A Gesture-Based Tool For Sterile Browsing Of Radiology Images Using IBM Watson

1. INTRODUCTION

1.1 Overview

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.

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 predictes is 1 then images is blurred;2, image is resized;3,image is rotated 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

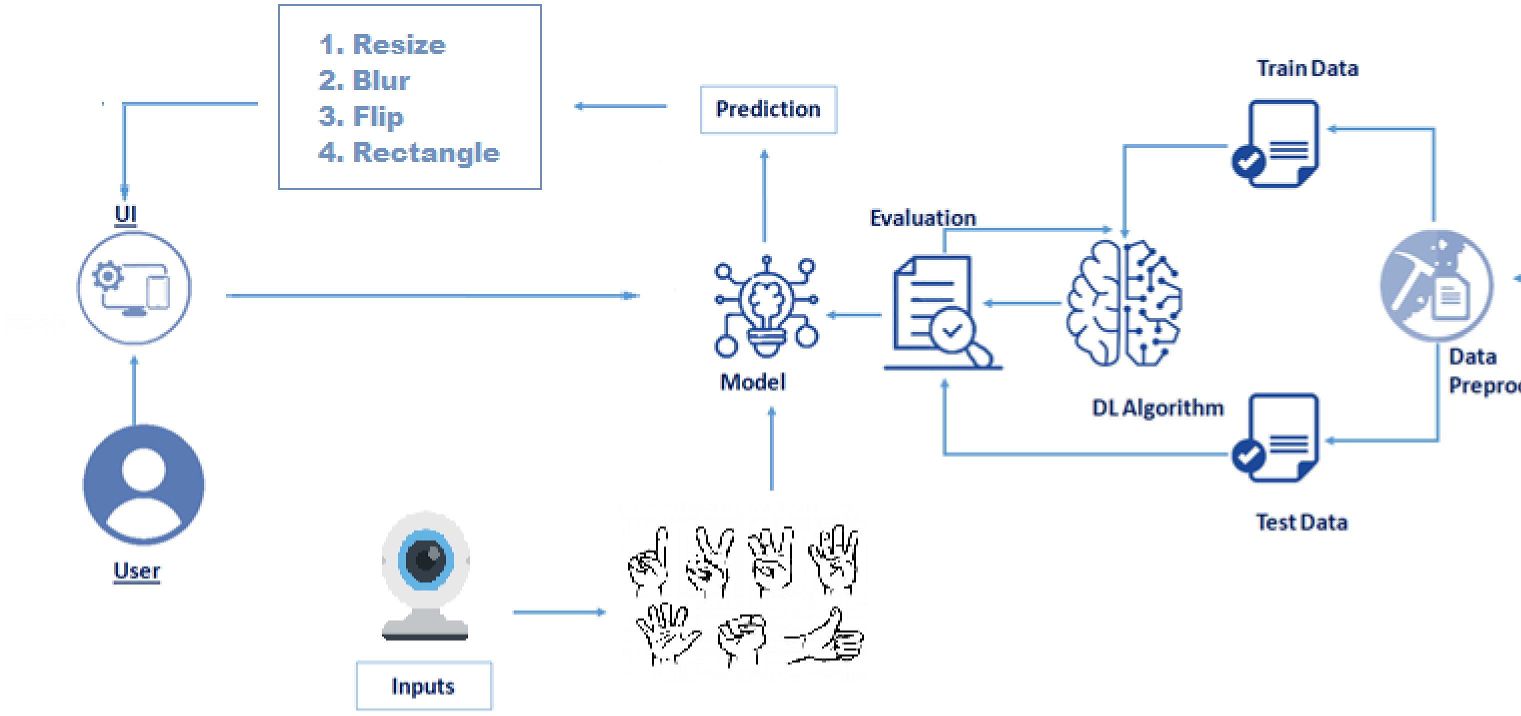
Humans are able to recognize body and sign language easily. This is possible due to the combination of vision and synaptic interactions that were formed along brain development . In order to replicate this skill in computers, some problems need to be solved: how to separate objects of interest in images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

2.2 Proposed solution

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

3. THEORITICAL ANALYSIS

3.1 Block Diagram



3.2 Hardware / Software designing

Software Requirements:

- Anaconda Navigator
- Tensor flow
- Keras
- Flask

Hardware Requirements:

- Processor : Intel Core i3

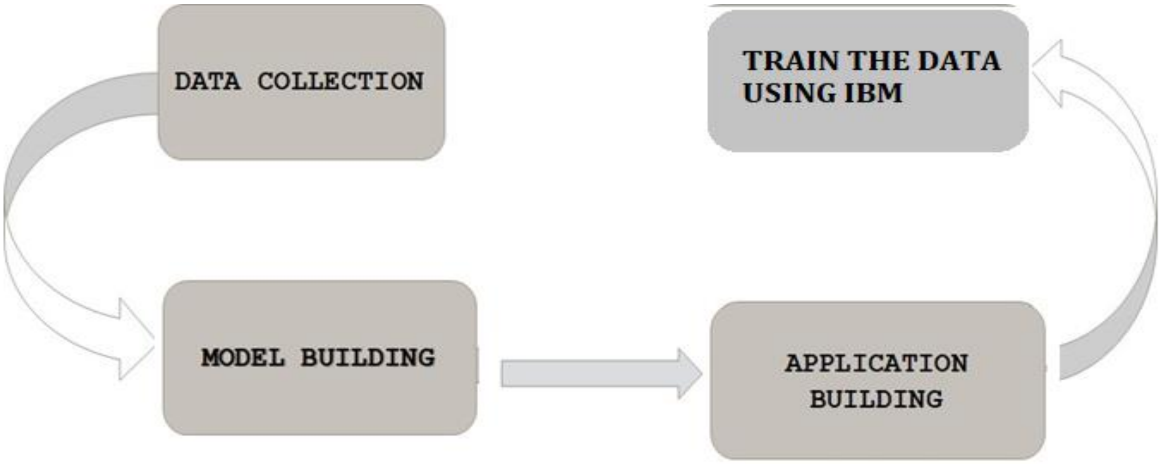
- Hard Disk Space : Min 100 GB
- Ram : 4 GB
- Display : 14.1 “Color Monitor(LCD, CRT or LED)

