Euroasia Journal of Mathematics, Engineering, Natural & Medical Sciences
International Indexed and Refereed
ISSN 2667-6702

DEVELOPMENT OF A REAL-TIME TRAFFIC SIGN RECOGNITION SYSTEM BASED ON DEEP LEARNING APPROACH WITH CONVOLUTIONAL NEURAL NETWORKS AND INTEGRATING TO THE EMBEDDED SYSTEMS

EVRİŞİMLİ SİNİR AĞLARI İLE DERİN ÖĞRENME YAKLAŞIMINA DAYALI GERÇEK ZAMANLI TRAFİK İŞARETİ TANIMA SİSTEMİNİN GELİŞTİRİLMESİ VE GÖMÜLÜ SİSTEMLERE ENTEGRE EDİLMESİ

Batur Alp AKGÜL[1]*, Bülent HAZNEDAR[2], M. Fatih HASOĞLU[2], K. Sercan BAYRAM[1]

[1]Department of Electrical and Electronics Engineering, Hasan Kalyoncu University, Gaziantep

[2]Department of Computer Engineering, Hasan Kalyoncu University, Gaziantep

**Corresponding Author: Batur Alp Akgül*

Geliş Tarihi / Received:10.02.2021
Kabul Tarihi / Accepted:21.03.2021

Araştırma Makalesi/Research Article
DOI: 10.38065/euroasiaorg.481

ABSTRACT

Traffic signs are mandatory features of road traffic regulations worldwide. Automatic detection and recognition of traffic signs by vehicles may increase the safety level of drivers and passengers. For this reason, Real Time-Traffic Sign Recognition (RT-TSR) system is one of the essential components for smart transportation systems and high-tech vehicles. Recently, very good performances have been achieved in public datasets, especially with advanced Computer Vision (CV) approaches like Deep Learning (DL). Nevertheless, these CV techniques still need to be improved to provide the requirements of Real-Time (RT) applications. Although hopefully outcomes have been obtained theoretically in previous Traffic Sign Recognition (TSR) studies, there are very few studies that offer RT solutions in the real world. Therefore, in this study, a DL-based RT-TSR system is developed because of its high rate of recognition and quick execution. Besides, the CV approach has been included in the software developed to achieve classification as RT and support digital imaging. This developed system is capable of running smoothly in embedded systems with mobile GPU or CPU thanks to its low computational cost and high performance. Therefore, this study makes two important contributions to this field: software and hardware. First, RT-TSR software has been developed by using Convolutional Neural Networks (CNN) built on DL techniques along with CV techniques. Secondly, the developed software is adapted to embedded devices and hardware design is made. This developed system is also a technology product that offers software and hardware solutions together. Coding is accomplished under TensorFlow and OpenCV frameworks with the python programming language and CNN training is carried out by using parallel architecture. The experimental findings indicate that the developed CNN architecture achieves 99,71% accuracy and confirms the high efficiency of the system.

Keywords: Real-Time, Traffic Sign Recognition, Deep Learning, Convolutional Neural Networks, Computer Vision, Embedded Systems, Developed.

ÖZET

The study was carried out to determine Trafik işaretleri, dünya çapında karayolu trafik düzenlemelerinin zorunlu bir özelliğidir. Trafik işaretlerinin araçlar tarafından otomatik olarak algılanması ve tanınması sürücülerin ve yolcuların güvenlik düzeyini artıracaktır. Bu nedenle Gerçek Zamanlı Trafik İşareti Tanıma (RT-TSR) sistemi akıllı ulaşım sistemleri ve yüksek teknolojili araçlar için önemli bileşenlerden biridir. Son zamanlarda, kamuya açık veri setlerinde, özellikle de Derin Öğrenme (DL) gibi son teknoloji yaklaşımlarla oldukça iyi performanslar elde edilmiştir. Ancak bu yöntemlerin Gerçek Zamanlı (RT) uygulamalardaki gereksinimleri karşılamak için hala

iyileştirilmesi gerekmektedir. Daha önceki TSR çalışmalarında kuramsal olarak ümit verici sonuçlar elde edilmiş olsa da gerçek dünyada RT çözümler sunan çok az çalışma vardır. Bu nedenle, bu çalışmada, yüksek tanıma oranı ve hızlı yürütülmesi nedeniyle DL tabanlı bir RT-TSR sistemi geliştirilmiştir. Ayrıca, RT olarak sınıflandırmayı gerçekleştirmek ve dijital görüntülemeyi desteklemek için geliştirilen yazılıma Bilgisayar Görüşü (CV) yaklaşımı da dahil edilmiştir. Geliştirilen bu sistem, düşük hesaplama maliyeti ve yüksek başarım nedeniyle mobil GPU ya da CPU ya sahip gömülü sistemlerde sorunsuz çalışabilecek kapasitededir. Bu çalışma bu alana yazılım and hardware olmak üzere iki önemli katkı sağlamaktadır. Birinci olarak, Evrişimli Sinir Ağları (CNN) tabanlı DL teknikleri kullanılarak ve CV teknikleri de ilave edilerek bir RT-TSR yazılımı geliştirilmiştir. İkinci olarak, geliştirilen yazılım gömülü cihazlara uyarlanmış ve donanımsal tasarımı yapılmıştır. Geliştirilen bu sistem aynı zamanda yazılım ve donanım çözümünü bir arada sunan bir teknoloji ürünüdür. Python programlama dili ile TensorFlow ve OpenCV çatısı altında kodlama yapılmıştır ve CNN eğitimi paralel mimari kullanılarak gerçekleştirilmiştir. Deneysel sonuçlar, geliştirilen CNN mimarisinin %99,71 doğruluk elde ettiğini göstermektedir ve sisteminin yüksek verimliliğini doğrulamaktadır.

Anahtar Kelimeler: Gerçek Zamanlı Trafik İşareti Tanıma, Bilgisayar Görüşü, Derin Öğrenme, Evrişimli Sinir Ağları, Trafik İşaretleri, Gömülü Sistemler.

1. INTRODUCTION

Traffic signs are warning, and caution signs are put on roads to advise drivers of road conditions and constraints or the way to go. Problems with road safety are mostly due to driver-specific subjective factors such as carelessness, inappropriate use of the driver, and non-compliance with traffic laws. For these reasons, high-tech cars have recently become an efficient method for eliminating these human factors [1-5]. RT-TSR is the method of identifying traffic signs as automatic and it will provide the capability for smart cars and smart driving. Developing RT-TSR systems that will enable state-of-the-art vehicles to automatically recognize traffic signs such as speed limits, pedestrian crossings and inform the driver will provide extra security for drivers and passengers [6-8]. In summary, TSR is a 3-step process. The first one is that preprocessing is a process to reduce negative effects by arranging the pixels and dimensions in the images. The second one is the localization process that detects and localizes where a traffic sign is found in an image. Lastly, recognition is a process to recognize and classify the traffic sign by taking the localized traffic sign.

Figure 1. Overview of the designed RT-TSR device

The development of the technology of new mobile processors has made it possible to install hardware TSR systems. Thanks to the TSR, which will be placed on the vehicles, the cameras scan and classify traffic signs at the sides of the road. As a warning to the driver, the recognized traffic sign is displayed on the LCD instrument panel at that time. Figure 1 gives the general view of the designed RT-TSR embedded system. Such systems help to increase safety dramatically on the autonomous driving path.

The RT-TSR problem is one of the best known and most discussed by many researchers and can be solved with DL-CV. However, poor detection accuracy and high requirement for hardware computing efficiency are the key issues for such systems. In this study, software and hardware technology developed for detecting and recognizing traffic signs on a real-time based is explained. The classification, recognition of traffic signs using CV-DL techniques and their integration into embedded systems are discussed in detail.

