Multitask with Gestures, not Controls.

Prithvi T Bachireddy
Department of Electrical and Computer Engineering
University of Michigan
Dearborn, USA
prithvit@umich.edu

Veena Arumugam
Department of Electrical and Computer Engineering
University of Michigan
Dearborn, USA
veenaa@umich.edu

Yeswanth Kumar Chilamkurthy
Department of Electrical and Computer Engineering
University of Michigan
Dearborn, USA
ykchilam@umich.edu

***Abstract*— Many of the incidents arise due to driver distractions, and sometimes drivers put their lives at risk due to lack of timely medical attention when traveling in isolated areas where there is no one around to respond. At the same time, drivers are not able to communicate or call for support. In this paper, a system is proposed based on computer vision and Machine learning approach using Convolutional Neural Networks on the Raspberry Pi Platform which is a real-time embedded system (firm) that performs synchronous events with minimal driver distraction and sends a message with GPS Coordinates and pre-recorded voice call when an emergency occurs that may save a life with a minimal investment. The CNN classifier is built by arranging the convolution layers with 3*3 filters along with max pooling. The combined two-layer block is repeated by increasing the depth of filters in each block like 16, 32, and 64. Open source datasets are not readily available that meet the project requirements and so the system is trained with 2000 images per gesture that are created by group members and manually downloaded from the internet sources. The final accuracy of the system is 89.34%.**

Keywords—embedded system, raspberry pi, computer vision, the convolutional neural network machine learning, recognition, gestures, alert, message, max pooling, GPS Coordinates, filters, datasets

I. INTRODUCTION

Why do most of the car accidents happen? Most accidents happen due to what experts call it "driver error". Driver errors can have various categories but the most common one is "Distraction".[18] In other words, accidents happen because of multitasking while driving. The most common causes of distraction are daydreaming, texting, or talking to someone, setting locations, adjusting audio or climate controls in the car. Designing a system that performs actions with minimal driver distraction and sending a message when an emergency occurs, may save a life for a minimal investment.

Safety is very important in everyone's life. These days technology is moving very quickly from buttons to touch screens and from touch screens to hand gestures and facial recognition. These are not only fun but also helps the driver from distraction. The proposed system monitors a driver in a car, performs actions based on body behavior and gestures of the driver to set adjusting controls and sends an alert in any kind of emergency. The system is a combination of an Embedded System, Computer Vision, and Machine Learning application. The system is trained with body behaviors and hand gestures where it can perform actions for each behavior and gesture.

II. LITERATURE SURVEY

Deep learning is used to recognize the facial expression and image classification method is used to classify the recognized expression that is discussed in [1]. All the facial expressions/behaviors are labeled automatically using the inception model and generated the spectrogram of the image which is used as an input for training the system. A weighted mixture deep neural network method is proposed in [2] to automatically extract the hand-crafted features that are effective for facial expression recognition tasks.

In [3], a convolution neural network is implemented with two convolution layers and two subsampling layers. The first layer has 6 masks and the second layer has 12 masks to extract the features. An end to end framework is developed in [4] using deep learning models based on attentional convolutional networks for recognizing and classifying the facial expressions. The network has less than ten layers where four layers are used for feature extraction where every two layers followed by a max-pooling layer to automatically reduce the resolution of the image and rectified linear unit activation function.

The authors have presented a framework in [6], for representing and recognizing hand postures used in sign language. An eigenspace size function and Hu moments are used for classifying hand postures. The data sets used for this hand postures are Jochen-Triesch static hand posture and ISL data set. The two main architectures VGG-16 and ResNet50 were used in identifying seven different emotions in paper [9]: anger, disgust, fear, happiness, sadness, surprise, and neutrality. Through this paper [10], they have proposed four architectures that are BKVGG8, BKVGG10, BKVGG12, BKVGG14. All the architectures follow the same principles of VGG (Visual Geometry Group) and have the same input image size of 42x42x1. This is an automatic process that runs without designing by hand.

A new method was proposed in [13] that is called deep comprehensive multipatches aggregation convolutional neural networks. This method contains two branches of the CNNs. One of which extracts expressional details from image patches and the second branch extracts high-level semantic information. Also, in this paper, a new pooling strategy called transformational- invariant pooling is introduced for dealing with different variations such as noises, rotations, etc., An article was written on hand gesture recognition that is based on the NN shape fitting technique. The image goes through four stages of techniques before recognizing the gesture. Firstly, the color segmentation technique is applied to detect

the region of the skin and then Self- Growing and Self-Organized Neural Gas Network (SGONG) is applied to identify the shape. Later in this process, palm morphologic characteristics are extracted based on the neurons' output grid. Finally, gesture recognition is achieved successfully by applying a likelihood-based classification technique.

In [20] the system is proposed to detect gender based on computer vision and machine learning using convolutional neural networks on the Raspberry Pi Platform. CNN is used to extract the various facial features and the best features are selected to train and test the dataset.

The proposed system in [21] implements an intruder detection system using convolution Neural Networks, Raspberry Pi, Twilio Cloud Interface, and Microsoft's Azure. The algorithm that is used in CNN classifies the input data as either intruder or user. The Raspberry Pi is used to capture the input image and acts as a middleware. The algorithm will be stored in the cloud. The cloud system is also programmed to send an MMS alert using the Twilio cloud interface when users or intruders are detected.

The authors in [22] have implemented an embedded system application that is feasible and efficient. The CNN algorithm is interfaced with Raspberry Pi. This algorithm classifies and quantifies any frame dissimilarities that represents health and damaged structural conditions.

In [23], an automated weed removal Bot is designed using Raspberry Pi 3 and CNN. The proposed approach focuses on exploiting the information by analyzing the sequence of the images to improve the performance of the classification. Like the above papers, the input image is captured through the Pi camera. This method also extracts the spatial arrangement of the plantation to detect crops and weeds individually.

III. SOFTWARE ENGINEERING

A. *Requirements*

1. System Requirements

- [SYSFUN001] The system consists of an Embedded system, Machine Learning Application, and Hardware that has the capability to continuously capture the behavior of the driver, identify five different gestures of the driver behavior and perform certain actions accordingly such as send an alert to the subscriber in case of emergency or control the interfaces of the car.
- [SYSFUN002] The system turns ON the DC motor when the driver's gesture is the Victory symbol.
- [SYSFUN003] The system turns OFF the DC motor when the driver's gesture is the Palm symbol.
- [SYSFUN004] The system sends a Text with GPS Coordinates and a pre-recorded automated Call during an emergency like a heart attack.
- [SYSFUN005] The system makes the alarm to beep (through speakers) when the driver is drowsy.
- [SYSFUN006] The system has a user-configurable delay to perform the actions and alert the subscriber.
- [SYSFUN007] The system does not perform any action or send any alert in the condition of "No Gesture".
- [SYSPER001] The system sends text alerts within 20sec of occurrence of an emergency event.
- [SYSPER002] The system sends a call alert within 30sec of occurrence of an emergency event.
- [SYSPER003] The system makes an alarm action within 5 sec of the gesture identified.
- [SYSPER004] The system controls the DC motor within 5sec of the gesture identified.