Clock Speed : 1.67 GHz

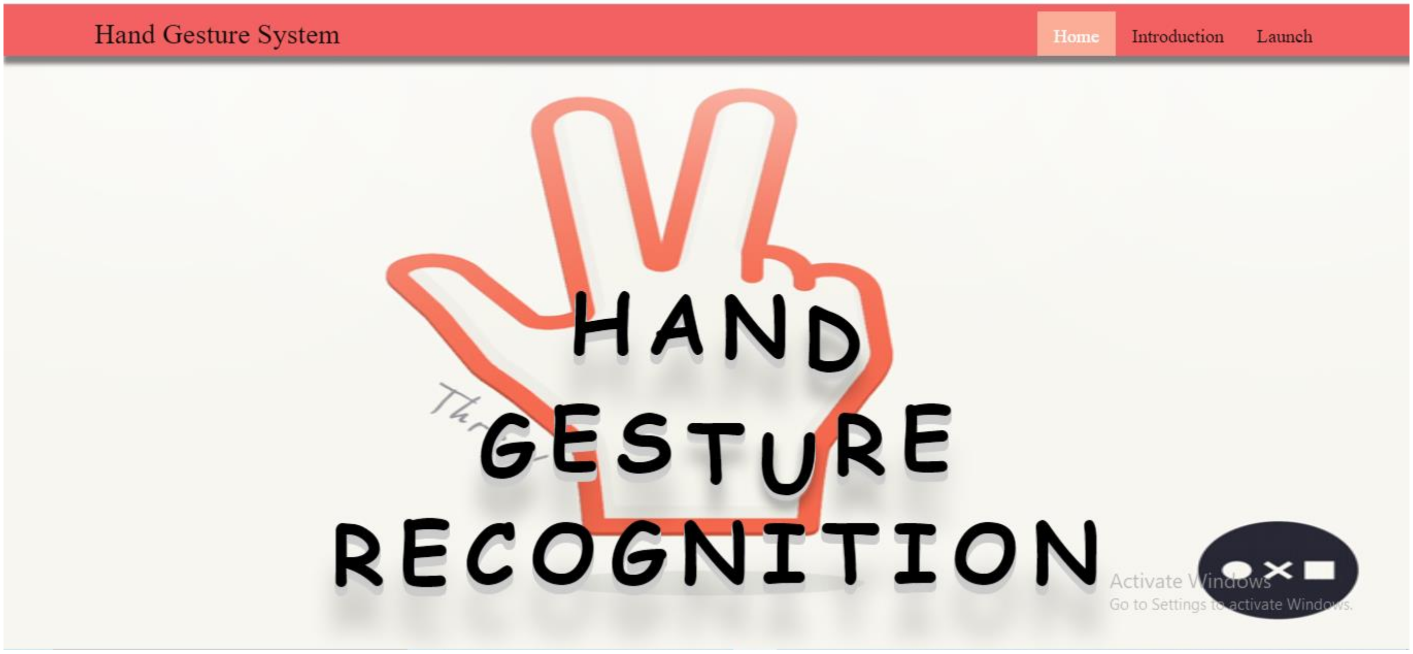
4. EXPERIMENTAL INVESTIGATIONS

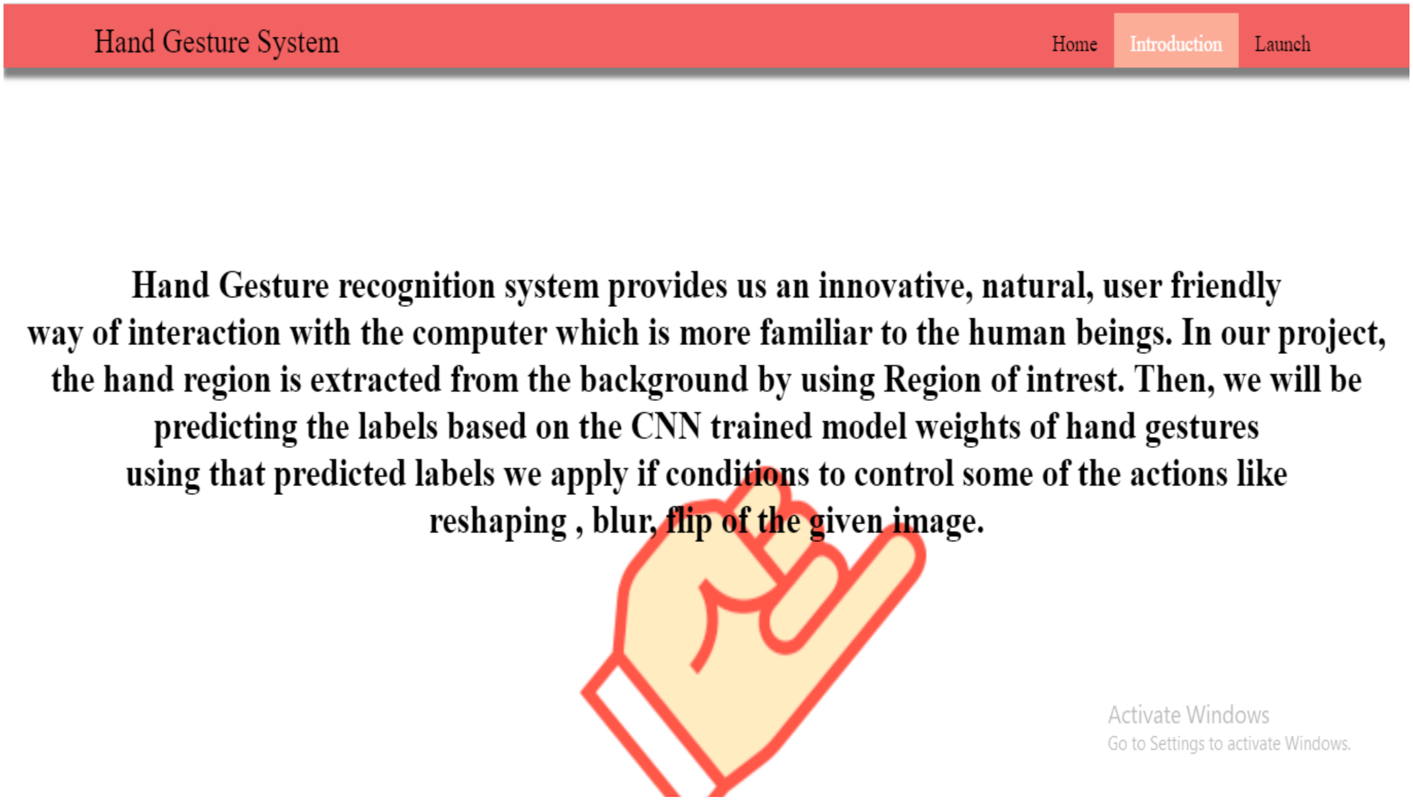
- User interacts with the UI (User Interface) to upload the image as input
- Depending on the different gesture inputs different operations are applied to the input image.
- Once model analyses the gesture, the prediction with operation applied on image is showcased on the UI.

