

A Gesture Based Tool for Sterile Browsing of Radiology Images Using IBM Watson

1 INTRODUCTION

1.1 Overview

In this project we use gestures to browse images obtained during radiology. Gestures refer to non-verbal form of communication made using hands.

A major challenge involved in this process is to provide doctors with efficient, intuitive, accurate and safe means of interaction without affecting the quality of their work. Keyboards and pointing devices, such as a mouse, are today's principal method of human—computer interaction. However, the use of computer keyboards and mice by doctors and nurses in intensive care units (ICUs) is a common method for spreading infections. Humans can recognize body and sign language easily. This is possible due to the combination of vision and synaptic interactions that were formed along brain development.

In order to replicate this skill in computers, some problems need to be solved: how to separate objects of interest in images and which image capture technology and classification technique are more appropriate, among others. In this project Gesture based Desktop automation, First the model is trained pre trained on the images of different hand gestures, such as a showing numbers with fingers as 1,2,3,4. This model uses the integrated webcam to capture the video frame. The image of the gesture captured in the video frame is compared with the pre-trained model and the gesture is identified. If the gesture predicts is 0 - then images is converted into rectangle, 1 - image is Resized into (200,200), 2 - image is rotated by -45° , 3 - image is blurred, 4 - image is Resized into (400,400), 5 - image is converted into grayscale etc.

1.2 Purpose

It is used to browse through the images obtained using radiology using hand gestures rather than using mouse, keyboard, etc thereby maintaining sterility.

2 LITERATURE SURVEY

2.1 Existing Problem

Humans are able to recognize body and sign language easily. This is possible due to the combination of vision and synaptic interactions that were formed along brain development. In order to replicate this skill in computers, some problems need to be solved: how to separate objects of interest in images and which image capture technology and classification technique are more appropriate, among others.

2.2 Proposed Solution

The hand gesture control system “*Gestix*” developed by the authors helped the doctor to remain in place during the entire operation, without any need to move to the main control wall since all the commands were performed using hand gestures. The sterile gesture interface consists of a Canon VC-C4 camera, whose pan/tilt/zoom can be initially set using an infrared (IR) remote.

This camera is placed just over a large flat screen monitor.

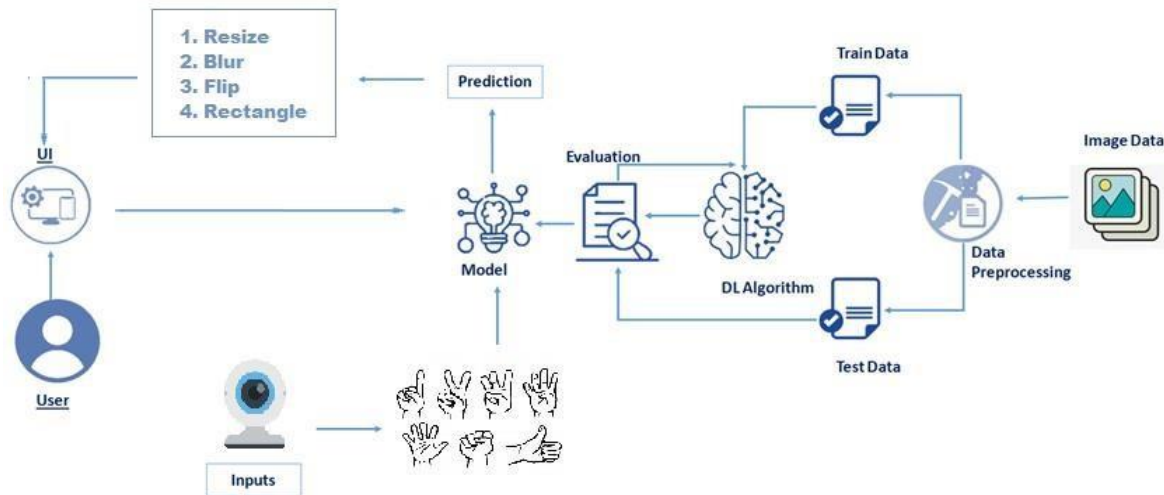
Additionally, an Intel Pentium IV, (600MHz, OS: Windows XP) with a Matrox Standard II video-capturing device is used.



The “*Gibson*” image browser is a 3D visualization medical tool that enables examination of images, such as: MRIs, CT scans and X-rays. The images are arranged over a multiple layer 3D cylinder. The image of interest is found through rotating the cylinder in the four cardinal directions. To interface the gesture recognition routines with the “*Gibson*” system, information such as the centroid of the hand, its size, and orientation are used to enable screen operations in the “*Gibson*” graphical user interface.

