Al-Rafidain Engineering Journal (AREJ) Vol.28, No.2, September 2023, pp. 261-283

Enhancing Drivers' Attention By A Smart Binary Matching Machine to Avoid Accidents

Orjuwan M. Abduljawad Al-Jawadi
arjuwan_m@ntu.edu.iq
Computer Engineering Technology Department, Engineering Technical College\Mosul, Northern Technical University, Mosul, Iraq

Received: March 9th, 2023 Received in revised form: April 26th , 2023 Accepted: Augest 6th, 2023

ABSTRACT

Stress and sudden difficult situations have raised the risks of accidents down the roads. The drivers' attention might be distracted out in seconds under unexpected circumstances, which could take place due to bad weather, vision problems, fatigue for long driving hours, damaged or broken Traffic light , and even children's noise inside the car. In this paper, I proposed to develop a special colourful Deep Back Propagation Neural Network to enhance drivers' attention by observing different traffic light cases using a suggested smart binary matching machine system in Python. The smart machine system will analyse and identify the real Traffic light from art signs, broken or damaged ones; in addition to pedestrian signs based on a Database symbols for each case, which have taken the basic Traffic light and signs, and developed them to damaged cases or unreal one, before making the right decision by the learned network, then send an enhanced feedback signal to the driver. The algorithm consisted of accurate image processing steps, with two long stages of full contents features extraction vectors to be handled by Red-Yellow-Green Shallow and Deep Back Propagation Neural Networks (SBPNN) and (DBPNN) for each complex case. As a result, the algorithm rated a high accuracy of 100%, which is the most important factor to maintain safety, recoding the true label output as 1-value, with a predicated tested ouput 1.0-value. The suggested system does not replace the driver's one decision, yet it is an enhancing backup classification and recognition system before things move out of control. The feedback signal calculated based on reducing costs for 2500 iterations with The leas minimum value 000012,and can be developed as a voice signal warning Message, to increase the awareness of the drivers, besides the warning text on the screen.

Keywords:

Red-Yellow-Green SBPNN, Red-Yellow-Green DBPNN, Smart Binary Matching System, enhancing backup classification and recognition system.

https://rengj.mosuljournals.com
Email: alrafidain_engjournal1@uomosul.edu.iq

==========

1. INTRODUCTION

Recently, deep learning neural networks have made a robust technique in representing Real - World applications for computer vision tasks. It has become a powerful tool to support detection, classification , and recognition processes, working on different sets of Databasses, including structured and unstructured data. Traffic Light Recognition (TLR) is still presented by many researchers to develop the concept of Intelligent Transportation Systems (ITS) [1,2].

Deep learning networks are not a replacement for human Drivers but work mostly as a saving option, and warning systems when things go out of control. In addition, driving for long hours increases the risk of fatigue, problems, which in turn increase accidents. Distracted driving not only puts drivers' lives at risk but poses a great risk to other passengers and people passing on the same road. The limitation of Artificial Intelligence Neural Networks (AINNs) has been reduced gradually with the computer Hardware's ontinuous improvement [3].

Deep learning has also developed the Machine learning methods to simulate and mimic the neural

model of humans' brains on how to process information, and make decisions [2,4,5].

Traffic facilities are available with guidance and instructions to direct the drivers on the road. It contains Traffic light , pedestrian signs, Crosswalks, Stops And under-construction signs and symbols [6]. Therefore, in this proposed paper, I focused on Traffic light , classifying the lights by their colours to make the right decision by the intelligent machine, and follow the correct action accordingly when Traffic light Conditions are working Probably There are some cases that must be taken into onsideration when working with intelligent Machines to mimic human brains:

1.Traffic light is damaged, broken, or not working at all, which means all the lights are turned off.

2. One of the lights' colours is fading, or giving a weak signal.

3. Transition between lights' states, and it happened in the case of red-yellow lights, the machine should know it is only in a ready state, and it should not move.

4. All lights are working at the same time, this case is rare, but not impossible under bad weather circumstances, power outage, or other mechanical failure.

Real world problems are very challenging to intelligent machines, it has to predict the right decision before taking the Accurate action. However, deep learning algorithms are can deal with dynamic events in real time, and work on massive numbers of data with many parameters, and huge databases [7].

2. LITERATURE REVIEW

Traffic light , signs detection, classification, and recognition have been recently under investigation for massive numbers of research. With deep learning, artificial intelligence took a new course in developing auto driving systems, smart vehicles, and intelligent transportation systems. The concepts are quite different with the very same principles, but the aim is one, safety. According to [1], The authors proposed a deep Learning -based detection network with prior maps for autonomous cars. The authors in refernce [2] roposed to use Traffic Sign Recognition (TSR) not only to protect Drivers but also to inspect the using of traffic signs on roads accurately, reducing the need to depend on human power and resources. Moreover, feature extraction is one of the most important steps before working with deep learning, it could be a challenging step in selecting images and building databases. They integrate the power of deep learning to recognize relevant Traffic light of predefined routes by combining a state – of – deep detector with precise location. Another work made use of a real-time detection concept with a based camera system [4], proposing a deepTLR and classifying the lights with a single deep convolutional network. As Traffic Sign Recognition (TSR) became an important application to avoid traffic accidents, [6] Proposed lightweight real-time traffic signs images based on YOLO by combining the methods of deep learning, working on real scenes' data sets, and comparing them to the current detecting model. So working with real-time Traffic light gives practical significance to improving road traffic safety[8].

Another challenging problem in working with images recognition is driving in complex scenarios, weather conditions, and external environments could affect how the autonomous cars recognize images, according to [9], integrated deep learning and multi-sensor data fusion assist (MSDA) by improving vision – based Traffic light recognition algorithm, according to the used technique in [10], also integrated the Computer vision and machine learning with CNN to extract and detect features from Avisual camera, improving recognition accuracy with an on-board GPS sensor to identify region – of – interest in the image, which contains the Traffic light , assisting in improve recognition under low illumination conditions. Besides, other research papers worked on the problem of identifying and perceiving traffic symbols and signs using CNN deep networking, based on color segmentation and shape matching [11,12,13,14].

Using automatic Large–scale data or satellite imagery with deep learning also enhanced the process of image classification as in [11], who proposed a comparison study of models trained to solve the problem of crosswalk classification. [13,15], proposed two points of view, traffic sign detection, and evaluating deep neural network architectures for object detection[16].

Another technique with a lightweight deep network to classify traffic signs was proposed by [17], who designed a training model (the teacher network) to transfer knowledge to a smaller model (the student).

From exploring many numbers of researches, most of them have used CNN for features extraction, [18] introduced a MicroNet, A highly compact

CNN for classifying real-time embedded traffic signs, while [19] introduced a Fully Convolutional Network (FCN), to investigate the effect of extracting features on an imbalanced dataset of small objects based on shape and colors. In addition, many techniques were produced in Traffic light image classification, [20], proposed (SSRN), a supervised deep learning, using a 3D convolutional layer, in which every layer regularizes the learning process and improves the classification performance.

[21], proposed (TSingNet) to detect and recognize occluded and small traffic signs in the wilds, based on a pyramid network to learn scale-aware and context-rich features. Image processing pipeline and CNN were suggested to learn the behavior of self-driving [22]. Other papers used the powerful effect of CNN. [23,24,25,26,27,28] So, using deep learning in the classification f traffic networks has produced higher accuracy than traditional machine learning [29,30,31].

Analyzing Traffic light and signs for detection, classification , and recognition processes are still considered in developing Computer-vision and machine learning [32]. Further investigation will require more papers on the same subject, following the changes in the roads, cities, and countries around the world.

3. THE PROPOSED METHODOLOGY

3.1 Shallow Back Propagation Neural Network as a Binary Classifier for Traffic light (SBPNN)

Supervised learning is an important part of machine learning, it requires a mapping between input and a correct data output to predict an actual output from unknown data [25, 30, eq. (1)]. It aims to accurately predict the classification of tested data based on a previously defined Database with trained samples. Shallow Neural Networks algorithms are used in supervised learning, with a single hidden layer, where features are simply extracted from the training data to identify patterns of the database, as in the following equation:

$$Y = f(X). \quad \dots (1)$$

Where,

Y: The predicted output.

$f(X)$: Function the input

In the paper, the traffic light is assumed to be a supervised binary classifier for light colors, in which only one state will be logic one as the green light turns on, and the driver should move, while the other states require the driver to stop when the logic zero is resulted on the output. Table (1) summaries the required action for each light color.

Table 1: Binary Mapping of Traffic Light Colors

Lights Colours Input	Binary States	Action	Lights Output
Red	00	Stop	0
Red – Yellow	01	Wait	0
Yellow	10	Ready	0
Green	11	Go	1

The suggested SBPNN was trained with two inputs-binary states, five hidden units, and one output, [2-5-1] layers dimension, built and evaluated in Python to satisfy the results of Table(1), recording a 100% accuracy with a decision boundary, and classifying the states of the lights colours according to their output states, see Figure (1), where 3-binary inputs under different labels (Stop, Wait, and Ready) ask the driver to stop the car in the red boundary when the output is logic zero, while one state requires the driver to move its car when the output is logic one for green light input. Consequently, the results will be the same no matter how many hidden units are added to the hidden layer. Increasing the number of training iterations will result in decreasing the cost value as can be noticed, and the decision boundary result will be the same, reporting cost after iteration 0 is 0.693147, and cost after iteration 9000 is 0.00003. The error has decreased to a minimum value that no longer training is required.

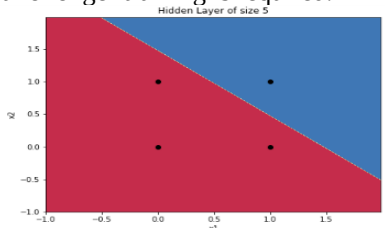


Fig.1 Binary Classifier (SBPNN) for Traffic light ' Results by Python

3.2 Deep Back Propagation Neural Network as a Binary Classifier for Traffic light (DBPNN)

The proposed SBPNN-binary classifier works well with the basic rules of classifying Traffic light according to binary values only, but it is limited to work with unstructured data like images, as they are hard to understand by computers, containing many complicated features, such as colours, lights, illumination, shape-based, textures and morphological operations. So, with Traffic light classification and recognition, the probability of

changing the above states is very possible in the real world, considering weather conditions, sudden shutdown of electrical power, and damaged Trafficlight From this concept, the (SBPNN) was developed to work as a supervised (DBPNN), where the features of images can be extracted gradually as they pass through the network's layers, causing the image to diminish eventually passing each layer [28,33]. The low level features are extracted by the initial layers, and the high-level features are extracted by the deeper layers, combining a full representation for the input, where the binary classifier output is either one or zero. The algorithm aims to reduce the error between its predicted result, and training data models to recognize the patterns [31,32,34].
Tables (2, 3, and 4) show the proposed binary classifier map for the developed (Deep Back Propagation Neural Network), and how to understand its work hypothetically for repressing each light color.

Table 2: Binary Mapping of Red-DBPNN when Red Light turns on

Red	Yellow	Green	Lights Inputs States	Logical Output	Lights Output States	Intelligent Machine Action
0	0	0	Off - Off – Off	0	Power Off	Stop
0	0	1	Off - Off – On	0	Damaged	Stop
0	1	0	Off - On - Off	0	Wait	Stop
0	1	1	Off - On – On	0	Ready	Stop
1	0	0	On - Off - Off	1	Red is On	Stop
1	0	1	On - Off – On	0	Wrong Transition	Stop
1	1	0	On - On - Off	0	Ready	Stop
1	1	1	On - On – On	0	Damaged	Stop

According to Table 2 at the time that the red ight is on, other lights should be off, it is the case of (1-0-0), where the logical output should be (1), and the driver should stop, if it happens and the green (0-0-1), or yellow light (0-1-0), turns on when it is not supposed to be then the logical output should be (0), reporting a damaged state in that case. By counting the required seconds for each light when it turns on, the damaged or wrong state can be accurately specified. The same rules have been applied for binary mapping yellow light compared to other lights' cases as can be seen in Table (3), and for green light as can be seen in Table (4).