2. Embedded System Requirements

- [ESFUN001] The hardware of the embedded system consists of a Pi-camera that continuously captures the behavior of the driver, the speaker for an emergency alarm, L293D IC to control DC Motor, mouse, and keyboard to operate the system.
- [ESFUN002] The software of the embedded system consists of Miniconda and other important libraries like Tensorflow, Opencv, Keras, etc. to program in a high-level language such as python to build Machine Learning Application.
- [ESFUN003] The software of the embedded system consists of a function to extract frames from the capturing video and load it to the Machine Learning Application.
- [ESFUN004] The software of the embedded system consists of a Wi-Fi module and Twilio Interface that sends a text with GPS-Coordinates and an emergency call to the Twilio Cloud.
- [ESPER001] The Embedded system extracts 1 frame per second.
- [ESPER002] The Embedded System loads the image to the Machine Learning Application within 2 sec after extracting the frame from the video.
- [ESPER003] The Embedded System transmits a text alert within 10 sec and calls alert within 15 sec to the Twilio cloud after receiving input from the Machine Learning Application.

3. Machine Learning Requirements

- [MLFUN001] The Machine Learning System consists of a Trained CNN classifier model that predicts five different gestures of driver behavior.
- [MLPERF001] The Machine Learning System should predict the gestures and behavior of the driver within 5 seconds.

4. Twilio Cloud Requirements

- [TWFUN001] The Twilio cloud consists of a free account registered with the subscriber number.
- [TLPERF001] The Twilio Cloud sends an alert to the subscriber within 5sec after receiving an alert from the embedded system.

B. *Software Development Methodology*

Agile Software Development Methodology is followed to build the system because there were no defined system specifications at the beginning, due to time constraints rapid development is needed, and regular adaptation to changing circumstances. Even late changes in requirements are possible, Specification, development, testing, and validation are interleaved rather than separate, with rapid feedback across activities.

C. Design Methodology

The project was divided into three phases, each comprising several weeks. They are requirements phase, design and implementation phase, and system integration and testing. The requirements phase was focused on identifying the system and subsystem requirements. This process involves identifying the functional and performance needs of the system as well as the amount of available time to complete the integration of the system. Further, creating requirements for the system such a way that it is reasonable to accomplish and together create a functional system. The design and implementation phase involves the system-level design, including block diagrams and flowcharts, as well as selecting the hardware that would be used for the project. The final phase involves combining the two parts of the systems and testing them for functionality and requirement validation. Short weekly meetings were used to collaborate between team members to make major design decisions as well as to make any necessary adjustments to the design.

1. Requirements Phase

The initial phase of the design revolved around defining the system that would be developed, including requirements definition (see Requirements) for the functionality and performance of the system.

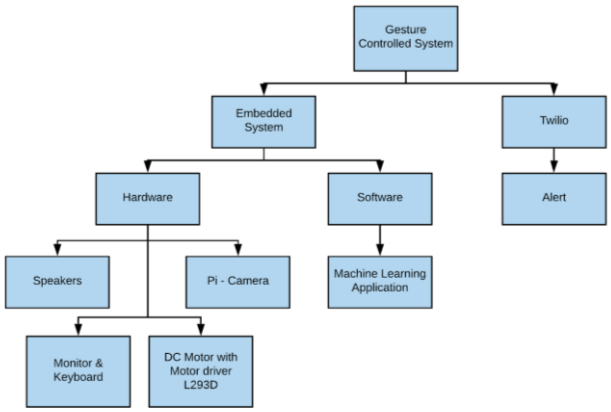

Figure 1: System Block Diagram

The system was divided into two main parts: the embedded system and Twilio. The embedded system is categorized into 2 parts: 1) External peripherals such as camera sensors, DC motor, and speakers, 2) Software i.e. Machine Learning Application. The Twilio is used to send the text alert with Latitude, Longitude coordinates and call alert with a pre-recorded voice message for the subscribers when the driver has a health emergency. The system was laid out using a block diagram to define the parts of the hardware and the major software modules of the system (see Figure 1). The main components of the system are:

- Pi Camera

Raspberry Pi Camera V2 has an 8-megapixel high-quality Sony IMX219 image sensor. The pi camera is used to read the real-time behavior of the driver which is placed in the vehicle. The video is converted to frames and each frame is stored. The pi camera is connected to the Raspberry Pi Board.[23]

- Raspberry Pi 3 B+

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV and uses a standard keyboard and mouse. The Raspberry Pi 3 B+ has significantly huge processing power in a compact board(portable). It has many interfaces like HDMI, multiple USB, Ethernet, onboard Wi-Fi, Bluetooth, USB powered and many GPIOs which is a very important interface to complete the required task.[24]

- Twilio Cloud

Twilio is a developer software platform for communications. The Twilio is used to send the text alert with latitude, longitude coordinates and call alert with a pre-recorded voice message for the subscribers when the driver has health emergencies and is not able to perform any immediate contact.[25]

- Speaker

Speaker is used to making an alarm to alert the driver when the driver is feeling drowsy or sleepy. It is connected to the Raspberry Pi Board.

- L293D

An L293D is a Dual H-bridge Motor Driver integrated circuit works as an amplifier by converting the low current control signal to a high current control signal that makes the DC motor run.

- DC Motor

DC Motor is used to show applications like AC control or any other in the vehicle when the driver shows some gestures.

2. Design, Implementation, and Subsystem Testing

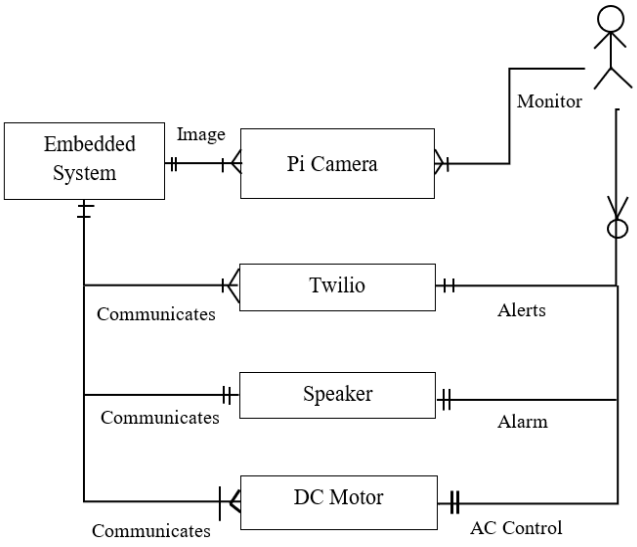

Figure 2: ERD for the System.

The second phase began with designing the Entity-Relationship Diagram (ERD) for the system. This defines the relationship between the various parts of the system. Figure

2 shows the ERD for the system. The embedded system (Raspberry PI) monitors the Pi Camera which continuously monitors the driver for behavior and gestures. Depending on the gesture behavior, various subsystems are implemented.

Following the ERD diagram completion, the logical flow of the code was designed using data flow diagrams (Figure 3). Using the flow charts in Section C, the code was written for the separate parts of the system. The hardware and parts of the code were then tested separately (see Section VI) for functionality using the test plans detailed Section VI.A.