3 THEORITICAL ANALYSIS

3.1 Block Diagram



3.2 Hardware/Software Designing

Software Requirements:

- Anaconda Navigator
- Tensor flow
- Keras
- FlaskUpload

Hardware Requirements:

- Processor: Intel Core i3
- Hard Disk Space: Min 100 GB
- Ram: 4 GB
- Display: 14.1 Colour Monitor (LCD, CRT or LED)
- Clock Speed: 1.67 GHz

4 EXPERIMENTAL INVESTIGATIONS

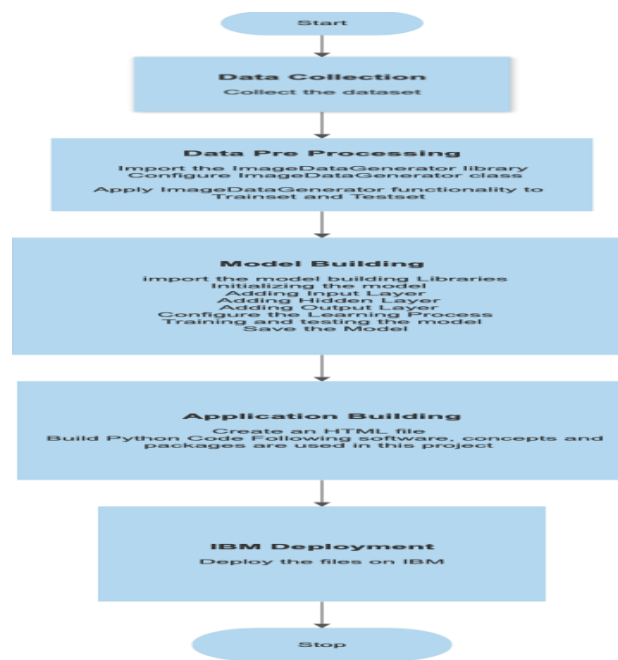
We found that many hospitals rely on mouse and keyboard to browse the images that are obtained during different surgeries, scans, etc. This can contaminate the environment with various infections thus compromising the sterility.

Various technologies have been developed to overcome this issue and one such technology was called ‘Gestix’.

This hand gesture system for MRI manipulation in an EMR image database called “*Gestix*” was tested during a brain biopsy surgery. This system is a real-time hand-tracking recognition technique based on colour and motion fusion. In an in vivo experiment, this type of interface prevented the surgeon's focus shift and change of location while achieving rapid intuitive interaction with an EMR image database. In addition to allowing sterile interaction with EMRs, the “*Gestix*” hand gesture interface provides:

1. ease of use—the system allows the surgeon to use his/her hands, their natural work tool;
2. rapid reaction—nonverbal instructions by hand gesture commands are intuitive and fast
3. an unencumbered interface—the proposed system does not require the surgeon to attach a microphone, use head-mounted (body-contact) sensing devices or to use foot pedals
4. distance control—the hand gestures can be performed up to 5 meters from the camera and still be recognized accurately.

5 FLOWCHART



- User interacts with the UI (User Interface) to upload the image as input.
- Depending on the different gesture inputs different operations are applied to the input image.
- Once model analyses the gesture, the prediction with operation applied on image is showcased on the

UI. To accomplish this, we have to complete all the activities and tasks listed below:

