VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY

UNIVERSITY OF INFORMATION TECHNOLOGY

ADVANCED PROGRAM IN INFORMATION SYSTEMS

NGUYEN TRAN TRONG NHAN

THESIS

DEEP LEARNING FOR TRAFFIC SIGN RECOGNITION

BECHELOR OF ENGINEERING IN INFORMATION SYSTEMS

HO CHI MINH CITY, 2019

VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY

UNIVERSITY OF INFORMATION TECHNOLOGY

ADVANCED PROGRAM IN INFORMATION SYSTEMS

NGUYEN TRAN TRONG NHAN - 15520569

THESIS

DEEP LEARNING FOR TRAFFIC SIGN RECOGNITION

BECHELOR OF ENGINEERING IN INFORMATION SYSTEMS

THESIS ADVISOR

Dr. Cao Thi Nhan

HO CHI MINH CITY, 2019

Ho Chi Minh City, December 30th, 2019

Thesis Approval **Signature of Thesis Reviewer**	**Signature of Student**

ASSESSMENT COMMITTEE

The Assessment Committee is established under the Decision of University of Information Technology on the date of March 04th, 2020 by Rector of the University of Information Technology.

1. Assoc. Prof. Nguyen Dinh Thuan	– Chairman.
2. Dr. Duong Minh Duc	– Secretary.
3. Dr. Nguyen Thanh Binh	– Member.

ACKNOWLEDGEMENT

First, I would like to express my sincere gratitude to my advisor, Dr. Cao Thi Nhan for the support me in my study. Her patience, encouragement and insights to details helped me a lot to produce this work.

Beside my advisor, I would like to send my thank to Dr. Ngo Duc Thanh for reviewing my report carefully and giving me the advices. His courses of Computer Science 1, Computer Science 2 and Artificial Intelligence have immensely inspired me to work. Also, his and Dr. Le Dinh Duy's free courses on Google Classroom have been bringing me a lot of helpful knowledge.

I would also like to express my thanks to all of lecturers. They have conveyed valuable specialized knowledge, as well as the advice, dedicated help to me during the time I study at the university.

I would like to thank all of my friends who have sent me a lot of motivations and valuable memories.

Last but not least, I would like to send my grateful thanks to my parents. They have always believed me, been my side and encouraged me a lot.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

LIST OF FORMULAS

ABBREVIATIONS

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
CNN	Convolutional Neural Network
GUI	Graphical User Interface
CLI	Command Line Interface
GTSRB	German Traffic Sign Recognition Benchmark
ROI	Region Of Interest

ABSTRACT

In recent years, with the strong development of Artificial Intelligence (AI) and automotive engineering technology, traffic sign recognition has played a very important role in autonomous driving systems or driver assistance systems. This helps minimize problems in traffic, especially traffic accidents. In this thesis, we solve the problem by developing a tool applied Convolutional Neural Network model (CNN) to classify the traffic signs. After training the model, we input a new image. The tool will analyze and recognize the traffic sign in the image and show its name. Experimental results of the tool are conducted and it is now possible to recognize traffic signs through the input image.

Chapter 1. INTRODUCTION

1.1. Overview

At present, human life have been improving rapidly, resulting in many improved needs, particularly travel needs. Transportation is a problem in many countries around the world. This is a real-world problem of people when more and more vehicles join the traffic. This can easily lead to many problems, especially traffic accidents, partly due to driver not being able to see traffic signs because of weather conditions, eyes problems or blind spot. Another reason is that the driver is a foreign tourist, so they do not clearly understand the traffic signs.

Since then, we have developed a tool that can help driver recognize the traffic signs in front of their vehicles. This tool works with three main processes: (1) Train the traffic signs dataset by CNN, (2) Receive and process the input image, and (3) Use the trained model to recognize the traffic sign.

1.2. Problem Statement

Traffic accidents are serious problems in practice. One of the ways to help reduce the number of accidents is to assist driver to recognize traffic signs in front of their vehicles. This helps them to be better prepared when participating in traffic. This is also a motivation to help us develop the tool for traffic sign recognition. This tool will analyze and classify the traffic sign in the input image applying CNN model and image processing technique then display the traffic sign name through the GUI. Specifically, we use the LeNet-5 model to train data. After receiving the input image, we convert the color space from RGB to HSV and use the OpenCV library to extract image features to find the location of the traffic signs. We then take that region of interest (ROI) into the model we have trained to recognize the traffic sign.

