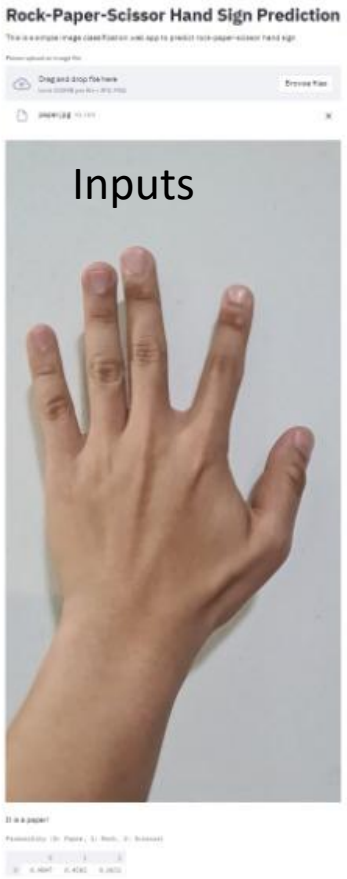
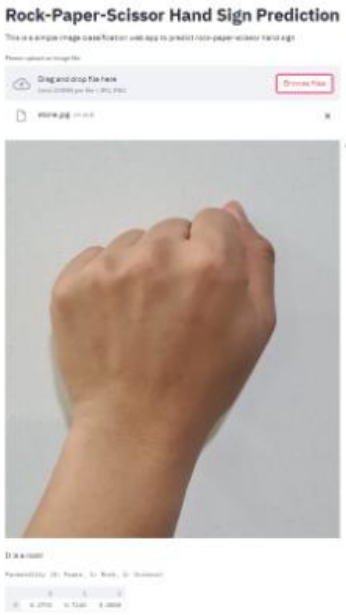
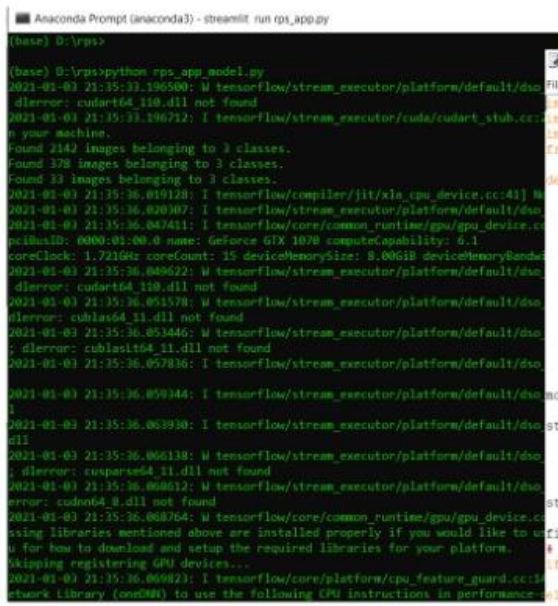
Exercise:
Rock-Paper-Scissor Image Classification