5. FLOWCHART

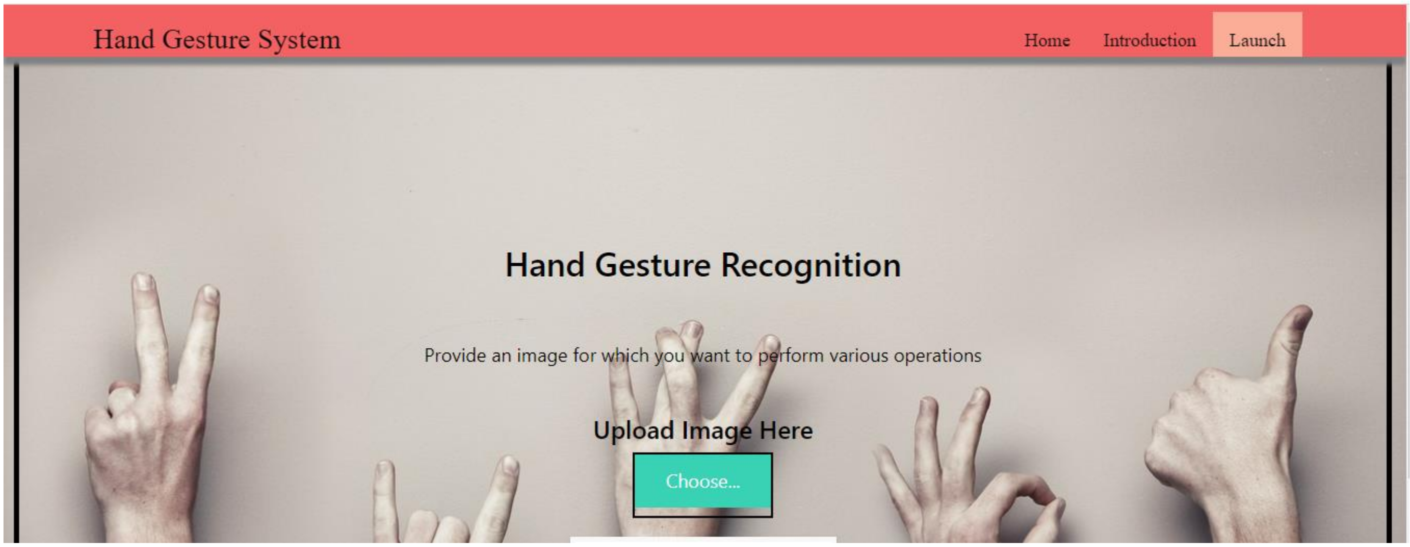


6. RESULT





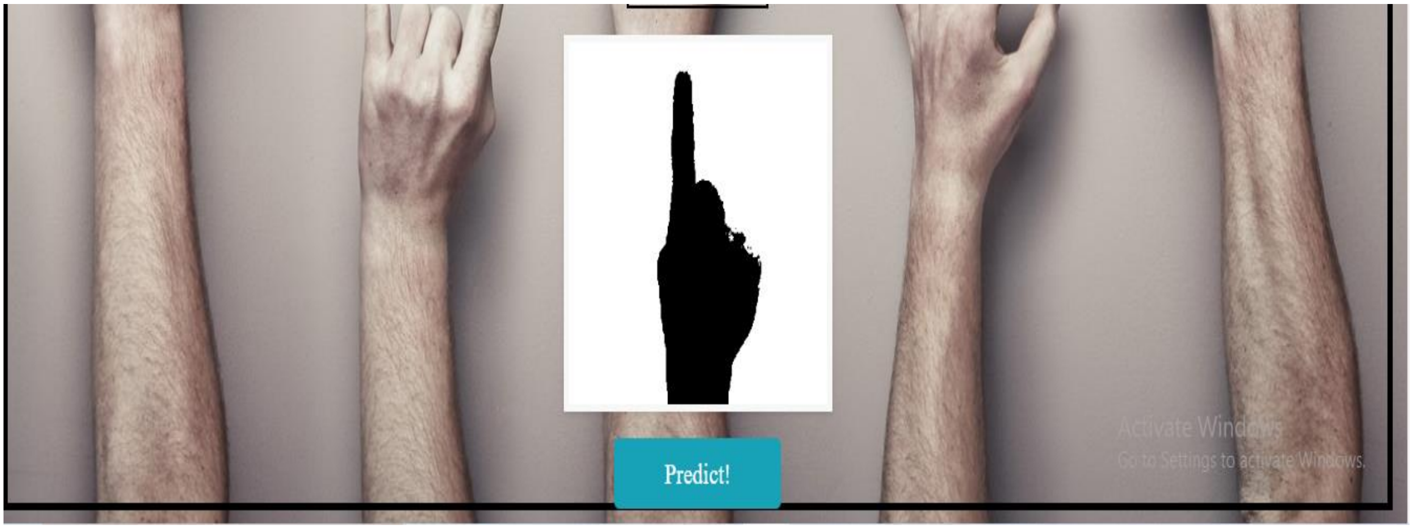
Hand Gesture recognition system provides us an innovative, natural, user friendly way of interaction with the computer which is more familiar to the human beings. In our project, the hand region is extracted from the background by using Region of intrest. Then, we will be predicting the labels based 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.



Hand Gesture Recognition

Provide an image for which you want to perform various operations

Upload Image Here



7. ADVANTAGES & DISADVANTAGES

Advantages:

- know fundamental concepts and techniques of Convolutional Neural Network.
- gain a broad understanding of image data.
- Know how to pre-process/clean the data using different data preprocessing techniques.
- know how to build a web application using Flask framework.

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 have established the application for a gesture-based tool for sterile browsing of radiology images using IBM. Humans are able to recognize body and sign language easily. This tool is also easy to use and is quicker than the regular method of using mouse/keyboard. It also does not require the user to have any device on them to use it.