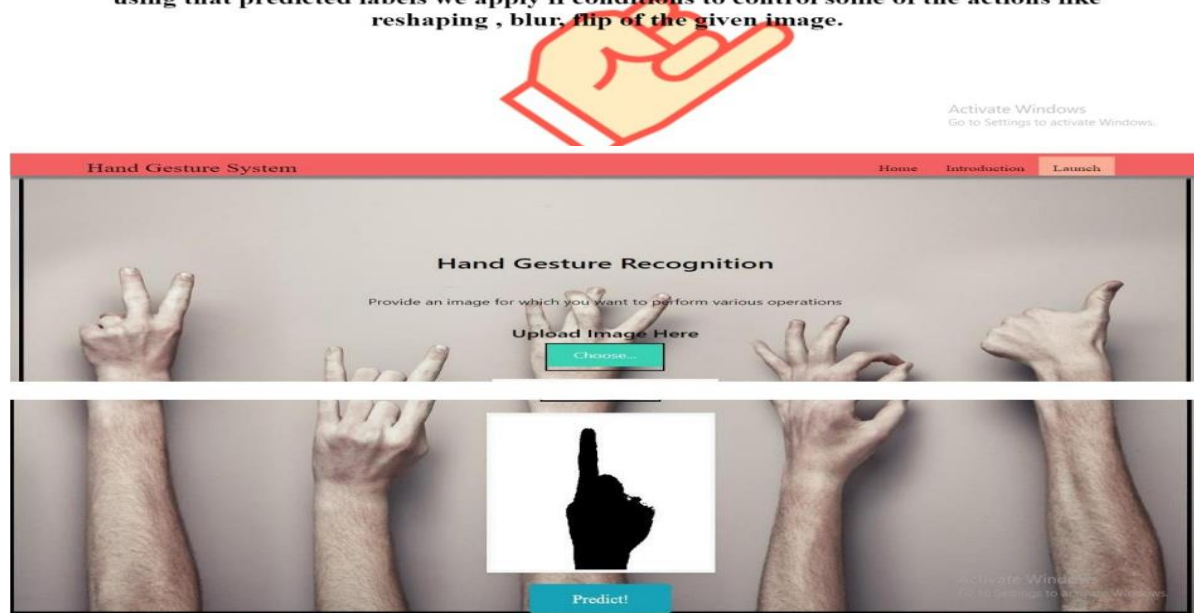
- Data Collection.
 - Collect the dataset or create the dataset
- Data Pre processing
 - Import the ImageDataGenerator library
 - Configure ImageDataGenerator class
 - Apply ImageDataGenerator functionality to Trainset and Testset
- Model Building
 - Import the model building Libraries
 - Initializing the model
 - Adding Input Layer
 - Adding Hidden Layer
 - Adding Output Layer
 - Configure the Learning Process
 - Training and testing the model
 - Save the Model
- Application Building
 - Create an HTML file
 - Build Python Code Following software, concepts and packages are used in this project
- Anaconda navigator
- Python packages:
 - open anaconda prompt as administrator
 - Type “pip install TensorFlow” (make sure you are working on python 64 bit)
 - Type “pip install OpenCV-python”
 - Type “pip install flask”

6 RESULT

Final findings (Output) of the project along with screenshots.



Hand Gesture recognition system provides us an innovative, natural, user friendly way of interaction with the computer which is more familiar to the human beings. In our project, the hand region is extracted from the background by using Region of interest. Then, we will be predicting the labels based on the CNN trained model weights of hand gestures using that predicted labels we apply if conditions to control some of the actions like reshaping , blur, flip of the given image.



Through this project we found that we can maintain the sterility of an operation theatre, etc by using hand-based gesture tools to browse the images obtained.

7 ADVANTAGES & DISADVANTAGES

Advantages:

- Major advantage of this tool is that it helps to maintain the sterility of the environment.
- It is also easy to use and is quicker than the existing methods to browse images.
- It can also be performed even if the surgeon is a bit far away from the system, this helps to save time.
- The tool does not need the person using it to have an apparatus or any devices on them to use it.
- They can simply move their hands to browse through the images.

Disadvantages:

- The tool can be quite expensive as it requires cameras and other expensive devices to capture images and process it.

8 APPLICATIONS

- This hand based gesture tool developed can be mainly used in the medical industry to browse images without compromising the sterility.
- However, it can also be used in different industries while presenting certain ideas, during meetings, and can be used by teachers while teaching.

9 CONCLUSION

In this project we developed a tool which recognises hand gestures and enables doctors to browse through radiology images using these gestures. This enables doctors and surgeons to maintain the sterility as they would not have to touch any mouse or keyboard to go through the images. This tool is also easy to use and is quicker than the regular method of using mouse/keyboard.

It can be used regardless of the user's location since they don't have to be in contact with any device. It also does not require the user to have any device on them to use it. Further this technology can be extended to other industries like it can be used by presenters, by teachers for show images in the classroom, etc.

10 FUTURE SCOPE

The tool can be made quicker by increasing the recognition speed.

More number of gestures can be added thereby increasing this tool's functionality and useability for different purposes.

Tracking of both hands can be added to increase the set of commands.

Voice commands can also be added to further increase the functionality.

