COMMUNICATION THROUGH GESTURES

1.1 Introduction

Artificial Intelligence (AI) is the **intelligence of machines** and the branch of computer science which aims to create it. Artificial Intelligence (AI) is the study of how computer systems can **simulate intelligent processes** such as **learning, reasoning, and understanding symbolic information** in context. The theory and development of computer systems able to perform tasks normally requiring human intelligence, such as **visual perception**, **speech recognition**, **decision making** and **translation between languages**.

1.2 Objectives of Research

- Problem Solving
- Machine learning
- Language processing
- Motion and perception
- Social intelligence
- Creativity

1.3 Problem Statement

Touchless hand gesture recognition systems are becoming important in automotive user interfaces as they improve safety and comfort. Various computer vision algorithms have employed color and depth cameras for hand gesture recognition, but robust classification of gestures from different subjects performed under widely varying lighting conditions is still challenging. We propose an algorithm for hand gesture recognition from challenging depth and intensity data using Convolutional Neural Networks.

2. Literature Review

The essential aim of building hand gesture recognition systems is to create a natural interaction between humans and a computer where the recognized gestures can be used for controlling a robot or conveying meaningful information. Human computer interaction (HCI) also named Man-Machine Interaction (MMI) refers to the relation between the human and the computer, or more precisely the machine. There are two main characteristics should be deemed when designing a HCI system: functionality and usability. **System functionality** refers to the set of functions or services that the system equips to the users, while **system usability** refers to the level and scope that the system can operate and perform specific user purposes efficiently. The system that attains a suitable balance between these concepts considered as influential performance and powerful system. Gestures can be **static** (posture or certain pose), which requires less computational complexity, or **dynamic** (sequence of postures), which are more complex but suitable for real time environments. Different methods have been proposed for acquiring information necessary for recognition gestures system. Some methods use additional hardware devices such as data glove devices and color markers to easily extract comprehensive description of gesture features. Other methods are based on the use of the appearance of the hand and the skin color to segment the hand and extract necessary features. Some recent reviews explained gesture recognition system applications and its growing importance in our life especially for **Human-Computer Interaction (HCI)**, **Robot control**, **games**, and **surveillance**, using different tools and algorithms. This work demonstrates the advancement of gesture recognition systems, with the discussion of different stages required to build a complete system with fewer errors using a different algorithm.

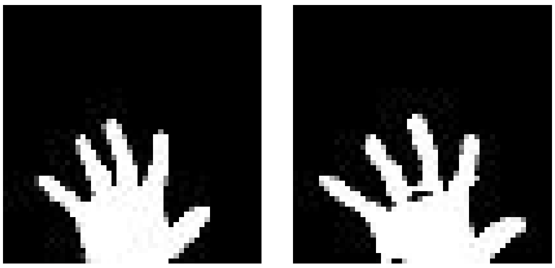
3. Data Collection

The Data collection process turned out to be the most important step, since finding the right dataset is crucial to the performance and accuracy of the CNN model. After various datasets were tried and tested with, one dataset found on Github.com. This dataset worked out to train the model in the right way to produce the desired results and a good accuracy rate.