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. BIBILOGRAPHY

A gesture-based tool for sterile browsing of radiology images - PubMed (nih.gov)

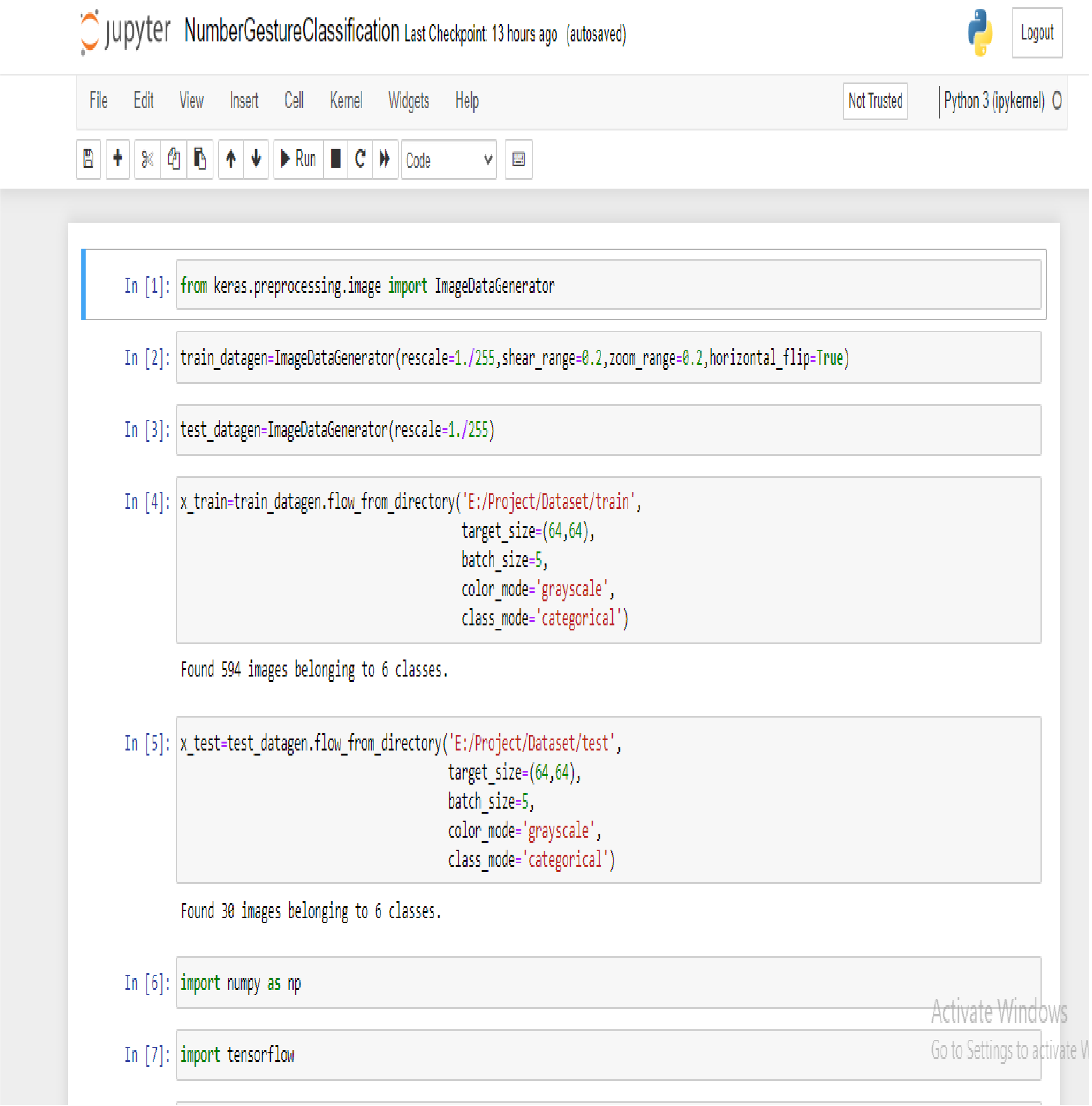
https://github.com/Guided-Projects/University_Admission_Prediction

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

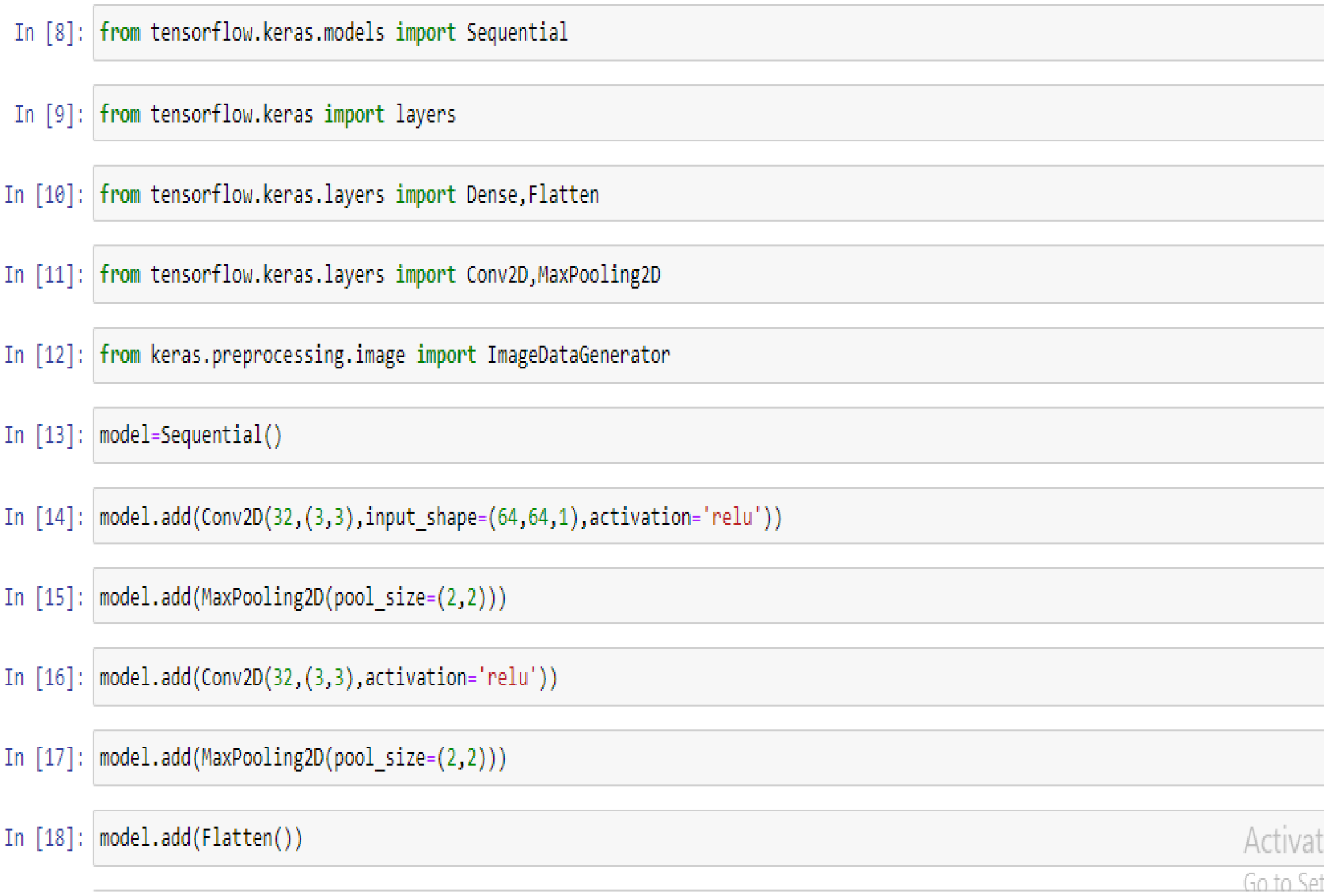
Smartinternz Website: https://smartinternz.com/Student/guided_project_info/319049#