Table 3: Binary Mapping of Yellow-DBPNN when Yellow turns on

Red	Yellow	Green	Lights Inputs States	Logical Output	Lights Output States	Intelligent Machine Action
0	0	0	Off - Off – Off	0	Power Off	Stop
0	0	1	Off - Off – On	0	Damaged	Stop
0	1	0	Off - On – Off	1	Yellow is On	Wait
0	1	1	Off - On – On	0	Ready	Stop
1	0	0	On - Off – Off	0	Stop	Stop
1	0	1	On - Off – On	0	Wrong Transition	Stop
1	1	0	On - On – Off	0	Ready	Stop
1	1	1	On - On – On	0	Damaged	Stop

Table 4:Binary Mapping of Green-DBPNN when Green Light turns on

Red	Yellow	Green	Lights Inputs States	Logical Output	Lights Output States	Intelligent Machine Action
0	0	0	Off – Off – Off	0	Power Off	Stop
0	0	1	Off - Off – On	0	Green is On	Go
0	1	0	Off - On – Off	1	Wait	Stop
0	1	1	Off - On – On	0	Ready	Stop
1	0	0	On - Off – Off	0	Stop	Stop
1	0	1	On - Off – On	0	Wrong Transition	Stop
1	1	0	On - On – Off	0	Ready	Stop
1	1	1	On - On – On	0	Damaged	Stop

3.3 DATA COLLECTION AND IMAGE PROCESSING

3.3.1 Building a Database

The work does not depend on real- time images, but on a collection of real scenes and art images that represent the states of light colors under different circumstances as can be noticed of light output states in Tables (2, 3, 4). The database of the collected images contain three categories:

A. Traffic light images, which function correctly, as shown in Figure (2).

Fig. 2 Traffic Light functions correctly

B. Traffic light images, which reflect a damaged state, as shown in Figure (3).

1. All lights are on
2. All lights are off
3. Two lights are on: the probability of red and green turning on at the same time. It is a rare but not impossible.

1 2 3

Fig. 3 Traffic Light with Damaged Lights Colours states

C. Traffic Symbols on streets, which only represent an art, but not real raffic light images, as shown in Figure (4).

Fig. 4 Traffic Light Symbols

D. Signs on the roads, which represent the following states, as shown in Figure(5):

1. Stop Sign
2. Crosswalk
3. under construction

Fig. 5 Signs on the Road with Different States

3.3.2 Image Processing in Python

Image processing required two necessary steps before starting the main processing in Python:

1. All RGB-images were converted to jpg format.

2. All Images were resized to a new dimension (157 x 374) pixels.

In the paper, a Jupyter notebook was used to set the code and import the necessary libraries for the interactive code environment.

Matplotlib: to plot graphics

Numpy: a fundamental package for scientific computing.

Skimage: to import images from database environment files.

PIL and Scipy: to test the network with other images.

The main Processing required the following steps:

1. Vectorization: this step requires creating a feature vector after reading each RGB- colourful image in a Numpy array pattern to fit the algorithm, each RGB image was separated tinto hree colour channels red, green, and blue respectfully, as shown in [26, Fig. A-3], [26, Appendix A]. And because all images imensions were set to (157 x 374) pixels, the three channel matrices are set to (157 x 374) each. Therefore, to create a feature vector X with n-dimension, each pixel intensity value in the image cells must be unrolled or reshaped for each colour channel.

[23, eq. (2)], is used to reshape the new dimension for each image. Now we have a feature vector, or called an object for each traffic colour image, with total pixels (176154).

The technique of vectorization is very helpful when dealing not only with Traffic light images, but also with other signs and symbols images, as it considers all the features of images, representing an object vector, which can be easily understood by the computer, or any intelligent machines, it is faster compared to loops, considering the fact that Jupyter notebook is working on C.P.U

2. Normalization: It is a process to standardize object features for each RGB-colourful image, considering the maximum value for the pixel channel, see [26, eq. (3)]

$$X = \frac{Each\ Row\ in\ Object\ Vector}{255} \quad \ldots(3)$$

Where,

X: Represents the input for RGB image

3.3.3 BUILDING A BINARY CLASSIFIER USING DPBNN ALGORITHM

Back Propagation Neural Network (BPNN) is one of the most popular, and powerful artificial neural networks that can be used in deep learning

algorithms to reduce the time of training. It uses optimization algorithms to calculate the gradient of the function for each iteration, reducing the randomness in neural networks [31]. DBPNN was built in a two-stage model for each traffic light color, Red-DBPNN, Yellow-DBPNN, and Green-DBPNN.

A. Two-layers Model (SBPNN)

A two layer eural network was built with [1-5-1] dimensions in the first stage after passing image processing steps.
X is the input of a trafficlight colour image with (176154) pixels, learning rate is of .0075, and A number of iteration 3000, the network was trained to return parameters in Python cache, saving them to be used in the second stage of DBPNN. See Figure(A-4, Appendix A), [26 34, Appendix A]. It shows the architecture of the SBPPNN as a binary classifier for traffic light red colour, the same process was repeated for traffic light green and yellow colours.

B. Four - layers Model (DBPNN)

Four layers model of DBPNN was built in the second stage with [176154-1-7-5-1] dimensions, seven hidden units were used in the first hidden layer, five hidden units were used in the second hidden layer, as shown in Figure (A-5,Appendix A),[34, Appendix A], Parameter values, which represent the weights of the neural network, were selected as small as random values, too big values will lead hidden Unit values to increase, and the sigmoid function will be saturated, slowing the learning process.
$W[L]$:is the weight matrix for L- layer
$b[L]$:is the bias vector in the L^{th} layer
Both matrices were used in the post layer and the next layer of the network, where L represents the number of layers in DBPNN. In order to keep weight random values from changing through layers, a Numpy random seed function was used, it saves the states of randomness, and calling the function multiple times will result in the same random numbers.
For the output post layer and the next layer of the network, two types of activation functions were used:

1. Relu Function:

A rectified linear Unit that belong to non-linear activation function. With Relu view neurons are activated at a time, which makes training of DBPNN easier, and avoids slow learning that could lead the values of hidden units to reach zero [27,34].

$$F(x) = \frac{1}{1+e^{-x}} \quad \ldots..(4)$$

2. Sigmoid Function:

Is used for binary classification [28], where the predicted output or called the actual output $\hat{y}$ is in the following binary range:

$$0 \le \hat{y} \le 1$$

Where, $\hat{y}$ can be calculated according to the following equation with ith training iteration:

$$\hat{y}(i) = \sigma(W^T X^{(i)} + b) \quad \ldots \quad (5)$$

With sigmoid activation function equation (6):

$$\sigma(Z) = \frac{1}{1+e^{-Z(i)}} \quad \ldots \quad (6)$$

For error calculation, two important functions were calculated:

a. Loss Function:

It is a measurement of the discrepancy between the predictions or called the actual output $\hat{y}$(i) and the correct output or called the target $y(i)$, see [27,34 eq. (7)], it also depicts if the model failed to predict the required classification during training and testing the model.

$$L(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{2}(\hat{y}^{(i)} - y^{(i)})^2$$

$$L(\hat{y}^{(i)}, y^{(i)}) = -(y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

- If $y^{(i)} = 1$: $L(\hat{y}^{(i)}, y^{(i)}) = -\log(\hat{y}^{(i)})$ where $\log(\hat{y}^{(i)})$ and $\hat{y}^{(i)}$ should be close to 1
- If $y^{(i)} = 0$: $L(\hat{y}^{(i)}, y^{(i)}) = -\log(1 - \hat{y}^{(i)})$ where $\log(1 - \hat{y}^{(i)})$ and $\hat{y}^{(i)}$ should be close to 0

… (7)

Where,
$Y(i)$^: The predicated output
$Y(i)$: The desired output

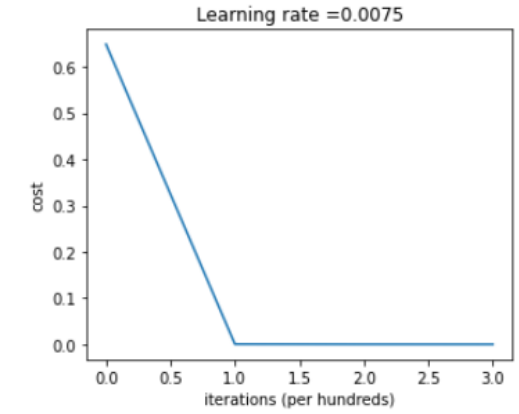
b. Cost Function:

Parameters training requires defining cost function, which is the average of loss function calculated as in [26,34, eq.(8)], of the complete training set as in [26, fig.(6)] show the calculated cost function for both red light-SBPPNN and red light-DBPNN respectfully.

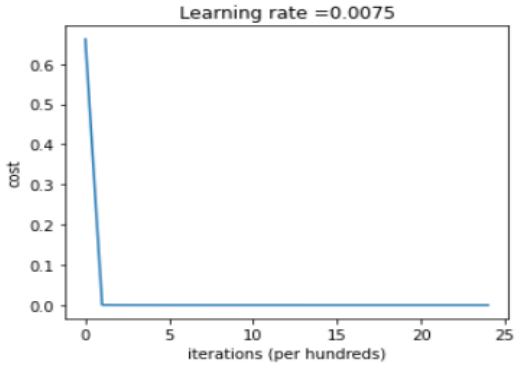
$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m} \left[(y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}) \right]$$

… (8)

Where, W: *the weighted values for the inputs*
b: *the biases values*
$Y(i)$^: The predicated output
$Y(i)$: The desired output



-a-



-b-

Fig. 6 Cost Functions for
a. Red Light-SBPNN and b.Red Light- DBPNN

The deep neural network algorithm's steps can be summarized as shown in Figure (7), from [34] the algorithm starts from initializing parameters to calculating activated output during forward propagation, and updating the parameters after passing the backward propagation, the training will continue for 2500-3000 iterations in order to reduce the cost, decreasing the loss rate between the predicted output by DBPN and the correct one.

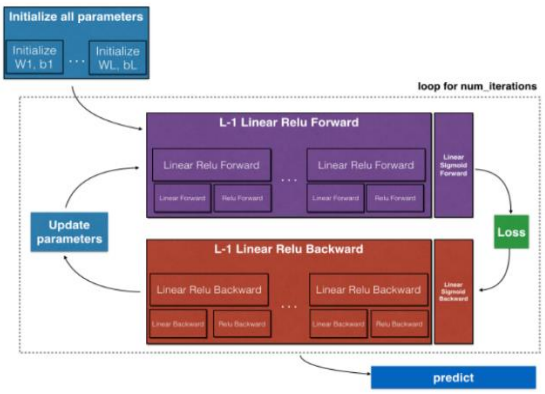


Fig. 7 Deep Neural Network Algorithm's Steps

4. MATCHING AND TESTING PROCESSES

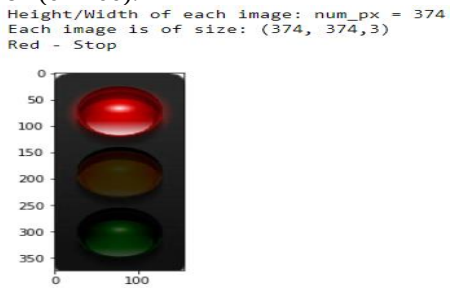
Since our DBPNN is working as a binary classifier, the matching process will depend on binary states describing the traffic light output, and the decision that must be taken into consideration for each state as shown in Table (5), which summarizes the matching and testing results. When one of the traffic ight colors turns on, the captured image will pass to red-DBPNN, yellow DBPNN, and green-DBPNN at the same time, if the tested image matches the defined traffic light color by DBPNN, the predicted output should be logic one, a text Message printed on screen will warn the driver for the next action to take, the intelligent machine should send a feedback signal, helping to draw the attention of the driver more on the road.