2. LITERATURE REVIEW

There is plenty of literature on TSR issues, and some review articles are available [9-10]. Recently, for its high precision and accuracy, CNN has been widely adopted in object detection [11-14]. Very good performances and promising results have been obtained in TSR with the latest technology approaches such as DL and CV [15-20]. However, these methods still need to be improved to meet the requirements of applications for Real-Time (RT). On the other hand, most of these studies are software, coding, algorithm, and method-based improvement studies without implementation. These studies on how to operate these techniques in RT-TSR systems and how to adapt them to embedded systems are very few and insufficient. For example, the point-like noise algorithm was described in [21], the effective adoption of the noise algorithm was described in [22], handles with the algorithms for detecting and recognizing traffic signs was described in [23].

In TSR, CNNs were used to automatically learn feature extraction and perform final classification [24-26], and an ensemble classifier composed of a few CNNs has been suggested in [27]. A variation of the CNN and multilayer perceptron approach was applied to further improve the accuracy, detection, and recognition [28]. Some methods using CNN and applied as a classifier to learn the properties were used in deep CNN [29-30]. Most of such studies have remained at the theoretical level, there are still deficiencies and question marks about how to put it into use in real life and how to develop hardware devices. There are still issues regarding the software being developed running at high speed and capacity in computers, reducing the hardware to micro levels, or developing software suitable for the hardware. The present work contributes to this particular field due to the development of an RT-TSR system in which hardware and software work together.

3. FUNDAMENTALS AND BACKGROUND

In this section, the libraries, frameworks, algorithms, techniques, background, and fundamentals used in this study are explained and their contributions to the study are presented.

3.1. Computer Vision (CV) and OpenCV

OpenCV is a CV and DL application library that is open source, and it is designed to provide applications with common infrastructure and to accelerate the use of DL. CV libraries that contain algorithms such as image processing, pattern recognition, and histogram extraction are needed in the design and development of Real-Time systems. OpenCV includes open-source versions of these algorithms and OpenCV is preferred to be used in this study due to its open-source and no copyright issues.

3.2. TensorFlow Ecosystem

TensorFlow is not just a computational engine and a deep learning library for training neural networks. It is a complete ecosystem used (TensorFlow Lite for mobile and embedded devices) for the development of production machine learning pipelines and deploying-quantization production models. It includes sessions and willing execution, automatic differentiation, model and layer sub-classification, and better multi-GPU/distributed training support functions. We can train, optimize, and quantify models designed to work on resource-constrained devices such as TensorFlow Lite, smartphones, and other embedded devices. Raspberry Pi and TensorFlow Lite were used in this study. The training and deployment steps of the TensorFlow ecosystem are shown in Figure 2.

3.3. Keras DL Framework

Keras is a high-developed deep learning framework for the phyton environment and Keras framework is used to create a deep CNN in this study. By using Keras libraries, a CNN is created and trained in the following steps: Loading data from disk, creating the training and the test sections, defining Keras model architecture, compiling Keras model, training the model with training data, evaluating the model over test data, and making predictions using the trained Keras model.

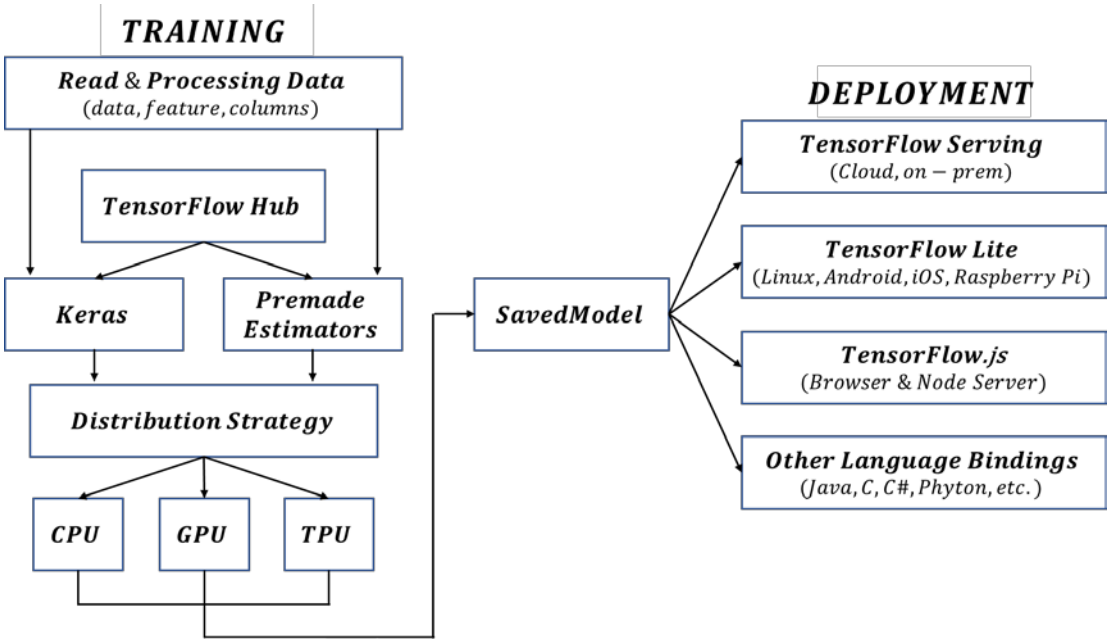

Figure 2. Simple diagram of the TensorFlow ecosystem

3.4. Convolutional Neural Networks (CNN)

Nowadays, solving classification, DL, and CV problems with CNN. Classifying with CNN is a highly famous and commonly used technique. Classifying with CNN is the best-in-class method for the CV process. A mathematical model based on connections through artificial neurons is a neural network. Usually, neurons are arranged in layers, and only appointed layers create connections between neural neurons. The entire process diagram of creating and optimizing a CNN is illustrated in Figure 3. The layer-concatenated CNN perception consists of 4 main parts: forward propagation, error calculation, backpropagation, and weights adjustment.

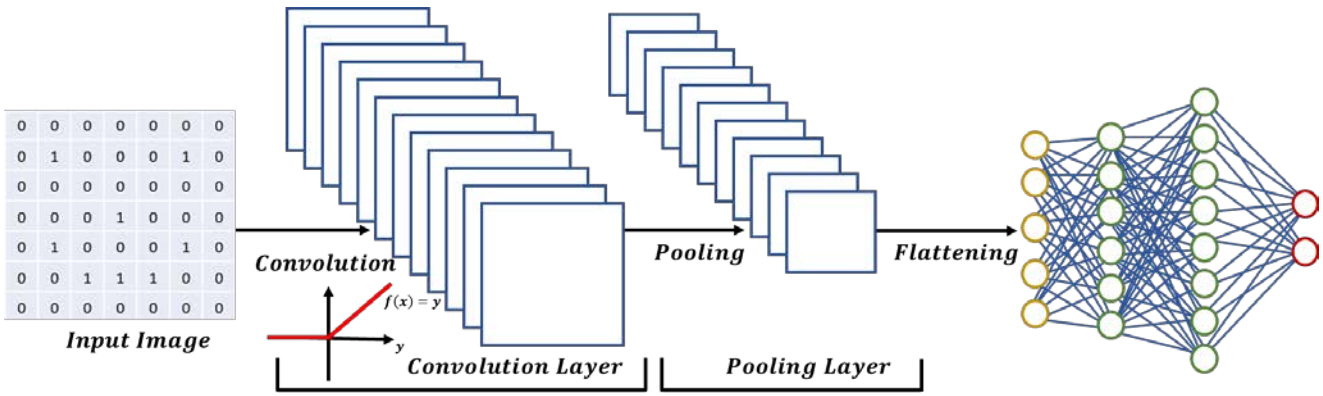

Figure 3. Entire process of creating and optimizing a CNN

4. MATERIALS AND METHODS

This section is discussed in two parts as software and hardware application. Firstly, the software application is explained by mentioning the subjects such as software development environment and programming. Later, the subjects of adapting the developed software to embedded systems and designing them as electronic devices by operating them in embedded systems are discussed.

