Hand Gesture Recognition Sytem

Using Deep Learning Technique

What is the Objective of Project

Primary Objective of this project is to develop deep learning model to detect the hand gesture pattern. Our objective is to integrate DL and Image Processing technology and bring some useful application to people.

What are technologies involved in the project ?

- We are implementing our whole project in **Python Language,** specifically in *Python Version 3+*.
- We are doing our project based on prototype model.
- Number of machine learning libraries and framework are used, some of the important libraries are mentioned below:
    - **Keras**, which is an open-source neural-network library
    - **Tensorflow** is also used as part of Keras support library, as it is written using this.
    - **OpenCV,** which is popular image processing library
    - Any many **other small supporting** libraries

- We use **Anaconda** and **Spyder** as our developing tools.

What are possible application of this project ?

Well, There are many application we can extract from this model.

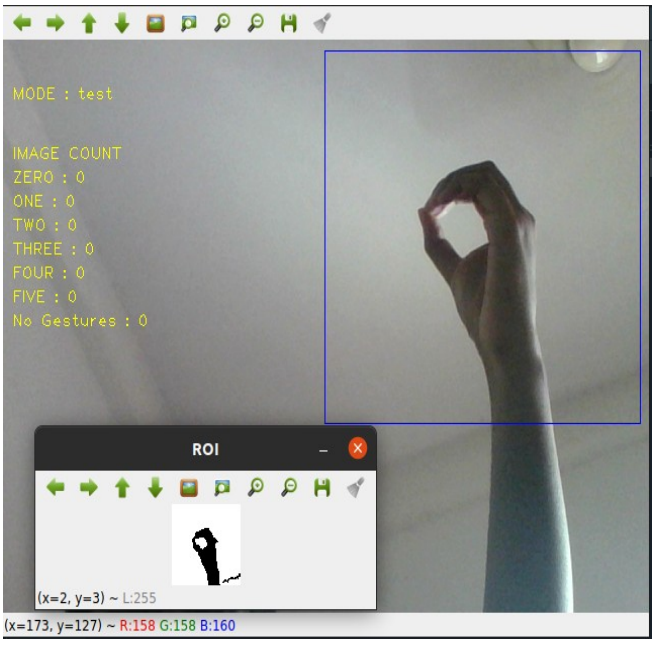
- We can deploy this hand gesture recognition system in robotics to control arms and movement of robots.
- We can also use this model in gaming as part of controller mechanism.