Table 5:Binary Mapping of Traffic Light Matching and Testing Processes

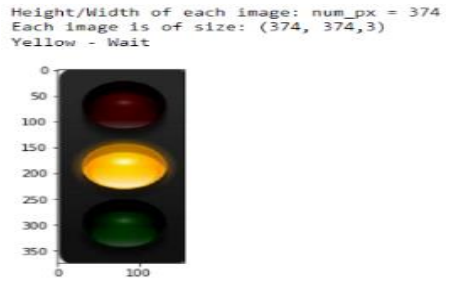
Red	Yellow	Green	True Output	Predicted Output State	Intelligent Machine Feedback Signal
0	0	0	0	All lights are off or it is a sign/symbol	Stop
0	0	1	1	Green is On	Go
0	1	0	1	Yellow is On	Wait
0	1	1	0	Yellow and Green are On	Ready
1	0	0	1	Red is On	Stop
1	0	1	0	Wrong Transition - damaged	Stop
1	1	0	0	Red and Yellow are On	Ready
1	1	1	0	All lights are on or it is a sign/symbol	Stop

To test the DBPNN, a test function was built with image processing steps to read other patterns of traffic light colors, signs, and symbols images, then predict the output to measure how accurate the network is in producing the desired output, which is consideres to be the right decision to take on the road by the intelligent machine. The matching steps were added to the same test function, to make a comparison between the defined images in the intelligent machine database and the one that was entered, two more steps were added to direct the intelligent machine on how to classify the unknown input images before giving the final recognition result. If the captured image represents the traffic red light, the output of Red-DBPNN should be logic one, while it is zero for both Yellow-DBPNN and Green-DBPNN, it means the red light is on, and the intelligent machine should warn the driver to stop the car, as shown in Figure (8-a), while Figure (8-b) shows an unmatched case of a captured image for the yellow traffic light, the tested output of Red-DBPNN and Green-DBPNN

is logic zero in that case, and one for Yellow-DBPNN, which means the yellow light is on, and the intelligent machine should warn the driver to get ready. Figure (9), shows the difference in cost function for Red-DBPNN between the defined images that represent red traffic light and unmatched state for a yellow color light from iteration (0-2400).

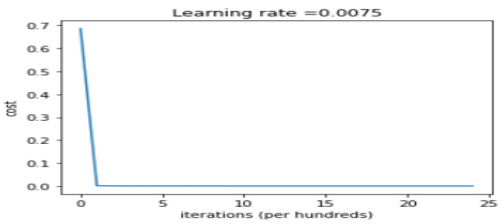


a. Matched state for Red-DBPNN

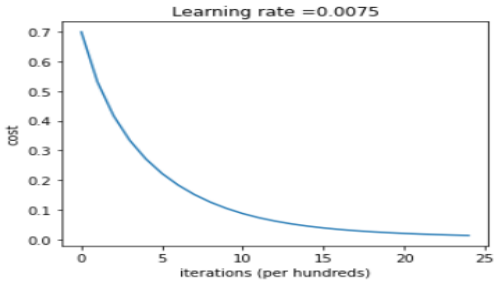


b. unmatched stated for Red-DBPNN

Fig. 8. Shows Different States for Red-DBPNN



-a-



-b-

Fig. 9 Cost Functions for the difference between the Red-DBPNN defined image and unmatched state for the yellow traffic image

The parameters were calculated in two stages:

1. Initialize Parameters, which represent the weights Matrices that were used in SBPNN.
2. Parameters for L-Layer Model, which represent the DBPNN and contains the following: (X,Y,Layers_dims,num_iterations=2500,Print_cost=true)

These parameters are used in two predictions functions:

1. Prediction function during training

Predictions_train (predict(X, Y, parameters)

2. Prediction function during the test

Predictions_test (X_test, Y_test, parameters)

Figure (A-1,Appendix-A), which shows the flowchart for the complete steps of classification, and recognition using three types of DBPNN, in order to reflect the state of Traffic light when turning on and off using a binary classifier.

5. RESULTS AND DISCUSSIONS

The results were arranged according to each DBPNN traffic light color, making decisions based on the predicted output, with true label values that reflect the correct output for the trained Color-DBPNN. The accuracy of DBPNN is 100%, a percentage of data that are correctly classified, and calculated as one in Python. The costs were taken briefly from iteration 0 to iteration 2400. To show the error rate value between the correct output in the database and the one that resulted according to one of the discussed cases before. Table (6) shows the proposed abbreviation for each image name, representing light colors. Table (7) represents the results of matching and testing Red-DBPNN, and how the intelligent machine sends the right feedback to the driver. The Yellow and Green DBPNN should be off in that case, reporting the binary state for the red light as 100, while the pedestrian and other signs were all set to (000) binary state. The required time between connecting to Jupyter Notebook and loading the program to apply the required parameters and calculate the cost function for each traffic light color is about one minute, while the matching process takes about 25 seconds, and it is the shortest time that could be needed to analyze the signs, symbols art, and traffic light's states down the road, then send a feedback Message to the driver, to inform him about the current state and enhance his decision to be more careful. See Figure(A-2, Appendix A).

Table 6:Images' Names and Abbreviations of Lights' Colours States

Image name	Abbreviation	Image name	Abbreviation	Image name	Abbreviation	Image name	Abbreviation
Red1	R1	Green1	G1	Yellow4	Y4	Lights On1	LO1
Red2	R2	Green2	G2	Yellow5	Y5	Lights On2	LO2
Red3	R3	Green3	G3	Yellow6	Y6	Lights On3	LO3
Red4	R4	Green4	G4	Yellow7	Y7	Lights On4	LO4
Red5	R5	Green5	G5	Yellow8	Y8	Lights On5	LO5
Red6	R6	Green6	G6	Yellow9	Y9	Traffic light Sign1	TLS1
Red7	R7	Green7	G7	Lights Off1	LOFF1	Traffic light Sign2	TLS2
Red-Green	RG	Green8	G8	Lights Off2	LOFF2	Traffic light Sign3	TLS3
Red-Yellow1	RY1	Yellow1	Y1	Lights Off3	LOFF3	Stop Sign1	SS1
Red-Yellow2	RY2	Yellow2	Y2	Lights Off4	LOFF4	Pedestrian Sign1	PS1
Red- Yellow3	RY3	Yellow3	Y3	Lights Off5	LOFF5	Pedestrian Sign2	PS2
-	-	-	-	-	-	Pedestrian Sign3	PS3
-	-	-	-	-	-	Pedestrian Sign4	PS4

Table 7:Binary Matching and Testing Results for Red-DBPNN

Light State	Binary Red Light	Binary State Input Image	True Label	Cost after iteration 0	Cost after iteration 2400	Predicted Output after Testing	Intelligent Machine Decision
R1	100	100	1	0.684525	0.000014	1.0	Stop
R2	100	100	1	0.693147	0.059820	1.0	Stop
R3	100	100	1	0.686982	0.000026	1.0	Stop
R4	100	100	1	0.662098	0.000012	1.0	Stop
R5	100	100	1	0.671969	0.000015	1.0	Stop
R6	100	100	1	0.669485	0.000012	1.0	Stop
R7	100	100	1	0.62988	0.000009	1.0	Stop
RG	100	101	0	0.722893	0.014257	0.0	Wrong Transition
RY1	100	110	0	0.722460	0.014260	0.0	Ready
RY2	100	110	0	0.720945	0.014271	0.0	Ready
RY3	100	110	0	0.732578	0.014183	0.0	Ready
G1	100	001	0	0.703466	0.014405	0.0	Go
G2	100	001	0	0.697305	0.014452	0.0	Go
G3	100	001	0	0.713591	0.014327	0.0	Go
G4	100	001	0	0.712953	0.014332	0.0	Go
G5	100	001	0	0.717389	0.014298	0.0	Go
G6	100	001	0	0.729773	0.014204	0.0	Go
G7	100	001	0	0.705261	0.014391	0.0	Go
G8	100	001	0	0.720897	0.014272	0.0	Go
Y1	100	010	0	0.695808	0.014463	0.0	Wait
Y2	100	010	0	0.699347	0.014436	0.0	Wait
Y3	100	010	0	0.711316	0.014345	0.0	Wait
Y4	100	010	0	0.708323	0.014368	0.0	Wait
Y5	100	010	0	0.742934	0.014104	0.0	Wait
Y6	100	010	0	0.735910	0.014158	0.0	Wait
Y7	100	010	0	0.712006	0.014340	0.0	Wait
Y8	100	010	0	0.726071	0.014232	0.0	Wait
Y9	100	010	0	0.735831	0.014158	0.0	Wait
LOFF1	100	000	0	0.744688	0.014091	0.0	Stop
LOFF2	100	000	0	0.739512	0.014130	0.0	Stop
LOFF3	100	000	0	0.693147	0.059820	0.0	Stop
LOFF4	100	000	0	0.722053	0.014263	0.0	Stop
LOFF5	100	000	0	0.700890	0.014424	0.0	Stop
LO1	100	111	0	0.735399	0.014162	0.0	Stop
LO2	100	111	0	0.734127	0.014171	0.0	Stop
LO3	100	111	0	0.711541	0.014343	0.0	Stop
LO4	100	111	0	0.721760	0.014265	0.0	Stop
LO5	100	111	0	0.727689	0.014220	0.0	Stop
TLS1	100	000	0	0.728816	0.014211	0.0	Stop

TLS2	100	000	0	0.730222	0.014201	0.0	Stop
TLS3	100	000	0	0.752483	0.014032	0.0	Stop
SS1	100	000	0	0.728323	0.014215	0.0	Stop
PS1	100	000	0	0.750809	0.014045	0.0	Stop
PS2	100	000	0	0.739059	0.014134	0.0	Stop
PS3	100	000	0	0.735876	0.014158	0.0	Stop
PS4	100	000	0	0.744767	0.014091	0.0	Stop

Table 8:Binary Matching and Testing Results for Yellow-DBPNN

Light State	Binary Yellow Light	Binary State Input Image	True Label	Cost after iteration 0	Cost after iteration 2400	Predicted Output after Testing	Intelligent Machine Decision
Y1	010	010	1	0.690493	0.000009	1.0	Wait
Y2	010	010	1	0.686986	0.000016	1.0	Wait
Y3	010	010	1	0.675302	0.000018	1.0	Wait
Y4	010	010	1	0.678198	0.000011	1.0	Wait
Y5	010	010	1	0.645722	0.000007	1.0	Wait
Y6	010	010	1	0.652138	0.000010	1.0	Wait
Y7	010	010	1	0.674638	0.000013	1.0	Wait
Y8	010	010	1	0.661273	0.000012	1.0	Wait
Y9	010	010	1	0.652211	0.000009	1.0	Wait
R1	010	100	0	0.701844	0.014417	0.0	Stop
R2	010	100	0	0.693147	0.059820	0.0	Stop
R3	010	100	0	0.699351	0.014436	0.0	Stop
R4	010	100	0	0.725191	0.014239	0.0	Stop
R5	010	100	0	0.714783	0.014318	0.0	Stop
R6	010	100	0	0.717382	0.014298	0.0	Stop
R7	010	100	0	0.724244	0.014246	0.0	Stop
G1	010	001	0	0.703466	0.014405	0.0	Stop
G2	010	001	0	0.697305	0.014452	0.0	Stop
G3	010	001	0	0.713591	0.014327	0.0	Stop
G4	010	001	0	0.712953	0.014332	0.0	Stop
G5	010	001	0	0.717389	0.014298	0.0	Stop
G6	010	001	0	0.729773	0.014204	0.0	Stop
G7	010	001	0	0.705261	0.014391	0.0	Stop
G8	010	001	0	0.720897	0.014272	0.0	Stop
G9	010	001	0	0.723350	0.014253	0.0	Stop
RG	010	101	0	0.722893	0.014257	0.0	Wrong Transition
RY1	010	110	0	0.722460	0.014260	0.0	Ready
RY2	010	110	0	0.720945	0.014271	0.0	Ready
RY3	010	110	0	0.732578	0.014183	0.0	Ready
LOff1	010	000	0	0.744688	0.014091	0.0	Stop
LOff2	010	000	0	0.739512	0.014130	0.0	Stop
LOff3	010	000	0	0.693147	0.059820	0.0	Stop
LOff4	010	000	0	0.722053	0.014263	0.0	Stop
LOff5	010	000	0	0.700890	0.014424	0.0	Stop
LOn1	010	000	0	0.735399	0.014162	0.0	Stop
LOn2	010	111	0	0.734127	0.014171	0.0	Stop
LOn3	010	111	0	0.711541	0.014343	0.0	Stop
LOn4	010	111	0	0.703844	0.014402	0.0	Stop
LOn5	010	111	0	0.712778	0.014334	0.0	Stop
TLS1	010	111	0	0.728816	0.014211	0.0	Stop
TLS2	010	000	0	0.730222	0.014201	0.0	Stop
TLS3	010	000	0	0.752483	0.014032	0.0	Stop
SS1	010	000	0	0.728323	0.014215	0.0	Stop
PS1	010	000	0	0.75080	0.014045	0.0	Stop
PS2	010	000	0	0.739059	0.014134	0.0	Stop
PS3	010	000	0	0.735876	0.014158	0.0	Stop
PS4	010	000	0	0.744767	0.014091	0.0	Stop