APPENDIX

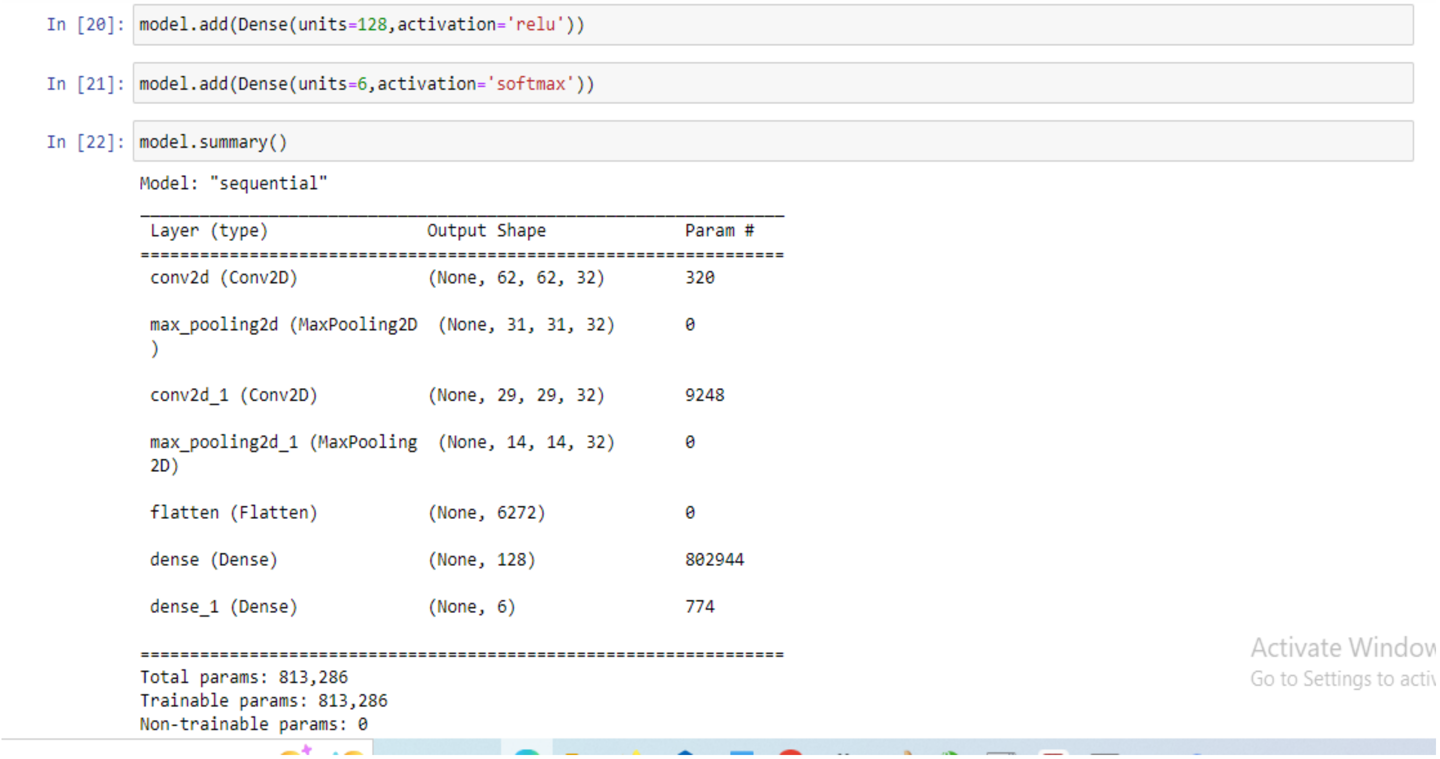
Source Code


jupyter NumberGestureClassification Last Checkpoint: 13 hours ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Activate Windows
Go to Settings to activate W