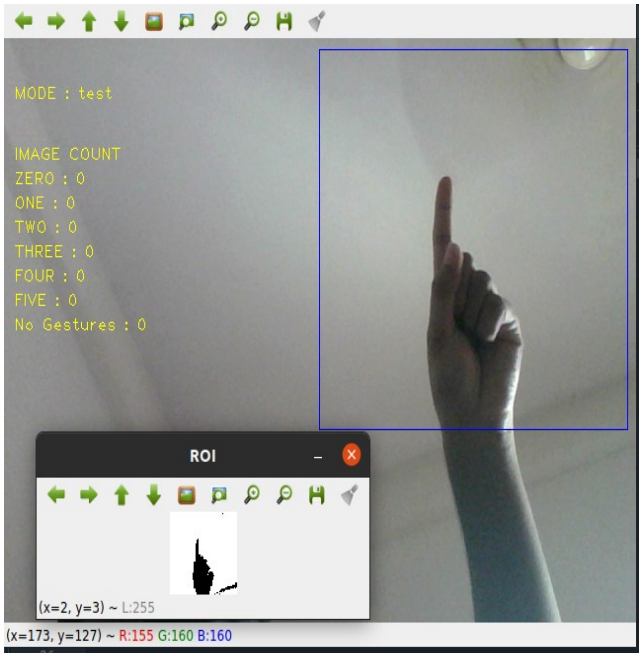
Implementation and code

We have created three python files
1. collect-data.py
2. train.py
3. predict.py

collect-data.py

```
# Train or test
mode = 'train'
directory = 'data/'+mode+'/'

cap = cv2.VideoCapture(0)

while True:
    _, frame = cap.read()
    # Simulating mirror image
    frame = cv2.flip(frame, 1)

    # Getting count of existing images
    count = {'zero': len(os.listdir(directory+"/0")),
             'one': len(os.listdir(directory+"/1")),
             'two': len(os.listdir(directory+"/2")),
             'three': len(os.listdir(directory+"/3")),
             'four': len(os.listdir(directory+"/4")),
             'five': len(os.listdir(directory+"/5")),
             'no_gestures': len(os.listdir(directory+"/no_gestures"))}

    # Printing the count in each set to the screen
    cv2.putText(frame, "MODE : "+mode, (10, 50), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
    cv2.putText(frame, "IMAGE COUNT", (10, 100), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
    cv2.putText(frame, "ZERO : "+str(count['zero']), (10, 120), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
    cv2.putText(frame, "ONE : "+str(count['one']), (10, 140), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
    cv2.putText(frame, "TWO : "+str(count['two']), (10, 160), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
    cv2.putText(frame, "THREE : "+str(count['three']), (10, 180), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
    cv2.putText(frame, "FOUR : "+str(count['four']), (10, 200), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
    cv2.putText(frame, "FIVE : "+str(count['five']), (10, 220), cv2.FONT_HERSHEY_PLAIN, 1, (0,255,255), 1)
```





train.py

Importing libraires

```
# Importing the Keras libraries and packages
import numpy as np
import keras
from keras import backend as K
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Dropout
from keras.layers import Dense
from keras.layers import Flatten

from keras.models import Sequential
```

```
model = Sequential()

# First convolution layer and pooling
model.add(Conv2D(32, (3,3), activation='relu', input_shape=(64,64,1)))
model.add(MaxPooling2D(2,2))
# Second convolution layer and pooling
model.add(Conv2D(64, (3,3), activation='relu'))
model.add(Conv2D(64, (3,3), activation='relu'))
# input_shape is going to be the pooled feature maps from the previous convolution
model.add(MaxPooling2D(2,2))

model.add(Conv2D(128, (3,3), activation='relu'))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(64, (3,3), activation='relu'))
model.add(MaxPooling2D(2,2))
# Flattening the layers
model.add(Flatten())
# Adding a fully connected layer
model.add(Dense(150, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(7, activation='softmax'))
```

- **Conv2D :-** used for convolution layer to create feature maps of 2D image
- **MaxPooling2D** :- does max pooling of 2D image
- **Dropout**:- used to drop some neurons in the hidden layer, so that the network can generalize enough as well as to avoid the dependence of neuron on a single neuron
- **Dense**:- Used to create a fully connected neural network
- **Flatten**:- used to unroll the convolutional layers into a 1-column vector to be used by a fully connected layer around the end of the network