11 BIBLIOGRAPHY

Research papers:

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2410001/>

<https://pubmed.ncbi.nlm.nih.gov/18451034/>

https://www.researchgate.net/publication/5401674_A_Gesturebased_Tool_for_Sterile_Browsing_of_Radiology_Images

Smartinternz Website:

https://smartinternz.com/Student/guided_project_workspace/

Appendix source code

```
from flask import Flask,render_template,request

# Flask-It is our framework which we are going to use to run/serve our application.

#request-for accessing file which was uploaded by the user on our application.

import operator

import cv2 # opencv library

from tensorflow.keras.models import load_model#to load our trained model

import os

from werkzeug.utils import secure_filename


app = Flask(__name__,template_folder="templates") # initializing a flask app
```



```
# Loading the model
```

```
model=load_model('gesture.h5')
```

```
print("Loaded model from disk")
```

```
@app.route('/')# route to display the home page
```

```
def home():
```

```
    return render_template('home.html')#rendering the home page
```

```
@app.route('/intro') # routes to the intro page
```

```
def intro():
```

```
    return render_template('intro.html')#rendering the intro page
```

```
@app.route('/image1',methods=['GET','POST'])# routes to the index html
```

```
def image1():
```

```
    return render_template("index6.html")
```

```
@app.route('/predict',methods=['GET', 'POST'])# route to show the predictions in a web UI
```

```
def launch():
```

```
    if request.method == 'POST':
```

```
        print("inside image")
```

```
        f = request.files['image']
```

```
        basepath = os.path.dirname(__file__)
```

```
        file_path = os.path.join(basepath, 'uploads', secure_filename(f.filename))
```

```

f.save(file_path)
print(file_path)
cap = cv2.VideoCapture(0)
while True:
    _, frame = cap.read() #capturing the video frame values
    # Simulating mirror image
    frame = cv2.flip(frame, 1)

    # Got this from collect-data.py
    # Coordinates of the ROI
    x1 = int(0.5*frame.shape[1])
    y1 = 10
    x2 = frame.shape[1]-10
    y2 = int(0.5*frame.shape[1])
    # Drawing the ROI
    # The increment/decrement by 1 is to compensate for the bounding box
    cv2.rectangle(frame, (x1-1, y1-1), (x2+1, y2+1), (255,0,0), 1)
    # Extracting the ROI
    roi = frame[y1:y2, x1:x2]

    # Resizing the ROI so it can be fed to the model for prediction
    roi = cv2.resize(roi, (64, 64))
    roi = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
    _, test_image = cv2.threshold(roi, 120, 255, cv2.THRESH_BINARY)
    cv2.imshow("test", test_image)

    # Batch of 1
    result = model.predict(test_image.reshape(1, 64, 64, 1))

```

```

prediction = {'ZERO': result[0][0],
             'ONE': result[0][1],
             'TWO': result[0][2],
             'THREE': result[0][3],
             'FOUR': result[0][4],
             'FIVE': result[0][5]}

# Sorting based on top prediction
prediction = sorted(prediction.items(), key=operator.itemgetter(1), reverse=True)

# Displaying the predictions
cv2.putText(frame, prediction[0][0], (10, 120), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255),
1)

cv2.imshow("Frame", frame)

#loading an image
image1=cv2.imread(file_path)
if prediction[0][0]=='ONE':

    resized = cv2.resize(image1, (200, 200))
    cv2.imshow("Fixed Resizing", resized)
    key=cv2.waitKey(3000)

    if (key & 0xFF) == ord("1"):
        cv2.destroyAllWindows("Fixed Resizing")

elif prediction[0][0]=='ZERO':

```

```

cv2.rectangle(image1, (480, 170), (650, 420), (0, 0, 255), 2)

cv2.imshow("Rectangle", image1)

cv2.waitKey(0)

key=cv2.waitKey(3000)

if (key & 0xFF) == ord("0"):
    cv2.destroyAllWindows("Rectangle")

elif prediction[0][0]=='TWO':
    (h, w, d) = image1.shape
    center = (w // 2, h // 2)
    M = cv2.getRotationMatrix2D(center, -45, 1.0)
    rotated = cv2.warpAffine(image1, M, (w, h))
    cv2.imshow("OpenCV Rotation", rotated)
    key=cv2.waitKey(3000)
    if (key & 0xFF) == ord("2"):
        cv2.destroyAllWindows("OpenCV Rotation")

elif prediction[0][0]=='THREE':
    blurred = cv2.GaussianBlur(image1, (11, 11), 0)
    cv2.imshow("Blurred", blurred)
    key=cv2.waitKey(3000)
    if (key & 0xFF) == ord("3"):
        cv2.destroyAllWindows("Blurred")
else:
    continue

