**TRAFFIC SIGN RECOGNITION USING CONVOLUTIONAL NEURAL NEYWORKS**

ABSTRACT

Traffic sign recognition (TSR) represents an important feature of advanced driver assistance systems, contributing to the safety of the drivers, pedestrians and vehicles as well. Developing TSR systems requires the use of computer vision techniques, which could be considered fundamental in the field of pattern recognition in general. Nowadays, more and more object recognition tasks are being solved with Convolutional Neural Networks (CNN). Due to its high recognition rate and fast execution, the convolutional neural networks have enhanced most of computer vision tasks, both existing and new ones.

We propose an approach for traffic sign detection based on Convolutional Neural Networks (CNN). The Traffic Sign Recognition project is to build a **Deep Neural Network (DNN)** which is used to classify traffic signs. We should train the model so it can decode traffic signs from natural images using the [German Traffic Sign Dataset](http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset). This data should be firstly pre-processed in order to maximize the model performance. After choosing model architecture, fine tuning and training, the model will be tested on new images of traffic signs.Training of the neural network code is written in **Matlab C** with use of Matlab. It's great for making quick, high-level changes in our model architecture.The experimental results confirmed high efficiency of the developed computer vision system.

CHAPTER 1

# INTRODUCTION

Recently the number of road vehicles has increased enormously thanks to the technological achievements in the motor industry and very precisely the availability of low rates. With this remarkable growth, the number of accidents is as well in an infinite raise year after year, due to different causes, in which the ignorance of traffic signs is considered as a major cause of these lasts. Development of the technical level of modern mobile processors enabled many vehicle producers to install computer vision systems into customer cars. These systems help to significantly improve the safety and implement an important step on the way to autonomous driving. Among other tasks solved with computer vision, the traffic sign recognition (TSR) problem is one of the most well-known and widely discussed by lots of researchers. However, the main problems of such systems are low detection accuracy and high demand for hardware computational performance, as well as the inability of some systems classifies the traffic signs from different countries.

Developing automated traffic sign recognition systems helps assisting the driver in different ways in order to guarantee his/her safety, which preserves as well the safety of other drivers and pedestrians. These systems have one main goal: detecting and recognizing traffic signs during the driving process. With these functionalities the system can guide and alert the drivers to prevent danger. Even though it is possible to develop a system that can recognize traffic signs, it doesn’t mean that any sign can be correctly recognized by the system due to some traffic environmental challenges, for example: lightning variations, bad illumination, weather changes and signs in a ruined condition.

In our modern age, around [1.3M people](http://asirt.org/initiatives/informing-road-users/road-safety-facts/road-crash-statistics) die on roads each year. This number would be much higher without our road signs. Naturally, autonomous vehicles must also abide by road legislation and therefore recognize andunderstand traffic signs. Traditionally, standard [computer vision](https://en.wikipedia.org/wiki/Computer_vision) methods were employed to detect and classify traffic signs, but these required considerable and time-consuming manual work to handcraft important features in images. Traffic sign detection has been a traditional problem for intelligent vehicles, especially as a preceding step for traffic sign recognition which provides useful information such as directions and alerts for autonomous driving or driverassistance systems.

The standard appearances of traffic signs make it efficient and robust to detect and match traffic signs under various conditions, and this forms a primary reason that traffic signs are a preferable choice as landmarks for road map reconstruction.

# BACKGROUND

# Traffic signs are an integral part of our road infrastructure. They provide critical information, sometimes compelling recommendations, for road users, which in turn requires them to adjust their driving behaviour to make sure they adhere with whatever road regulation currently enforced. Without such useful signs, we would most likely be faced with more accidents, as drivers would not be given critical feedback on how fast they could safely go, or informed about road works, sharp turn, or school crossings ahead.

Traffic signs (TS) are generally divided into three maincategories according to theirs functions: regulatory signs to givenotice of traffic laws or regulation, warning signs to give noticeof a situation that might cause danger and finally guide signs toshow information about route destinations, distances…etc. Ineach mentioned TS category, there are different subclasses withsimilar generic shape and appearance but differentdetails.Although the detection of traffic signs has been studied for years, there still exist many challenges. For example, the background clutter may introduce strong disturbances. In addition, the color of traffic sign is very sensitive to lighting conditions (sun, shadow), weather (sunny, rain, snow) and time (morning, noon, night), etc. Last but not least, the partial occlusion dramatically affects the detection performance.

Recently, Convolutional Neural Network has been adopted in object recognition for its high accuracy.

In multi-layer convolutional networks is proposed to boost traffic sign recognition, using a combination of supervised and unsupervised learning. This model can learn multi stages of invariant features of image, with each layer containing a filter bank layer, a non-linear transfor layer, and a spatial feature pooling layer.

**DOMAIN INTRODUCTION**

**DIGITAL IMAGE PROCESSING**

**1.1GENERAL**

Digital image processing is the use of computer algorithms to perform image processing on digital images. The 2D continuous image is divided into N rows and M columns. The intersection of a row and a column is called a pixel. The image can also be a function other variables including depth, color, and time. An image given in the form of a transparency, slide, photograph or an X-ray is first digitized and stored as a matrix of binary digits in computer memory. This digitized image can then be processed and/or displayed on a high-resolution television monitor. For display, the image is stored in a rapid-access buffer memory, which refreshes the monitor at a rate of 25 frames per second to produce a visually continuous display.

* + 1. **THE IMAGE PROCESSING SYSTEM**

FIG 1.1 BLOCK DIAGRAM OF IMAGE PROCESSING SYSTEM

Digitizer

Mass Storage

Hard Copy Device

Display

Image Processor

Digital Computer

Operator Console

* **Digitizer**

Digitizing or digitizationis the representation of an object, image, sound, document or a signal (usually an analog signal) by a discrete set of its points or samples. Digital information exists as one of two digits, either 0 or 1. These are known as bits.

An image is digitized to convert it to a form which can be stored in a computer's memory or on some form of storage media such as a hard disk or CD-ROM. This digitization procedure can be done by a scanner, or by a video camera connected to a frame grabber board in a computer. Once the image has been digitized, it can be operated upon by various image processing operations.

* Microdensitometer
* Flying spot scanner
* Image dissector
* Videocon camera
* Photosensitive solid- state arrays.
* **Digital computer**

A computer is an electronic device that accepts raw data, processes it according to a set of instructions and required to produce the desired result. Mathematical processing of the digitized image such as convolution, averaging, addition, subtraction, etc. are done by the computer.

* **MASS STORAGE**

Mass storage devices used in desktop and most server computers typically have their data organized in a file system.The secondary storage devices normally used are floppy disks, CD ROMs etc.

* **OPERATOR CONSOLE**

The operator console consists of equipment and arrangements for verification of intermediate results and for alterations in the software as and when require. The operator is also capable of checking for any resulting errors and for the entry of requisite data.

* **Display**

Popular display devices produce spots (display elements) for each pixel:

* Cathode ray tubes (CRTs).
* Liquid crystal displays (LCDs).
* Printers.

Spots may be binary (e.g., monochrome LCD), achromatic (e.g., so-called black-and-white, actually grayscale for intensity), pseudo color or false colors (e.g., for intensity or hyper spectral data), or true color (color data displayed as such).

* + 1. **IMAGE PROCESSING FUNDAMENTAL**

Digital image processing refers processing of the image in digital form. Modern cameras may directly take the image in digital form but generally images are originated in optical form. They are captured by video cameras and digitalized. The digitalization process includes sampling, quantization. Then these images are processed by the five fundamental processes, at least any one of them, not necessarily all of them.

**1.1.2.1 Fundamental steps in image processing**

* 1. Image acquisition
  2. Image preprocessing
  3. Image segmentation
  4. Image representation
  5. Image description
  6. Image recognition
  7. Image interpretation
* **Image acquisition**

First we need to produce a digital image from a paper envelope. This can be done using either a CCD camera, or a scanner

* **Image preprocessing**

This is the step taken before the major image processing task. The problem here is to perform some basic tasks in order to render the resulting image more suitable for the job to follow. In this case it may involve enhancing the contrast, removing noise, or identifying regions likely to contain the postcode.

* **Image segmentation**

Segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as super pixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain visual characteristics.

* **Image representation**

Image process is the process of convert the input data to a form suitable for computer processing

* **Image description**

Image description is the process of extract features that result in some quantitative information of interest or features that are basic for differentiating one class of objects from another.

* **Image recognition**

Image recognition is the process of assign a label to an object based on the information provided by its descriptors.

* **Image interpretation**

Image interpretation is the process of assign meaning to an ensemble of recognized objects.

**1.1.2.2 Image types**

There are several ways of encoding the information in an image.

1. Binary image
2. Grayscale image
3. Indexed image
4. True color or RGB image

* **Binary image**

Each pixel is just blackor white. Since there are only two possible values for each pixel (0, 1), we only need one bitper pixel.

* **Grayscale image**

Each pixel is a shade of gray, normally from 0 (black) to 255(white). This range means that each pixel can be represented by eight bits, or exactly one byte. Other grayscale ranges are used, but generally they are a power of 2.

* **Indexed image**

An indexed image consists of an array and a color map matrix. The pixel values in the array are direct indices into a color map. By convention, this documentation uses the variable name X to refer to the array and map to refer to the color map.

* **True Color or RGB image**

Each pixel has a particular color; that color is described by the amount of red, greenand bluein it. If each of these components has a range 0–255, this gives a total of 2563different possible colors. Such an image is a “stack” of three matrices; representing the red, greenand bluevalues for each pixel. This means that for every pixel there correspond 3 values.

**1.1.2.3 image processing goals**

In virtually all image processing applications, however, the goal is to extract information from the image data. Obtaining the information desired may require filtering, transforming, coloring, interactive analysis, or any number of other methods.

To be somewhat more specific, one can generalize most image processing tasks to be characterized by one of the following categories:

Problem Domain

Knowledge

Base

Segmentation

Preprocessing

Image Acquisition

Recognition & interpretation

Representation & Description

Result

**FIG 1.2 BLOCK DIAGRAM OF FUNDAMENTAL SEQUENCE INVOLVED IN AN IMAGE PROCESSING SYSTEM**

1. Image enhancement
2. Image restoration
3. Image analysis
4. Feature extraction
5. Image registration
6. Image compression
7. Image synthesis

* **image enhancement**

This simply means improvement of the image being viewed to the (machine or human) interpreter's visual system. Image enhancement types of operations include contrast adjustment, noise suppression filtering, application of pseudo color, edge enhancement, and many others.

* **image restoration**

The purpose of image restoration is to "compensate for" or "undo" defects which degrade an image. Degradation comes in many forms such as motion blur, noise, and camera misfocus. In cases like motion blur, it is possible to come up with a very good estimate of the actual blurring function and "undo" the blur to restore the original image. In cases where the image is corrupted by noise, the best we may hope to do is to compensate for the degradation it caused.

* **image analysis**

Image analysis is the extraction of meaningful information from images. Image analysis operations produce numerical or graphical information based on characteristics of the original image. They break into objects and then classify them. They depend on the image statistics. Common operations are extraction and description of scene and image features, automated measurements, and object classification. Image analyze are mainly used in machine vision applications.

* **feature extraction**

Feature extraction involves simplifying the amount of resources required to describe a large set of data accurately. When performing analysis of complex data one of the major problems stems from the number of variables involved. Analysis with a large number of variables generally requires a large amount of memory and computation power or a [classification](http://en.wikipedia.org/wiki/Statistical_classification) algorithm which [over fits](http://en.wikipedia.org/wiki/Overfitting) the training sample and generalizes poorly to new samples. Feature extraction is a general term for methods of constructing combinations of the variables to get around these problems while still describing the data with sufficient accuracy.

* **image registration**

Image registration is the process of overlaying two or more images of the same scene taken at different times, from different viewpoints, and/or by different sensors. It geometrically aligns two images the reference and sensed images. The present differences between images are introduced due to different imaging conditions. Image registration is a crucial step in all image analysis tasks in which the final information is gained from the combination of various data sources like in image fusion, change detection, and multichannel image restoration.

Typically, registration is required in remote sensing (multispectral classification, environmental monitoring, change detection, image mosaicing, weather forecasting, creating super-resolution images, integrating information into geographic information systems (GIS)), in medicine (combining computer tomography (CT) and NMR data to obtain more complete information about the patient, monitoring tumor growth, treatment verification, comparison of the patient’s data with anatomical atlases), in cartography (map updating), and in computer vision (target localization, automatic quality control), to name a few.

