ENS 492 – Graduation Project (Implementation)

FİNAL REPORT

Project Title: Autistic Soft Robot

Sabancı Üniversitesi

Group Members: Cem Akgüngör - 20139

Supervisor(s): Meltem Elitaş

Date: 22.12.2019

1. EXECUTİVE SUMMARY

This project aims to design and develop an Autistic Soft Robot (ASR) which will help autistic children to improve their individual skills and make friendship with them. The US Department of Health has a worldwide survey conducted. According to American Ministry of Health study, one in 59 children has autism [6]. However, education institutions are not enough to cover that number and these institutes are restricted with some topics for the education. Main purpose of ASR is to close this gap and provide educational and emotional supports to autistic children. Our robot aims to be friends with autistic children and help them in every aspects of their daily life. To do that, the robot should be able to classify the children and his/her emotions, and use right technics according to the children. On that purpose, ASR will be using machine learning and artificial intelligence technics in order to learn with children. In addition, the robot consists of soft manipulator, several sensors and actuators, and control systems. The robot will collect data from children with the help of cameras and sensors then it will use these data to classify some attributes. These data will help the robot to improve in teaching and understanding techniques of the robot. Since the robot requires technological equipment's, donation is required in order to carry out the project as a research. With helps of my instructor, funding constraint is overcome. Another constraint is that, there must be fast interaction between software and hardware parts of the robot. In addition, big data for machine learning algorithms will be processing while robot is working. That means powerful GPU and communication system is needed. We have implemented algorithms in both Amazon and Google Colab environments as will be explained in detailed in next sections. We have used Raspberry Pi and extracted python code there in order to overcome some problems. This helps the portability of the robot as well.

2. PROBLEM STATEMENT

As every children, autistic kids have educational rights with some precises. Main problem is about educating autistic kid with the uses of traditional methods. As mentioned before autistic kids perceptions, moods and emotions are different from other kids. They need special education both in school and home. Therefore I decided to work on this project which project purpose to gain experiences about some fundamental area to kid. Mainly I focused to make education consistent that means kids have chance to learn outside of school. According to research, articles, and experiments I determined challenges about learning that autistic kids face to daily life. Then I improved some technics and algorithms which are purposed to self learning.

2.1 Objectives/Tasks

ASR robot design consists of two parts which are hardware and software configurations.

Autistic Soft Robot has to be conducted from two different sides. One is the hardware of the robot which includes all sensors, controller and mechanical parts of the robot. The other is the software which includes all programming languages and environments we use in this project. Communication between those two parts should be done properly.

2.2.Realistic Constraints

- **Economic Constraints**
  This project is aimed to be implemented in limited production. Therefore, accomplishing the project at finding donate is goal. The tools such as cameras, some other sensors, controlling equipment will be chosen accordingly to achieve optimum quality and expense. With the use of Google Services we have free GPU so cost will be reduce significantly.

- **Technical Constraints**
  Robot consumes much energy this may cause storage more power. However, for a long working time there is no suitable battery exist. Therefore, I will focus two solutions first decreasing power consuming or different type of batteries as chemicals. And second constraints about reachable workspace because of robot is mobile artificial intelligence progress and sensors must be fast interaction between environment. That means powerful GPU and communication system needed.

- **Social Constraints**
  Robot will be recording children and environments via cameras. This may be considered as an intervention to the private life of other people. Another aspect is relationship between robot and children may be stronger than family. This may cause disconnection between children and families.

- **Risk Management**

Dataset selection may not meet the requirements of the project in some sense. This problem will be solved by providing and using diverse datasets. Other risk is our solution to the problem which is based on interaction with autistic children may not be realistic. Every child has different physiology and some children may not get well with our robot algorithms. Solution is to use machine learning algorithms efficiently that it can accommodate itself with different children easily. In the implementation process, real life experiments may yield unplanned issues. Project implementation may be different from our expectations. We have to regulate undesirable parts.

3. METHDOLOGY

3.1. Hardware: The robots hardware consists of three main parts which are sensors on it, controllers in order to control the robot and its soft material. To make connection between the data which come from sensors and actuators, and the software of the robot, controllers are needed. Controller must be included with powerful GPU (Graphical Processing Unit) to make calculations fast in real-time. So far, Raspberry Pi and Arduino devices are used while implementing programs, and Raspberry Pi is decided to be used. The Raspberry Pi is a low cost, high-performance, credit-card sized computer. It is capable of doing everything you'd expect a desktop computer to do. In addition, it has the ability to interact with the outside world. [14] Raspberry Pi 4, which is the last version of them, is improved itself and with new features, it has GPU capacity that we need in our project. In addition, it is energy-efficient which was a constraint for us because the robot should be mobile. Raspberry Pi uses very little energy while working, so this is very favorable for us. In addition, Raspberry Pi has a camera module that can be used for the robot. As mentioned before, our robot will be taking most of the data from children and outside world from camera. So, using a camera module that is already built for Raspberry Pi makes software part easier. The Raspberry Pi is a powerful tool when it comes to artificial intelligence (AI) and machine learning (ML). Its processing capabilities, matched with a small form factor and low power requirements, make it a great choice for smart robotics and embedded projects. [10] Therefore, because of its powerful processing capabilities, portability, and energy-efficiency we are using Raspberry Pi 4 as our controller. 5 ASR is continuously taking information data from its surroundings and from the children. Therefore, it should have all necessary equipment on it. Several sensors and actuators will be used. The most important sensor on the robot is the camera. Since we will be using Raspberry Pi as a controller, we will use its own camera module. This camera is easy to use with Raspberry Pi and we are able to acquire images in real-time efficiently using this module. Our robot will use those images in detection and recognition algorithms. One of the important properties of the hardware part of the project is its softness. Since the robot is designed to be very much in interaction with children, it should not accidently harm or scare them. In the robot body, soft material will be used because in hard robotics small measurement error causes large damage so that robot will be created from soft material as silicon. With the soft materials we can deal with more uncertain environment and unknown objects can be touched without damage. This is bio-inspired structure which can undergo large deformations. Non-rigid bodies exist unlike hard robotics

3.2. Software: Autistic Soft Robot(ASR) is based mostly on machine learning technics and learning from the children. Machine learning is subset of AI but with the permission of self-learning, robot will improve itself and do more than given task which is learnt from given past data [3]. Every child has different psychologies, and this will cause different reactions in similar situations. Our robot should be able to classify the reactions and emotions of the children. It should analyze the emotions of the children and act accordingly. Therefore, robot will be collecting data from the child continuously. After collection of this data it will improve its own learning skills. This data will be collected on the robots' own database.

Another aspect of the robot is based on teaching some fundamentals to children. The robot will be in interaction with the children and teach them while playing games with them. Robot may take attention of children as making noise or clapping hands. Then robot will offer playing game together or give a mission to the children. Robot will ask question to child and reply to him accordingly as if child answers true complement or clap hands. If answers wrong gently teach the right. Robot will collect data from children with using cameras and sensors until the end of the game.
Our robot collects data continuously and processes it in real time. In order to maintain the robots facilities precise in real time, we need a strong controller, big GPU and CPU, and big memory storage to our data. In addition we need portability in the controller. If we want to obtain strong enough hardware, this can be very costly.
We have worked on different solutions in order to handle these problems. Firstly, Raspberry Pi 4 is used to handle portability. In addition since it is not different from a normal size computer, it is strong enough to run our algorithms. Detailed properties of Raspberry Pi 4 and its difference from old versions are explained in detail in hardware section.
Another option to handle storage and GPU problems is to work on IoT Cloud basis systems. Cloud technology provides memory storage, processor etc. from network. That means even weakest device will be able to access wanted information from any place and process it as fast using Cloud technology. We don't need to have GPU or CPU on the device if we use environment using cloud technology.
We have worked on two environments which both use cloud technology. These are Amazon and Google Collab.

Firstly, Amazon is one of the first companies that use IoT system. Amazon Web Service (AWS) is bringing Artificial Intelligence and IoT together to make devices more intelligent. You can create models in the cloud, and then deploy them to devices where they run 2x faster compared to other offerings. AWS IoT sends data back to the cloud for continuous improvement of models. AWS IoT also supports more machine learning frameworks compared to other offerings [6]. Therefore, we tried to implement the machine learning algorithms in Amazon environment. AWS has its own Management Console that we can choose services from and use. In order to understand properties of AWS, we can look at an example.

One feature of our robot was to teach fundamentals to the children by playing games with them. We have tried to implement one of these games which is teaching the facial organs to the child. Below, preliminary results on this experiment working with Amazon are explained.

Learning Facial Organs (Eyes, lips, noise, ears)- OPEN-CV

One part of our project is that the robot will teach children their facial sense organs by playing game with
them. This game mainly consists of two stages. Firstly, the robot will give voice and visual orders to children such as "Touch your mouth." or "Where is your mouth?". Secondly, after child gives the answer, robot will check if the answer is accurate or not. If true, robot will clap hands for greeting the kid. We have achieved in this experiment that the robot can check if the children can show the right facial sense organs. Our algorithm returns a Boolean value which is true if the kid shows right facial organ, false otherwise.Every person has different sizes of facial landmarks. Therefore, our code should be adaptive for different sizes of faces since we need to determine eyes, lips, ears, noise and mouth landmarks clearly. In this experiment we used OpenCv (computer vision) for detection and recognition process. In addition, we had to determine children's hands position because we must catch on the camera when children touch his ear or eye. Therefore, we decided to use computer vision for hand recognition and hand tracking process. Mainly we examined articles about Human Pose Estimation. Human Pose Estimation algorithms are based on detection of wrist points on the all body and try to track these points with given coordinates. Final step is about how the program will detect when children touch facial sense organs.

In the first step we detect face and assign points with green color. Then we assigned points around eyes, lips and noise. Because of using computer vision, program assign green points to only my face. That means when other people look at camera with me their landmark points were not recognized. Therefore, some arrangements are needed in the code block. Therefore, we realized data training is needed. We decided to use Dlib which is a library and provide image processing and deep learning algorithms. We use data set which is called "shape predictor". And our model is trained for different faces. You can see a detected face and assigned landmarks in the Figure 5 below.

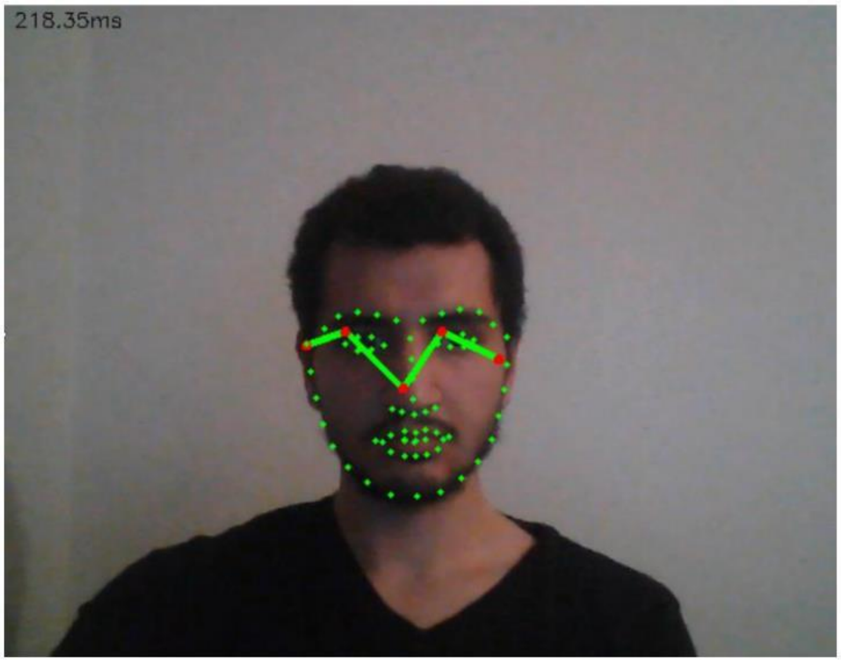


Figure 1. Face detection and landmarks assignment

Secondly, we need to recognize and track hands position. We use Human Pose Estimation technique. We specified some key points for the wrists and also for the eyes, ears and lips. Key points movement's which are colored red are tracked by camera. We achieved to measure distance between points on the wrist near the hands and facial landmarks. If we succeed we are going to define some integer variable for the limit distance and with using Boolean function. If distance measured by camera less than our predefined value, function return true and motor starts.

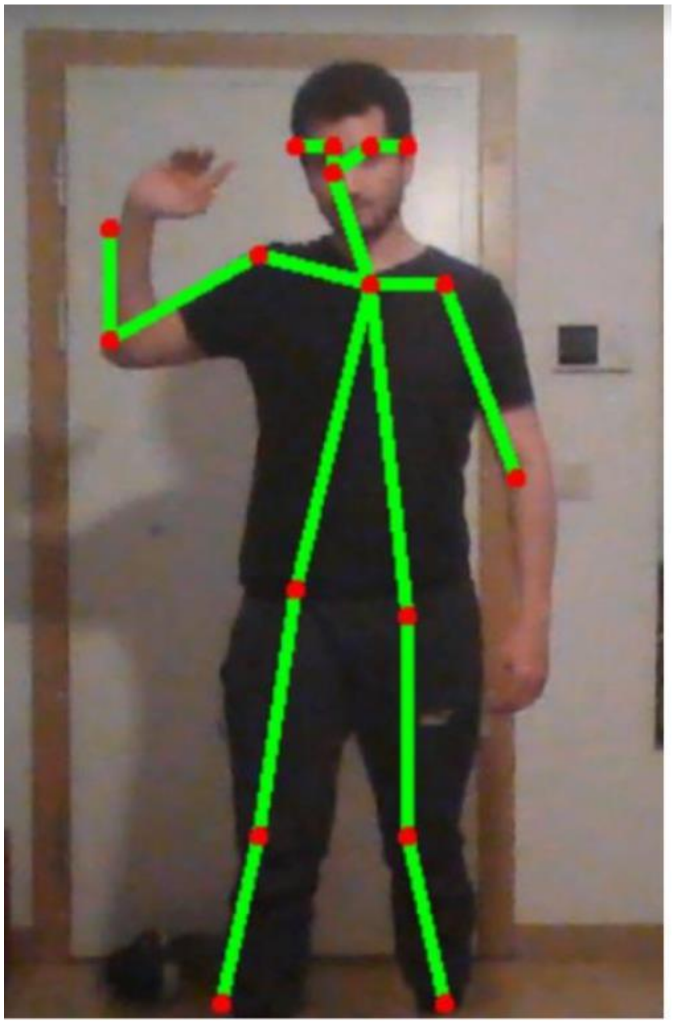


Figure 2. Body detection and tracking

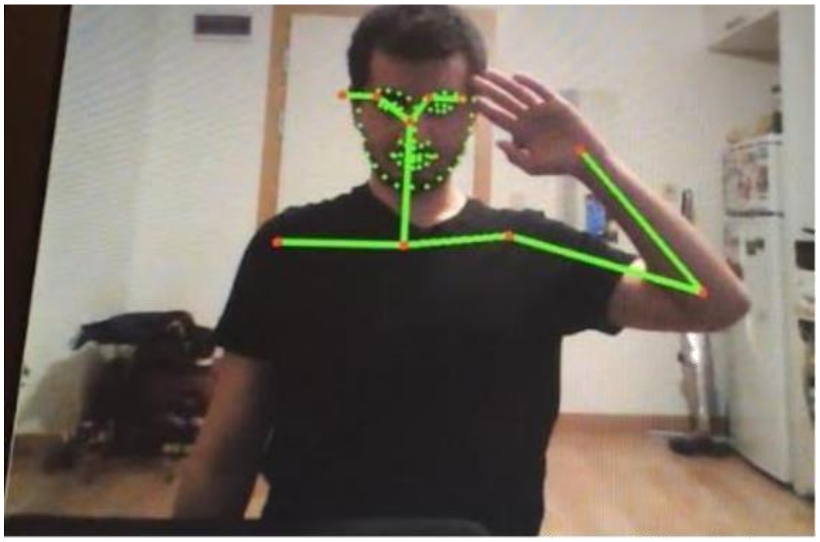


Figure 3. Tracking hand while touching ear

We have decided that using only computer vision techniques is not suitable for our algorithm. There are two important problems causing this decision. First one is Human Pose Estimation algorithm is not stable because when person moves, key points move and computer was not able to track accurate and fast enough. Distance were not calculated accurately for most of the time. In order to solve distance problem, we took instantaneous frames from video and try to measure distance. However, since raspberry pi has low CPU it causes delay. Other problem is that since all people have different sizes of bodies, distance we defined doesn't match for all people, and we need to make adjustments for every different people. This is unwanted. In addition, when some part of body or face is behind and not seen by the camera, camera was not able to detect face or body. These problems can be solved by using machine learning techniques. Other than using only computer vision which assigns coordinates and create landmarks, we have decided to use deep learning techniques that are faster and much accurate. We will use a huge dataset for face and hand and teach to our computer. We will use GPU and this will prevent delay. We can still separately train for face and hand, then detect if it touches ear or lip or etc. Other choice is we can directly train for if it is touching ears etc. By using second choice, we can prevent mathematical falsies.