1.3. Related Works

For the need of development and the rise of science and technology, people begin to focus on finding algorithms as well as solutions to a problem in order to find the best solution. Artificial intelligence has also been a hot topic in recent years. There are many network architecture in AI such as Single Layer Perceptrons, Multi Layer Perceptron, Recurrent Neural Network, Hopfield Network. Each architecture has a different approach and depend on the problem, the researchers will choose their approach and algorithm.

With the problem of recognizing the traffic signs, researchers have taken a number of approaches to find out the better way for next steps. Classification of traffic signs may be not a simple task due to many reasons. The researchers should care about how good their object detection algorithm is, how the quality of the input images is and which architecture will be chosen.

1.3.1. Real-time Traffic Sign Recognition System

The conference paper called Real-time Traffic Sign Recognition System [9] is published by Zsolf T. Kardkovacs, Adam Siegler, et al in August, 2011. They solved the problem by segmenting the color using CEICAM color space to improve the accuracy of image segmentation when they are glare or in unfavorable weather. They used the AdaBoost algorithm and the Viola-Jones framework to select the ROI. They then applied the Scale-invariant feature transform (SIFT) and Speeded up robust features (SURF) algorithms combined with the classifier such as Multi-layer Perceptrons (MLP) to identify the traffic sign in the image.

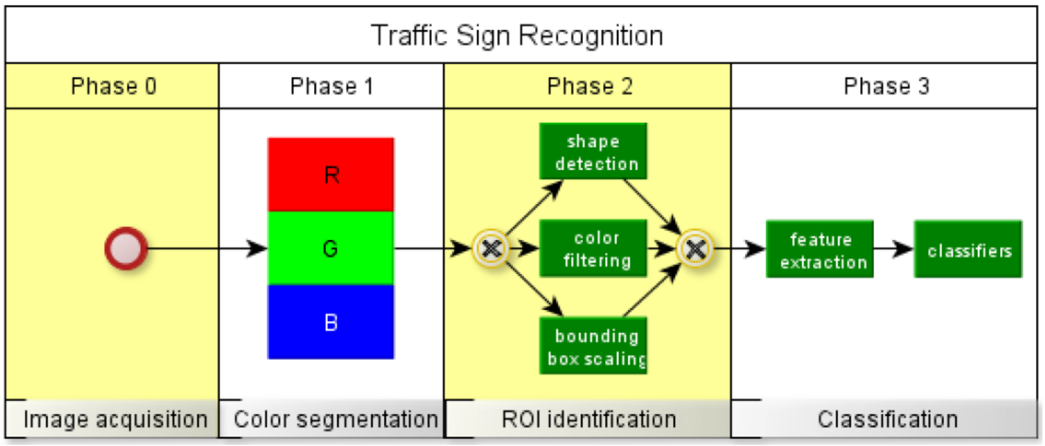


Figure 1.1 Real-time Traffic Sign Recognition System general architecture [9]

1.3.2. Traffic Sign Classification Using Deep Inception Based Convolutional Networks

Mrinal Haloi, the author of the paper called Traffic Sign Classification Using Deep Inception Based Convolutional Networks [10], has used the spatial transformer layers and the modified version of inception module to classify the traffic signs in the image. In the paper, the author has mentioned about the issue that the image undergoes deformation such as blurring, rotational, skew due to the moving camera. The spatial transformer network can generate automatic transformation of input image. This may make the classification more robust and accurate along with the modified version of GoogLeNet.

