Indian Institute of Information Technology

allahabad

5th Semester Mini Project

EMOTION CLASSIFICATION USING NAO ROBOT

Submitted to:
Prof. GC Nandi

Submitted By:
Mantek Singh (IIT2016007)
Gagan Ganapathy
(IIT2016038)

Abstract

Facial emotion recognition (FER) is an important topic in the fields of computer vision and artificial intelligence owing to its significant academic and commercial potential. Although FER can be conducted using multiple sensors, this review focuses on studies that exclusively use of real-time video input, because visual expressions are one of the main information channels in interpersonal communication.
This paper provides a brief review of researches in the field of FER conducted over the past decades. First, conventional FER approaches are described along with a summary of the representative categories of FER systems and their main algorithms. Deep-learning-based FER approaches using deep networks enabling "end-to-end" learning are then presented. This review focuses on an up-to-date hybrid deep-learning approach integrating a convolutional neural network (CNN) into a NaO robot, enabling the robot to be able to classify emotions based on facial expressions.

Contents

1 Introduction

The success of service robotics decisively depends on a smooth robot to user interaction. Thus, a robot should be able to extract information just from the face of its user. Interpreting correctly any of these elements using machine learning (ML) techniques has proven to be complicated due the high variability of the samples within each task. This leads to models with millions of parameters trained under thousands of samples. Furthermore, the human accuracy for classifying an image of a face in one of 7 different emotions is 65% $\pm$ 5%.
One can observe the difficulty of this task by trying to manually classify the FER-2013 dataset images as shown in the figure below, within the following classes “angry”, “disgust”, “fear”, “happy”, “sad”, “surprise”, “neutral”.

In spite of these difficulties, robot platforms oriented to attend and solve household tasks require facial expressions systems that are robust and computationally efficient. More- over, the state-of-the-art methods in image-related tasks such as image classification and object detection are all based on Convolutional Neural Networks (henceforth

referred to as CNNs).
These tasks require CNN architectures with millions of parameters; therefore, their deployment in robot platforms and real-time systems becomes unfesible, this paper focuses on the integration of our neural net into a simulated NaO robot.

2 Literature Review and Motivation

The world is observing an exponential growth in the use of personal robots to perform household chores. With he advent of human robot interaction, it is imperative to think of robots talking and interacting with humans the way humans do with each other. A crucial element of this interaction is the robot responding dynamically to the situations around it and having a conversation with the person right in front of it based on the emotions being felt by that person.

According to the stats collected by International Federation of Robotics[1], more than 3 million industrial robots will be in use in factories around the world by 2020. This means that number of robots installed is going to double within the span of just 7 years(between 2014 and 2020). In order to create smart interactive robots we need to ensure that they perform at their best efficiency level at identifying the emotions of the person they can interact with.

Historically,Deep Learning have been used to detect emotions using facial expressions. Currently, researchers use the distances between facial landmarks to detect emotion. A face is represented as the position of the nose, eyes, mouth, cheeks, and other areas, and then the distances between those points are calculated. Then, thresholds are established to detect the emotion. If the face in the image is smiling, the cheek positions would be closer to the eyes, the mouth would be stretched, and the eyes would be squinted. This approach works in controlled settings, but what if you can only see half the face in the image? What if the face is slightly turned? To get accurate facial landmarks, you would have to artificially transform the image so that the face is centered and looking straight at the camera. With a deep learning approach, the model can learn to be flexible and detect features in faces no matter how they're oriented. All you need is data.[2]

For our purpose, we decided to use the Nao Robot. Nao (pronounced now) is an autonomous, programmable humanoid robot developed by Aldebaran Robotics, a French robotics company headquartered in Paris. The robot's development began with the launch of Project Nao in 2004.[3]

3 Background

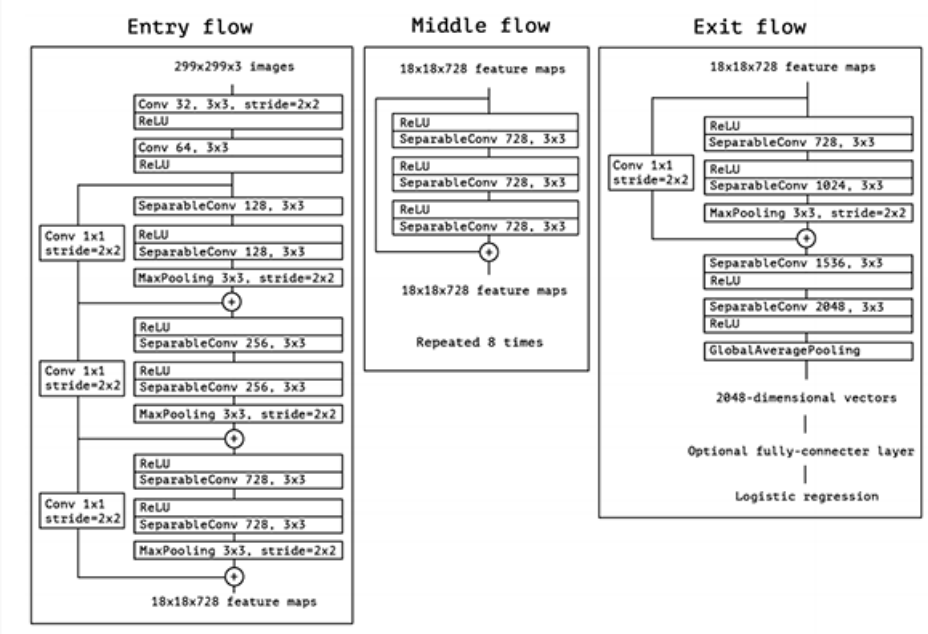
In this section, we describe the architecture and API that we have used inorder to integrate our CNN

3.1 Xception Architecture

Our model is based upon the Xception Architecture explained below having residual connections but is much smaller compared to the original Xception model, which we refer to as the mini-Xception model.