* **image compression**

The objective of image compression is to reduce irrelevance and redundancy of the image data in order to be able to store or transmit data in an efficient form.Image compression may be lossy or lossless. Lossless compression is preferred for archival purposes and often for medical imaging, technical drawings, clip art, or comics. This is because lossy compression methods, especially when used at low bit rates, introduce compression artifacts. Lossy methods are especially suitable for natural images such as photographs in applications where minor (sometimes imperceptible) loss of fidelity is acceptable to achieve a substantial reduction in bit rate. The lossy compression that produces imperceptible differences may be called visually lossless.

* **image synthesis**

Image synthesis operations create images from other images or non-image data. Image synthesis operations generally create images that are either physically impossible or impractical to acquire.

**1.1.2**.**4 Applications of image processing**

Image processing has an enormous range of applications; almost every area of science and technology can make use of image processing methods. Here is a short list just to give some indication of the range of image processing applications.

* **Medicine**

Inspection and interpretation of images obtained from X-rays, MRI or CAT scans, analysis of cell images, of chromosome karyotypes. In medical applications, one is concerned with processing of chest X-rays, cineangiograms, projection images of transaxial tomography and other medical images that occur in radiology, nuclear magnetic resonance (NMR) and ultrasonic scanning. These images may be used for patient screening and monitoring or for detection of tumors’ or other disease in patients.

* **Agriculture**

Satellite/aerial views of land, for example to determine how much land is being used for different purposes, or to investigate the suitability of different regions for different crops, inspection of fruit and vegetables distinguishing good and fresh produce from old.

* **DOCUMENT PROCESSING**

It is used in scanning, and transmission for converting paper documents to a digital image form, compressing the image, and storing it on magnetic tape. It is also used in document reading for automatically detecting and recognizing printed characteristics.

* **RADAR IMAGING SYSTEM**

Radar and sonar images are used for detection and recognition of various types of targets or in guidance and maneuvering of aircraft or missile systems.

* **DEFENSE/INTELLIGENCE**

It is used in reconnaissance photo-interpretation for automatic interpretation of earth satellite imagery to look for sensitive targets or military threats and target acquisition and guidance for recognizing and tracking targets in real-time smart-bomb and missile-guidance systems.

CHAPTER 2

**LITERATURE SURVEY AND OUTCOME OF EXISTING LITERATURE**

This chapter briefly reviews, explains and discusses on existing literature review related with the current project which is “Traffic sign recognition using convolutional neural networks” that will be developed later. This chapter comprises three sections. The first section describes the overviews of Traffic sign. The subsections are the definition, type of Traffic sign. The second section is the review of some existing system that used same techniques and methods.

# Traffic sign Overviews

In this section, Traffic signs are the silent speakers on the road. Be it the person behind the wheel or a pedestrian, having a sound knowledge about road safety is absolutely necessary for all before hitting the roads.

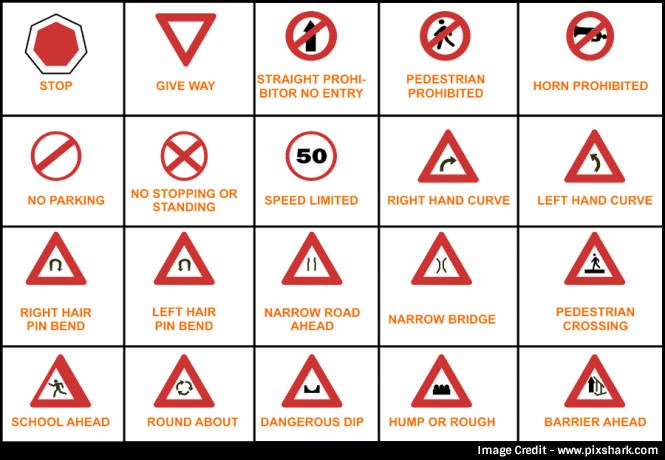
# Definition of Traffic sign

Traffic signs give information about the road conditions ahead, provide instructions to be followed at the major crossroads or junctions, warn or guide drivers, and ensure proper functioning of road traffic. Being unaware of road signs is akin to throwing caution to the wind. It can lead to loss of life and property. A person is supposed to be familiar (get through a written or oral test) with the traffic signs and symbols before acquiring a driving license in India.

**Road safety signs are primarily of three types:**

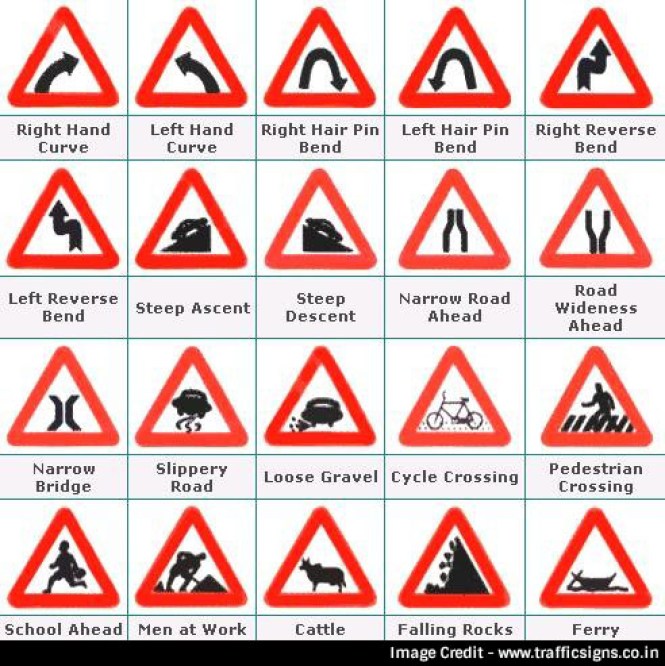
**Mandatory Signs:**

These signs are used to ensure free movement of traffic and make the road users cognisant of certain laws and regulations, restrictions and prohibitions. Violation of these signs is an offence, as per law.



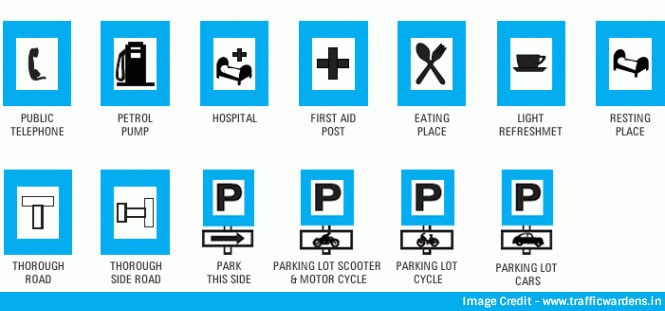
**Cautionary Signs:**

These signs make the road users conscious of hazardous conditions on the road beforehand. The drivers, accordingly, take necessary actions to handle the situation.



**Informatory Signs:**

These signs guide the road users about destinations, distance, alternative routes, and prominent locations like food joints, public toilets, nearby hospitals, etc.



**TITLE:** CNN Design for Real-Time Traffic Sign Recognition

**AUTHOR:** Alexander Shustanov, Pavel Yakimov

**YEAR: 2017**

**DESCRIPTION:**

Nowadays, more and more object recognition tasks are being solved with Convolutional Neural Networks (CNN). Due to its high recognition rate and fast execution, the convolutional neural networks have enhanced most of computer vision tasks, both existing and new ones. In this article, we propose an implementation of traffic signs recognition algorithm using a convolution neural network. The paper also shows several CNN architectures, which are compared to each other. Training of the neural network is implemented using the TensorFlow library and massively parallel architecture for multithreaded programming CUDA. The entire procedure for traffic sign detection and recognition is executed in real time on a mobile GPU. The experimental results confirmed high efficiency of the developed computer vision system.

**TITLE:** Towards Real-Time Traffic Sign Detection and Classification

**AUTHOR:** Yi Yang, Hengliang Luo, Huarong Xu, and Fuchao Wu

**YEAR: 2015**

**DESCRIPTION:** Traffic sign recognition plays an important role in driver assistant systems and intelligent autonomous vehicles. Its real-time performance is highly desirable in addition to its recognition performance. This paper aims to deal with real-time traffic sign recognition, i.e., localizing what type of traffic sign appears in which area of an input image at a fast processing time. To achieve this goal, we first propose an extremely fast detection module, which is 20 times faster than the existing best detection module. Our detection module is based on traffic sign proposal extraction and classification built upon a color probability model and a color HOG. Then, we harvest from a convolutional neural network to further classify the detected signs into their subclasses within each superclass. Experimental results on both German and Chinese roads show that both our detection and classification methods achieve comparable performance with the state-of-the-art methods, with significantly improved computational efficiency.

**TITLE:** Traffic Sign Detection based on Convolutional Neural Networks

**AUTHOR:** Yihui Wu, Yulong Liu, Jianmin Li, Huaping Liu, Xiaolin Hu

**YEAR: 2013**

**DESCRIPTION:** Proposed an approach for traffic sign detection based on Convolutional Neural Networks (CNN). We first transform the original image into the gray scale image by

using support vector machines, then use convolutional neural networks with fixed and learnable layers for detection and recognition. The fixed layer can reduce the amount of interest areas to detect, and crop the boundaries very close to the borders of traffic signs. The learnable layers can increase the accuracy of detection significantly. Besides, we use bootstrap methods to improve the accuracy and avoid over fitting problem. In the German Traffic Sign Detection Benchmark, we obtained competitive results, with an area under the precision-recall curve (AUC) of 99.73% in the category “Danger”, and an AUC of 97.62% in the category “Mandatory”.

**TITLE:** Multi-Column Deep Neural Network for Traffic Sign Classification.

**AUTHOR:** Dan Cire¸san, Ueli Meier, Jonathan Masci and J¨urgen Schmidhuber

**YEAR: 2012**

**DESCRIPTION:** This Paper describe the approach that won the final phase of the German traffic sign recognition benchmark. Our method is the only one that achieved a betterthan-human recognition rate of 99.46%. We use a fast, fully parameterizable GPU implementation of a Deep Neural Network (DNN) that does not require careful design of pre-wired feature extractors, which are rather learned in a supervised way. Combining various DNNs trained on differently preprocessed data into a Multi-Column DNN (MCDNN) further boosts recognition performance, making the system insensitive also to variations in contrast and illumination.

**TITLE:** Traffic Sign Recognition with Multi-Scale Convolutional Networks

**AUTHOR:** Pierre Sermanet and Yann LeCun.

**YEAR: 2011**

**DESCRIPTION:** —In This paper, apply Convolutional Networks (ConvNets) to the task of traffic sign classification as part of the GTSRB competition. ConvNets are biologically-inspired multi-stage architectures that automatically learn hierarchies of invariant features. While many popular vision approaches use handcrafted features such as HOG or SIFT, ConvNets learn features at every level from data that are tuned to the task at hand. The traditional ConvNet architecture was modified by feeding 1 st stage features in addition to 2 nd stage features to the classifier. The system yielded the 2nd-best accuracy of 98.97% during phase I of the competition (the best entry obtained 98.98%), above the human performance of 98.81%, using 32x32 color input images. Experiments conducted after phase 1 produced a new record of 99.17% by increasing the network capacity, and by using greyscale images instead of color. Interestingly, random features still yielded competitive results (97.33%).

CHAPTER 3

**PROBLEM STATEMENT OF PROJECT**

Although the detection of traffic signs has been studied for years, there still exist many challenges. For example, the background clutter may introduce strong disturbances. In addition, the color of traffic sign is very sensitive to lighting conditions (sun, shadow), weather (sunny, rain, snow) and time (morning, noon, night), etc. Last but not least, the partial occlusion dramatically affects the detection performance.

**MOTIVATION**

Traffic Sign Recognition is a basic, day by day task for self-driving cars. Recently, traffic sign detection has received another attention from navigation systems for intelligent vehicles, where traffic signs can be used as distinct landmarks for mapping and localization. Different from natural landmarks such as corner points or edges which have arbitrary appearance, traffic signs have standard appearances such as shapes, colors, and patterns defined by strict regulations. The standard appearances of traffic signs make it efficient and robust to detect and match traffic signs under various conditions, and this forms a primary reason that traffic signs are a preferable choice as landmarks for road map reconstruction.

**OBJECTIVES AND SCOPE OF PROJECT**

The objective is to learn, recognize and classify traffic signs. A German traffic sign database is used for the learning. As is immediately evident, the images have varying levels of clarity. All images are classified and labelled. The goal, hence, is for the system to extract and learn the underlying features that uniquely identify a traffic sign and recognize it when it appears in an image. Eventually, it should classify that sign into one of 16 categories. In order to make it happen, a **Convolutional Neural Network** is applied, being trained with more than 450 images beforehand.