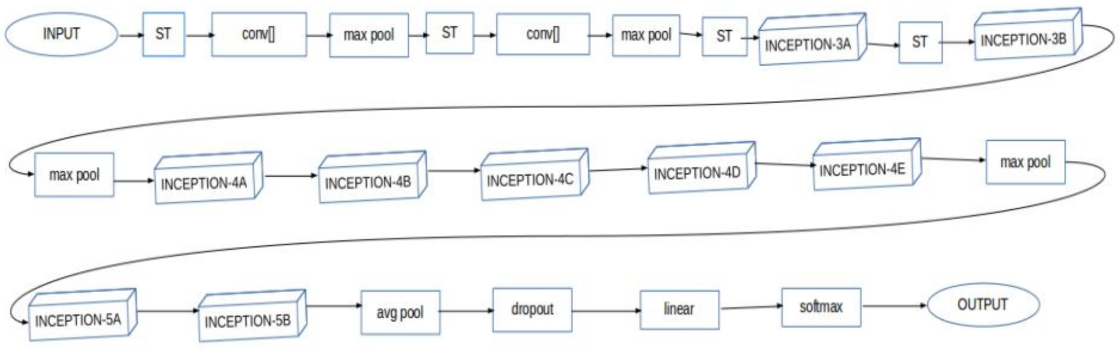


Figure 1.2 The modified inception network - GoogLeNet [10]

1.4. Objectives and Scope

Objectives: The general goal of the project is to build a tool to recognize traffic sign in the input image with actual landscape and traffic sign.

Scope: Apply Python programming language, image processing module (OpenCV) and Deep Learning (Convolutional Neural Network model – CNN model) to build the traffic sign recognition tool.

1.5. Thesis Structure

The thesis is organized as following:

Chapter 1: Introduction

Chapter 2: Fundamental theories and technology used

Chapter 3: System analysis and design

Chapter 4: Experiment and evaluation

Chapter 5: Conclusion and future works

Chapter 2. FUNDAMENTAL THEORIES AND TECHNOLOGIES USED

2.1. Fundamental Theories

2.1.1. Image Processing

An image may be defined as a two-dimensional function, f(x, y), where x and y are spatial (place) coordinates, and the amplitude of f at any pair of coordinates (x, y) is called intensity or gray level of the image at that point [1]. A advanced image is composed of a limited number of elements, each of which incorporates a specific area and value.

Image processing encompasses processes whose inputs and outputs are images and, in expansion, encompasses processes that extract properties from images, up to include counting the recognition of individual objects. As an outline to clarify these concepts, consider the zone of computerized investigation of content. The processes of procuring an image of the are containing the text, preprocessing that image, extracting (segmenting) the individual characters, depicting the characters in a shape reasonable for computer processing, and recognizing those individual characters are within the scope of what we call computerized image processing is.

Image processing may be a sub-sector of digital signal processing with image processing signals. Typically a new logical sub-branch created in later a long time. It is the method of changing over from an original image to a new image with the features and inclinations of the user.

Purpose of image processing is to transform images to increase quality and automatically identify images, recognize and evaluate the content of them.

Identifying and evaluating the contents of an image is the analysis of an image into meaningful parts to distinguish one object from another, based on which we can describe the structure of the original image.

Image processing has some major applications such as Photoshop, space image processing, cancer cells image processing, automatic character recognition, finger print, face detection and object detection.

2.1.2. Pixel

Pixel is the smallest element of an image. Each pixel corresponds to a value. In an 8-bit gray image, the value of the pixel is from 0 to 255. The value of the pixel at any point corresponds to the intensity of the photon at that point [1]. Each pixel value is directly proportional to the light intensity. Each pixel has a unique color, designated as a certain ratio of the three primary colors Red, Green and Blue.

2.1.3. RGB Color System

RGB stands for red, green, and blue. They are three main colors of light when separated from the prism. Mixing these three colors in a certain proportion can create different colors. Figure 2.1 shows the RGB color system.

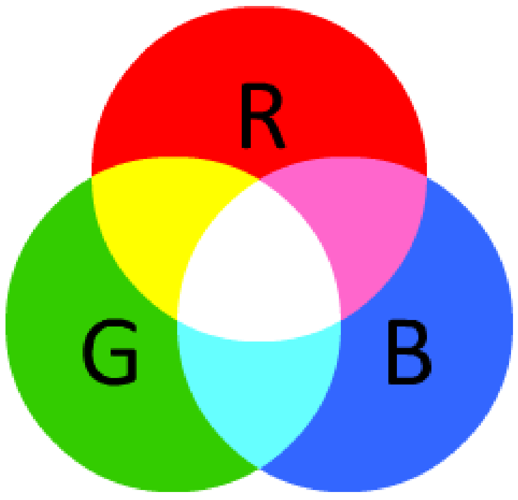


Figure 2.1 RGB color system

[1] https://www.tutorialspoint.com/dip/ (read on December 20th, 2019)

2.1.4. Gray Level

Gray level is the result of the conversion corresponding to a brightness value of one pixel with a positive integer value. It is defined from 0 to 255 depending on the value that each pixel is represented [2]. The normal gray scale values: 16, 32, 64, 128, 256. Figure 2.2 shows the gray level.