```

```

interrupt = cv2.waitKey(10)

if interrupt & 0xFF == 27: # esc key
    break

```

```

cap.release()

cv2.destroyAllWindows()

return render_template("home.html")

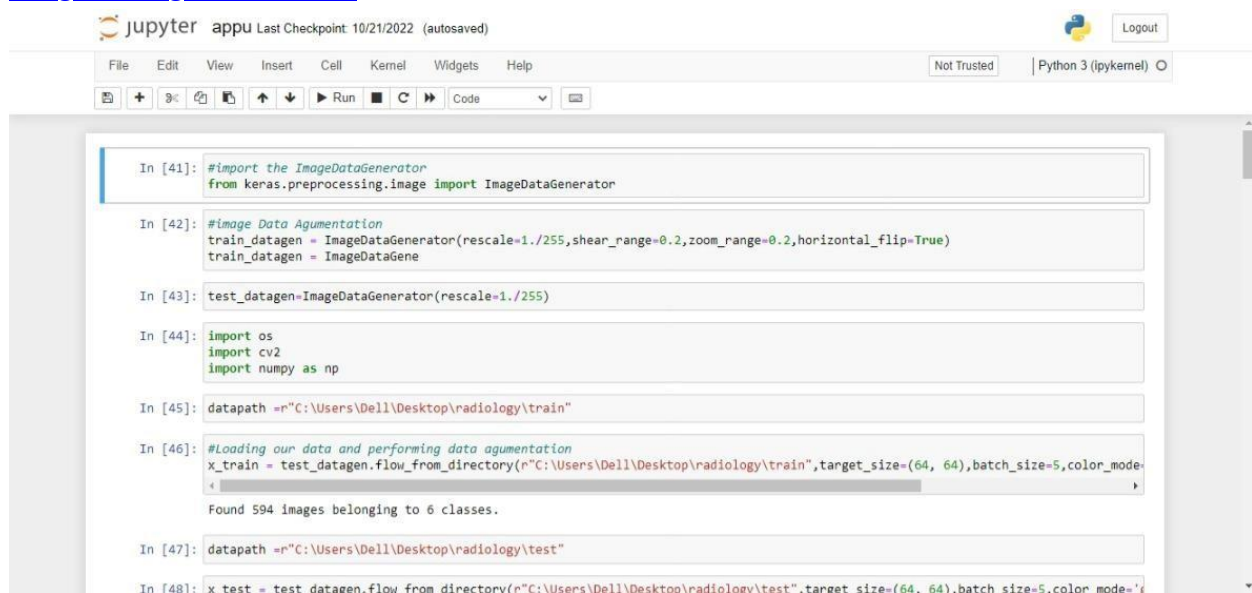
```

```
if _name_ == "_main_":
```

```
    # running the app
```

```
    app.run(debug=False)
```

<https://github.com/smartinternz02/Gesture-based-Tool-for-Sterile-Browsing-of-Radiology-Images-Using-IBM-Watson>



The screenshot shows a Jupyter Notebook interface with the following code cells:

```

In [41]: #import the ImageDataGenerator
from keras.preprocessing.image import ImageDataGenerator

In [42]: #image Data Augmentation
train_datagen = ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)
train_datagen = ImageDataGene

In [43]: test_datagen=ImageDataGenerator(rescale=1./255)

In [44]: import os
import cv2
import numpy as np

In [45]: datapath = r"C:\Users\Dell\Desktop\radiology\train"

In [46]: #Loading our data and performing data augmentation
x_train = test_datagen.flow_from_directory(r"C:\Users\Dell\Desktop\radiology\train", target_size=(64, 64), batch_size=5, color_mode='rgb')
Found 594 images belonging to 6 classes.

In [47]: datapath = r"C:\Users\Dell\Desktop\radiology\test"

In [48]: x_test = test_datagen.flow_from_directory(r"C:\Users\Dell\Desktop\radiology\test", target_size=(64, 64), batch_size=5, color_mode='rgb')