**CHAPTER 4**

**EXISTING SYSTEM AND PROPOSED SYSTEM METHODOLOGY**

**EXISTING SYSTEM**

There are many researches in the literature dealing with Road Traffic Sign Recognition (TSR) problem. The very first work on automated traffic sign detection was presented in Japan in 1984. Different researchers introduced several methods afterwards, to develop an efficient traffic sign recognition and detection (TSDR) system and to minimize all of the issues mentioned above. An efficient TSDR system can be divided into several phases, starting with preprocessing, detection, tracking, and recognition.

Most of the existing systems on traffic sign detection rely on hand-crafted image features to identify target signs. Where colour probability maps and multiple thresholding algorithms were used. Also Region-based features like histogram of gradient (HoG), are widely used where accurate decision on traffic sign/non-traffic sign is required. More sophisticated features such as the integral channel features (ICF) or the aggregated channel features (ACF) have also been applied to traffic sign recognition systems.

The last step in any TSDR system as we mentioned is recognition. Techniques used the template matching technique, which is a straight forward, and a very fast method and a genetic algorithm to deal with the illumination problem. Also AdaBosst that is known and confirmed with its simplicity, generalization, and feature selection for large dataset. Greenhalgh and Mirmehdi studied four important classifiers: SVM, MLP, HOG based and Decision Trees. After a comparison between these classifiers, they found that a Decision Tree has the highest accuracy rate and the lowest computational time.

In fact it is impossible to discuss recognition algorithms and approaches without mentioning CNNs. After their use in for image classification, the interest in using CNN was rekindled and quickly researchers started adapting CNNs more and more for object detection and recognition.

**EXISTING SYSTEM DRAWBACKS**

* Used hand-Crafted algorithms for the detection are features designed by hand are known to be weaker than features learned by CNN from massive training data.
* A simple classification algorithm like template matching was not able to achieve high-quality recognition because of a limited set of predefined templates.
* Accuracy is less.
* Very slow speed of the network.

**PROPOSED SYSTEM**

The main objective of our project is to design and construct a computer based system which can automatically recognise the road signs so as to provide assistance to the user or the machine so that they can take appropriate actions. The proposed approach consists of building a model using convolutional neural networks by extracting traffic signs. We have used convolutional neural networks (CNN) to classify the traffic signs.

A CNN is basically inspired by the connections between the neurons in the visual cortex of animals. The learning rate used to train the CNN was 0.0001. The CNN was trained for 20 iterations (magic numbers). Once the CNN has been trained, it is used to predict the sign of the contours obtained. Each of these contours are assigned the sign with the maximum probability which is the output of the CNN.

**PROPOSED SYSTEM BLOCK DIAGRAM**

**Trained Database**

**Traffic Sign**

**Image Acquisition**

**Classification(CNN)**

**Image Pre-processing**

**Fig 1: Block Diagram of the Proposed System**

**PROPOSED SYSTEM ADVANTAGES**

* Our proposed method shows better performance.
* The system shows robust result than some existing methods.
* The CNN is doing a good job in classifying different types of traffic signs.
* Training can update all network layers.

**PROJECT SETUP**

The dataset is split into training, test and validation sets, with the following characteristics:

* Images are 120 (width) x 120(height) x 3 (RGB color channels)
* Training set is composed of 320 images
* Validation set is composed of 50 images
* Test set is composed of 136 images
* There are 43 classes (e.g. Speed Limit 20km/h, No entry, Bumpy road, etc.)

### PROJECT PIPELINE

Pipeline consists of 7 steps, which are quite common in classification problems:

1. Data pre-processing.
2. Loading the data.
3. Dataset exploration and visualization.
4. Data augmentation.
5. Designing, training and testing a CNN model.
6. Using the model on new images.
7. Analyzing softmax probabilities.

**CHAPTER 4**

**SYSTEM REQUIREMENT SPECIFICATION**

System requirement specifications gathered by extracting the appropriate information to implement the system. It is the elaborative conditions which the system need to attain. Moreover, the SRS delivers a complete knowledge of the system to understand what this project is going to achieve without any constraints on how to achieve this goal. This SRS not providing the information to outside characters but it hides the plan.

**4.1 Hardware Requirements**

The necessary hardware regarding private PC that comprises configuration as specified as follows:-

1. Processor: Intel core i5.

2. Disk capability: 1GB for MATLAB only.

3. RAM: 2GB.

**4.2 Software Tool used**

The necessary program regarding private PC that comprises configuration as specified as follows:-

1. Windows 7(64-bit) operating system.

2. MATLAB 7.14 Version R2014a

.

**4.2.1 Overview of MATLAB**

MATLAB is built up by means of math works intended for fourth-generation programming language. A variety of process approved within MATLAB contains control concerning the matrix, purpose as well as plotting of data, execution regarding algorithms, design of user interface, as well as integrating by means of programs formed within other languages like C, C++, and java. Despite mathematical calculation, MATLAB can be meant for representational calculation as well. MATLAB can be meant for embedded methods and by the guide regarding extra package known as simulink. Specifically MATLAB permit intended for matrix estimation as well as thus can be intended for image processing. MATLAB is simple towards gaining knowledge of a variety of device boxes used for it; an illustration is image processing toolbox.

MATLAB interfaces programming surroundings, calculation as well as mental picture. This contains integrated correcting, data compositions as well as object-oriented correcting devices. These integrated tasks create MATLAB appropriate used for education as well as do research. To resolve scientific trouble MATLAB includes other benefits than usual programming language like c plus java. MATLAB arrived into promotion in 1984 in addition to now it is employed globally. Additional graphical instructions are offered within MATLAB that builds the visual effects obtainable right away. A variety of device box contains signal processing, simulation, control theory as well as some former that are employed extensively in science and technology. The lone disadvantage regarding MATLAB is expenditure worry.

**4.2.2 Image Processing Toolbox**

Image processing device box permits carrying out image improvement, deblurring of image, characteristic identification, decreasing of noise, image segmentation, arithmetical alteration, as well as registration of image. Image processing device intended for the execution regarding methods proposed are specified below:-

1. Fundamental import as well as export

2. Display

3. Thresholding

**Fundamental import as well as export functions**

Fundamental import as well as export functions permits images obtained by means of image accomplishment plans for example, digital cameras, medical imaging devices such as CT and MRI, microscopes, satellite and airborne sensors, telescopes, and other scientific instruments. Hence those images can be observed; analyzed, as well as process these images into numerous data types, together with single-accuracy as well as double-accuracy floating-point in addition to signed as well as unsigned 8-bit, 16-bit, plus 32-bit integers. Import as well as export functions are accustomed to carry out read as well as write process on images.

**Display function**

Display purpose is accustomed to illustrate the images that are interpreted by means of the import purpose. This purpose permits towards making displays by means of graphics as well as wording, images within a particular window as well as specific displays for example outline plot, histogram and so on.

**Thresholding**

Thresholding is a simple system concerning image segmentation. As of a gray scale image, thresholding can be able to be accustomed to generate binary images. Within thresholding section, the intensity not more than an appropriate value within input image will be displayed as black (intensity is zero) as well as the left out intensities will be made white (intensity is one) then displayed. This procedure is done in the direction of obtaining the segmented image.

**4.2.3 Features of Matlab**

* Interactive background meant for aim investigation as well as resolving the difficulty.
* MATLAB is a sophisticated language intended for creating, calculating as well as building up a purpose.
* It contains numerical tasks such as figures, calculus, sorting out, developments, mathematical integration, as well as working out equations.
* Graphics integrated intended for visualization.
* Intended for generating traditional plot integrated equipments is accessible.
* Troubles as well as way outs are given in well-known numerical symbol.

**4.2.4 SOFTWARE REQUIREMENTS**

* MATLAB 8.3 Version R2014a

**MATLAB**

The MATLAB high-performance language for technical computing integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation.

* Data Exploration ,Acquisition, Analyzing &Visualization
* Engineering drawing and Scientific graphics
* Analyzing of algorithmic designing and development
* Mathematical functions and Computational functions
* Simulating problems prototyping and modeling
* Application development programming using GUI building environment.

Using MATLAB, you can solve technical computing problems faster than with traditional programming languages, such as C, C++, and Fortran.

**CHAPTER 5**

**SYSTEM DESIGN**

**PROPOSED SYSTEM BLOCK DIAGRAM**

**DATASET**

**PRE-PROCESSING**

**PROCESSED DATA**

**FIG. PRE-PROCESSING**

**PROCESSED DATA**

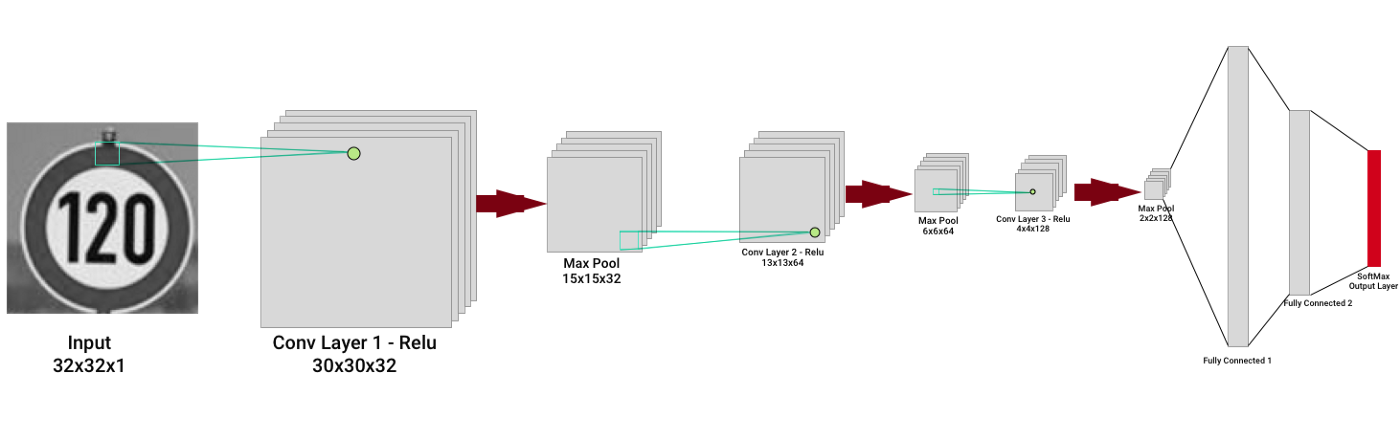
**CLASSIFIED OUTPUT**

**CNN\_LAYERS**

**COVN\_LAYER**

**POOL\_LAYER**

**FULLY CONNECTED\_LAYER**



**FIG. DETAILED PROPOSED CNN ARCHTECTURE.**

In this method two parts important. First part is training part. In which database formation is important work .Second one is testing part. In which actual classification takes place.

1. **Training Part**

In training part followed a simple, iterative process of finding the best model architecture. After changing one of the model parameters, I ran only 20 epochs of the training and observed the validation error trying to set it on minimum level. It is very important to consider mainly validation error while tuning the model. The corresponding validation accuracies after 20 epochs of training. Sometimes the differences between two given approaches seemed to be huge and it was hard to make a choice between them. But note that for each training procedure there is a random weight initialization which influences the final error. Especially when number of epochs is small. That's why during final model tuning I used more than 20 epochs - about 100.

1. **Testing Part**

Finally, we would like to test our Traffic Sign Recognition system on completely unseen sign images. Surely, the accuracy obtained on the test set is also a very good indication of the model performance. But let's find some new images which are not from our German Traffic Sign Dataset. Under images, there are model predictions, indication if the prediction is correct and model certainty.Implementation of algorithm may be by using MATLAB.

**5.1 Data Flow Diagram**

A data flow diagram (DFD) is an illustration in a graphical form. It is an illustration concerning the "run" of information all the way through in a sequence, representing its procedure part. Frequently they are a beginning move employed towards building a general idea regarding the method that will be detailed later on. A DFD demonstrates the variety of input data in sequence to with output as of the method, in which the information will approach as of plus exit towards, in addition to in what the information will be accumulated. It doesn’t demonstrate in sequence regarding the instance of procedures, or else in sequence regarding whether the procedures will work sequentially or else separately.

**FLOW CHART:**

TRAINING PHASE

**CONVNET**

**CLASSIFICATION (using SVM)**

**TRAIN\_SYSTEM**

**RESULT**

**IMAGE ACQUISITION**

**DATASET ACQUISITION**

**SPLIT TRAING AND TESTING DATASETS**

RESIZING

**INPUT\_LAYER (120\*120)**

**MAX\_POOL\_LAYER (2\*2)**

**FULLY CONNECTED\_LAYER (16)**

**COVN\_LAYER (5\*5), 20**

**MAX\_POOL\_LAYER (2\*2)**

**COVN\_LAYER (5\*5), 20**

**SOFTMAX**

Training Option(SGDM)

TESTING PHASE

**ACCURACY**

In training and testing part following procedure is followed

**DATASET**

The dataset is divided into training set (320 samples) and test set (136 samples). Each sample represents a traffic sign labeled as one of 16 classes. It can be e.g. a stop sign, yield, 30 km/h speed limit etc. The shape of a traffic sign image is scaled to 120×120 pixels in 3 channel RGB representation (120×120×3).