Learning Facial Organs (Eyes, lips, noise, ears) – Amazon Web Service

One part of our project is that the robot will teach children their facial sense organs by playing game with them. This game mainly consists of two stages. Firstly, the robot will give voice and visual orders to children such as “Touch your mouth.” or “Where is your mouth?”. Secondly, after child gives the answer, robot will check if the answer is accurate or not. If true, robot will clap hands for greeting the kid.
We have managed this game by using a machine learning algorithm and creating a model. On that purpose, we have created our own dataset –which we still didn’t finish on working- and used it. This dataset consists of images of people touching-or not touching their mouth, ear etc. For example, below image is an example from the dataset which a person is touching his eye

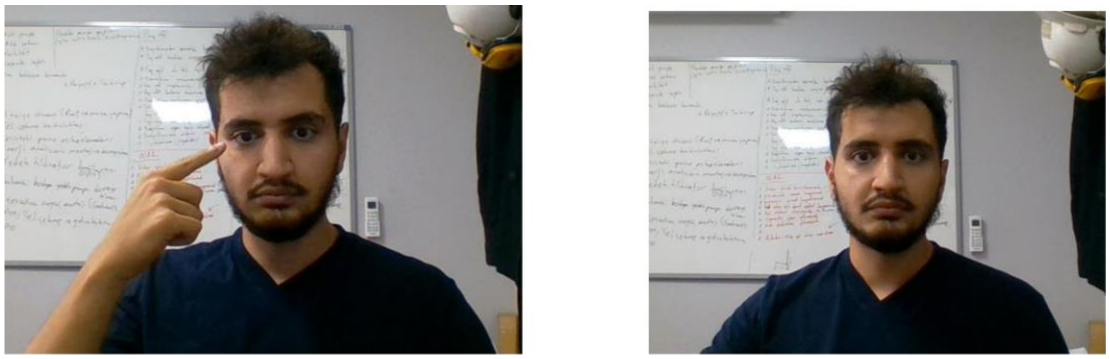


Figure 4. Example from the dataset

We have implemented the learning algorithm in Amazon Web Service. Amazon Web Service has its own storage service called Amazon S3. S3 is storage for internet that we can store and retrieve any amount of data at any time from the web. Since this is internet storage, this solves our big storage management problem. In addition this bucket is simple to use with other services of AWS.

We are creating our own dataset and we uploaded the dataset so far in Amazon S3 storage. Amazon has its own data labeling service which helps partitioning the labeling jobs of the images we have. We don’t use a present dataset and we are trying to create our own, therefore labeling should be done one by one to every image we have. Thanks to Amazon labeling service, we don’t have to create new folders to label each image. Labeling service while choosing labels for images is shown in the below figure. This image is selected to be not touching.

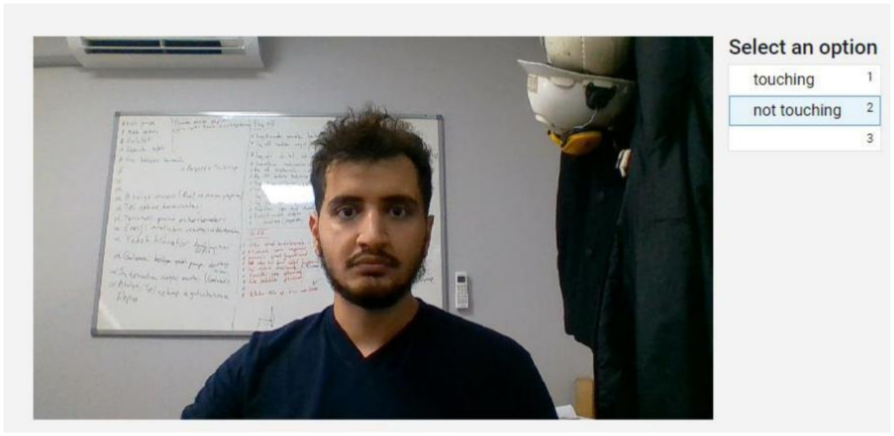


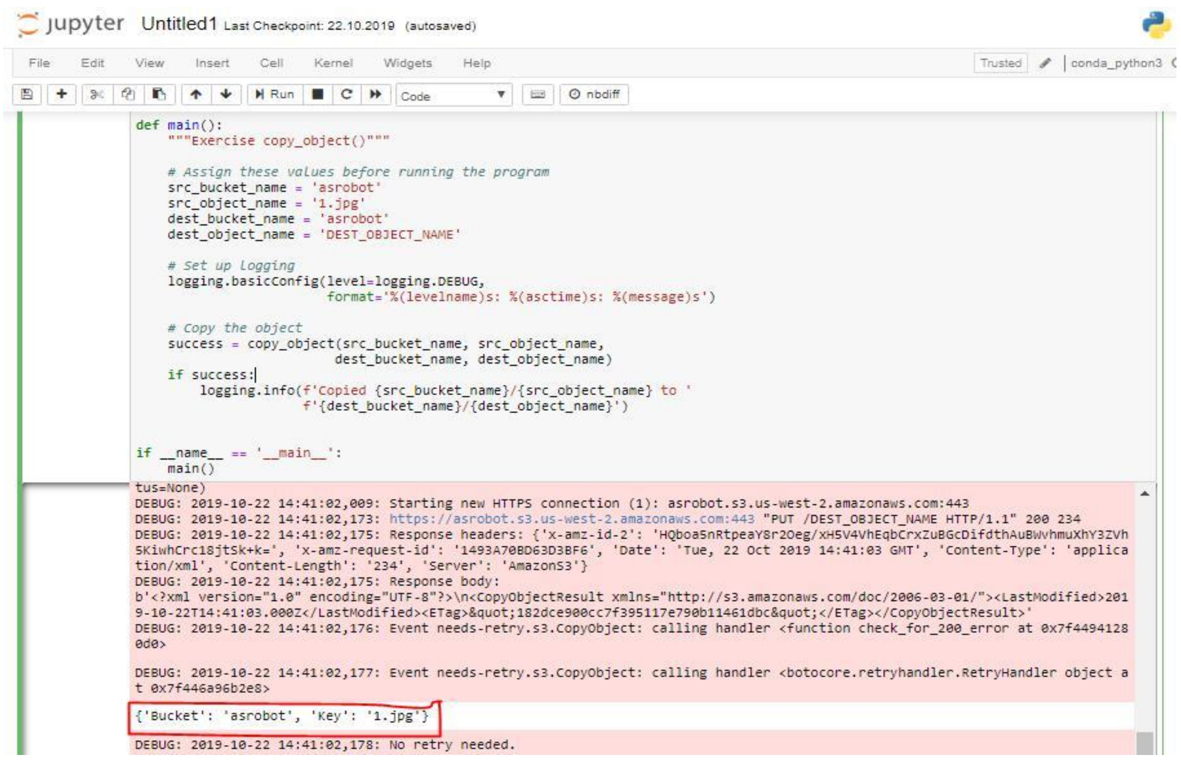
Figure 5. Amazon data labeling service

Amazon has a service called Sagemaker which we can work on Jupyter with Python programming language. We have implemented our learning algorithm in this platform and created learning model using our dataset from Amazon S3 storage. Below image shows a code part which enables the access to Amazon S3 bucket from code. Part inside red box is output which shows we were able to take image named “1.jpg” from Amazon S3 bucket that is named “asrobot”. Full codes are available in the Appendix section.

Google Colab is a free cloud service and now it supports free GPU! You can:

- improve your **Python** programming language coding skills.

- develop deep learning applications using popular libraries such as **Keras**, **TensorFlIOkow**, **PyTorch**, and **OpenCV**.


jupyter Untitled1 Last Checkpoint: 22.10.2019 (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Trusted
conda_python3
Run
Code
nbdiff

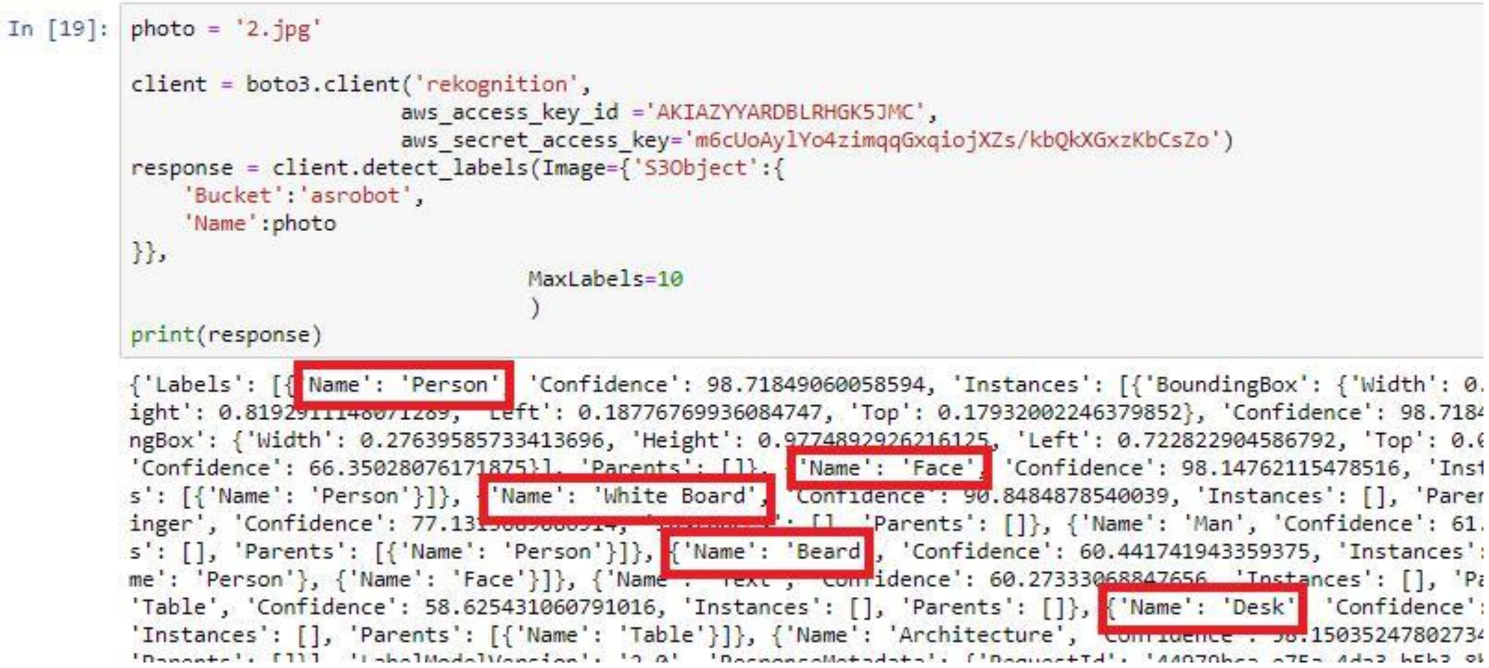
```
def main():
    """Exercise copy_object()"""

    # Assign these values before running the program
    src_bucket_name = 'asrobot'
    src_object_name = '1.jpg'
    dest_bucket_name = 'asrobot'
    dest_object_name = 'DEST_OBJECT_NAME'

    # Set up logging
    logging.basicConfig(level=logging.DEBUG,
                        format='%(levelname)s: %(asctime)s: %(message)s')

    # Copy the object
    success = copy_object(src_bucket_name, src_object_name,
                          dest_bucket_name, dest_object_name)
    if success:
        logging.info(f'Copied {src_bucket_name}/{src_object_name} to '
                     f'{dest_bucket_name}/{dest_object_name}')


if __name__ == '__main__':
    main()
```

```
tus=None)
DEBUG: 2019-10-22 14:41:02,009: Starting new HTTPS connection (1): asrobot.s3.us-west-2.amazonaws.com:443
DEBUG: 2019-10-22 14:41:02,173: https://asrobot.s3.us-west-2.amazonaws.com:443 "PUT /DEST_OBJECT_NAME HTTP/1.1" 200 234
DEBUG: 2019-10-22 14:41:02,175: Response headers: {'x-amz-id-2': 'HQboa5nRtpeaY8r2Oeg/xH5V4VhEqbCrxZuBGcDifdthAuBWvhmuXhY3ZVh
5KiwhCrc18jtSk+k=', 'x-amz-request-id': '1493A70BD63D3BF6', 'Date': 'Tue, 22 Oct 2019 14:41:03 GMT', 'Content-Type': 'applica
tion/xml', 'Content-Length': '234', 'Server': 'AmazonS3'}
DEBUG: 2019-10-22 14:41:02,175: Response body:
b'<?xml version="1.0" encoding="UTF-8"?>\n<CopyObjectResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/"><LastModified>201
9-10-22T14:41:03.000Z</LastModified><ETag>&quot;182dce900cc7f395117e790b11461dbc&quot;</ETag></CopyObjectResult>'
DEBUG: 2019-10-22 14:41:02,176: Event needs-retry.s3.CopyObject: calling handler <function check_for_200_error at 0x7f4494128
0d0>

DEBUG: 2019-10-22 14:41:02,177: Event needs-retry.s3.CopyObject: calling handler <botocore.retryhandler.RetryHandler object a
t 0x7f446a96b2e8>

{'Bucket': 'asrobot', 'Key': '1.jpg'}
DEBUG: 2019-10-22 14:41:02,178: No retry needed.
```

Figure 6. Amazon implementation of python code – calling the dataset from Amazon S3

Our robot will have a camera on it and it should be able to process video in real-time. It should see the child from its camera and put video stream in the machine learning algorithm. Then, robot should understand its state and accordingly. In addition, our robot should do this in real time since it will be in interaction with the children. In order to do that, Amazon offers services called Rekognition and Video Kinesis which can be used together in real time video processing. Amazon Rekognition Video is a deep learning powered video analysis service that detects activities and recognizes objects, states as we wanted. Another service of Amazon is Kinesis Video Streams which is storage for video data. In addition, it can be used to watch the video streams in real time as they are received in the cloud. We can monitor the live streams either on the AWS Management Console, or develop another monitoring application that uses the Kinesis Video Streams API library to display live video. Monitoring and using Amazon Kinesis Video Streams from different platform is very important for us.