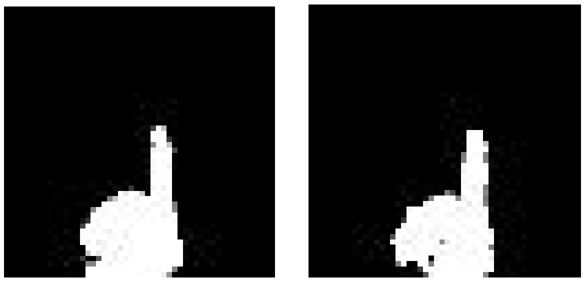
4. Methodology

4.1 Exploratory Data Analysis

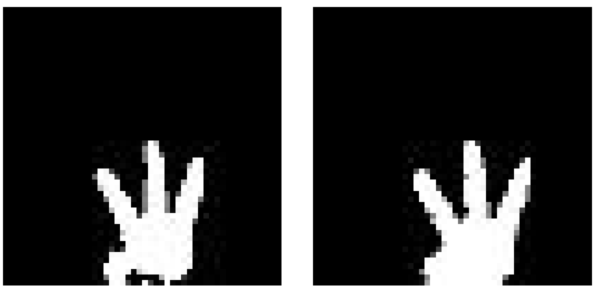
4.1.1 Figures and sample images



Class 0



Class 1

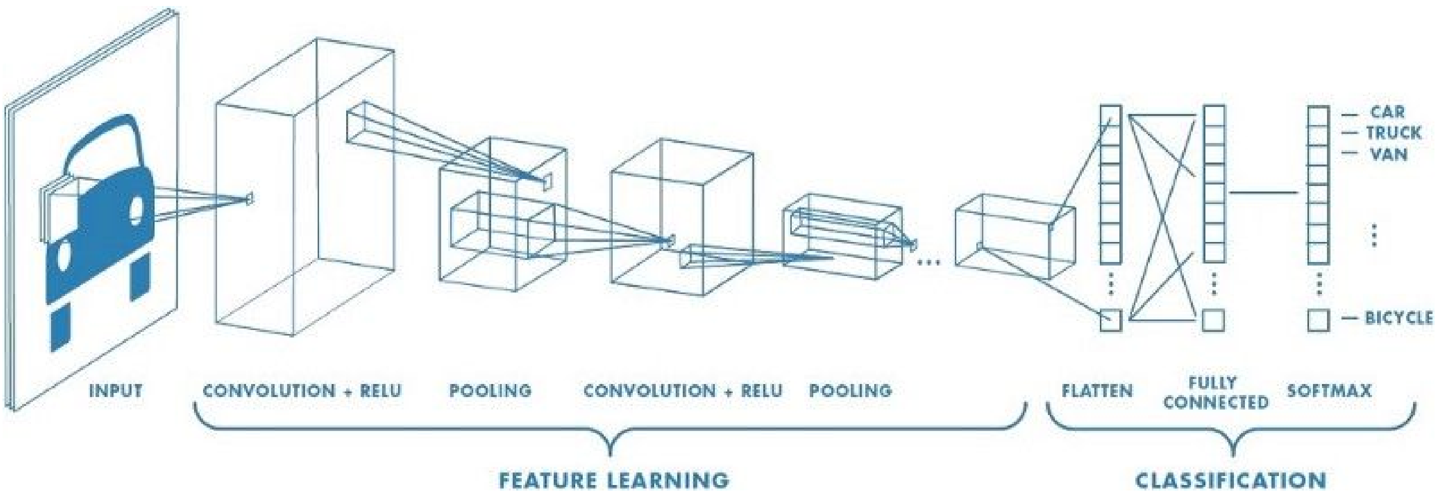


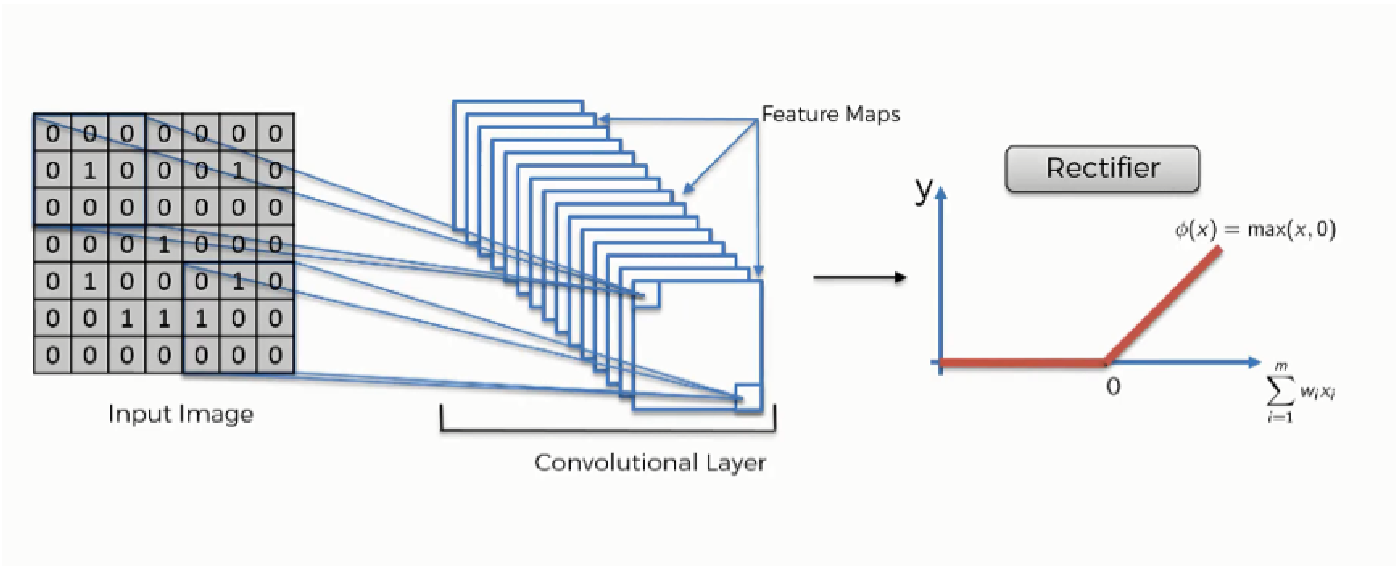
Class 2

Class 3

4.2 Data Modelling

Overview of the Convolutional Neural Network Model





Working CNN Code

In [1]:
```
#Importing the libraries.
import numpy as np
import pandas as pd
```

In [2]:
```
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
import tensorflow as tf
```

```
Using TensorFlow backend.
```

