Self-Driving Car Based CNN Deep Learning Model

Amin S. Ibrahim, Omer M. Ibrahim, Ahmed M. Na'eem, Andrew S. Hanna, Shady M. Ezz , Mohamed R. Mohamed, Hassan M. Abdel Hameed, Felopateer Sanad, Hossam I. Hassan, Adel Refky, A.M. Abdel Ghany

Thebes Higher Institute for Engineering, Egypt, engamin_s84@yahoo.com, omer.ibrahim@thebes.edu.eg, ahmed.na'eem @thebes.edu.eg, andrew.hanna@thebes.edu.eg, shady.ezz @thebes.edu.eg, mohamed.mohamed@thebes.edu.eg, hassan.abdel hameed@thebes.edu.eg, felopateer.sanad@thebes.edu.eg, hossam.hassan@thebes.edu.eg, adel.refky@thebes.edu.eg, and *abdelghany.mohamed@thebes.edu.eg*

Supervisor: Faculty of Engineering Helwan University, Department of Electric Power and Machines
Thebes Higher Institute for Engineering, Egypt, abdelghany.mohamed@thebes.edu.eg

Abstract– ***Artificial Intelligent (AI) technology is capable of thinking and recognizing environmental things based on the vast amount of historical training data. AI technology can mimic the human brain with large and complicated computations and short processing time. One of the main challenges in our daily life is the rapid growth of accidents and deaths due to the wrong driving by citizens and unrespecting the traffic rules. Thus, AI technology is coming as a solution to solve this issue through the so-called self-driving car. In this paper, we proposed a self-driving car based on the Convolutional Neural Network (CNN) deep learning model in AI technology. The paper designed and implemented a self-driving car prototype to prove the concept and validate our experimental self-driving car model. It is remarked that a porotype is successfully trained and tested about 5000 images with very low training and validation loss of less than 0.05 and 0.12 respectively.***

Keywords-- Artificial Intelligent, Deep Learning, CNN model, Self-driving Car.

I. Introduction

Recently, the traditional cars and vehicles exhibit air pollution and lake of the Oxygen percentage on the roads due to their CO emissions. Electrical cars and vehicles are considered a solution for the air pollution problems, but in expense of the human being. Both types can suffer from high rates of traffic accidents, which in turn cause deaths and injuries.

In this paper, self-driving car were proposed to reduce the rates of the traffic accidents. In addition, it has a crucial role for developing the smart roads and the smart traffic systems. A proposed self-driving car basically relies on the Artificial Intelligent technology that mimic the driver brain such as thinking, decision making to perform smartly the human processes and tasks during the driving process. Such as rotation, driving left or right…etc [1].

AI technology is to think and interact with things based on a historical experiment in an intelligent way. It can solve a processing issues in the Big data analytics, compared to IoT technology. It can deal with diversified data, handle a hard mathematical computation and process it easily in a very short time. It has an impressive role in our daily life and it is more suited for diverse applications such as automated translation services, Robotics, self-driving car, natural language processing, computer vision, speech processing, voice and face recognition…etc [1].

Deep Learning (DL) algorithms is a subset of Machine Learning (ML) as part of AI technology. DL models are neural networks that include neurons, connectors, bias, and weights [2]. A simple DL model consists of one input layer, one hidden layer, and one output layer. DL network could be extended to diverse hidden layer based on the use case. As the number of collected data to be trained increases in the DL model, the hidden layers will increase and the neural network becomes very complicated. DL models aims to minimize the error difference between desired output and the actual output. In DL models, error was minimized by adjusting the weight and bias of the neural network neurons. which is called backpropagation [2].

CNN is one of many Deep learning models that is more suited for image recognition, computer vision, and self-driving car applications. It can handle massive number of images with huge amount of data to extract the major and correlated features and then obtain a suitable output. As seen in Figure 1, CNN model is composed of three main layers: Convolution layer, pooling layer, and Fully Connected Network (FCN). ReLU activation function is added at the end of each mentioned layer. The convolution layer extract features from images using feature detector (filter). Each input image is convolved with the filter. The result is biased and activated using ReLU activation function to obtain a main feature. The pooling layer is reducing the size of an extracted image features (image dimensioning) before flatting onto the fully connected network, called pooling window. The pooling layer includes max. pooling and average pooling. Each pooling window as part of the convolutional window image (main features) is Flattening into one-dimension matrix before training in the FCN model. Finally, main features were trained and tested for certain number of trails until error was closed to zero as possible. In this moment, a car can drive itself [3, 4].

