International Journal of Applied Sciences and Smart Technologies

Volume 3, Issue 1, pages 65–82

p-ISSN 2655-8564, e-ISSN 2685-9432

OpenCV Image Processing for AI Pet Robot

Abhishek Ghoshal[1,*], Aditya Aspat[1], Elton Lemos[1]

[1]*Department of Computer Engineering,*
Xavier Institute of Engineering, Mahim, Mumbai, Maharashtra, India
[*]*Corresponding Author: abhighosh98@gmail.com*

(Received 23-07-2020; Revised 21-12-2020; Accepted 21-12-2020)

Abstract

The Artificial Intelligence (AI) Pet Robot is a culmination of multiple fields of computer science. This paper showcases the capabilities of our robot. Most of the functionalities stem from image processing made available through OpenCV. The functions of the robot discussed in this paper are face tracking, emotion recognition and a colour-based follow routine. Face tracking allows the robot to keep the face of the user constantly in the frame to allow capturing of facial data. Using this data, emotion recognition achieved an accuracy of 66% on the FER-2013 dataset. The colour-based follow routine enables the robot to follow the user as they walk based on the presence of a specific colour.

Keywords: OpenCV, image processing, AI Pet robot

1 Introduction

There are many health benefits of owing a pet [1]. They can increase opportunities to exercise, get outside and socialize. There are also a slew of health benefits such as decreased blood pressure, decreased cholesterol levels and better mental health. However, there are also a lot of drawbacks of having to own a pet [2]. There can be issues of disease transmission and infections from pets. There also other problems like restrictions from housing societies, training, cleaning up, extra expenses and spending time with the pet to ensure its well being. There have been numerous cases of cruelties and animals across the world. We propose the development of a cheap “AI Pet” which can help provide the benefits of having a pet. This machine will be able to seamless interact with humans and indulge in activities like real pets can. As this pet will be a machine, a lot of the drawbacks of owing a pet will be nullified such as the transmission of diseases and infections, requirement to spend time, cleaning up.

2 Existing Systems

As part of the research conducted before starting the implementation of the pet robot, we looked through some of the commercially available products. We have also mentioned below the algorithms that we have looked at before creating the different functionalities of the pet robot.

2.1. Commercial Products

Some of the earliest products that resembled an artificial pet can be pinpointed back to the late 1990s. Tamagotchis first went on sale in Japan in 1997. They were small, handheld devices with a screen and buttons. The ‘pet’ on the screen would evolve in various ways depending on user inputs. Over the years there have been many more artificial pets created but pet robots are much more recent. One of the latest products which has been hugely acclaimed is Sony’s Aibo. It boasts of numerous artificial intelligence based features including but not limited to facial recognition, emotion recognition, automatic battery charging and others. However it also comes with a price tag to match it; $1800. There are other pet robots with similar features but all of them cost more than $1000. It is an important consideration for us to be able to limit the development cost of our robot and to use cheap and commercially available parts.

2.2. Face Detection using Haar Cascade

Haar Cascade is a machine learning object detection algorithm proposed by Paul Viola and Michael Jones in their paper “Rapid Object Detection using a Boosted

Cascade of Simple Features" in 2001 [3]. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images. OpenCV offers pre-trained Haar cascade algorithms, organized into categories (faces, eyes and so forth), depending on the images they have been trained on.

We will be using the Haar Cascade algorithm for facial recognition from a live feed. For this we will have to incorporate the "haarcascade frontalface default.xml" into our project. From an input video feed of say 640 X 480 pixels, the xml file applies the algorithm on individual frames. It then returns the X and Y coordinates that help in forming a rectangle around the face of the user. These X and Y coordinates can then be used to cut out the face from the image for further processing.

2.3. Facial Expression Recognition

This implementation is using a Convolutional Neural Network to classify the FER-2013 [4] dataset for the task of emotion recognition. There is also a creation of a real time system for validation of the model. FER-2013 is a large dataset with over 30,000 monochrome images of size 48 X 48 pixels. The initial model achieved an accuracy of 56%. After making changes based on the results, we improved the accuracy to 66%. We will try to further improve this result. Two models were compared in this paper with the aim to get good results with simpler architectures. Initial model that was proposed was a standard fully-convolutional neural network. This model achieved an accuracy of 66% with 35,887 images in the dataset. For real-time images, the input images were pre-processed before being passed to the model. This including conversion of images to grayscale and resizing the images to 48 X 48 pixels as this was the format for the FER-2013 dataset.

3 Implementation Methodology

Our Robot has an architecture closely related to that of a living pet. The Computer works as the brain of the pet, the Pi4B acts as the Spine, the camera as the eyes while the Arduino UNO and its motors act as Limbs.

In this subsection, we describe our hardware design, which includes vision unit, motor unit, communication unit, and interfacing with robot.

- **Vision Unit**

The Vision Unit is responsible for providing vision to the computer for performing Image processing. The Vision Unit comprises of the Pi4B and the Raspberry Pi camera module. The Raspberry Pi camera module captures a video stream and that stream is hosted by the Pi4B by using the RPi-Cam-Web-Interface. The Computer picks up the stream from the hosted site and can now use the data for its programs.