In [3]:
```
#Initializing the model.
model=Sequential()
```

In [4]:
```
model.add(Conv2D(128,3,3,input_shape=(128,128,3),activation='relu'))

#128-no.of feature extraction,for better extraction
```

```
WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\framework\op_def_library.py:263: colocate_
with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: UserWarning: Update your `Conv2D` call to the Keras 2 API:
`Conv2D(128, (3, 3), input_shape=(128, 128,..., activation="relu")`
```


```
`Conv2D(128, (3, 3), input_shape=(128, 128,..., activation="relu")`
```


```
  This is separate from the ipykernel package so we can avoid doing imports until
```

In [5]:
```
#Adding maxpooling layer here.

model.add(MaxPooling2D(pool_size=(2,2)))
```

In [6]:
```
#Adding a Flatten layer to the cnn which converts many dimensions many dimensions into 1-Dimensional.

model.add(Flatten())
```

In [7]:
```
#Hidden layer of ANN.

model.add(Dense(output_dim=128,activation='relu',init='random_uniform'))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: UserWarning: Update your `Dense` call to the Keras 2 API: `
Dense(activation="relu", units=128, kernel_initializer="random_uniform")`
  This is separate from the ipykernel package so we can avoid doing imports until
```

In [8]:
```
#Output layer of ANN.
# As the data is categorical,we use 'softmax' as the activation function.

model.add(Dense(output_dim=4,activation='softmax',init='random_uniform'))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: UserWarning: Update your `Dense` call to the Keras 2 API: `
Dense(activation="softmax", units=4, kernel_initializer="random_uniform")`
  This is separate from the ipykernel package so we can avoid doing imports until
```

In [9]:
```
#We use 'categorical_crossentropy' as loss.
```

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

In [10]:
```
#Importing libraries.
from keras.preprocessing.image import ImageDataGenerator
```

In [11]:
```
#Steps for rescaling the images into the range of 0 to 1.

train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
test_datagen = ImageDataGenerator(rescale=1./255)
```

In [12]:
```
#Splitting the data into x_train and x_test variables for further training and testing of the data.

x_train=train_datagen.flow_from_directory("C:/Users/Sai surya teja/Desktop/last/train_set",target_size=(128,128),batch_size=10,c
x_test=test_datagen.flow_from_directory("C:/Users/Sai surya teja/Desktop/last/test_set",target_size=(128,128),batch_size=10,clas
```

```
Found 3100 images belonging to 4 classes.
Found 900 images belonging to 4 classes.
```

In [13]:
```
#classes in the dataset.

print(x_train.class_indices)
```

```
{'hello': 0, 'one': 1, 'three': 2, 'victory': 3}
```

In [14]:
```
#Training the model.

model.fit_generator(x_train, samples_per_epoch = 3100, epochs = 10, validation_data = x_test, nb_val_samples = 900)
```


```
model.fit_generator(x_train, samples_per_epoch = 3100, epochs = 10, validation_data = x_test, nb_val_samples = 900)
```


```
WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\ops\math_ops.py:3066: to_int32 (from tenso
rflow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: UserWarning: The semantics of the Keras 2 argument `steps_p
er_epoch` is not the same as the Keras 1 argument `samples_per_epoch`. `steps_per_epoch` is the number of batches to draw from
the generator at each epoch. Basically steps_per_epoch = samples_per_epoch/batch_size. Similarly `nb_val_samples`->`validation_
steps` and `val_samples`->`steps` arguments have changed. Update your method calls accordingly.
  """Entry point for launching an IPython kernel.
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: UserWarning: Update your `fit_generator` call to the Keras
2 API: `fit_generator(<keras_pre..., epochs=10, validation_data=<keras_pre..., steps_per_epoch=310, validation_steps=900)`
  """Entry point for launching an IPython kernel.
```

```
Epoch 1/10
310/310 [==============================] - 448s 1s/step - loss: 0.4015 - acc: 0.8594 - val_loss: 0.4258 - val_acc: 0.8378
Epoch 2/10
310/310 [==============================] - 675s 2s/step - loss: 0.1222 - acc: 0.9645 - val_loss: 0.3401 - val_acc: 0.8778
Epoch 3/10
310/310 [==============================] - 469s 2s/step - loss: 0.0442 - acc: 0.9868 - val_loss: 0.5288 - val_acc: 0.8600
Epoch 4/10
310/310 [==============================] - 749s 2s/step - loss: 0.0361 - acc: 0.9874 - val_loss: 0.5622 - val_acc: 0.8700
Epoch 5/10
310/310 [==============================] - 738s 2s/step - loss: 0.0345 - acc: 0.9871 - val_loss: 0.3871 - val_acc: 0.8789
Epoch 6/10
310/310 [==============================] - 1297s 4s/step - loss: 0.0171 - acc: 0.9952 - val_loss: 0.6673 - val_acc: 0.8489
Epoch 7/10
310/310 [==============================] - 579s 2s/step - loss: 0.0128 - acc: 0.9961 - val_loss: 0.3202 - val_acc: 0.9178
Epoch 8/10
310/310 [==============================] - 378s 1s/step - loss: 0.0322 - acc: 0.9887 - val_loss: 0.3778 - val_acc: 0.8878
```