Below in the figure, there is an example of object detection using Rekognition Service of Amazon. Image named “2.jpg” is uploaded and recognition is applied on it. Output can be seen and red boxes shows detected objects. Full code is available in the appendix section.



```
In [19]: photo = '2.jpg'

         client = boto3.client('rekognition',
                              aws_access_key_id ='AKIAZYYARDBLRHGK5JMC',
                              aws_secret_access_key='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')
         response = client.detect_labels(Image={'S3Object':{
             'Bucket':'asrobot',
             'Name':photo
         }},
                                        MaxLabels=10
                                        )
         print(response)
```

```
{'Labels': [{'Name': 'Person', 'Confidence': 98.71849060058594, 'Instances': [{'BoundingBox': {'Width': 0.
ight': 0.81929111480071289, 'Left': 0.18776769936084747, 'Top': 0.17932002246379852}, 'Confidence': 98.718
ngBox': {'Width': 0.27639585733413696, 'Height': 0.9774892926216125, 'Left': 0.722822904586792, 'Top': 0.0
'Confidence': 66.35028076171875}], 'Parents': []}, {'Name': 'Face', 'Confidence': 98.14762115478516, 'Ins
s': [{'Name': 'Person'}]}, {'Name': 'White Board', 'Confidence': 90.8484878540039, 'Instances': [], 'Pare
inger', 'Confidence': 77.13[illegible], 'Instances': [], 'Parents': []}, {'Name': 'Man', 'Confidence': 61.
s': [], 'Parents': [{'Name': 'Person'}]}, {'Name': 'Beard', 'Confidence': 60.441741943359375, 'Instances':
me': 'Person'}, {'Name': 'Face'}]}, {'Name': 'Text', 'Confidence': 60.27333068847656, 'Instances': [], 'P
'Table', 'Confidence': 58.625431060791016, 'Instances': [], 'Parents': []}, {'Name': 'Desk', 'Confidence':
'Instances': [], 'Parents': [{'Name': 'Table'}]}, {'Name': 'Architecture', 'Confidence': 58.1503524780273
'Parents': []}], 'LabelModelVersion': '2.0', 'ResponseMetadata': {'RequestId': '44970bca-e75a-4da3-b5b3-8
```

Figure 7. Code part from Amazon Rekognition Service – output showing recognized object from image

All of the services of Amazon mentioned above makes it very favorable to use in the project. However Amazon is not a free service. Since we want to use it with a lot of data, Amazon becomes very costly for our project. So, we decided to work on another cloud platform called Google Colaboratory. We have worked on the same example mentioned above in Google Colab.

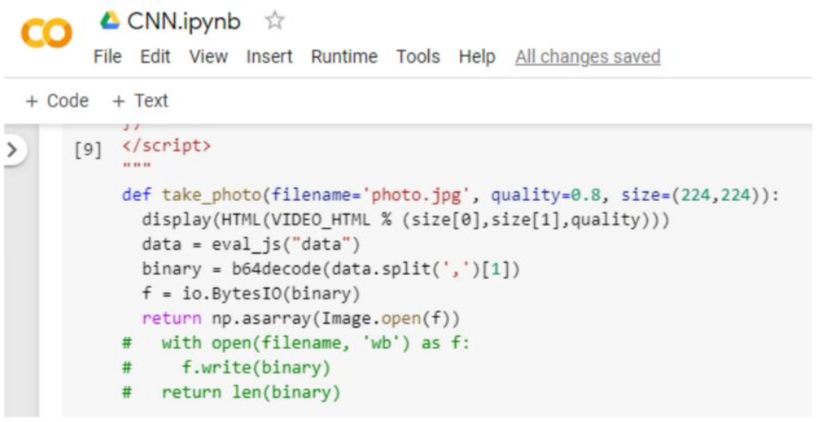
Learning Facial Organs (Eyes, lips, noise, ears) – Google Colaboratory

As mentioned above, this example is about teaching children their facial sense organs by playing game with them. Machine learning is used in classifying if child touches the right facial organ or not. On that purpose, we use Google Colab platform while implementing our algorithm.

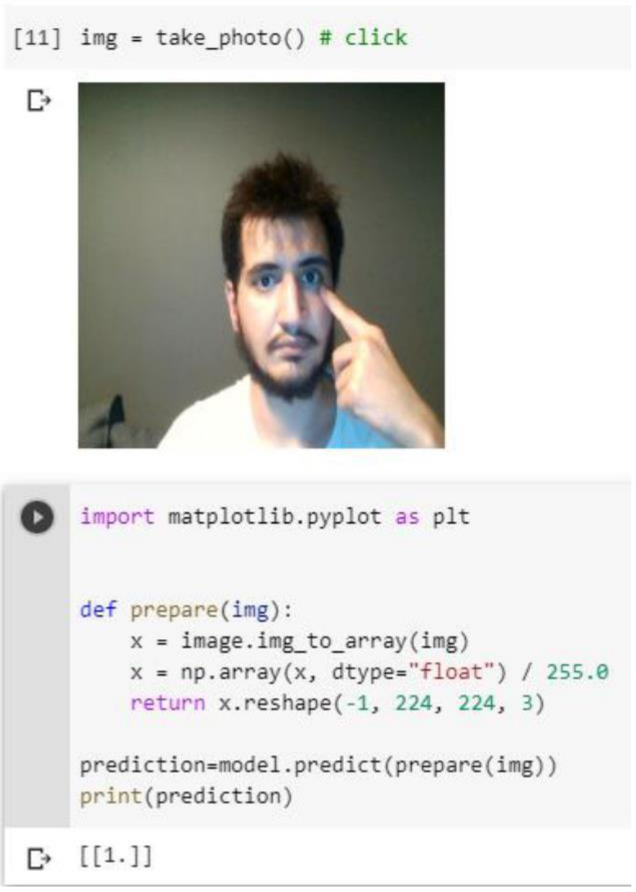
Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud. With Colaboratory we can write and execute code, save and share the analyses, and access powerful computing resources, all for free from the browser. [8] Since it is working from the cloud and it uses GPU and CPU, it is favorable for us to use Google Colab. We implemented code in this platform. Accessed Google Colab using Raspberry Pi and this give us portability, storage and fast real-time processing ability.

As a storage for our dataset, we are using Google Drive and accessing the images from there. We uploaded zip folder of images and called it from the program. Different from Amazon environment, we prepared a labeling document which is in csv. format.

We have worked with python programming language. We were able to take data from video streams in real-time using Google Colab. Code part for this is shown in the figure below.



```
[9]  </script>
     """
     def take_photo(filename='photo.jpg', quality=0.8, size=(224,224)):
       display(HTML(VIDEO_HTML % (size[0],size[1],quality)))
       data = eval_js("data")
       binary = b64decode(data.split(',')[1])
       f = io.BytesIO(binary)
       return np.asarray(Image.open(f))
     #   with open(filename, 'wb') as f:
     #     f.write(binary)
     #   return len(binary)
```



```
[11] img = take_photo() # click
```

```
import matplotlib.pyplot as plt


def prepare(img):
    x = image.img_to_array(img)
    x = np.array(x, dtype="float") / 255.0
    return x.reshape(-1, 224, 224, 3)

prediction=model.predict(prepare(img))
print(prediction)
```

```
[[1.]]
```

Figure 8. Google Colaboratory code for taking photo from a video stream

In addition, we were able to create a machine learning model using our dataset. Since the dataset consists of very little images for now, accuracy for the machine learning model was low, around %55. However via improving the dataset and the algorithm, this accuracy will be fixed.

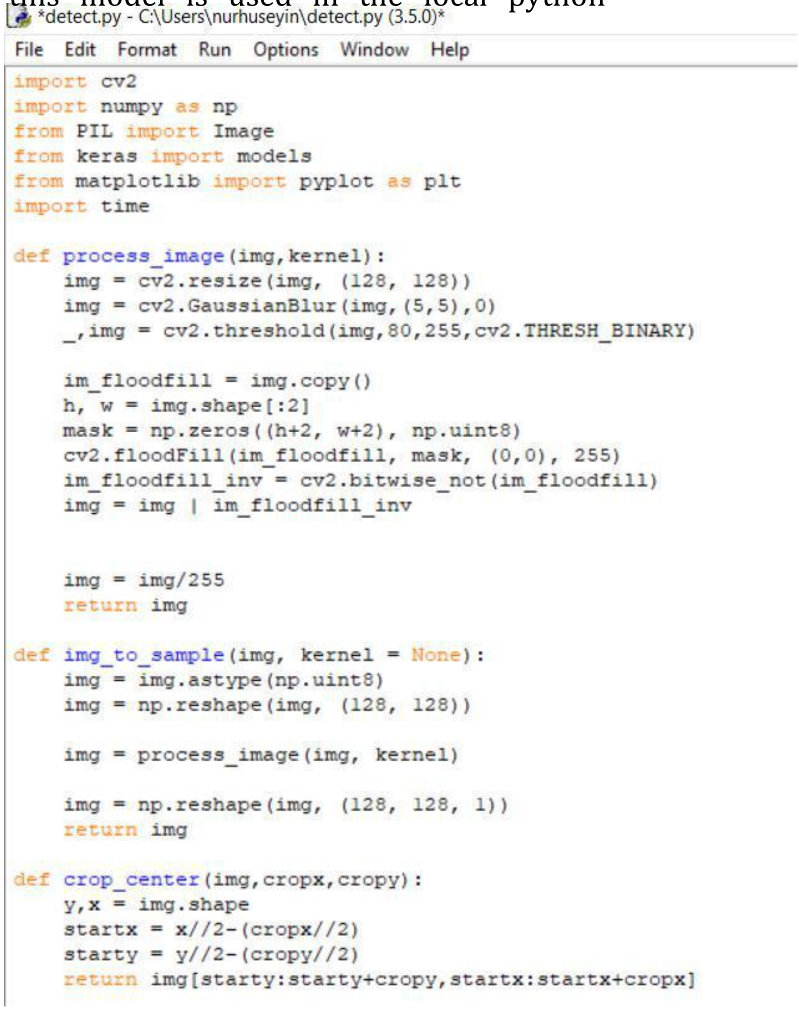
Another experiment based on Colab which purposed to teach fundamental math to kids with using hand gestures. As mentioned main goal of Project improve autistic kids with the question – response method that means robot should take several type of answer as body language and voice.

This program is designed to teach the autistic children simple mathematics. The running of the program is as follows:

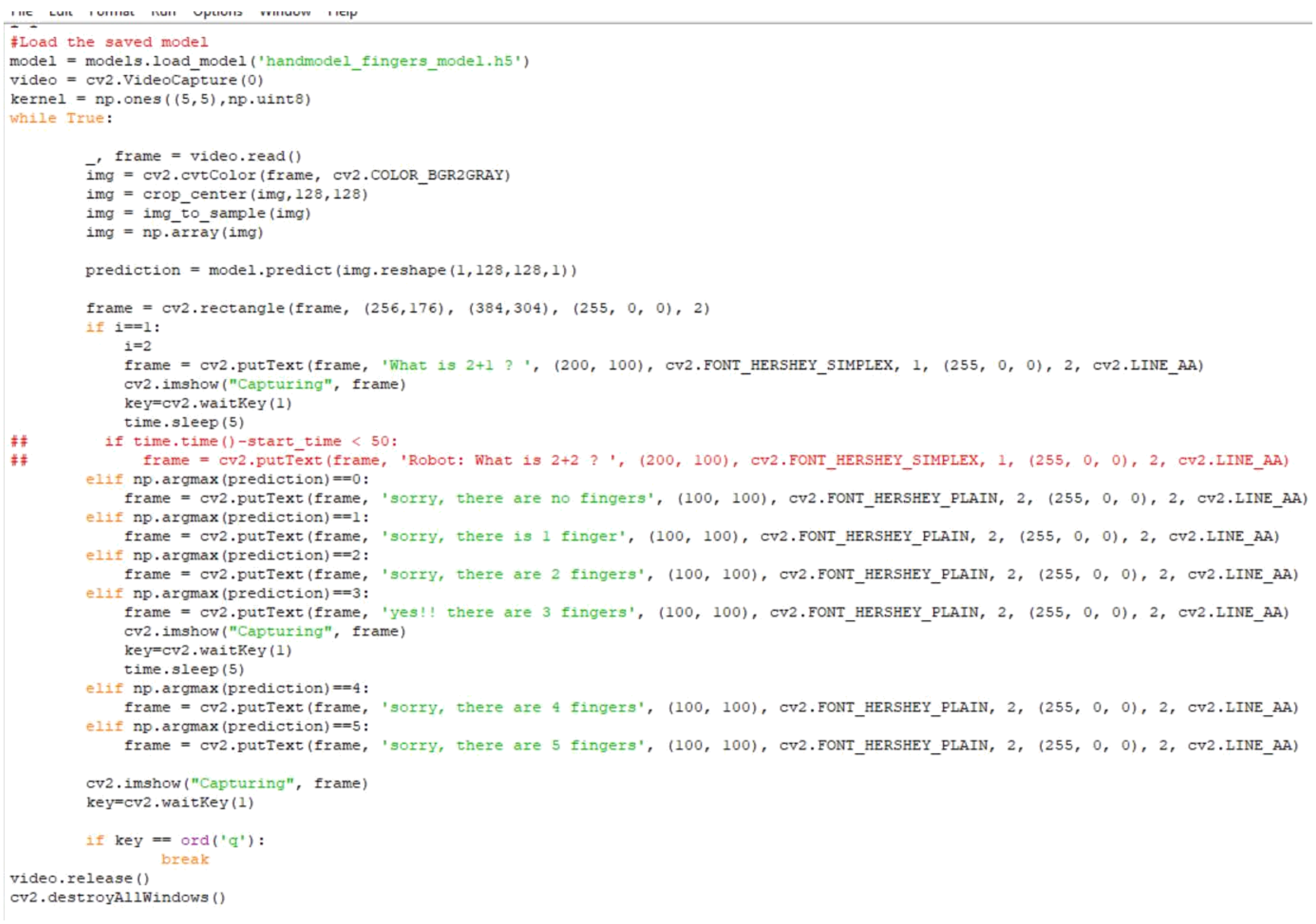
- Robot asks the child a simple math question, such as “What is 3-1?”
- Child answers and shows the answer with his/her hand
- Robot use its machine learning program in order to predict the hand and how many fingers
- If answer is right, robot greets the child.