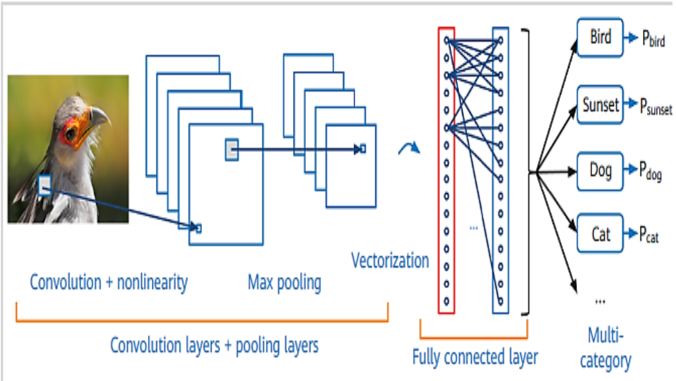


Fig.1 Construction of the CNN deep learning model.

II. SYSTEM MODEL

A proposed car can drive itself based on the incoming data about steering angels, center of the road, traffic signs, and obstacles on the road. Figure 1 shows the generic block diagram of the proposed self-driving car. The self-driving car is passing through four stages as seen in Figure 2. (a) it can collect learn the traffic data by using external sensors and cameras. (b) it can also learn and recognize these data by the training method through the deep learning algorithms as one of AI methods. (c) the traffic data were tested and classified in the output of the deep learning method. (d) the self-driving car remotely adapts itself by using its actuators in order to decide and make a suitable decision based on the traffic data.

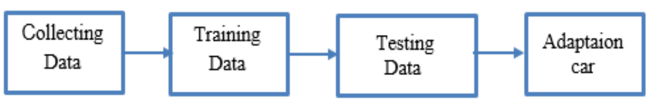


Fig.2 Framework of the proposed Self-driving care.

To perform our proposed self-driving car, some modifications should be added to the normal electric car such as controller module, sensors, actuators, and camera. Figure 3 shows the block diagram of the proposed self-driving car.

The controller is the brain of the proposed self-driving car. It can control and coordinate the driving processes inside the car. It interfaces among camera, sensors, actuators. Sensors detect obstacles location during the movement of car on the road. Camera captures continuous images about the road borders and traffic signs. Thus, the real data traffic was monitored by sensors and camera. A controller continuously collects data traffic from its associated sensors and camera from on one side. It sends main driving commands into the actuators after recognizing, thinking, and performing a suitable task on the other side. At this moment, the actuators begin to move both front and back wheels of the car onto the correct path on the road.

The PC or mobile phone was connected wirelessly with the controller to test the manual driving on the car. Finally, the power system develops all components of the proposed self-driving car by DC battery.

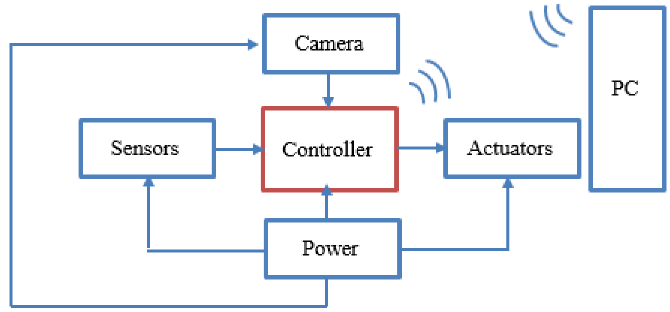


Fig. 3 Generic block diagram of the self-driving car.

III. EXPERIMENTAL SETUP

An experimental setup explores the hardware components used in our proposed system, connectivity among them, and the methodology design to implement it with the help of the software programing, respectively.

A. Hardware Components

A proposed self-driving car consists of six main parts such as sensors, actuators, controller, camera, Personal Computer (PC), and power supply.

1) *Sensors:* The HC-SR04 Ultrasound Sensor was used to measure the distance between the car and the object (obstacles) by a sonar wave. As listed in Table 1, HC-SR04 Ultrasound Sensor has a range detection of 4-meters, coverage angle of 15 degrees, 5V DC supply voltage, and operating frequency of 40KHz. Ultrasound Sensor consists of a transmitting unit and a receiving unit. A transmitting unit radiates an ultrasonic wave with the sound speed of 340 m/s onto an object. While, a receiving unit receives the reflected wave (echo) from an object. Thus, Ultrasonic sensor can easily measure the round trip time of the sound wave during transmission and reflection. Accordingly, the distance between the car and an obstacle could be evaluated from the following equation as [5, 6]:

$$\mathbf{d} = \mathbf{v} \times \mathbf{t} \qquad (1)$$

TABLE I
HC-SR04 ULTRASOUND SENSOR SPECS

Parameters	Specifications
Operating voltage	5V DC
Operating frequency	40KHz
Range	2 Cm~4m
Coverage angle	15°
Ranging Accuracy	3mm