Table (8) shows the matching and testing results for Yellow-DBPNN, where Red-DBPNN and Green-DBPNN should be off, reporting the binary state for yellow light as 010, while Table (9) shows the matching and testing results for Green-DBPNN, where Red-DBPNN and Yellow-DBPNN should be off, reporting the binary state for green light as 001.

Table 9:Binary Matching and Testing Results for Green-DBPNN

Light State	Binary Green Light	Binary State for the Input Image	True Label	Cost after iteration 0	Cost after iteration 2400	Predicted Output after Testing	Intelligent Machine Decision
G1	001	001	1	0.682934	0.000017	1.0	Go
G2	001	001	1	0.689007	0.000030	1.0	Go
G3	001	001	1	0.673113	0.000009	1.0	Go
G4	001	001	1	0.673726	0.000012	1.0	Go
G5	001	001	1	0.669479	0.000010	1.0	Go
G6	001	001	1	0.657816	0.000009	1.0	Go
G7	001	001	1	0.681178	0.000013	1.0	Go
G8	001	001	1	0.666147	0.000012	1.0	Go
G9	001	001	1	0.663830	0.00009	1.0	Go
R1	001	100	0	0.701844	0.014417	0.0	Go
R2	001	100	0	0.693147	0.059820	0.0	Stop
R3	001	100	0	0.699351	0.014436	0.0	Stop
R4	001	100	0	0.725191	0.014239	0.0	Stop
R5	001	100	0	0.714783	0.014318	0.0	Stop
R6	001	100	0	0.717382	0.014298	0.0	Stop
R7	001	100	0	0.724244	0.014246	0.0	Stop
Y1	001	010	0	0.695808	0.014463	0.0	Wait
Y2	001	010	0	0.699347	0.014436	0.0	Wait
Y3	001	010	0	0.711316	0.014345	0.0	Wait
Y4	001	010	0	0.708323	0.014368	0.0	Wait
Y5	001	010	0	0.742934	0.014104	0.0	Wait
Y6	001	010	0	0.735910	0.014158	0.0	Wait
Y7	001	010	0	0.712006	0.014340	0.0	Wait
Y8	001	010	0	0.726071	0.014232	0.0	Wait
Y9	001	010	0	0.735831	0.014158	0.0	Wait
RG	001	101	0	0.722893	0.014257	0.0	Wrong Transition
RY1	001	110	0	0.722460	0.014257	0.0	Ready
RY2	001	110	0	0.720945	0.014271	0.0	Ready
RY3	001	110	0	0.732578	0.014183	0.0	Ready
LOff1	001	000	0	0.746488	0.014091	0.0	Stop
LOff2	001	000	0	0.739512	0.014091	0.0	Stop
LOff3	001	000	0	0.693147	0.059820	0.0	Stop
LOff4	001	000	0	0.722053	0.014263	0.0	Stop
LOff5	001	000	0	0.700890	0.014424	0.0	Stop
LOn1	001	000	0	0.696925	0.014455	0.0	Stop
LOn2	001	111	0	0.734127	0.014171	0.0	Stop
LOn3	001	111	0	0.711541	0.014343	0.0	Stop
LOn4	001	111	0	0.703844	0.014402	0.0	Stop
LOn5	001	111	0	0.712778	0.014334	0.0	Stop
TLS1	001	111	0	0.728816	0.014211	0.0	Stop
TLS2	001	000	0	0.730222	0.014201	0.0	Stop
TLS3	001	000	0	0.752483	0.014032	0.0	Stop
SS1	001	000	0	0.728323	0.014215	0.0	Stop
PS1	001	000	0	0.750809	0.014045	0.0	Stop
PS2	001	000	0	0.739059	0.014134	0.0	Stop
PS3	001	000	0	0.735876	0.014158	0.0	Stop
PS4	001	000	0	0.744767	0.014091	0.0	Stop

The results shown in Tables (7, 8, 9) were calculated according to Jupyter Notebook using the following Python instrtuctions or the final test operation:

Parameters = L_layer_model(X, Y, layers_dims, num_iterations = 2500, print_cost = True)

With parameters passing to the cost function, the cost will be calculated for 2500 iterations, while the predict function will use the current input (traffic light image), with the assigned output, and the parameters to find the predicted output **predic_test = predict (X_test, Y_test, parameters)**

6. CONCLUSION

In this paper, I have investigated the importance of deep learning in developing the performance of shallow or standard neural networks, in particular Back Propagation Neural Network (BPNN). Most of the paper's research focused on using deep CNN in order to satisfy the recognition process.

The comparison of the proposed system with other papers is based on measuring accuracy, and cost function reduction with lost values between what we already have from defined cases or scenarios in the database, and what the driver may account really on the road.

There are many techniques that help to minimize the effects of driver distraction, but still, the number of hazards on the road is very high.

Neural networks were applied on the road infrastructure to alert drivers about any potential hazards, Germany is one of those countries, which uses self-learning system with radar, sensors, and cameras to pick out moving objects on the road, and the alert will be sent to drivers in car warning displays or using streets lights[35,36]. In Iraq, we don't have such techniques, so we could have a smart system in cars with deep learning neural network, making use of cameras to pick up Traffic changes, and signs and update them every few seconds, comparing their states whether they were matching the defined cases in the database, so we are focusing on the driver to keep attention.

The evolution in Python was powerful in satisfying image processing, and normalizing all the images to fit in the database. Besides designing a binary matching system to compare the defined image in (DBPNN) according to the trained color Yellow-DBPNN and Green-DBPNN, and the input image, which reflected different cases under unexpected circumstances that might take place in now days.

The results are promising though the difficulties in collecting images, and processing them to be in one format and shape to avoid any difference in extracting features; furthermore, the difficulty in searching and finding new concepts among many researches to design traffic light classification and recognition system using deep neural network, a new concept was proposed to design a model of two stages, first stage was represented by building SBPNN for each traffic light color, and connected it to the second stage, which was represented by building DBPNN for each color as was mentioned before. The results showed how successful as the model in predicting the on/off state for each traffic light color, directing the intelligent machine to send the right feedback to the driver, using the proposed binary matching system, though the evolution in Python as not easy, especially in designing the match and test processes, it took time before satisfying the required results, the purpose of the feedback signals were to enhance the awareness of the driver more on the road in order to keep safety. This system can be developed in the future, to add certain hardware that uses audio voices as warning Messages, it will not enhances the awareness of the driver only but helps to understand how to deal with unexpected situations related to traffic light work. There is no 100% of any system that could function properly without getting affected by the bad weather, or sudden electrical shutdown that may affect the whole system.

7. REFERENCES

[1] L. C. Possatti *et al.*, "Traffic Light Recognition Using Deep Learning and Prior Maps for Autonomous Cars," *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019, doi:https://doi.org/10.1109/IJCNN.2019.8851927.

[2] [1] Y. Zhu and W. Q. Yan, "Traffic sign recognition based on Deep Learning," Multimedia Tools and Applications, vol. 81, no. 13, pp. 17779–17791, 2022. doi:10.1007/s11042-022-12163-0

[3] M. Y. Taha, "Evaluation of the Acceptance of the Hot Mix Asphalt Paving Mixture Using Backpropagation Artificial Neural Network," *AL-Rafdain Engineering Journal (AREJ)*, vol. 19, no. 2, pp. 40–54, Apr. 2011, doi: https://doi.org/10.33899/rengj.2011.27341.

[4] M. A. Weber *et al.*, "DeepTLR: A single deep convolutional network for detection and classification of Traffic light ," Jun. 2016, doi: https://doi.org/10.1109/ivs.2016.7535408.

[5] Sahar Qaddoori *et al.*, "An Efficient Security Model for Industrial Internet of Things (IIoT) System Based on Machine Learning Principles," vol. 28, no. 1, pp. 329–340, Mar. 2023, doi:https://doi.org/10.33899/rengj.2022.134932.1191.

[6] Y. Gu *et al.*, "A Novel Lightweight Real-Time Traffic Sign Detection Integration Framework Based on YOLOv4," *Entropy*, vol. 24, no. 4, p. 487, Mar. 2022, doi: https://doi.org/10.3390/e24040487.

[7] A. Navarro-Espinoza *et al.*, "Traffic Flow Prediction for Smart Traffic light Using Machine Learning Algorithms," *Technologies*, vol. 10, no.

1, p. 5, Jan. 2022, doi:https://doi.org/10.3390/technologies10010005.

[8] Q. Wang *et al.*, "Traffic light Detection and Recognition Method Based on the Improved YOLOv4 Algorithm," *Sensors*, vol. 22, no. 1, p. 200, Dec. 2021, doi: https://doi.org/10.3390/s22010200.

[9] Z. Li *et al.*, "An improved Traffic light recognition algorithm for autonomous driving in complex scenarios," vol. 17, no. 5, p. 155014772110183-155014772110183, May 2021, doi: https://doi.org/10.1177/15501477211018374.

[10] V. T. John *et al.*, "Traffic light recognition in varying illumination using deep learning and saliency map," Oct. 2014, doi: https://doi.org/10.1109/itsc.2014.6958056.

[11] R. F. Berriel *et al.*, "Deep Learning-Based Large-Scale Automatic Satellite Crosswalk Classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 9, pp. 1513–1517, Sep. 2017, doi: https://doi.org/10.1109/lgrs.2017.2719863.

[12] V. Usha *et al.*,"Traffic Sign Classification Using Deep Learning," *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, vol. 12, no. 9, pp. 250–253, Apr. 2021, doi: https://doi.org/10.17762/turcomat.v12i9.3007

[13] M. Bichkar *et al.*, "Traffic Sign Classification and Detection of Indian Traffic Signs using Deep Learning," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, pp. 215–219, May 2021, doi: https://doi.org/10.32628/cseit217325.

[14] W. Farag, "Traffic signs classification by deep learning for advanced driving assistance systems," *Intelligent Decision Technologies*, vol. 13, no. 3, pp. 305–314, Sep. 2019, doi: https://doi.org/10.3233/idt-180064.