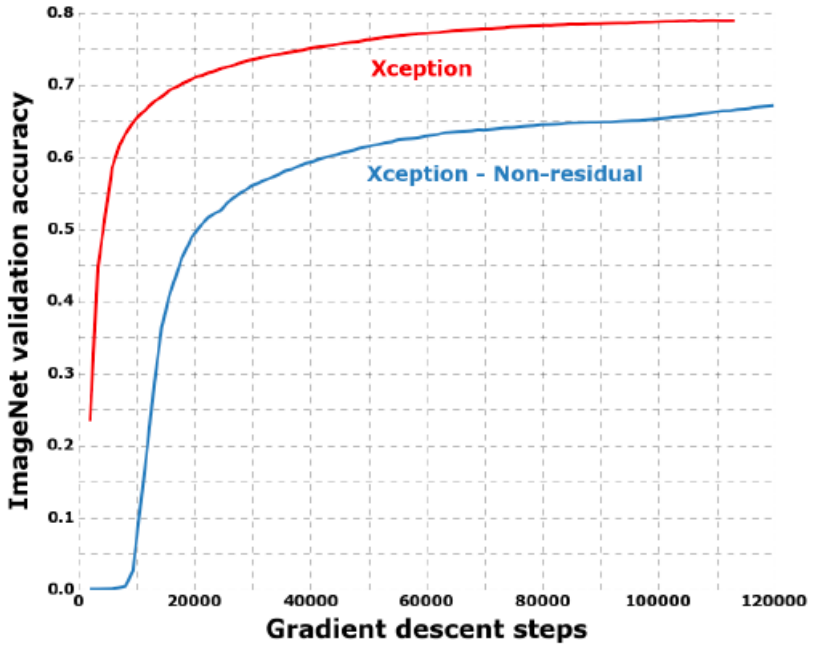
Xception was proposed by none other than François Chollet himself, the creator and chief maintainer of the Keras library.

Xception is an extension of the Inception architecture which replaces the standard Inception modules with depthwise separable convolutions.



As in the figure above, SeparableConv is the modified depthwise sep-

arable convolution. We can see that SeparableConvs are treated as Inception Modules and placed throughout the whole deep learning architecture.



ImageNet: Validation Accuracy Against Gradient Descent Steps

As seen in the architecture, there are residual connections. From the above figure, we can see that the accuracy is much higher when using residual connections. Thus, the residual connection is extremely important.

3.2 NAOqi

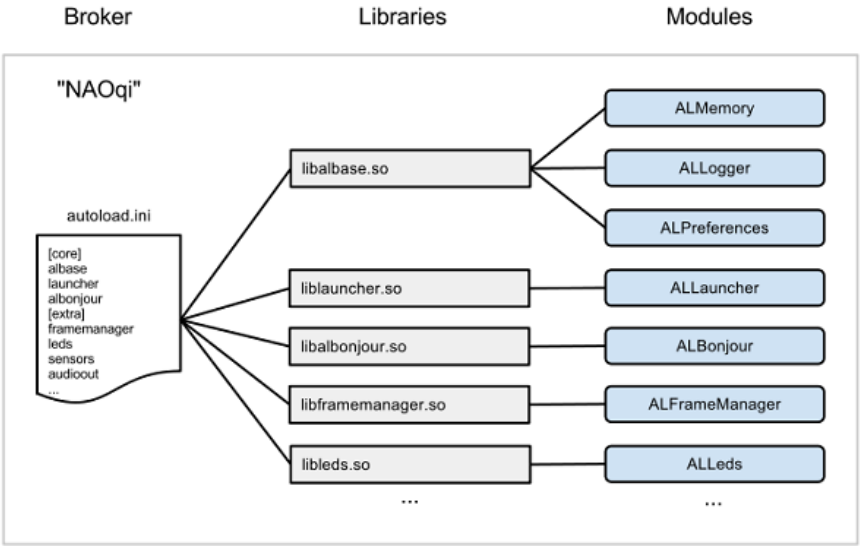
NaOqi is the name of the main software that runs on the robot and controls it.

The NaOqi Framework is the programming framework used to pro-

gram NAO.

It answers to common robotics needs including: parallelism, resources, synchronization, events.

This framework allows homogeneous communication between different modules (motion, audio, video), homogeneous programming and homogeneous information sharing.[4]



Broker
A broker is an object that provides two main roles:

1. It provides directory services: Allowing you to find modules and methods.
2. It provides network access: Allowing the methods of attached modules to be called from outside the process.

Most of the time, you don't need to think about brokers. They do their work transparently, allowing you to write code that will be the same for calls to "local modules" (in the same process) or "remote

modules" (in another process or on another machine).

Proxy

A proxy is an object that will behave as the module it represents.

For instance, if you create a proxy to the ALMotion module, you will get an object containing all the ALMotion methods.

To create a proxy to a module, (and thus calling the methods of a module) you have two choices:

Simply use the name of the module. In this case, the code you are running and the module to which you want to connect to must be in the same broker. This is called a local call. Use the name of the module, and the IP and port of a broker. In this case, the module must be in the corresponding broker.

Modules

Typically each Module is a class within a library. When the library is loaded from the autoload.ini, it will automatically instantiate the module class.

In the constructor of a class that derives from ALModule, you can "bind" methods. This advertises their names and method signatures to the broker so that they become available to others.

A module can be either remote or local.

1. If it is remote, it is compiled as an executable file, and can be run outside the robot. Remote modules are easier to use and can be debugged easily from the outside, but are less efficient in terms of speed and memory usage.
2. If it is local, it is compiled as a library, and can only be used on the robot. However, they are more efficient than a remote module.

Each module contains various methods. Among them, some methods are bound, which means they can be called from outside the module, for example inside another module, from an executable etc. The way to call these bound functions does not vary if the module is remote or local: the module automatically adapts.

4 Problem Statement

4.1 Statement

Use real time images from the camera of a simulated NaO Robot(Webcam) and feed them into a CNN that detects emotions using facial expressions. Make the NaO robot react based on the output of the CNN.

4.2 Constraints

- The image must be of a dimension that is graspable by the webcam.
- The environment of the image must be uncluttered, i.e. only face of a single person should be present in the image.
- The background of the image must be a solid background.

