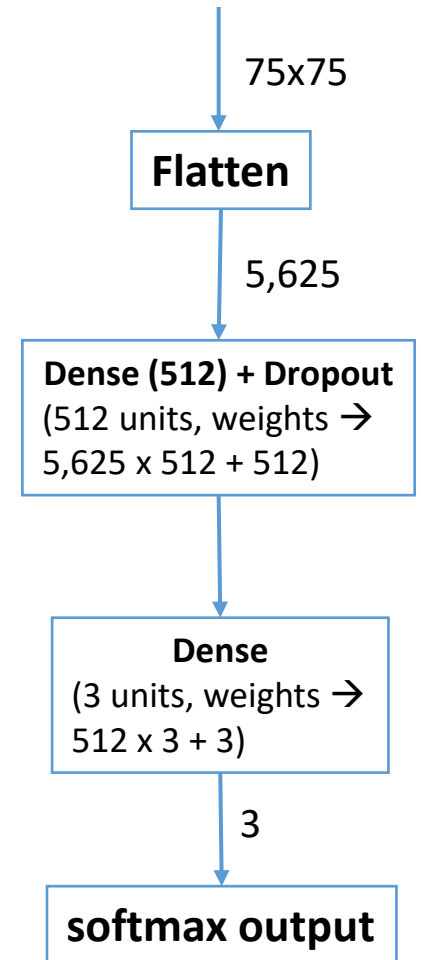
```
test_dir = os.path.join('C:/Python/rps/datasets/test/')
```

```
train_generator, validation_generator, test_generator = image_gen_w_aug(train_dir, test_dir)
```

```
pre_trained_model = InceptionV3(input_shape = (75, 75, 3),
                                include_top = False,
                                weights = 'imagenet')
```



Getting train and test datasets



Dense Neural Network

Getting started with Rock-Paper-Scissors Classification App

```
for layer in pre_trained_model.layers:
    layer.trainable = False      #freeze the layer, frozen layer won't be updated during training

last_layer = pre_trained_model.get_layer('mixed3') #define output as last_layer
last_output = last_layer.output

model_TL = model_output_for_TL(pre_trained_model, last_output)
model_TL.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

history_TL = model_TL.fit(
    train_generator,
    steps_per_epoch=10,
    epochs=10,
    verbose=1,
    validation_data = validation_generator)

tf.keras.models.save_model(model_TL, 'my_model.hdf5')
```

Creating and naming
last output layer

Setting the model
hyperparameters.

Google Search on
Hyperparameters!

Saving the model which
will be generated under
"C:\Python\rps\".

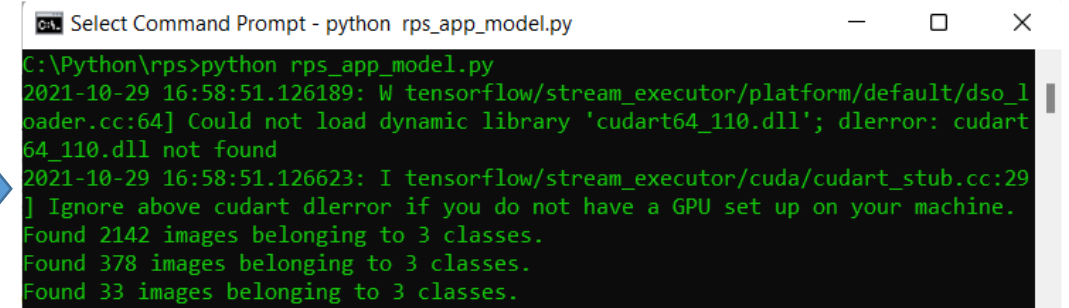
Remember to save the program as "C:\Python\rps\rps_app_model.py".

Running a real-time Rock-Paper-Scissors classification model training

Open Windows Command Prompt. (Recommended)
(Alternatively, you may use IDLE shell.)

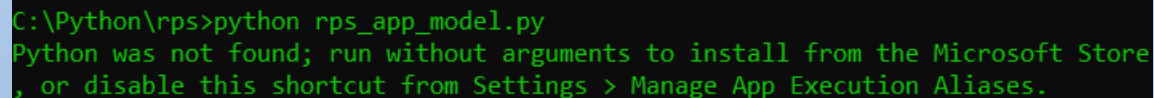
Then, type “cd /d C:\Python\rps” to go into “C:\Python\rps” folder.