4.1. Software Development

In this section, the mathematical and scientific foundations of the subjects such as the used definition of the software development environment, the used programming language, the prepared database-dataset classes, the used language libraries, and framework structures are explained. The development steps in general are very useful for developers to understand the logic and structure of this study.

4.1.1. Software Structure and Proposed System

In this study, the software is developed by installing the following packages on the phyton environment respectively; OpenCV, NumPy, Keras, Pandas, Scikit-learn, Scikit-image, Imutils, Matplotlib, Battery, TensorFlow for CPU and GPU. After the installation of these packages, the Phyton programming environment is created completely. GPU-based encodings accelerate the learning time and reduce the computational cost [31]. The flowchart of the proposed study is illustrated in Figure 4. The collected dataset is used as an input to the CNN architecture for training, validation, testing, and classification.

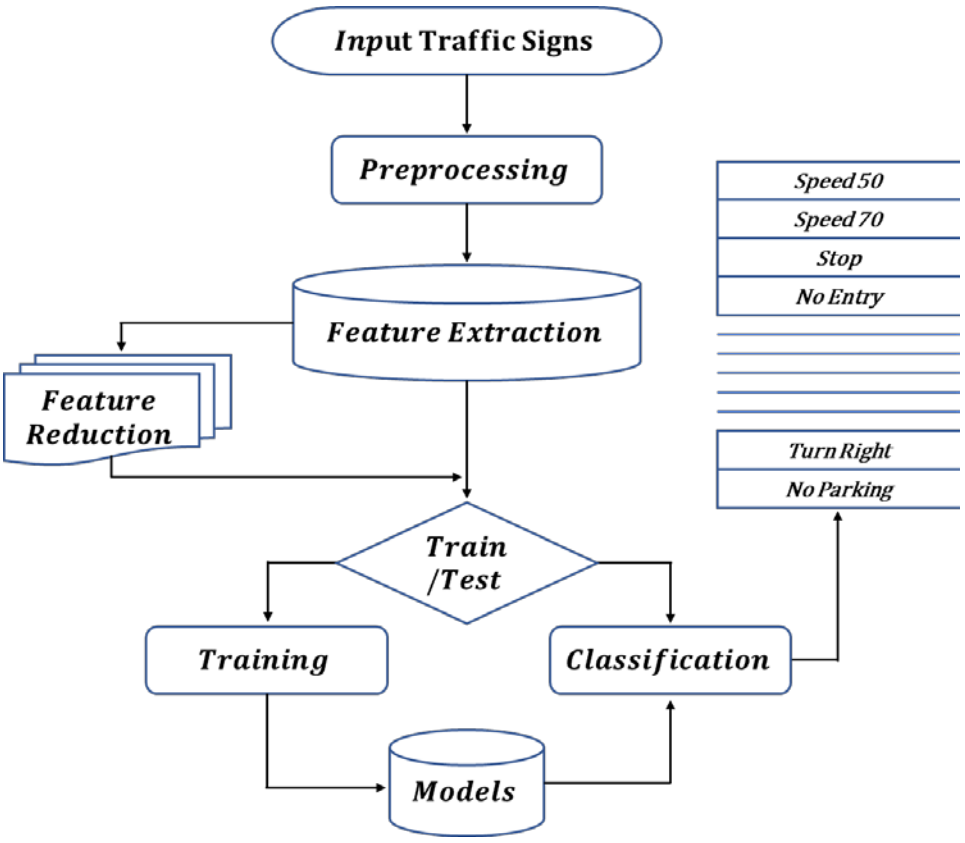

Figure 4. Flowchart of the proposed system

4.1.2. Experimental Dataset

The dimension of the training range is an important aspect to take into account in DL learning models. The system becomes ambiguous without a sufficient number of training examples. In this study, public open-source datasets are used. Used TSR Benchmarks of the dataset is shown in Figure 5.

Traffic signs are manually created and tagged, in this way TSR data dataset is created. Dataset consists of 40 traffic sign classes and thousands of images. The low resolution and poor contrast of the images in the used dataset can reduce the success rate. In some cases, it can be difficult for the learning algorithm to recognize the traffic sign. Therefore, the better dataset is optimized and the higher the resolution images are used, the higher the success rate reaches of the TSR. The dataset used in this study is designed as multi-class, designed as a large and realistic database. The class distribution of the dataset is given in Figure 6. The dataset is broken into sets for training, validation, and testing processes. This segmentation is used a statistical approach which is the Hold-out cross-validation method to measure and estimate the success rate of DL algorithms. This method is widely used because of its effectiveness and ease of application [32]. As there are no set common instructions for the data set's percentage partitioning, commonly used partitioning percentages are used.

4.1.3. Preprocessing and Implementing of CV-CLAHE Algorithm

Preprocessing procedures are conducted to the images of the dataset before entering the CNN such as resizing and gray scaling in order to eliminate negative impacts such as insufficient illumination, low contrast, dark, partial occlusion, and serious deformation. Preprocessing is applied in two steps. Firstly, the problem of light and contrast in the pictures is solved with the CLAHE algorithm. Secondly, images fixed with CLAHE are converted to gray scaling type.