Below, there are a few random samples from the dataset:



**Figure. Traffic signs classes of the DATASET.**

We should firstly explore the dataset, understand it against the problem to solve. Let's see how many samples we have here for each traffic sign class. We wouldn't like the model to be biased towards any of the class.

Right now, we can see that some labels are greatly underrepresented while others have quite many representatives in the dataset.  The subset of images which are belonging to the same class.

[](https://tomaszkacmajor.pl/wp-content/uploads/2017/10/data.png)We can observe that images from the same class can be represented quite differently in the dataset. Generally, there can be different **lighting conditions**, image can be **blurred**, **rotated** or **scaled**. Indeed, these are samples which are extracted from real world images. And our model has to handle all of these conditions. So, it's probably better not to truncate our dataset in order to obtain data balance.

### DESIGNING A DEEP NEURAL NETWORK MODEL

Now, it's finally time to feed the data to the neural network. Choosing the network architecture, tuning different parameters again and again is probably the most demanding task. There are no clear rules for the model optimization. Besides some proven rule of thumbs, our experience often plays a big role. Furthermore, when dealing with deep neural networks you have to wait for the results of each tested model a relatively long time.

#### CONVOLUTION NEURAL NETWORK

the concept of convolutional neural networks. They are very successful in image recognition. The key part to understand, which distinguishes CNN from traditional neural networks, is the **convolution** operation. Having an image at the input, CNN scans it many times to look for certain **features**. This scanning (convolution) can be set with 2 main parameters: stride and padding type. As we see on below picture, process of the first convolution gives us a set of new frames, shown here in the second column (layer). Each frame contains an information about one feature and its presence in scanned image. Resulting frame will have larger values in places where a feature is strongly visible and lower values where there are no or little such features. Afterwards, the process is repeated for each of obtained frames for a chosen number of times. In this project we chose a classic model which contains only two convolution layers.

The latter layer we are convolving, the more high-level features are being searched. It works similarly to human perception. To give an example, below is a very descriptive picture with features which are searched on different CNN layers. As you can see, the application of this model is face recognition. You may ask how the model knows which features to seek. If you construct the CNN from the beginning, searched features are random. Then, during training process, weights between neurons are being adjusted and slowly CNN starts to find such features which enable to meet predefined goal, i.e. to recognize successfully images from the training set.

Between described layers there are also **pooling** (sub-sampling) operations which reduce dimensions of resulted frames. Furthermore, after each convolution we apply a non-linear function (called **ReLU**) to the resulted frame to introduce non-linearity to the model.

Eventually, there are also **fully connected layers** at the end of the network. The last set of frames obtained from convolutional operations is flattened to get a one-dimensional vector of neurons. From this point we put a standard, fully-connected neural network. At the very end, for classification problems, there is a **softmax** layer. It transforms results of the model to probabilities of a correct guess of each class, here a traffic sign index.

Below, there is a summary of the model I chose and fine-tuned with marked dimensions for each layer.

**10x1 Layer array with layers:**

**1 '' Image Input 120x120x3 images with 'zerocenter' normalization**

**2 '' Convolution 20 5x5 convolutions with stride [1 1] and padding [0 0]**

**3 '' ReLU ReLU**

**4 '' Max Pooling 2x2 max pooling with stride [2 2] and padding [0 0]**

**5 '' Convolution 20 5x5 convolutions with stride [1 1] and padding [0 0]**

**6 '' ReLU ReLU**

**7 '' Max Pooling 2x2 max pooling with stride [2 2] and padding [0 0]**

**8 '' Fully Connected 16 fully connected layer**

**9 '' Softmax softmax**

**10 '' Classification Output crossentropyex**

#### TUNING THE MODEL

Followed a simple, iterative process of finding the best model architecture. After changing one of the model parameters, I ran only 20 epochs of the training and observed the validation error trying to set it on minimum level. It is very important to consider mainly validation error while tuning the model. Minimizing only the error based on training data can easily lead to unwanted model [overfitting](https://en.wikipedia.org/wiki/Overfitting" \t "_blank).

Below, there are details of intermediate steps that we took and the corresponding validation accuracies after 20 epochs of training. Sometimes the differences between two given approaches seemed to be huge and it was hard to make a choice between them. But note that for each training procedure there is a random weight initialization which influences the final error. Especially when number of epochs is small. That's why during final model tuning I used more than 20 epochs - about 100.

### TESTING THE MODEL ON NEW IMAGES

Finally, we would like to test our Traffic Sign Recognition system on completely unseen sign images. Surely, the accuracy obtained on the test set is also a very good indication of the model performance. But let's find some new images which are not from our German Traffic Sign Dataset. Under images, there are model predictions, indication if the prediction is correct and model certainty.

CHAPTER 6

**FUNCTIONAL MODULES**

1. **Training Phase**

Some disease affected paddy leaf images are used to train the SVM. The steps involved:

1. **Data Acquisition:**

The dataset is divided into training set (320 samples) and test set (136 samples). Each sample represents a traffic sign labeled as one of 16 classes. It can be e.g. a stop sign, yield, 30 km/h speed limit etc. The shape of a traffic sign image is scaled to 120×120 pixels in 3 channel RGB representations (120×120×3).

**MODULE 1**

**LAYER DEFINATION**

In [deep learning](https://en.wikipedia.org/wiki/Deep_learning), a convolutional neural network (CNN, or ConvNet) is a class of [deep neural networks](https://en.wikipedia.org/wiki/Deep_neural_network), most commonly applied to analysing visual imagery.

A neural network is a system of hardware and/or software patterned after the operation of neurons in the human brain. Traditional neural networks are not ideal for image processing and must be fed images in reduced-resolution pieces. CNN have their “neurons” arranged more like those of the frontal lobe, the area responsible for processing visual stimuli in humans and other animals. The layers of neurons are arranged in such a way as to cover the entire visual field avoiding the piecemeal image processing problem of traditional neural networks.



A CNN uses a system much like a multilayer [perceptron](https://whatis.techtarget.com/definition/perceptron) that has been designed for reduced processing requirements. The layers of a CNN consist of an input layer, an output layer and a hidden layer that includes multiple convolutional layers, pooling layers, fully connected layers and normalization layers. The removal of limitations and increase in efficiency for image processing results in a system that is far more effective, simpler to trains limited for image processing and natural language processing.

The first step of creating and training a new convolutional neural network (ConvNet) is to define the network architecture. The network architecture can vary depending on the types and numbers of layers included. The types and number of layers included depends on the particular application or data. For example, if you have categorical responses, you must have a softmax layer and a classification layer, whereas if your response is continuous, you must have a regression layer at the end of the network. A smaller network with only one or two convolutional layers might be sufficient to learn on a small number of grayscale image data. On the other hand, for more complex data with millions of colored images, you might need a more complicated network with multiple convolutional and fully connected layers.

Layers of CNN

* + - 1. Input layer
      2. [Convolutional Layer](http://cs231n.github.io/convolutional-networks/#conv)
      3. [Normalization Layer](http://cs231n.github.io/convolutional-networks/#norm)
      4. [Pooling Layer](http://cs231n.github.io/convolutional-networks/#pool)
      5. [Fully-Connected Layer](http://cs231n.github.io/convolutional-networks/#fc)

layers is an array of Layer objects. You can then use layers as an input to the training function trainNetwork.

### Image Input Layer

Create an image input layer using [imageInputLayer](https://ch.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.imageinputlayer.html).

An image input layer inputs images to a network and applies data normalization.

Specify the image size using the inputSize argument. The size of an image corresponds to the height, width, and the number of color channels of that image. For example, for a grayscale image, the number of channels is 1, and for a color image it is 3.

### Convolutional Layer

A 2-D convolutional layer applies sliding convolutional filters to the input. Create a 2-D convolutional layer using [convolution2dLayer](https://ch.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.convolution2dlayer.html).

The convolutional layer consists of various components.[[1](https://ch.mathworks.com/help/deeplearning/ug/layers-of-a-convolutional-neural-network.html;jsessionid=86b40ad18c5dd522e53e961a7d93" \l "ftn.d117e3004)]

#### Filters and Stride

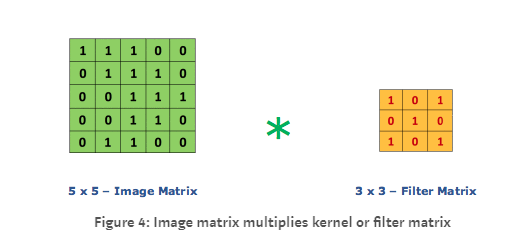
A convolutional layer consists of neurons that connect to subregions of the input images or the outputs of the previous layer. The layer learns the features localized by these regions while scanning through an image. When creating a layer using the [convolution2dLayer](https://ch.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.convolution2dlayer.html) function, you can specify the size of these regions using the filterSize input argument.

For each region, the trainNetwork function computes a dot product of the weights and the input, and then adds a bias term. A set of weights that is applied to a region in the image is called a filter. The filter moves along the input image vertically and horizontally, repeating the same computation for each region. In other words, the filter convolves the input.

This image shows a 3-by-3 filter scanning through the input. The lower map represents the input and the upper map represents the output.

The step size with which the filter moves is called a *stride*. You can specify the step size with the Stride name-value pair argument. The local regions that the neurons connect to can overlap depending on the filterSize and 'Stride' values.

This image shows a 3-by-3 filter scanning through the input with a stride of 2. The lower map represents the input and the upper map represents the output.



The number of weights in a filter is h \* w \* c, where h is the height, and w is the width of the filter, respectively, and c is the number of channels in the input. For example, if the input is a color image, the number of color channels is 3. The number of filters determines the number of channels in the output of a convolutional layer. Specify the number of filters using the numFilters argument with the convolution2dLayer function.

#### Dilated Convolution

A dilated convolution is a convolution in which the filters are expanded by spaces inserted between the elements of the filter. Specify the dilation factor using the 'DilationFactor' property.

Use dilated convolutions to increase the receptive field (the area of the input which the layer can see) of the layer without increasing the number of parameters or computation.

The layer expands the filters by inserting zeros between each filter element. The dilation factor determines the step size for sampling the input or equivalently the upsampling factor of the filter. It corresponds to an effective filter size of (Filter Size – 1) .\* Dilation Factor + 1. For example, a 3-by-3 filter with the dilation factor [2 2] is equivalent to a 5-by-5 filter with zeros between the elements.

This image shows a 3-by-3 filter dilated by a factor of two scanning through the input. The lower map represents the input and the upper map represents the output.

#### Feature Maps

As a filter moves along the input, it uses the same set of weights and the same bias for the convolution, forming a feature map. Each feature map is the result of a convolution using a different set of weights and a different bias. Hence, the number of feature maps is equal to the number of filters. The total number of parameters in a convolutional layer is ((h\*w\*c + 1)\*Number of Filters), where 1 is the bias.

#### Zero Padding

You can also apply zero padding to input image borders vertically and horizontally using the 'Padding' name-value pair argument. Padding is rows or columns of zeros added to the borders of an image input. By adjusting the padding, you can control the output size of the layer.

This image shows a 3-by-3 filter scanning through the input with padding of size 1. The lower map represents the input and the upper map represents the output.

#### Output Size

The output height and width of a convolutional layer is (Input Size – ((Filter Size – 1)\*Dilation Factor + 1) + 2\*Padding)/Stride + 1. This value must be an integer for the whole image to be fully covered. If the combination of these options does not lead the image to be fully covered, the software by default ignores the remaining part of the image along the right and bottom edges in the convolution.

#### Number of Neurons

The product of the output height and width gives the total number of neurons in a feature map, say Map Size. The total number of neurons (output size) in a convolutional layer is Map Size\*Number of Filters.

For example, suppose that the input image is a 32-by-32-by-3 color image. For a convolutional layer with eight filters and a filter size of 5-by-5, the number of weights per filter is 5 \* 5 \* 3 = 75, and the total number of parameters in the layer is (75 + 1) \* 8 = 608. If the stride is 2 in each direction and padding of size 2 is specified, then each feature map is 16-by-16. This is because (32 – 5 + 2 \* 2)/2 + 1 = 16.5, and some of the outermost zero padding to the right and bottom of the image is discarded. Finally, the total number of neurons in the layer is 16 \* 16 \* 8 = 2048.