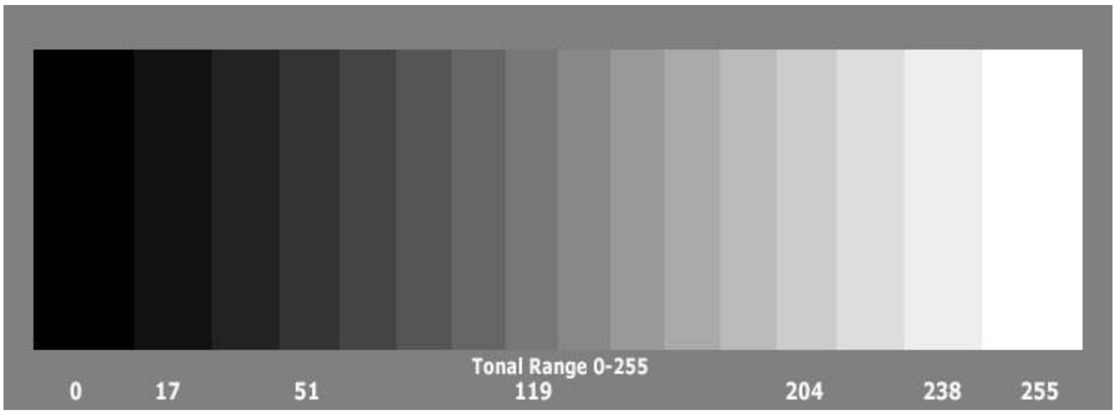


Figure 2.2 Gray level

2.1.5. Image Classification

Depending on the value used to represent the pixel, there are three main types of image:

- Binary image: A binary image is one that comprises of pixels that can have one of precisely two colors, ordinarily black and white. Binary images are also called bi-level or two-level. The binary images contain two pixel values 0 and 1 (0 refers to black and 1 refers to white). It is also known as Monochrome [3]. The names black-and-white, B&W, monochrome or monochromatic are frequently utilized for this concept, but may also assign any images that have as it were one test per pixel, such as grayscale images. Figure 2.3 shows the binary image.

[2, 3] https://www.tutorialspoint.com/dip/ (read on December 20th, 2019)

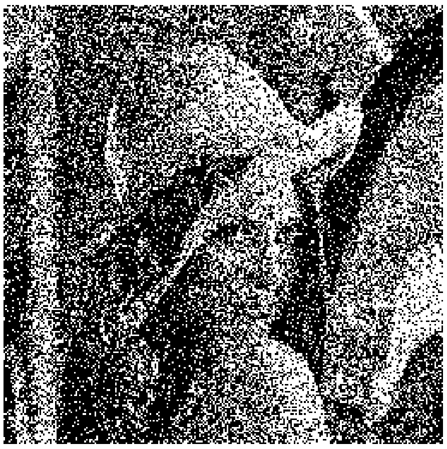


Figure 2.3 Binary image

- Grayscale image: Grayscale images comprises of only gray tones of colors (256 gray colors). The main characteristic of grayscale images is the equality of the red, green, and blue color levels. The color code will be like RGB(R,R,R), RGB(G,G,G), or RGB(B,B,B) where 'R,G,B' is a number between 0 and 255 individually [4]. Figure 2.4 shows the grayscale image.

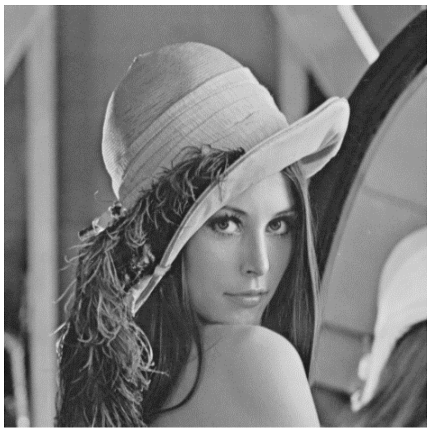


Figure 2.4 Grayscale image

- Color image: Each pixel has a value of 3 colors: red, green and blue. Each color has a value from 0 to 255, meaning that each pixel needs 24 bits or 3 bytes to represent. Figure 2.5 shows the color image.