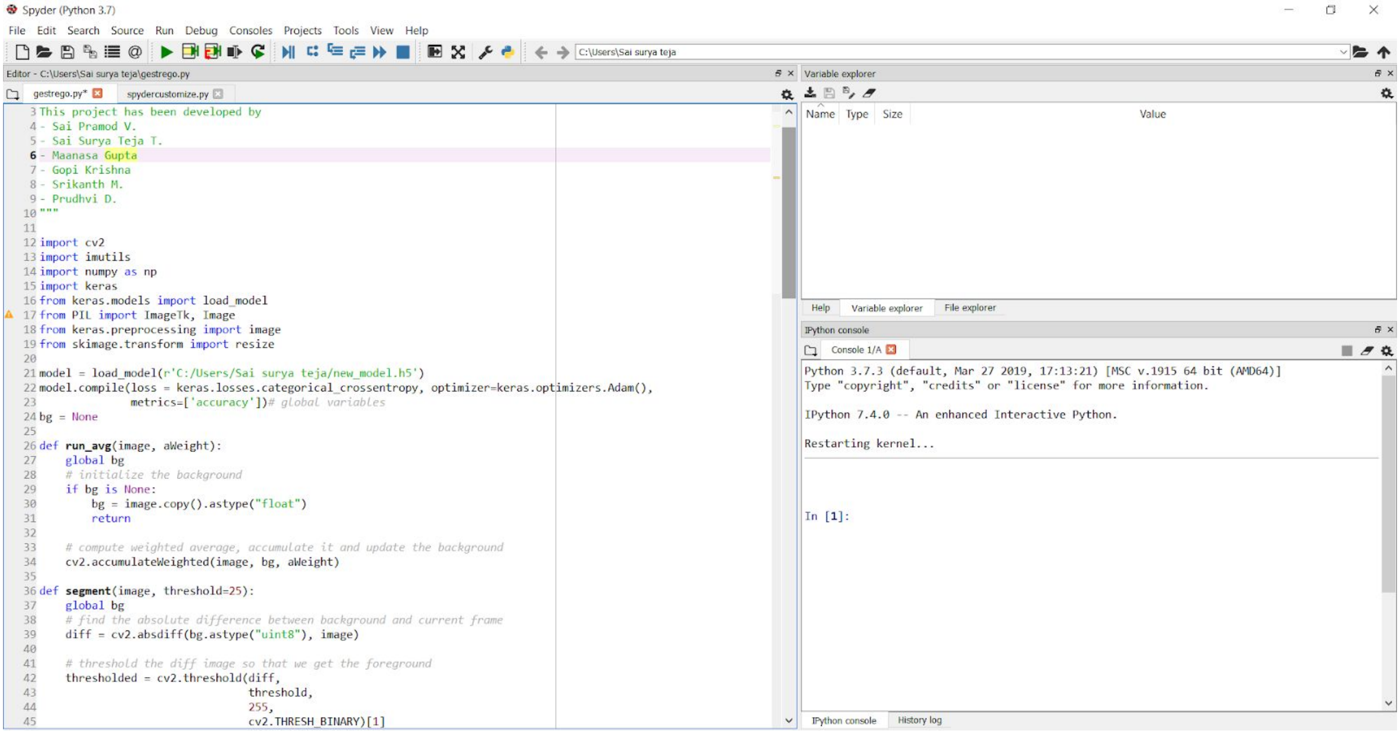

```
steps` and `val_samples`->`steps` arguments have changed. Update your method calls accordingly.
  """Entry point for launching an IPython kernel.
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: UserWarning: Update your `fit_generator` call to the Keras
2 API: `fit_generator(<keras_pre..., epochs=10, validation_data=<keras_pre..., steps_per_epoch=310, validation_steps=900)`
  """Entry point for launching an IPython kernel.
```

```
Epoch 1/10
310/310 [==============================] - 448s 1s/step - loss: 0.4015 - acc: 0.8594 - val_loss: 0.4258 - val_acc: 0.8378
Epoch 2/10
310/310 [==============================] - 675s 2s/step - loss: 0.1222 - acc: 0.9645 - val_loss: 0.3401 - val_acc: 0.8778
Epoch 3/10
310/310 [==============================] - 469s 2s/step - loss: 0.0442 - acc: 0.9868 - val_loss: 0.5288 - val_acc: 0.8600
Epoch 4/10
310/310 [==============================] - 749s 2s/step - loss: 0.0361 - acc: 0.9874 - val_loss: 0.5622 - val_acc: 0.8700
Epoch 5/10
310/310 [==============================] - 738s 2s/step - loss: 0.0345 - acc: 0.9871 - val_loss: 0.3871 - val_acc: 0.8789
Epoch 6/10
310/310 [==============================] - 1297s 4s/step - loss: 0.0171 - acc: 0.9952 - val_loss: 0.6673 - val_acc: 0.8489
Epoch 7/10
310/310 [==============================] - 579s 2s/step - loss: 0.0128 - acc: 0.9961 - val_loss: 0.3202 - val_acc: 0.9178
Epoch 8/10
310/310 [==============================] - 378s 1s/step - loss: 0.0322 - acc: 0.9887 - val_loss: 0.3778 - val_acc: 0.8878
```