- **Motor Unit**

The Motor Unit is responsible for the movement of the robot. It consists of actuators like DC Motor and Servos that are controlled by the Arduino UNO. The Motor Unit is powered by a 12V Battery. The Arduino is responsible for overseeing the motor operations. However the decision as to what should the Arduino UNO do is taken by the Computer.

- **Communication Unit**

While the robot receives most of its inputs through the mic, camera and other sensors, most of the processing happens on a computer. To be able to send data from the robot to the computer a socket connection for communication has been set up using the TCP protocol. This helps to get information to and fro from the computer and the Pi4B and Arduino Uno on the robot.

- **Interfacing with Robot**

There are two ways to interact with the robot. The user can “shake the hand” of the robot by moving a joystick mounted on the robot and proceed by saying something voice commands. Alternatively commands can also be input in the CLI present on the computer.

4 Implementation

In this section we elaborate on the specific algorithms we have implemented. As our major processing tasks are dominated by image processing, there is extensive use of OpenCV and Deep Learning algorithms in this section.

4.1. Face Recognition

The Robot is trained to remember its creators. We have trained a model to remember our faces using Convolution Neural Network. The Computer takes the data from the camera and compares it with the model. If it does not find a match, it assumes that it does not know that person and hence calls it an unknown face. However if it does find a match, it is able to print out the name associated with the matched face. This Module works in conjunction with the Emotion Recognition Module.

4.2. Emotion Recognition

The Robot is also trained to remember our expressions when we feel a certain emotion like Happiness, Sadness, Fear, Anger and Surprise. For this we have used Convolutional Neural Network. The computer takes the data from the camera and compares it with the model. It will then print out the emotion that matches the most features of the tested data. This module is explained below further in detail.

4.2.1. Dataset

The process began with acquiring the right dataset for our project. This was the FER-2013 Dataset that was also used in [5] and can be acquired from kaggle [4]. This dataset contains 32,557 labeled images of size 48 X 48 pixels. The given labels are Happy, Sad, Angry, Fear, Surprise and Neutral. We performed some basic exploratory data analysis on this dataset and understood the distribution of the data is show in Figure 1.

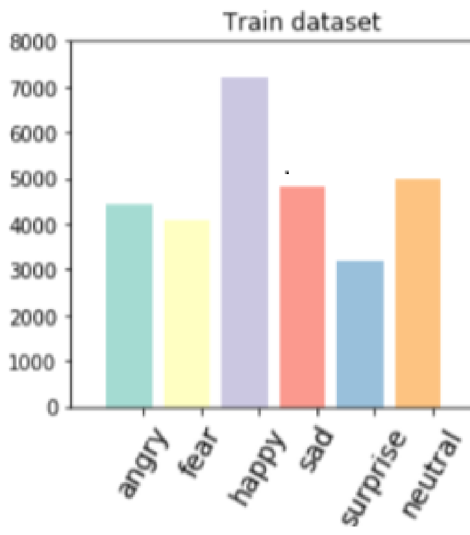


Figure 1. Dataset bar graph

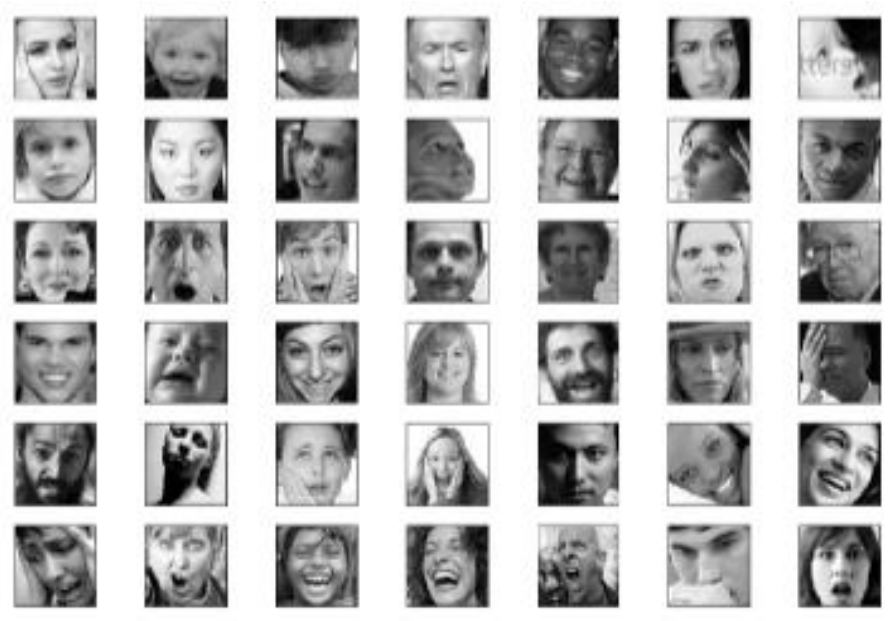


Figure 2. Example of images in dataset

To try to improve the performance of the model, we took a preemptive measure and added 255 images of ourselves spread into the different emotion categories. We took images of our upper bodies and to match our own data to the data in FER-2013, some processing was needed. Figure 2 shows an example of images in dataset. Figure 3 illustrates types of images added. The steps taken to achieve are explained in Figure 4.