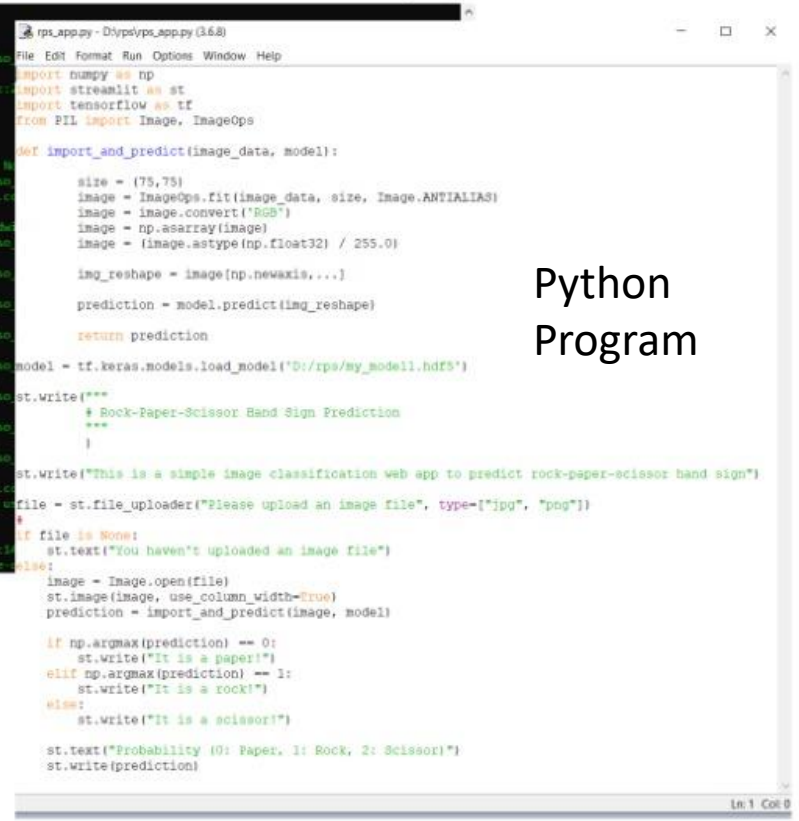




Predict <Using Streamlit UI>



Command (or Anaconda) Prompt



Python Program

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

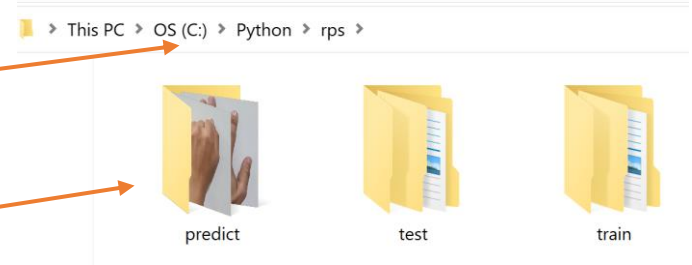
Credit: Rock Paper Scissors Dataset – lmoroney@ (laurencemoroney.com)

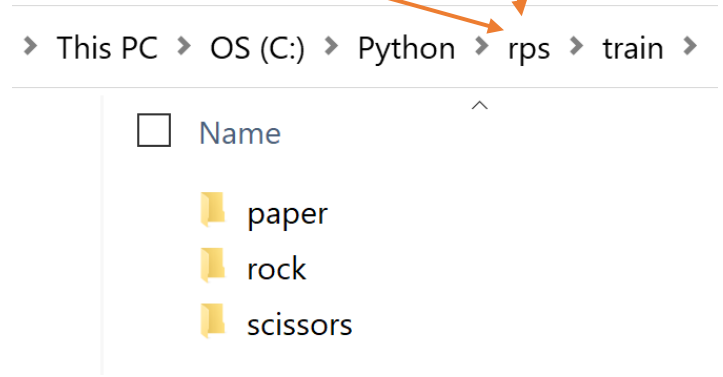
- These images have all been generated using CGI techniques as an experiment in determining if a CGI-based dataset can be used for classification against real images.

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 at https://storage.googleapis.com/laurencemoroney-blog.appspot.com/rps.zip and unzip it into ‘train’ folder
   - the test set 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

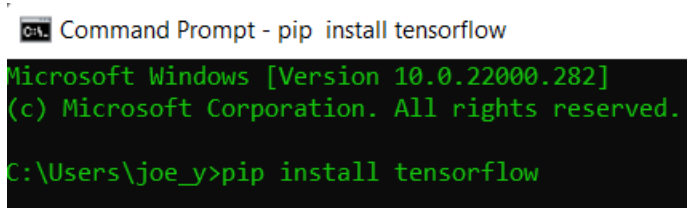
Requirements

- Python IDLE - 3.9.7 64bits (Recommended)