```

jupyter appu Last Checkpoint: 10/21/2022 (autosaved)
Logout

File Edit View Insert Cell Kernel Widgets Help
Not Trusted Python 3 (ipykernel)

```

In [46]: #Loading our data and performing data augmentation
x_train = test_datagen.flow_from_directory(r"C:\Users\Dell\Desktop\radiology\train",target_size=(64, 64),batch_size=5,color_mode='rgb')

Found 594 images belonging to 6 classes.

In [47]: datapath = r"C:\Users\Dell\Desktop\radiology\test"

In [48]: x_test = test_datagen.flow_from_directory(r"C:\Users\Dell\Desktop\radiology\test",target_size=(64, 64),batch_size=5,color_mode='rgb')

Found 30 images belonging to 6 classes.

In [49]: #Importing Necessary Libraries
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 [50]: #Initializing the model
model=Sequential()

In [51]: #First convolution Layer and pooling
model.add(Conv2D(32,(3,3),input_shape=(64, 64, 1),activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

In [52]: #Second convolution Layer and pooling
model.add(Conv2D(32,(3, 3),activation='relu'))


| Layer (type)                   | Output Shape       | Param # |
|--------------------------------|--------------------|---------|
| conv2d_2 (Conv2D)              | (None, 62, 62, 32) | 320     |
| max_pooling2d_2 (MaxPooling2D) | (None, 31, 31, 32) | 0       |
| conv2d_3 (Conv2D)              | (None, 29, 29, 32) | 9248    |


```

jupyter appu Last Checkpoint: 10/21/2022 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

```
In [55]: model.summary()
Model: "sequential_1"
Layer (type) Output Shape Param #
-----
conv2d_2 (Conv2D) (None, 62, 62, 32) 320
max_pooling2d_2 (MaxPooling2D) (None, 31, 31, 32) 0
conv2d_3 (Conv2D) (None, 29, 29, 32) 9248
max_pooling2d_3 (MaxPooling2D) (None, 14, 14, 32) 0
flatten_1 (Flatten) (None, 6272) 0
dense_2 (Dense) (None, 128) 802944
dense_3 (Dense) (None, 6) 774
-----
Total params: 813,286
Trainable params: 813,286
Non-trainable params: 0
```

jupyter appu Last Checkpoint: 10/21/2022 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

```
In [56]: #compiling the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

In [57]: #Fitting the model
model.fit_generator(
    generator=x_train, steps_per_epoch = len(x_train),
    epochs=20, validation_data=x_test, validation_steps = len(x_test))

Epoch 1/20
C:\Users\Dell\AppData\Local\Temp\ipykernel_11256\197708238.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(

119/119 [=====] - 4s 23ms/step - loss: 1.3687 - accuracy: 0.4596 - val_loss: 0.7462 - val_accuracy: 0.8000
Epoch 2/20
119/119 [=====] - 3s 22ms/step - loss: 0.5288 - accuracy: 0.8199 - val_loss: 0.5674 - val_accuracy: 0.8000
Epoch 3/20
119/119 [=====] - 3s 21ms/step - loss: 0.2812 - accuracy: 0.8990 - val_loss: 0.5679 - val_accuracy: 0.8333
Epoch 4/20
119/119 [=====] - 2s 21ms/step - loss: 0.1245 - accuracy: 0.9680 - val_loss: 0.6717 - val_accuracy: 0.8667
Epoch 5/20
119/119 [=====] - 3s 22ms/step - loss: 0.0348 - accuracy: 0.9933 - val_loss: 0.5370 - val_accuracy: 0.8667
Epoch 6/20
119/119 [=====] - 3s 22ms/step - loss: 0.0308 - accuracy: 0.9955 - val_loss: 0.5305 - val_accuracy: 0.8667
```

jupyter appu Last Checkpoint: 10/21/2022 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted | Python 3 (ipykernel) O

In [57]: `#Fitting the model`
`model.fit_generator(`
 `generator=x_train, steps_per_epoch = len(x_train),`
 `epochs=20, validation_data=x_test, validation_steps = len(x_test))`

Epoch 1/20

C:\Users\Dell\AppData\Local\Temp\ipykernel_11256\197708238.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(`