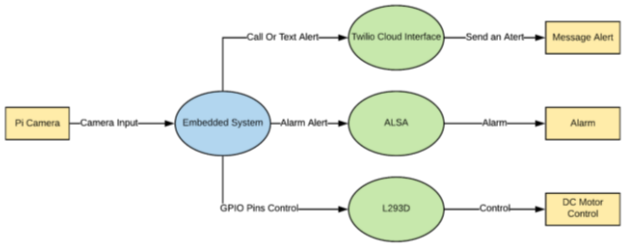

Figure 3: Level 1 Data Flow Diagram

3. Integration and System Testing

The final phase of the project was focused on integrating the various parts of the system and testing them together (system testing). Once the parts were integrated, the system was tested using the requirements specified in phase one.

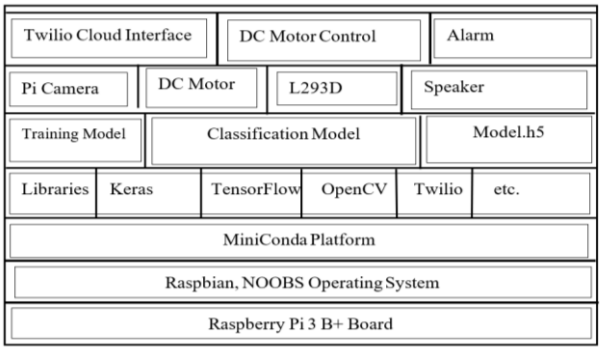

Figure 4: System Architecture

IV. DESIGN

A. Overview

The Pi Camera continuously monitors the driver's behaviors and gestures. The input video is converted to frames. Each frame is compared to the classifier which has 5 different classifications: 1)Heart attack, 2)Drowsiness, 3)Gesture 1, 4)Gesture2, and 5)No Gesture. If any of the 5 categories match, the output signal is generated accordingly.

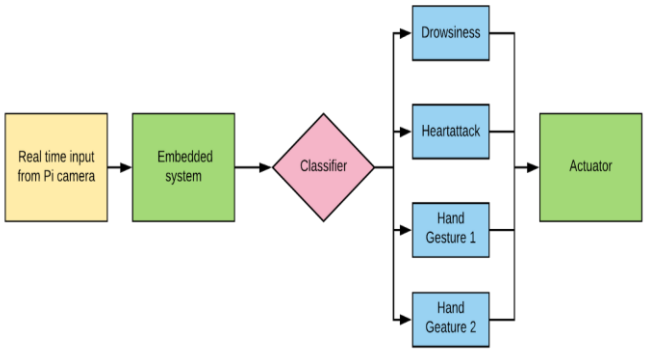

Figure 5: Flowchart of the System

Figure 6: a) Heart Attack

b) Drowsiness

Figure 7: c) Gesture 1

d) Gesture 2

The expected output from the system:

1. Heart Attack Detection: Immediately send an alert message to the subscriber with latitude and longitude of the location using Twilio Cloud Interface (detailed explanation in Section IV. B)
2. Drowsiness Detection: An alarm is activated to alert the driver using ALSA. (detailed explanation in Section IV. B)
3. Gesture1(Victory Gesture): AC controls ON. To indicate this operation, DC Motor control is turned ON using the L293D IC.
4. Gesture2(Palm Gesture): AC controls OFF. To indicate this operation, DC Motor control is turned OFF using the L293D IC.
5. No Gesture: Any gesture other than the above gestures are considered as No Gesture indicating "No Output".

B. Hardware Design

1. Raspberry Pi Module

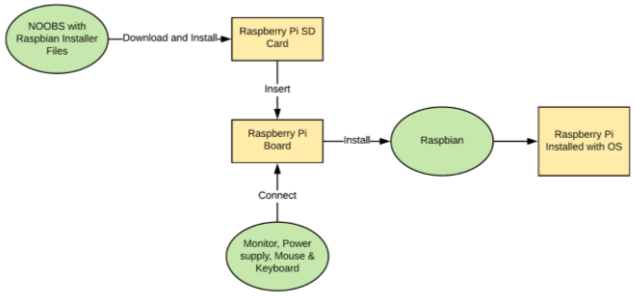

Figure 8: Data Flow Diagram of Raspberry Pi Module

Steps:

1. Insert the SD card into the SD card reader, connect to computer and format the card.
2. Download NOOBS with Raspbian installer files and copy them to the SD card.
3. Insert the SD card to Raspberry Pi board and connect the board to TV or Computer Monitor.
4. Connect Power Supply, Mouse, and Keyboard to Pi.
5. Set the Language and Choose the OS to install.
6. Once the installation is done, click ok and the system will reboot.
7. Select the options to reboot directly on the system desktop.

2. Raspberry Pi Camera Module

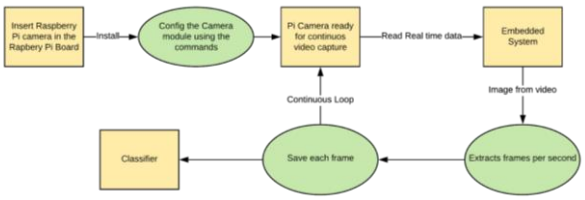

Figure 8: Data Flow Diagram of the Raspberry Pi Camera Module

3. L293D Motor Driver Chip & DC Motor

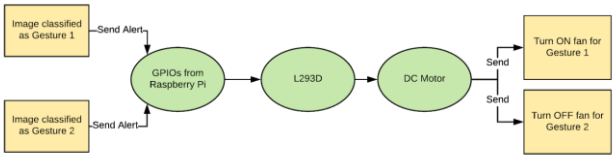

Figure 9: Data Flow Diagram of L293D

A motor driver chip is an integrated circuit that is used to control DC motors. Motor drivers act as an interface between Raspberry Pi and the DC motors. A DC motor is used to show that we can control some applications in a vehicle like increasing and decreasing AC, etc. with some simple gestures without much distraction from driving. Gesture 1 and Gesture 2 are used to control the motor as shown in Figure 9.

Steps:

1. Place the IC on the breadboard, Connect Pin 1 to +5 VCC of Raspberry Pi using Connectors
2. Connect Pin 2 and 7 to GPIOs 13 & 15 of Raspberry Pi
3. Connect Pin 4,5 to Ground of Raspberry Pi
4. Connect Pin 3 to one terminal of DC Motor and Pin 6 to another terminal of DC Motor
5. Connect Pin 8 to another +5 VCC of Raspberry Pi for output power supply.

C. Software Design

1. Data Set Collection

The first step of this design involves data set collection. Dataset is a collection of related sets of information that is composed of separate elements but can be manipulated as a unit by a computer. The photos will be reshaped into a standard size(200*200) before modeling so that all the images have the same size. Around 2000 images are captured and collected for each category, saved in their respective folders with the class name saved inside a folder named Dataset. The data set is divided into the Test Set and Training Set. 25% of the data is taken for testing and 75% of the data is taken for training the CNN Classifier. Each image is named with its respective class for training dataset but not for testing dataset.

