

Mini Project Report On Hand-sign recognition media control:

19.06.2021

Name: Hrithik Saini

University roll no.: 2013339

Problem Statement :

Build a Hand-sign recognition system that captures the real time hand signs the user makes through a webcam and classify them and send the commands to the media player to act accordingly..

Motivation for doing the project :

The main motivation for this project comes from a desire to control the system without touching it with just a wave of a hand and hence what better to control in a system rather than a media player where you have to do only few operations like play/pause, volume up, volume down, forward a file and go backwards. Hence, I wanted to learn a way such that I can make computer learn to classify between different gestures of mine and do the tasks accordingly.

Methodology Followed:

A **Convolutional Neural Network (ConvNet/CNN)** is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

Library Used:

The libraries used in this project are OpenCv, Numpy, Keras and pyautogui.

```
import numpy as np
from keras.models import model_from_json
import cv2
import pyautogui as pag
from keras.models import Sequential
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense
```

Collecting data:

To collect the images for the training of the model , I made a simple python script which had a region of interest where I made different hand signs and captured their images and stored them in their respective folders by processing them and making them of size 75 x 75 and making them of binary colors, that is black and white.

```
roi = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
_, roi = cv2.threshold(roi, 165, 255, cv2.THRESH_BINARY)
```

Building the model:

When the images are collected in data folder and within the test and train folders separately, it was time to make a classifier model. I used sequential model from keras to make the classifier object as we require plain stack of layers with one input and one output. Then I made CNN to extract the feature map from the data we used.

```
classifier = Sequential()

classifier.add(Convolution2D(32, (3, 3), input_shape=(75, 75, 1), activation='relu'))
classifier.add(MaxPooling2D(pool_size=(2, 2)))

classifier.add(Convolution2D(32, (3, 3), activation='relu'))
classifier.add(MaxPooling2D(pool_size=(2, 2)))

classifier.add(Flatten())

classifier.add(Dense(units=128, activation='relu'))
classifier.add(Dense(units=6, activation='softmax'))

classifier.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['a
```

Training the model :

Then I trained the model by getting the images in the dataset that we collected

```
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=(75, 75),
                                                batch_size=5,
                                                color_mode='grayscale',
                                                class_mode='categorical')

test_set = test_datagen.flow_from_directory('data/test',
                                            target_size=(75, 75),
                                            batch_size=5,
                                            color_mode='grayscale',
                                            class_mode='categorical')

classifier.fit(
    training_set,
    steps_per_epoch=len(training_set)//5,
    epochs=15,
    validation_data=test_set,
    validation_steps=len(test_set)//5)
```

Exporting json file:

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

Predicting the hand signs:

I made a dictionary with the respective values of the hand signs and used it for the prediction of the correct output

```
result = loaded_model.predict(roi.reshape(1, 75, 75, 1))
prediction = {'NONE': result[0][0],
             'UP': result[0][1],
             'TWO': result[0][2],
             'RIGHT': result[0][3],
             'LEFT': result[0][4],
             'PALM': result[0][5]}






prediction = sorted(prediction.items(), key=operator.itemgetter(1), reverse=True)
```

Key actions on the basis of predictions:

Then I mapped the keys using pyautogui on the basis of predictions to send hot key signals to the media player to control it.

```
if prediction[0][0]=='UP':
    pag.press('up')
elif prediction[0][0]=='TWO':
    pag.press('down')
elif prediction[0][0]=='RIGHT':
    pag.press('right')
elif prediction[0][0]=='LEFT':
    pag.press('left')
elif prediction[0][0]=='PALM':
    pag.press('space')
    time.sleep(1)
```

Signs and their actions:

-  → Volume up
-  → Volume down
-  → Make media go forward
-  → Make media go backward
-  → Play/Pause

THANK-YOU.....