Type “python rps_app_model.py” to run the model training.



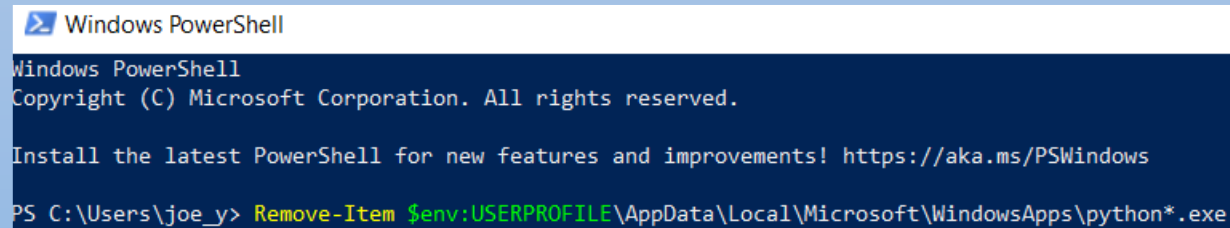
```
C:\Python\rps>python rps_app_model.py
2021-10-29 16:58:51.126189: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cuda64_110.dll'; dlderror: cuda64_110.dll not found
2021-10-29 16:58:51.126623: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
Found 2142 images belonging to 3 classes.
Found 378 images belonging to 3 classes.
Found 33 images belonging to 3 classes.
```

If there is an error, Python was not found...(below)



```
C:\Python\rps>python rps_app_model.py
Python was not found; run without arguments to install from the Microsoft Store
, or disable this shortcut from Settings > Manage App Execution Aliases.
```

Open Windows PowerShell, type “Remove-Item \$env:USERPROFILE\AppData\Local\Microsoft\WindowsApps\python*.exe”.



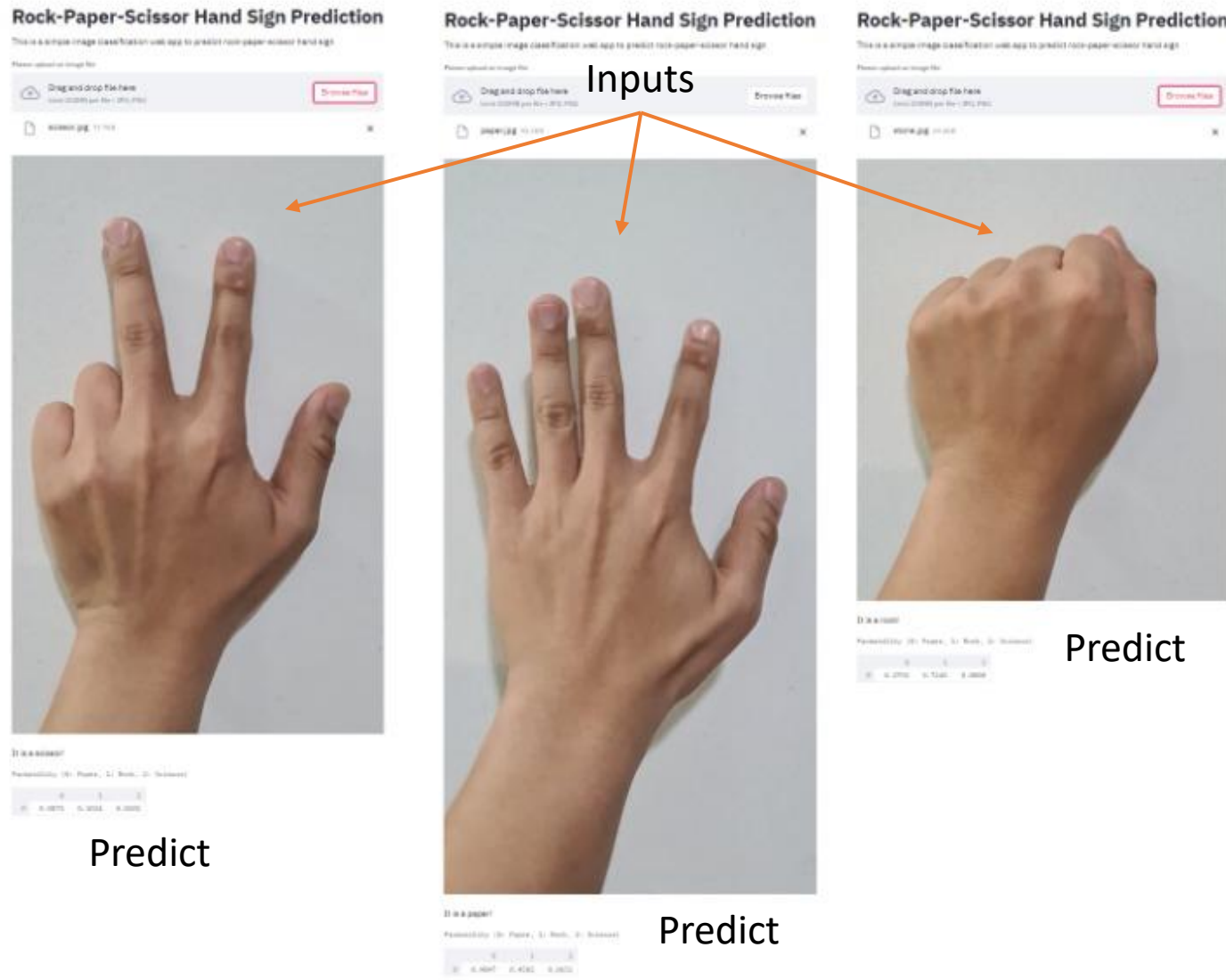
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\joe_y> Remove-Item $env:USERPROFILE\AppData\Local\Microsoft\WindowsApps\python*.exe
```

Re-run “python rps_app_model.py”.

Creating a Rock-Paper-Scissors classification app



Creating a Rock-Paper-Scissors classification app

1. Create a new python file under “C:\Python\rps\” and name it as “rps_app.py”.
2. Type or copy the code below to rps_app.py and save it.