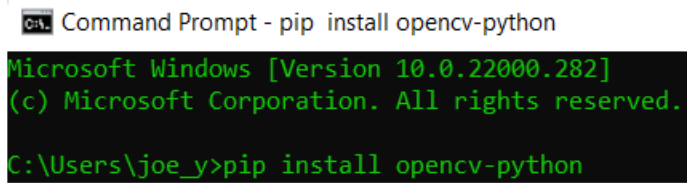
Python Package Installation

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

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



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



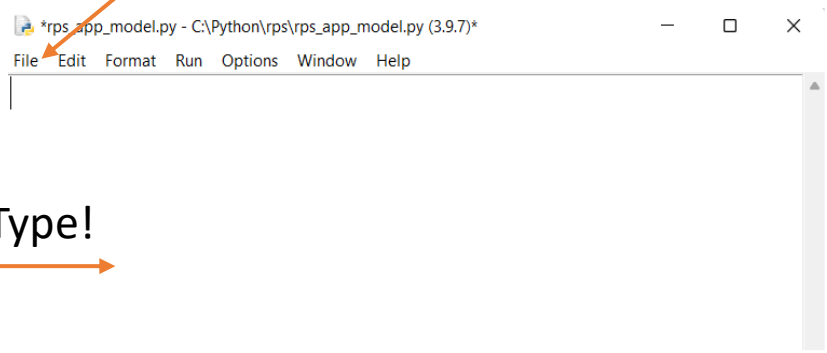
Getting started with Rock-Paper-Scissors Classification App

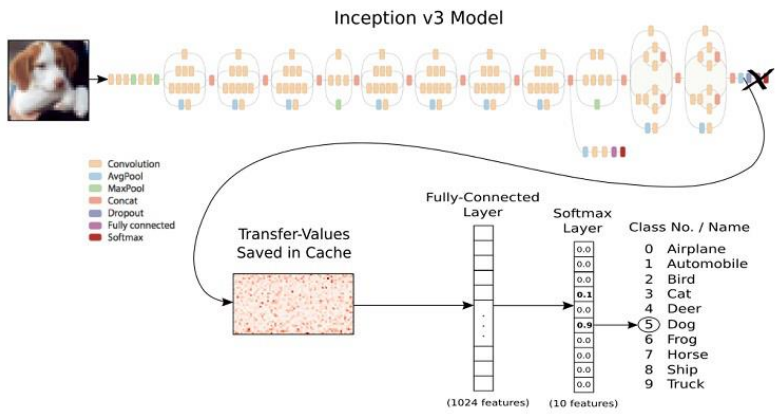
Importing libraries and getting iris datasets

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

```
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
```

Type!





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

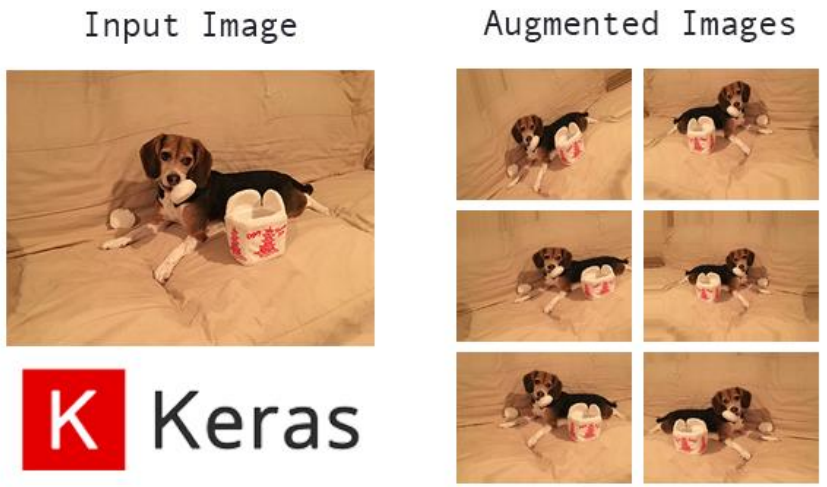
Getting started with Rock-Paper-Scissors Classification App

```
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)

    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
```

Image Augmentation

ImageDataGenerator (Keras)

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



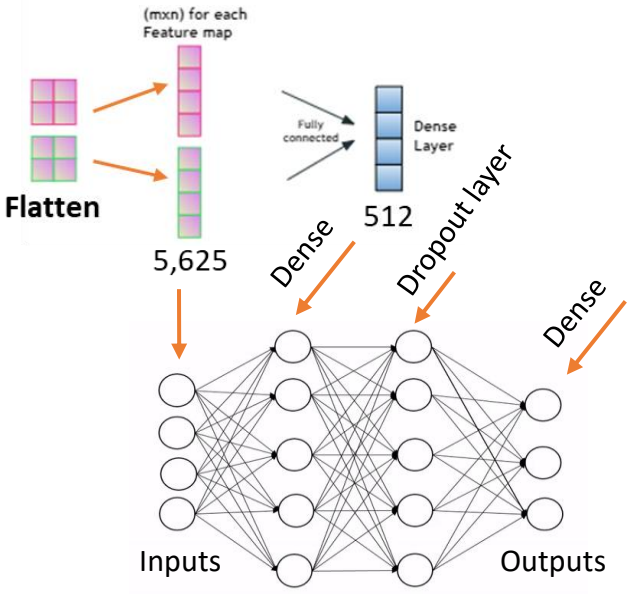
image_gen_w_aug function

Getting started with Rock-Paper-Scissors Classification App

```
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')
```