Preprocessing

```
from keras.preprocessing.image import ImageDataGenerator

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

test_datagen = ImageDataGenerator(rescale=1./255)

training_set = train_datagen.flow_from_directory('data/train',
                                                 target_size=(64, 64),
                                                 batch_size=5,
                                                 color_mode='grayscale',
                                                 class_mode='categorical')

test_set = test_datagen.flow_from_directory('data/test',
                                            target_size=(64, 64),
                                            batch_size=5,
                                            color_mode='grayscale',
                                            class_mode='categorical')
model.fit_generator(
        training_set,
        steps_per_epoch=700, # No of images in training set
        epochs=10,
        validation_data=test_set,
        validation_steps=70)# No of images in test set
```

```
_________________________________________________________________
Found 700 images belonging to 7 classes.
Found 70 images belonging to 7 classes.
Epoch 1/10
700/700 [==============================] - 41s 58ms/step - loss: 1.2308 - accuracy: 0.4886 - val_loss: 0.3414 - val_accuracy: 0.7571
Epoch 2/10
700/700 [==============================] - 50s 72ms/step - loss: 0.5343 - accuracy: 0.7900 - val_loss: 0.5046 - val_accuracy: 0.8143
Epoch 3/10
700/700 [==============================] - 58s 82ms/step - loss: 0.2721 - accuracy: 0.9043 - val_loss: 0.5614 - val_accuracy: 0.8286
Epoch 4/10
700/700 [==============================] - 46s 65ms/step - loss: 0.2158 - accuracy: 0.9303 - val_loss: 2.5323 - val_accuracy: 0.8000
Epoch 5/10
700/700 [==============================] - 39s 56ms/step - loss: 0.1510 - accuracy: 0.9454 - val_loss: 0.0048 - val_accuracy: 0.8571
Epoch 6/10
700/700 [==============================] - 42s 60ms/step - loss: 0.1523 - accuracy: 0.9511 - val_loss: 1.6612 - val_accuracy: 0.8571
Epoch 7/10
700/700 [==============================] - 46s 65ms/step - loss: 0.1214 - accuracy: 0.9580 - val_loss: 0.0476 - val_accuracy: 0.9143
Epoch 8/10
700/700 [==============================] - 49s 69ms/step - loss: 0.1132 - accuracy: 0.9634 - val_loss: 0.9388 - val_accuracy: 0.8286
Epoch 9/10
700/700 [==============================] - 45s 64ms/step - loss: 0.0906 - accuracy: 0.9697 - val_loss: 0.0040 - val_accuracy: 0.8714
Epoch 10/10
700/700 [==============================] - 46s 65ms/step - loss: 0.0684 - accuracy: 0.9814 - val_loss: 0.7268 - val_accuracy: 0.8571
```

Saving the trained model

```
# Saving the model
model_json = model.to_json()
with open("model-bw.json", "w") as json_file:
    json_file.write(model_json)
model.save_weights('model-bw.h5')
```