```
import numpy as np
import streamlit as st
import tensorflow as tf
from PIL import Image, ImageOps
```

Import Libraries

```
def import_and_predict(image_data, model):
```

```
    size = (75,75) #set the image size
    image = ImageOps.fit(image_data, size, Image.ANTIALIAS) #prepare image with anti-aliasing
    image = image.convert('RGB') #convert image to RGB, Red Green Blue format
    image = np.asarray(image) #convert image into array
    image = (image.astype(np.float32) / 255.0) #create image array matrix
    img_reshape = image[np.newaxis,...] #np.newaxis will create new dimension
    prediction = model.predict(img_reshape) #give predicted output based on the input image
    return prediction #return predicted output
```

Import image and predict the output
(0:paper, 1:rock or 3:scissor)

```
model = tf.keras.models.load_model('C:/Python/rps/my_model.hdf5')
pre_trained_model = InceptionV3(input_shape = (75, 75, 3),
                                include_top = False,
                                weights = 'imagenet')
```

Load the trained model from 'C:/.../rps/mymodel.hdf5'

Creating a Rock-Paper-Scissors classification app

```
st.write("""
    # Rock-Paper-Scissor Hand Sign Prediction
    """)

st.write("This is a simple image classification web app to predict rock-paper-scissor hand sign")

file = st.file_uploader("Please upload an image file", type=["jpg", "png"])
#
if file is None:
    st.text("You haven't uploaded an image file")
else:
    image = Image.open(file)
    st.image(image, use_column_width=True)
    prediction = import_and_predict(image, model)

    if np.argmax(prediction) == 0:
        st.write("It is a paper!")
    elif np.argmax(prediction) == 1:
        st.write("It is a rock!")
    else:
        st.write("It is a scissor!")