Usually, the results from these neurons pass through some form of nonlinearity, such as rectified linear units (ReLU).

#### Learning Parameters

You can adjust the learning rates and regularization options for the layer using name-value pair arguments while defining the convolutional layer. If you choose not to specify these options, then trainNetwork uses the global training options defined with the trainingOptions function.

#### Number of Layers

A convolutional neural network can consist of one or multiple convolutional layers. The number of convolutional layers depends on the amount and complexity of the data.

### ReLU Layer

Create a ReLU layer using [reluLayer](https://ch.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.relulayer.html).

A ReLU layer performs a threshold operation to each element of the input, where any value less than zero is set to zero.

Convolutional and batch normalization layers are usually followed by a nonlinear activation function such as a rectified linear unit (ReLU), specified by a ReLU layer. A ReLU layer performs a threshold operation to each element, where any input value less than zero is set to zero, that is,

*f*(*x*)={*x*,0,*x*

*x*≥0<0.

The ReLU layer does not change the size of its input.

There are other nonlinear activation layers that perform different operations and can improve the network accuracy for some applications

### Max and Average Pooling Layers

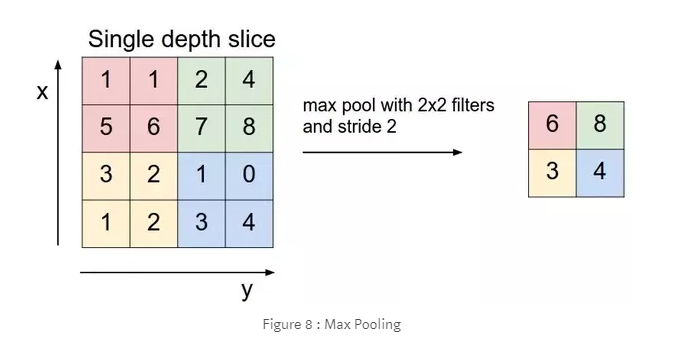
A max pooling layer performs down-sampling by dividing the input into rectangular pooling regions, and computing the maximum of each region. Create a max pooling layer using [maxPooling2dLayer](https://ch.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.maxpooling2dlayer.html).

* An average pooling layer performs down-sampling by dividing the input into rectangular pooling regions and computing the average values of each region. Create an average pooling layer using [averagePooling2dLayer](https://ch.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.averagepooling2dlayer.html).

Pooling layers follow the convolutional layers for down-sampling, hence, reducing the number of connections to the following layers. They do not perform any learning themselves, but reduce the number of parameters to be learned in the following layers. They also help reduce overfitting.

* A max pooling layer returns the maximum values of rectangular regions of its input. The size of the rectangular regions is determined by the poolSize argument of maxPoolingLayer. For example, if poolSize equals [2,3], then the layer returns the maximum value in regions of height 2 and width 3. An average pooling layer outputs the average values of rectangular regions of its input. The size of the rectangular regions is determined by the poolSizeargument of averagePoolingLayer. For example, if poolSize is [2,3], then the layer returns the average value of regions of height 2 and width 3.

Pooling layers scan through the input horizontally and vertically in step sizes you can specify using the 'Stride' name-value pair argument. If the pool size is smaller than or equal to the stride, then the pooling regions do not overlap.



For non overlapping regions (Pool Size and Stride are equal), if the input to the pooling layer is n-by-n, and the pooling region size is h-by-h, then the pooling layer down-samples the regions by h [[6]](https://ch.mathworks.com/help/deeplearning/ug/layers-of-a-convolutional-neural-network.html#bu741xw). That is, the output of a max or average pooling layer for one channel of a convolutional layer is n/h-by-n/h. For overlapping regions, the output of a pooling layer is (Input Size – Pool Size + 2\*Padding)/Stride + 1.

### Fully Connected Layer

Create a fully connected layer using [fullyConnectedLayer](https://ch.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.fullyconnectedlayer.html).

A fully connected layer multiplies the input by a weight matrix and then adds a bias vector.

The convolutional (and down-sampling) layers are followed by one or more fully connected layers.

As the name suggests, all neurons in a fully connected layer connect to all the neurons in the previous layer. This layer combines all of the features (local information) learned by the previous layers across the image to identify the larger patterns. For classification problems, the last fully connected layer combines the features to classify the images. This is the reason that the output Size argument of the last fully connected layer of the network is equal to the number of classes of the data set. For regression problems, the output size must be equal to the number of response variables.

You can also adjust the learning rate and the regularization parameters for this layer using the related name-value pair arguments when creating the fully connected layer. If you choose not to adjust them, then trainNetwork uses the global training parameters defined by the trainingOptions function. A fully connected layer multiplies the input by a weight matrix W and then adds a bias vector b.

### Output Layers

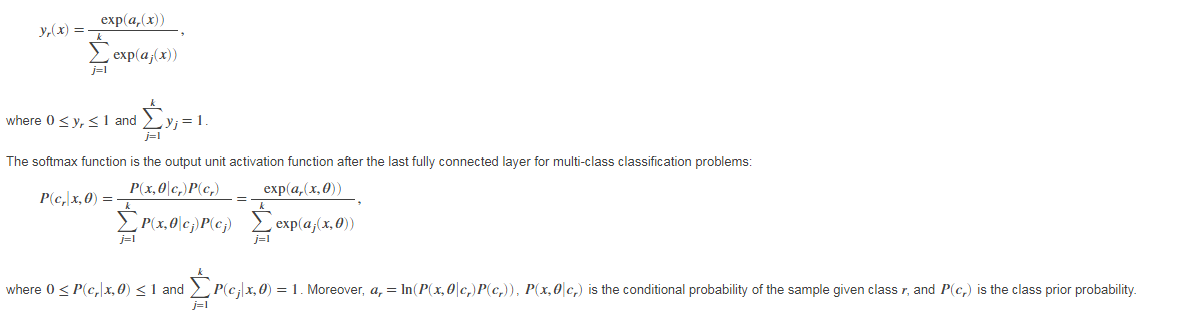
#### Softmax and Classification Layers

A softmax layer applies a softmax function to the input. Create a softmax layer using [softmaxLayer](https://ch.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.softmaxlayer.html).

A classification layer computes the cross entropy loss for multi-class classification problems with mutually exclusive classes. Create a classification layer using [classificationLayer](https://ch.mathworks.com/help/deeplearning/ref/classificationlayer.html).

For classification problems, a softmax layer and then a classification layer must follow the final fully connected layer.

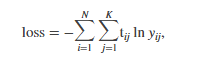
The output unit activation function is the softmax function:



The softmax function is also known as the normalized exponential and can be considered the multi-class generalization of the logistic sigmoid function.

For typical classification networks, the classification layer must follow the softmax layer. In the classification layer, trainNetwork takes the values from the softmax function and assigns each input to one of the K mutually exclusive classes using the cross entropy function for a 1-of-K coding scheme:

,



where N is the number of samples, K is the number of classes, *tij* is the indicator that the ith sample belongs to the jth class, and *yij* is the output for sample i for class j, which in this case, is the value from the softmax function. That is, it is the probability that the network associates the ith input with class j.

**MODULE 2**

TRAINING OPTION

Create a set of options for training a network using stochastic gradient descent with momentum. Reduce the learning rate by a factor of 0.0001 every epoch. Set the maximum number of epochs for training to 20, and use a mini-batch with 64 observations at each iteration.

Solver for training network, specified as one of the following:

* 'sgdm' — Use the stochastic gradient descent with momentum (SGDM) optimizer. You can specify the momentum value using the ['Momentum'](https://ch.mathworks.com/help/deeplearning/ref/trainingoptions.html#bu59f0q-Momentum) name-value pair argument.

### Stochastic Gradient Descent

The standard gradient descent algorithm updates the network parameters (weights and biases) to minimize the loss function by taking small steps at each iteration in the direction of the negative gradient of the loss,

*θ*ℓ+1=*θ*ℓ−*α*∇*E*(*θ*ℓ),

where ℓis the iteration number, *α*>0 is the learning rate, *θ* is the parameter vector, and *E*(*θ*) is the loss function. In the standard gradient descent algorithm, the gradient of the loss function, ∇*E*(*θ*), is evaluated using the entire training set, and the standard gradient descent algorithm uses the entire data set at once.

By contrast, at each iteration the stochastic gradient descent algorithm evaluates the gradient and updates the parameters using a subset of the training data. A different subset, called a mini-batch, is used at each iteration. The full pass of the training algorithm over the entire training set using mini-batches is one epoch. Stochastic gradient descent is stochastic because the parameter updates computed using a mini-batch is a noisy estimate of the parameter update that would result from using the full data set. You can specify the mini-batch size and the maximum number of epochs by using the ['MiniBatchSize'](https://ch.mathworks.com/help/deeplearning/ref/trainingoptions.html#bu59f0q-MiniBatchSize) and ['MaxEpochs'](https://ch.mathworks.com/help/deeplearning/ref/trainingoptions.html#bu59f0q-MaxEpochs) name-value pair arguments, respectively.

#### Stochastic Gradient Descent with Momentum

The stochastic gradient descent algorithm can oscillate along the path of steepest descent towards the optimum. Adding a momentum term to the parameter update is one way to reduce this oscillation [[2]](https://ch.mathworks.com/help/deeplearning/ref/trainingoptions.html#bu812m0). The stochastic gradient descent with momentum (SGDM) update is

*θ*ℓ+1=*θ*ℓ−*α*∇*E*(*θ*ℓ)+*γ*(*θ*ℓ−*θ*ℓ−1),

where *γ* determines the contribution of the previous gradient step to the current iteration. You can specify this value using the ['Momentum'](https://ch.mathworks.com/help/deeplearning/ref/trainingoptions.html#bu59f0q-Momentum) name-value pair argument. To train a neural network using the stochastic gradient descent with momentum algorithm, specify [solverName](https://ch.mathworks.com/help/deeplearning/ref/trainingoptions.html" \l "mw_4f311de8-ad53-4201-ac1a-47f66e0de52d) as 'sgdm'. To specify the initial value of the learning rate α, use the['InitialLearnRate'](https://ch.mathworks.com/help/deeplearning/ref/trainingoptions.html#bu59f0q-InitialLearnRate) name-value pair argument. You can also specify different learning rates for different layers and parameters.

**MODULE 3**

**TRAIN NETWORK**

Use trainNetwork to train a convolutional neural network (ConvNet, CNN), [trainedNet](https://ch.mathworks.com/help/deeplearning/ref/trainnetwork.html?searchHighlight=trainNetwork&s_tid=doc_srchtitle" \l "bu6sn4c-trainedNet) = trainNetwork([imds](https://ch.mathworks.com/help/deeplearning/ref/trainnetwork.html?searchHighlight=trainNetwork&s_tid=doc_srchtitle" \l "bu6sn4c_sep_shared-imds),[layers](https://ch.mathworks.com/help/deeplearning/ref/trainnetwork.html?searchHighlight=trainNetwork&s_tid=doc_srchtitle#bu6sn4c_sep_mw_a4be7938-74e9-410c-8d2d-98ae715a53af),[options](https://ch.mathworks.com/help/deeplearning/ref/trainnetwork.html?searchHighlight=trainNetwork&s_tid=doc_srchtitle#bu6sn4c-options)) trains a network for image classification problems. imds stores the input image data, layers defines the network architecture, and options defines the training options.

**TESTING PHASE**

The query image is processed and features of this image are extracted using the same processes of the training phase. Then a feature vector is created for query image.

**MODULE 4**

**CLASSIFY**

Classify the validation images using the fine-tuned network, and calculate the classification accuracy.

Display four sample validation images with predicted labels and the predicted probabilities of the images having those labels.

**CHAPTER 7**

**IMPLEMENTATION**

The execution stage regarding the task is that the complete aim is essentially changed keen on running code. Intend regarding the stage is towards interpreting the aim keen on a finest likely result within an appropriate programmed language. In this section, it covers up the execution phase concerning the task, providing particulars regarding the programmed language as well as improvement background employed. It as well provides a general idea about the important sections regarding the task by means of its bit by bit course.

The execution phase involves the following tasks:-

* Cautious scheduling.
* Examination regarding structure as well as constraints.
* Aim concerning the techniques towards accomplishing the conversion.
* Assessment concerning the conversion technique.
* Accurate judgment about the choosing of the proposal.
* Suitable choosing regarding the language intended for function growth.