Figure 5. Used TSR Benchmarks of the dataset

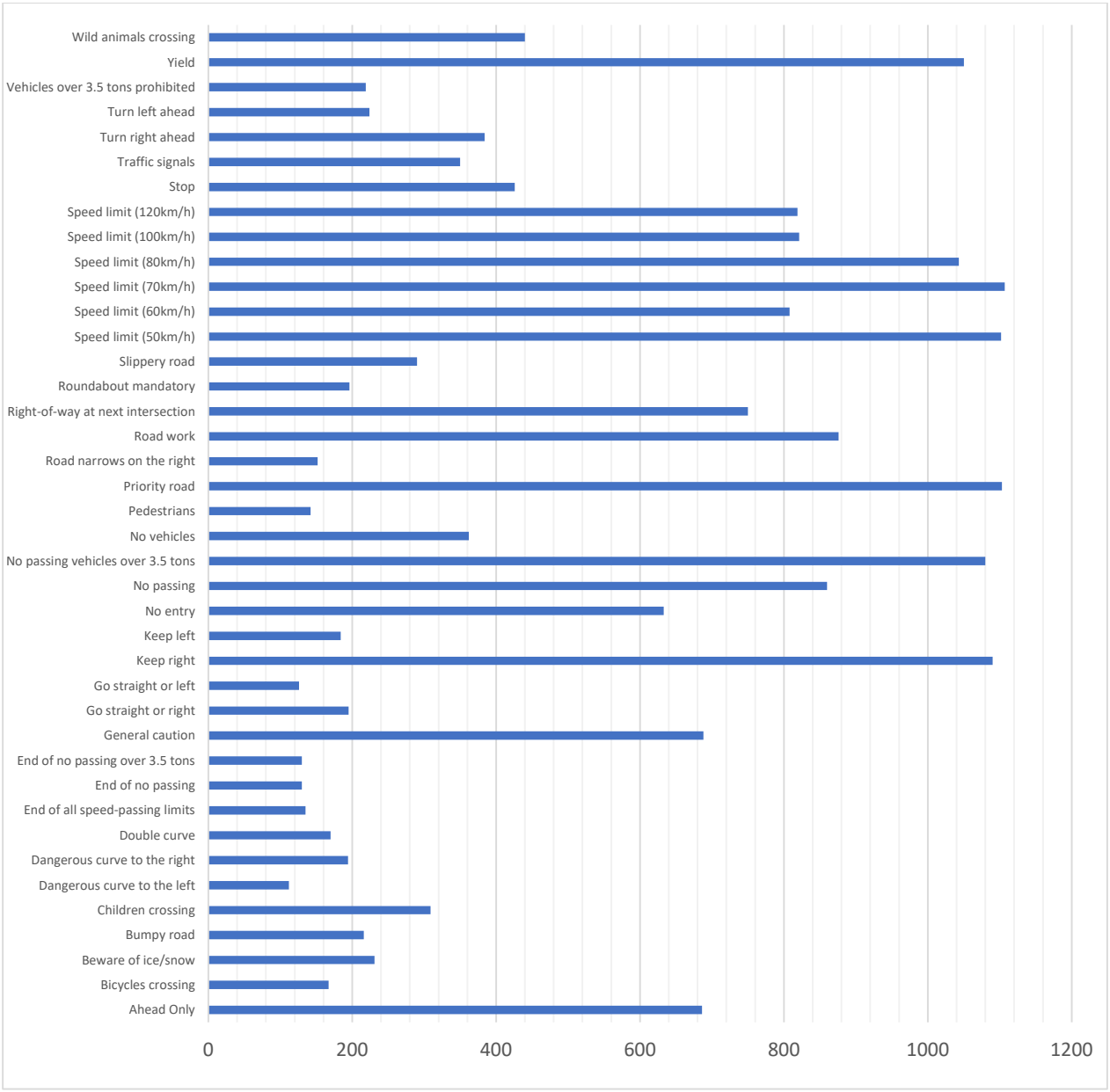

Figure 6. The class distribution of the dataset

One of the main problems for the images in the dataset is there are much low contrast images, this problem makes it difficult for the DL model to recognize a traffic sign. Therefore, CV techniques need to be added to the software. The image contrast is automatically improved by applying the CLAHE in the Scikit-Image library not only normalizes the images but also increases local contrast details in areas that are darker or lighter than other images. Figure 7 shows the preprocessed samples of images with the CLAHE and Figure 8 shows the samples of the gray scaling with preprocessing of images. Although the modified images look abnormal and synthetic to the human eyes with this imaging, the correction of contrast helps CLAHE to automatically recognize traffic signs even better [33]. As part of preprocessing samples for deep learning classification of traffic signs, the CLAHE is used to improve image contrast. It is seen that from the original images (input images) in Figure 7 (left), the contrast is very poor, and it is not easy to identify any signals. Also, it is seen that from the processed images by applying CLAHE in figure 7 (right) how to improve image contrast.

Figure 7. Preporecessed samples of images with the CLAHE algorithm

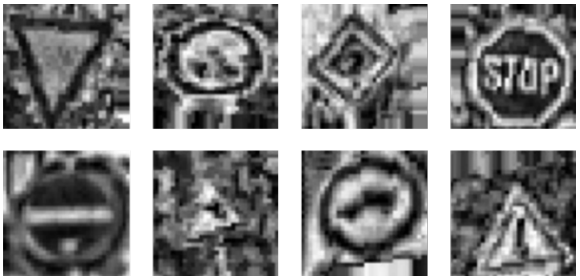

Figure 8. Preprocessed samples of the gray scaling with preprocessing of images

4.1.4. Creating CNN with Keras Framework

The CNN architecture is done by using the Keras framework. The proposed architecture of CNN consists of multiple hidden layers between the layers of input and output [34]. Since the convergence rate is quicker and it has a higher generalization efficiency, the max-pooling strategy is used. The Rectified Linear Unit (ReLU) is an activation method commonly seen in the CNN architecture. ReLU has the benefit of having quicker training from CNN. When multiple classes are used, the SoftMax function and categorical cross-entropy are commonly used in CV studies. To predict the label of that specific input, the SoftMax activation function is used and this function also provides a distribution of probability. The categorical cross-entropy function is used to observe the loss of SoftMax.

The purpose of parameter adjustment is to increase the accuracy of training and shorten the training time. The choice of different parameters for the specification like nonlinearity and pooling variants directly affects the performance of the CNN [35-36]. Using the correct configuration of techniques inactivation, normalization and pooling layers are essential to increase the capacity of the CNN [37-

38]. After setting the required parameters, it is compiled with CNN Adam optimizer. Optimization algorithms try to optimize a particular function according to its parameters, CNN weights and biases are set as automatic with optimization algorithms. Adam is an algorithm for optimization used for updating CNN weights and it is efficient and requires less memory in terms of the computation cost [40]. A simple diagram of the architecture of the CNN classifier created in this study is presented in Figure 8.

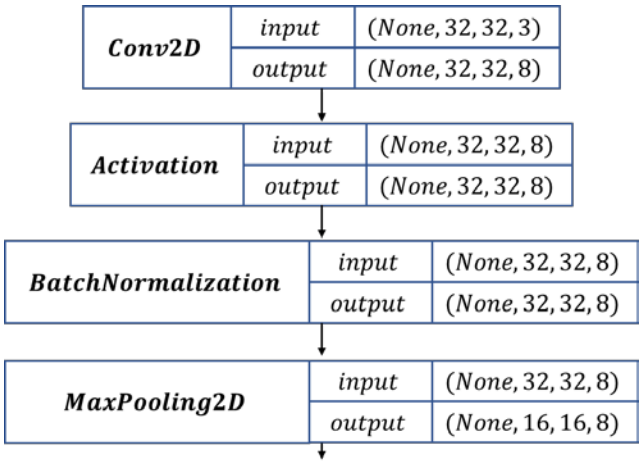

Figure 8. Simple diagram CNN for traffic sign classification architecture.

4.1.5. Implementing of Training and Prediction Scripts

After the design of CNN architecture, the Python training script is created and especially responsible for; loading the training and testing part from the dataset, preprocessing the images, training the model, evaluating the accuracy of the model, and serializing the model to make predictions about the new traffic sign data. In order to accomplish these operations, it is necessary to import some packages into the training file (matplotlib, classification_report, skimage, numpy, argparse, random, etc.). After training the traffic sign recognition model, prediction operations are performed and consist of the following steps; Model is loaded from disk, sample images are loaded from disk, sample images are pre-processed, images are passed through traffic sign classifier, and final output estimates are obtained.

4.1.6. Implementing of Real-Time Recognition

In this study, the traffic sign recognition process is treated as a classification problem, that is to say, datasets and classes are created by pre-cropping the entrance images of traffic signs. On the other hand, the recognition of traffic signs is an object detection problem in the real world. It allows us not only to recognize traffic signs but also to determine where traffic signs are in the entrance frame. It is important to emphasize that the method of object detection is not as simplistic and effortless as classifying pictures. In reality, it requires much more complex and detailed procedures due to the parameters of location, time, and speed. At this point, some high-tech cameras that can automatically detect and focus on objects can be used to make real-time recognition. In this study, a simple USB-computer camera and Raspberry Pi integrated camera were used for testing.