    st.text("Probability (0: Paper, 1: Rock, 2: Scissor)")
    st.write(prediction)
```

Rock-Paper-Scissor Hand Sign Prediction

This is a simple image classification web app to predict rock-paper-scissor hand sign

Please upload an image file



Drag and drop file here

Limit 200MB per file • JPG, PNG

Browse files



paper.jpg 95.1KB



Create Streamlit UI Elements

It is a paper!

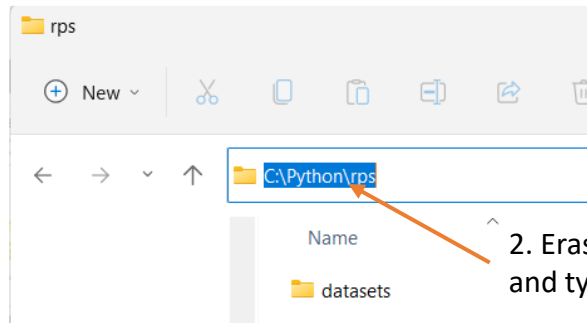
Probability (0: Paper, 1: Rock, 2: Scissor)

	0	1	2
0	0.4847	0.4502	0.0651

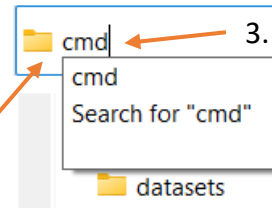
Predict output <Display on streamlit>
(1:rock, 2:paper or 3:scissor)

Running a Rock-Paper-Scissors classification app

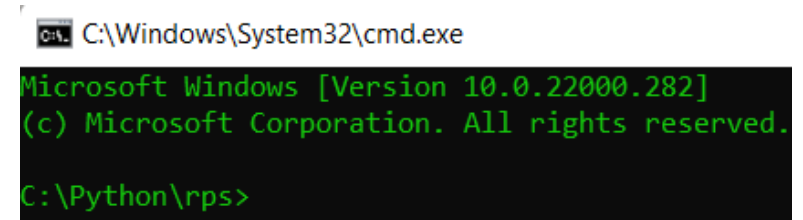
1. Open “C:\Python\rps” folder using Windows File Explorer.



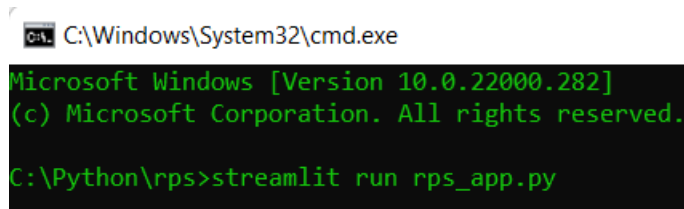
2. Erase 'C:\Python\rps' and type 'cmd'.



3. Hit Enter key.



4. Then, type “streamlit run rps_app.py” and hit Enter key to run the app.



5. Check

6. Click on “Allow access”.

Running a Rock-Paper-Scissors classification app

A browser will be launched with URL: localhost:8501



Rock-Paper-Scissor Hand Sign Prediction

This is a simple image classification web app to predict rock-paper-scissor hand sign

Please upload an image file



Drag and drop file here
Limit 200MB per file • JPG, PNG

Browse files

You haven't uploaded an image file

Click on "Browser files".

You may take the picture of your hand gesture (rock, paper and scissor) and upload to test the prediction.

Can you able to train and classify with different images?

Rock-Paper-Scissor Hand Sign Prediction

This is a simple image classification web app to predict rock-paper-scissor hand sign

Please upload an image file

Drag and drop file here
Limit 200MB per file • JPG, PNG

Browse files

paper.jpg 95.1KB



Predicted Output →

It is a paper!

Probability (0: Paper, 1: Rock, 2: Scissor)

	0	1	2
0	0.4847	0.4502	0.0651

Creating a real-time Rock-Paper-Scissors Classification App using OpenCV (Optional)

```
from PIL import Image, ImageOps
import tensorflow as tf
```

```
import cv2
import numpy as np
import os
import sys
```

Import libraries

Open Spyder IDE and Type!
Alternatively, you can open detect.py.
(downloadable from Blackboard)

```
label = ""
frame = None
```

```
def import_and_predict(image_data, model):
    size = (75,75) #set the image size
    image = ImageOps.fit(image_data, size, Image.ANTIALIAS) #prepare image with anti-aliasing
    image = image.convert('RGB') #convert image to RGB, Red Green Blue format
    image = np.asarray(image) #convert image into array
    image = (image.astype(np.float32) / 255.0) #create image array matrix
    img_reshape = image[np.newaxis,...] #np.newaxis will create new dimension
    prediction = model.predict(img_reshape) #give predicted output based on the input image
```