In the machine learning part, CNN is used, and the dataset trained in the cloud platform, as mentioned in the progress report. Reason for using cloud platform is to reduce GPU and hardware needs. A model is created, and this model is used in the local python environment of the computer.

Program from local python environment, using the created model: I used to pre-trained model for the train my dataset. Also at least two thousand photo or frame from my hands needed so I using special algorithm which took 2000 frames per second .Then with using CNN module without landmark localization I trained my algorithm easily. Therefore some problems occurred, using pre-trained dataset decrease my model validation as %52

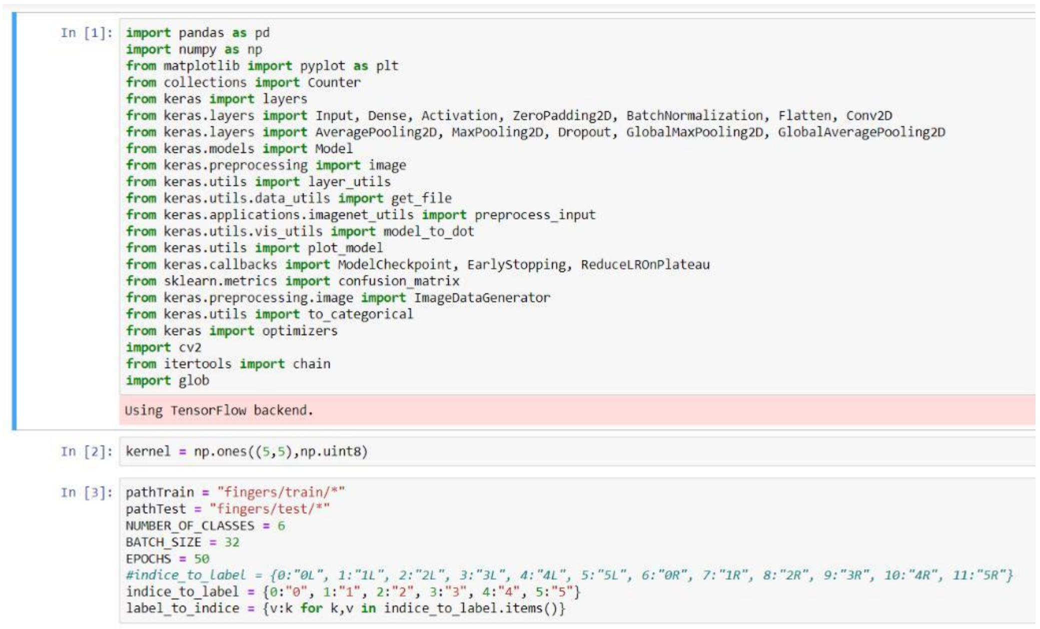

detect.py - C:\Users\nurhuseyin\detect.py (3.5.0)
File Edit Format Run Options Window Help

```
import cv2
import numpy as np
from PIL import Image
from keras import models
from matplotlib import pyplot as plt
import time

def process_image(img,kernel):
    img = cv2.resize(img, (128, 128))
    img = cv2.GaussianBlur(img,(5,5),0)
    _,img = cv2.threshold(img,80,255,cv2.THRESH_BINARY)

    im_floodfill = img.copy()
    h, w = img.shape[:2]
    mask = np.zeros((h+2, w+2), np.uint8)
    cv2.floodFill(im_floodfill, mask, (0,0), 255)
    im_floodfill_inv = cv2.bitwise_not(im_floodfill)
    img = img | im_floodfill_inv


    img = img/255
    return img

def img_to_sample(img, kernel = None):
    img = img.astype(np.uint8)
    img = np.reshape(img, (128, 128))

    img = process_image(img, kernel)

    img = np.reshape(img, (128, 128, 1))
    return img

def crop_center(img,cropx,cropy):
    y,x = img.shape
    startx = x//2-(cropx//2)
    starty = y//2-(cropy//2)
    return img[starty:starty+cropy,startx:startx+cropx]
```

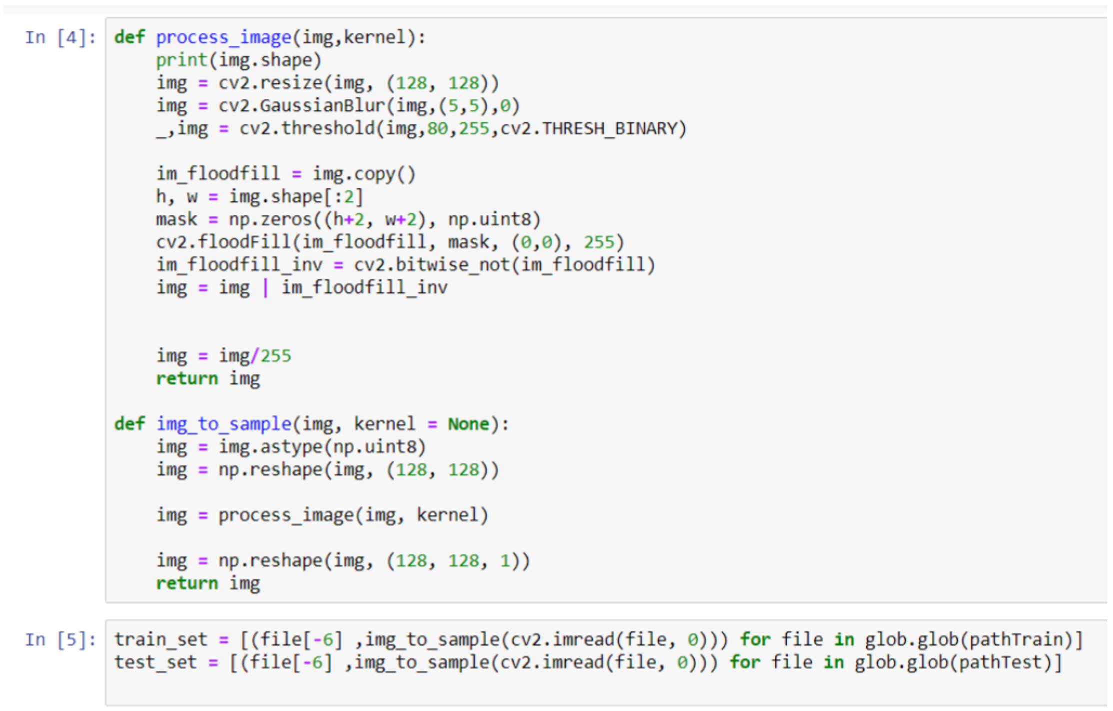


```
#Load the saved model
model = models.load_model('handmodel_fingers_model.h5')
video = cv2.VideoCapture(0)
kernel = np.ones((5,5),np.uint8)
while True:

        _, frame = video.read()
        img = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        img = crop_center(img,128,128)
        img = img_to_sample(img)
        img = np.array(img)

        prediction = model.predict(img.reshape(1,128,128,1))

        frame = cv2.rectangle(frame, (256,176), (384,304), (255, 0, 0), 2)
        if i==1:
            i=2
            frame = cv2.putText(frame, 'What is 2+1 ? ', (200, 100), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2, cv2.LINE_AA)
            cv2.imshow("Capturing", frame)
            key=cv2.waitKey(1)
            time.sleep(5)
##        if time.time()-start_time < 50:
##            frame = cv2.putText(frame, 'Robot: What is 2+2 ? ', (200, 100), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2, cv2.LINE_AA)
        elif np.argmax(prediction)==0:
            frame = cv2.putText(frame, 'sorry, there are no fingers', (100, 100), cv2.FONT_HERSHEY_PLAIN, 2, (255, 0, 0), 2, cv2.LINE_AA)
        elif np.argmax(prediction)==1:
            frame = cv2.putText(frame, 'sorry, there is 1 finger', (100, 100), cv2.FONT_HERSHEY_PLAIN, 2, (255, 0, 0), 2, cv2.LINE_AA)
        elif np.argmax(prediction)==2:
            frame = cv2.putText(frame, 'sorry, there are 2 fingers', (100, 100), cv2.FONT_HERSHEY_PLAIN, 2, (255, 0, 0), 2, cv2.LINE_AA)
        elif np.argmax(prediction)==3:
            frame = cv2.putText(frame, 'yes!! there are 3 fingers', (100, 100), cv2.FONT_HERSHEY_PLAIN, 2, (255, 0, 0), 2, cv2.LINE_AA)
            cv2.imshow("Capturing", frame)
            key=cv2.waitKey(1)
            time.sleep(5)
        elif np.argmax(prediction)==4:
            frame = cv2.putText(frame, 'sorry, there are 4 fingers', (100, 100), cv2.FONT_HERSHEY_PLAIN, 2, (255, 0, 0), 2, cv2.LINE_AA)
        elif np.argmax(prediction)==5:
            frame = cv2.putText(frame, 'sorry, there are 5 fingers', (100, 100), cv2.FONT_HERSHEY_PLAIN, 2, (255, 0, 0), 2, cv2.LINE_AA)

        cv2.imshow("Capturing", frame)
        key=cv2.waitKey(1)

        if key == ord('q'):
            break
video.release()
cv2.destroyAllWindows()
```

Program from the cloud platform is as follows:

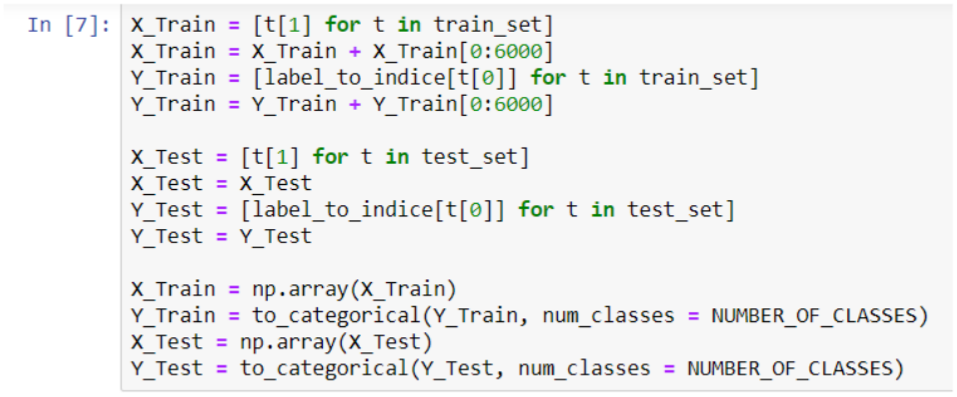


```
In [1]: import pandas as pd
        import numpy as np
        from matplotlib import pyplot as plt
        from collections import Counter
        from keras import layers
        from keras.layers import Input, Dense, Activation, ZeroPadding2D, BatchNormalization, Flatten, Conv2D
        from keras.layers import AveragePooling2D, MaxPooling2D, Dropout, GlobalMaxPooling2D, GlobalAveragePooling2D
        from keras.models import Model
        from keras.preprocessing import image
        from keras.utils import layer_utils
        from keras.utils.data_utils import get_file
        from keras.applications.imagenet_utils import preprocess_input
        from keras.utils.vis_utils import model_to_dot
        from keras.utils import plot_model
        from keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau
        from sklearn.metrics import confusion_matrix
        from keras.preprocessing.image import ImageDataGenerator
        from keras.utils import to_categorical
        from keras import optimizers
        import cv2
        from itertools import chain
        import glob

        Using TensorFlow backend.

In [2]: kernel = np.ones((5,5),np.uint8)

In [3]: pathTrain = "fingers/train/*"
        pathTest = "fingers/test/*"
        NUMBER_OF_CLASSES = 6
        BATCH_SIZE = 32
        EPOCHS = 50
        #indice_to_label = {0:"0L", 1:"1L", 2:"2L", 3:"3L", 4:"4L", 5:"5L", 6:"0R", 7:"1R", 8:"2R", 9:"3R", 10:"4R", 11:"5R"}
        indice_to_label = {0:"0", 1:"1", 2:"2", 3:"3", 4:"4", 5:"5"}
        label_to_indice = {v:k for k,v in indice_to_label.items()}
```

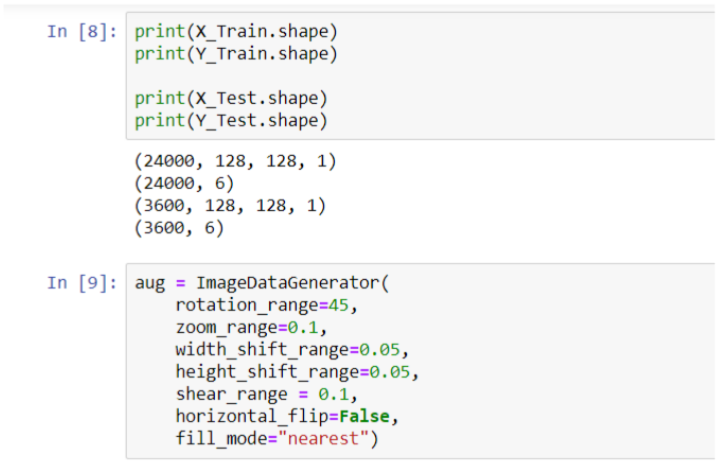


```
In [4]: def process_image(img,kernel):
            print(img.shape)
            img = cv2.resize(img, (128, 128))
            img = cv2.GaussianBlur(img,(5,5),0)
            _,img = cv2.threshold(img,80,255,cv2.THRESH_BINARY)

            im_floodfill = img.copy()
            h, w = img.shape[:2]
            mask = np.zeros((h+2, w+2), np.uint8)
            cv2.floodFill(im_floodfill, mask, (0,0), 255)
            im_floodfill_inv = cv2.bitwise_not(im_floodfill)
            img = img | im_floodfill_inv


            img = img/255
            return img

        def img_to_sample(img, kernel = None):
            img = img.astype(np.uint8)
            img = np.reshape(img, (128, 128))

            img = process_image(img, kernel)

            img = np.reshape(img, (128, 128, 1))
            return img
```

```
In [5]: train_set = [(file[-6] ,img_to_sample(cv2.imread(file, 0))) for file in glob.glob(pathTrain)]
        test_set = [(file[-6] ,img_to_sample(cv2.imread(file, 0))) for file in glob.glob(pathTest)]
```