2. Training CNN Classifier Models

The second step of this process is training the CNN Neural Network. Six Different CNN classifier filters are trained with the same dataset. The CNN Classifier is built by arranging the convolutional layers with 3*3 filters along with max pooling. Combined, these two layers form a block, now blocks can be repeated by increasing the depth of filters in each block like 32, 64, 128 and 256. To ensure that the Height and Width shapes of the output match the input, Padding is used on the convolutional layers. The Model with high accuracy is selected and made improvements by applying Dropout Regularization and Image Data Augmentation.

3. Testing the Classifier Model

The completion of the training process leads to the testing of the classifier to calculate the accuracy of the model. Each Classifier Model is tested with the Unknown Test Dataset and the accuracy of each classifier model is noted. The model with high accuracy is selected as the final model and used for prediction.

4. CNN Final Model Training & Prediction Model

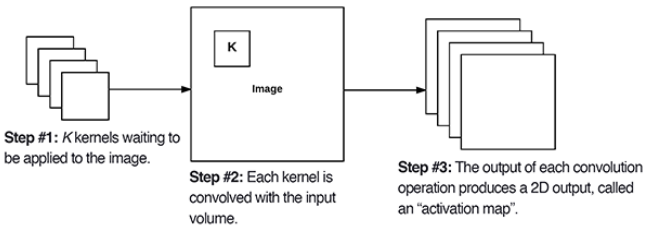

Figure 10: The Keras Conv2D parameter, filters determines the number of kernels to convolve with the input volume. Each of these operations produces a 2D activation map.

The final model in this system uses Keras CNN i.e. 2D Convolution layers and Max Pooling layers. The two layers combine to act as one block. The system is built with five blocks as shown in figure 11 with each block having one conv2D and Max2D layer. [19]

```
def define_model():
    model = tf.keras.models.Sequential([
        # Note the input shape is the desired size of the image 200x 200 wi
        # The first convolution
        tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_shape=(2
        tf.keras.layers.MaxPooling2D(2, 2),
        # The second convolution
        tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
        tf.keras.layers.MaxPooling2D(2,2),
        # The third convolution
        tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
        tf.keras.layers.MaxPooling2D(2,2),
        # The fourth convolution
        tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
        tf.keras.layers.MaxPooling2D(2,2),
        # The fifth convolution
        tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
        tf.keras.layers.MaxPooling2D(2,2),
        # Flatten the results to feed into a dense layer
        tf.keras.layers.Flatten(),
        # 128 neuron in the fully-connected layer
        tf.keras.layers.Dense(128, activation='relu'),
```

Figure 11: Final Classifier

The important hyperparameters required to build Conv2D and Maxpooling2D are:

The first required parameter is the **number of filters** that corresponds to the **depth** of the output volume that each learns to look something different in the input. Here we are using 16, 32 and 64 filters.

The second parameter to specify is strides that slide the filter. When the stride is 1 (default), the filters move one pixel at a time. The other parameter to set is padding to pad the input volume with zero's around the border which is set to zero padding.

The third required parameter to provide the Keras Conv2D layer is the kernel/filter size, the **filter size** is how many neighbors information can be seen when processing the current layer. When the **filter size** is 3*3, that means each neuron can see its left, right, upper, down, upper left, upper right, lower left, lower right, as a total of 8 neighbor information.[15]

3x3

0.91	0.32	0.07
0.73	0.26	0.81
0.53	0.68	0.14

Figure 12: The Keras deep learning Conv2D parameter, filter_size, determine the dimensions of the kernel.

The kernel_size must be an *odd* integer as well. Typical values for kernel_size include: (1, 1) , (3, 3) , (5, 5) , (7, 7) . It's rare to see kernel sizes larger than *7×7*. If the input images are greater than *128×128* it is preferable to choose kernel size > 3 to help learn larger spatial filters and to help reduce volume size.

1. The output volume size of the layer is calculated by
2. Accepts a volume of size W1 x H1 x D1

- Requires four hyperparameters:
  - Number of filters K,
  - Their spatial extent(kernel size) F,
  - the stride S,
  - the amount of zero padding P,
- Produces a volume of size W2 x H2 x D2 where:
  - W2 = (W1 - F+2P)/S + 1
  - H2 = (H1 - F + 2P)/S + 1
  - D2 = K

After Passing the image through the convolutional layer, the image becomes abstracted to a feature map with shape(number of images) x (feature map width) x (feature map height) x(feature map channels)[16]. Max pooling is used to reduce the spatial dimensions of the output volume by combining the outputs of neuron clusters at one layer into a single neuron in the next layer by keeping the max value(activated features) in the sub-regions binned.

The two parameters to set in the pooling layer are pool size and strides. Pool size is given as 2x2 that reduces the output volume size into half and strides as default 1 x 1. In between the convolutional layer and the fully connected layer, there is a '**Flatten**' layer. **Flattening** transforms a two-dimensional matrix of features into a vector that can be fed into a fully connected neural network classifier.[17]

The output of the flatten layer is given to the fully connected layer with 128 neurons. It is a linear operation in which every input is connected to every output by weight (so there is n_inputs * n_outputs weights). Generally followed by a non-linear activation function. Weights: Each neuron in a neural network computes an output value by applying a specific function to the input values coming from the receptive field in the previous layer.[15]

After finalizing the model, to predict the real-time data or image, the final model is trained with a complete data set without dividing it into training and testing datasets. While training the final model, the data weights are stored in the .h5 file which shall be further used for prediction. The function of each layer, the output from each layer is clearly explained in the table below:

Layers	Structure
Input Image	Shape (number of images) x (image width) x (image height) x (image colour depth) 11895 x 200x200 pixels x 3(RGB)
Conv 1	Convolutional Layer: 16 filters of characteristics with a receptive field of 3 x 3 x 3. The Conv layer output volume size [200x200x16]. Each of the 200*200*16 neurons in this volume is connected to a region of size [3x3x3] in the input volume.
Relu 1	ReLU, Activation Function y = max(0, x)
Pool 1	Max Pooling 2D with respective field of 2 x 2, strides 1 x 1and padding 0 x 0 This layer output volume size [100x100x16].
Conv 2	Convolutional Layer: 36 filters of characteristics with a receptive field of 3 x 3 x 3. The Conv layer output volume size [100x100x36].
Relu 2	ReLU
Pool 2	Max Pooling 2D with respective field of 2 x 2, strides 1 x 1and padding 0 x 0 This layer output volume size [50x50x36].
Conv 3	Convolutional Layer: 64 filters of characteristics with a receptive field of 3 x 3 x 3. The Conv layer output volume size [50x50x64].
Relu 3	ReLU
Pool 3	Max Pooling 2D with respective field of 2 x 2, strides 1 x 1and padding 0 x 0 This layer output volume size [25x25x64].
Conv 4	Convolutional Layer: 64 filters of characteristics with a receptive field of 3 x 3 x 3. The Conv layer output volume size [25x25x64].
Relu 4	ReLU
Pool 4	Max Pooling 2D with respective field of 2 x 2, strides 1 x 1and padding 0 x 0 This layer output volume size [14x14x64].
Conv 5	Convolutional Layer: 64 filters of characteristics with a receptive field of 3 x 3 x 3. The Conv layer output volume size [14x14x64].
Relu 5	ReLU
Pool 5	Max Pooling 2D with respective field of 2 x 2, strides 1 x 1and padding 0 x 0 This layer output volume size [7x7x64].
Flatten	Flatten the output of the convolutional layers to create a single long feature vector.
Dense 1	128 neurons, Activation Function = ReLU Each neuron receives input from all the neurons in the previous layer. The layer has a weight matrix W, a bias vector b, and the activations of the previous layer a.
Dense 2	No_of_Classes = 5, Activation Function = SoftMax, Loss = Categorical_Crossentropy, optimizer=RMSprop

Figure 12: The detailed layer by layer architecture of the CNN

5. Miniconda Software

Miniconda is a free minimal installer for conda. It is a small, bootstrap version of Anaconda that includes only conda, Python, dependent packages, and a small number of other useful packages, including pip, zlib, and a few others. Use the conda install command to install 720+ additional conda packages from the Anaconda repository.