2) *Actuators:* A self-driving car requires two main types of actuators: 28BYJ48 Stepper motors and the control relay. A steeper motor was installed to precisely move both front and end wheels inside the

car. It is more suited for many applications that require precise motion control as car because it can move step by step instead of continuous rotation and works on direct current. The shaft of the steeper motor can rotate every 1.8° with a rate of 200 step per revolution (rev/step). Thus, a steeper motor guarantees a precise and accurate movement for the wheels of the car on the road [7]. Its engine has many coils distributed in groups called "phases". By activating each stage individually, the motor rotates one step at a time (1.8°). The motor was controlled and programmed using the controller for precise positioning and speed control. However, the steeper motor can't directly interface with the controller module. Thus, ULN2003 Driver module enables 28BYJ48 Stepper to connect with the controller module. In addition, it feeds the steeper motor with the required supply voltage (5-12 V) and it regulates the electrical signal onto a series winding coils inside the steeper motor, which in turn moves in a sequence steps until exceeds 200 steps/revolution based on the required movement. On the other hand, the control relay was designed to control the powering in the self-driving car. It turns the electricity ON or OFF based on the status of the car during the driving process on the road. The control relay was operated by programming the controller module.

3) *Controller:* Raspberry pi version 3 module was installed in this paper to control the proposed self-driving car. As depicted in Table 2, it acts as a small computer that includes a processor of 1.2GHz quad-core ARM Cortex-A53 (64-bit), Linux/windows 10 operating system booted by micro SD card, IEEE 802.11 b / g / n Wi-Fi wireless communication, IEEE 802.15 Bluetooth, Micro USB power, Ethernet socket as a gateway for Internet applications, Graphical Processing Unit (GPU) for graphic and video applications, HDMI output, and 1GB memory. In addition, Raspberry pi 3 supports the camera device [8, 9]. Thus, the Raspberry pi 3 was widely used for many use cases such as: Surveillance applications, Robotics, Internet of Things (IoT) applications, Artificial intelligent applications…etc.

TABLE II
RASPBERRY PI 3 MODULE SPECS [8, 9]

Parameters	Specifications
Processor	1.2GHz quad-core ARM Cortex-A53 (64-bit)
Operating system	Linux/windows 10
Wi-Fi	IEEE 802.11 b/g/n-maximum range is up to 50 meters
Bluetooth	IEEE 802.15-maximum range is up to 50 meters
memory	Micro SD card
power	+3.3V, 5V- 2.5A
Video output	HDMI (Speed 1.3 and 1.4) Composite RCA (PAL and NTSC)
Ethernet	10/100 Base T Ethernet socket
memory	1GB LPDDR2
GPIO	40-pins
Camera	Available
GPU	Dual Core Video Core IV® Multimedia Co-Processor. Provides Open GL ES 2.0, with hardware acceleration Open the high-level VG 1080p30 H.264 decoder. Capable of 1Gpixel/s or 1.5Gtexel/s or 24GFLOPs with fabric filtering and DMA infrastructure

4) *Camera:* A camera could be easily connected onto the raspberry pi 3 by the act of the camera connector that attributed to15-pin MIPI Camera Serial Interface (CSI-2). Table 3 depicts the specs of the Raspberry pi 3 camera. A camera is a source of video stream and image information that coexists with raspberry pi 3 to perform the required surveillance and image processing applications. GPU built in the raspberry pi 3 is a graphical processor that controls the video and image applications, specifically in the AI technology.

TABLE III
SPECS OF CAMERA V.2 [3]

Parameters	Specifications
Resolution	8 Mega pixels
Video modes	1080p30, 720p60, and 640×480p60/90
Linux integration	V4L2 driver available
Sensor	Sony IMX219
Sensor resolution	3280×2464 pixels
Picture formats	JPG, JPEG+DNG(raw), BMP, PNG, RGB888, YUV420