[4] https://www.sciencedirect.com/topics/engineering/grayscale-image (read on December 20th, 2019)

Figure 2.5 Color image

2.1.6. Image Format

There are three main types of image formats:

- JPG or JPEG (Joint Photographic Experts Group): JPEG could be a 16-bit image format that can combine red, blue and green light to show millions of colors. This makes JPEG "friendly" and is the standard organize for most digital cameras nowadays. JPEG is utilized for static image, photography and image with complex colors.
- GIF (Graphic Interchange Format): GIF is an 8-bit color file. It is limited to a palette of up to 256 colors. Every GIF is pre-selected and there is no way to combine those colors into new colors. GIF also supports animation, limiting every frame to 256 pre-selected colors. GIF is used for simple animations, small icons and graphics with low pixel-to-pixel variations.
- PNG (Portable Network Graphics): PNG is a newer image format than GIF and JPEG. It not only supports 8-bit color like GIF, but also supports 24-bit RGB color like JPEG. PNG belongs to lossless compression, which does not degrade the image. PNG file size is usually larger than JPEG, GIF and not supported by some browsers. PNG is used for web graphics requiring transparency, colorful pictures or graphics with high complexity and images need to be edited and exported many times.

2.1.7. Artificial Intelligence - Machine Learning and Deep Learning

There are some differences as well as relationships between Artificial Intelligence (AI), Machine Learning (ML) and Deep Learning (DL):

- AI is the ability of a computer program or a machine to think and learn. It is also a field of study, which tries to make computers "smart". They work on their own without being encoded with commands. In general, the term "artificial intelligence" means a program which mimics human cognition. At least, some of the things we associate with other minds such as learning and problem solving can be done by computers, though not in the same way as we do [2].
- ML is a subfield of AI that involves the research and development of techniques that enable automated systems to "learn" from data to solve specific problems. Special applications of ML include data tracing, medical diagnostics, fake credit card detection, stock market analysis, DNA sequencing, speech and writing recognition, automatic translation, game play and movements of locomotion robots.
- DL is subset of machine learning methods based on Artificial Neural Networks (ANN). It is a field based on learning and improving on its own by examining computer algorithms. While machine learning uses simpler concepts, DL works with ANN designed to imitate how humans think and learn. Until recently, neural networks were limited by computing power and thus were limited in complexity. However, advancements in Big Data analytics have permitted larger, sophisticated neural networks, allowing computers to observe, learn, and react to complex situations faster than humans. DL has aided image classification, language translation, speech recognition. It can be used to solve any pattern recognition problem and without human intervention.

We can briefly understand these three things as follows:

- Artificial Intelligence: Human Intelligence Exhibited by Machines.
- Machine Learning: An Approach to Achieve Artificial Intelligence.
- Deep Learning: A Technique for Implementing Machine Learning.

Figure 2.6 shows the AI - ML - DL - NN illustration.