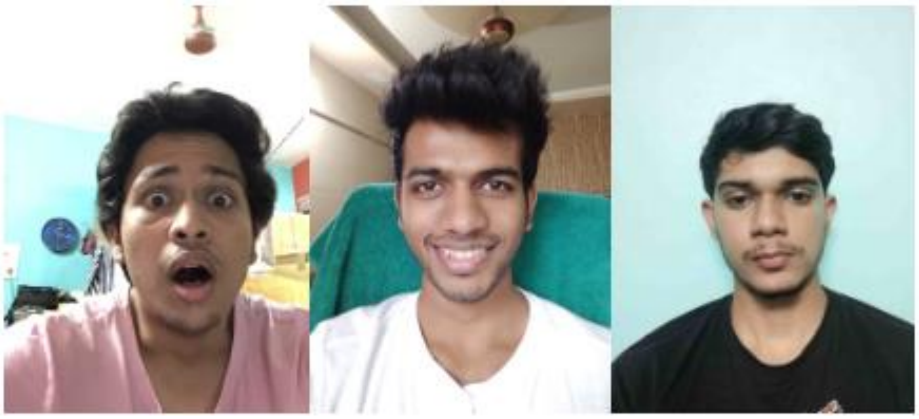


Figure 3. Types of images added

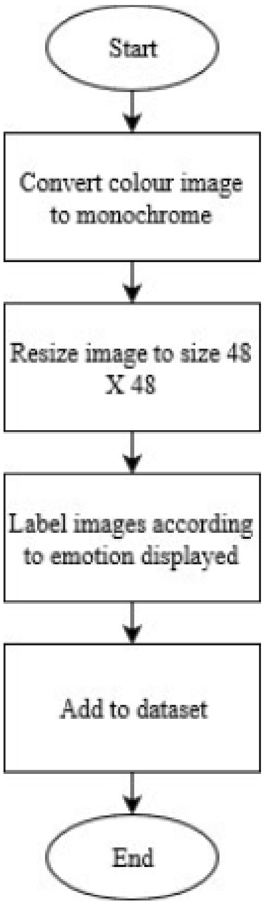

Start
Convert colour image to monochrome
Resize image to size 48 X 48
Label images according to emotion displayed
Add to dataset
End

Figure 4. Pre-processing of images before adding to dataset

Before we could proceed to create the model, the target variable that is the "emotion" column was converted in to a encoded column. This is done because Machine Learning models are only able to perceive numerical values. Words like "happy" or "sad" hold no meaning for it. Hence, we convert such columns using encoding that replaces the possible words with numbers. For example, Happy = 1, Sad = 2, Fear = 3 and so on.