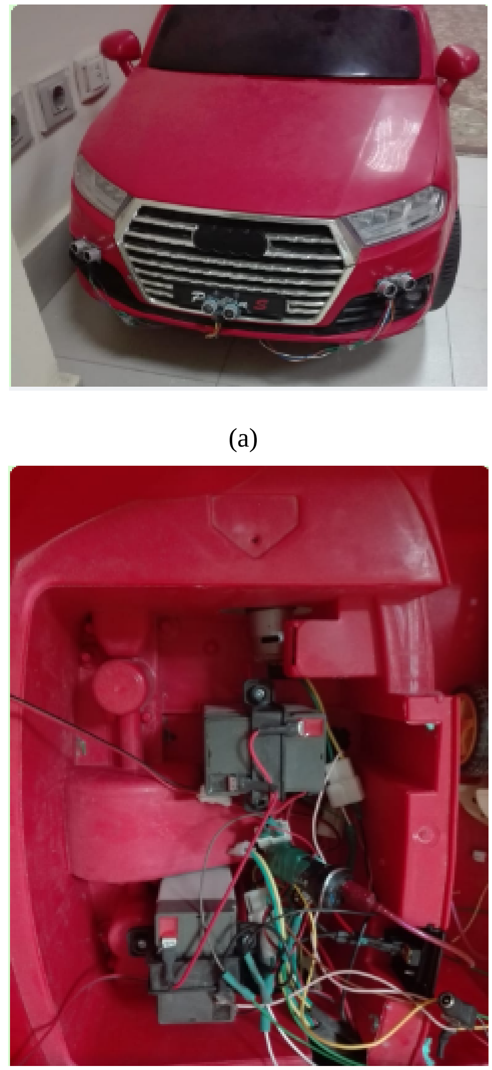
5) *Personal Computer:* PC or the mobile phone has a crucial role in the proposed self-driving car. It connected with Raspberry pi 3 using HDMI cable to setup Linux operating system on Raspberry pi processor chip, typing python codes for programming ultrasonic sensor, steeper motor, configuration of camera on Raspberry pi3. While, it also connected wirelessly to Wi-Fi of the Raspberry pi 3 using its wireless network interface card in order to operate the manual driving python code later in the experimental scenario.

6) *Power Supply:* Our proposed self-driving car has two rechargeable batteries of the type Valve Regulated

Sealed Lead-Acid that recharges immediately after discharge process. It is characterized as 6V4AH/20HR to achieve long battery life time with high number of charges. Two batteries energize all hardware components inside the self-driving car based their requirements. Such as Raspberry pi 3 requires 3.3~5 VDC, camera energized from the Raspberry pi 3 itself, Ultrasonic sensor powered by GPIO of the Raspberry pi 3, control relay needs 5V DC supply to operate, steeper motors operate at 5-12VDC.

Finally, a proposed self-driving car consists of 4 Ultrasonic sensors, 3 steeper motors, one Raspberry pi 3 module, 2 control relay, one camera, and two rechargeable batteries. A 4 Ultrasonic sensors are distributed over the car as 3 ultrasonic sensors in the front of the prototype, each with its associated direction (right, left, center), and one ultrasonic sensor located in the end of the prototype. Each one has its associated direction. Three steeper motors are located in the car wheels as two steeper motors for the two back wheels and one steeper motor for the front wheels. The camera is centered at the front of the car. While, Raspberry pi 3 module, batteries, and control relays could be located at any place inside the prototype based on their connections. Figures 4 and 5 show a proposed self-driving car prototype during setup the model and its associated hardware components.

Fig. 4 Proposed self-driving car prototype.



(a)

(b)

Fig. 5: Parts of hardware components inside the self-driving car prototype (a) front ultrasonic sensors, (b) Batteries and Raspberry pi 3 module.

B. Design Connectivity

The aforementioned prototype elements are connected with the Raspberry pi 3 controller unit via 40-GPIO pins according to the schematic diagram and datasheets for each hardware components except camera that connected by 15-pins video output and the batteries that energize the Raspberry pi 3 module by its power socket. We connect Raspberry pi 3 module with any monitor unit by using HDMI cable to operate it. Figure 6 shows the connection between Raspberry pi 3 with Ultrasonic sensor.

C. Design programming

Before the programming process, we begin to power ON the Raspberry pi 3 then setup the Linux (Fedora) operating system on it through the Micro SD memory card. After Linux setup process, we setup Python GPIO library to enable Raspberry pi 3 by typing the command " sudonano file name.py" in Linux to control and manage all prototype elements using python language programming.

In the programming process, a set of procedures were executed based on the experimental scenario of the self-driving car.

D. Design Methodology

The realistic experiment was tested and implemented to prove the concept. A self-driving car are placed on the actual road to move as the traditional cars without human interaction. The experiment includes three main scenarios: standby scenario, driving scenario, and termination scenario.