Steps:

1. Download the Miniconda files and run to start the installation process.
2. After installation, check the conda installation and python version.
3. Install the Anaconda Client and create a Virtual Environment.
4. Install the required packages in the created environment.

6. Twilio Cloud

Twilio Cloud Interface is used for sending SMS and calling during emergencies. In this system, Twilio Cloud Interface is used when an image is classified as heart attack. As soon as the image is identified as a heart attack, an emergency SMS or automated call is sent to the contact the user wants to contact during an emergency.

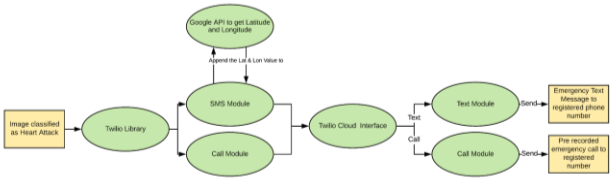

Figure 13: Data Flow Diagram of Twilio Cloud Interface

Steps:

1. Create a Free Account and login into Twilio Cloud.
2. Twilio Account Create an Account Sid and Authentication Token. Save them.
3. Register the Number to which call and text need to be sent.
   Get the Twilio Number from which the alert should be sent.
4. Write a Python code to send the alert with the given Account Sid and Authentication token by giving the registered and Twilio Number.
5. The health emergency message is sent with the current location of the driver.

7. ALSA

Using the Advanced Linux Sound Architecture module of Raspberry Pi, the sample audio file will be routed to external Speaker output. ALSA is used when an image is classified as drowsiness.

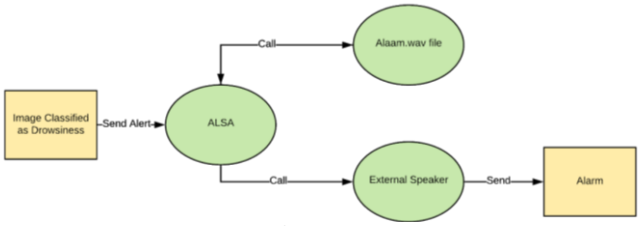

Figure 14: Data Flow Diagram of ALSA

V. IMPLEMENTATION

In this section, the complete system integration procedure is shown.

Step 1: Pre-trained CNN classifier

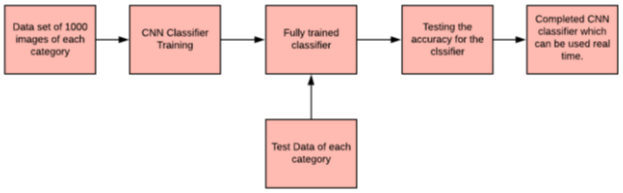

Figure 15: Flowchart of Trained CNN Classifier.

The first step involves collecting data sets and training the final chosen classifier model. This trained classifier is tested to find out the final accuracy that can be reached. The result of the first step is a final trained CNN model ready to be used for real-time data input.

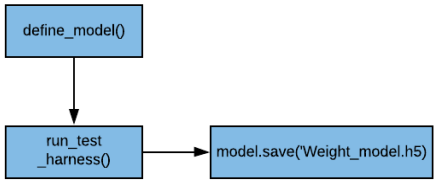

Figure 16: Final trained function call

Major functions:

The code for training the model is organized systematically. The define_model() function performs the Keras Conv2D explained in section IV.C.4. It uses the filters 16, 32, 64, 64, 64, and 128 accordingly with Max Pooling.

The run_test_harness() function tests the above-trained system for accuracy in classifying into 5 different classes as mentioned in section IV.A.

The model.save('Weight_model.h5) function stores the weights of each class that can be used further for classifying real-time images.

Step 2: Hardware Implementation

This step involves connecting all the external peripherals to the Raspberry Pi Board as shown below:

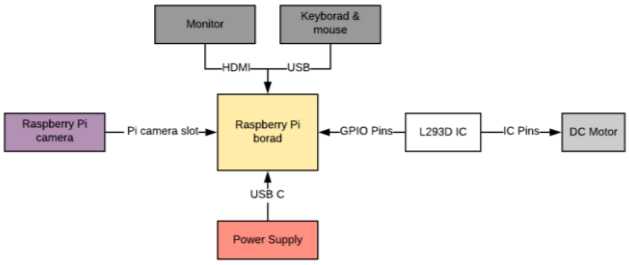

Figure 17: Flowchart of the Hardware Implementation of the system.

Step 3: Raspberry Pi Camera Input Processing

Now it's the time for classification and recognizing the images. To do that, first, a python code is written to capture the video and to extract the frame from the video to save it as an image. This is achieved by capturing the real-time dataset from the Pi camera.

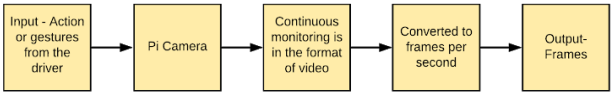

Figure 18: Flowchart of the Raspberry Pi camera module

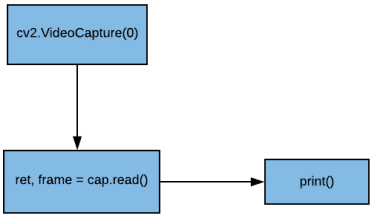

Figure 19: Pi camera module function call

Major functions:

The cv2,VideoCapture(0) function is used for real-time video capture. It continuously monitors the driver for gestures. The ret, frame = cap.read() function allows to convert video into each frame.

Step 4: CNN Classifier

This step involves classifying the input frames into 5 different categories. The output is sent to the embedded system for controlling the actuators.

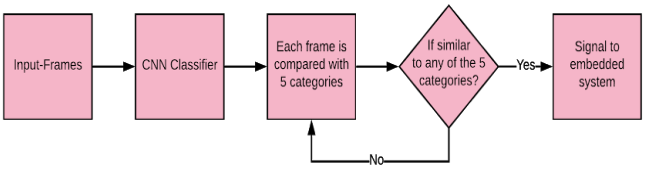

Figure 20: Flowchart of CNN Classifier.

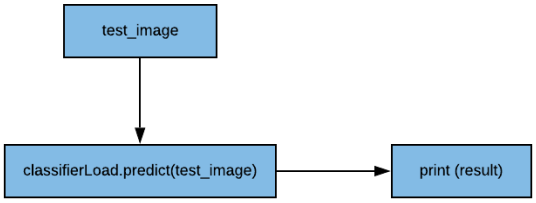

Figure 21: CNN Classifier function call

Major Functions:

The tf.keras.models.load_model('Weight_model.h5') loads a model saved via save_model while training the

classifier. The classifierLoad.predict(test_image) predicts the input comparing the weights from the previous command.