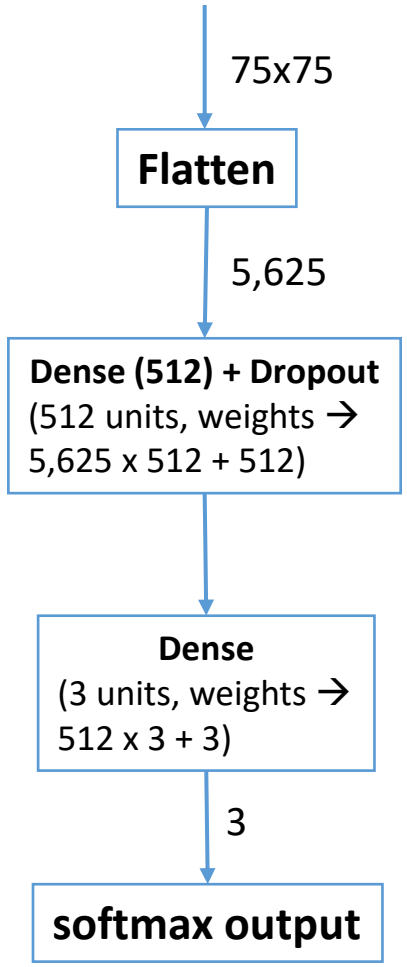
Dropout prevents overfitting and reduces training time!