[15] Á. Arcos-García *et al.*,"Evaluation of deep neural networks for traffic sign detection systems," *Neurocomputing*, vol. 316, pp. 332–344, Nov. 2018, doi: https://doi.org/10.1016/j.neucom.2018.08.009

[16] A. Fadhil *et al.*,"Real-Time Signature Recognition Using Neural Network," *Al-Rafidain Engineering Journal (AREJ)*, vol. 26, no. 1, pp. 159–165, Jan. 2021, doi:https://doi.org/10.33899/rengj.2021.129871.1088

[17] J. Zhang *et al.*, "Lightweight deep network for traffic sign classification," *Annals of Telecommunications*, vol. 75, no. 7–8, pp. 369–379, Jul. 2019, doi: https://doi.org/10.1007/s12243-019-00731-9

[18] A. Wong *et al.*,"MicronNet: A Highly Compact Deep Convolutional Neural Network Architecture for Real-Time Embedded Traffic Sign Classification," *IEEE Access*, vol. 6, pp. 59803–59810, 2018, doi: https://doi.org/10.1109/access.2018.2873948

[19] H. M. Elhawary *et al.*,"Investigation on the Effect of the Feature Extraction Backbone for Small Object Segmentation using Fully Convolutional Neural Network in Traffic Signs Application," *IOP Conference Series: Materials Science and Engineering*, vol. 1051, no. 1, p. 012006, Feb. 2021,https://doi.org/10.1088/1757-899x/1051/1/012006.

[20] Z. Zhong *et al.*,"Spectral–Spatial Residual Network for Hyperspectral Image Classification: A 3-D Deep Learning Framework," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 2, pp. 847–858, Feb. 2018, doi: https://doi.org/10.1109/tgrs.2017.2755542.

[21] Y. Liu *et al.*,"TSingNet: Scale-aware and context-rich feature learning for traffic sign detection and recognition in the wild," *Neurocomputing*, vol. 447, pp. 10–22, Aug. 2021, doi: https://doi.org/10.1016/j.neucom.2021.03.049.

[22] W. Farag, "Recognition of traffic signs by convolutional neural nets for self-driving vehicles," *International Journal of Knowledge-based and Intelligent Engineering Systems*, vol. 22, no. 3, pp. 205–214, Nov. 2018, doi: https://doi.org/10.3233/kes-180385.

[23] J. Credi, "Traffic sign classification with deep convolutional neural networks Master's thesis in Complex Adaptive Systems.", Dept. of Applied Mechanics, Chalmers University of Technology, Gothenburg,Sweden. https://odr.chalmers.se/server/api/core/bitstreams/fdef1142-92cb-4f8c-9c8a-f17f72260c00/content

[24] Á. Arcos-García *et al.*,"Deep neural network for traffic sign recognition systems: An analysis of spatial transformers and stochastic optimisation methods," *Neural Networks*, vol. 99, pp. 158–165, Mar. 2018, doi: https://doi.org/10.1016/j.neunet.2018.01.005.

[25] [1] M. Lin, Q. Chen, and S. Yan, "Network in Network," arXiv.org, https://arxiv.org/abs/1312.4400 (accessed Aug. 13, 2023).

[26] F. Shao *et al.*,"Real-Time Traffic Sign Detection and Recognition Method Based on Simplified Gabor Wavelets and CNNs," *Sensors*, vol. 18, no. 10, p. 3192, Sep. 2018, doi: https://doi.org/10.3390/s18103192.

[27] P. Kondamari *et al.*,"A Deep Learning Application for Traffic Sign Classification," Thesis, Blekinge

Institute of Technology, 371 79 Karlskrona,Sweden, 2021.

[28] T. Vijayakumaran, "Recognition of Traffic Signs using Deep Learning," Oct. 2020. Available: https://www.researchgate.net/profile/Tharmi-Vijayakumaran

[29] J. H. Shu *et al.*,"Network Traffic Classification Based on Deep Learning," *Journal of Physics: Conference Series*, vol. 1087, p. 062021, Sep. 2018, https://doi.org/10.1088/1742-596/1087/6/062021.

[30] "The Power of Deep Learning Models: Applications," *International Journal of Recent Technology and Engineering*, vol. 8, no. 2S11, pp. 3700–3705, Nov. 2019, doi:https://doi.org/10.35940/ijrte.b1468.0982s1119

[31] A. Mathew *et al.*, "Deep Learning Techniques: An Overview, " *Advances in Intelligent Systems and Computing*, pp. 599–608, May 2020, doi:https://doi.org/10.1007/978-981-15-3383-9_54.

[32] S. B. Wali *et al.*, "Vision-Based Traffic Sign Detection and Recognition Systems: Current Trends and Challenges," *Sensors*, vol. 19, no. 9, p. 2093, May 2019, https://doi.org/10.3390/s19092093.

[33] Sunitha. A. Shanthi, "Traffic Sign Classification and Detection Using Deep Learning," Apr. 2020. https://www.ijirt.org/master/publishedpaper/IJIRT149112_PAPER.pdf

[34] Andrew Ng," Deep Learning Specialization DeepLearning.AI," 2017.DeepLearning.AI: Start or Advance Your Career in AI.

[35] [1] D. Edwards and Polly, "June 22, 2020," Robotics & Automation News, https://roboticsandautomationnews.com/2020/06/22/.

[36] S. Soleymanpour *et al.*, "Traffic Classification using Deep Learning," *Ayandegan Institute of Higher Education*, Mar. 2020. doi:https://doi:10.13140/RG.2.2.33375.82087

تعزيز انتباه السائقين بواسطة آلة مطابقة ثنائية ذكية لتجنب الحوادث

أرجوان محمد عبد الجواد الجوادي
arjuwan_m@ntu.edu.iq

قسم هندسة تقنيات الحاسوب، الكلية التقنية الهندسية /الموصل، الجامعة التقنية الشمالية، الموصل، العراق

تاريخ الاستلام: 9 مارس 2023 استلم بصيغته المنقحة: 26 ابريل 2023 تاريخ القبول: 6 اغسطس 2023

الملخص

يسبب الإجهاد والمواقف الصعبة المفاجئة زيادة مخاطر الحوادث على الطرق. حيث قد يتشتت انتباه السائقين في ثوانٍ في ظل ظروف غير متوقعة ، والتي يمكن أن تحدث بسبب سوء الأحوال الجوية ، ومشاكل الرؤية ، والتعب نتيجة ساعات القيادة الطويلة ،بالأضافة الى وجود أشارات المرور التالفة أو المكسورة ، وحتى ضوضاء الأطفال ، والحركة داخل السيارة. في ورقة هذا البحث ، تم أقتراح تطوير شبكة عصبية خاصة بالانتشار العميقة لتعزيز انتباه السائقين من خلال مراقبة حالات إشارات المرور المختلفة باستخدام نظام ماكينة المطابقة الثنائية الذكية المقترحة في Python. سيقوم نظام الآلات الذكية بتحليل وتحديد إشارات المرور الحقيقية من وسائل الأيضاح فقط ، مكسورة أو تالفة ؛ بالإضافة إلى علامات المشاة بناءً على رموز قاعدة بيانات مقترحة لكل حالة ، قبل اتخاذ القرار الصحيح من قبل الشبكة المستفادة ، ثم أرسل إشارة معززة إلى السائق. تتألف الخوارزمية من خطوات دقيقة لمعالجة الصور ، مع مرحلتين للمعالجة المعقدة تتضمن استخلاص السمات المميزة لكل تفاصيل الصورة وخاصة الألوان لأعطاء التعريف الصحيح لأضواء وأشارات المرور المتعارف عليها ، يتم التعامل مع متجهات الاستخراج الكاملة من قبل الشبكات العصبية للون الأخضر والأصفر و الأحمر (SBPNN) و (DBPNN) لكل حالة معقدة. نتيجة لذلك ، صنفت الخوارزمية دقة عالية ، وهو العامل الأكثر أهمية للحفاظ على السلامة. لا يحل النظام المقترح محل قرار السائق ، ومع ذلك ، فهو نظام تعزيزي للتحذير و تنبيه السابق لأتخاذ القرار السليم قبل أن تصبح الأمور خارج نطاق السيطرة. يمكن تطوير إشارة التغذية المرتدة كرسالة تحذير للإشارة الصوتية ، لزياة وعي السائقين إلى جانب نص التحذير على الشاشة.

الكلمات الدالة:

الشبكات العصبية السطحية ، الشبكات العصبية العميقة التعلم ، نظام المطابقة الثنائية الذكية ، وتعزيز التمييز و التصنيف الاحتياطي ونظام التعرف.

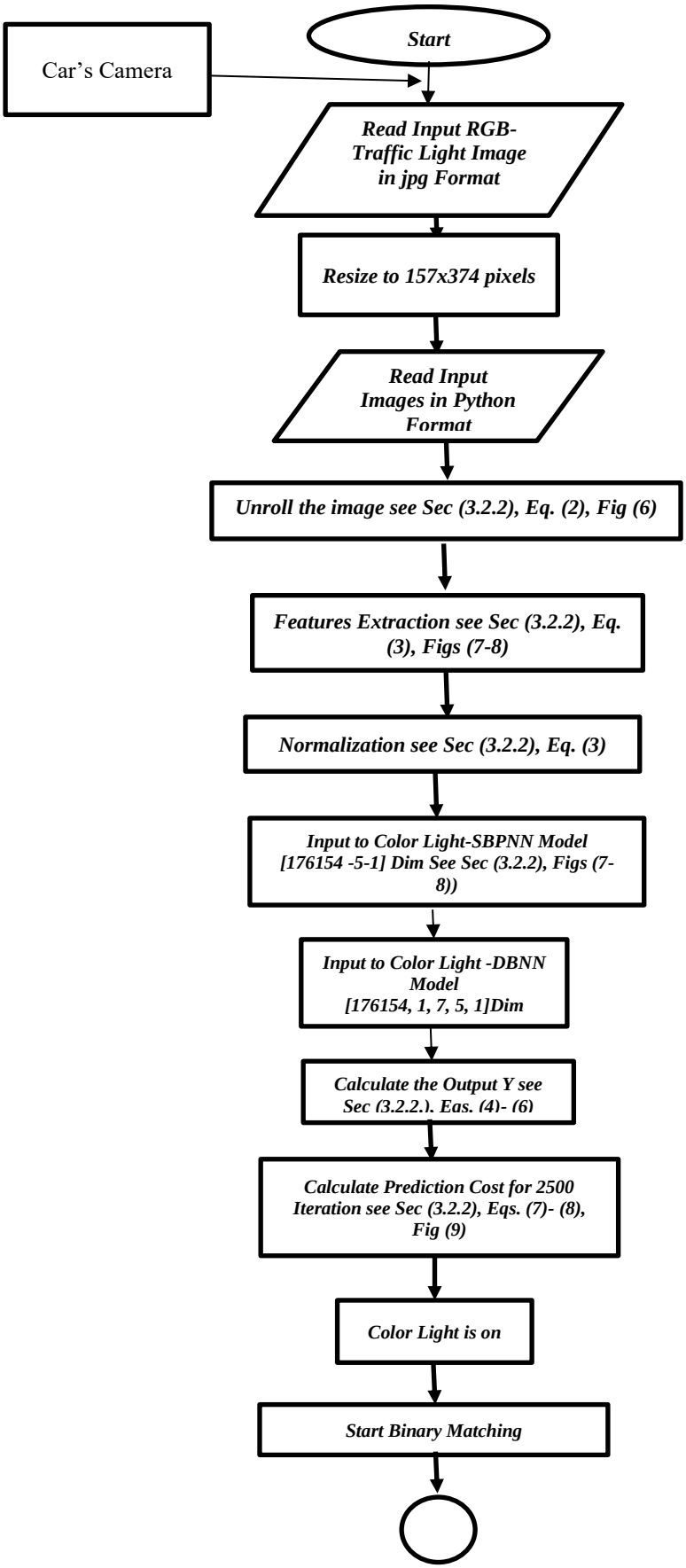
Appendix A
Computer Program

A.1 Introduction

A computer code, for the prediction of the correct output to make the right decision by an intelligent machine was presented in section 3.3, to satisfy the matching and testing results as it was proposed in Table(5) for Red-DBPNN, Yellow-DBPNN, and Green-DBPNN.
The computer program serves several stages, image processing, building SBPNN in the first stage, and DBPNN in the second stage with a binary tester at the end of the program to check the matching results, and combines them to the defined right actions.