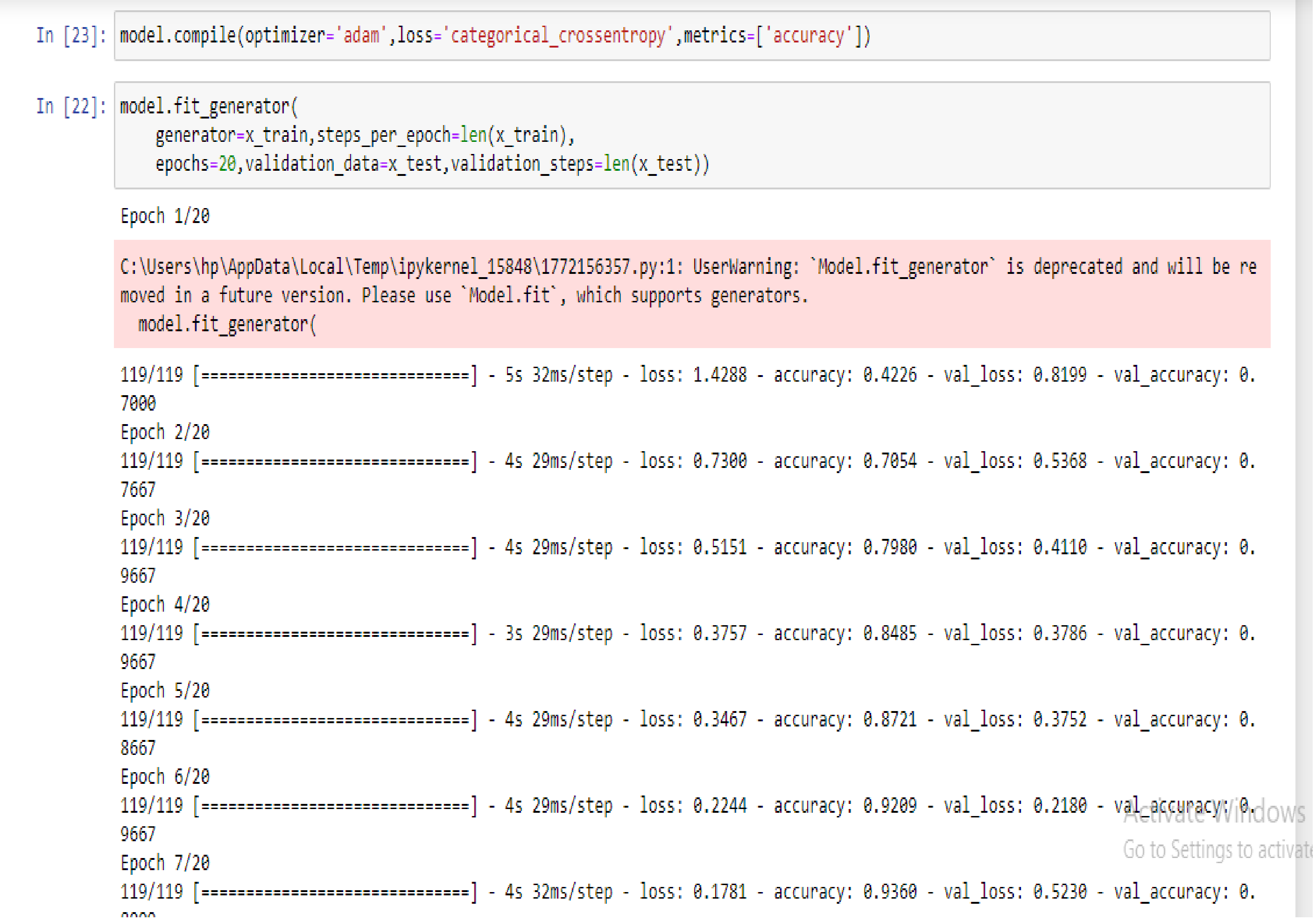
```
In [1]: from keras.preprocessing.image import ImageDataGenerator

In [2]: train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)

In [3]: test_datagen=ImageDataGenerator(rescale=1./255)

In [4]: x_train=train_datagen.flow_from_directory('E:/Project/Dataset/train',
                                                 target_size=(64,64),
                                                 batch_size=5,
                                                 color_mode='grayscale',
                                                 class_mode='categorical')

        Found 594 images belonging to 6 classes.

In [5]: x_test=test_datagen.flow_from_directory('E:/Project/Dataset/test',
                                               target_size=(64,64),
                                               batch_size=5,
                                               color_mode='grayscale',
                                               class_mode='categorical')

        Found 30 images belonging to 6 classes.

In [6]: import numpy as np

In [7]: import tensorflow
```



```
In [8]: from tensorflow.keras.models import Sequential

In [9]: from tensorflow.keras import layers

In [10]: from tensorflow.keras.layers import Dense,Flatten

In [11]: from tensorflow.keras.layers import Conv2D,MaxPooling2D

In [12]: from keras.preprocessing.image import ImageDataGenerator

In [13]: model=Sequential()

In [14]: model.add(Conv2D(32,(3,3),input_shape=(64,64,1),activation='relu'))

In [15]: model.add(MaxPooling2D(pool_size=(2,2)))

In [16]: model.add(Conv2D(32,(3,3),activation='relu'))

In [17]: model.add(MaxPooling2D(pool_size=(2,2)))

In [18]: model.add(Flatten())
```



```
In [20]: model.add(Dense(units=128,activation='relu'))

In [21]: model.add(Dense(units=6,activation='softmax'))

In [22]: 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
```



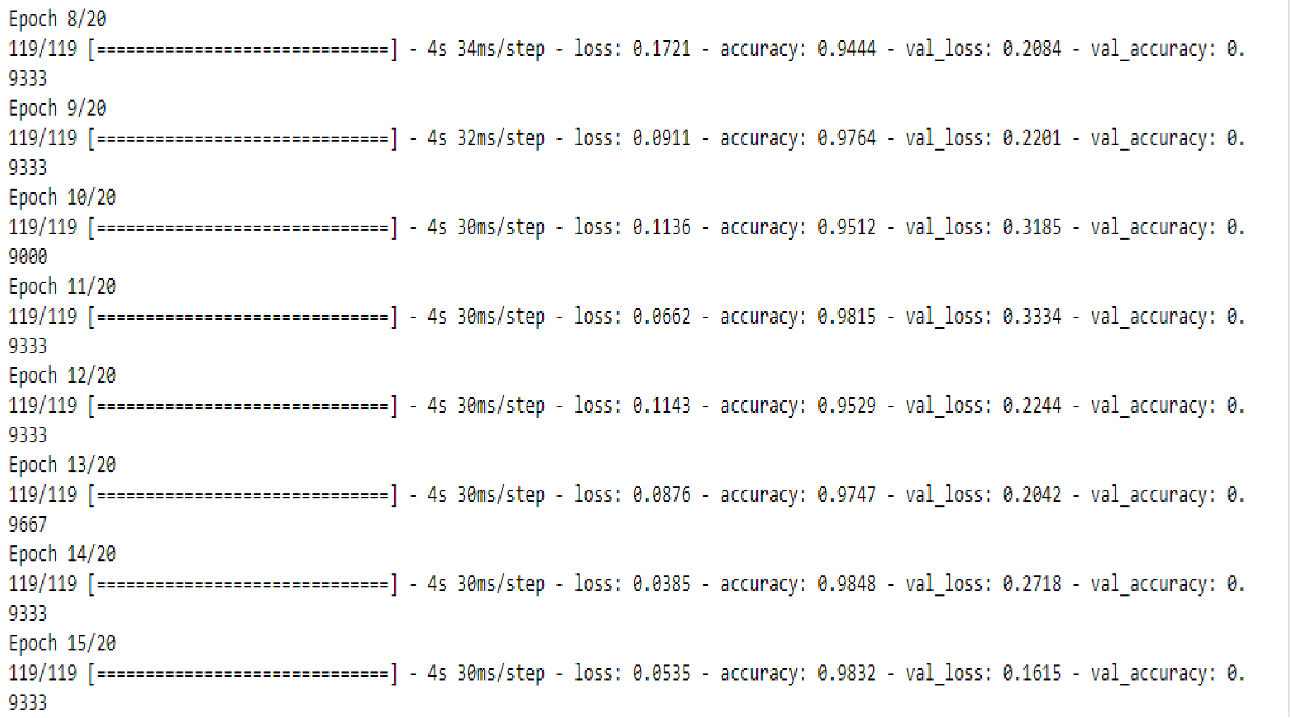
```
In [23]: model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
```

```
In [22]: model.fit_generator(
             generator=x_train,steps_per_epoch=len(x_train),
             epochs=20,validation_data=x_test,validation_steps=len(x_test))
```