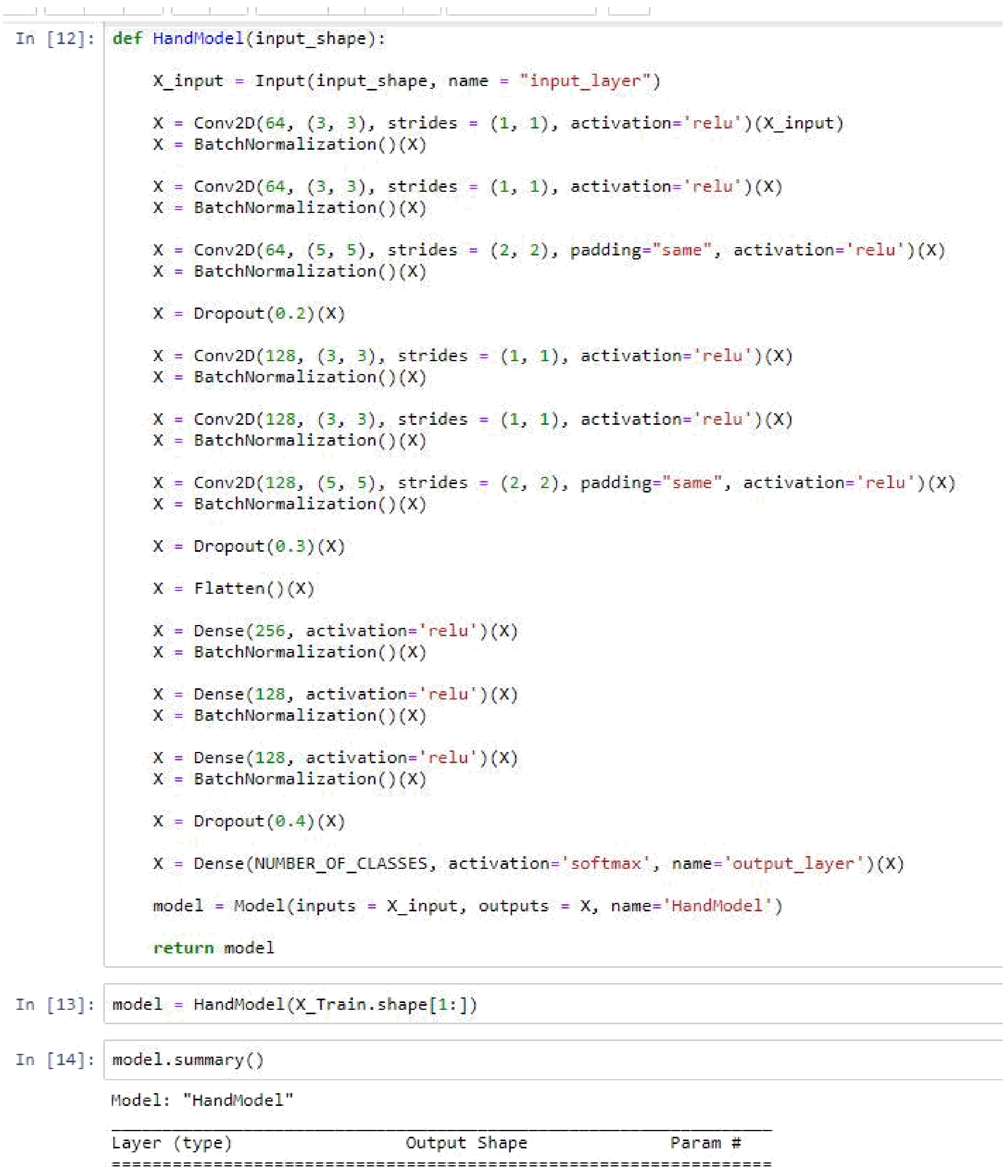


```
In [7]: X_Train = [t[1] for t in train_set]
        X_Train = X_Train + X_Train[0:6000]
        Y_Train = [label_to_indice[t[0]] for t in train_set]
        Y_Train = Y_Train + Y_Train[0:6000]

        X_Test = [t[1] for t in test_set]
        X_Test = X_Test
        Y_Test = [label_to_indice[t[0]] for t in test_set]
        Y_Test = Y_Test

        X_Train = np.array(X_Train)
        Y_Train = to_categorical(Y_Train, num_classes = NUMBER_OF_CLASSES)
        X_Test = np.array(X_Test)
        Y_Test = to_categorical(Y_Test, num_classes = NUMBER_OF_CLASSES)
```

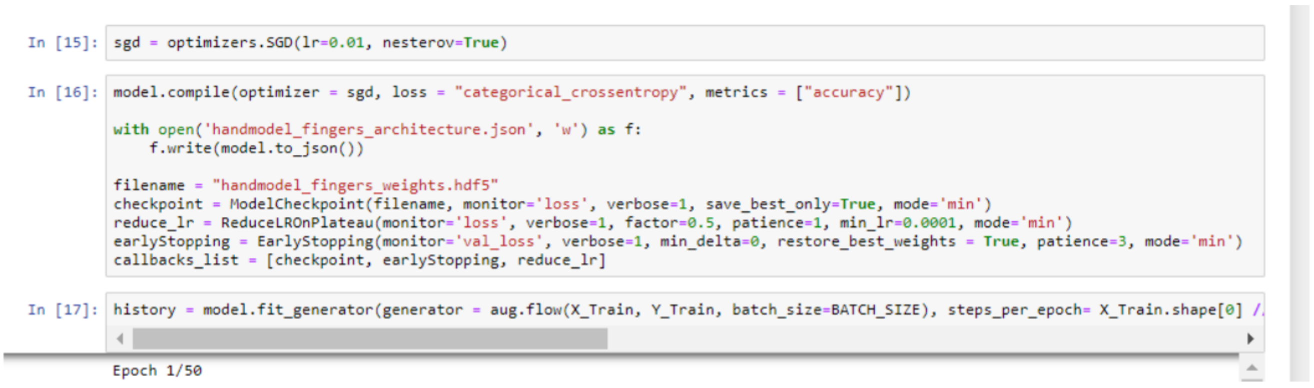


```
In [8]: print(X_Train.shape)
        print(Y_Train.shape)

        print(X_Test.shape)
        print(Y_Test.shape)

        (24000, 128, 128, 1)
        (24000, 6)
        (3600, 128, 128, 1)
        (3600, 6)
```

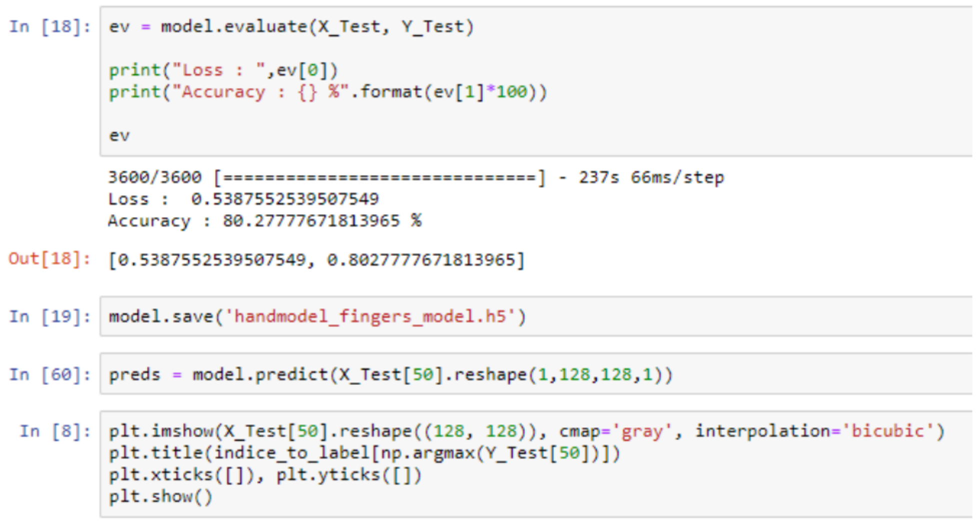
```
In [9]: aug = ImageDataGenerator(
            rotation_range=45,
            zoom_range=0.1,
            width_shift_range=0.05,
            height_shift_range=0.05,
            shear_range = 0.1,
            horizontal_flip=False,
            fill_mode="nearest")
```



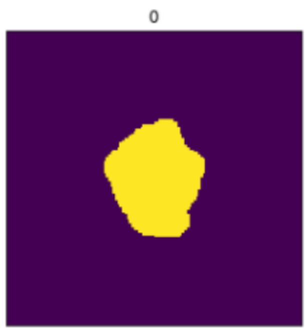
```
In [12]: def HandModel(input_shape):

             X_input = Input(input_shape, name = "input_layer")

             X = Conv2D(64, (3, 3), strides = (1, 1), activation='relu')(X_input)
             X = BatchNormalization()(X)

             X = Conv2D(64, (3, 3), strides = (1, 1), activation='relu')(X)
             X = BatchNormalization()(X)

             X = Conv2D(64, (5, 5), strides = (2, 2), padding="same", activation='relu')(X)
             X = BatchNormalization()(X)

             X = Dropout(0.2)(X)

             X = Conv2D(128, (3, 3), strides = (1, 1), activation='relu')(X)
             X = BatchNormalization()(X)

             X = Conv2D(128, (3, 3), strides = (1, 1), activation='relu')(X)
             X = BatchNormalization()(X)

             X = Conv2D(128, (5, 5), strides = (2, 2), padding="same", activation='relu')(X)
             X = BatchNormalization()(X)

             X = Dropout(0.3)(X)

             X = Flatten()(X)

             X = Dense(256, activation='relu')(X)
             X = BatchNormalization()(X)

             X = Dense(128, activation='relu')(X)
             X = BatchNormalization()(X)

             X = Dense(128, activation='relu')(X)
             X = BatchNormalization()(X)

             X = Dropout(0.4)(X)

             X = Dense(NUMBER_OF_CLASSES, activation='softmax', name='output_layer')(X)

             model = Model(inputs = X_input, outputs = X, name='HandModel')

             return model

In [13]: model = HandModel(X_Train.shape[1:])

In [14]: model.summary()

         Model: "HandModel"
         _________________________________________________________________
         Layer (type)                 Output Shape              Param #
         =================================================================
```



```
In [15]: sgd = optimizers.SGD(lr=0.01, nesterov=True)

In [16]: model.compile(optimizer = sgd, loss = "categorical_crossentropy", metrics = ["accuracy"])

         with open('handmodel_fingers_architecture.json', 'w') as f:
             f.write(model.to_json())

         filename = "handmodel_fingers_weights.hdf5"
         checkpoint = ModelCheckpoint(filename, monitor='loss', verbose=1, save_best_only=True, mode='min')
         reduce_lr = ReduceLROnPlateau(monitor='loss', verbose=1, factor=0.5, patience=1, min_lr=0.0001, mode='min')
         earlyStopping = EarlyStopping(monitor='val_loss', verbose=1, min_delta=0, restore_best_weights = True, patience=3, mode='min')
         callbacks_list = [checkpoint, earlyStopping, reduce_lr]

In [17]: history = model.fit_generator(generator = aug.flow(X_Train, Y_Train, batch_size=BATCH_SIZE), steps_per_epoch= X_Train.shape[0] /

         Epoch 1/50
```



```
In [18]: ev = model.evaluate(X_Test, Y_Test)

         print("Loss : ",ev[0])
         print("Accuracy : {} %".format(ev[1]*100))

         ev

         3600/3600 [==============================] - 237s 66ms/step
         Loss :  0.5387552539507549
         Accuracy : 80.27777671813965 %

Out[18]: [0.5387552539507549, 0.8027777671813965]

In [19]: model.save('handmodel_fingers_model.h5')

In [60]: preds = model.predict(X_Test[50].reshape(1,128,128,1))

 In [8]: plt.imshow(X_Test[50].reshape((128, 128)), cmap='gray', interpolation='bicubic')
         plt.title(indice_to_label[np.argmax(Y_Test[50])])
         plt.xticks([]), plt.yticks([])
         plt.show()
```



Facial Expression Recognation – Google Collab & Raspberyy Pi

In the last step of project we purposed to teach facial expression as smiling,crying,surprising based on related emotions like happy, angry ,fear.According to experiments autistic childrens aren't very success to show their emotions that means in some situations they can not describe themselves easily. As an example when they feel happy they should smile but they can not use their gestures exactly true they show reactions as crying.In the previous stages of project we made some algorithm about to teach children fundamental math and facial sense organs and scnerio about robot act is simple, robot ask question about math or facial organs and children should give answer with using body language if answer is true robot will give positive feedback as slapping hand or dancing.In this part of project scnerio is also same.

First give some explaniation about software structure and methods.We purposed to reduce use of Gpu to the reduce hardware cost so as mentioned before we use Google collaboraty to the train our dataset and create our model.First we needed to find data set for training and testing.Then we decided use famous facial expression dataset which is called fer2013 which dataset purpose to challenges in representation learning as mentioned facial expression recognation challenge.

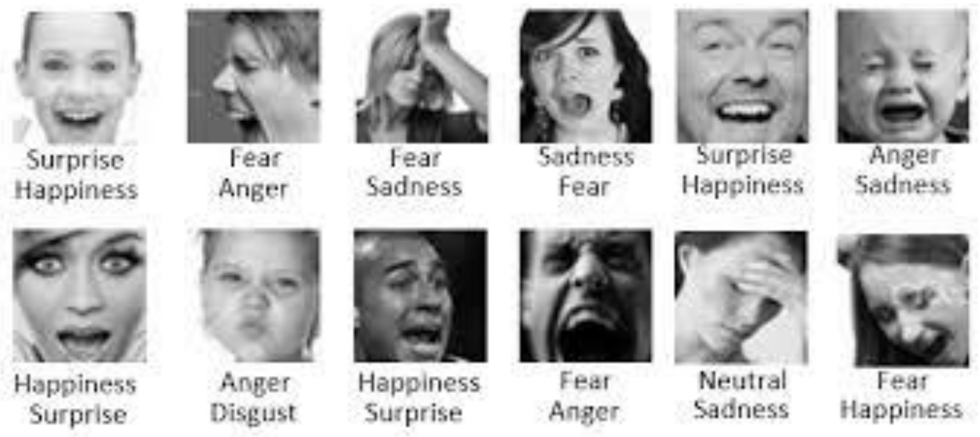


Figure 9. fer2013 Dataset from Kaggle

Data Description: The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centered and occupies about the same amount of space in each image. The task is to categorize each face based on the emotion shown in the facial expression in to one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral).

train.csv contains two columns, "emotion" and "pixels". The "emotion" column contains a numeric code ranging from 0 to 6, inclusive, for the emotion that is present in the image. The "pixels" column contains a string surrounded in quotes for each image. The contents of this string a space-separated pixel values in row major order. test.csv contains only the "pixels" column and your task is to predict the emotion column.