4.3 Input

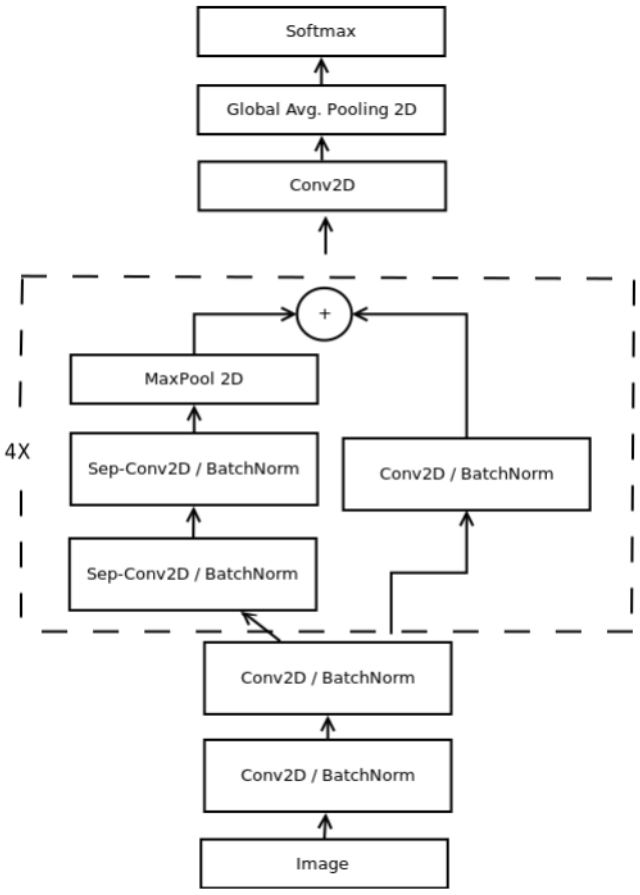
Each frame captured via the webcam(representing the camera of the simulated NaO Robot) is fed into the CNN.

4.4 Output

Based on the emotion detected by the used CNN, the NaO Robot shows a reaction. The reactions are different for different emotions detected.

5 Methodology

5.1 Training the CNN

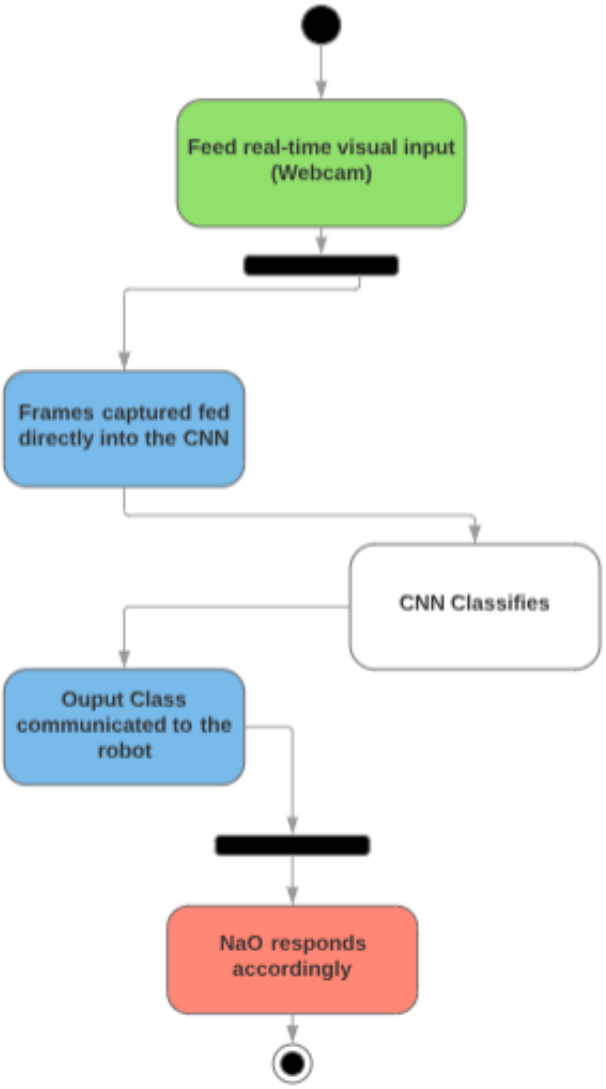


The above figure reflects the model used for emotion detection. The initial image is preprocessed into a 64 x 64 matrix. The matrix is then normalized. Note that Global Average Pooling has been used before going to the softmax function, instead of using a fully connected network. The idea is to generate one feature map for each corresponding category of the classification task in the last mlpconv layer. Instead of adding fully connected layers on top of the feature maps, we take the