4.2.2. Training

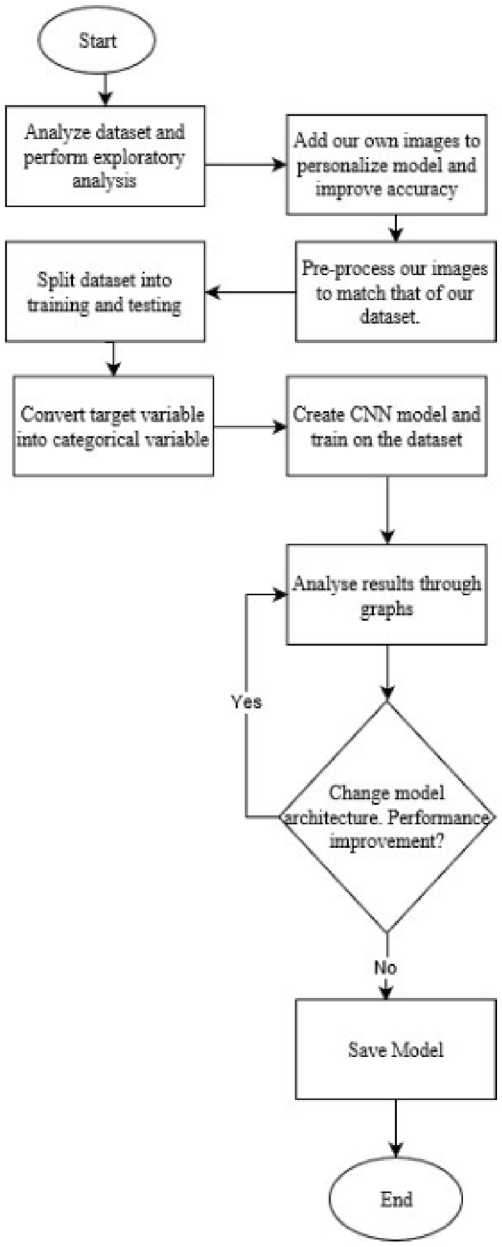
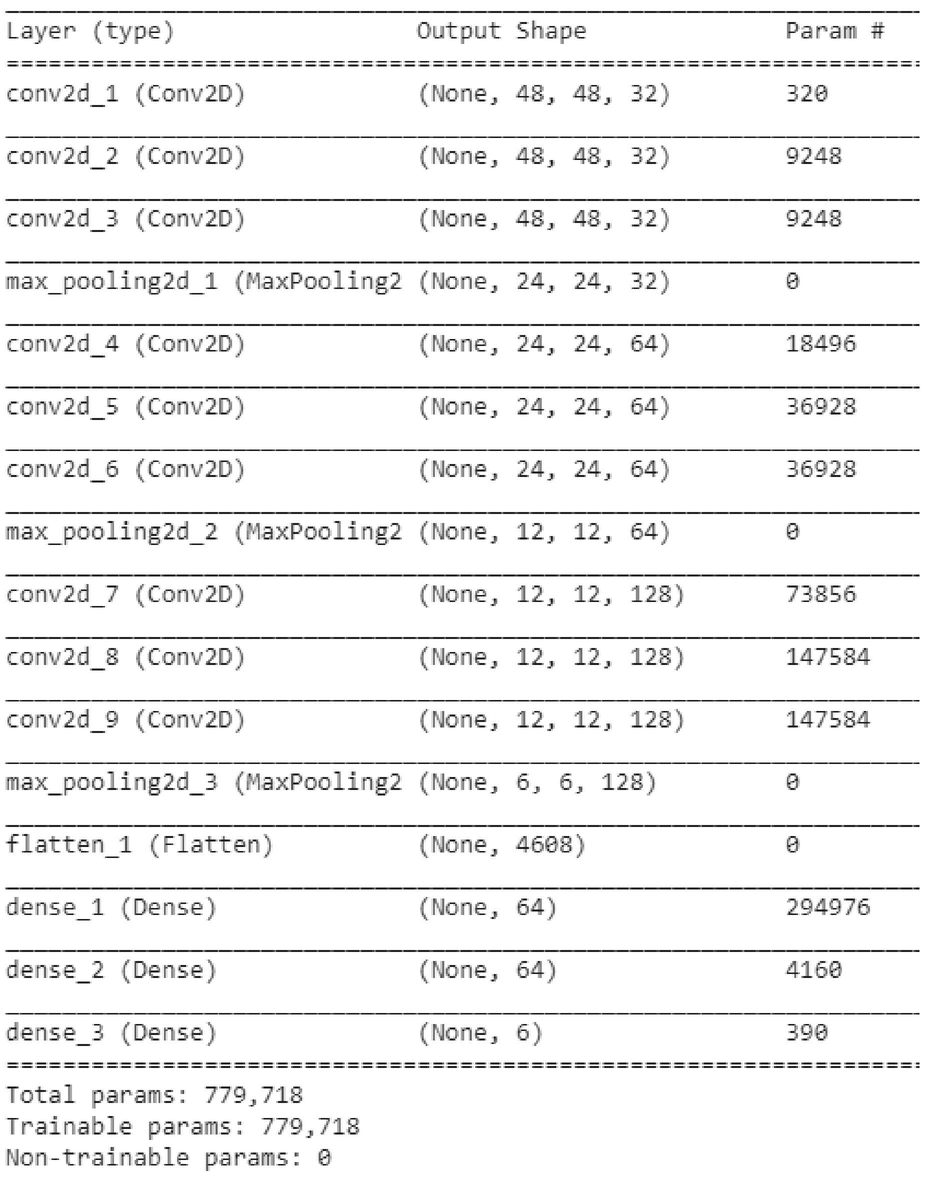

Start
Analyze dataset and perform exploratory analysis
Add our own images to personalize model and improve accuracy
Pre-process our images to match that of our dataset.
Split dataset into training and testing
Convert target variable into categorical variable
Create CNN model and train on the dataset
Analyse results through graphs
Yes
Change model architecture. Performance improvement?
No
Save Model
End