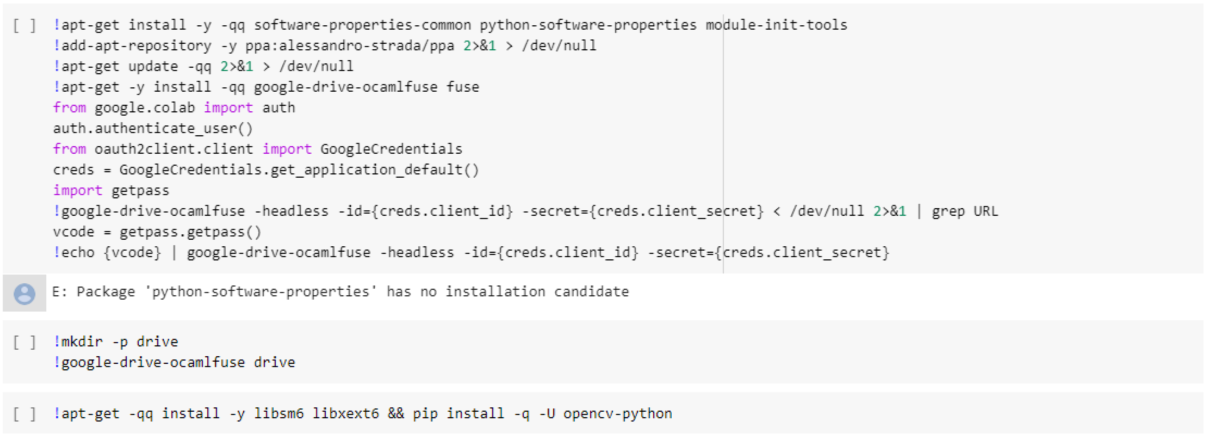
The training set consists of 28,709 examples. The public test set used for the leaderboard consists of 3,589 examples. The final test set, which was used to determine the winner of the competition, consists of another 3,589 examples.

This dataset was prepared by Pierre-Luc Carrier and Aaron Courville, as part of an ongoing research project. They have graciously provided the workshop organizers with a preliminary version of their dataset to use for this contest.
We used to Convolutional Neural Network(CNN) model, if we use R-CNN model instead of CNN model training will be much faster but in the previous step of projects we used CNN model so to the prevent errors and complication we decided to use some structure. According to our experiment about previous steps I decided to support fer2013 dataset with special algorithm thm which took 2000 frames per second with the help of this method validation rate increase %66.6 and winner validiation has 71.161. I separated dataset 80-10-10 ratio for training-validation-test sets.

As mentioned before to the reduce cost we decided to use cloud platforms to the train dataset. First experiment based on Google Collaboraty.

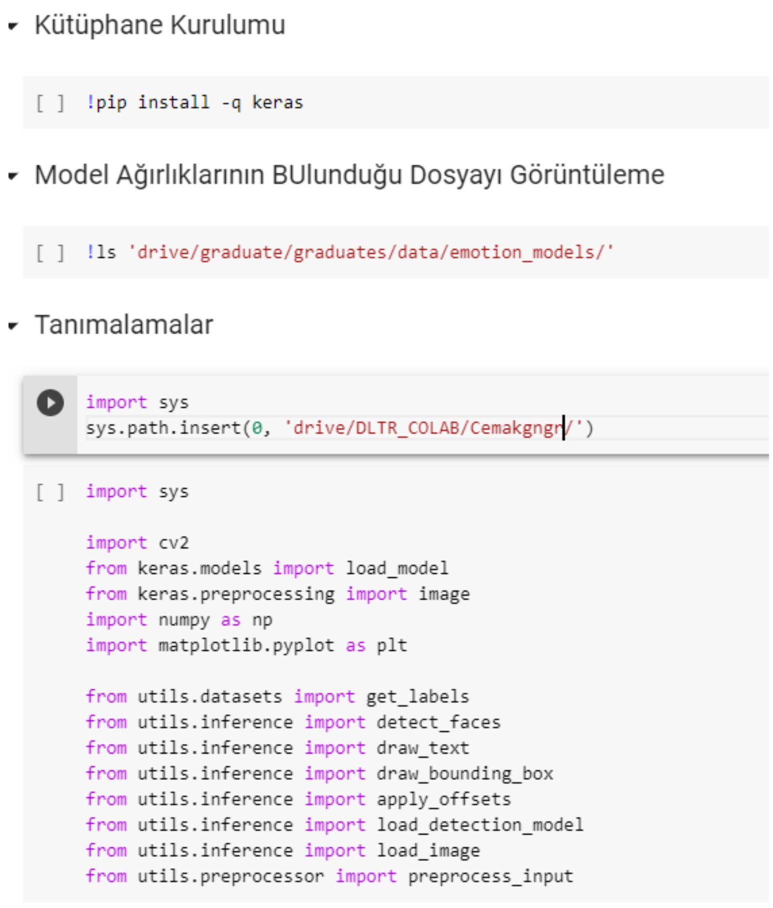
Program Flow:

1.Storing and reading dataset from Google Drive dynamically.



```
[ ] !apt-get install -y -qq software-properties-common python-software-properties module-init-tools
    !add-apt-repository -y ppa:alessandro-strada/ppa 2>&1 > /dev/null
    !apt-get update -qq 2>&1 > /dev/null
    !apt-get -y install -qq google-drive-ocamlfuse fuse
    from google.colab import auth
    auth.authenticate_user()
    from oauth2client.client import GoogleCredentials
    creds = GoogleCredentials.get_application_default()
    import getpass
    !google-drive-ocamlfuse -headless -id={creds.client_id} -secret={creds.client_secret} < /dev/null 2>&1 | grep URL
    vcode = getpass.getpass()
    !echo {vcode} | google-drive-ocamlfuse -headless -id={creds.client_id} -secret={creds.client_secret}

    E: Package 'python-software-properties' has no installation candidate

[ ] !mkdir -p drive
    !google-drive-ocamlfuse drive

[ ] !apt-get -qq install -y libsm6 libxext6 && pip install -q -U opencv-python
```

2. Defining needed libraries and dataset.In this step I used different kind of dataset which is called MINIST.



▾ Kütüphane Kurulumu

```
[ ]  !pip install -q keras
```

▾ Model Ağırlıklarının BUlunduğu Dosyayı Görüntüleme

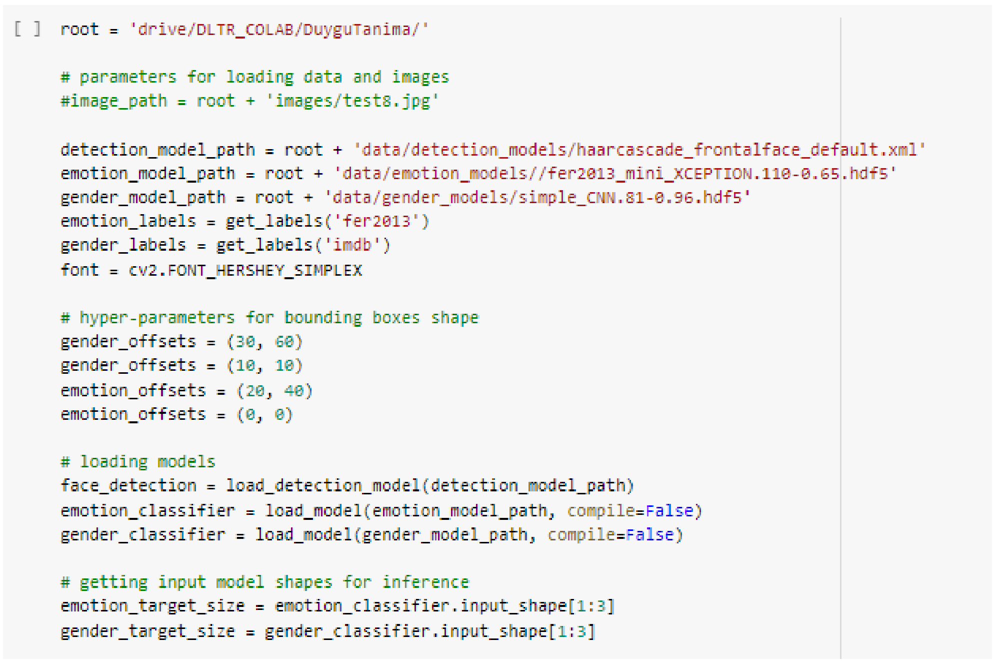
```
[ ]  !ls 'drive/graduate/graduates/data/emotion_models/'
```

▾ Tanımalamalar

```
import sys
sys.path.insert(0, 'drive/DLTR_COLAB/Cemakgngr/')
```

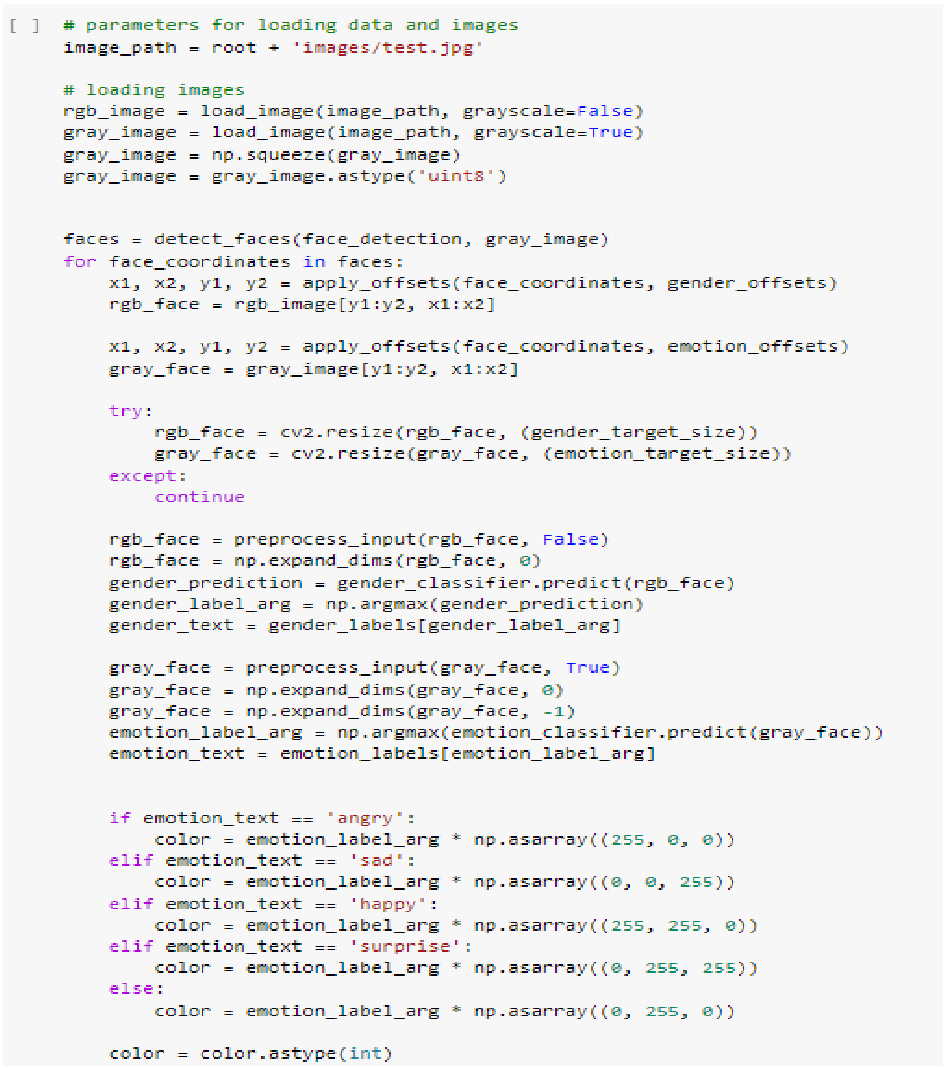
```
[ ]  import sys

     import cv2
     from keras.models import load_model
     from keras.preprocessing import image
     import numpy as np
     import matplotlib.pyplot as plt

     from utils.datasets import get_labels
     from utils.inference import detect_faces
     from utils.inference import draw_text
     from utils.inference import draw_bounding_box
     from utils.inference import apply_offsets
     from utils.inference import load_detection_model
     from utils.inference import load_image
     from utils.preprocessor import preprocess_input
```

3.Loading Trained model weights



```
[ ]  root = 'drive/DLTR_COLAB/DuyguTanima/'

     # parameters for loading data and images
     #image_path = root + 'images/test8.jpg'

     detection_model_path = root + 'data/detection_models/haarcascade_frontalface_default.xml'
     emotion_model_path = root + 'data/emotion_models//fer2013_mini_XCEPTION.110-0.65.hdf5'
     gender_model_path = root + 'data/gender_models/simple_CNN.81-0.96.hdf5'
     emotion_labels = get_labels('fer2013')
     gender_labels = get_labels('imdb')
     font = cv2.FONT_HERSHEY_SIMPLEX

     # hyper-parameters for bounding boxes shape
     gender_offsets = (30, 60)
     gender_offsets = (10, 10)
     emotion_offsets = (20, 40)
     emotion_offsets = (0, 0)

     # loading models
     face_detection = load_detection_model(detection_model_path)
     emotion_classifier = load_model(emotion_model_path, compile=False)
     gender_classifier = load_model(gender_model_path, compile=False)

     # getting input model shapes for inference
     emotion_target_size = emotion_classifier.input_shape[1:3]
     gender_target_size = gender_classifier.input_shape[1:3]
```

4. Model Test



```
[ ]  # parameters for loading data and images
     image_path = root + 'images/test.jpg'

     # loading images
     rgb_image = load_image(image_path, grayscale=False)
     gray_image = load_image(image_path, grayscale=True)
     gray_image = np.squeeze(gray_image)
     gray_image = gray_image.astype('uint8')


     faces = detect_faces(face_detection, gray_image)
     for face_coordinates in faces:
         x1, x2, y1, y2 = apply_offsets(face_coordinates, gender_offsets)
         rgb_face = rgb_image[y1:y2, x1:x2]

         x1, x2, y1, y2 = apply_offsets(face_coordinates, emotion_offsets)
         gray_face = gray_image[y1:y2, x1:x2]

         try:
             rgb_face = cv2.resize(rgb_face, (gender_target_size))
             gray_face = cv2.resize(gray_face, (emotion_target_size))
         except:
             continue

         rgb_face = preprocess_input(rgb_face, False)
         rgb_face = np.expand_dims(rgb_face, 0)
         gender_prediction = gender_classifier.predict(rgb_face)
         gender_label_arg = np.argmax(gender_prediction)
         gender_text = gender_labels[gender_label_arg]

         gray_face = preprocess_input(gray_face, True)
         gray_face = np.expand_dims(gray_face, 0)
         gray_face = np.expand_dims(gray_face, -1)
         emotion_label_arg = np.argmax(emotion_classifier.predict(gray_face))
         emotion_text = emotion_labels[emotion_label_arg]


         if emotion_text == 'angry':
             color = emotion_label_arg * np.asarray((255, 0, 0))
         elif emotion_text == 'sad':
             color = emotion_label_arg * np.asarray((0, 0, 255))
         elif emotion_text == 'happy':
             color = emotion_label_arg * np.asarray((255, 255, 0))
         elif emotion_text == 'surprise':
             color = emotion_label_arg * np.asarray((0, 255, 255))
         else:
             color = emotion_label_arg * np.asarray((0, 255, 0))

         color = color.astype(int)
```