119/119 [=====] - 4s 23ms/step - loss: 1.3687 - accuracy: 0.4506 - val_loss: 0.7462 - val_accuracy: 0.8000
Epoch 2/20
119/119 [=====] - 3s 22ms/step - loss: 0.5288 - accuracy: 0.8199 - val_loss: 0.5674 - val_accuracy: 0.8000
Epoch 3/20
119/119 [=====] - 3s 21ms/step - loss: 0.2812 - accuracy: 0.8990 - val_loss: 0.5679 - val_accuracy: 0.8333
Epoch 4/20
119/119 [=====] - 2s 21ms/step - loss: 0.1245 - accuracy: 0.9680 - val_loss: 0.6717 - val_accuracy: 0.8667
Epoch 5/20
119/119 [=====] - 3s 22ms/step - loss: 0.0348 - accuracy: 0.9933 - val_loss: 0.5370 - val_accuracy: 0.8667
Epoch 6/20
119/119 [=====] - 3s 22ms/step - loss: 0.0208 - accuracy: 0.9966 - val_loss: 0.5995 - val_accuracy: 0.8667
Epoch 7/20
119/119 [=====] - 3s 22ms/step - loss: 0.0175 - accuracy: 0.9975 - val_loss: 0.6017 - val_accuracy: 0.8667
Epoch 8/20
119/119 [=====] - 3s 22ms/step - loss: 0.0152 - accuracy: 0.9983 - val_loss: 0.6566 - val_accuracy: 0.8667
Epoch 9/20
119/119 [=====] - 3s 21ms/step - loss: 0.0052 - accuracy: 0.9983 - val_loss: 0.6566 - val_accuracy: 0.8667
Epoch 10/20
119/119 [=====] - 3s 21ms/step - loss: 9.4692e-04 - accuracy: 1.0000 - val_loss: 0.7115 - val_accuracy: 0.8667
Epoch 11/20
119/119 [=====] - 3s 21ms/step - loss: 3.6690e-04 - accuracy: 1.0000 - val_loss: 0.7240 - val_accuracy: 0.8667
Epoch 12/20
119/119 [=====] - 3s 21ms/step - loss: 2.6763e-04 - accuracy: 1.0000 - val_loss: 0.7329 - val_accuracy: 0.8667
Epoch 13/20
119/119 [=====] - 2s 21ms/step - loss: 2.0857e-04 - accuracy: 1.0000 - val_loss: 0.7449 - val_accuracy: 0.8667
Epoch 14/20
119/119 [=====] - 2s 21ms/step - loss: 1.7099e-04 - accuracy: 1.0000 - val_loss: 0.7540 - val_accuracy: 0.8667
Epoch 15/20
119/119 [=====] - 2s 21ms/step - loss: 1.7099e-04 - accuracy: 1.0000 - val_loss: 0.7540 - val_accuracy: 0.8667
Epoch 16/20
119/119 [=====] - 2s 21ms/step - loss: 1.7099e-04 - accuracy: 1.0000 - val_loss: 0.7540 - val_accuracy: 0.8667
Epoch 17/20
119/119 [=====] - 2s 21ms/step - loss: 1.7099e-04 - accuracy: 1.0000 - val_loss: 0.7540 - val_accuracy: 0.8667
Epoch 18/20
119/119 [=====] - 2s 21ms/step - loss: 1.7099e-04 - accuracy: 1.0000 - val_loss: 0.7540 - val_accuracy: 0.8667
Epoch 19/20
119/119 [=====] - 2s 21ms/step - loss: 1.7099e-04 - accuracy: 1.0000 - val_loss: 0.7540 - val_accuracy: 0.8667
Epoch 20/20
119/119 [=====] - 2s 21ms/step - loss: 1.7099e-04 - accuracy: 1.0000 - val_loss: 0.7540 - val_accuracy: 0.8667

Out[57]: <keras.callbacks.History at 0x2278d5c6c10>

In [58]: `#Save the model`
`model.save('gesture.h5')`

In [59]: `model_json = model.to_json()`
`with open("model-bw.json", "w") as json_file:`
 `json_file.write(model_json)`

In [60]: `#Predicting our results`
`from tensorflow.keras.models import load_model`


jupyter appu Last Checkpoint: 10/21/2022 (autosaved)
 Logout

File Edit View Insert Cell Kernel Widgets Help
Not Trusted Python 3 (ipykernel)

```

In [58]: #Save the model
model.save('gesture.h5')


In [59]: model_json = model.to_json()
with open("model-bw.json","w") as json_file:
    json_file.write(model_json)



In [60]: #Predicting our results
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
model = load_model("gesture.h5")
path = r"C:\Users\Dell\Desktop\radiology\test\1\1.jpg"

In [61]: %pylab inline
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
imgs = mpimg.imread(r"C:\Users\Dell\Desktop\radiology\test\1\1.jpg")
imgplot = plt.imshow(imgs)
plt.show()

%pylab is deprecated, use %matplotlib inline and import the required libraries.
Populating the interactive namespace from numpy and matplotlib