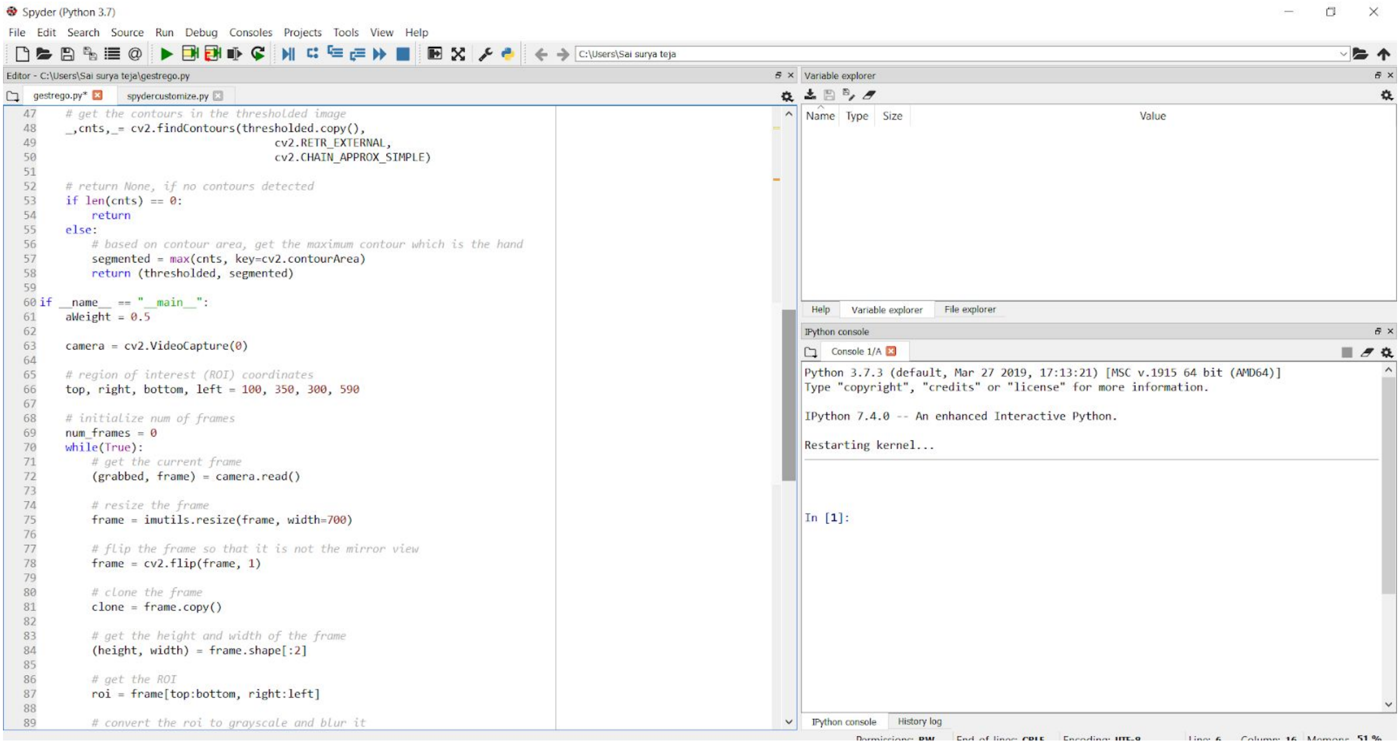

```
Epoch 9/10
310/310 [==============================] - 378s 1s/step - loss: 0.0297 - acc: 0.9894 - val_loss: 0.6068 - val_acc: 0.8622
Epoch 10/10
310/310 [==============================] - 393s 1s/step - loss: 0.0158 - acc: 0.9958 - val_loss: 0.4063 - val_acc: 0.9056
```