Step 5: Final Output

The final output of this system involves controlling the actuators such as the DC motor, speakers, sending text and call alerts to the mobile phone when a particular gesture is detected. The flowchart of this section is given below:

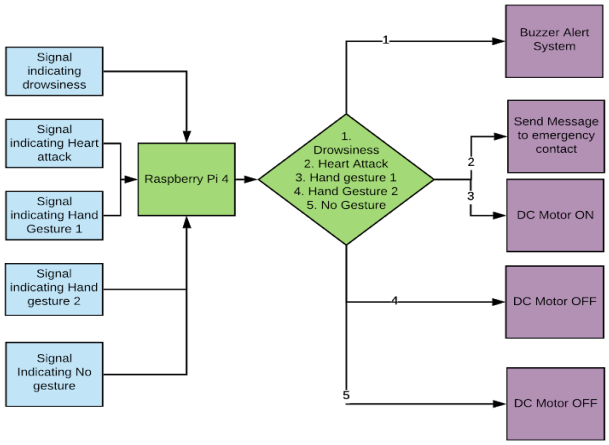

Figure 22: Flowchart of the Embedded System Module

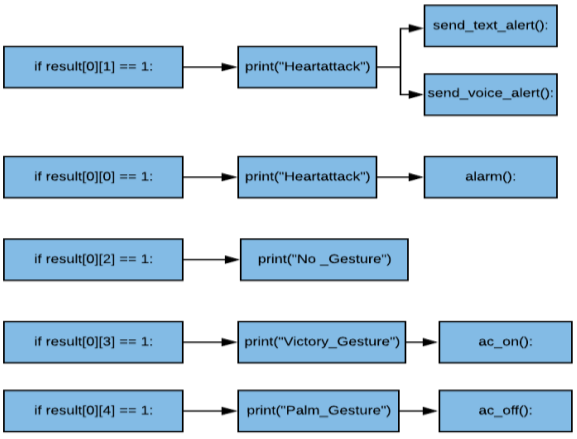

Figure 23: Prediction function call

a. Major Functions:

The send_text_alert() function sends a heath emergency message to a preselected phone number with the location i.e Latitude and longitude using the commands g = geocoder.ip('me'); latlong = g.latlng; lat = latlong[0]; long = latlong[1]. The send_voice_alert() function is used to send a pre-recorded automated call using Twilio Cloud Interface mentioned in section IV.C.6. The above two functions are activated when a heart-attack gesture is detected.
The alarm() function is activated when a drowsiness gesture is detected. Beeping sound is played through the speakers of the car.

The ac_on() and ac_off() functions are used to control the DC motor in this system to demonstrate the gestures displayed.

b. VI. FUNCTIONALITY AND SYSTEM TESTING

In this paper, both unit and integration testing is performed to check the functionality of individual units as well as the integrated system with all the individual units. Detailed testing instructions for both unit and integration testing is provided below:

A. Hardware and Software Sub-System Testing

The first test performed was to ensure that Raspbian OS (NOOBS) is installed in the Raspberry Pi Board. After this test was successfully completed, the second test was performed to check if the board could connect to the WiFi.

Further, tests were conducted to check the external peripherals that have been connected to the Raspberry Pi board. This required wiring the Pi camera, L293D IC and DC Motor to the respective GPIO pins and Camera Slot. The Pi camera communication with the Raspberry Pi board is done by enabling the camera libraries. The images were captured and stored in memory.

The final external peripheral test was to make sure if the Raspberry Pi board could control the DC motor. This was done by sending signals from the GPIO pins to run the DC motor in the clockwise and anti-clockwise direction. These tests were deemed successful.

The software testing involves, first checking the installation of Miniconda. This involves activating the source virtual environment and conda client. The next test was performed to check if all the required libraries are installed correctly with upgraded versions. The conda list is checked to verify the libraries and versions.

The next tests were performed to make sure that the finally trained model is able to classify the unseen sample images. This involves loading the unseen sample images to the trained model and verifying the output.

The next subsystem unit to be tested is the Twilio cloud that allows the system to send an alert to the user. A function was created to send text alerts and pre-recorded automated calls to test the working of the Twilio cloud which were received successfully. The next test was performed to check GPS Coordinates that needed to be sent along with the above-mentioned alert message. To do this, the Google geocoder API function is called. The testing is done by checking the location coordinates on google maps. The final software subsystem test was done on ALSA. Using the play beep sound command, the sound is tested through the speakers. The testing document is embedded below:

Module_Testing.xlsx

B. Integration Testing

Integration testing plays an important role in the software development lifecycle as this will prove whether the whole system is working or not together.

The integration test begins initially with testing the

classification of continuous real-time input from the Pi Camera. This is done by performing each gesture in front of the Pi Camera and checking if the gestures are identified and classified accordingly. These tests were successful except for a few images.

The next texts were performed to check the connectivity between the classifier and Embedded system application. If the output from the classifier is (0,0) the ALSA module of the Embedded system should produce a beeping alarm through the speaker. If the output from the classifier is (0,1) the text alert and pre-recorded call should be sent to the user using the Twilio cloud. If the output from the classifier is (0,2) then no action or alert should be performed by the system. If the output is (0,3), the DC motor should stop the DC motor and if the output is (0,4), The DC motor should run. All tests were verified and successful. The testing document is embedded below:

Integration Testing.xlsx

C. Functionality and Performance Testing

All the Functionality and Performance requirements mentioned in section III were tested and verified. These requirements are met by the successful performance of the system. The testing document is embedded below:

function&performanc etesting.xlsx

VII. CONTRIBUTION

The project has a great contribution to the research industry of deep learning and Convolutional Neural Networks. There are many existing technologies and methods that use deep CNN to either classify and recognize images or perform actions based on gestures but the proposed approach is a single real-time embedded system that classifies the images and also performs actions/ synchronous events based on the gesture type using Raspberry Pi Board.

1. Collected and Captured a dataset of 11,895 images for five categories.
2. Modified and built the CNN Classifier accordingly to get higher accuracy.
3. We followed the Agile Software Development Methodology to build the system, weekly meetings were conducted to check the progress of the project and maintained detailed documentation for each stage.
4. A function to get GPS Coordinates using google geocoder API is implemented and integrated the coordinate values to the body of text alert message and pre-recorded automated call.
5. A function to play an alarm.wav file as Drowsiness Alert using system audio functions is implemented.
6. Tried to Implement Interrupt Driven based Real-Time system which we are not able to get good performance as output.
7. Finally Implemented Pooling method based real-time system with much better performance.