The RT process consists of three steps. The first step is to take the traffic sign image as an input from a camera. Traffic signs on the images are processed as inputs with a camera. To access the camera, an endless loop has been made to capture every frame in the image. The methods provided by OpenCV are used to access the camera and set the capture objects. The camera reads every frame in the image, and it is stored in a frame variable file. The second step is to detect and feed the traffic sign on the image to the classifier. The “Haar Cascade” which is the one of best kinds of classifier methods is used to detect the traffic sign in the image [41]. This method returns a series of detections

with x=, y= coordinates, and height, which are the width of the object's bounding box, and the border of boxes are drawn for the image. Afterward, the boundary box of the image is extracted, and it is achieved the traffic sign image from the frame with this code. The third step is to calculate the score to check the accuracy for a detected sign. The score is a value to determine the accuracy percent of the traffic sign estimation.

4.2. Hardware Development

In this section, the used LCD screen, camera, mainboard, etc., embedded system design with electronic devices are explained. The developed RT-TSR system is operated on the embedded system.

4.2.1. Implementation of the Raspberry PI-3

Raspberry Pi-3 development boards are used to design the developed RT-TSR software on the basis of an embedded system. It can be easy to think of this Pi-3 model as a single microcomputer board running on the LINUX operating system.

The board also has many features and satisfactory processing speed, making the Raspberry Pi board appropriate to use for sophisticated studies. The Pi board is specially designed for engineers doing research and development with LINUX operating systems and the Internet of Things (IoT). The pi-3 card requires 5.1V supply, 2.5A PSU current capacity, 1.2A maximum total USB peripheral current draw, and 500mA typical bare-board active current consumption [42]. Figure 10 shows the Raspberry Pi-3 board and technical specifications for it are listed in Table 1.

Table 1. Technical specifications of the Pi-3 Card

Microprocessor	Broadcom BCM2837 64bit Quad Core Processor
Processor Operating Voltage	3.3V
Raw Voltage input	5V, 2A power source
Maximum current through each I/O pin	16mA
Maximum total current drawn from all I/O pins	54mA
Flash Memory (Operating System)	16Gbytes SSD memory card
Internal RAM	1Gbytes DDR2
Clock Frequency	1.2GHz
GPU	Dual-Core Video Core IV® Multimedia Co-Processor. Provides Open GLES 2.0, hardware-accelerated Open VG, 1080p H.264 decode.
Ethernet	10/100 Ethernet, Base T Ethernet Socket
Wireless Connectivity	BCM43143 (802.11 b/g/n Wireless LAN and Bluetooth 4.1)
Operating Temperature	-40ºC to +85ºC
USB	2.0 (Four sockets)
Audio Output	3.5mm Jack and HDMI
Video output	HDMI
Camera Connector	15-pin MIPI Camera Serial Interface (CSI-2)
Display Connector	Display Serial Interface (DSI) 15-way flat flex with two data lanes and a clock lane.
Memory Card Slot	Push/Pull Micro SDIO

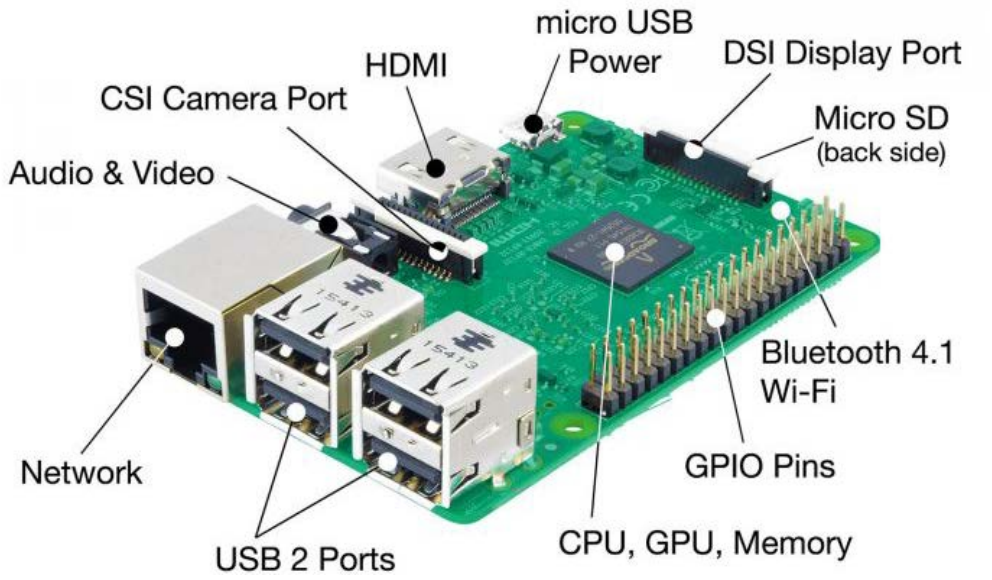

Figure 10. Raspberry Pi-3 kartı

4.2.2. Implementing of the OS and Programming Environment

The software is developed in the Windows OS environment and later adapted to the Linux Rasp OS with the programming environment by making the necessary revisions. The RT-TSR results obtained with the Raspberry PI-3 card as an embedded system are presented in the next section.

4.2.3. Implementation of the Camera Devices

Raspberry Pi Camera v2 HD module is integrated into the Pi-3 development board. This official camera board for the Pi-3 card is a high-quality 8-megapixel image sensor with a fixed focus lens, specially designed for Raspberry Pi. It is capable of a 3280 x 2464 pixels static display. Also, it supports 1080p30, 720p60 and 640x480p60/90 videos [43]. It has been preferred by developers due to its capability for high image quality, color accuracy, low light performance, and dimensions. It connects to the Pi card through one of the pin sockets on the top surface of the board and uses a proprietary CSI interface which is specifically designed to connect the camera.

Another camera used in this study is the A4 Tech HD 1080P USB Web camera. Resolution is HD 1080P, 1920x1080 Pixels [44]. It is cheaper compared to the PI camera board. It can find in grocery stores and has simple features compared to the Pi camera module. This camera is used during software development in a Windows OS environment. Later, in the embedded system design, the Pi camera is also used and added to the design. Figure 11b shows the camera module integrated into the Raspberry Pi card, and Figure 11c shows the web camera used by connecting via the USB port.

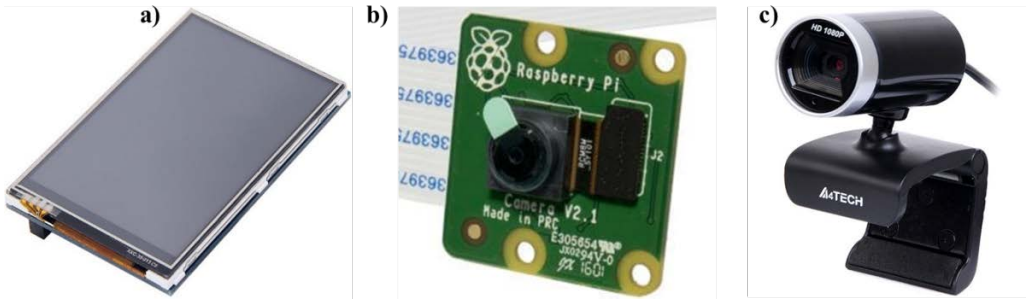

Figure 11. Equipment used in the designed TSR device, a) Screen, b) USB camera, c) PI camera.

4.2.4. Implementation of the LCD Display and UPS Circuit with Battery

There is no need to add an LCD panel and battery to the designed system for high-tech vehicles, automotive engineers will adapt this system to the LCD panel or battery of the vehicle according to the technological structure of the vehicles to which the system will be integrated. However, all the design of the system and all equipment should be tested with multiple conditions in the development environment. Also, important criteria such as the system performance with the battery or the effect of the energy consumed by the LCD panel on the system performance should be considered carefully by the developers. On the other hand, if the RT-TSR system to be integrated into existing (already manufactured) vehicles, this equipment should be added to the design. Therefore, an LCD panel and battery are used in this study. The used LCD panel and the UPS circuit with the battery model are shown in Figure 11a and Figure 12.