Figure 5. Facial expression recognition flow chart

Figure 5 shows the basic steps taken to perform the training process. These steps have been further elaborated below. We referred to [5] to help get an idea of what type of model needs to be created and what the architecture of this model should be. We created a CNN that takes an input shape array of (48,48) and the final output is an array of probabilities which shows which of the classes the given image most closely belongs to. Figure 6 shows the model architecture.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 48, 48, 32)        320
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 48, 48, 32)        9248
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 48, 48, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 24, 24, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 24, 24, 64)        18496
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 24, 24, 64)        36928
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 24, 24, 64)        36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 12, 12, 64)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 12, 12, 128)       73856
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 12, 12, 128)       147584
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 12, 12, 128)       147584
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4608)              0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                294976
_________________________________________________________________
dense_2 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 390
=================================================================
Total params: 779,718
Trainable params: 779,718
Non-trainable params: 0
```

Figure 6. Model architecture

Looking at the model we can see the input shape of the first layer is 48 x 48 since the size of the images in the dataset in 48 pixels by 48 pixels. The goal of the MaxPooling layers is to reduce the dimensionality of the image so that we can extract meaningful features from it.

4.3. Colour-based Follow Function

The Robot is trained to follow colour. We used the tools available in OpenCV to extract information like color from an image. We set a color range matching a very unique color of a particular shoe we have. The idea is that the robot should not receive presence of multiple areas of the same colour through its camera. Hence we have choosen a relatively unique colour for the follow mechanism; fluorescent yellow.

As shown in [6], we are able to create a circle around the pixels with the required color. This in turn gives us the radius of that circle. Using the size of this radius we then determine the approximate distance of the robot from the user. We have used simple if-condition statements based on the radius of this circle. Along with this we have defined two thresholds experimentally. These thresholds are used with the if-condition statements to give inputs to the motors of the robot.

- If the radius is smaller than lower threshold, move forward (Robot is too far from the user).
- If radius is greater than the upper threshold, move back (Robot is to close to the user).
- If radius is between lower and upper threshold, stay.

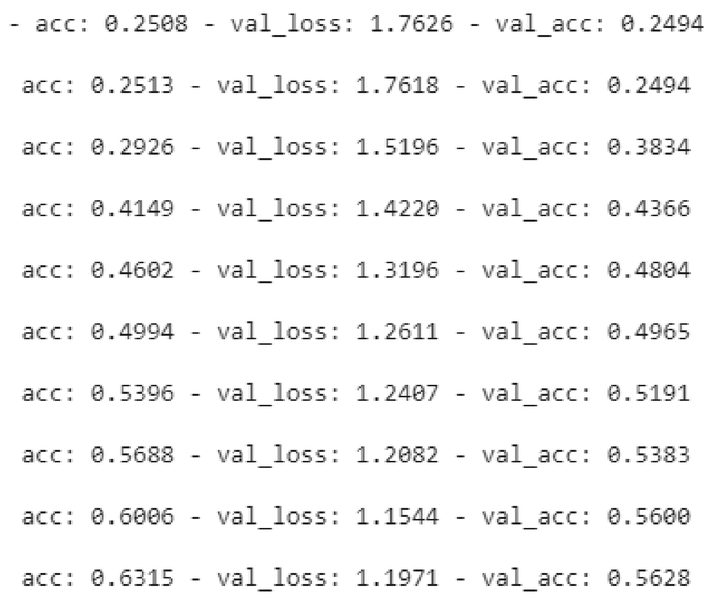
Using the location of the circle on the screen we have also defined conditions for the robot to turn left or right.

5 Analysis

We provide analysis in this section. This includes emotion recognition and colour-based follow function.

5.1. Emotion Recognition

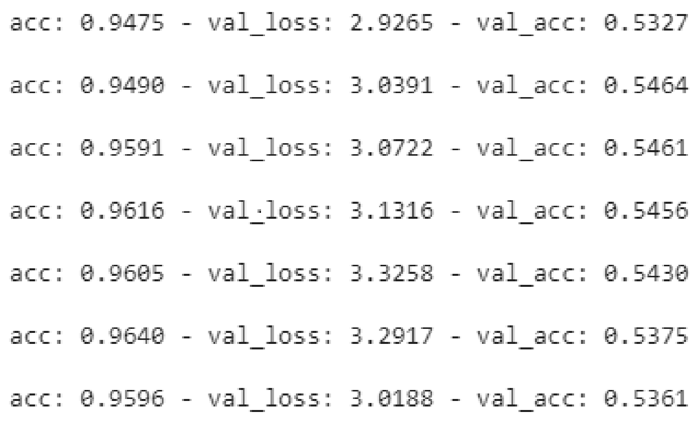
Following is the performance of our model. We achieved an accuracy of 56% in the testing set however we find that we have greater accuracy in real time testing conditions of specific emotions like happy, angry, neutral and surprise. Figure 7 shows the performance of this model through the epochs.



```
- acc: 0.2508 - val_loss: 1.7626 - val_acc: 0.2494

acc: 0.2513 - val_loss: 1.7618 - val_acc: 0.2494

acc: 0.2926 - val_loss: 1.5196 - val_acc: 0.3834

acc: 0.4149 - val_loss: 1.4220 - val_acc: 0.4366

acc: 0.4602 - val_loss: 1.3196 - val_acc: 0.4804

acc: 0.4994 - val_loss: 1.2611 - val_acc: 0.4965

acc: 0.5396 - val_loss: 1.2407 - val_acc: 0.5191

acc: 0.5688 - val_loss: 1.2082 - val_acc: 0.5383

acc: 0.6006 - val_loss: 1.1544 - val_acc: 0.5600

acc: 0.6315 - val_loss: 1.1971 - val_acc: 0.5628
```

Figure 7. Model performance per epoch

In the figure above we have shown the performance of the model for 10 epochs. We can see that the model’s performance on the training set starts surpassing its performance on the validation set. This gap widens as we train for more epochs as shown below in Figure 8.