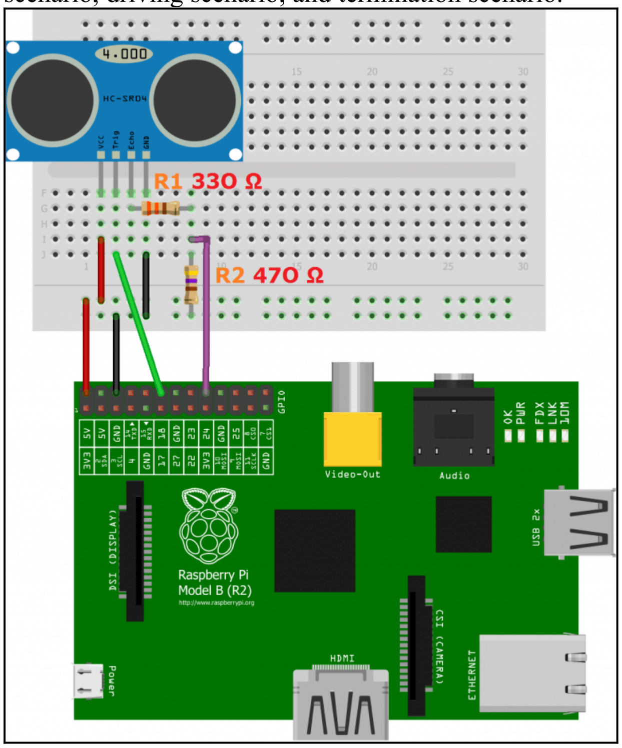


Fig. 6 Schematic diagram of Ultrasonic sensor with Raspberry pi 3.

1) *Standby Scenario:* In the standby scenario, the batteries supply the Raspberry pi 3 module by the required voltage (5VDC), which in turn energize control relays, ultrasonic sensors, camera, and steeper motors by the voltage source. In addition, the python codes that are burned on the Raspberry pi processor chip are ready to be operated. In this mode, the control relays will turn ON and allow steeper motors ready to move the wheels of the self-*driving car. The* speed of steeper motors gradually speeds up until saturated onto the needed speed. Figure 7 shows a part of the python code for the ultrasonic sensor.

```
    # save time of arrival
    while GPIO.input(GPIO_ECHO) == 1:
StopTime = time.time()
    # time difference between start and arrival
TimeElapsed = StopTime - StartTime
    # multiply with the sonic speed (34300 cm/s)
    # and divide by 2, because there and back
    distance = (TimeElapsed * 34300) / 2
    return distance
```

Fig.7 Part of ultrasonic sensor using python code.

2) Driving Scenario: In the driving process scenario, a self-driving car was moved along the road. This scenario is divided into two main states: manual driving state and training model state.In the manual driving state, Putty desktop application makes a connection between PC and raspberry pi. It enables a user to send Linux commands to Raspberry pi 3 and runs a manual driving python code. VNC viewer program acts as graphical user interface that views or displays the python codes on Raspberry pi. "MainRaspCode.py" python code were executed on Raspberry pi 3 module to manually drive a car many times on a variety of roads with different traffic signs, borders, geometric shapes, center…etc. In the same time, ultrasonic sensors send sound waves to the road and wait echo (reflected waves) if any object or obstacles are presented. In case of any obstacles were detected, Raspberry pi 3 was programmed to receive reflected wave from the four ultrasonic sensors whether in the front or the back of the car to measure the distance between a car and the object (object location), then it controls the movement of the steeper motors to shift right or left based on the direction of the detected object on the road. For example, if any object is detected in the left of the road by an ultrasonic sensor, Raspberry pi 3 controls steeper motors of the two front wheels to shift right to avoid the collision and vice versa. In the training model state, a car was trained to enable the car to drive itself. Figure 8 shows the block diagram of the sequential processes required to obtain our proposed self-driving car. Table 4 lists the main parameters for creating CNN model.

TABLE VI
METHODOLOGY DESIGN SIMULATION PARAMETERS

parameters	values
Image #	5649
Training data	4332
Testing data	1084
epochs	30
# trails	38
Epoch	100 images
Activation function	ReLU
Optimizer	Adam

Fig. 8 Methodology design of a proposed self-driving car.

- Before the training process, data were collected by the act of the manual process or the simulation process. In case of the manual driving car, a camera continuously captures massive number of images about the road attributes. Such as center, left, Right, Steering, Throttle. The acquired images were collected onto Raspberry pi 3 module. In case of the simulation process, a self-driving model simulator were installed from Udamy simulation platform to acquire steering angels and capture diverse images by its camera in order to obtain roads attributes.
- In the preprocessing data, corrupted images acquired by camera will be removed. Images that describe un-useful information about the main features of the road will be discarded. Captured images were visualized versus the steering angels that ranged from -1 to +1 with the center of 0 value. Figure 9 shows the steering angles of thousands of images. It is noted that images that captured steering angels of [-1,+1] range should be discarded and it is not allowed to train these images. During the preprocessing data, about 233 images were removed from 5416 images because these images record steering angels out of range.
- In the splitting process, data images were split into 80% training data and 20 % testing data as shown in Figure 10.
- In the training and testing processes, the CNN model python code was created using Tensor Flow libraries in Figure 11 to extract features from the collected images and convert features onto discrete output (actions). As shown in Figure 12, the acquired images were preprocessed by using convolution layer and pooling layer to obtain the important features from images. The extracted features are sent onto the fully connected neural network. The extracted features the fully connected network were back propagated to adapt or change weights of the neurons until the output data fit the desired output and the error difference is close to zero. Before the training process, the training data consists of multiple epochs to obtain high processing time and minimize the training and testing loss.
- In the output process, a car recognized the center road based on the historical steering angels and automatically drive itself (rotate, shift right, shift left) based on the historical image captured by camera. A car was tested on the real road, carrying the trained CNN model code on its Raspberry pi 3 module. The new captured images on the real road are considered the testing data.