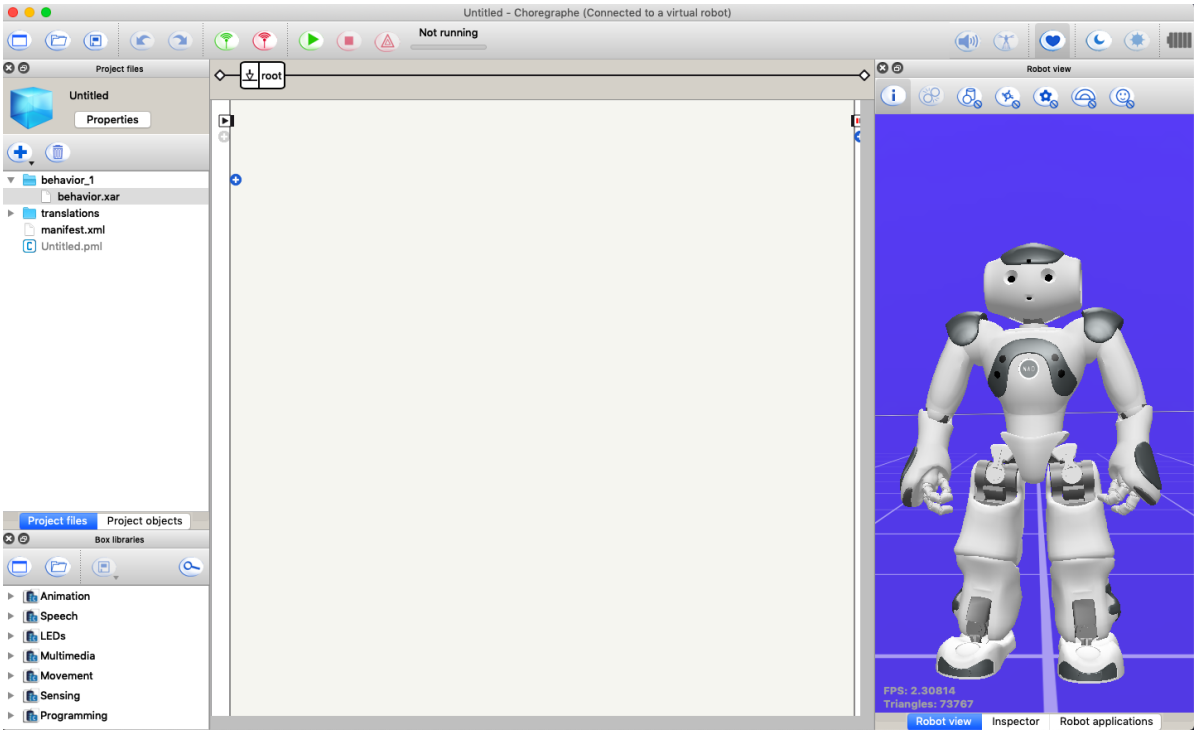
average of each feature map, and the resulting vector is fed directly into the softmax layer. One advantage of global average pooling over the fully connected layers is that it is more native to the convolution structure by enforcing correspondences between feature maps and categories. Thus the feature maps can be easily interpreted as categories confidence maps. Another advantage is that there is no parameter to optimize in the global average pooling thus overfitting is avoided at this layer. Futhermore, global average pooling sums out the spatial information, thus it is more robust to spatial translations of the input.[7]
In the last convolutional layer we have the same number of feature maps as number of classes, and applying a softmax activation function to give the us as output the emotion being depicted in the caputerd image that has maximum probability as detected by the CNN.
Overall, the neural network composed of 9 convolution layers, ReLUs, Batch Normalization and Global Average Pooling.The final architecture is a fully-convolutional neural network that contains 4 residual depth-wise separable convolutions where each convolution is followed by a batch normalization operation and a ReLU activation function. The last layer applies a global average pooling and a soft-max activation function to produce a prediction. Note: For integrating the CNN with the NAO robot, we have used the pretrained model of this CNN, made by Octavio Arriaga, etc.[6]

5.2 Integrating CNN using NAOqi

First step to integrating our CNN model into our NaO simulation is to install the python-sdk for the NAOqi API. Using the functions defined in the NAoqi API we are to communicate data to and forth our NaO simulation. The flowchart below shall illustrate the actual flow which the input image follows on its course of being classified by our CNN.



The input to the NaO robot is simulated by the input from our webcam, frames are captured and fed directly into our network, here the webcam acts as the 'eye' for our simulated NaO robot.
The simulation is rendered using **Choreographe**. Image below displays the interface of the simulating software with a NaO robot rendered on the right.



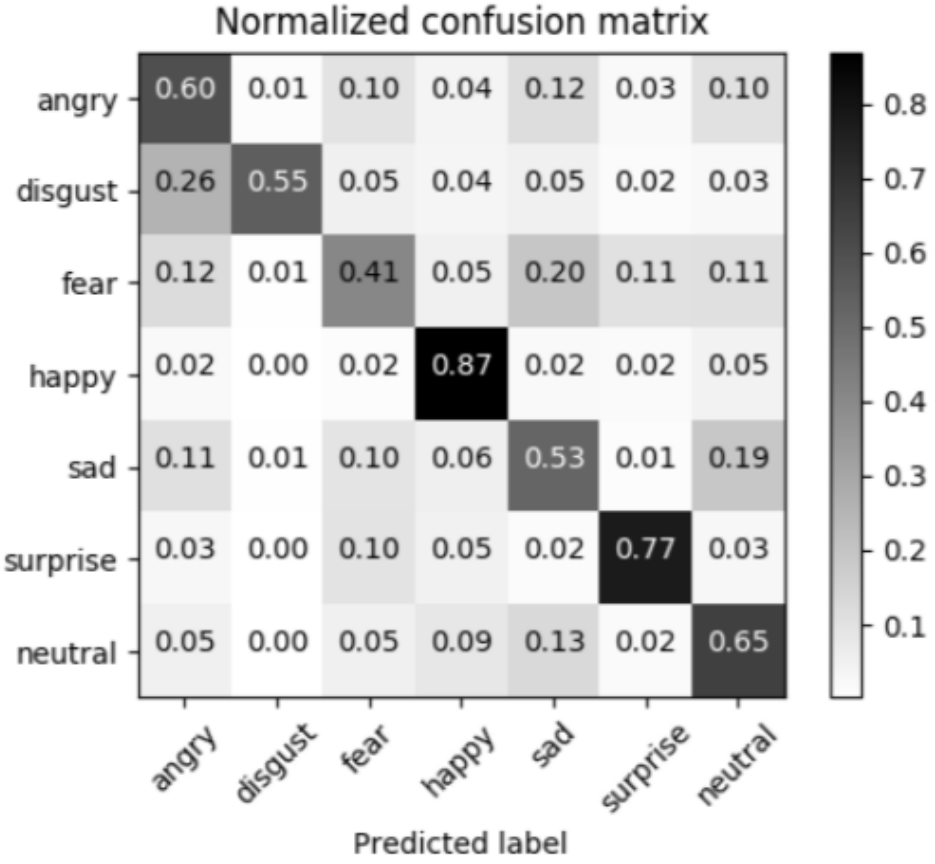
The suite has a **video Monitor** to display what the camera of the robot is capturing at the moment. Since, our robot is a simulated one, it'll not display anything thus to simulate the eye of the robot we use our webcam. Therefore, to achieve this we use OpenCV which can capture frames from the system's webcam and store it in frames which can then be pre-processed.