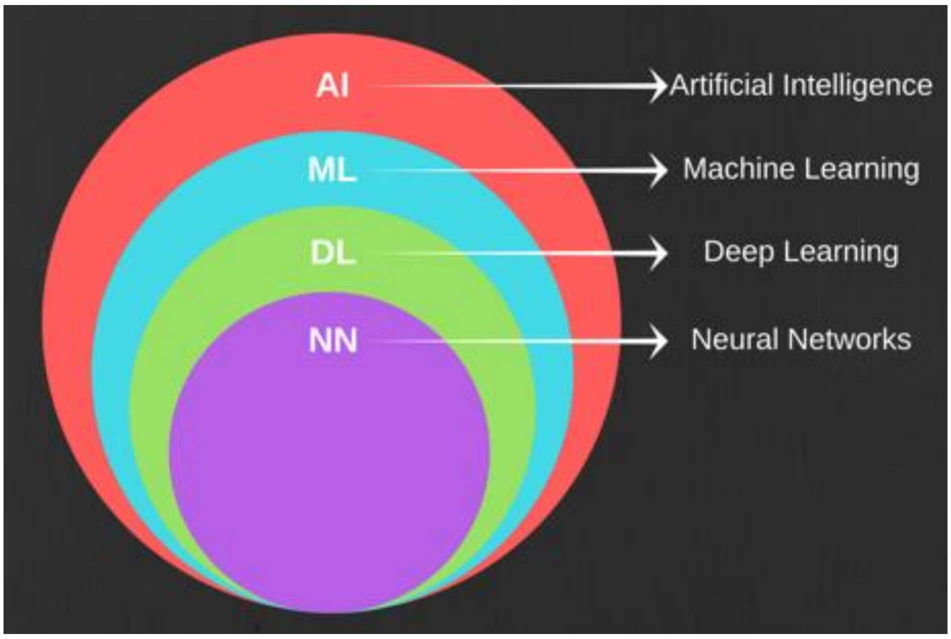


Figure 2.6 AL- ML - DL - NN illustration

2.1.8. Artificial Neural Network

ANN is an information processing system which contains a huge number of exceedingly interconnected processing neurons. These neurons work together in a distributed way to memorize from the input information, to facilitate inside processing, and to optimize its last output. As various algorithms have been reported within the literature applying neural systems to medical image analysis, we offer a centered overview on computational intelligence with neural systems covering medical image registration, segmentation and edge detection for medical image substance analysis, computer-aided discovery and conclusion with particular scope on mammogram analysis towards breast cancer screening, and other applications providing a worldwide view on the variety of neural network applications and their potential for advance research and developments [3].

2.1.9. Deep Neural Network

Deep Neural Network (DNN) is a complex neural network system consisting of many neural network units in which, in addition to the input layer, the output has more than one hidden layer. Each of these classes will implement its own sorting and sorting process in a process called "feature hierarchy," and each class has its own responsibility, the output of this class will be the input of the next class.

DNN built with the purpose of simulating complex human brain activity and applied in many different fields, bringing success and incredible effects to people.

2.1.10.Convolutional Neural Network

Convolutional Neural Organize (CNN) may be a extraordinary kind of multi-layer neural networks. Like nearly each other neural networks they are trained with a adaptation of the back-propagation algorithm [5]. CNNs are designed to recognize visual patterns specifically from pixel images with minimal preprocessing. They can recognize patterns with extraordinary inconstancy, and with vigor to twists and basic geometric changes.

CNN is one of the foremost well known neural network models nowadays. It has the capacity to distinguish and classify images with exceptionally high accuracy, even better than people in numerous cases. This model has been created, applied to huge image processing systems of Facebook, Google or Amazon ... for diverse purposes such as automatic tagging algorithms, image searching or product recommendations for customers.

[5] http://yann.lecun.com/exdb/lenet/ (read on December 20th, 2019)

CNN network based on the idea of improving the way ANN traditionally learn information in images. Due to the use of fully connections between the pixels and the node, the feedforward neural network is limited by the size of the image, the larger the image, the more the number of fully connections increasing rapidly and resulting in an explosion of computational volume. In addition, this fully connection is also redundant when with each photo, the information mainly expressed through the dependence between the pixel and the points around it without much attention to the ones far from them in the image. CNN network was born with a changed architecture, capable of building fully connections using only a local part of the image connecting to the node in the next layer, instead of the entire image as in a feedforward neural network. Figure 2.7 shows the CNN example.