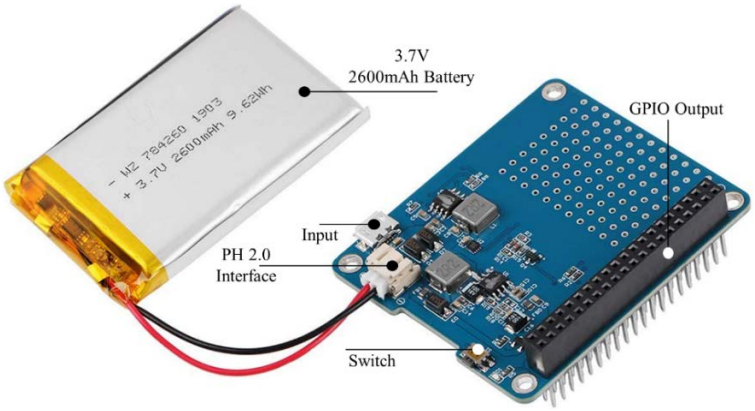

Figure 12. UPS Circuit and its battery of the Raspberry PI

The UPS board enables the Raspberry Pi to be used with a portable power source and is specially designed for Raspberry Pi. It supports up to 2A faster-charging technology to avoid overcharge and over-discharge issues, also it quickly charges the battery entirely with a safety circuit. It is more convenient to use an external 3.7V-2600mAh lithium battery [45]. Table 2 provides technical specifications of the used UPS card. The used LCD flat panel screen is a TFT (thin-film transistor) technology and is controlled via a resistive touch (touch screen controller). Raspberry Pi has a standard female head for GPIO pins, and it is easy to connect to Raspberry Pi. The screen size is the same as the Pi board size (3.5-inch). It has 480x320 pixel resolution with 24-Bit color and high speed 48Mhz SPI connection [46].

Table 2. Technical specifications of the UPS module

	Item	Min	General	Max	Unit	Note
Input	Voltage	4.5	5	5.5	V	
	Charging current			2	A	
	Undervoltage protection		4.5		V	
Output	Voltage	4.9	5	5.1	V	
	Current			2	A	
	Charging Current			1.4	A	up to 1.8A
	Overcurrent protection		30		ms	to be less than 4.2V
	Short Circuit protection	150		200	us	to be greater than 3.5V
Operation	Short press to wake up		50		ms	wake up
	Long press switches the output interface		2	2.5	s	
	Long press with double-click shutdown					Power off
	Intelligent sleep mode					to be less than 45mA

5. RESULTS AND DISCUSSIONS

In this section, estimates are made about traffic sign data by using the TSR model (CNN, Train, and Prediction script files, dataset) and the results are presented.

5.1. Discussions of the Dataset

The distribution of the training data set is given in Figure 6. The distribution is not properly balanced when analyzed. Therefore, it is important to find out how many data sets are required for a good classification. As depicted in Figure 6, while there are 1,107 images in the top class (Speed limit (70km / h)), there are 112 images in the least represented class (Dangerous curve to the left). While the accuracy rate of the class with the max image is 100%, the accuracy rate of the class with the min image is below 95%. Since the success rate target in this study is at least 99%, all classes in the data set should have 600 views. Therefore, the threshold value of the data set classes is set as 600. The dataset should be developed with a minimum of 600 images of all classes with less than 600 images in the dataset.

One of the important issues related to the dataset is maintaining a balance between classes. The number of images in the classes should be optimized (neither too much nor too much) and the number of images in the classes should be close to each other as much as possible. This will increase the speed and performance of the learning algorithm as well as increase the success rate. In order to successfully train the classifier algorithm, the image contrasts must be improved, and our input images must be preprocessed. Also, class label curvature should be taken into account. In order to account for this distortion in the dataset when coding (some classes have significantly more views than others) it is necessary to assign weights to each class in the dataset for use throughout the training.

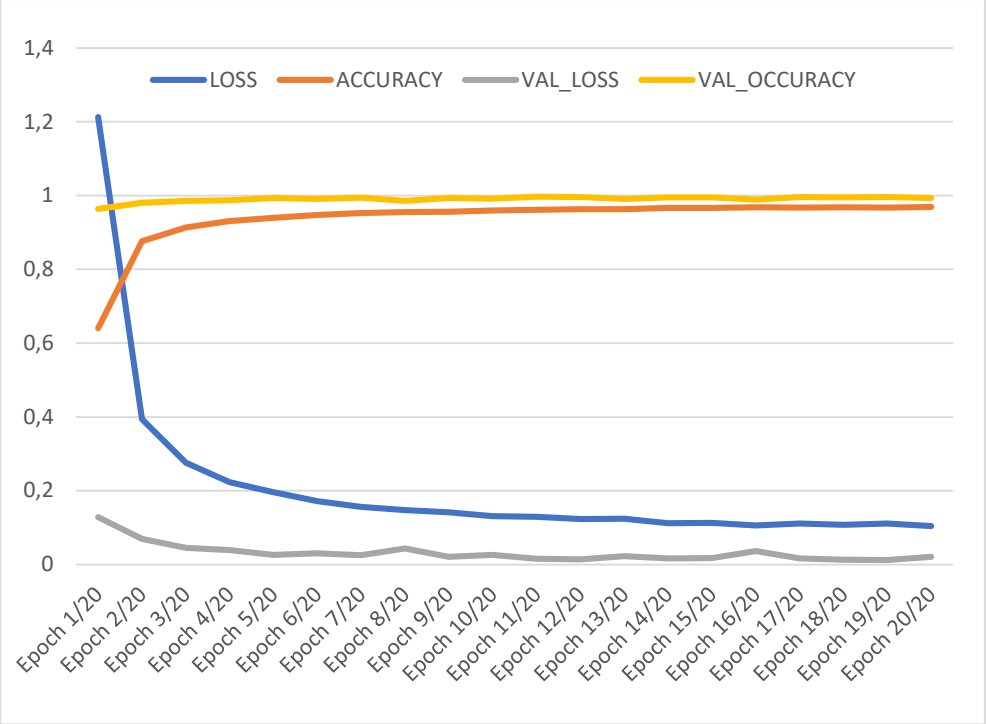

Figure 13. The accuracy and loss charts for all 20 epochs.

5.2. Discussions of the Epochs and Accuracy Rates

The Epochs value indicates how many iterations it takes to achieve the best accuracy. It is important to find out how much accuracy and loss are taken for each iteration. The epochs number is directly proportional to the accuracy of the test, incrementing epoch size incrementing the accuracy of the test. However, when the model reaches saturation point, incrementing the number of epochs does not affect accuracy performance. Figure 13 shows charts of accuracy and loss values for all 20 epochs. As seen from the charts in Figure 13, the value between 10 and 15 epochs period is a good predictive value to train the network model. It is calculated with 20 epochs and reaches the almost same level after about 15 epochs. Therefore, the epochs threshold value is set to 15 for this study. Besides, it is understood from Table 1 that as the number of periods increments, the accuracy rate increments, and the loss rate decreases. Figure 13 supports the decision to set the period threshold as 15. From the results in Table 1, it is seen that the developed model is very well trained.