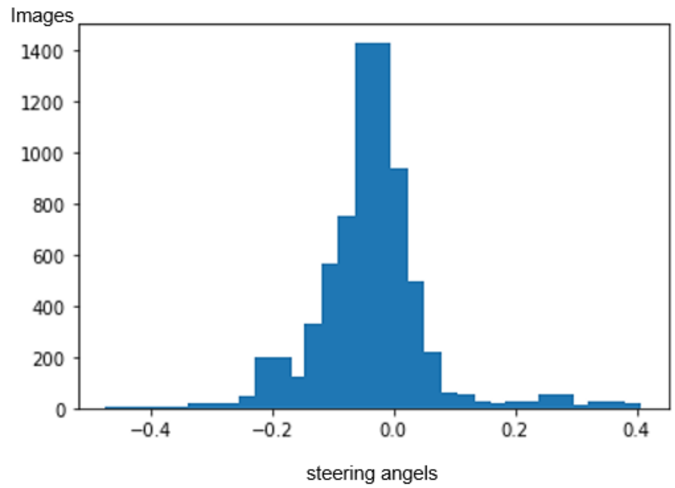


Fig. 9 Visualization of images over various steering angels.

```
xTrain, xVal, yTrain, yVal = train_test_split(imagesPath, steerings,
test_size=0.2,random_state=10)
print('Total Training Images: ',len(xTrain))
print('Total Validation Images: ',len(xVal))

Total Training Images:  4332
Total Validation Images:  1084
```

Fig. 10 Splitting input data before training model [10].

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import
Convolution2D,Flatten,Dense,Lambda,Dropout
from tensorflow.keras.optimizers import Adam
import h5py

from tensorflow.keras.models import load_model
from keras.callbacks import ModelCheckpoint
```

Fig. 11 Tensor Flow libraries for creating CNN model.

```
def createModel():
    model = Sequential()
    model.add(Convolution2D(24, (5, 5), (2, 2), input_shape=(66, 200,
3), activation='elu'))
    model.add(Convolution2D(36, (5, 5), (2, 2), activation='elu'))
    model.add(Convolution2D(48, (5, 5), (2, 2), activation='elu'))
    model.add(Convolution2D(64, (3, 3), activation='elu'))
    model.add(Convolution2D(64, (3, 3), activation='elu'))

    model.add(Flatten())
    model.add(Dense(100, activation = 'elu'))
    model.add(Dense(50, activation = 'elu'))
    model.add(Dense(10, activation = 'elu'))
    model.add(Dense(1))