```




jupyter appu Last Checkpoint: 10/21/2022 (autosaved)
 Logout

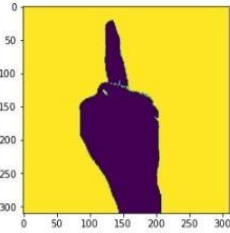
File Edit View Insert Cell Kernel Widgets Help
Not Trusted Python 3 (ipykernel)

```

In [61]: %pylab inline
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
imgs = mpimg.imread(r"C:\Users\Dell\Desktop\radiology\test\1\1.jpg")
imgplot = plt.imshow(imgs)
plt.show()

%pylab is deprecated, use %matplotlib inline and import the required libraries.
Populating the interactive namespace from numpy and matplotlib

```



```

In [62]: img = image.load_img(r"C:\Users\Dell\Desktop\radiology\test\1\1.jpg", grayscale=True,
                             target_size= (64,64))

```

```
jupyter appu Last Checkpoint 10/21/2022 (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

In [62]: img = image.load_img("C:\Users\De11\Desktop\radiology\test\1\1.jpg", grayscale=True,
        x = image.img_to_array(img)
        x.shape
C:\Users\De11\AppData\Roaming\Python\Python39\site-packages\keras\utils\image_utils.py:409: UserWarning: grayscale is deprecated. Please use color_mode = "grayscale"
warnings.warn(
Out[62]: (64, 64, 1)
In [63]: type(x)
Out[63]: numpy.ndarray
In [181]: x = np.expand_dims(x, axis = 0)
In [183]: x.shape
Out[183]: (1, 1, 1, 64, 64, 1)
In [224]: !pip install pred
Defaulting to user installation because normal site-packages is not writeable
ERROR: Could not find a version that satisfies the requirement pred (from versions: none)
ERROR: No matching distribution found for pred

jupyter appu Last Checkpoint 10/21/2022 (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

In [284]: index=['0','1','2','3','4','5']
        result=str(i)
        result
Out[284]: '1'
In [285]: !pip install flask
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: flask in c:\programdata\anaconda3\lib\site-packages (1.1.2)
Requirement already satisfied: Werkzeug>=0.15 in c:\programdata\anaconda3\lib\site-packages (from flask) (2.0.3)
Requirement already satisfied: click>=5.1 in c:\programdata\anaconda3\lib\site-packages (from flask) (8.0.4)
Requirement already satisfied: Jinja2>=2.10.1 in c:\programdata\anaconda3\lib\site-packages (from flask) (2.11.3)
Requirement already satisfied: itsdangerous>=0.24 in c:\programdata\anaconda3\lib\site-packages (from flask) (2.0.1)
Requirement already satisfied: colorama in c:\programdata\anaconda3\lib\site-packages (from click>=5.1->flask) (0.4.4)
Requirement already satisfied: MarkupSafe>=0.23 in c:\programdata\anaconda3\lib\site-packages (from Jinja2>=2.10.1->flask) (2.0.1)
In [286]: from flask import Flask,render_template,request
        import operator
        import cv2
        from tensorflow.keras.models import load_model
        import os

In [285]: !pip install flask
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: flask in c:\programdata\anaconda3\lib\site-packages (1.1.2)
Requirement already satisfied: Werkzeug>=0.15 in c:\programdata\anaconda3\lib\site-packages (from flask) (2.0.3)
Requirement already satisfied: click>=5.1 in c:\programdata\anaconda3\lib\site-packages (from flask) (8.0.4)
Requirement already satisfied: Jinja2>=2.10.1 in c:\programdata\anaconda3\lib\site-packages (from flask) (2.11.3)
Requirement already satisfied: itsdangerous>=0.24 in c:\programdata\anaconda3\lib\site-packages (from flask) (2.0.1)
Requirement already satisfied: colorama in c:\programdata\anaconda3\lib\site-packages (from click>=5.1->flask) (0.4.4)
Requirement already satisfied: MarkupSafe>=0.23 in c:\programdata\anaconda3\lib\site-packages (from Jinja2>=2.10.1->flask) (2.0.1)
In [286]: from flask import Flask,render_template,request
        import operator
        import cv2
        from tensorflow.keras.models import load_model
        import os
        from werkzeug.utils import secure_filename
In [ ]: app = Flask(__name__,template_folder="templates")
        model=load_model('gesture.h5')
        print("Loaded model from disk")
In [ ]:
```

RUN