Table 3. Accuracy table of the trained model

Epochs	ETA/SC	Loss	Accuracy	Val_Loss	Val_Accuracy
Epoch 1/20	612	1,2131	0,6403	0,1281	0,9656
Epoch 5/20	640	0,1963	0,9395	0,0259	0,9941
Epoch 10/20	606	0,1311	0,9598	0,0262	0,9931
Epoch 15/20	628	1,1126	0,9664	0,0171	0,9954
Epoch 20/20	611	0,1044	0,9695	0,0211	0,9947
Test Score	**0,0144**				
Test Accuracy	**0,9971**				

5.3. TSR Discussions in Real Environment

RT- TSR systems, the evaluations are carried out on real traffic signs on the road captured by vehicle-mounted cameras. While testing the technology that was developed to detect and classify traffic signs in true circumstances with a different location and signs using videos from cameras built on Raspberry Pi, it is observed that the software and hardware combination used worked in harmony with a high performance between 98% minimum probability and maximum 100% probability have been reached. Figure 15 confirms this determination. Considering that high-tech cameras that can automatically focus on objects with the zoom option will be integrated into the vehicles, the probability ratios seem acceptable. This designed RT-TSR system has also been tested with the Raspberry PI Zero card and the results are positive. This card has fewer features compared to Raspberry PI-3, but it can be preferred in terms of size and cost. Considering that this system will be mounted on vehicles, cost and size appear as very important criteria and this would be a suitable solution.

Figure 15. The RT-TSR tests are done in the real environment

6. REGIONAL CONTRIBUTIONS AND R&D QUALITY

With this study, it is planned to offer integration and adaptation options to end-users and automotive companies. End users can integrate devices as an extra to the existing system. Automotive companies can adapt the RT-TSR technology emerging in this study to their software.

Considering the contributions of the presented method in terms of material, it is seen that the developed software and algorithms recognize traffic signs in real-time with a 99% performance rate in hardware devices with low processor and RAM. Considering the contributions of the presented method in terms of material, the developed system can be integrated into all previously produced and used cars. It can be adapted to the own brain and panels of new-generation vehicles. In addition, by developing decision-making algorithms, it is possible for vehicles produced with autopilot to make decisions according to traffic signs (such as speed limit control, stopping at a pedestrian crossing).

It is quite possible for this study to be activated in the determined pilot regions and to present the

developed system to the worldwide use of automotive companies with some international competence. Due to the fact that this work, which has commercialization potential, has an open-source system infrastructure, its development and upgrade processes can be done by large masses around the world.

One of the significant characteristics of the RT-TSR system is presented to the user by logging the driver performance and errors on the detailed reporting screen. In this way, it is aimed that the users can control themself and see their errors. Retrospective data can be recorded for a long time. In addition, the system updates and upgrades of the devices can be done by remote connection.

7. CONCLUSIONS

In this study, starting from the foundations of DL, practical studies are carried out on how to create and train CNN, and how to make inferences. CLAHE algorithm is used for the purpose of preprocessing images. A CNN-based model with a CV-DL approach has been proposed for detecting, recognizing, and classifying traffic signs from images captured as RT in difficult conditions. The developed software is adapted to embedded devices and hardware design is made with its low training time and low computational cost. This developed RT-TSR is also a technology product that offers software and hardware solutions together. Solutions to use this technology are offered to manufacturers and end-users separately. RT-TSR has yielded very good results with an accuracy rate of 99.71% in the real environment with the best in terms of precision or optimization. Experiments yielded consistent results with the correct classification of traffic sign patterns for the complex background images. Using the TensorFlow and Keras frameworks with the CV-DL approach and the Python programming language, the model is created with extremely low application time.

8. FEATURE WORKS

The current study aimed for common traffic signs. Multilanguage support and multiple national traffic sign recognition skills can also be added to this work. In this way, it will be possible to use it in all regions or nationally. It should not be forgotten that although some of the traffic signs of each country are not the same. In order to use such a system in vehicles in different countries, datasets should be created for traffic signs of that country and software should be developed on those datasets. It is understood that no such research is done in Turkey, thus forming a dataset covering the whole of Turkey's traffic signs, and accordingly the development of embedded software and electronic systems can be worked the next.

REFERENCES

[1] Eichberger A, Wallner D. “Review of recent patents in integrated vehicle safety, advanced driver assistance systems and intelligent transportation systems”, Recent Pat. Mechanical Engineering, pp. 32-44, (2010), DOI:10.2174/1874477X11003010032.

[2] Montreal CO. “Road safety: Human factors aspects of intelligent vehicle technologies” Intelligent Transport Systems (VEHITS), pp. 318-332, (2017), DOI:10.1007/978-3-030-02907-4-16

[3] Andreev S, Petrov V, Huang K, Lema MA, and Dohler M. “Dense moving fog for intelligent IoT: Key challenges and opportunities”, IEEE Communications Magazine, pp. 1-7, (2018), DOI:10.1109/MCOM.2019.1800226.

[4] Campbell S, Naeem W and Irwin GW. “A review on improving the autonomy of unmanned surface vehicles through intelligent collision avoidance maneuvers”, Annuals Reviews in Control, pp. 267-283, (2012), DOI:10.1016/j.arcontrol.2012.09.008

[5] Luo Y, Gao Y, and You ZD. “Overview research of influence of in-vehicle intelligent terminals

on drivers' distraction and driving safety", 17th COTA International Conference of Transportation Professionals (CICTP), pp. 4197-4205, (2017), DOI:10.1061/9780784480915.435

[6] Alghmgham DA, Latif G, Alghozo j and Alzubaidi L. "Autonomous Traffic Sign (ATSR) Detection and Recognition using Deep CNN", ScienceDirect-Elsevier, 16th International Learning & Technology Conference, pp. 266-274, (2019), DOI:10.1016/j.procs.2019.12.108.

[7] Shustanov A, and Yakimov P. "CNN Design for Real-Time Traffic Sign Recognition", ScienceDirect-Elsevier, 3rd International Conference-Information Technology and Nanotechnology, PP. 718-725, (2017), DOI:10.1016/j.proeng.2017.09.594.

[8] Lina HY and Chang CC. "Traffic Sign Detection and Recognition for Driving Assistance System", AIVP-advance image and video processing, pp. 17-25, (2018), DOI:10.14738/aivp.63.4603.

[9] Mogelmose A, Trivedi MM, and Moeslund TB. "Vision-Based Traffic Sign Detection and Analysis for Intelligent Driver Assistance Systems: Perspectives and Survey", Transactions on Intelligent Transportation Systems, pp. 1484-1497, (2012), DOI:10.1109/TITS.2012.2209421.

[10] Wali SB, Hannan MA, Hussain A, and Samad SA. "Comparative Survey on Traffic Sign Detection and Recognition: A Review", Przeglad Elektrotech, pp. 38-42, (2015), DOI:10.15199/48.2015.12.08.

[11] Sermanet P and LeCun Y. "Traffic sign recognition with multi-scale convolutional networks", in Neural Networks (IJCNN), The International Joint Conference, IEEE Xplore, pp. 2809-2813, (2011), DOI:10.1109/IJCNN.2011.6033589.

[12] Jarrett K, Kavukcuoglu K, Ranzato M, and LeCun Y. "What is the best multi-stage architecture for object recognition?", Computer Vision, IEEE Xplore 12th International Conference, pp. 2146-2153, (2009), DOI:10.1109/ICCV.2009.5459469.

[13] Ciresan D, Meier U, Masci J, and Schmidhuber J. "A committee of neural networks for traffic sign classification" IEEE Xplore International Joint Conference on Neural Networks-IJCNN, pp. 1918-1921, (2011), DOI:10.1109/IJCNN.2011.6033458.

[14] Krizhevsky A, Sutskever I and Hinton GE. "ImageNet classification with deep convolutional neural networks", NIPS, pp. 1106-1114, (2012), DOI:10.1145/3065386

[15] Belaroussi R, Foucher P, Tarel JP, Soheilian B, Charbonnier P and Paparoditis N. "Road Sign Detection in Images, A Case Study", 20th International Conference on Pattern Recognition (ICPR), pp. 484-488, (2010), DOI:10.1109/ICPR.2010.1125.