predict.py

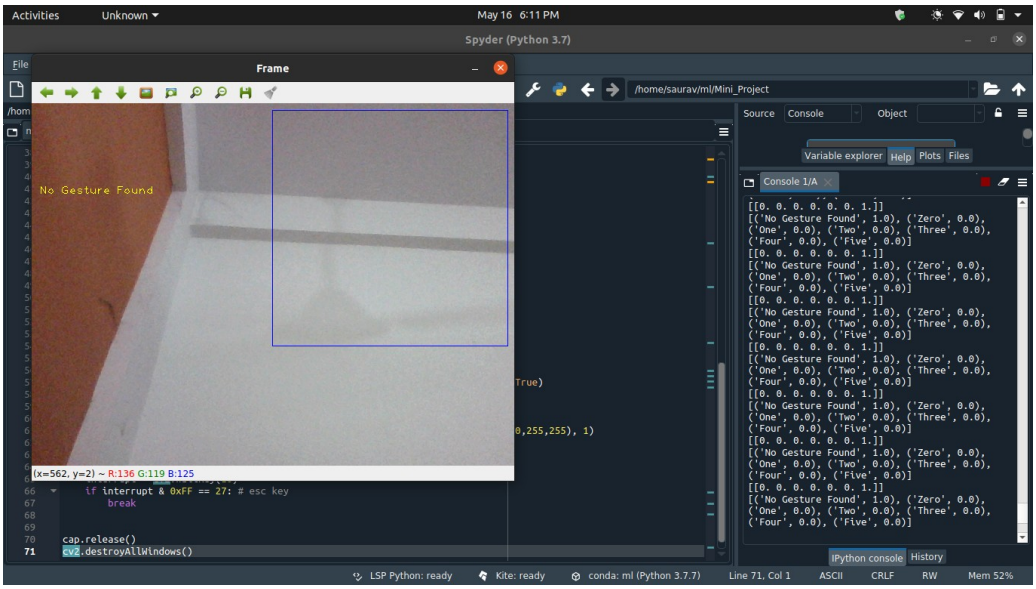
Loading the saved model

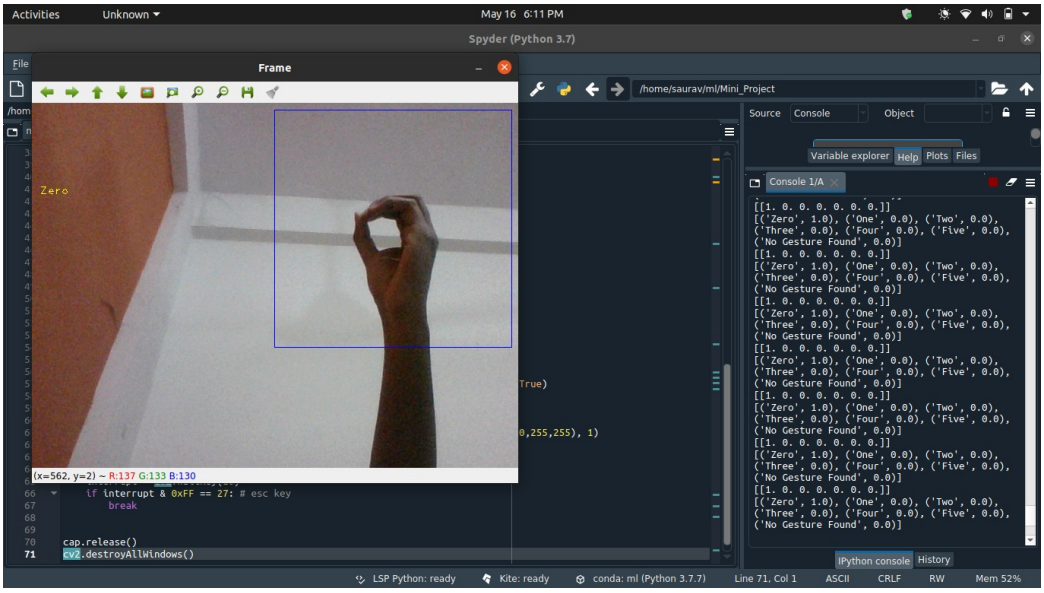
```
# Loading the model
json_file = open("model-bw.json", "r")
model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(model_json)
# load weights into new model
loaded_model.load_weights("model-bw.h5")
print("Loaded model from disk")
```

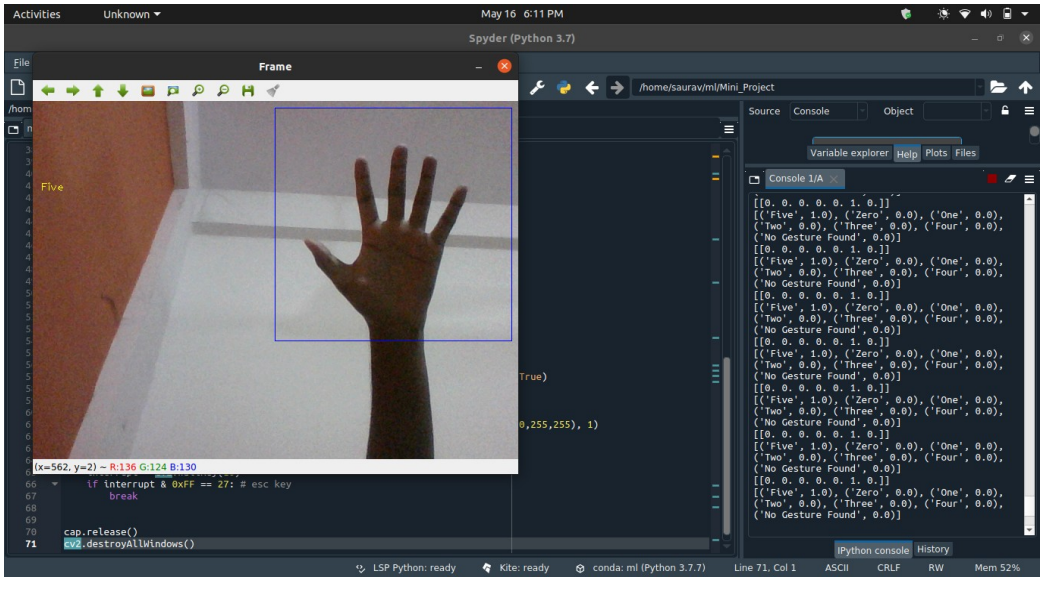
Prediction

```
while True:
    _, frame = cap.read()
    # 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 = loaded_model.predict(test_image.reshape(1, 64, 64, 1))

    print(result)

    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],
                  'No Gesture Found': result[0][6]}
```