A.2 Program Structure and Description of Subroutines

Python39 version is used in programming the prediction methods. The main flow chart of the programme is shown in Fig. A-1.and it represents the required steps to evaluate the DBPNN for each traffic light color.


Car's Camera
Start
Read Input RGB-Traffic Light Image in jpg Format
Resize to 157x374 pixels
Read Input Images in Python Format
Unroll the image see Sec (3.2.2), Eq. (2), Fig (6)
Features Extraction see Sec (3.2.2), Eq. (3), Figs (7-8)
Normalization see Sec (3.2.2), Eq. (3)
Input to Color Light-SBPNN Model [176154 -5-1] Dim See Sec (3.2.2), Figs (7-8))
Input to Color Light -DBNN Model [176154, 1, 7, 5, 1]Dim
Calculate the Output Y see Sec (3.2.2.), Eqs. (4)- (6)
Calculate Prediction Cost for 2500 Iteration see Sec (3.2.2), Eqs. (7)- (8), Fig (9)
Color Light is on
Start Binary Matching

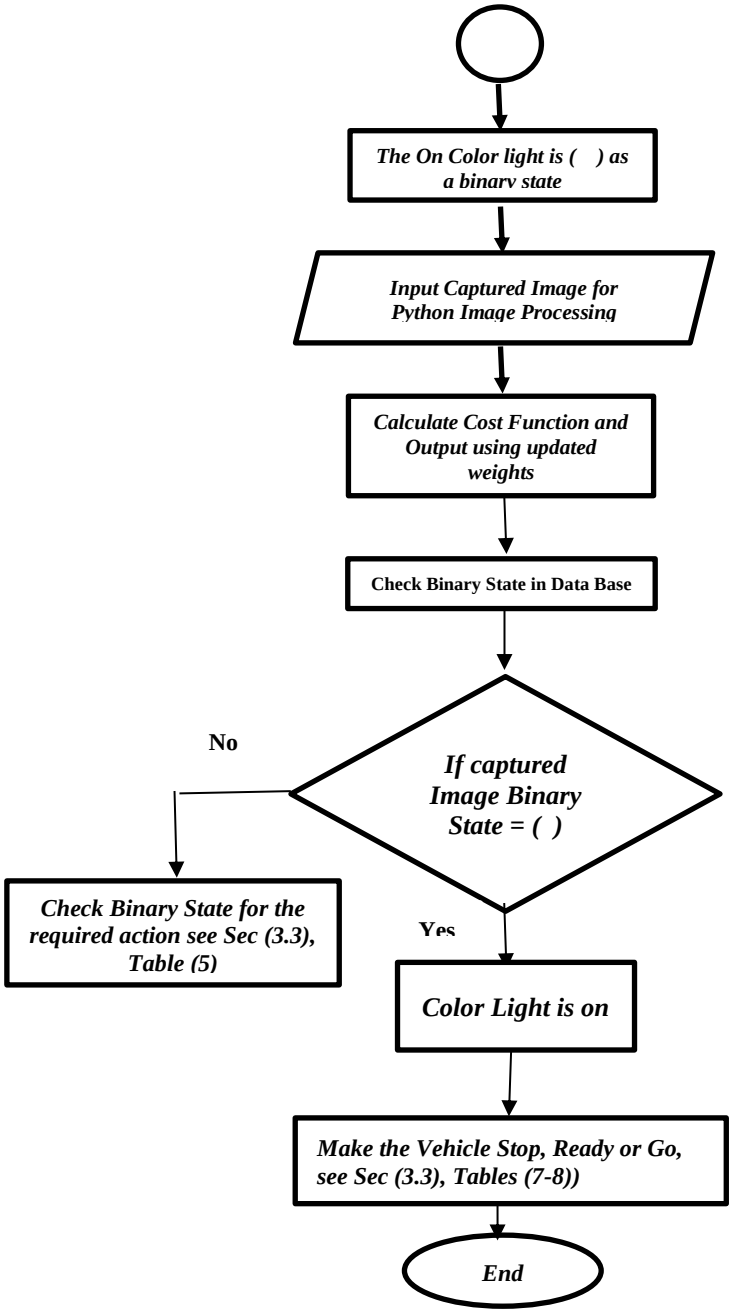
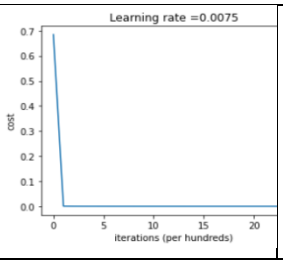
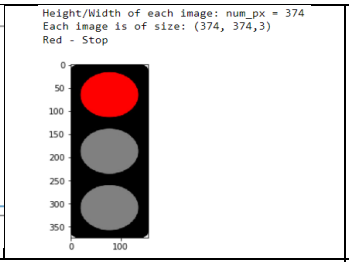


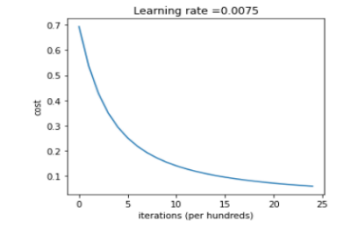
Fig. A-1. Main flow chart of the computer program used in this Paper research

Figure (A-2) shows the cost function from iteration 0 to iteration 2400 in Red-DBPNN, it was calculated to show the error value between the correct output, and the predicted one for each matched and tested traffic light color, and signs that reflect other information on the road. The same database of images were used in matching and testing Yellow-DBPNN and Green-DBPNN.

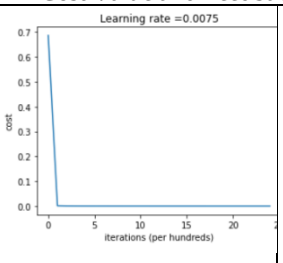


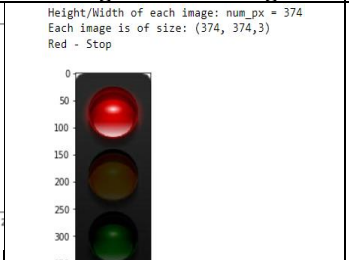


Cost Value and Feedback Message for Red1 Image

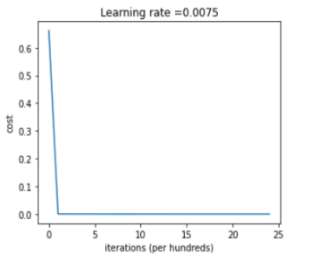


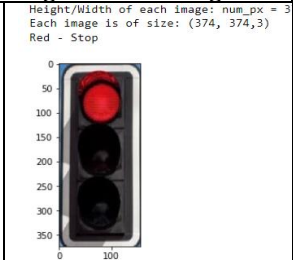
Cost Value and Feedback Message for Red2 Image



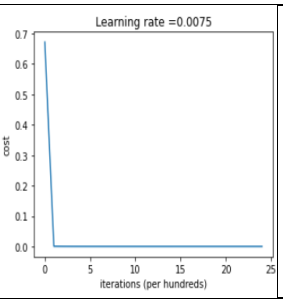


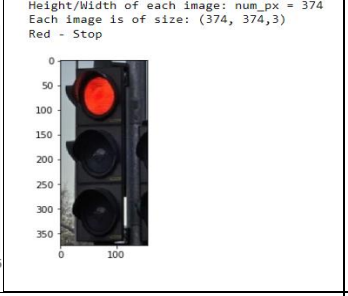
Cost Value and Feedback Message for Red3 Image



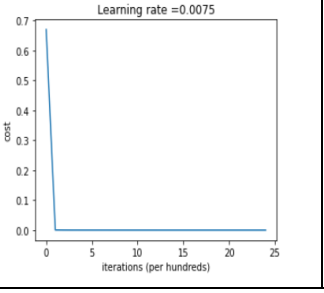


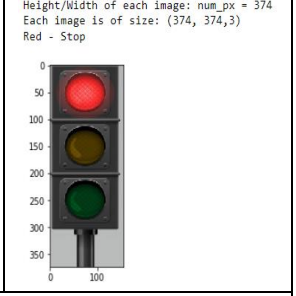
Cost Value and Feedback Message for Red4 Image



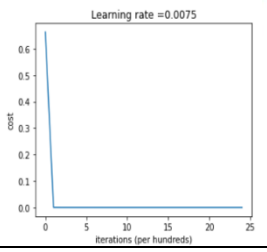


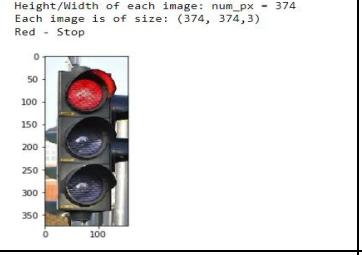
Cost Value and Feedback Message for Red5 Image



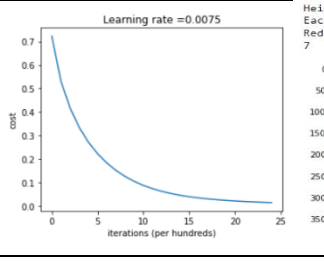


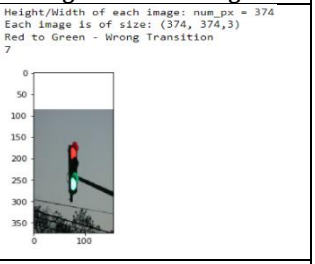
Cost Value and Feedback Message for Red6 Image



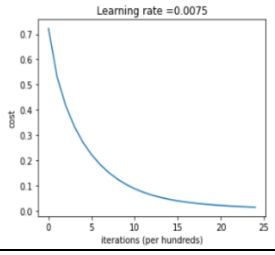


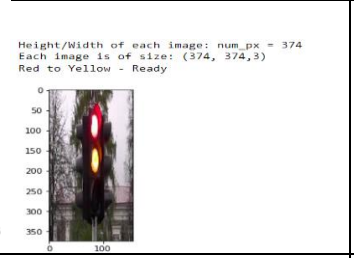
Cost Value and Feedback Message for Red7 Image



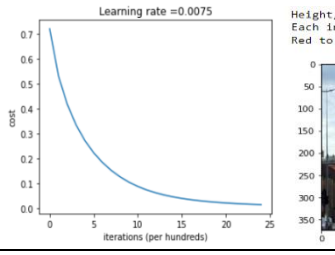


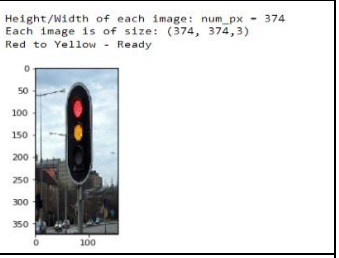
Cost Value and Feedback Message for Red and Green Ima ge



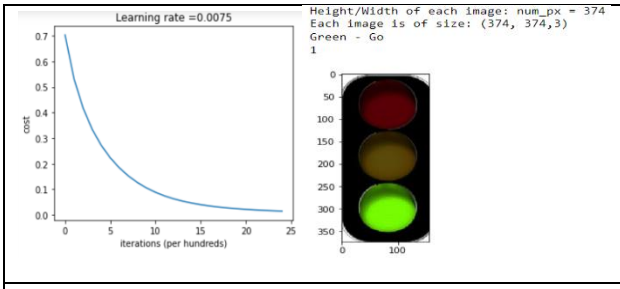


Cost Value and Feedback Message for Red and Yellow1 I mage

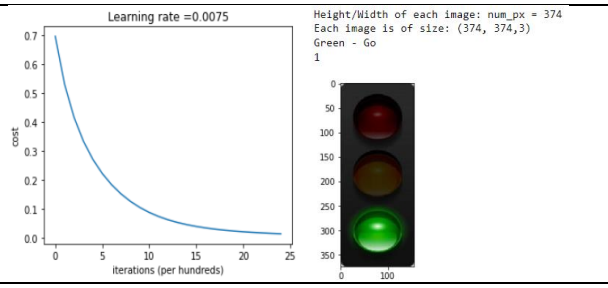




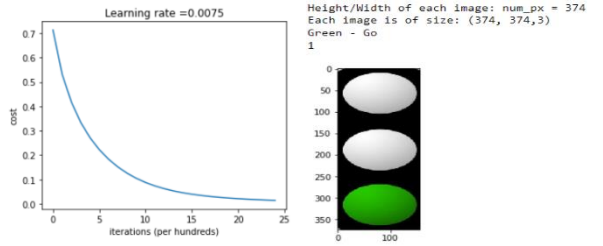
Cost Value and Feedback Message for Red and Yellow2 Im age