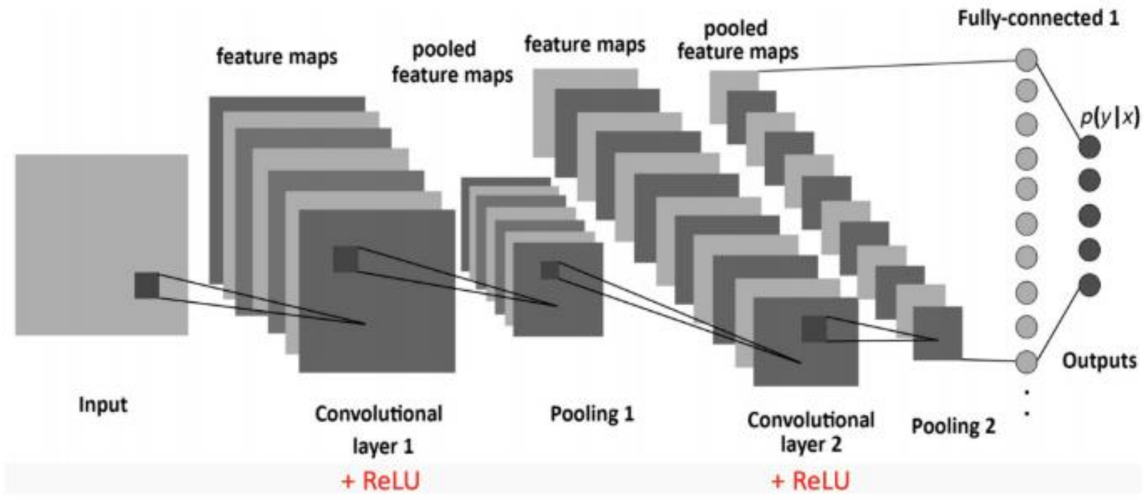


Figure 2.7 Convolutional neural network example [6]

2.1.11.Convolutional Neural Network Structure

CNN is a collection of overlapping convolutional layers and uses nonlinear activation functions like ReLU and tanh to activate weights in nodes. After activating, each layer will generate abstract information for the next class.

[6] https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac

Regular CNN are organized in strict feed-forward layered structures in which the output of one layer is fed as it were to the layer over. Instead, the output of the first stage is branched out and fed to the classifier, in addition to the output of the second stage. We use the output of the first stage after pooling/subsampling instead of before. Moreover, applying a moment subsampling stage on the branched output yielded higher accuracies than with fair one. Subsequently the branched 1st-stage outputs are more subsampled than in traditional CNN but generally undergoes the same sum of subsampling than the 2nd-stage outputs. The motivation for combining representation from numerous stages within the classifier is to supply distinctive scales of responsive fields to the classifier. Within the case of 2 stages of features, the second stage extracts "global" and invariant shapes and structures, whereas the first stage extracts "local" themes with more exact points of interest [4].

The convolutional layer is a collection of feature maps and each of these feature maps is a scan of the original input, but extracted from the features. This is a matrix which scans the input data matrix, from left to right, top to bottom, and multiply corresponding values of the input matrix that the kernel matrix then sums up, and passes through the activation function (sigmoid, relu, elu ...), the result will be a specific number, the set of numbers is another matrix, which is the feature map. Figure 2.8 shows the convolutional layer.

- An image matrix (volume) of dimension **($h \times w \times d$)**
- A filter **($f_h \times f_w \times d$)**
- Outputs a volume dimension **($(h - f_h + 1) \times (w - f_w + 1) \times 1$)**

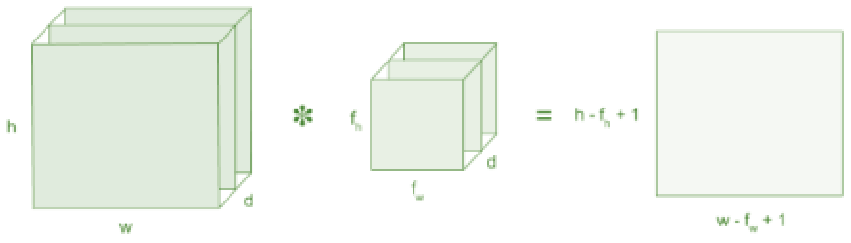


Figure 2.8 Convolutional layer

Figure 2.9 shows the convolutional layer and filter example.