**7.3 INTRODUCTION**

**MATLAB** (**mat**rix **lab**oratory) is a [numerical computing](http://en.wikipedia.org/wiki/Numerical_analysis) environment and [fourth-generation programming language](http://en.wikipedia.org/wiki/Fourth-generation_programming_language). Developed by [Math Works](http://en.wikipedia.org/wiki/MathWorks), MATLAB allows [matrix](http://en.wikipedia.org/wiki/Matrix_(mathematics)) manipulations, plotting of [functions](http://en.wikipedia.org/wiki/Function_(mathematics)) and data, implementation of [algorithms](http://en.wikipedia.org/wiki/Algorithm), creation of [user interfaces](http://en.wikipedia.org/wiki/User_interface), and interfacing with programs written in other languages, including [C](http://en.wikipedia.org/wiki/C_(programming_language)), [C++](http://en.wikipedia.org/wiki/C%2B%2B), [Java](http://en.wikipedia.org/wiki/Java_(programming_language)), and [Fortran](http://en.wikipedia.org/wiki/Fortran).

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the [MuPAD](http://en.wikipedia.org/wiki/MuPAD" \o "MuPAD) [symbolic engine](http://en.wikipedia.org/wiki/Computer_algebra_system), allowing access to [symbolic computing](http://en.wikipedia.org/wiki/Symbolic_computing) capabilities. An additional package, [Simulink](http://en.wikipedia.org/wiki/Simulink), adds graphical multi-domain simulation and [Model-Based Design](http://en.wikipedia.org/wiki/Model_based_design) for [dynamic](http://en.wikipedia.org/wiki/Dynamical_system) and [embedded systems](http://en.wikipedia.org/wiki/Embedded_systems).

In 2004, MATLAB had around one million users across industry and academia. MATLAB users come from various backgrounds of [engineering](http://en.wikipedia.org/wiki/Engineering), [science](http://en.wikipedia.org/wiki/Science), and [economics](http://en.wikipedia.org/wiki/Economics). MATLAB is widely used in academic and research institutions as well as industrial enterprises.

MATLAB was first adopted by researchers and practitioners in [control engineering](http://en.wikipedia.org/wiki/Control_engineering), Little's specialty, but quickly spread to many other domains. It is now also used in education, in particular the teaching of [linear algebra](http://en.wikipedia.org/wiki/Linear_algebra) and [numerical analysis](http://en.wikipedia.org/wiki/Numerical_analysis), and is popular amongst scientists involved in [image processing](http://en.wikipedia.org/wiki/Image_processing). The MATLAB application is built around the MATLAB language. The simplest way to execute MATLAB code is to type it in the Command Window, which is one of the elements of the MATLAB Desktop. When code is entered in the Command Window, MATLAB can be used as an interactive mathematical [shell](http://en.wikipedia.org/wiki/Shell_(computing)). Sequences of commands can be saved in a text file, typically using the MATLAB Editor, as a [script](http://en.wikipedia.org/wiki/Shell_script) or encapsulated into a [function](http://en.wikipedia.org/wiki/Functional_programming), extending the commands available.

MATLAB provides a number of features for documenting and sharing your work. You can integrate your MATLAB code with other languages and applications, and distribute your MATLAB algorithms and applications.

**7.4 FEATURES of matlab**

* High-level language for technical computing.
* Development environment for managing code, files, and data.
* Interactive tools for iterative exploration, design, and problem solving.
* Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, and numerical integration.
* 2-D and 3-D graphics functions for visualizing data.
* Tools for building custom graphical user interfaces.
* Functions for integrating MATLAB based algorithms with external applications and languages, such as C, C++, Fortran, Java™, COM, and Microsoft Excel.

MATLAB is used in vast area, including signal and image processing, communications, control design, [test and measurement](http://www.mathworks.in/applications/t_m), financial modeling and analysis, and computational. Add-on toolboxes (collections of special-purpose MATLAB functions) extend the MATLAB environment to solve particular classes of problems in these application areas.

MATLAB can be used on personal computers and powerful server systems, including the [Cheaha](http://docs.uabgrid.uab.edu/wiki/Cheaha" \o "Cheaha) compute cluster. With the addition of the Parallel Computing Toolbox, the language can be extended with parallel implementations for common computational functions, including for-loop unrolling. Additionally this toolbox supports offloading computationally intensive workloads to [Cheaha](http://docs.uabgrid.uab.edu/wiki/Cheaha" \o "Cheaha) the campus compute cluster.MATLAB is one of a few languages in which each variable is a matrix (broadly construed) and "knows" how big it is. Moreover, the fundamental operators (e.g. addition, multiplication) are programmed to deal with matrices when required. And the MATLAB environment handles much of the bothersome housekeeping that makes all this possible. Since so many of the procedures required for Macro-Investment Analysis involves matrices, MATLAB proves to be an extremely efficient language for both communication and implementation.

**7.4.1 INTERFACING WITH OTHER LANGUAGES**

MATLAB can call functions and subroutines written in the [C programming language](http://en.wikipedia.org/wiki/C_(programming_language)) or [FORTRAN](http://en.wikipedia.org/wiki/Fortran). A wrapper function is created allowing MATLAB data types to be passed and returned. The dynamically loadable object files created by compiling such functions are termed "[MEX-files](http://en.wikipedia.org/wiki/MEX_file)" (for **M**ATLAB **ex**ecutable).

Libraries written in [Java](http://en.wikipedia.org/wiki/Java_(programming_language)), [ActiveX](http://en.wikipedia.org/wiki/ActiveX) or [.NET](http://en.wikipedia.org/wiki/.NET_Framework) can be directly called from MATLAB and many MATLAB libraries (for example [XML](http://en.wikipedia.org/wiki/XML) or [SQL](http://en.wikipedia.org/wiki/SQL) support) are implemented as wrappers around Java or ActiveX libraries. Calling MATLAB from Java is more complicated, but can be done with MATLAB extension, which is sold separately by Math Works, or using an undocumented mechanism called JMI (Java-to-Mat lab Interface), which should not be confused with the unrelated Java that is also called JMI.

As alternatives to the [MuPAD](http://en.wikipedia.org/wiki/MuPAD" \o "MuPAD) based Symbolic Math Toolbox available from Math Works, MATLAB can be connected to [Maple](http://en.wikipedia.org/wiki/Maple_(software)) or [Mathematica](http://en.wikipedia.org/wiki/Mathematica" \o "Mathematica).

Libraries also exist to import and export [MathML](http://en.wikipedia.org/wiki/MathML" \o "MathML).

* **Development Environment**
* Startup Accelerator for faster MATLAB startup on Windows, especially on Windows XP, and for network installations.
* [Spreadsheet Import Tool](http://www.mathworks.in/videos/matlab/new-spreadsheet-import-tool-in-r2011b.html?type=shadow) that provides more options for selecting and loading mixed textual and numeric data.
* Readability and navigation improvements to warning and error messages in the MATLAB command window.
* [Automatic variable and function renaming](http://www.mathworks.in/videos/matlab/new-automatic-variable-and-function-renaming-in-r2011b.html?type=shadow) in the MATLAB Editor.
* **Developing Algorithms and Applications**

MATLAB provides a high-level language and development tools that let you quickly develop and analyze your algorithms and applications.

* **The MATLAB Language**

The MATLAB language supports the vector and matrix operations that are fundamental to engineering and scientific problems. It enables fast development and execution. With the MATLAB language, you can program and develop algorithms faster than with traditional languages because you do not need to perform low-level administrative tasks, such as declaring variables, specifying data types, and allocating memory. In many cases, MATLAB eliminates the need for ‘for’ loops. As a result, one line of MATLAB code can often replace several lines of C or C++ code.

At the same time, MATLAB provides all the features of a traditional programming language, including arithmetic operators, flow control, data structures, data types, [object-oriented programming](http://www.mathworks.in/products/matlab/object_oriented_programming.html) (OOP), and debugging features.

MATLAB lets you execute commands or groups of commands one at a time, without compiling and linking, enabling you to quickly iterate to the optimal solution. For fast execution of heavy matrix and vector computations, MATLAB uses processor-optimized libraries. For general-purpose scalar computations, MATLAB generates machine-code instructions using its JIT (Just-In-Time) compilation technology.

This technology, which is available on most platforms, provides execution speeds that rival those of traditional programming languages.

* **Development Tools**

MATLAB includes development tools that help you implement your algorithm efficiently. These include the following:

**MATLAB Editor**

Provides standard editing and debugging features, such as setting breakpoints and single stepping

**Code Analyzer**

Checks your code for problems and recommends modifications to maximize performance and maintainability

**MATLAB Profiler**

Records the time spent executing each line of code

**Directory Reports**

Scan all the files in a directory and report on code efficiency, file differences, file dependencies, and code coverage

**Designing Graphical User Interfaces**

By using the interactive tool GUIDE (Graphical User Interface Development Environment) to layout, design, and edit user interfaces. GUIDE lets you include list boxes, pull-down menus, push buttons, radio buttons, and sliders, as well as MATLAB plots and Microsoft ActiveX® controls. Alternatively, you can create [GUIs](http://www.mathworks.in/discovery/matlab-gui.html) programmatically using MATLAB functions.

**5.5 The MATLAB System**

The MATLAB system consists of five main parts:

* **Development Environment**.

This is the set of tools and facilities that help you use MATLAB functions and files. Many of these tools are graphical user interfaces. It includes the MATLAB desktop and Command Window, a command history, and browsers for viewing help, the workspace, files, and the search path.

* **The MATLAB Mathematical Function Library**.

This is a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.

* **The MATLAB Language**.

This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create complete large and complex application programs.

* **Handle Graphics**.

This is the MATLAB graphics system. It includes high-level commands for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level commands that allow you to fully customize the appearance of graphics as well as to build complete graphical user interfaces on your MATLAB applications.

* **The MATLAB Application Program Interface (API).**

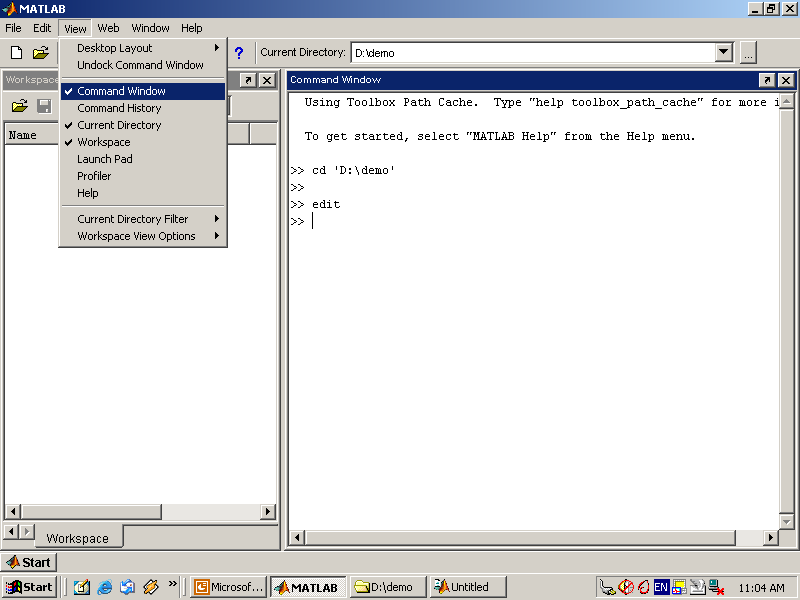
This is a library that allows you to write C and FORTRAN programs that interact with MATLAB. It include facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

**7.5.1 DESKTOP TOOLS**

This section provides an introduction to MATLAB's desktop tools. You can also use MATLAB functions to perform most of the features found in the desktop tools. The tools are:

* Current Directory Browser
* Workspace Browser
* Array Editor
* Editor/Debugger
* Command Window
* Command History
* Launch Pad
* Help Browser

**Command Window**



Use the Command Window to enter variables and run functions and M-files.

* **Command History**

Lines you enter in the Command Window are logged in the Command History window. In the Command History, you can view previously used functions, and copy and execute selected lines. To save the input and output from a MATLAB session to a file, use the diary function.

* **Running External Programs**

You can run external programs from the MATLAB Command Window. The exclamation point character! is a shell escape and indicates that the rest of the input line is a command to the operating system. This is useful for invoking utilities or running other programs without quitting MATLAB. On Linux, for example,!emacs magik.m invokes an editor called emacs for a file named magik.m. When you quit the external program, the operating system returns control to MATLAB.

* **Launch Pad**

MATLAB's Launch Pad provides easy access to tools, demos, and documentation.

* **Help Browser**

Use the Help browser to search and view documentation for all your Math Works products. The Help browser is a Web browser integrated into the MATLAB desktop that displays HTML documents.

To open the Help browser, click the help button in the toolbar, or type helpbrowser in the Command Window. The Help browser consists of two panes, the Help Navigator, which you use to find information, and the display pane, where you view the information.