Out[14]:
```
<keras.callbacks.History at 0x21b5aaa0748>
```

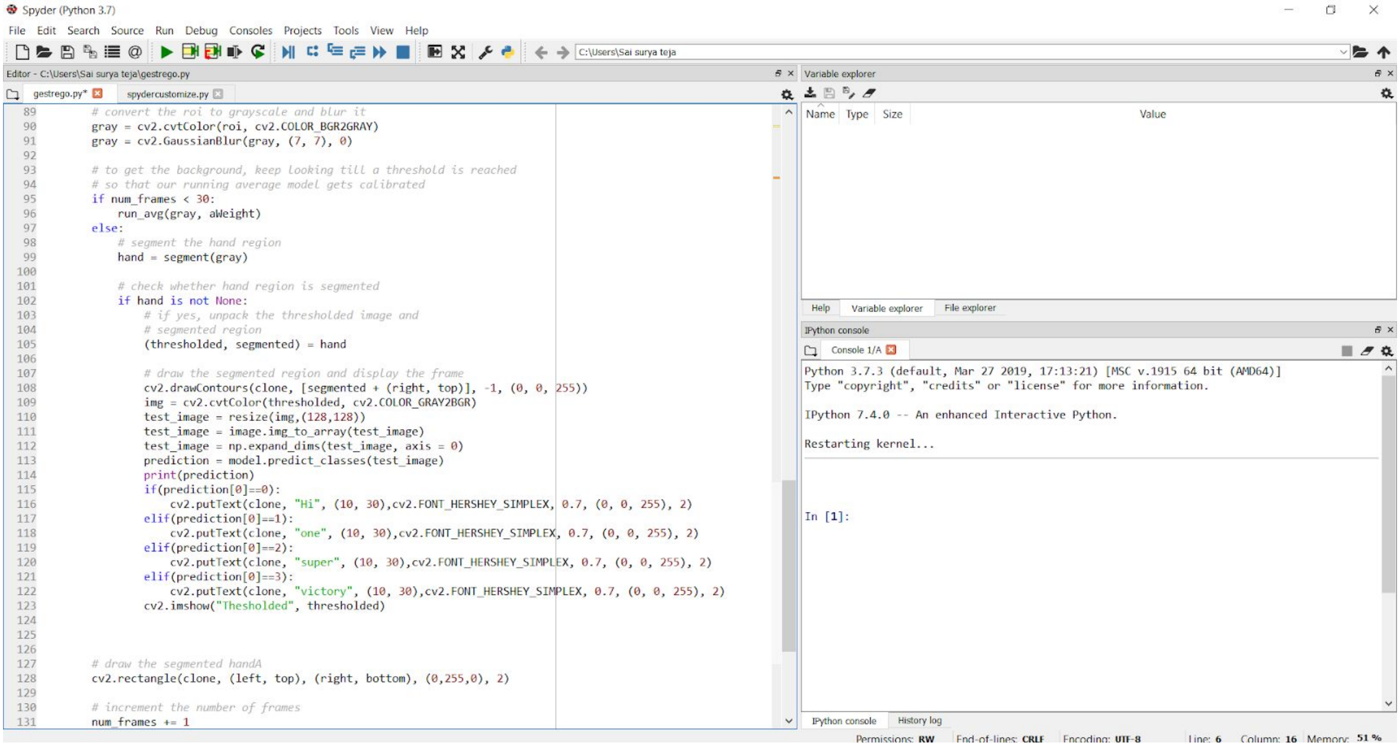
In [16]:
```
#Saving the model.

model.save('new_model.h5')
```



Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Sai surya teja
Editor - C:\Users\Sai surya teja\gestrego.py
gestrego.py*
spydercustomize.py
This project has been developed by
- Sai Pramod V.
- Sai Surya Teja T.
- Maanasa Gupta
- Gopi Krishna
- Srikanth M.
- Prudhvi D.
"""
import cv2
import imutils
import numpy as np
import keras
from keras.models import load_model
from PIL import ImageTk, Image
from keras.preprocessing import image
from skimage.transform import resize
model = load_model(r'C:/Users/Sai surya teja/new_model.h5')
model.compile(loss = keras.losses.categorical_crossentropy, optimizer=keras.optimizers.Adam(),
metrics=['accuracy'])# global variables
bg = None
def run_avg(image, aWeight):
global bg
# initialize the background
if bg is None:
bg = image.copy().astype("float")
return
# compute weighted average, accumulate it and update the background
cv2.accumulateWeighted(image, bg, aWeight)
def segment(image, threshold=25):
global bg
# find the absolute difference between background and current frame
diff = cv2.absdiff(bg.astype("uint8"), image)
# threshold the diff image so that we get the foreground
thresholded = cv2.threshold(diff,
threshold,
255,
cv2.THRESH_BINARY)[1]
Variable explorer
Name Type Size Value
Help Variable explorer File explorer
IPython console
Console 1/A
Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.
IPython 7.4.0 -- An enhanced Interactive Python.
Restarting kernel...
In [1]:
IPython console History log



Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Sai surya teja
Editor - C:\Users\Sai surya teja\gestrego.py
gestrego.py*
spydercustomize.py
get the contours in the thresholded image
,cnts, = cv2.findContours(thresholded.copy(),
cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
# return None, if no contours detected
if len(cnts) == 0:
return
else:
# based on contour area, get the maximum contour which is the hand
segmented = max(cnts, key=cv2.contourArea)
return (thresholded, segmented)
if __name__ == "__main__":
aWeight = 0.5
camera = cv2.VideoCapture(0)
# region of interest (ROI) coordinates
top, right, bottom, left = 100, 350, 300, 590
# initialize num of frames
num_frames = 0
while(True):
# get the current frame
(grabbed, frame) = camera.read()
# resize the frame
frame = imutils.resize(frame, width=700)
# flip the frame so that it is not the mirror view
frame = cv2.flip(frame, 1)
# clone the frame
clone = frame.copy()
# get the height and width of the frame
(height, width) = frame.shape[:2]
# get the ROI
roi = frame[top:bottom, right:left]
# convert the roi to grayscale and blur it
Variable explorer
Name Type Size Value
Help Variable explorer File explorer
IPython console
Console 1/A
Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.
IPython 7.4.0 -- An enhanced Interactive Python.
Restarting kernel...
In [1]:
IPython console History log


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Sai surya teja
Editor - C:\Users\Sai surya teja\gestrego.py
gestrego.py*
spydercustomize.py
        # convert the roi to grayscale and blur it
        gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
        gray = cv2.GaussianBlur(gray, (7, 7), 0)

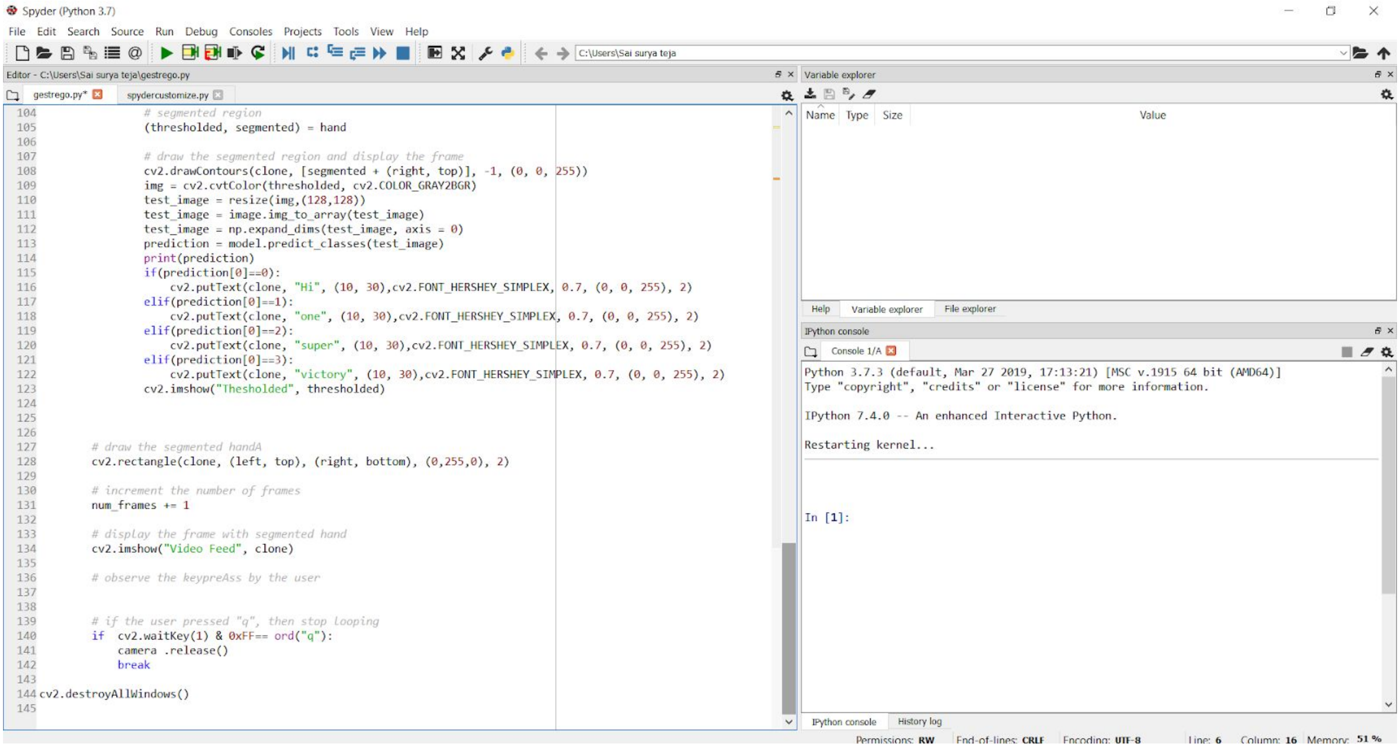
        # to get the background, keep looking till a threshold is reached
        # so that our running average model gets calibrated
        if num_frames < 30:
            run_avg(gray, aWeight)
        else:
            # segment the hand region
            hand = segment(gray)