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

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

    cv2.imshow("Frame", frame)

    interrupt = cv2.waitKey(10)
    if interrupt & 0xFF == 27: # esc key
        break


cap.release()
cv2.destroyAllWindows()
```


Activities
Unknown
May 16 6:11 PM
Spyder (Python 3.7)
Frame
No Gesture Found
(x=562, y=2) ~ R:136 G:119 B:125
if interrupt & 0xFF == 27: # esc key
break
cap.release()
cv2.destroyAllWindows()
/home/saurav/ml/Mini_Project
Source
Console
Object
Variable explorer
Help
Plots
Files
Console 1/A
[[0. 0. 0. 0. 0. 0. 1.]]
[('No Gesture Found', 1.0), ('Zero', 0.0), ('One', 0.0), ('Two', 0.0), ('Three', 0.0), ('Four', 0.0), ('Five', 0.0)]
IPython console
History
LSP Python: ready
Kite: ready
conda: ml (Python 3.7.7)
Line 71, Col 1
ASCII
CRLF
RW
Mem 52%


Activities
Unknown
May 16 6:11 PM
Spyder (Python 3.7)
Frame
Zero
(x=562, y=2) ~ R:137 G:133 B:130
if interrupt & 0xFF == 27: # esc key
break
cap.release()
cv2.destroyAllWindows()
/home/saurav/ml/Mini_Project
Source
Console
Object
Variable explorer
Help
Plots
Files
Console 1/A
[[1. 0. 0. 0. 0. 0. 0.]]
[('Zero', 1.0), ('One', 0.0), ('Two', 0.0), ('Three', 0.0), ('Four', 0.0), ('Five', 0.0), ('No Gesture Found', 0.0)]
IPython console
History
LSP Python: ready
Kite: ready
conda: ml (Python 3.7.7)
Line 71, Col 1
ASCII
CRLF
RW
Mem 52%


Activities
Unknown
May 16 6:11 PM
Spyder (Python 3.7)
Frame
Five
(x=562, y=2) ~ R:136 G:124 B:130
if interrupt & 0xFF == 27: # esc key
break
cap.release()
cv2.destroyAllWindows()
/home/saurav/ml/Mini_Project
Source
Console
Object
Variable explorer
Help
Plots
Files
Console 1/A
[[0. 0. 0. 0. 0. 1. 0.]]
[('Five', 1.0), ('Zero', 0.0), ('One', 0.0), ('Two', 0.0), ('Three', 0.0), ('Four', 0.0), ('No Gesture Found', 0.0)]
IPython console
History
LSP Python: ready
Kite: ready
conda: ml (Python 3.7.7)
Line 71, Col 1
ASCII
CRLF
RW
Mem 52%

Conclusion

In this way we have done our project and got satisfactory result. We also created interface using OpenCV to demonstrate model in live image data. This project has more application which can be achieved by implemeting on IOT Devices.