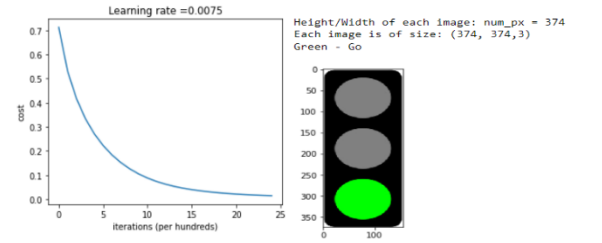
Cost Value and Feedback Message for Green1 Image



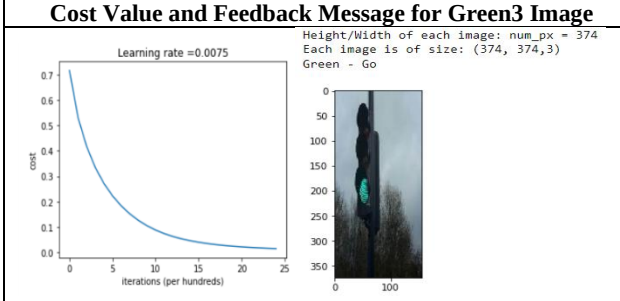
Cost Value and Feedback Message for Green2 Image



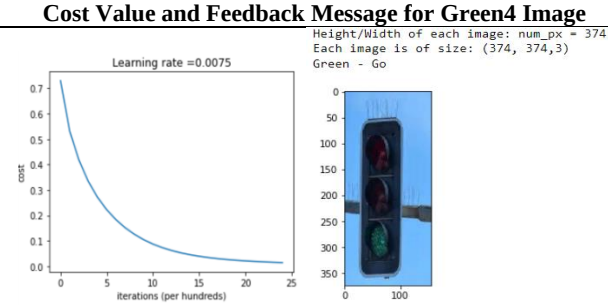
Cost Value and Feedback Message for Green3 Image



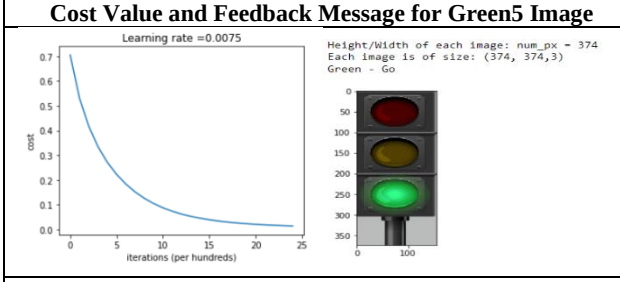
Cost Value and Feedback Message for Green4 Image



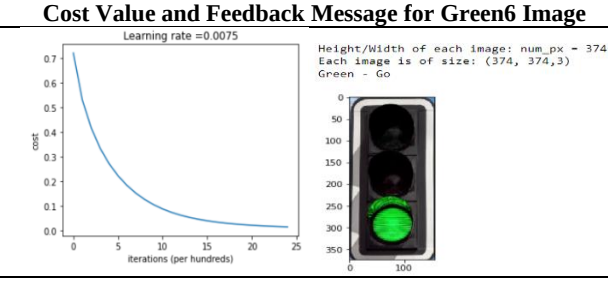
Cost Value and Feedback Message for Green5 Image



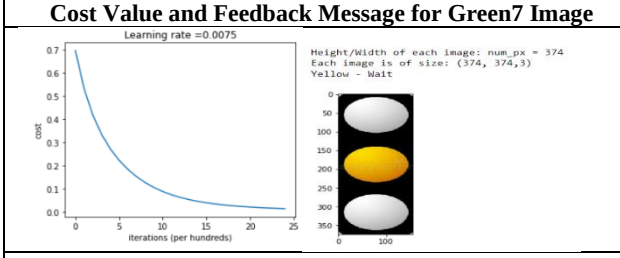
Cost Value and Feedback Message for Green6 Image



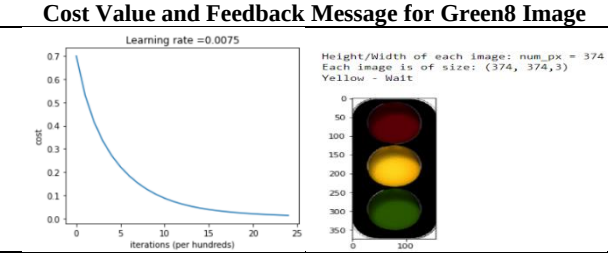
Cost Value and Feedback Message for Green7 Image



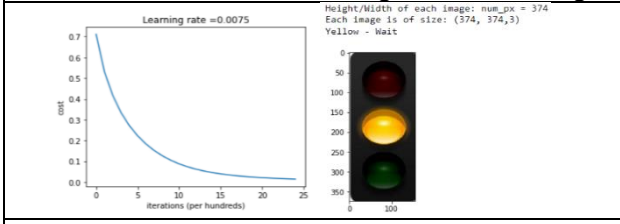
Cost Value and Feedback Message for Green8 Image



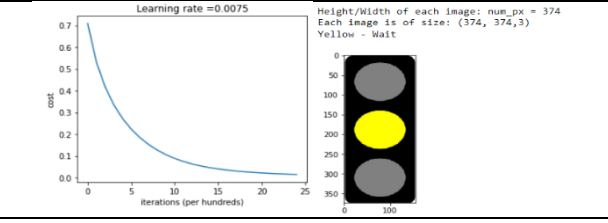
Cost Value and Feedback Message for Yellow1 Image



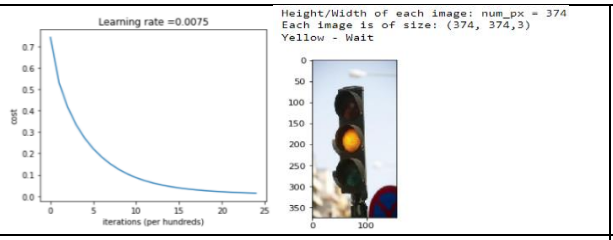
Cost Value and Feedback Message for Yellow2 Image



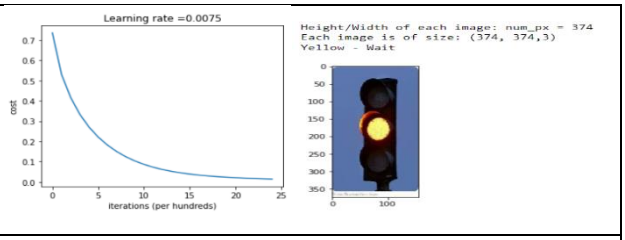
Cost Value and Feedback Message for Yellow3 Image



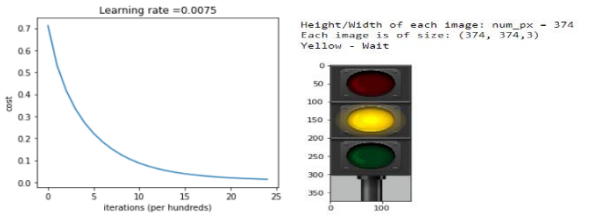
Cost Value and Feedback Message for Yellow4 Image



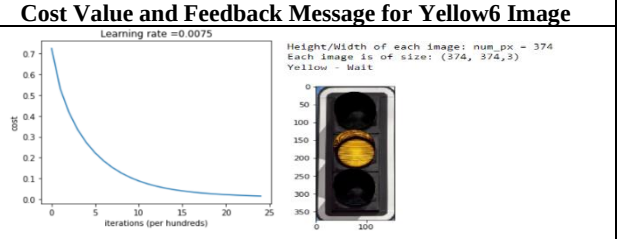
Cost Value and Feedback Message for Yellow5 Image



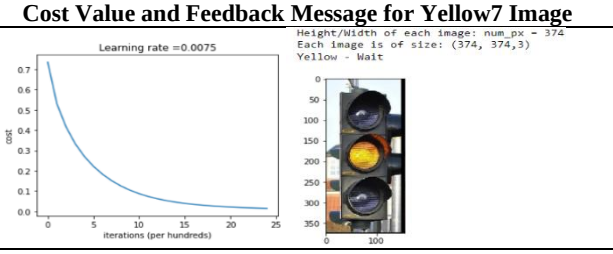
Cost Value and Feedback Message for Yellow6 Image



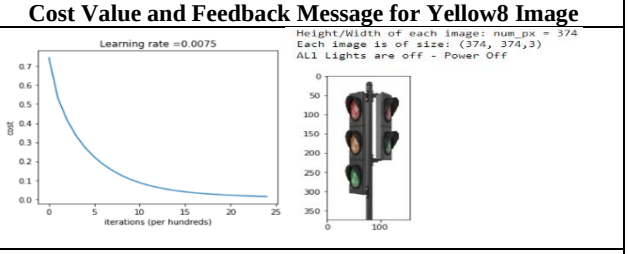
Cost Value and Feedback Message for Yellow7 Image



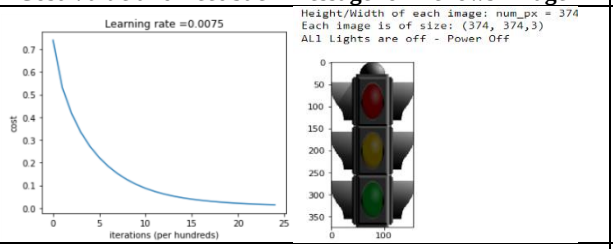
Cost Value and Feedback Message for Yellow8 Image



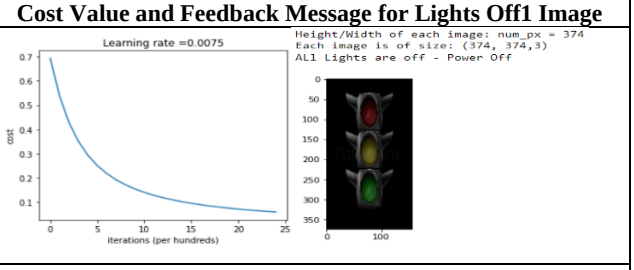
Cost Value and Feedback Message for Yellow9 Image



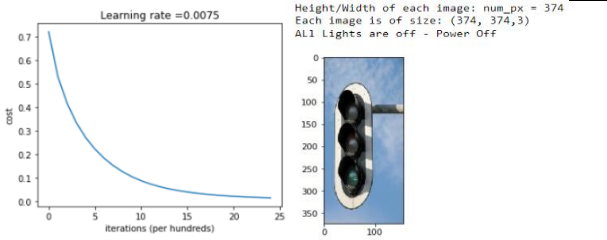
Cost Value and Feedback Message for Lights Off1 Image



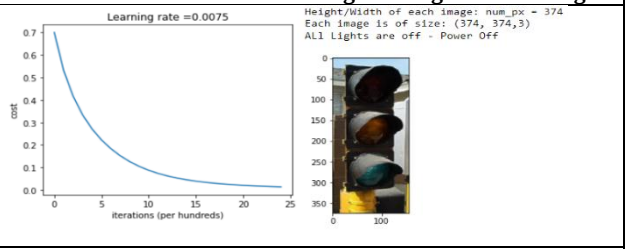
Cost Value and Feedback Message for Lights Off2 Image