The CNN classifies the images and produces an output, accordingly we produce a reaction from Nao using the **Text-to-speech** function

of the NAOqi API. This produces a dialog box on top of the robot displaying the reaction to the user's expression.

5.3 Testing the network

In order to test the network, we take images from the webcam and feed the frames directly into the CNN. Before the frames could be fed into the network we need a little pre-processing to be done on the images. The dataset contains 48x48 sized images of 7 different classes which are predicted and on the span of 10 observations the mode is taken which is then decided to be the output of the network.



6 Result and Discussion

6.1 Dataset

The fer2013 dataset was used for training the CNN. fer2013 is an open-source dataset which is first, created for an ongoing project by Pierre-Luc Carrier and Aaron Courville, then shared publicly for a Kaggle competition, shortly before ICML 2013. This dataset consists of 35.887 grayscale, 48x48 sized face images with various emotions -7 emotions, all labeled-.[8]

0: -4593 images- Angry
1: -547 images- Disgust
2: -5121 images- Fear
3: -8989 images- Happy
4: -6077 images- Sad
5: -4002 images- Surprise
6: -6198 images- Neutral

Due to the small number of images under the "disgust" label, for the purposes of this project we have clubbed the reactions of our simulated NaO Robot for the "anger" and "disgust" emotions. 28,709 images were used as trainign set and the remaining were used for testing purposes.

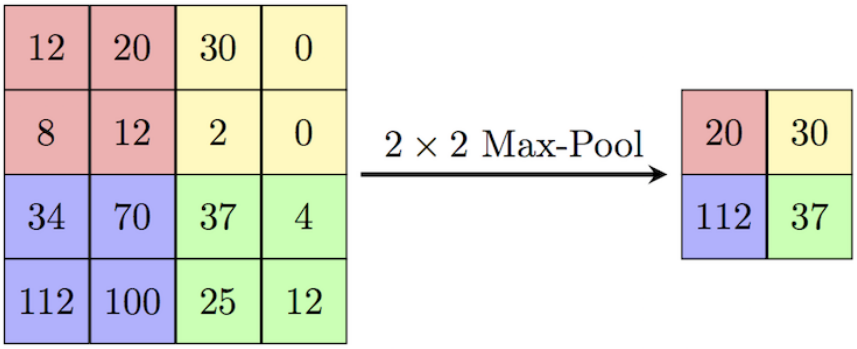
6.2 Features and Accuracy

Our final architecture is a fully-convolutional neural net- work that contains 4 residual depth-wise separable convolu- tions where each convolution is followed by a batch nor- malization operation and a ReLU activation function. The last layer applies a global average

pooling and a soft-max activation function to produce a prediction. On testing the network on the FER-2013 dataset, we obtain an accuracy of **66%** which is quite comparable to actual human accuracy which is about **65% ± 5%**.

6.3 Max Pooling

Max pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned.[9]



Hence, maxpooling has been used in the CNN. One of the advantages of pooling is that it reduces overfitting, and adds translation invariance to the system.

6.4 Sep-Conv2D

Instead of the original Conv2D operation we replace it with a much more optimized algorithm - sep-conv2D which uses depth-wise separable convolutions. Depth-wise separable convolutions are composed of two different layers: depth-wise convolutions and point- wise convolutions. The main purpose of these layers is to separate the spatial cross-correlations from the channel cross- correlations. They do this by first applying a D × D filter on every M input channels and then applying N 1 × 1 × M convolution filters to combine the M input channels into N output channels. Applying 1 × 1 × M convolutions combines each value in the feature map without considering their spatial relation within the channel. Depth-wise separable convolutions reduces the computation with respect to the standard convolutions by a factor of $\frac{1}{N} + \frac{1}{D^2}$. A visualization of the difference between a normal Convolution layer and a depth-wise separable convolution can be observed in Fig. 4. [6]