* **Help Navigator**

Use to Help Navigator to find information. It includes:

* **Product filter**

Set the filter to show documentation only for the products you specify.

* **Contents tab**

View the titles and tables of contents of documentation for your products.

* **Index tab**

Find specific index entries (selected keywords) in the MathWorks documentation for your products.

* **Search tab**

Look for a specific phrase in the documentation. To get help for a specific function, set the Search type to Function Name.

* **Favorites tab**

View a list of documents you previously designated as favorites.

* **Display Pane**

After finding documentation using the Help Navigator, view it in the display pane. While viewing the documentation, you can:

* **Browse to other pages**

Use the arrows at the tops and bottoms of the pages, or use the back and forward buttons in the toolbar.

* **Bookmark pages**

Click the Add to Favorites button in the toolbar.

* **Print pages**

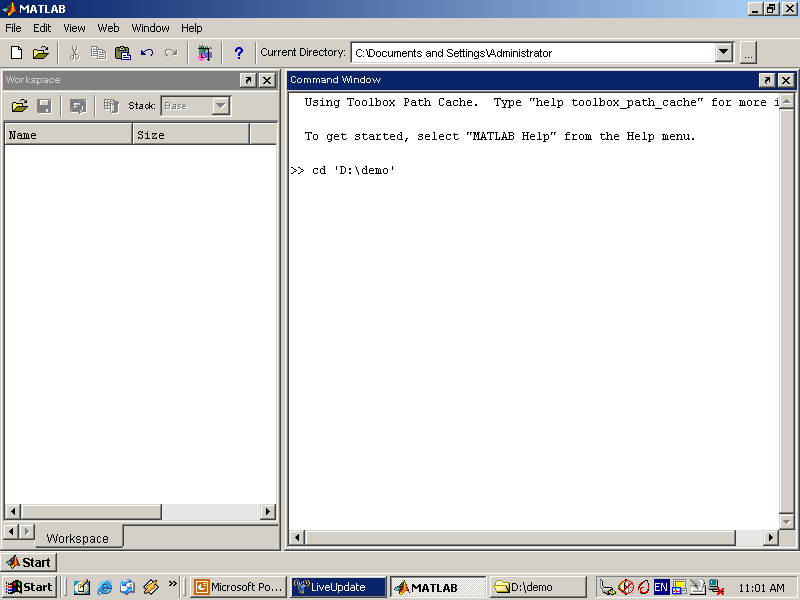
Click the print button in the toolbar.

* **Find a term in the page**

Type a term in the Find in page field in the toolbar and click Go.

Other features available in the display pane are: copying information, evaluating a selection, and viewing Web pages.

**Current Directory Browser**

****

MATLAB file operations use the current directory and the search path as reference points. Any file you want to run must either be in the current directory or on the search path.

**Search Path**

To determine how to execute functions you call, MATLAB uses a search path to find M-files and other MATLAB-related files, which are organized in directories on your file system. Any file you want to run in MATLAB must reside in the current directory or in a directory that is on the search path. By default, the files supplied with MATLAB and MathWorks toolboxes are included in the search path.

* **Workspace Browser**

The MATLAB workspace consists of the set of variables (named arrays) built up during a MATLAB session and stored in memory. You add variables to the workspace by using functions, running M-files, and loading saved workspaces.

To view the workspace and information about each variable, use the Workspace browser, or use the functions who and whos.

To delete variables from the workspace, select the variable and select Delete from the Edit menu. Alternatively, use the clear function.

The workspace is not maintained after you end the MATLAB session. To save the workspace to a file that can be read during a later MATLAB session, select Save Workspace As from the File menu, or use the save function. This saves the workspace to a binary file called a MAT-file, which has a .mat extension. There are options for saving to different formats. To read in a MAT-file, select Import Data from the File menu, or use the load function.

* **Array Editor**

Double-click on a variable in the Workspace browser to see it in the Array Editor. Use the Array Editor to view and edit a visual representation of one- or two-dimensional numeric arrays, strings, and cell arrays of strings that are in the workspace.

* **Editor/Debugger**

Use the Editor/Debugger to create and debug M-files, which are programs you write to run MATLAB functions. The Editor/Debugger provides a graphical user interface for basic textediting, as well as for M-file debugging.

You can use any text editor to create M-files, such as Emacs, and can use preferences (accessible from the desktop File menu) to specify that editor as the default. If you use another editor, you can still use the MATLAB Editor/Debugger for debugging, or you can use debugging functions, such as dbstop, which sets a breakpoint.

If you just need to view the contents of an M-file, you can display it in the Command Window by using the type function.

**7.5.2 ANALYZING AND ACCESSING DATA**

MATLAB supports the entire data analysis process, from acquiring data from external devices and databases, through preprocessing, visualization, and numerical analysis, to producing presentation-quality output.

* **Data Analysis**

MATLAB provides interactive tools and command-line functions for data analysis operations, including:

* Interpolating and decimating
* Extracting sections of data, scaling, and averaging
* Thresholding and smoothing
* Correlation, Fourier analysis, and filtering
* 1-D peak, valley, and zero finding
* Basic statistics and curve fitting
* Matrix analysis

**Data Access**

MATLAB is an efficient platform for accessing data from files, other applications, databases, and external devices. You can read data from popular file formats, such as Microsoft Excel; ASCII text or binary files; image, sound, and video files; and scientific files, such as HDF and HDF5. Low-level binary file I/O functions let you work with data files in any format. Additional functions let you read data from Web pages and XML.

**Visualizing Data**

All the graphics features that are required to visualize engineering and scientific data are available in MATLAB. These include 2-D and 3-D plotting functions, 3-D volume visualization functions, tools for interactively creating plots, and the ability to export results to all popular graphics formats. You can customize plots by adding multiple axes; changing line colors and markers; adding annotation, Latex equations, and legends; and drawing shapes.

**2-D Plotting**

Visualizing vectors of data with 2-D plotting functions that create:

* Line, area, bar, and pie charts.
* Direction and velocity plots.
* Histograms.
* Polygons and surfaces.
* Scatter/bubble plots.
* Animations.

**3-D Plotting and Volume Visualization**

MATLAB provides functions for visualizing 2-D matrices, 3-D scalar, and 3-D vector data. You can use these functions to visualize and understand large, often complex, multidimensional data. Specifying plot characteristics, such as camera viewing angle, perspective, lighting effect, light source locations, and transparency.

3-D plotting functions include:

* Surface, contour, and mesh.
* Image plots.
* Cone, slice, stream, and isosurface.

**7.5.3 PERFORMING NUMERIC COMPUTATION**

MATLAB contains mathematical, statistical, and engineering functions to support all common engineering and science operations. These functions, developed by experts in mathematics, are the foundation of the MATLAB language. The core math functions use the LAPACK and BLAS linear algebra subroutine libraries and the FFTW Discrete Fourier Transform library. Because these processor-dependent libraries are optimized to the different platforms that MATLAB supports, they execute faster than the equivalent C or C++ code.

MATLAB provides the following types of functions for performing mathematical operations and analyzing data:

* Matrix manipulation and linear algebra.
* Polynomials and interpolation.
* Fourier analysis and filtering.
* Data analysis and statistics.
* Optimization and numerical integration.
* Ordinary differential equations (ODEs).
* Partial differential equations (PDEs).
* Sparse matrix operations.

MATLAB can perform arithmetic on a wide range of data types, including doubles, singles, and integers.

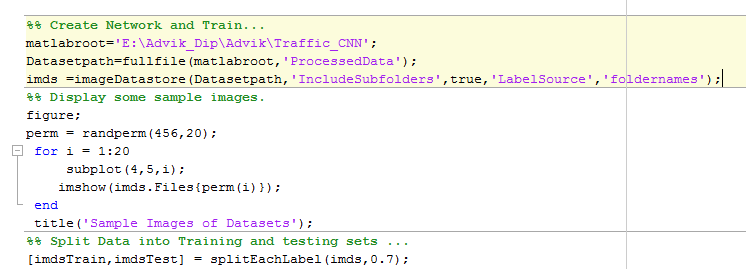
The network takes an image as input, and then outputs a label for the object in the image together with the probabilities for each of the object categories.

* Load and explore image data.
* Define the network architecture.
* Specify training options.
* Train the network.
* Predict the labels of new data and calculate the classification accuracy.

### LoadData

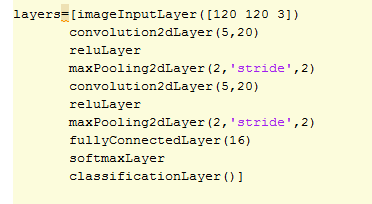
Load the digit sample data as an image datastore. imageDatastore automatically labels the images based on folder names and stores the data as an ImageDatastore object. An image datastore enables you to store large image data, including data that does not fit in memory, and efficiently read batches of images during training of a convolutional neural network.

Load the new images as an image datastore. This very small data set contains only 456 images. Divide the data into training and validation data sets. Use 70% of the images for training and 30% for validation.



### Define Network Architecture

Define the convolutional neural network architecture.



**Image Input Layer** An [imageInputLayer](https://ch.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.imageinputlayer.html) is where you specify the image size, which, in this case, is 28-by-28-by-1. These numbers correspond to the height, width, and the channel size. The digit data consists of grayscale images, so the channel size (color channel) is 1. For a color image, the channel size is 3, corresponding to the RGB values. You do not need to shuffle the data because trainNetwork, by default, shuffles the data at the beginning of training. trainNetwork can also automatically shuffle the data at the beginning of every epoch during training.

**Convolutional Layer** In the convolutional layer, the first argument is filterSize, which is the height and width of the filters the training function uses while scanning along the images. In this example, the number 3 indicates that the filter size is 3-by-3. You can specify different sizes for the height and width of the filter. The second argument is the number of filters, numFilters, which is the number of neurons that connect to the same region of the input. This parameter determines the number of feature maps. Use the 'Padding' name-value pair to add padding to the input feature map. For a convolutional layer with a default stride of 1, 'same' padding ensures that the spatial output size is the same as the input size. You can also define the stride and learning rates for this layer using name-value pair arguments of [convolution2dLayer](https://ch.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.convolution2dlayer.html).

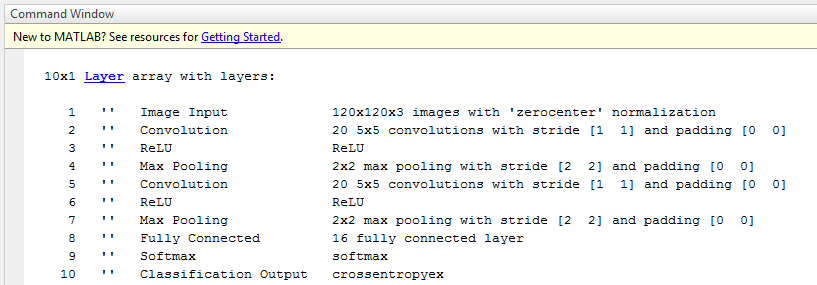
**Batch Normalization Layer** Batch normalization layers normalize the activations and gradients propagating through a network, making network training an easier optimization problem. Use batch normalization layers between convolutional layers and nonlinearities, such as ReLU layers, to speed up network training and reduce the sensitivity to network initialization. Use [batchNormalizationLayer](https://ch.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.batchnormalizationlayer.html) to create a batch normalization layer.

**ReLU Layer** The batch normalization layer is followed by a nonlinear activation function. The most common activation function is the rectified linear unit (ReLU). Use [reluLayer](https://ch.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.relulayer.html) to create a ReLU layer.

**Max Pooling Layer** Convolutional layers (with activation functions) are sometimes followed by a down-sampling operation that reduces the spatial size of the feature map and removes redundant spatial information. Down-sampling makes it possible to increase the number of filters in deeper convolutional layers without increasing the required amount of computation per layer. One way of down-sampling is using a max pooling, which you create using [maxPooling2dLayer](https://ch.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.maxpooling2dlayer.html). The max pooling layer returns the maximum values of rectangular regions of inputs, specified by the first argument, poolSize. In this example, the size of the rectangular region is [2,2]. The 'Stride' name-value pair argument specifies the step size that the training function takes as it scans along the input.

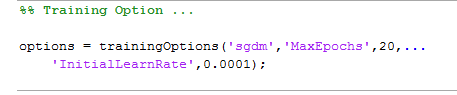
**Fully Connected Layer** The convolutional and down-sampling layers are followed by one or more fully connected layers. As its name suggests, a fully connected layer is a layer in which the neurons connect to all the neurons in the preceding layer. This layer combines all the features learned by the previous layers across the image to identify the larger patterns. The last fully connected layer combines the features to classify the images. Therefore, the OutputSize parameter in the last fully connected layer is equal to the number of classes in the target data. In this example, the output size is 10, corresponding to the 10 classes. Use [fullyConnectedLayer](https://ch.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.fullyconnectedlayer.html) to create a fully connected layer.