            # check whether hand region is segmented
            if hand is not None:
                # if yes, unpack the thresholded image and
                # segmented region
                (thresholded, segmented) = hand

                # draw the segmented region and display the frame
                cv2.drawContours(clone, [segmented + (right, top)], -1, (0, 0, 255))
                img = cv2.cvtColor(thresholded, cv2.COLOR_GRAY2BGR)
                test_image = resize(img,(128,128))
                test_image = image.img_to_array(test_image)
                test_image = np.expand_dims(test_image, axis = 0)
                prediction = model.predict_classes(test_image)
                print(prediction)
                if(prediction[0]==0):
                    cv2.putText(clone, "Hi", (10, 30),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                elif(prediction[0]==1):
                    cv2.putText(clone, "one", (10, 30),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                elif(prediction[0]==2):
                    cv2.putText(clone, "super", (10, 30),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                elif(prediction[0]==3):
                    cv2.putText(clone, "victory", (10, 30),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                cv2.imshow("Thesholded", thresholded)

        # draw the segmented handA
        cv2.rectangle(clone, (left, top), (right, bottom), (0,255,0), 2)

        # increment the number of frames
        num_frames += 1
Variable explorer
Name Type Size Value
Help Variable explorer File explorer
IPython console
Console 1/A
Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.
IPython 7.4.0 -- An enhanced Interactive Python.
Restarting kernel...
In [1]:
IPython console History log
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 6 Column: 16 Memory: 51 %


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Sai surya teja
Editor - C:\Users\Sai surya teja\gestrego.py
gestrego.py*
spydercustomize.py
                # segmented region
                (thresholded, segmented) = hand

                # draw the segmented region and display the frame
                cv2.drawContours(clone, [segmented + (right, top)], -1, (0, 0, 255))
                img = cv2.cvtColor(thresholded, cv2.COLOR_GRAY2BGR)
                test_image = resize(img,(128,128))
                test_image = image.img_to_array(test_image)
                test_image = np.expand_dims(test_image, axis = 0)
                prediction = model.predict_classes(test_image)
                print(prediction)
                if(prediction[0]==0):
                    cv2.putText(clone, "Hi", (10, 30),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                elif(prediction[0]==1):
                    cv2.putText(clone, "one", (10, 30),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                elif(prediction[0]==2):
                    cv2.putText(clone, "super", (10, 30),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                elif(prediction[0]==3):
                    cv2.putText(clone, "victory", (10, 30),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                cv2.imshow("Thesholded", thresholded)

        # draw the segmented handA
        cv2.rectangle(clone, (left, top), (right, bottom), (0,255,0), 2)

        # increment the number of frames
        num_frames += 1

        # display the frame with segmented hand
        cv2.imshow("Video Feed", clone)

        # observe the keypreAss by the user

        # if the user pressed "q", then stop looping
        if cv2.waitKey(1) & 0xFF== ord("q"):
            camera .release()
            break

cv2.destroyAllWindows()
Variable explorer
Name Type Size Value
Help Variable explorer File explorer
IPython console
Console 1/A
Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.
IPython 7.4.0 -- An enhanced Interactive Python.
Restarting kernel...
In [1]:
IPython console History log
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 6 Column: 16 Memory: 51 %

5. Findings and Suggestions

The following sources have been used for reference and guidance throughout this project.

- ❖ Kaggle.com
- ❖ Github.com
- ❖ towardsdatascience.com

This project can be further enhanced and upgraded to create a real-time gesture recognition software that interacts with humans and acts in response to the gestures recognized by the algorithm. This algorithm can then be embedded into an interactive robot that assists human beings in day-to-day activities.

This algorithm can also be updated to play the audio of the intended gesture, thus rendering it very helpful in assisting people with speech and hearing disabilities.

6. Conclusion

CNNs showcase the incredible levels of control over performance that can be achieved by making effective use of theoretical and mathematical insights. Many real world problems are being efficiently tackled using CNNs, and MNIST represents a simple, “Hello World”-type use-case of this technique. More complex problems such as object and image recognition require the use of Deep Neural Networks with millions of parameters to obtain state-of-the-art results.

Furthermore, applications are not limited to computer vision. The win of Google’s AlphaGo Project over Lee Sedol in the Go game series in 2017 relied on a CNN at its core. The self-driving cars which, in the coming years, will arguably become a regular sight on our streets, rely on CNNs for steering. Google even held an art-show for imagery created by its DeepDream project that showcased beautiful works of art created by visualizing the transformations of the network!

Thus a Machine Learning researcher or engineer in today’s world can rejoice at the technological melange of techniques at her disposal, among which an in-depth understanding of CNNs is both indispensable and empowering.