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



Getting train and test datasets

Image dimension and number of labels



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')
```

steps = (epoch * examples)/batch size

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\".

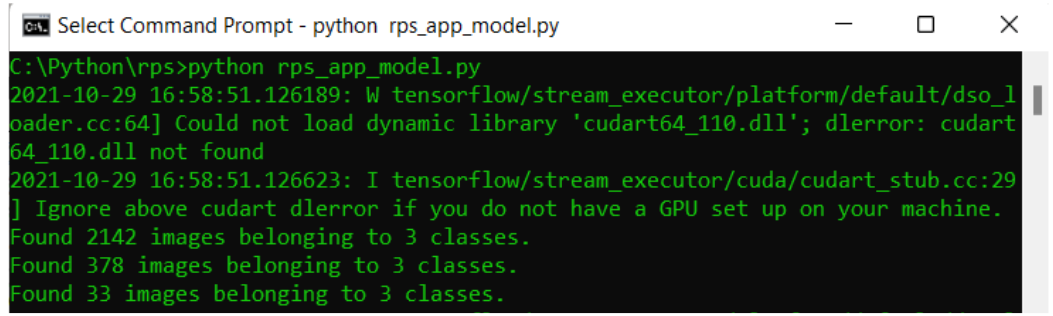
Remember to save the program as "C:\Python\rps\rps_app_model.py".

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.

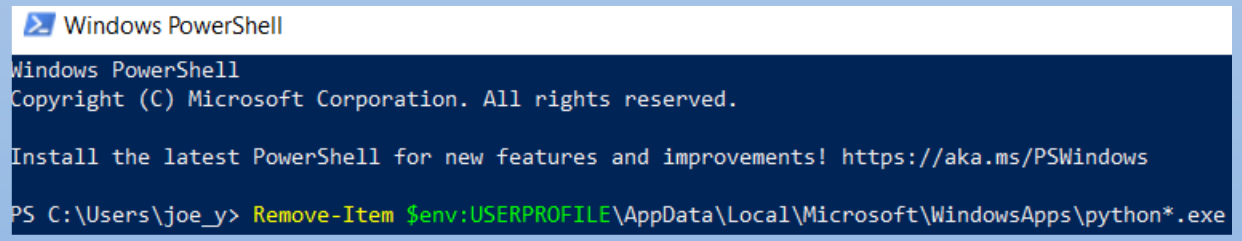
Type “python rps_app_model.py” to run the model training.



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

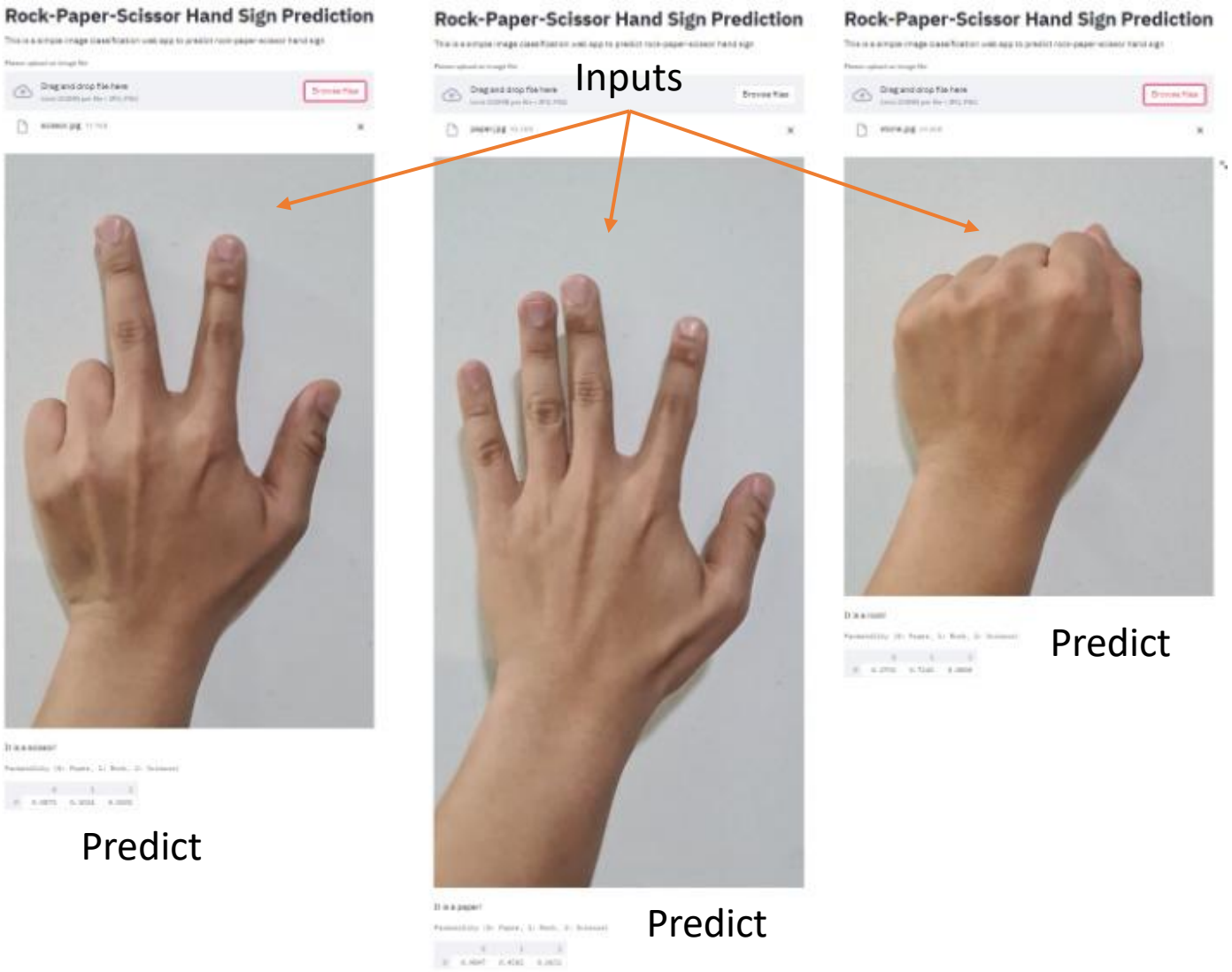
```
C:\Python\rps>python rps_app_model.py
Python was not found; run without arguments to install from the Microsoft Store
, or disable this shortcut from Settings > Manage App Execution Aliases.
```

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



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

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

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')
```