**Softmax Layer** The softmax activation function normalizes the output of the fully connected layer. The output of the softmax layer consists of positive numbers that sum to one, which can then be used as classification probabilities by the classification layer. Create a softmax layer using the [softmaxLayer](https://ch.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.softmaxlayer.html) function after the last fully connected layer.

**Classification Layer** The final layer is the classification layer. This layer uses the probabilities returned by the softmax activation function for each input to assign the input to one of the mutually exclusive classes and compute the loss. To create a classification layer, use [classificationLayer](https://ch.mathworks.com/help/deeplearning/ref/classificationlayer.html). 

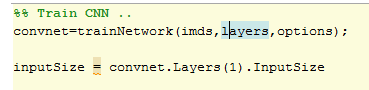
### Specify Training Options

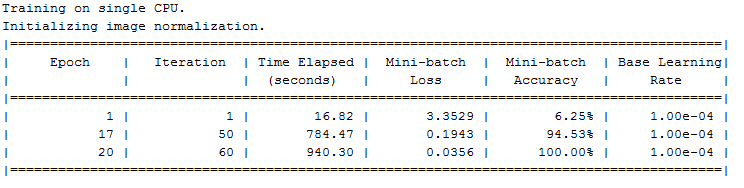
After defining the network structure, specify the training options. Train the network using stochastic gradient descent with momentum (SGDM) with an initial learning rate of 0.0001. Set the maximum number of epochs to 20. An epoch is a full training cycle on the entire training data set. Monitor the network accuracy during training by specifying validation data and validation frequency. Shuffle the data every epoch. The software trains the network on the training data and calculates the accuracy on the validation data at regular intervals during training. The validation data is not used to update the network weights. Turn on the training progress plot, and turn off the command window output.

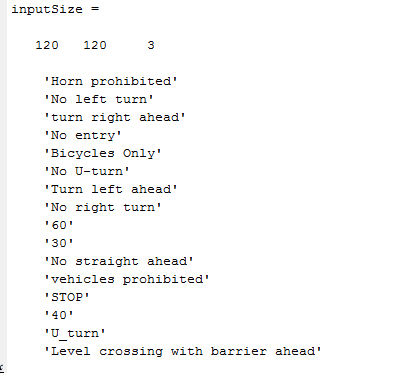


### Train Network Using Training Data

Train the network using the architecture defined by layers, the training data, and the training options.





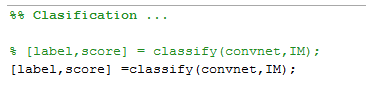


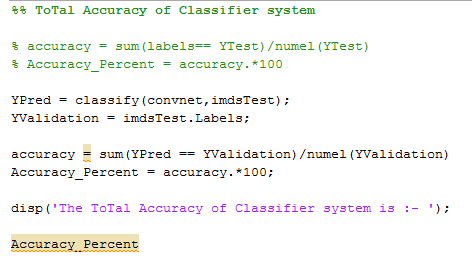
### Testing

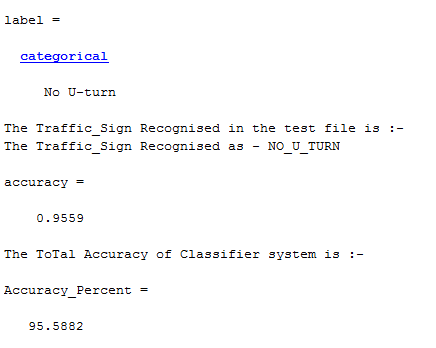
### 

### Classify Validation Images and Compute Accuracy

Predict the labels of the validation data using the trained network, and calculate the final validation accuracy. Accuracy is the fraction of labels that the network predicts correctly. In this case, more than 99% of the predicted labels match the true labels of the validation set.

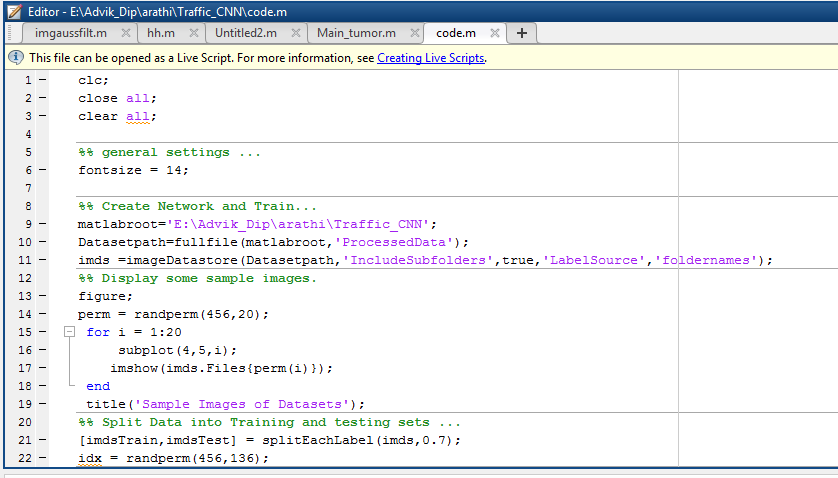




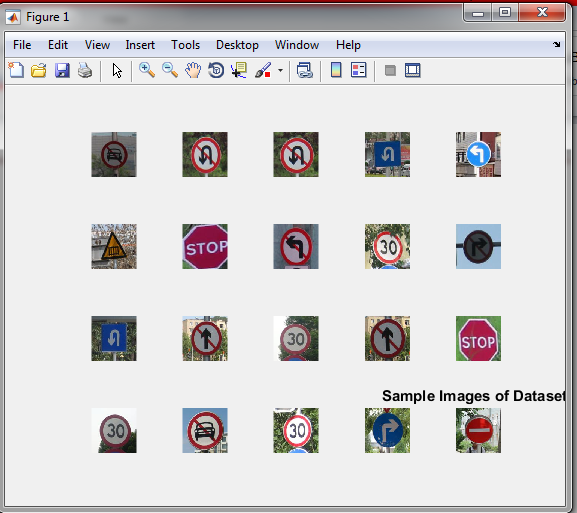


# Chapter 8

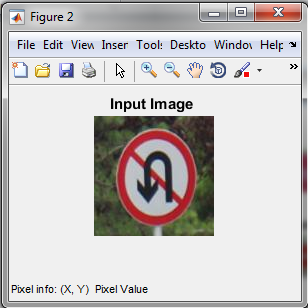
## RESULTS



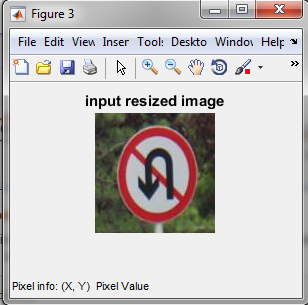
**FIG: MATLAB CODE**



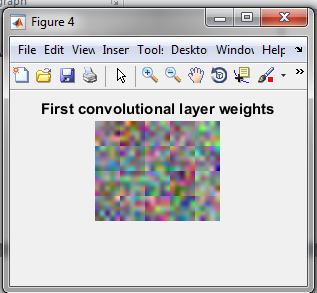
**FIG: SAMPLE IMAGES OF DATASETS.**



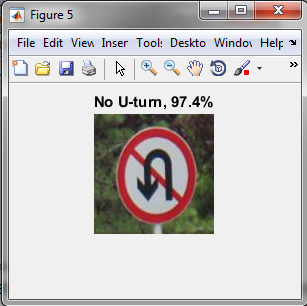
**FIG: INPUT COLOUR IMAGE**



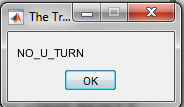
**FIG: I** **NPUT RESIZED IMAGE .**



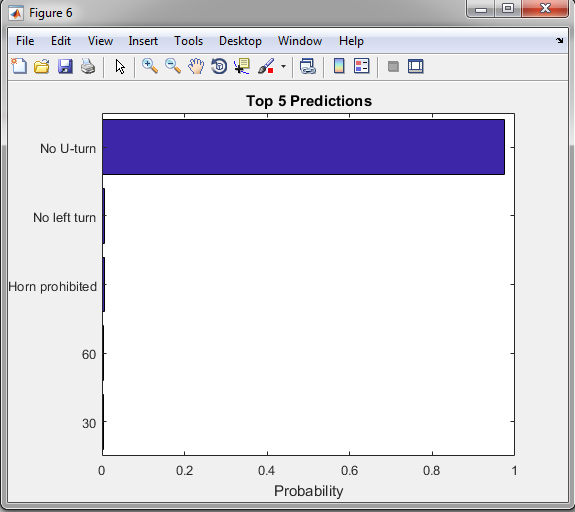
**FIG: FIRST CONVOLUTIONAL LAYER WEIGHTS IMAGE .**



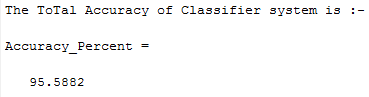
**FIG: ClASSIFICATION RESULT**



**FIG: CLASSIFICATION LABEL**



**FIG: TOP 5 PREDICTIONS OF PROBABLITIES**



**FIG: ACCURACY**

# Chapter 9

**CONCLUSION AND FUTURE SCOPE**

This paper considers an implementation of the classification algorithm for the traffic signs recognition task. Combined with preprocessing and localization steps from previous works, the proposed method for traffic signs classification shows very good results: 9.94 % of correctly classified images. The proposed classification solution is implemented using the MATLAB framework. The use of our TSR algorithms allows processing of video streams in real-time with high resolution, and therefore at greater distances and with better quality than similar TSR systems have. Full HD resolution makes it possible to detect and recognize a traffic sign at a distance up to 50 m. In future research, we plan to train the CNN to consider more traffic sign classes and possible bad weather conditions. Also, we plan to use a CNN not only for classification but for object detection too.

REFERENCES

[1] A. Nikonorov, P. Yakimov, M. Petrov, Traffic sign detection on GPU using color shape regular expressions, VISIGRAPP IMTA-4, Paper 8 (2013).

[2] R. Belaroussi, P. Foucher, J.P. Tarel, B. Soheilian, P. Charbonnier, N. Paparoditis, Road Sign Detection in Images, A Case Study, 20th International Conference on Pattern Recognition (ICPR), 2010, pp. 484-488.

[3] A. Ruta, F. Porikli, Y. Li, S. Watanabe, H. Kage, K. Sumi, A New Approach for In-Vehicle Camea Traffic Sign Detection and Recognition, IAPR Conference on Machine Vision Applications (MVA), Session 15: Machine Vision for Transportation, 2009.

[4] V. Fursov, S. Bibkov, P. Yakimov, Localization of objects contours with different scales in images using Hough transform [in Russian], Computer Optics. 37, 4 (2013) 502-508.

[5] P. Yakimov, Tracking traffic signs in video sequences based on a vehicle velocity [in Russian], Computer Optics. 39, 5 (2015) 795-800.

[6] J. Stallkamp, M. Schlipsing, J. Salmen, C. Igel, Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition, Neural networks. 32 (2012) 323-332.

[7] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, C. Igel, Detection of Traffic Signs in Real-World Images: The {G}erman {T}raffic {S}ign {D}etection {B}enchmark, in: Proc. International Joint Conference on Neural Networks, 2013.

[8] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, S. Hu, Traffic-Sign Detection and Classification in the Wild. Proceedings of CVPR, 2016, рp. 2110-2118.

[9] Y. LeCun, P. Sermanet, Traffic Sign Recognition with Multi-Scale Convolutional Networks, Proceedings of International Joint Conference on Neural Networks (IJCNN'11), 2011.

[10] P. Yakimov, Preprocessing of digital images in systems of location and recognition of road signs [in Russian], Computer Optics. 37, 3 (2013) 401-405.

[11] V. Terehov, D. Efimov, I. Tiukin, Neural network control system [in Russian], Textbook for high schools, High school, 2002.

[12] Djork-Arné Clevert, Thomas Unterthiner, Sepp Hochreiter, Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs), 2015.

[13] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, ´ M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous distributed systems, arXiv preprint, 1603.04467, 2016. arxiv.org/abs/1603.04467. Software available from tensorflow.org.

[14] M. Mathias, R. Timofte, R. Benenson, L. Gool, Traffic sign recognition - how far are we from the solution? Proceedings of IEEE International Joint Conference on Neural Networks, 2013, pp. 1-8.

[15] H. Aghdam, E. Heravi, D. Puig, A practical approach for detection and classification of traffic signs using Convolutional Neural Networks, Robotics and Autonomous Systems. 84 (2016) 97-112.