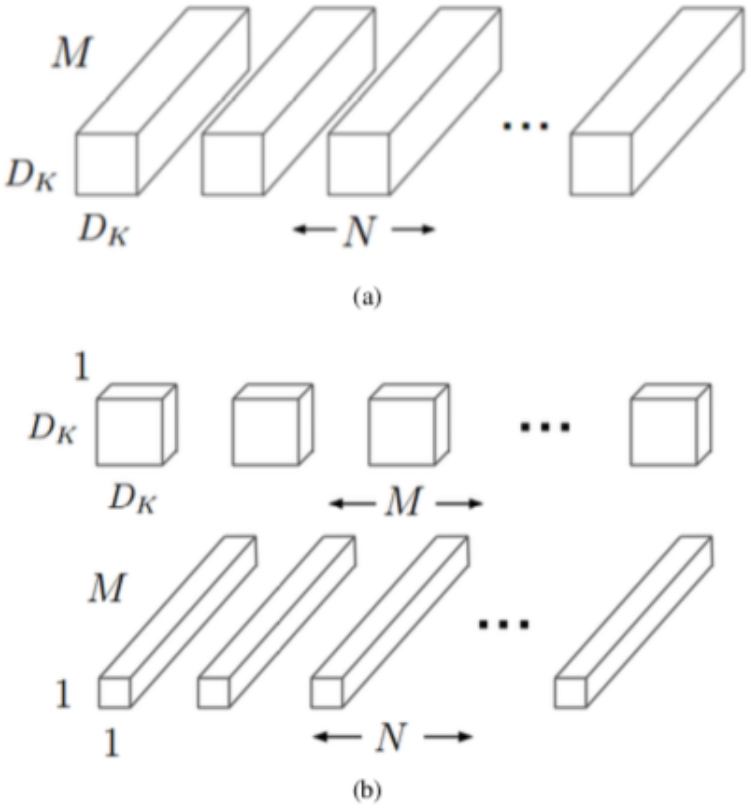
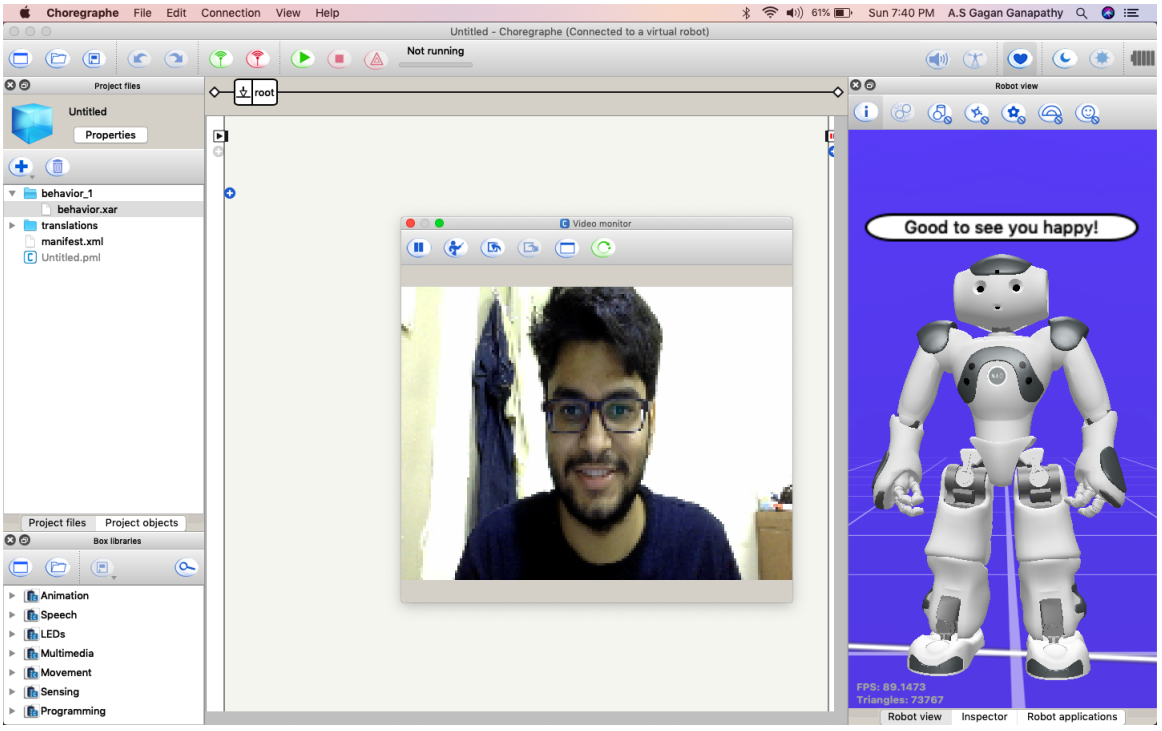
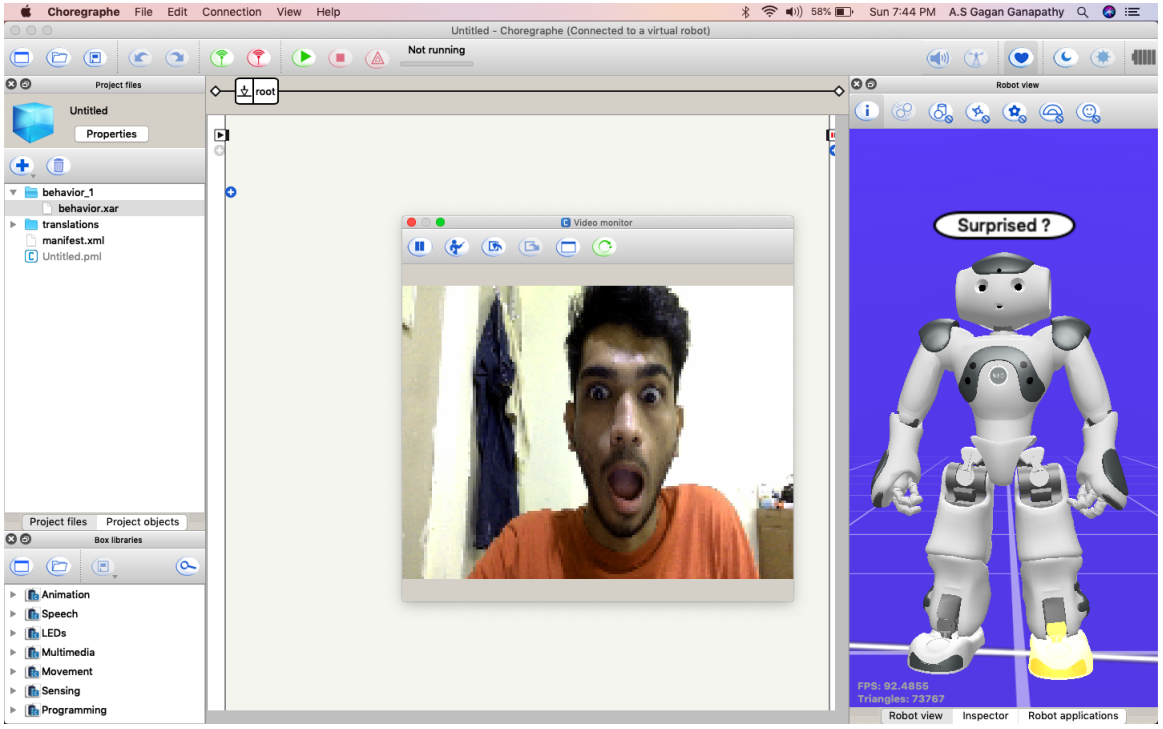