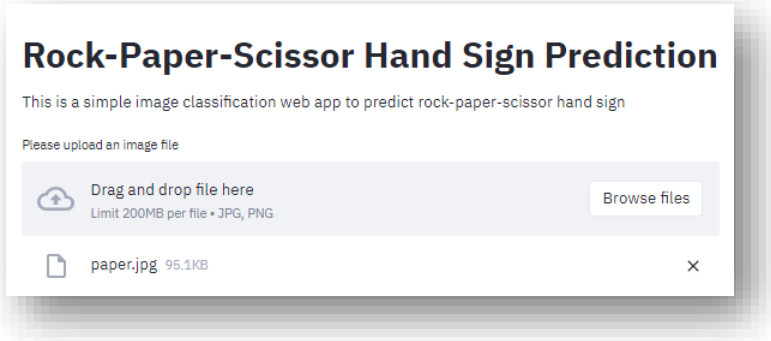
Import Libraries

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

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)
```

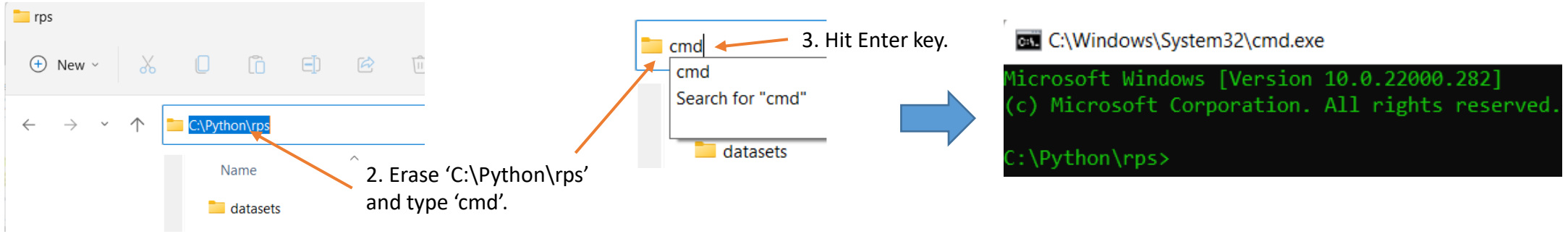


Create Streamlit UI Elements

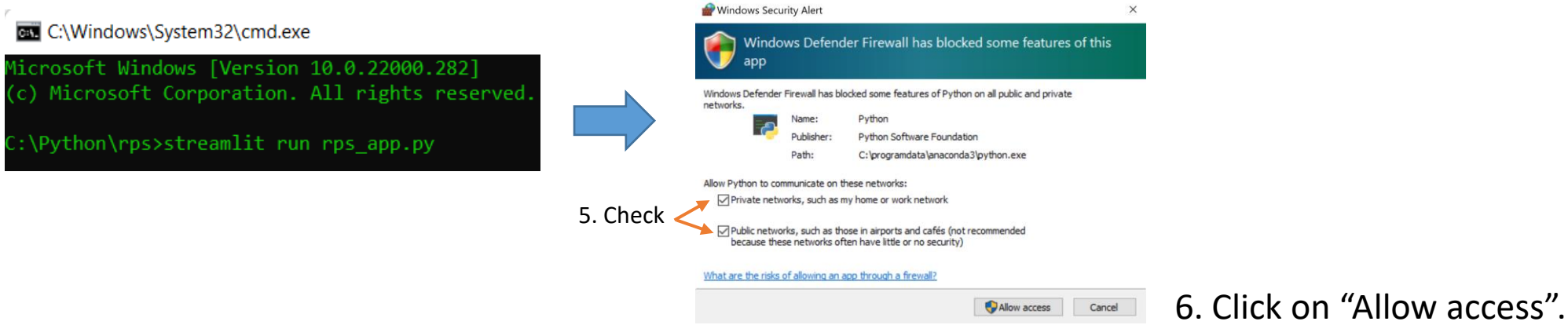
Predict output <Display on streamlit>
(1:rock, 2:paper or 3:scissor)

Running a Rock-Paper-Scissors classification app

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



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

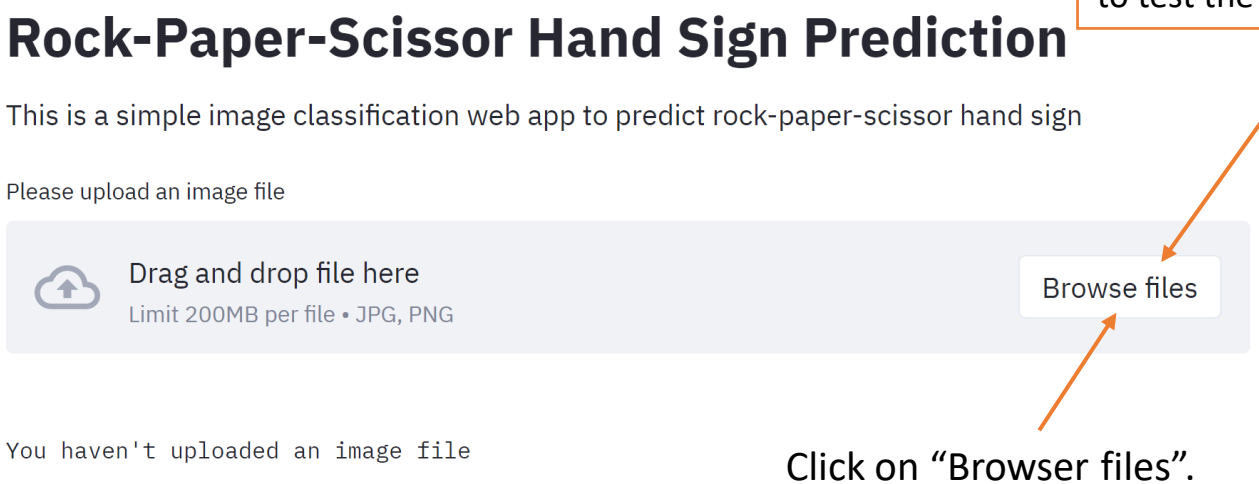


6. Click on “Allow access”.

Running a Rock-Paper-Scissors classification app

A browser will be launched with URL: localhost:8501