VIII. RESULTS

Convolutional Neural networks have proved to work efficiently for image classification problems with an overall accuracy of 89.34%. Features from around 11,895 images were extracted. This formed the input database for the convolutional neural network. After training CNN, the real-time input images from the Raspberry PI camera were successfully classified into the 5 categories- 1. Drowsiness, 2. Heart Attack, 3. Gesture 1, 4. Gesture 2, and 5. No Gesture. The network is designed such that expected output is

- (0,0) for drowsiness
- (0,1) for Heart Attack
- (0,2) for No Gesture
- (0,3) for Gesture 2 (Palm Gesture)
- (0,4) for Gesture 1(Victory Sign Gesture)

The Embedded system is designed such that the expected output for

- Drowsiness - Beeping of the Alarm using ALSA.
- Heart Attack- Emergency Text alert with GPS location and pre-recorded automated call.
- Gesture 1- DC motor turned on, indicating AC control ON.
- Gesture 2- DC motor turned off, indicating AC control OFF.
- No gesture- No output.

Results discussed based on various cases:

CASE 1: Drowsiness

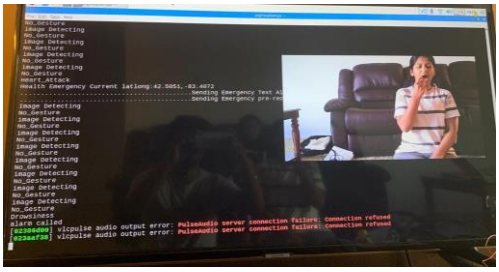

Fig 24: Sample Drowsiness Image Input

Fig 25: Actuator- Speaker

CASE 2: Heart Attack

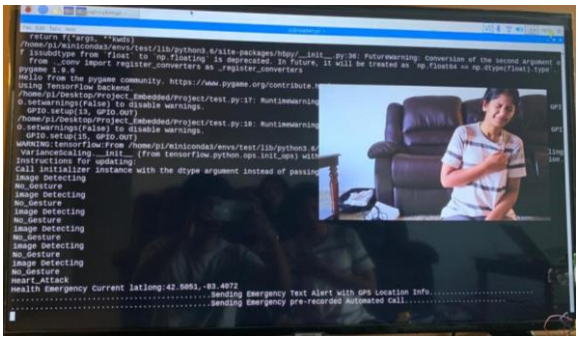

Fig 26: Sample Heart attack Image Input

Emergency alert on phone:

Fig 27: Emergency Text and Call Alert

CASE 3: Gesture 1(Victory Gesture)

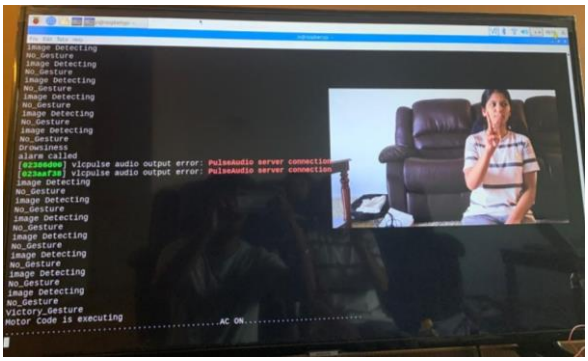

Fig 28: Sample Victory Gesture Image Input

Motor ON:

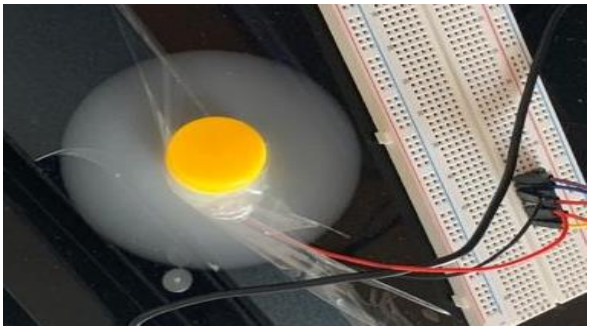

Fig 29: Running DC Motor

CASE 4: Gesture 2(Palm Gesture)

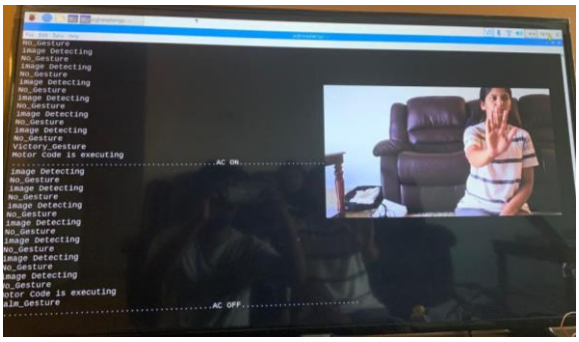

Fig 30: Victory_Gesture Input

Motor OFF:

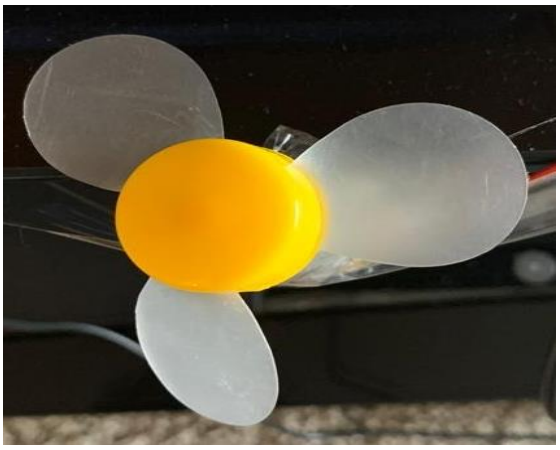

Fig 31: DC Motor OFF

CASE 5: No Gesture

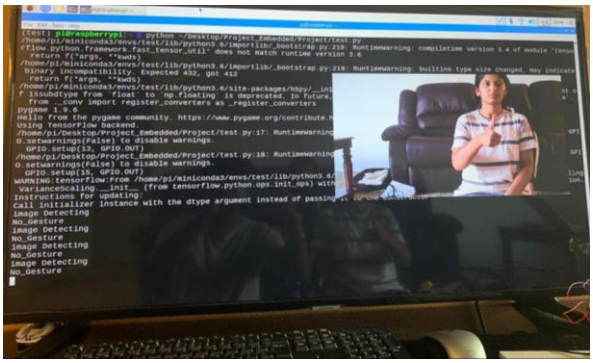

Fig 32: Sample No Gesture Image Input

Final Accuracy:

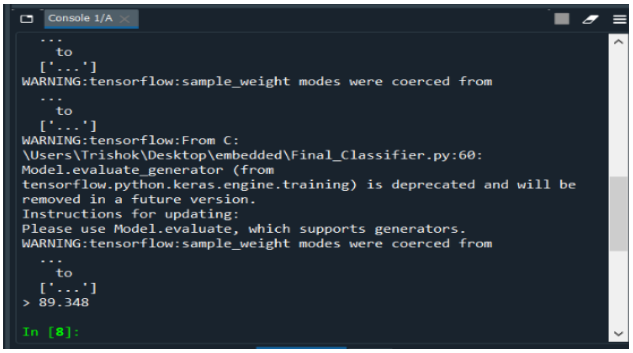

Fig 33: Overall Final Accuracy of the CNN Classifier

IX. CONCLUSION

The purpose of the project was to build a system for reducing driver distraction and to increase driver safety that uses embedded systems and machine learning applications. A Deep Convolutional Neural Network architecture was designed for recognizing the gesture and behavior of the driver. A total of 5 gesture/behavior categories were successfully classified. Different Classifier Models were tested with the test dataset and the accuracy of each classifier model is noted. The model with high accuracy is selected as a final model and used for prediction. The accuracy of the system reaches 89.34%. Based on the above classification, the embedded system was designed to perform 5 different applications in the interface of the automotive vehicle.