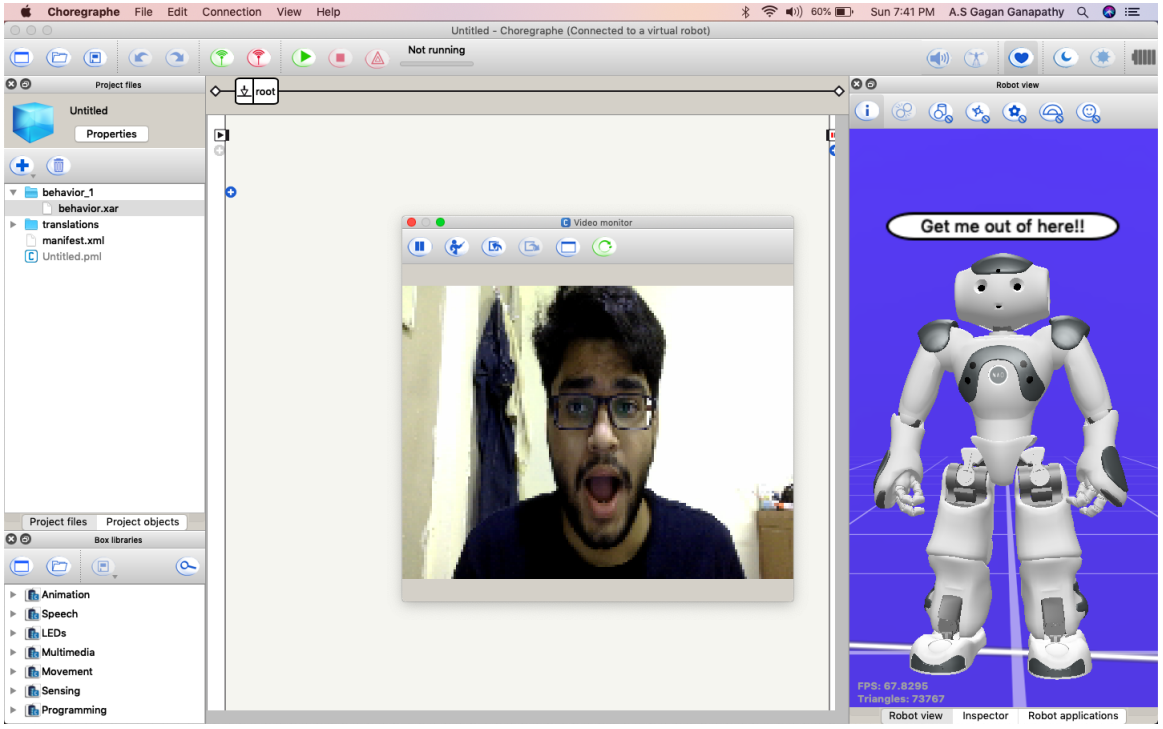
Fig. 4: [2] Difference between (a) standard convolutions and (b) depth-wise separable convolutions.