Figure 10. Expressive face reference dataset from MINIST

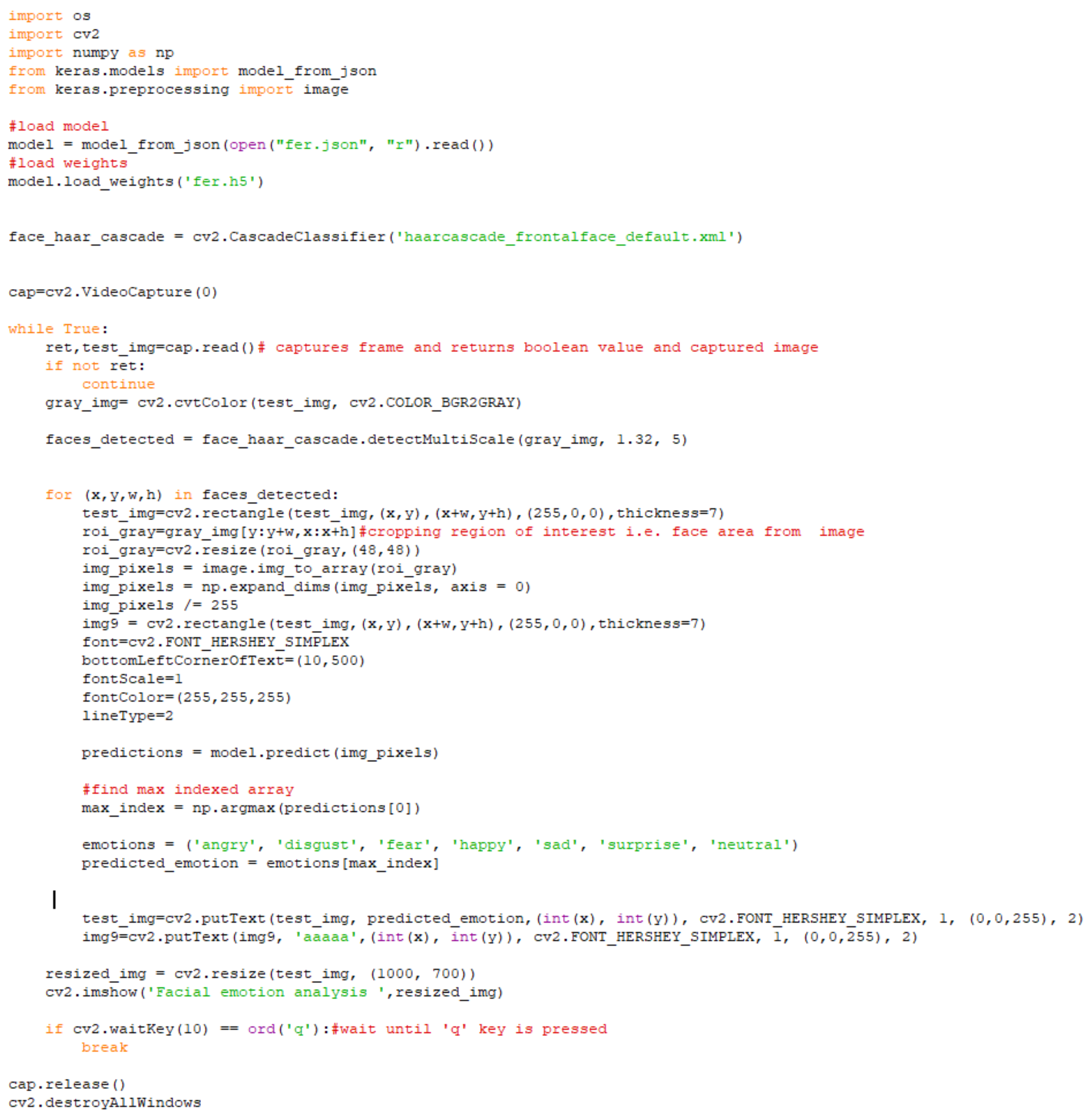
In this experiment I did not take good result for validation.However actual problem was caused by Google Services because of some technical problem they withdrew their support.Therefore I had to train my dataset with the using local gpu but Raspberry Pi's GPU is not enough to traning process.Therefore I used SSSH protocol to traing dataset with using my personal computer then load to Raspberry Pi.

In the local training I used pyhton language and pretrained model from collab.Training process take 16 hours.Because I had to reconstruct previous models layouts and compiling model with using Adam method.Also I had to save model as json to the able to use it again because without drive we needed to storage our model.Project consist of two python file first one for the testing, traning proccess and the second one about to open-cv and labelling procces.Code block for the first python file shown below.

```
import sys, os
import pandas as pd
import numpy as np

from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D, BatchNormalization,AveragePooling2D
from keras.losses import categorical_crossentropy
from keras.optimizers import Adam
from keras.regularizers import l2
from keras.utils import np_utils
# pd.set_option('display.max_rows', 500)
# pd.set_option('display.max_columns', 500)
# pd.set_option('display.width', 1000)

df=pd.read_csv('fer2013.csv')

# print(df.info())
# print(df["Usage"].value_counts())

# print(df.head())
X_train,train_y,X_test,test_y=[],[],[],[]

for index, row in df.iterrows():
    val=row['pixels'].split(" ")
    try:
        if 'Training' in row['Usage']:
           X_train.append(np.array(val,'float32'))
           train_y.append(row['emotion'])
        elif 'PublicTest' in row['Usage']:
           X_test.append(np.array(val,'float32'))
           test_y.append(row['emotion'])
    except:
        print(f"error occured at index :{index} and row:{row}")


num_features = 64
num_labels = 7
batch_size = 64
epochs = 30
width, height = 48, 48


X_train = np.array(X_train,'float32')
train_y = np.array(train_y,'float32')
X_test = np.array(X_test,'float32')
test_y = np.array(test_y,'float32')

train_y=np_utils.to_categorical(train_y, num_classes=num_labels)
test_y=np_utils.to_categorical(test_y, num_classes=num_labels)

#cannot produce
#normalizing data between oand 1
X_train -= np.mean(X_train, axis=0)
X_train /= np.std(X_train, axis=0)

X_test -= np.mean(X_test, axis=0)
X_test /= np.std(X_test, axis=0)

X_train = X_train.reshape(X_train.shape[0], 48, 48, 1)
```

```
X_test = X_test.reshape(X_test.shape[0], 48, 48, 1)

# print(f"shape:{X_train.shape}")
##designing the cnn
#1st convolution layer
model = Sequential()

model.add(Conv2D(64, kernel_size=(3, 3), activation='relu', input_shape=(X_train.shape[1:])))
model.add(Conv2D(64,kernel_size= (3, 3), activation='relu'))
# model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2), strides=(2, 2)))
model.add(Dropout(0.5))

#2nd convolution layer
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))
# model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2), strides=(2, 2)))
model.add(Dropout(0.5))

#3rd convolution layer
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(Conv2D(128, (3, 3), activation='relu'))
# model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2), strides=(2, 2)))

model.add(Flatten())

#fully connected neural networks
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(num_labels, activation='softmax'))

# model.summary()

#Compliling the model
model.compile(loss=categorical_crossentropy,
              optimizer=Adam(),
              metrics=['accuracy'])

#Training the model
model.fit(X_train, train_y,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(X_test, test_y),
          shuffle=True)


#Saving the  model to  use it later on
fer_json = model.to_json()
with open("fer.json", "w") as json_file:
    json_file.write(fer_json)
model.save_weights("fer.h5")
```

The second pyhton file contains labeling, video capturing, and localization proccess and include interface connection with Raspberry Pi with using SSSH method.



```
import os
import cv2
import numpy as np
from keras.models import model_from_json
from keras.preprocessing import image

#load model
model = model_from_json(open("fer.json", "r").read())
#load weights
model.load_weights('fer.h5')


face_haar_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')


cap=cv2.VideoCapture(0)

while True:
    ret,test_img=cap.read()# captures frame and returns boolean value and captured image
    if not ret:
        continue
    gray_img= cv2.cvtColor(test_img, cv2.COLOR_BGR2GRAY)

    faces_detected = face_haar_cascade.detectMultiScale(gray_img, 1.32, 5)


    for (x,y,w,h) in faces_detected:
        test_img=cv2.rectangle(test_img,(x,y),(x+w,y+h),(255,0,0),thickness=7)
        roi_gray=gray_img[y:y+w,x:x+h]#cropping region of interest i.e. face area from  image
        roi_gray=cv2.resize(roi_gray,(48,48))
        img_pixels = image.img_to_array(roi_gray)
        img_pixels = np.expand_dims(img_pixels, axis = 0)
        img_pixels /= 255
        img9 = cv2.rectangle(test_img,(x,y),(x+w,y+h),(255,0,0),thickness=7)
        font=cv2.FONT_HERSHEY_SIMPLEX
        bottomLeftCornerOfText=(10,500)
        fontScale=1
        fontColor=(255,255,255)
        lineType=2

        predictions = model.predict(img_pixels)

        #find max indexed array
        max_index = np.argmax(predictions[0])

        emotions = ('angry', 'disgust', 'fear', 'happy', 'sad', 'surprise', 'neutral')
        predicted_emotion = emotions[max_index]


        test_img=cv2.putText(test_img, predicted_emotion,(int(x), int(y)), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 2)
        img9=cv2.putText(img9, 'aaaaa',(int(x), int(y)), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,255), 2)

    resized_img = cv2.resize(test_img, (1000, 700))
    cv2.imshow('Facial emotion analysis ',resized_img)

    if cv2.waitKey(10) == ord('q'):#wait until 'q' key is pressed
        break

cap.release()
cv2.destroyAllWindows
```

Test result for emotions recognation:

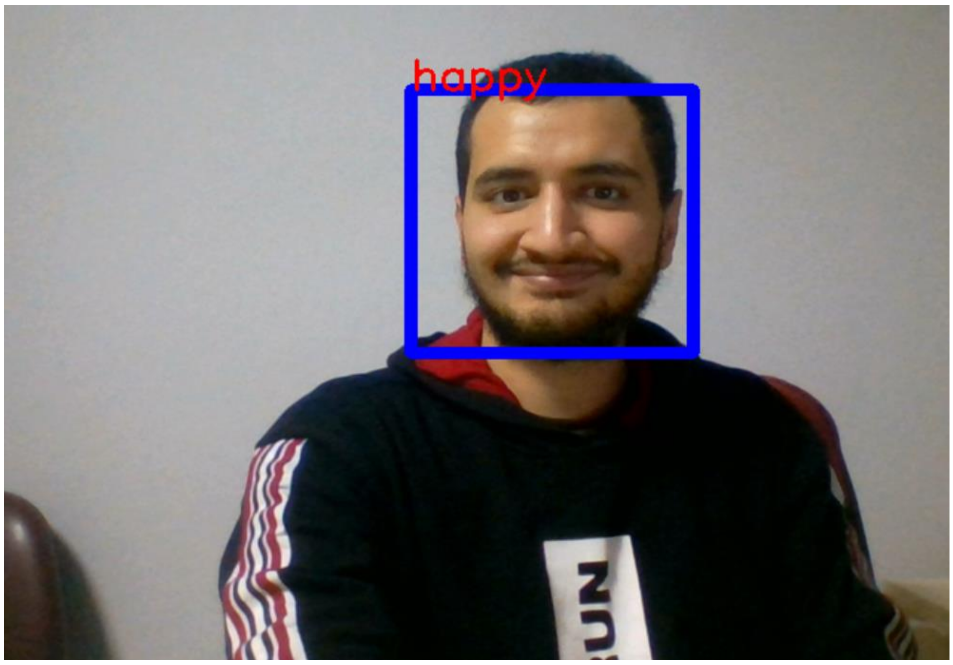


Figure 11. Emotions 'Happy'

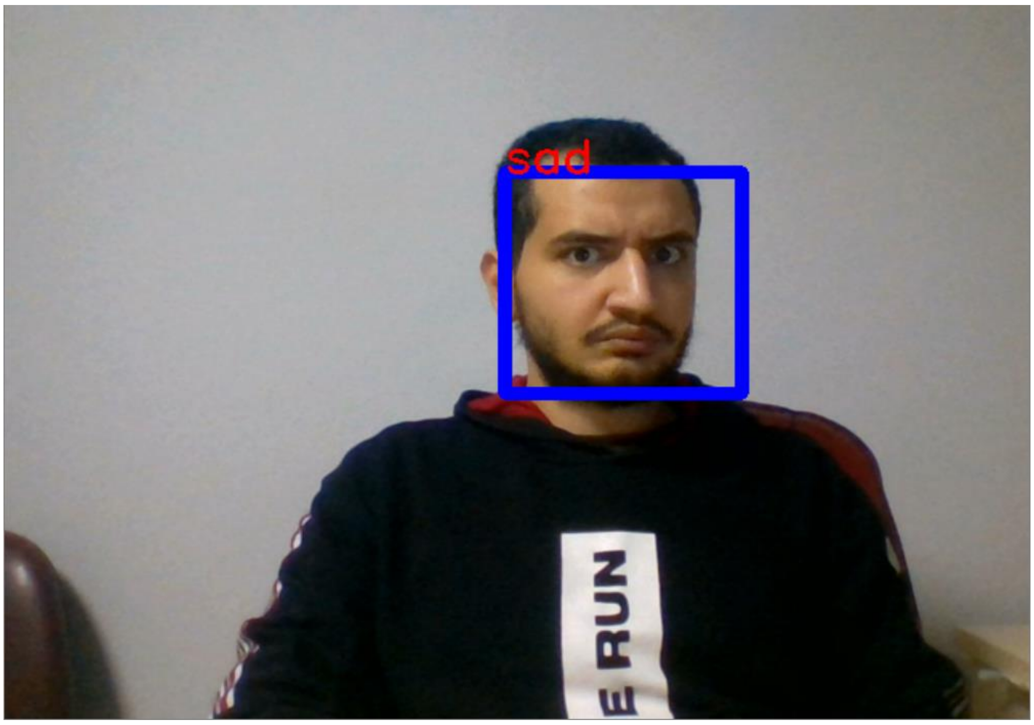


Figure 12. Emotions 'Sad'