    model.compile(Adam(lr=0.0001),loss='mse')
    return model
```

Fig. 12 Creating of the CNN model [11, 12].

3) *Release Scenario:* the final scenario, the self-driving car stop moving along the road when the Raspberry pi 3 decides to gradually decrease the speed of the wheels by the steeper motors and then stop working by the control relays.

IV. EXPERIMENTAL RESULTS

The experimental results shed light on the sequential processes executed in the CNN deep learning model to obtain self-driving car.

Figure 13 shows the results of creating CNN deep learning model, including the structure of CNN neural networks as the number of layers, size of each layer number, and number of trainable parameters for each CNN layer.

After CNN model creation, data were trained through 30 epochs. Each epoch is trained using backpropagation 300 times with total 900,000 operations in the full training process. Figure 14 shows the relationship between the training loss and the validation loss versus the number of epochs. It is noticed that as the number of epochs increases, the training loss and the validation loss will exponentially decrease. As seen in Figure 15, The training loss is decreased from 0.2758 at the first epoch onto 0.0792 at the 15th epoch. The validation loss is reduced from 0.3716 at the first epoch to 0.1537 at the 15th epoch. With low number of epochs, it is remarked that both training and validation loss values appear slightly decrease, as seen in Figure 16.

Layer (type)	Output Shape	Param #
conv2d_5 (Conv2D)	(None, 31, 98, 24)	1824
conv2d_6 (Conv2D)	(None, 14, 47, 36)	21636
conv2d_7 (Conv2D)	(None, 5, 22, 48)	43248
conv2d_8 (Conv2D)	(None, 3, 20, 64)	27712
conv2d_9 (Conv2D)	(None, 1, 18, 64)	36928
flatten_1 (Flatten)	(None, 1152)	0
dense_4 (Dense)	(None, 100)	115300
dense_5 (Dense)	(None, 50)	5050
dense_6 (Dense)	(None, 10)	510
dense_7 (Dense)	(None, 1)	11

Total params: 252,219
Trainable params: 252,219
Non-trainable params: 0

Fig.13 The results of creating a sequential CNN deep learning model

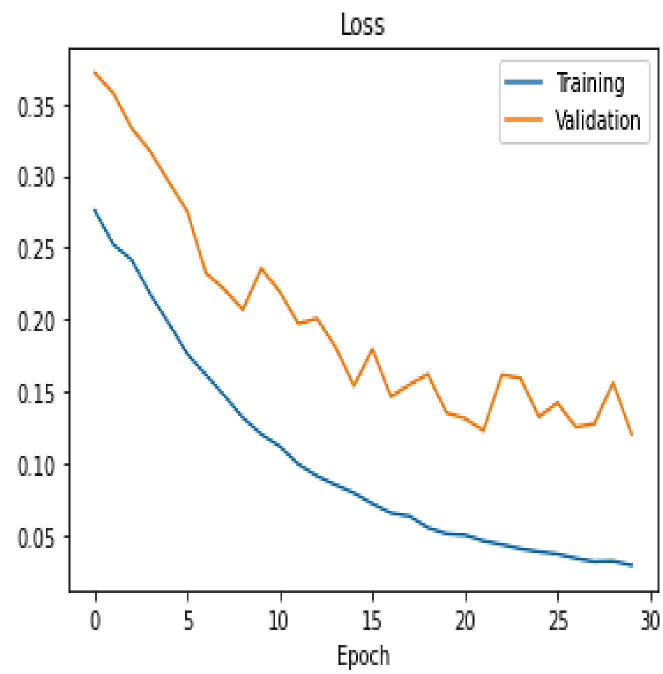
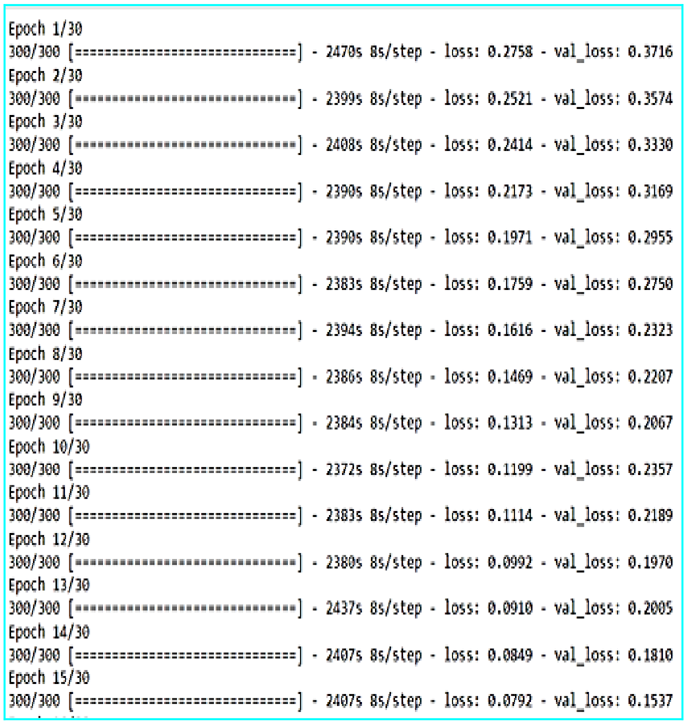


Fig.14 Training and validation loss versus number of epochs