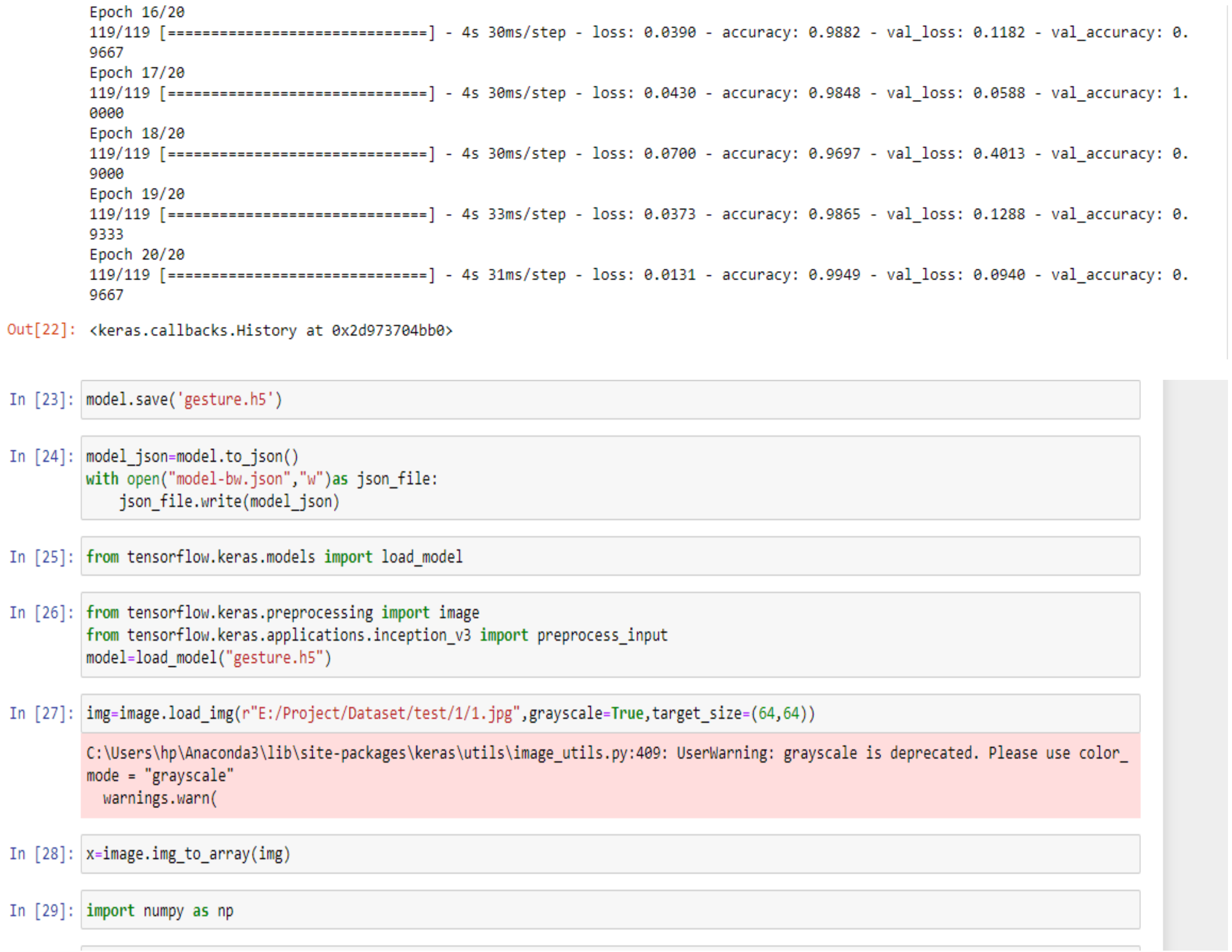
```
Epoch 1/20
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_15848\1772156357.py:1: UserWarning: `Model.fit_generator` is deprecated and will be re
moved in a future version. Please use `Model.fit`, which supports generators.
  model.fit_generator(
```