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



Click on “Browser files”.

Can you able to train and classify with different images?

Predicted Output →

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

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

Import libraries

Create import_and_predict function and return prediction

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

import cv2
import numpy as np
import os
import sys

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

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

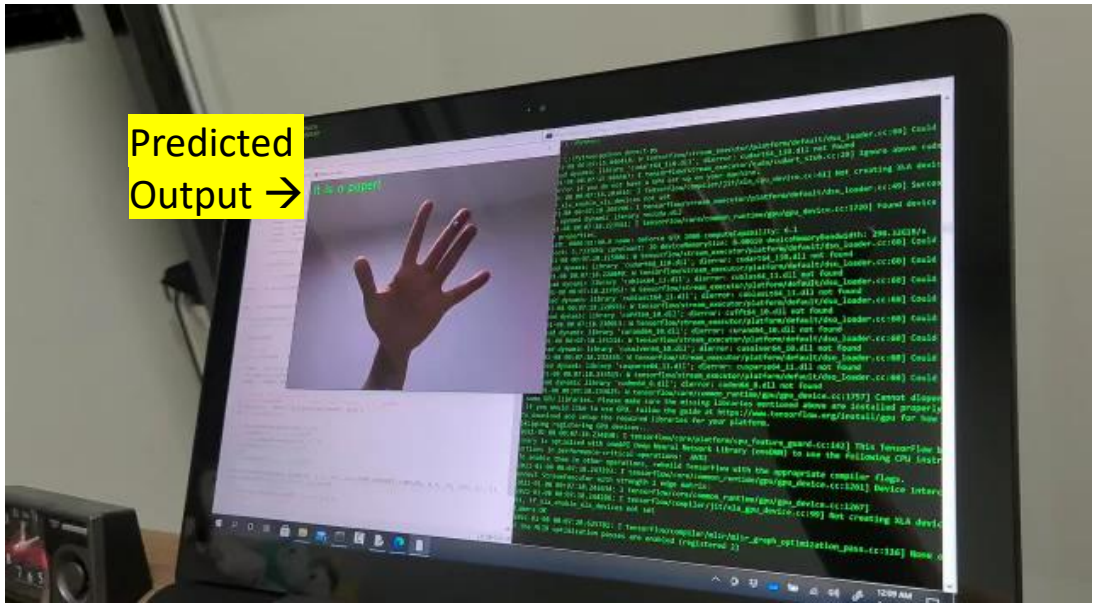
Open Command Prompt.

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.