The project uses the concepts of real-time embedded systems, I/O events, software development life cycle, and real-time system characteristics. The system is a firm real-time embedded system in which missing few deadlines will not lead to the total failure of the system (misclassifying 2 or 3 images) but missing more than a few deadlines results in the failure of the system. The system performs synchronous events such as sending text messages with GPS Coordinates, alerting drivers with buzzer sound and sending a pre-recorded automated call to the subscribers based on driver gestures defined.

IX. FUTURE WORK

A. *Training with Video Data Set*

In this paper, the network is trained with images data set for all the gestures. In the future, to improve accuracy and performance, the network can be trained with a video data set. This also allows us to have moving gestures which can be more practical for real-time applications.

B. *Generic GPS Receiver*

The paper describes the use of google geocoder API for proving the GPS coordinates of the driver. This results in the location are approximately 2 mins from the actual location. Thus to improve this, a generic GPS receiver can be used to extract the exact location.

C. *Drowsiness detection using IRIS scan*

Drowsiness detection can be further improved by scanning the iris of the eye. The dilation in the iris indicates a clear difference for drowsiness detection.

D. *Gesture recognition with Voice control*

Gesture detection with voice feedback and control would be an added advantage to this system and for the driver use. This would help the driver to have a communication with the car interface.

E. *Heart Attack detection with body wearable bands*

The detection of Heart Attack would be more accurate if the heartbeat of the driver is monitored along with the behavior.

XI. REFERENCES

[1] Nithya Roopa. S “Emotion Recognition from Facial Expression using Deep Learning”, International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-8 Issue-6S, August 2019.

[2] Biao Yang, Jinmeng Cao, Rongrong Ni, Yuyu Zhang “Facial Expression Recognition Using Weighted Mixture Deep Neural Network Based on Double-Channel Facial Images”, IEEE Access, 2018, Volume 6.

[3] D Y Liliana “Emotion recognition from facial expression using deep convolutional neural network”, 2018 International Conference of Computer and Informatics Engineering (IC2IE).

[4] Shervin Minaee, Amirali Abdolrashidi “Deep-Emotion: Facial Expression Recognition Using Attentional Convolutional Network”, arXiv:1902.01019v1 [cs.CV] 4 Feb 2019.

[5] Automatic Facial Expression Recognition Using DCNN Veena Mayya, Radhika M. Pai*, Manohara Pai M. M. 6th International Conference On Advances In Computing & Communications, ICACC 2016, 6-8 September 2016, Cochin, India

[6] A person independent system for recognition of hand postures used in sign language Daniel Kelly *, John McDonald, Charles Markham D. Kelly et al./Pattern Recognition Letters 31 (2010)

[7] Facial Expression Recognition Based on Auxiliary Models Yingying Wang, Yibin Li, Yong Song, and Xuewen Rong Published: 31 October 2019

[8] Facial expression recognition with Convolutional Neural Networks: Coping with few data and the training sample order André Teixeira Lopes, Edilson de Aguiar, Alberto F. De Souza, Thiago Oliveira-Santos Available online 19 July 2016

[9] Recognizing Facial Expressions Using Deep Learning Alexandru Savoiu James Wong Stanford University

[10] Dinh Viet Sang, Nguyen Van Dat, Do Phan Thuan “Facial Expression Recognition Using Deep Convolutional Neural Networks”, 2017 9th International Conference on Knowledge and Systems Engineering(KSE).

[11] Akash Saravanan, Gurudutt Perichetla, Dr. K.S.Gayathri “Facial Emotion Recognition using Convolutional Neural Networks”, arXiv:1910.05602v1 [cs.CV] 12 Oct 2019.

[12] Ali Mollahosseini, David Chan, and Mohammad H. Mahoor “Going Deeper in Facial Expression Recognition using Deep Neural Networks”, IEEE Winter Conference on Applications of Computer Vision (WACV), arXiv:1511.0410 [cs.NE], 2016.

[13] Siyue Xie and Haifeng Hu “Facial Expression Recognition Using Hierarchical Features With Deep Comprehensive Multi patches Aggregation Convolutional Neural Networks”, IEEE Transaction on Multimedia, VOL. 21, NO. 1, JANUARY 2019 211.

[14] Stergiopoulou, N. Papamarkos “Hand gesture recognition using a neural network shape fitting technique”, Engineering Applications of Artificial Intelligence 22 (2009) 1141-1158.

[15] https://www.google.com/search?q=filter+size+in+keras+cnn&oq=filter+size+in+keras+cnn&aqs=chrome..69i57j33l2.12588j0j7&sourceid=chrome&ie=UTF-8

[16] https://en.wikipedia.org/wiki/Convolutional_neural_network

[17] https://www.google.com/search?q=flatten+()+in+cnn&oq=flatten+()+in+cnn&aqs=chrome..69i57j0l7.7204j0j9&sourceid=chrome&ie=UTF-8

[18] https://www.lginjuryfirm.com/common-causes-distracted-driving/

[19] https://www.pyimagesearch.com/2018/12/31/keras-conv2d-and-convolutional-layers/

[20] Implementation of machine learning for gender detection using CNN on raspberry Pi platform Published in 2018 2nd International Conference on Inventive Systems and Control(ICISC)

[21]Michael Christopher Xenya, Crentsil Kwayie, Kester Quist-Aphesti ”Intruder Detection with Alert Using Cloud Based Convolutional Neural Network and Raspberry Pi”, 2019 International Conference on Computing, Computational Modelling and Applications (ICCMA).

[22]Andre Vinicius Borges de Gois Monteiro, Mario Anderson de Oliveira, Ruy De Oliveira, T.Da Silva “Embedded application of Convolutional Neural Networks on Raspberry Pi for SHM”, 2018, Electronics Letters, Vol 54

[23]Chander Subhash, Subhash Chander Verma, Tejaswini, Somya Kumar, M B Siddesh, Kuthika Kr “CNN Based automated weed removal Bot using Raspberry Pi 3”, 2019 JETIR, Volume 6, Issue 5

[24]https://www.google.com/search?q=pi+cmarea&oq=pi+cmarea&aqs=chrome..69i57j0l7.2278j1j9&sourceid=chrome&ie=UTF-8

[25] https://www.adafruit.com/product/3775

[26]https://www.google.com/search?ei=ebmXXp_lBYyStAa-z57ADw&q=twilio+cloud&oq=twilio+cloud&gs_lcp=CgZwc3ktYWIQAzICCAAyAggAMgIIADICCAAyAggAMgIIADICCAAyAggAMgQIABAeMgQIABAeOgQIABBHSgkIFxIFMTAtMzIKCAgYEgQxMC0zUMweWMweYKUhaABwAngAgAFbiAFbkgEBMZgBAKABAaoBB2d3cy13aXo&sclient=psy-ab&ved=0ahUKEwjfheCO6uvoAhUMCc0KHb6nB_gQ4dUDCAw&uact=5