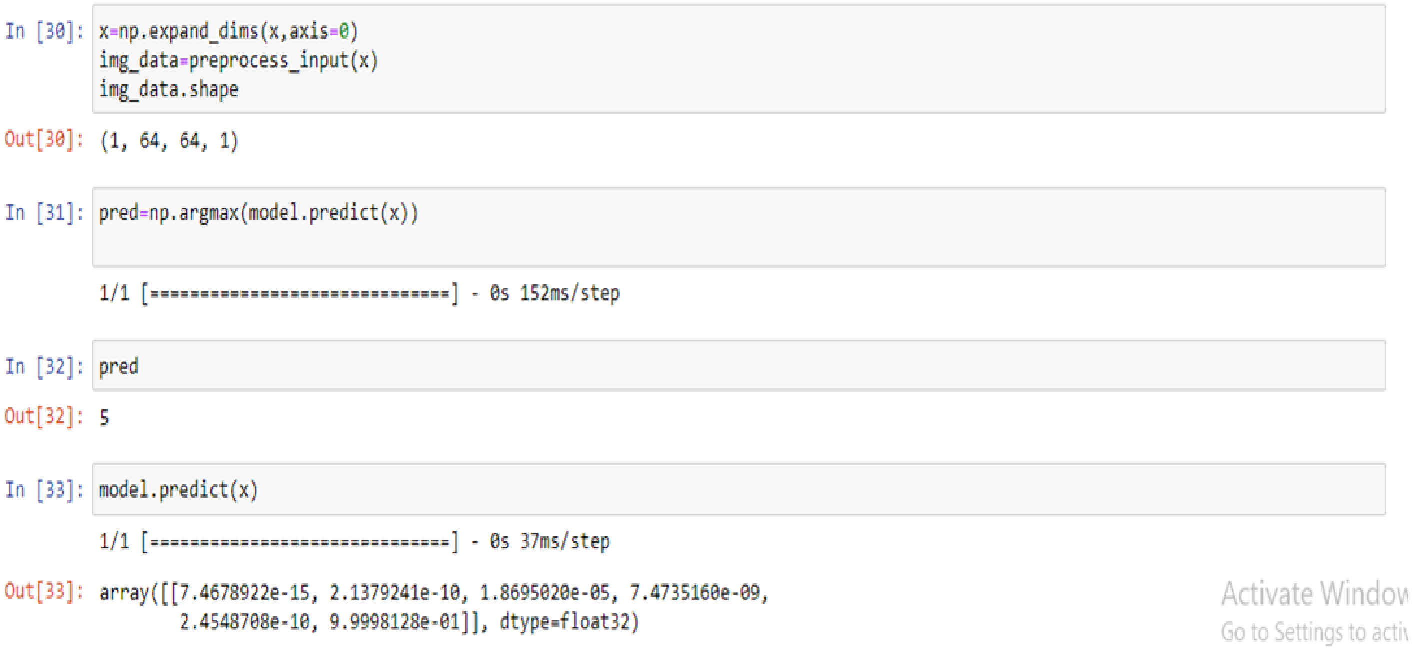
```
119/119 [==============================] - 5s 32ms/step - loss: 1.4288 - accuracy: 0.4226 - val_loss: 0.8199 - val_accuracy: 0.
7000
Epoch 2/20
119/119 [==============================] - 4s 29ms/step - loss: 0.7300 - accuracy: 0.7054 - val_loss: 0.5368 - val_accuracy: 0.
7667
Epoch 3/20
119/119 [==============================] - 4s 29ms/step - loss: 0.5151 - accuracy: 0.7980 - val_loss: 0.4110 - val_accuracy: 0.
9667
Epoch 4/20
119/119 [==============================] - 3s 29ms/step - loss: 0.3757 - accuracy: 0.8485 - val_loss: 0.3786 - val_accuracy: 0.
9667
Epoch 5/20
119/119 [==============================] - 4s 29ms/step - loss: 0.3467 - accuracy: 0.8721 - val_loss: 0.3752 - val_accuracy: 0.
8667
Epoch 6/20
119/119 [==============================] - 4s 29ms/step - loss: 0.2244 - accuracy: 0.9209 - val_loss: 0.2180 - val_accuracy: 0.
9667
Epoch 7/20
119/119 [==============================] - 4s 32ms/step - loss: 0.1781 - accuracy: 0.9360 - val_loss: 0.5230 - val_accuracy: 0.
```



```
Epoch 8/20
119/119 [==============================] - 4s 34ms/step - loss: 0.1721 - accuracy: 0.9444 - val_loss: 0.2084 - val_accuracy: 0.
9333
Epoch 9/20
119/119 [==============================] - 4s 32ms/step - loss: 0.0911 - accuracy: 0.9764 - val_loss: 0.2201 - val_accuracy: 0.
9333
Epoch 10/20
119/119 [==============================] - 4s 30ms/step - loss: 0.1136 - accuracy: 0.9512 - val_loss: 0.3185 - val_accuracy: 0.
9000
Epoch 11/20
119/119 [==============================] - 4s 30ms/step - loss: 0.0662 - accuracy: 0.9815 - val_loss: 0.3334 - val_accuracy: 0.
9333
Epoch 12/20
119/119 [==============================] - 4s 30ms/step - loss: 0.1143 - accuracy: 0.9529 - val_loss: 0.2244 - val_accuracy: 0.
9333
Epoch 13/20
119/119 [==============================] - 4s 30ms/step - loss: 0.0876 - accuracy: 0.9747 - val_loss: 0.2042 - val_accuracy: 0.
9667
Epoch 14/20
119/119 [==============================] - 4s 30ms/step - loss: 0.0385 - accuracy: 0.9848 - val_loss: 0.2718 - val_accuracy: 0.
9333
Epoch 15/20
119/119 [==============================] - 4s 30ms/step - loss: 0.0535 - accuracy: 0.9832 - val_loss: 0.1615 - val_accuracy: 0.
9333
```



```
Epoch 16/20
119/119 [==============================] - 4s 30ms/step - loss: 0.0390 - accuracy: 0.9882 - val_loss: 0.1182 - val_accuracy: 0.
9667
Epoch 17/20
119/119 [==============================] - 4s 30ms/step - loss: 0.0430 - accuracy: 0.9848 - val_loss: 0.0588 - val_accuracy: 1.
0000
Epoch 18/20
119/119 [==============================] - 4s 30ms/step - loss: 0.0700 - accuracy: 0.9697 - val_loss: 0.4013 - val_accuracy: 0.
9000
Epoch 19/20
119/119 [==============================] - 4s 33ms/step - loss: 0.0373 - accuracy: 0.9865 - val_loss: 0.1288 - val_accuracy: 0.
9333
Epoch 20/20
119/119 [==============================] - 4s 31ms/step - loss: 0.0131 - accuracy: 0.9949 - val_loss: 0.0940 - val_accuracy: 0.
9667

Out[22]: <keras.callbacks.History at 0x2d973704bb0>

In [23]: model.save('gesture.h5')

In [24]: model_json=model.to_json()
         with open("model-bw.json","w")as json_file:
             json_file.write(model_json)

In [25]: from tensorflow.keras.models import load_model

In [26]: from tensorflow.keras.preprocessing import image
         from tensorflow.keras.applications.inception_v3 import preprocess_input
         model=load_model("gesture.h5")

In [27]: img=image.load_img(r"E:/Project/Dataset/test/1/1.jpg",grayscale=True,target_size=(64,64))

         C:\Users\hp\Anaconda3\lib\site-packages\keras\utils\image_utils.py:409: UserWarning: grayscale is deprecated. Please use color_
         mode = "grayscale"
           warnings.warn(

In [28]: x=image.img_to_array(img)

In [29]: import numpy as np
```



```
In [30]: x=np.expand_dims(x,axis=0)
         img_data=preprocess_input(x)
         img_data.shape

Out[30]: (1, 64, 64, 1)

In [31]: pred=np.argmax(model.predict(x))

         1/1 [==============================] - 0s 152ms/step

In [32]: pred

Out[32]: 5

In [33]: model.predict(x)

         1/1 [==============================] - 0s 37ms/step

Out[33]: array([[7.4678922e-15, 2.1379241e-10, 1.8695020e-05, 7.4735160e-09,
                 2.4548708e-10, 9.9998128e-01]], dtype=float32)
```