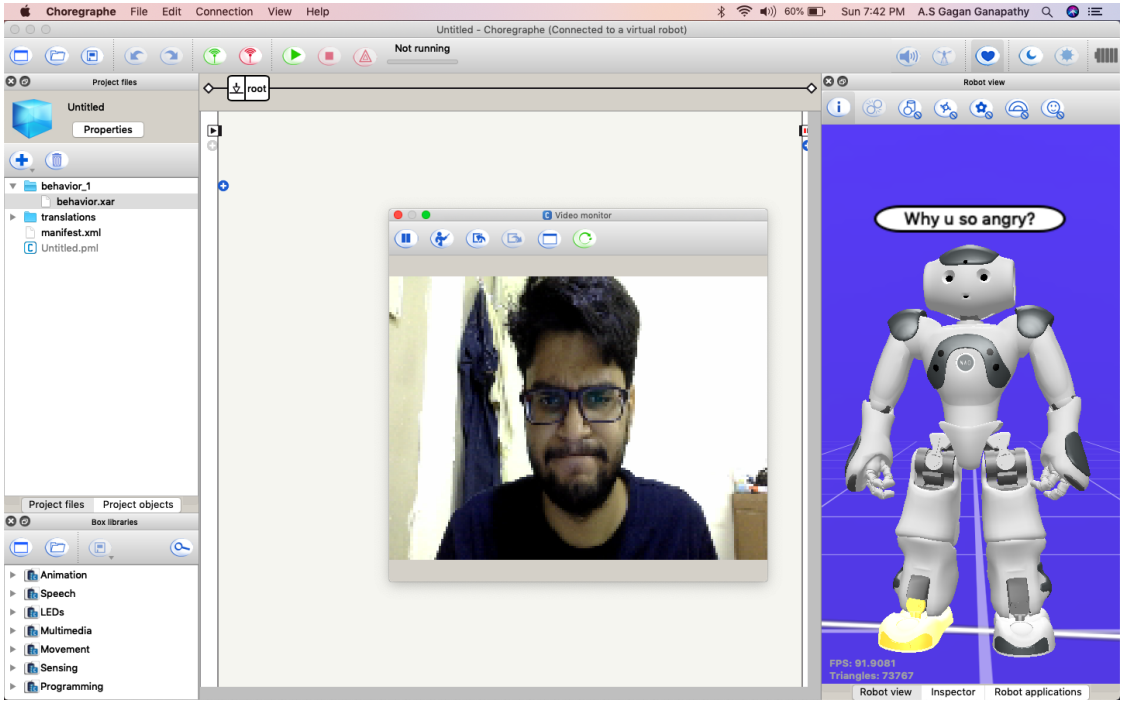
6.5 Results

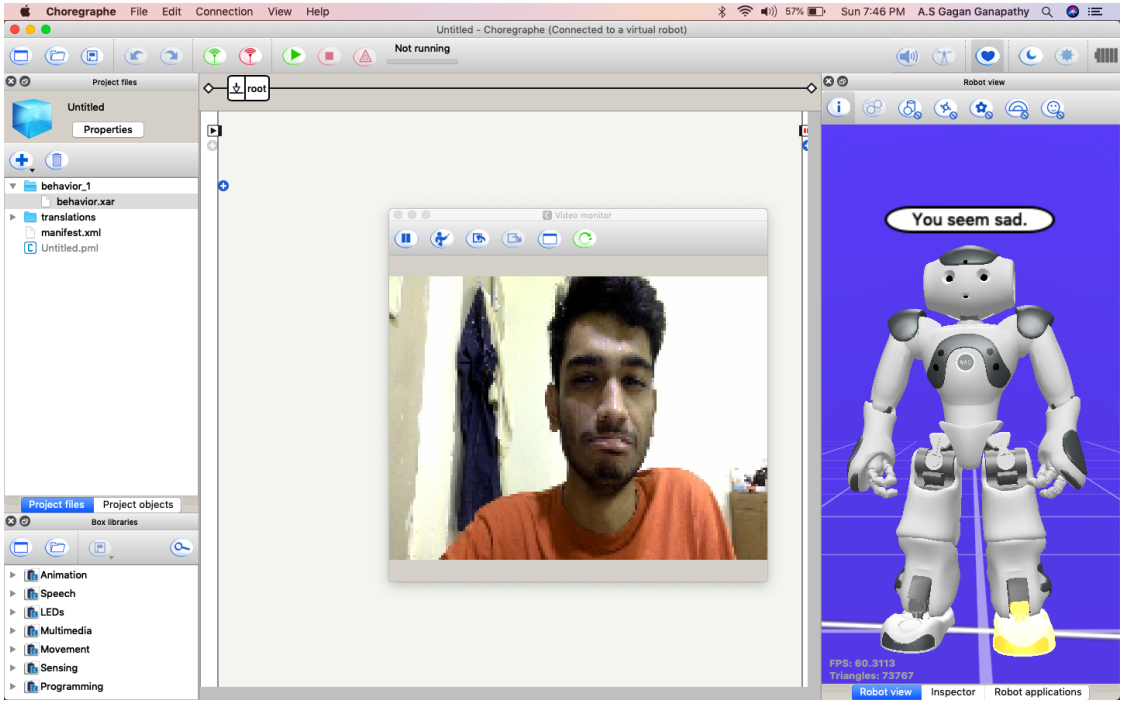
Following are the a few screenshots of the emotions successfully classified by our CNN and shows the corresponsding output from the simulated robot.




Choregraphe File Edit Connection View Help
Sun 7:44 PM A.S Gagan Ganapathy
Untitled - Choregraphe (Connected to a virtual robot)
Not running
Project files
Untitled
Properties
behavior_1
behavior.xar
translations
manifest.xml
Untitled.pml
Project files Project objects
Box libraries
Animation
Speech
LEDs
Multimedia
Movement
Sensing
Programming
root
Video monitor
Robot view
Surprised ?
FPS: 92.4855
Triangles: 73767
Robot view Inspector Robot applications


Choregraphe File Edit Connection View Help
Sun 7:41 PM A.S Gagan Ganapathy
Untitled - Choregraphe (Connected to a virtual robot)
Not running
Project files
Untitled
Properties
behavior_1
behavior.xar
translations
manifest.xml
Untitled.pml
Project files Project objects
Box libraries
Animation
Speech
LEDs
Multimedia
Movement
Sensing
Programming
root
Video monitor
Robot view
Get me out of here!!
FPS: 67.8295
Triangles: 73767
Robot view Inspector Robot applications


Choregraphe File Edit Connection View Help
Sun 7:42 PM A.S Gagan Ganapathy
Untitled - Choregraphe (Connected to a virtual robot)
Not running
Project files
Untitled
Properties
behavior_1
behavior.xar
translations
manifest.xml
Untitled.pml
Project files Project objects
Box libraries
Animation
Speech
LEDs
Multimedia
Movement
Sensing
Programming
root
Video monitor
Robot view
Why u so angry?
FPS: 91.9081
Triangles: 73767
Robot view Inspector Robot applications


Choregraphe File Edit Connection View Help
Sun 7:46 PM A.S Gagan Ganapathy
Untitled - Choregraphe (Connected to a virtual robot)
Not running
Project files
Untitled
Properties
behavior_1
behavior.xar
translations
manifest.xml
Untitled.pml
Project files Project objects
Box libraries
Animation
Speech
LEDs
Multimedia
Movement
Sensing
Programming
root
Video monitor
Robot view
You seem sad.
FPS: 60.3113
Triangles: 73767
Robot view Inspector Robot applications

7 Conclusion and Future Work

The CNN was successfully integrated into the simulated NaO robot after which it was able to classify facial expression into one of 7 different classes of emotions. We have achieved human-level performance in our classifications tasks using a single CNN.

We can observe several common mis-classifications such as predicting "sad" instead of "fear" and predicting "angry" instead "disgust". We can observe that the CNN learned to get activated by considering features such as the frown, the teeth, the eyebrows and the widening of one's eyes, and that each feature remains constant within the same class. These results reassure that the CNN learned to interpret understand- able human-like features, that provide generalizable elements. These interpretable results have helped us understand several common misclassification such as persons with glasses being classified as "angry". This happens since the label "angry" is highly activated when it believes a person is frowning and frowning features get confused with darker glass frames.

Machine learning models are biased in accordance to their training data. In our specific application we have empirically found that our trained CNNs for emotion classification are biased towards western facial features and facial accessories. We hypothesize that this misclassfications occurs since our training dataset consist of mostly western: actors, writers and cinematographers. Such shortcomings could be rectified so that the CNN is able to classify better and thus also allow the robot to classify with greater accuracy.

8 References

[1] https://ifr.org/ifr-press-releases/news/robots-double-worldwide-by-2020

[2] https://medium.com/@chrisprinzz/facial-emotion-detection-using-deep-learning-44dbce28349c

[3] https://en.wikipedia.org/wiki/Nao(robot)

[4] http://doc.aldebaran.com/1-14/dev/naoqi/index.html

[5] https://towardsdatascience.com/review-xception-with-depthwise-separable-convolution-better-than-inception-v3-image-dc967dd42568

[6] https://github.com/oarriaga/face$_c lassification$

[7] https://arxiv.org/pdf/1312.4400.pdf

[8] https://medium.com/deep-learning-turkey/deep-learning-lab-episode-3-fer2013-c38f2e052280

[9] https://computersciencewiki.org/index.php/Max-pooling/Pooling