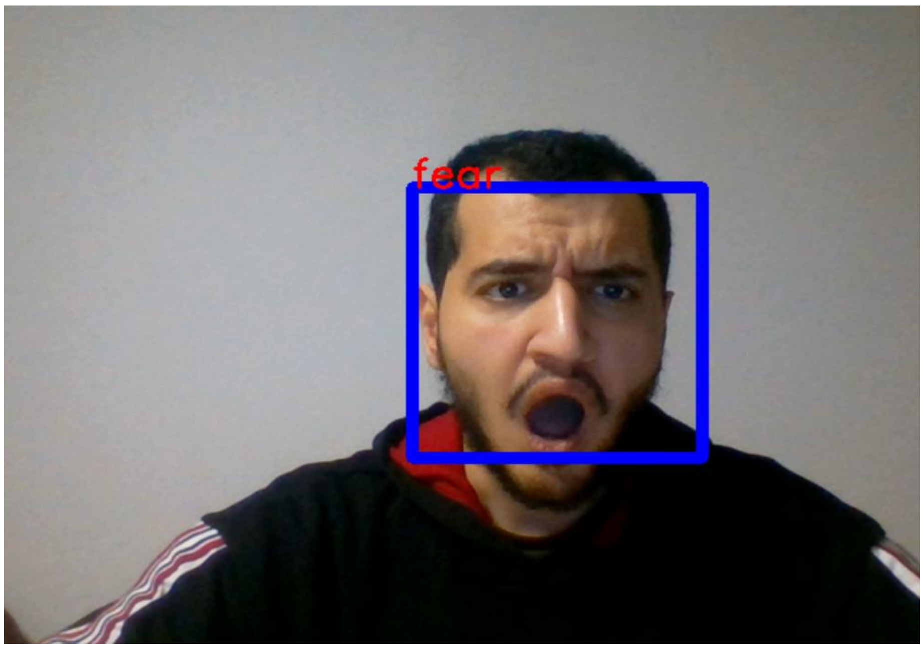


Figure 13. Emotions ‘Fear’

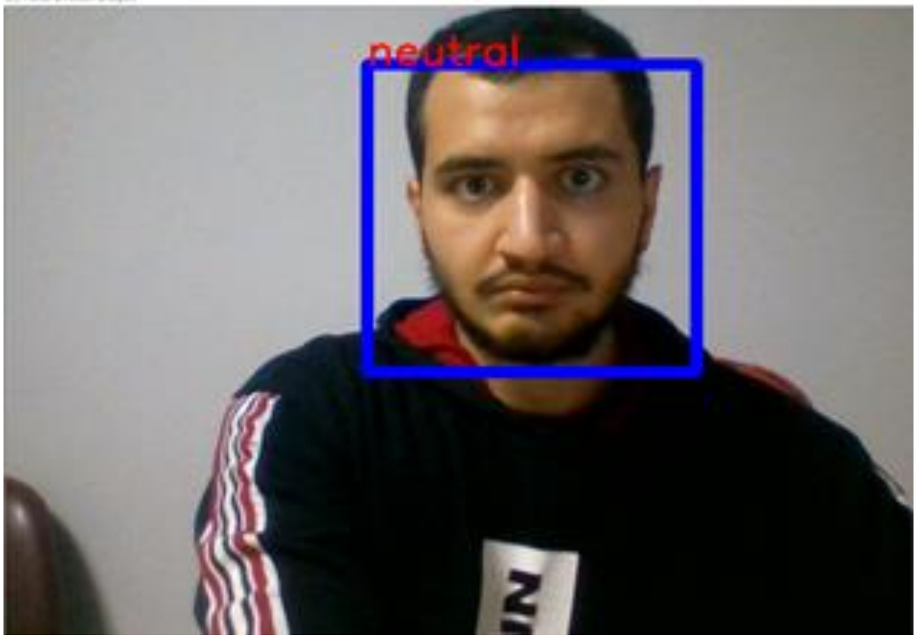


Figure 14. Emotions ‘Neutral’

- **RASPBERRY Pİ 4 HARDWARE CONFİGURATİON:**

  Raspberry pi consist of two servo motor one infrared camera and lcd.

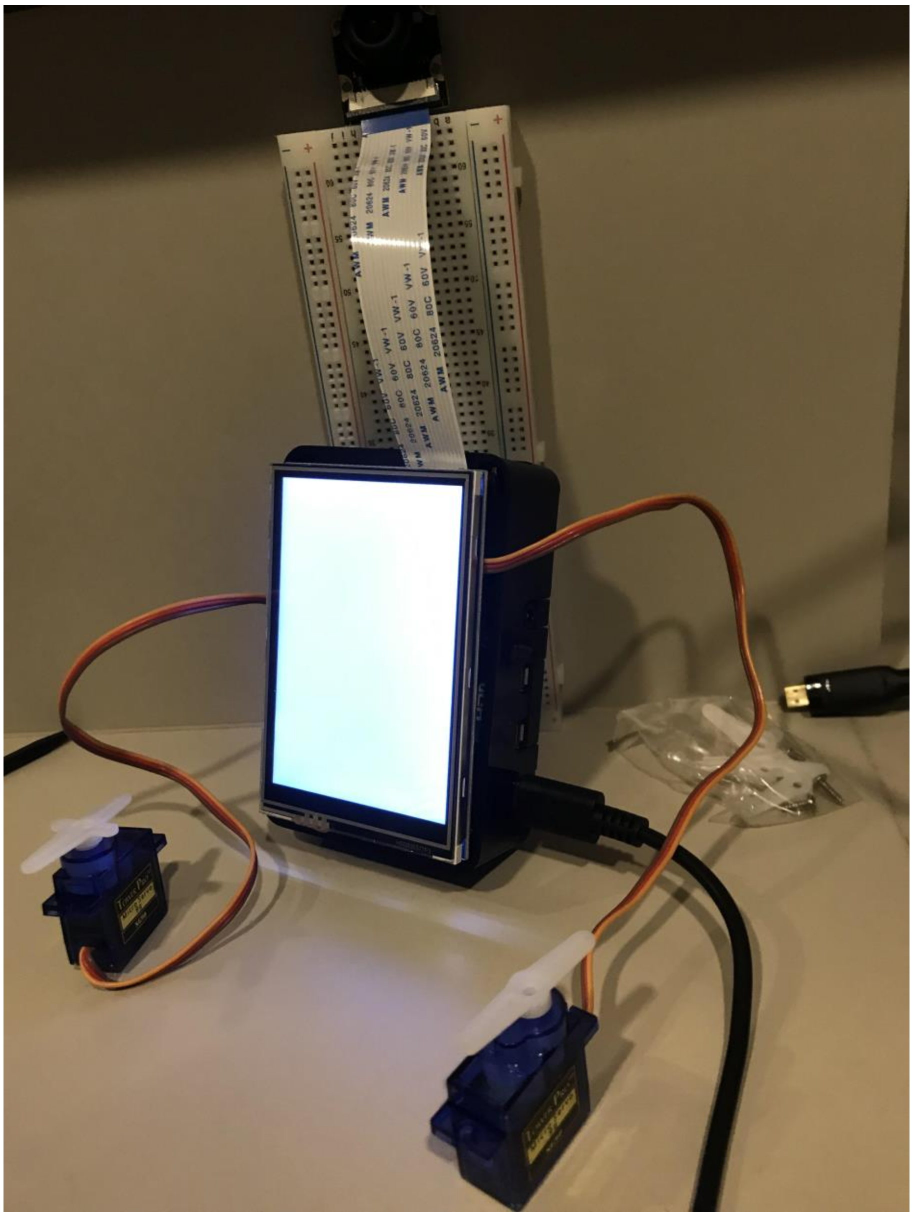


Figure 15. Raspberyy Pi4

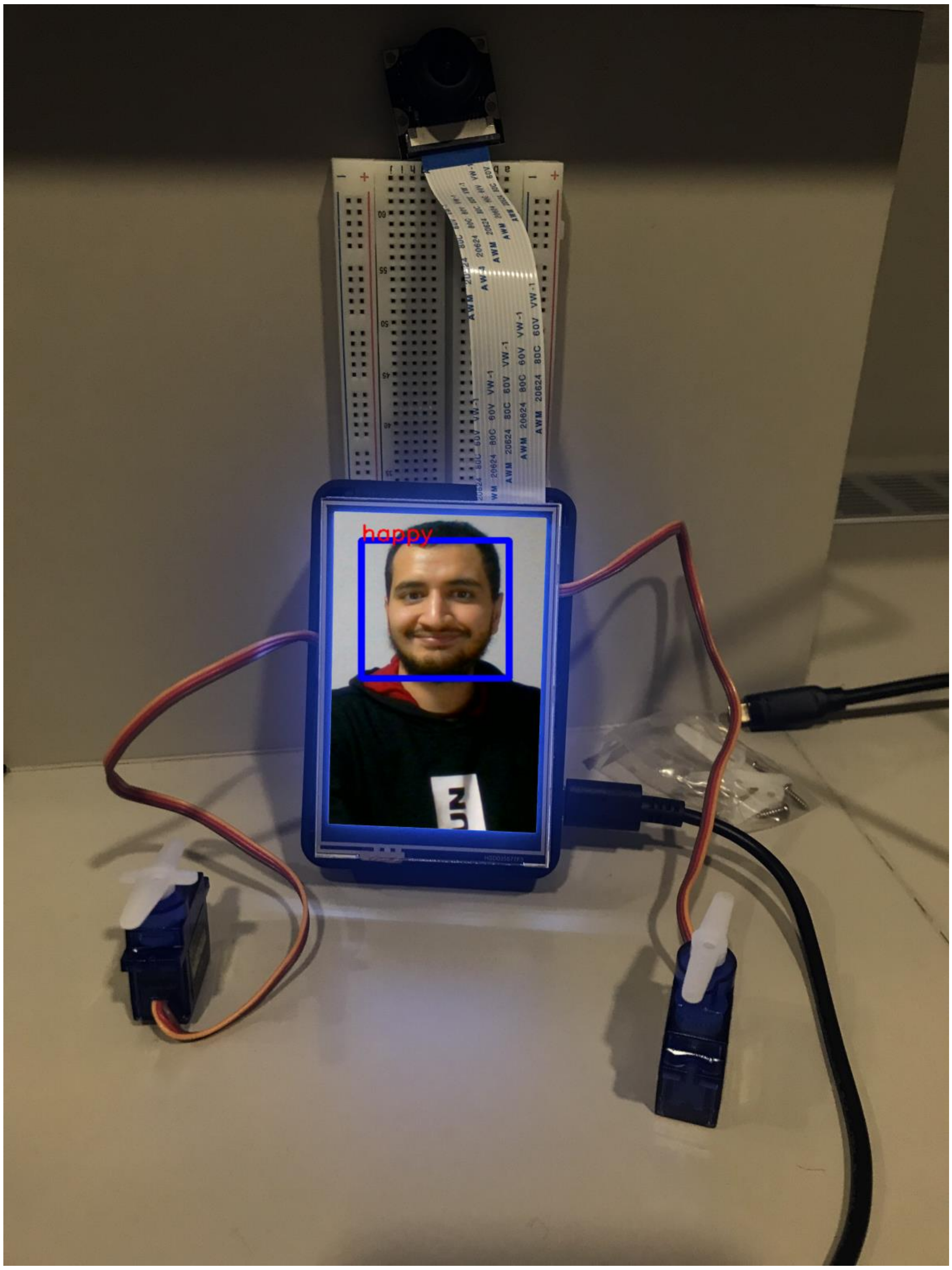


Figure 16. SSSH connection with raspberry pi

4.RESULTS & DISCUSSION

As I mentioned above I tried several methods these are computer vision ,machine learning and using both methods at the same time then I realized problem is about our dataset which is not enough to train our models. Therefore I decided to pass this step and start emotion detection algorithm with the using same model.Because there are many datasets exist about emotions as smiling, crying etc. In the beginning of project I planned to construct strong hardware configuration as using expensive embeded system like Nvidia Cuda then with the help of my instructor's suggestion I turned cloud technology which tecnology reduces costs at 80 percent. Throughout the project I experienced about machine learning algorithms and I trained many dataset for the autistic kids and uploaded public platforms it will be useful in the future. Project is successfully completed without one part which is about adapt hand tracking and facial organs detection progress. In the computer vision part we solved with the using euclidian distance method but our threshold function is not satisfiable to eliminate background effects. Briefly I am failed to training two datasets at the same time based on one test dataset which are hand and facial organs.

5.IMPACT

With the combining several algorithm and methods we reduced time as we needed and costs and we trained different kind of dataset which dataset are focused specific points about autistic children.

6.ETHICAL ISSUES

We just must identify protection of personal data because our project is public and recording data from public also personal private life should be keep secret.

7.PROJECT MANAGEMENT

As I mentioned result part project plans was change due to costs and time. According to these two terms we decided use machine learning algorithms instead of computer vision.

8.CONCLUSION AND FUTURE WORK

Most significant results that I founded are feeding dataset with the webcam frame method and testing two traning dataset with the one dataset I am failed in this part but I have learned many tricks from there. And I experienced about the using pretrained dataset from cloud platforms in the local platform it makes things faster.

9.REFERENCES

[1] Shekhar S. (2016) A Survey on Actuators, Sensors and Control Mechanism Used in Soft Robotics. Retrieved February 25, 2019, from : https://www.academia.edu/29928296/A_Survey_on_Actuators_Sensors_and_Control_Mech anism_Used_in_Soft_Robotics

[2] https://biodesign.seas.harvard.edu/soft-robotics

[3] William B.(1983) An Overview of Artificial Intelligence and Robotics. Retrieved February 25, 2019, from : https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19830023108.pdf

[4] Vernon D. (2007) A Short Introduction to Robotics and AI. Retrieved February 25, 2019, from : http://www.vernon.eu/courses/David_Vernon_Short_Intro_to_Robotics_and_AI.pdf

[5] Sarthak B.(2018) Deep Reinforcement Learning for Soft Robotic Applications: Brief Overview with Impending Challenges. Retrieved February 25, 2019, from : https://www.researchgate.net/publication/329368817_Deep_Reinforcement_Learning_f or_Soft_Robotic_Applications_Brief_Overview_with_Impending_Challenges

[6] https://www.elveflow.com/microfluidic-tutorials/microfluidic-reviews-and-tutorials/soft-robot/

[7] https://softroboticstoolkit.com/pds

[8] https://softroboticstoolkit.com/book/pneumatic-artificial-muscles

[9] (2014) Pneumatic Networks for Soft Robotics that Actuate Rapidly. Retrieved February 25, 2019, from : https://dash.harvard.edu/bitstream/handle/1/25922120/66841469.pdf?sequence =1

[10]Gerboni G.(2018) The incredible potential of flexible, soft robots. Retrieved February 25, 2019, from : https://www.youtube.com/watch?v=AI7M-JTC6_w

[11] https://www.doc.ic.ac.uk/project/examples/2005/163/g0516334/

[12]https://towardsdatascience.com/building-a-deep-learning-model-using-keras-1548ca149d37

[13]https://www.sciencedirect.com/science/article/abs/pii/S106474811260606X