```
acc: 0.9475 - val_loss: 2.9265 - val_acc: 0.5327

acc: 0.9490 - val_loss: 3.0391 - val_acc: 0.5464

acc: 0.9591 - val_loss: 3.0722 - val_acc: 0.5461

acc: 0.9616 - val_loss: 3.1316 - val_acc: 0.5456

acc: 0.9605 - val_loss: 3.3258 - val_acc: 0.5430

acc: 0.9640 - val_loss: 3.2917 - val_acc: 0.5375

acc: 0.9596 - val_loss: 3.0188 - val_acc: 0.5361
```

Figure 8. Overfitting of model

As we see that the training accuracy of the model crosses 90% whereas the validation accuracy actually starts dropping from 56%. This suggests that our model is overfitting the dataset. Figure 9 shows the accuracy and loss representation of our results.

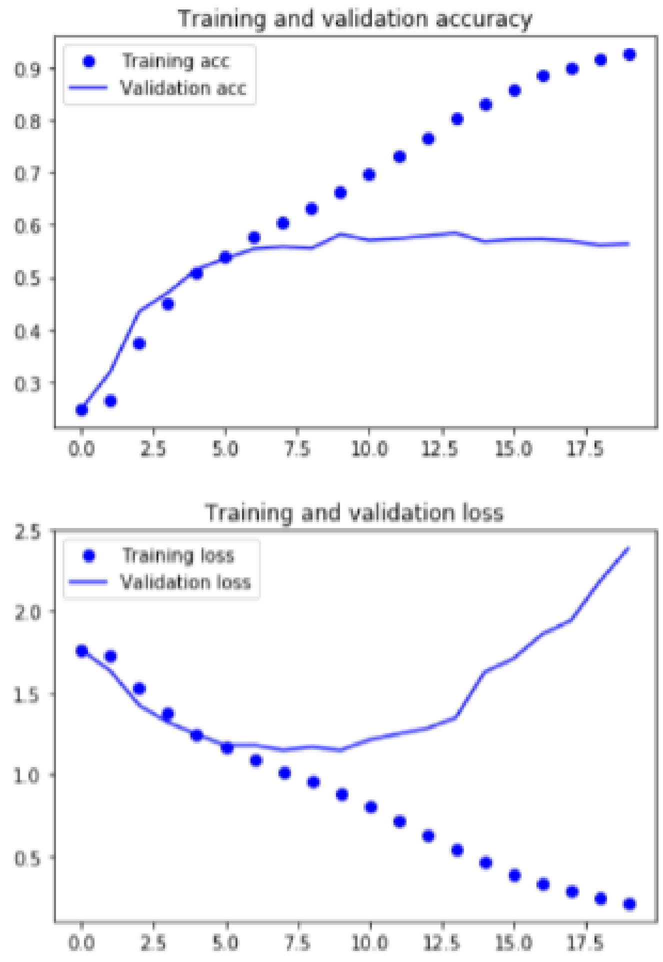


Figure 9. Accuracy and loss representation

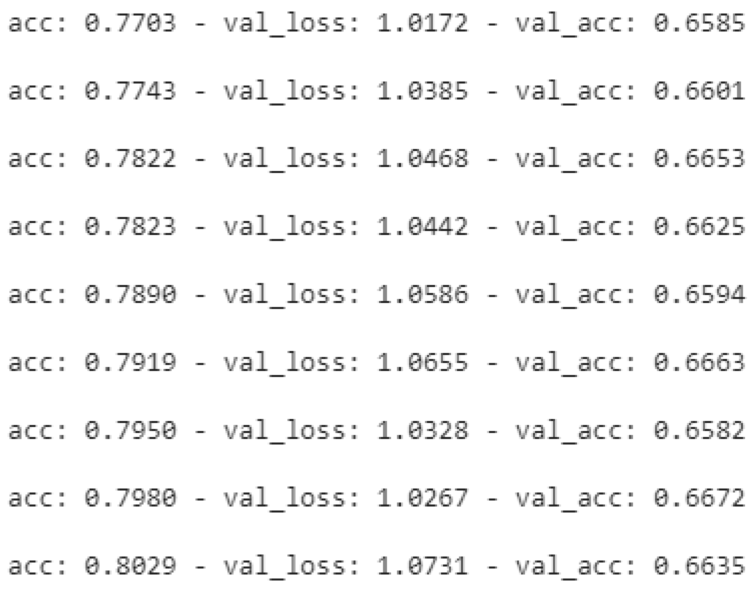
We can see from both these graphs that after a certain point, training accuracy keeps on increasing but validation accuracy plateaus out. This further shows us that the model is overfitting the data and certain measures need to be taken. Reference [7] suggests a few steps we can try to stop our CNN from overfitting a dataset. The steps we have used are:

1. Use a simpler model.
2. Add Regularization.
3. Use batch normalization.

4. Use dropout.

Let us understand what these do to our model. References [7,8] explains that regularization works on the assumption that using smaller weights simplifies the model and prevents overfitting. L2 Regularization or Ridge Regularization term is the sum of square of all feature weights as shown in the equation below. L2 forces weights to be small but does not make then zero. Batch Normalization is a relatively new concept that was introduced after the VGG model. It is recommended to do this process for every model. This adds a normalization layer which helps the model to converge much faster in training. This in turn allows us to use higher learning rates. Another common measure used to stop overfitting is to use dropout. This randomly sets the activations of neurons to 0.

After taking into account the aforementioned steps, a new model was created which uses L2 Regularization, Batch Normalization, and Dropout layers. These help the model break past the previous highest accuracy by a large margin. We were able to achieve and accuracy of 66%. Figure 10 shows the performance of our model in different epochs.