```
Epoch 1/30
300/300 [==============================] - 2470s 8s/step - loss: 0.2758 - val_loss: 0.3716
Epoch 2/30
300/300 [==============================] - 2399s 8s/step - loss: 0.2521 - val_loss: 0.3574
Epoch 3/30
300/300 [==============================] - 2408s 8s/step - loss: 0.2414 - val_loss: 0.3330
Epoch 4/30
300/300 [==============================] - 2390s 8s/step - loss: 0.2173 - val_loss: 0.3169
Epoch 5/30
300/300 [==============================] - 2390s 8s/step - loss: 0.1971 - val_loss: 0.2955
Epoch 6/30
300/300 [==============================] - 2383s 8s/step - loss: 0.1759 - val_loss: 0.2750
Epoch 7/30
300/300 [==============================] - 2394s 8s/step - loss: 0.1616 - val_loss: 0.2323
Epoch 8/30
300/300 [==============================] - 2386s 8s/step - loss: 0.1469 - val_loss: 0.2207
Epoch 9/30
300/300 [==============================] - 2384s 8s/step - loss: 0.1313 - val_loss: 0.2067
Epoch 10/30
300/300 [==============================] - 2372s 8s/step - loss: 0.1199 - val_loss: 0.2357
Epoch 11/30
300/300 [==============================] - 2383s 8s/step - loss: 0.1114 - val_loss: 0.2189
Epoch 12/30
300/300 [==============================] - 2380s 8s/step - loss: 0.0992 - val_loss: 0.1970
Epoch 13/30
300/300 [==============================] - 2437s 8s/step - loss: 0.0910 - val_loss: 0.2005
Epoch 14/30
300/300 [==============================] - 2407s 8s/step - loss: 0.0849 - val_loss: 0.1810
Epoch 15/30
300/300 [==============================] - 2407s 8s/step - loss: 0.0792 - val_loss: 0.1537
```

Fig.15 The results of the data training process

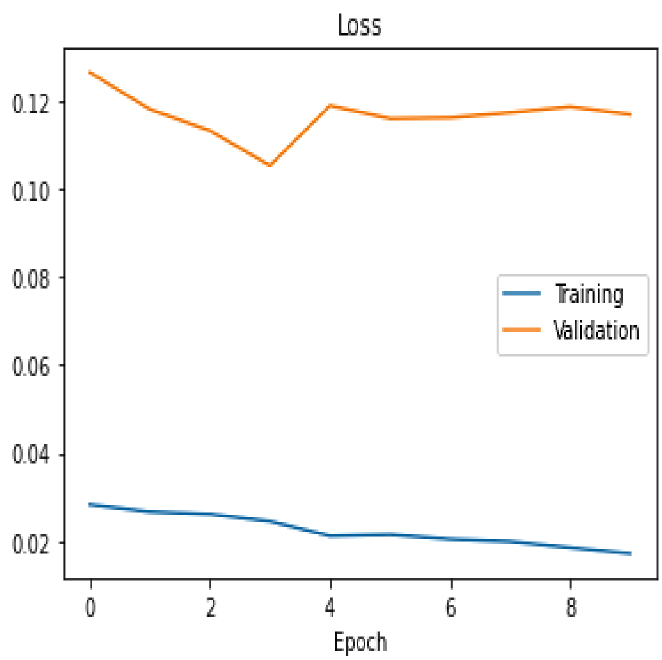


Fig. 16 Training and validation loss for low number of epochs

V. CONCLUSION

The paper proposed a self-driving car based on the deep learning model as part of AI technology to meet the human driving mistakes, which increase the rates of the deaths. We install a realistic prototype, including hardware components, connectivity among components, programing using python code, and methodology design to implement this work in the real time. Raspberry pi module was considered a good candidate for performing the AI neural network models. Our proposed prototype was trained and tested using CNN model in order to drive itself automatically without t\he human being.

REFERENCES

[1] Roza Dastres, and Mohsen Soori, " Artificial Neural Network Systems", International Journal of Imaging and Robotics, volume 21; Issue No. 2; Year 2021.

[2]Amitha Mathew, P.Amudha, and S.Sivakumari, " Deep Learning Techniques: An Overview",n book: Advanced Machine Learning Technologies and Applications (pp.599-608), 2021.

[3]Anirudha Ghosh, Abu Sufian, Farhana Sultana, Amlan Chakrabarti, Debashis De, "Fundamental Concepts of Convolutional Neural Network", Recent Trends and Advances in Artificial Intelligence and Internet of Things (pp.519-567), 2020.

[4] Sakshi Indoliaa, Anil Kumar Goswamib , S. P. Mishrab , Pooja Asopaa, " Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach", International Conference on Computational Intelligence and Data Science (ICCIDS 2018).

[5] Fares Ng, "Ultrasonic Sensors", 2020.

[6] https://www.ia.omron.com/data_pdf/guide/50/ultrasonic_tg_e_1_1.pdf

[7]https://lastminuteengineers.com/l298n-dc-stepper-driver-arduino-tutorial/2015

[8] https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf

[9]https://www.raspberrypi.com/documentation/accessories/camera.html, last accessed 2022.

[10]https://docs.microsoft.com/arsa/aspnet/core/signalr/hubs?view=aspnetcore-6.0 2020/2

[11] https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html 2009

[12] https://www.nvidia.com/en-us/self-driving-cars/2021