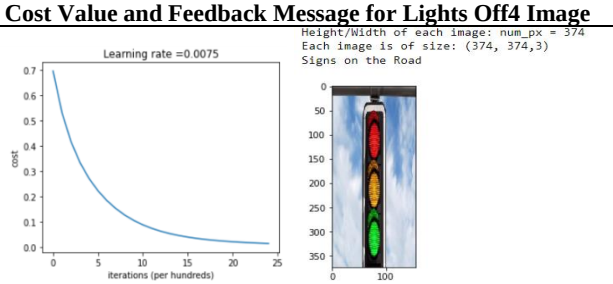
Cost Value and Feedback Message for Lights Off3 Image



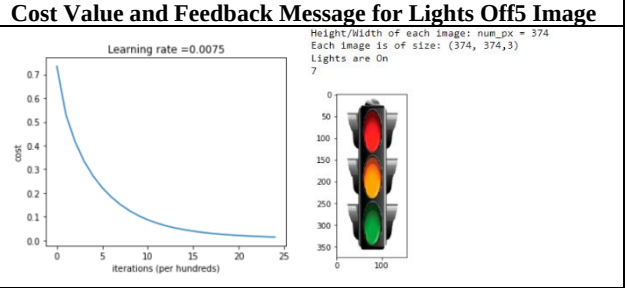
Cost Value and Feedback Message for Lights Off4 Image



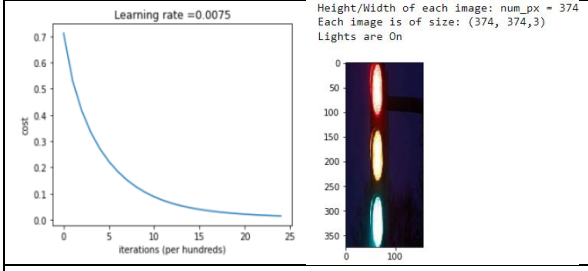
Cost Value and Feedback Message for Lights Off5 Image



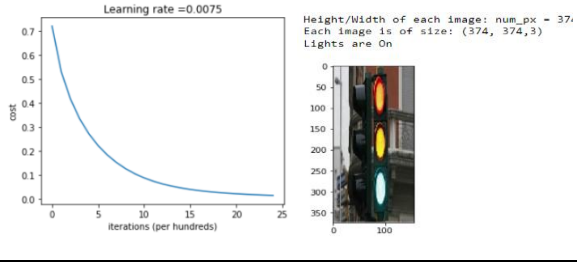
Cost Value and Feedback Message for Lights On1 Image



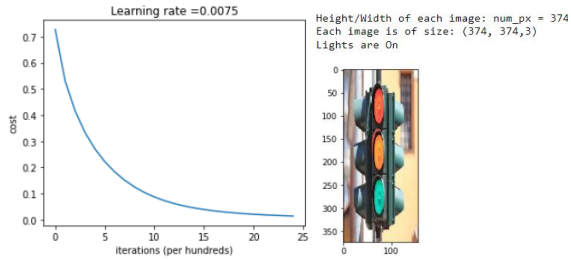
Cost Value and Feedback Message for Lights On2 Image



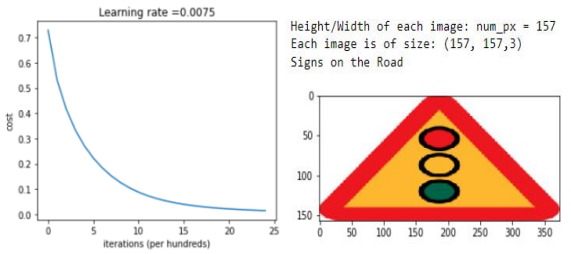
Cost Value and Feedback Message for Lights On2 Image



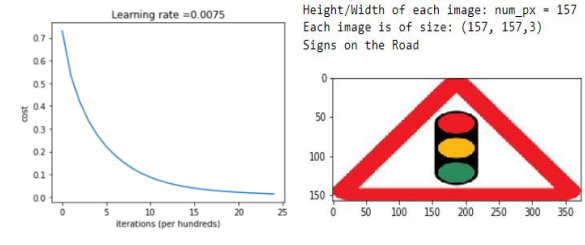
Cost Value and Feedback Message for Lights On3 Image



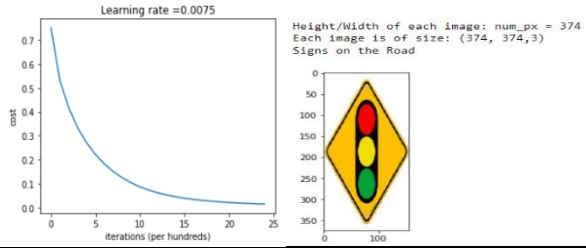
Cost Value and Feedback Message for Lights On4 Image



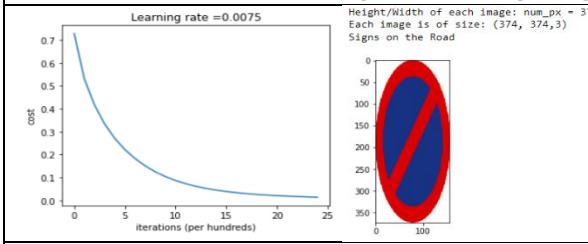
Cost Value and Feedback Message for Traffic light Sign1



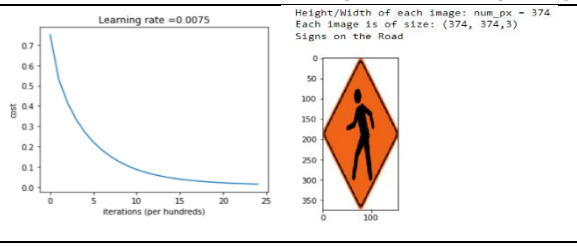
Cost Value and Feedback Message for Traffic light Sign2



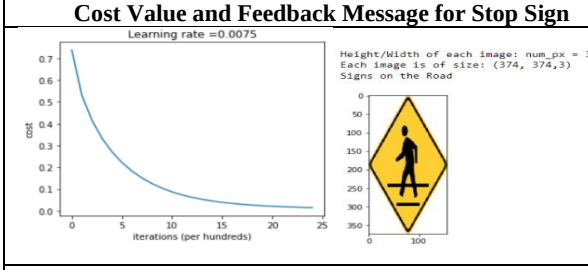
Cost Value and Feedback Message for Traffic light Sign3



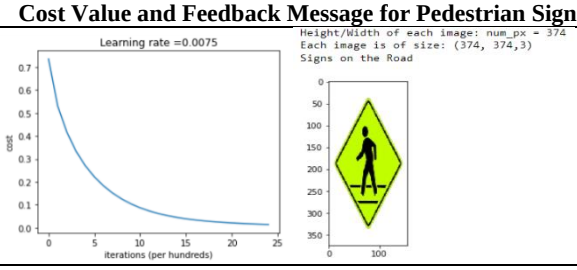
Cost Value and Feedback Message for Stop Sign



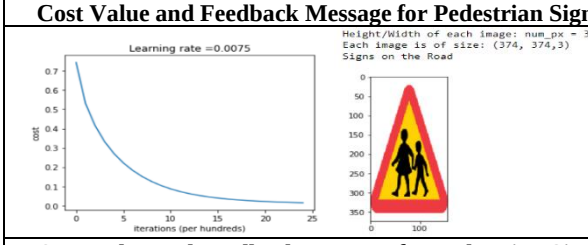
Cost Value and Feedback Message for Pedestrian Sign1



Cost Value and Feedback Message for Pedestrian Sign2



Cost Value and Feedback Message for Pedestrian Sign3



Cost Value and Feedback Message for Pedestrian Sign4

Fig. A-2. Traffic light Images, Stop, and Pedestrian Signs for Red-DBPNN

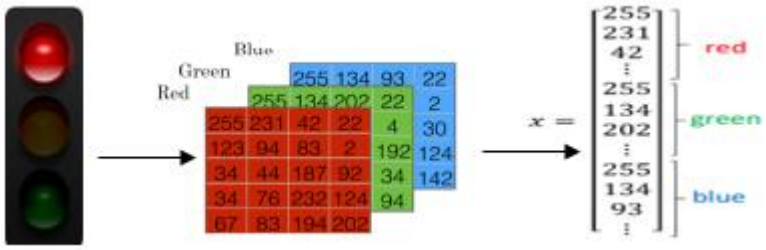


Fig. A-3. A Vector of RGB-Images Features Separated into 3-Channels

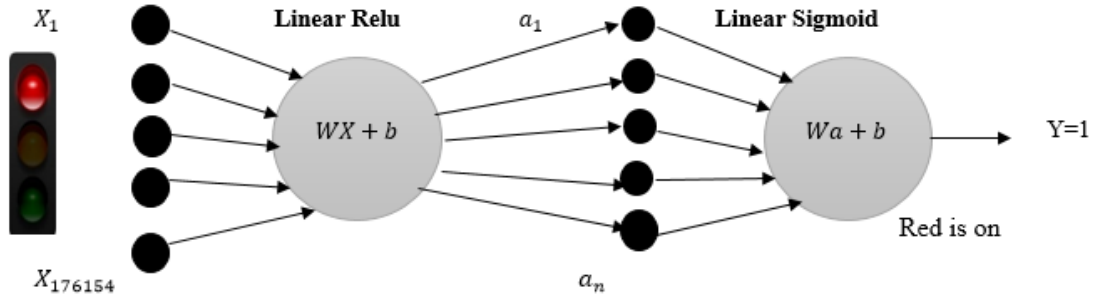


Fig. A-4. Red SBPNN Binary Classifier

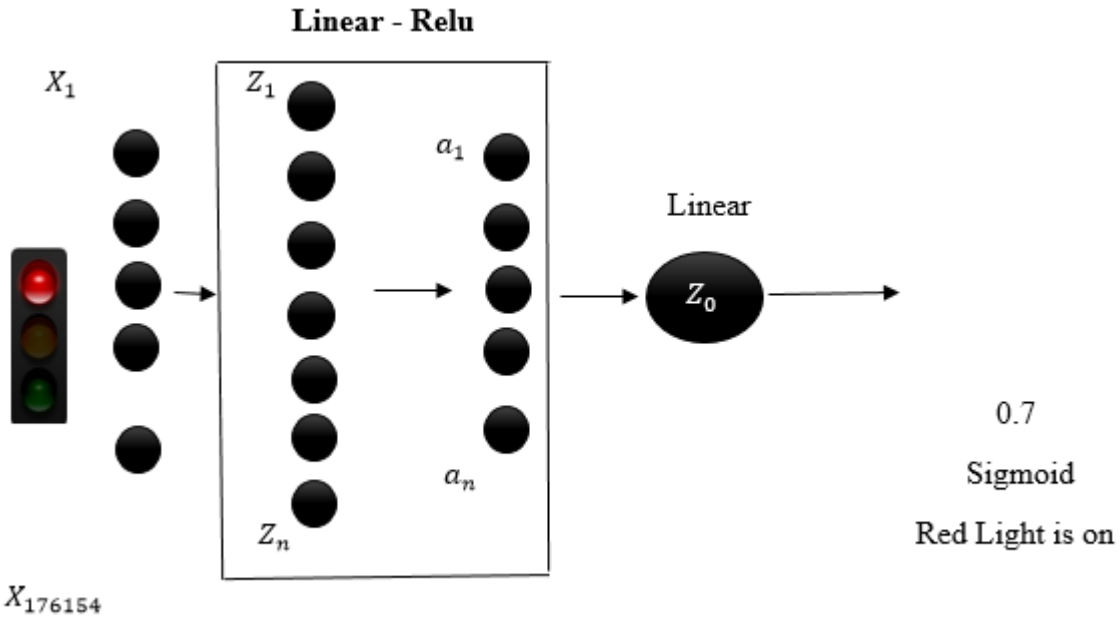


Fig. A-5. Red DBPNN Binary Classifier