[16] S. Lyu and E. P. Simoncelli, "Nonlinear image representation using divisive normalization" pp. 1-2, (2008), DOI:10.1109/CVPR.2008.4587821.

[17] Houben S, Stallkamp J, Salmen J, Schlipsing M and Igel C. "Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark," in International Joint Conference on Neural Networks, pp. 1-5, (2013), DOI:10.1109/IJCNN.2013.6706807.

[18] Fan RE, Chang KW, Hsieh CJ, Wang XR, and Lin CJ. "LIBLINEAR: A library for large linear classification", Journal of Machine L, pp. 1871-1874, (2008), DOI:10.1145/1390681.1442794.

[19] Liang M, Yuan M, Hu X, Li J and Liu H. "Traffic sign detection by ROI extraction and histogram features-based recognition", IJCNN, pp. 1-5, (2013), DOI:10.1109/IJCNN.2013.6706810.

[20] Boureau YL, Ponce J and LeCun Y. "A theoretical analysis of feature pooling in visual recognition", Proceedings of the 27th Int. Conference on Machine L., ICML, pp. 111-118, (2010).

[21] Fursov V, Bibkov S and Yakimov P. "Localization of objects contours with different scales in images using Hough transform", Journal of Computer Optics., pp. 502-508, (2013), DOI:10.18287/0134-2452-2013-37-4-496-502.

[22] Yakimov P. “Preprocessing of digital images in systems of location and recognition of road signs”, Journal of Computer Optics, pp. 401-405, (2013), DOI:10.18287/0134-2452-2013-37-3-401-405.

[23] Yakimov P. “Tracking traffic signs in video sequences based on a vehicle velocity”, Computer Optics, pp. 795-800, (2015), DOI: 10.18287/0134-2452-2015-39-5-795-800.

[24] Ciresan D, Meier U, Masci J and Schmidhuber J. “Multi-column deep neural network for traffic sign classification”, Neural Networks, pp. 333-338, (2012), DOI:10.1016/j.neunet.2012.02.023.

[25] Sermanet P and Le Cun Y. “Traffic sign recognition with multi-scale Convolutional Networks”, IJCNN, pp. 2809-2813, (2011), DOI:10.1109/IJCNN.2011.6033589.

[26] Vukotic V. Krapac J. and Segvic S. “Convolutional Neural Networks for Croatian Traffic Signs Recognition”, CCVW, pp. 15-20, (2014), DOI:10.20532/ccvw.2014.0016.

[27] Jin J, Fu K, and Zhang C. “Traffic Sign Recognition with Hinge Loss Trained Convolutional Neural Networks”, Transportation, IEEE, pp. 1991-2000, (2014), DOI:10.1109/TITS.2014.2308281.

[28] Ciresan D, Meier U, Masci J, and Schmidhuber J. “A committee of neural networks for traffic sign classification”, IJCNN, pp. 1918-1921, (2011), DOI:10.1109/IJCNN.2011.6033458.

[29] Zeng Y, Xu X, Fang Y and Zhao K. “Traffic Sign Recognition Using Deep Convolutional Networks and Machine L.”, IScIDE, pp. 272-280, (2015), DOI:10.1007/978-3-319-23989-7-28.

[30] Haloi M. “Traffic Sign Classification Using Deep Inception Based Convolutional Networks”, CoRR, vol. abs/1511.0, Indian Institute of Technology, pp. 1-5, (2015).

[31] Shustanov A, and Yakimov P. “A Method for Traffic Sign Recognition with CNN using GPU”, ICETE-14th International Joint Conference on e-Business and Telecommunications, pp. 42-47, (2017), DOI:10.5220/0006436100420047.

[32] Kim B, Park S, Kim K, Lim J, and Nahm K. “Systematic process to determine DNBR limit of CHF correlation with repetitive cross-validation technique”, Journal of Nuclear Science and Technology, pp. 1-9, (2018), DOI:10.1080/00223131.2018.1467287.

[33] Uday K, Sowmitra D, Abid A and Kamrul H. “Traffic-Sign Detection and Classification Under Challenging Conditions: A Deep Neural Network Based Approach”, Bangladesh University of Engineering and Technology, pp. 266-274, (2017), DOI:10.13140/RG.2.2.19795.63529.

[34] Hoo-Chang S, Roth HR, Gao M, Lu L, Xu Z, Nogues I and Summers RM. “Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning”, IEEE transactions on medical imaging, pp. 1-15, (2016), DOI:10.1109/TMI.2016.2528162.

[35] Lai Y, Wang N, Yang Y and Lin L. “Traffic Signs Recognition and Classification based on Deep Feature Learning”, ICPRAM-7th International Conference on Pattern Recognition Applications and Methods, pp. 622-629, (2018), DOI:10.5220/0006718806220629.

[36] Cao J, Song C, Peng S, X Feng and Song S. “Improved traffic sign detection and recognition algorithm for intelligent vehicles”, Sensors-MDPI, pp. 1-21, (2019), DOI:10.3390/s19184021.

[37] Latif G, Alghazo J, Alzubaidi L, Naseer MM and Alghazo Y. “Deep Convolutional Neural Network for Recognition of Unified Multi-Language Handwritten Numerals", IEEE 2nd Int. Derived Script Analysis and Recognition (ASAR), pp. 90-95, (2018), DOI:10.1109/ASAR.2018.8480289.

[38] Miikkulainen R, Liang J, Meyerson E, Rawal A, Fink D, Francon O, and Hodjat B. “Evolving deep neural networks. In Artificial Intelligence in the Age of Neural Networks and Brain Computing” Cornell University Academic Press, Computer Science, and Neural and Evolutionary, pp. 293-312, (2019).

[39] Latif G, Iskandar DA, Alghazo J, Butt M, and Khan AH. “Deep CNN based MR Image

Denoising for Tumor Segmentation using Watershed Transform", International Journal of Engineering & Technology, pp. 37-42, (2018), DOI:10.14419/ijet.v7i2.3.9964.

[40] Xiong F, Xiao Y, Cao Z, Gong K, Fang Z, and Zhou JT. "Towards good practices on building effective CNN baseline model for person re-identification", 10th International Conference on Graphics and Image Processing-ICGIP PP. 1-8, (2018), DOI:10.1117/12.2524386.

[41] Abdi L and Meddeb A. "Deep Learning Traffic Sign Detection, Recognition and Augmentation", Proceedings of the Symposium on Applied Computing, pp. 131-136, (2017), DOI:10.1145/3019612.3019643.

[42] Raspberry Ltd, PI-3 Card technical specifications, "Official documentation web site", https://www.raspberrypi.org/documentation/hardware/raspberrypi/, last access is 01 November 2020.

[43] Raspberry Ltd, PI-3 Card technical specifications, "Official documentation web site", https://www.raspberrypi.org/documentation/hardware/camera/, last access is 01 Nov. 2020, Farnell Rewark Element I4 manufacturer dataset pp. 1-3, (2017).

[44] A4TECH Fstayler Collection, 1080P HD Webcam (PK-910H) "A4Tech official product specifications web site", https://www.a4tech.com/product.aspx?id=36, last access is 01 Nov. 2020.

[45] Geekworm Company, Manufacturer, "Technical specifications web site for Raspy UPS HAT Board", https://raspberrypiwiki.com/index.php/Raspi, UPS HAT Board, last access is 01 Nov. 2020.

[46] Raspberry Ltd, PI-3 Card technical specifications, "Official documentation web site", https://www.raspberrypi.org/documentation/hardware/display/, last access is 01 Nov. 2020.