```
acc: 0.7703 - val_loss: 1.0172 - val_acc: 0.6585

acc: 0.7743 - val_loss: 1.0385 - val_acc: 0.6601

acc: 0.7822 - val_loss: 1.0468 - val_acc: 0.6653

acc: 0.7823 - val_loss: 1.0442 - val_acc: 0.6625

acc: 0.7890 - val_loss: 1.0586 - val_acc: 0.6594

acc: 0.7919 - val_loss: 1.0655 - val_acc: 0.6663

acc: 0.7950 - val_loss: 1.0328 - val_acc: 0.6582

acc: 0.7980 - val_loss: 1.0267 - val_acc: 0.6672

acc: 0.8029 - val_loss: 1.0731 - val_acc: 0.6635
```

Figure 10. New model performance per epoch

We can see by comparing Figure 8 and Figure 10 (previous model and current model) that the amount of overfitting is less in the new model. The accuracy has also increased significantly.

5.2. Colour-Based Follow Function

This follow function was tested in two different environments; indoors and outdoors. Due to the lack of proximity sensors in our robot we were worried about it crashing into other objects. However due to the colour-based following, these issues were not encountered. The robot simply did not move towards an object if it was not the required color. Due to our choosen color being rarely encountered in the everyday environment, we did not experience any unwanted behaviour during our testing. In Figure 11 we can see the robot's view of the follow routine. We have stuck a fluorescent yellow coloured paper to the shoe to emulate a sticker or an entire shoe that can be an accessory for the robot. In this example, we observe that the size of the circle is relatively small. This would prompt the robot to move towards the user. Depending on further movements from the user, the robots movements will also chance.

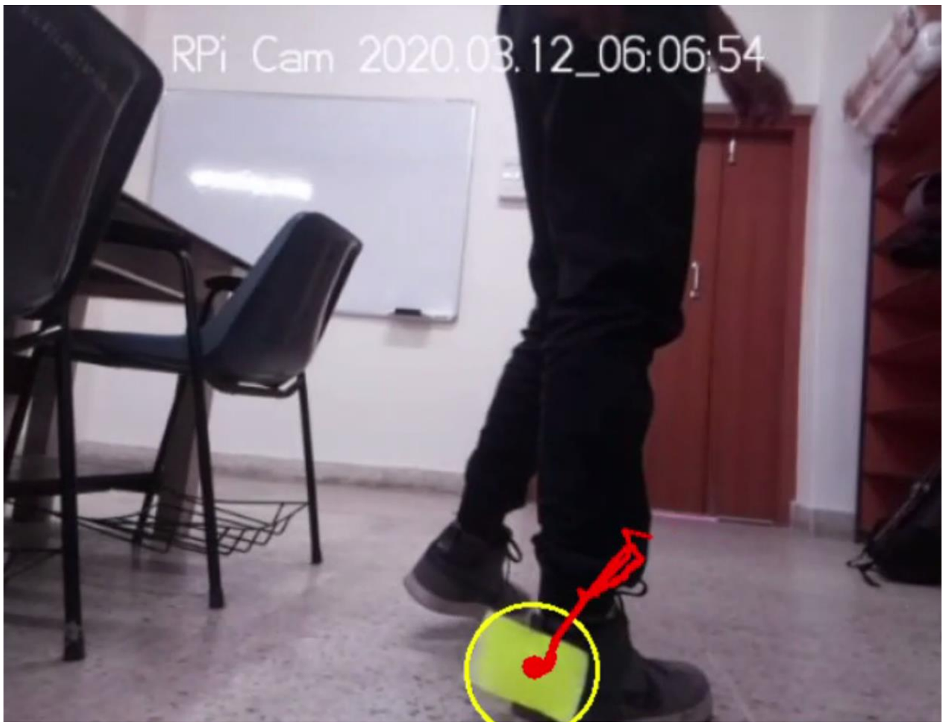


Figure 11. Real time follow functionality

6 Results and Discussion

Our results are provided in this section consisting of colour-based follow function and real time results for emotion recognition.

6.1. Colour-Based Follow Function

Using OpenCV and a normal camera we have created a basic distance measurement system. This enables the robot to judge how far or close it is to our desired colour in the frame. Our system is able to perform the required function in both indoor and outdoor environments provided there is ample illumination.

One of the inherent issues with the system is that when there is presence of more than one object with the desired colour, the robot cannot distinguish between the two. We would recommend using another kind of tagging mechanism which the robot can follow. For example, using RFID tags would prevent the robot from following any unwanted objects.

6.2. Real Time results for Emotion Recognition

After we have saved the model, we need to run this model for real time data. The real time module begins with the computer receiving the video feed from the Pi Camera mounted on the Raspberry Pi 4. Haar Cascade is used for finding a face in the individual frames of the video feed. If a face is not found in this step, Haar Cascade is run again on the subsequent frames till the face is found. In the meanwhile, the camera that is mounted on servos begin to sweep its 1800 field of view in a systematic way. If a face is found in this sweep, the motion stops and a snapshot of the frame is taken for further processing. The face is cut out from the entire frame and the resulting image is converted to monochrome and then resized to 48 X 48 pixels before it can be fed to the model for making a prediction:

1. A picture is captured from the camera (see Figure 12).
2. The Face is extracted using Haar-Cascade and converted to gray scale (see Figure 13).
3. This is then fed to the model and the model returns the predicted emotion (see Figure 14).

4. The final prediction can be seen on the screen (see Figure 15).

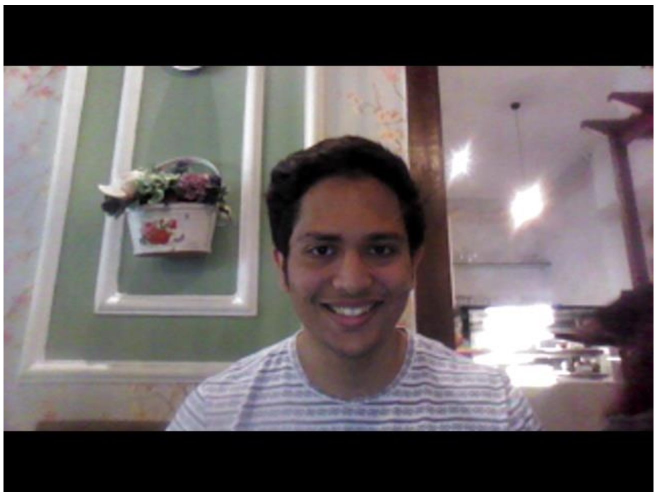


Figure 12. Emotion recognition input image

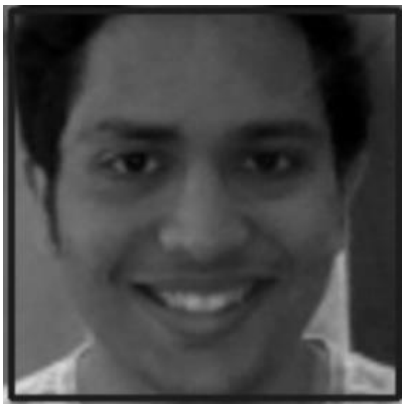
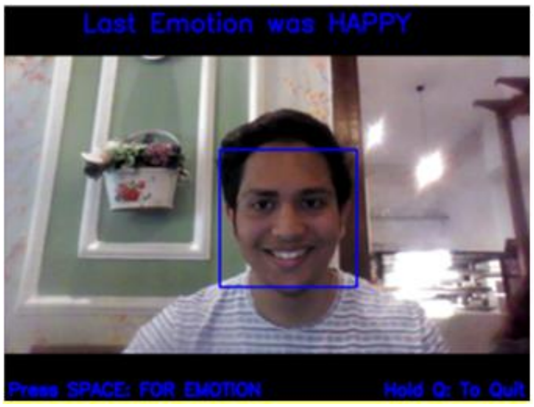


Figure 13. Grayscaling and face cutout



Last Emotion was HAPPY

Press SPACE: FOR EMOTION

Hold Q: To Quit

Figure 14. Emotion recognition output

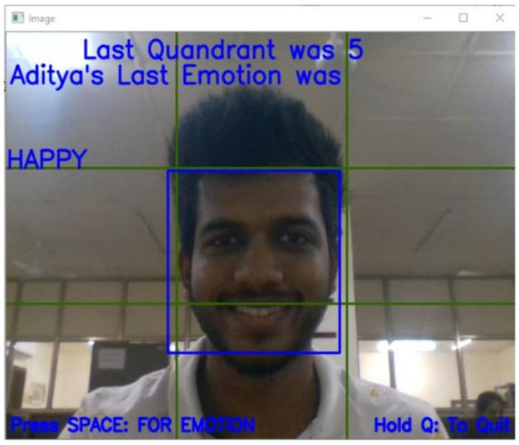


Figure 15. Detecting happy emotion with face recognition

7 Conclusion

The proposed algorithms act as some of the functions that our AI Pet Robot can perform. It is capable of differentiating between a known person and an unknown person. This will allow implementation of further functions such as "Sentry Mode" where the bot could sound an alarm if there is presence of unknown people. The robot is also capable of recognizing the emotions of the user. This could be a useful feature for someone undergoing therapy or counselling. Finally the follow functionality can allow the user to take the AI Pet Robot on walks like he would take any other pet, for example a dog.

References

[1] Center for Disease Control and Prevention, Healthy Pets, Healthy People, https://www.cdc.gov/healthypets/health-benefits/index.html, Accessed: 12/01/2020.

[2] E. Paul Cherniack, M. D and Ariella R. Cherniack, “Assessing the benefits and risks of owning a pet.” *Canadian Medical Association Journal*, 2015.

[3] P. Viola and M. Jones, “Rapid Object Detection using a Boosted Cascade of Simple Features.” *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, **1**, 2001.

[4] Challenges in Representation Learning: Facial Expression Recognition Challenge, https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data, 2013.

[5] N. Poornadithya C., P.C. Chengappa, T. Raman, S. Pandey and G. K. Shyam, "Emotion Identification and Classification using Convolutional Neural Networks." *International Journal of Advanced Research in Computer Science*, 2018.

[6] A. Rosebrock, *Ball Tracking with OpenCV*, https://www.pyimagesearch.com/2015/09/14/ball-tracking-with-opencv/, 2015, Accessed 02/02/2020.

[7] R. Ruizendaal, *Deep Learning #3: More on CNNs Handling Overfitting,* https://towardsdatascience.com/deep-learning-3-more-on-cnns-handling-overfitting-2bd5d99abe5d, 2017, Accessed 23/02/2020.

[8] R. Khandelwal, *L1 and L2 Regularization*, https://medium.com/datadriveninvestor/l1-l2-regularization-7f1b4fe948f2, Nov. 4, 2018, Accessed 23/02/2020.