```
return prediction #return predicted output
```

detect.py

Create
import_and_predict
function and return
prediction

Creating a real-time Rock-Paper-Scissors Classification App using OpenCV (Optional)

```
model = tf.keras.models.load_model('C:/Python/rps/my_model.hdf5')  
  
cap = cv2.VideoCapture(0)  
  
if (cap.isOpened()):  
    print("Camera OK")  
else:  
    cap.open()  
  
while (True):  
    ret, original = cap.read()  
  
    frame = cv2.resize(original, (224, 224))  
    cv2.imwrite(filename='img.png', img=original)  
    image = Image.open('img.png')  
  
    # Display the predictions  
    prediction = import_and_predict(image, model)  
    #print(prediction)
```

Load model

Initialize OpenCV with webcam

Resize the image captured

detect.py

Creating a real-time Rock-Paper-Scissors Classification App using OpenCV (Optional)

```
if np.argmax(prediction) == 0:
    predict="It is a paper!"
elif np.argmax(prediction) == 1:
    predict="It is a rock!"
else:
    predict="It is a scissor!"

cv2.putText(original, predict, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)
cv2.imshow("Classification", original)

if (cv2.waitKey(1) & 0xFF == ord('q')):
    break;

cap.release()
frame = None
cv2.destroyAllWindows()
sys.exit()
```

Get the prediction (0: Paper, 1: Rock, 2: Scissor)

Display Image

Waiting for quit command word ('q').

Release image capturing, set frame to None and close OpenCV.

detect.py

Save 'detect.py' into 'C:\Python\rps' folder.

Running a real-time Rock-Paper-Scissors classification app

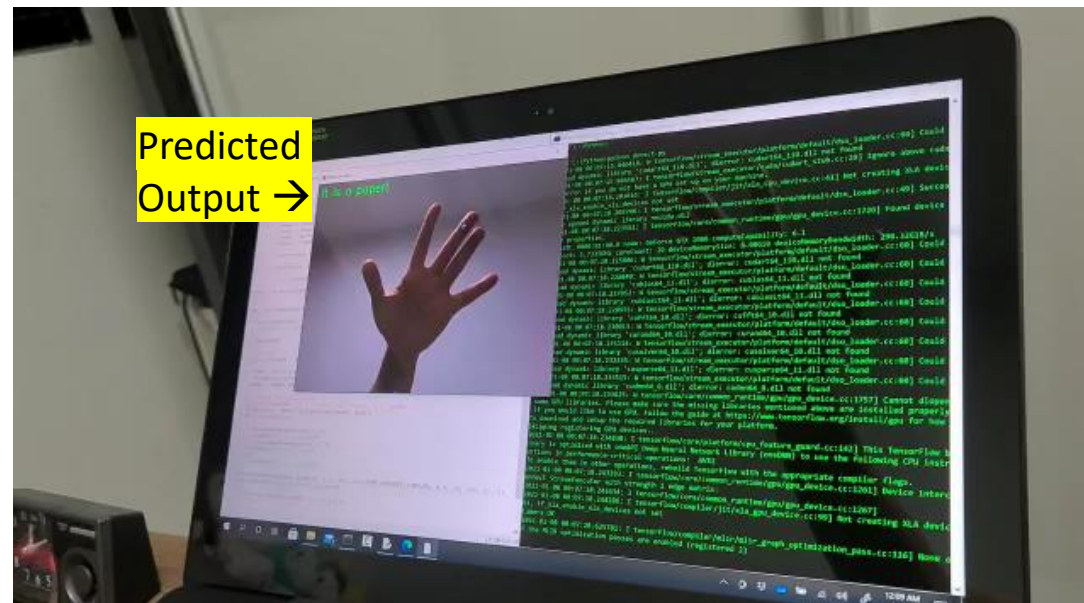
Open Command Prompt.

You will need to install OpenCV package for Python as follow.

Then, type “cd /d C:\Python\rps” to go into “C:\Python\rps” folder.

Type “python detect.py” to run the app.

Can you able to train and classify with different images?



Note: The ‘detect.py’ is also tested working also on Raspberry Pi with camera.