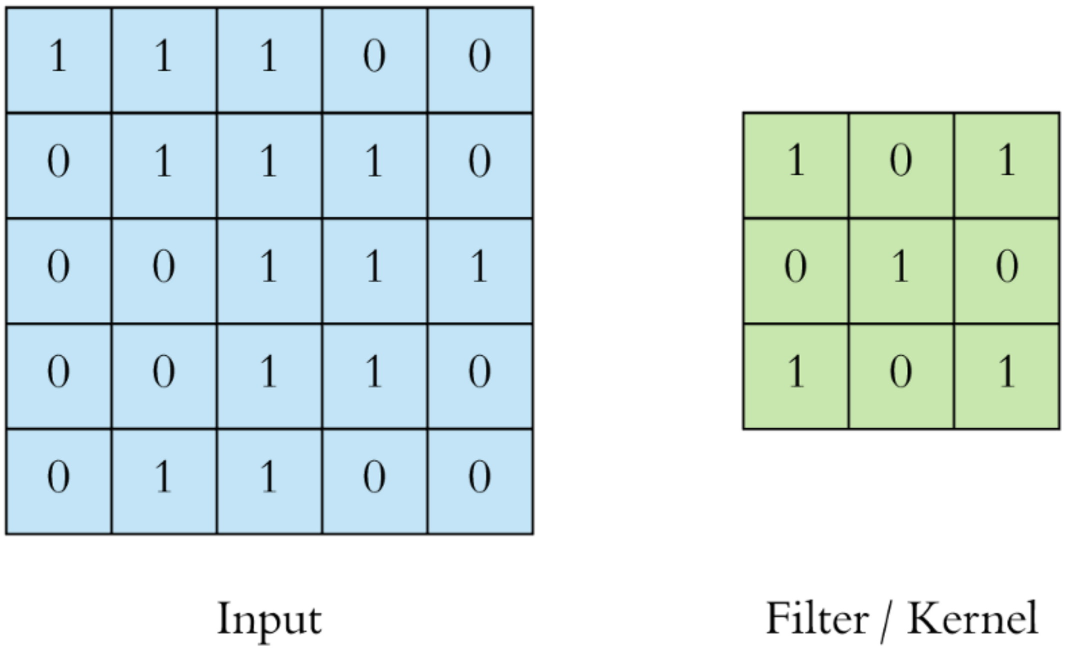


Figure 2.9 Convolutional layer and filter example

Scan the kernel through each element of the input and calculate as above, we get a value at the feature map. Figure 2.10 shows the feature map after the first scan.

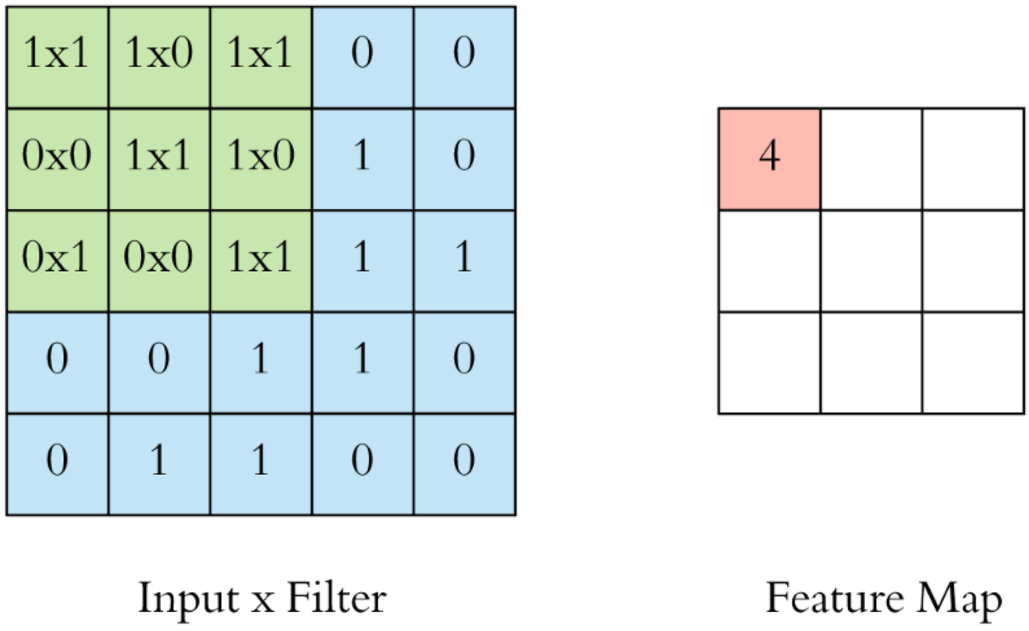


Figure 2.10 Feature map after the first scan

Similarly, we have the following result:

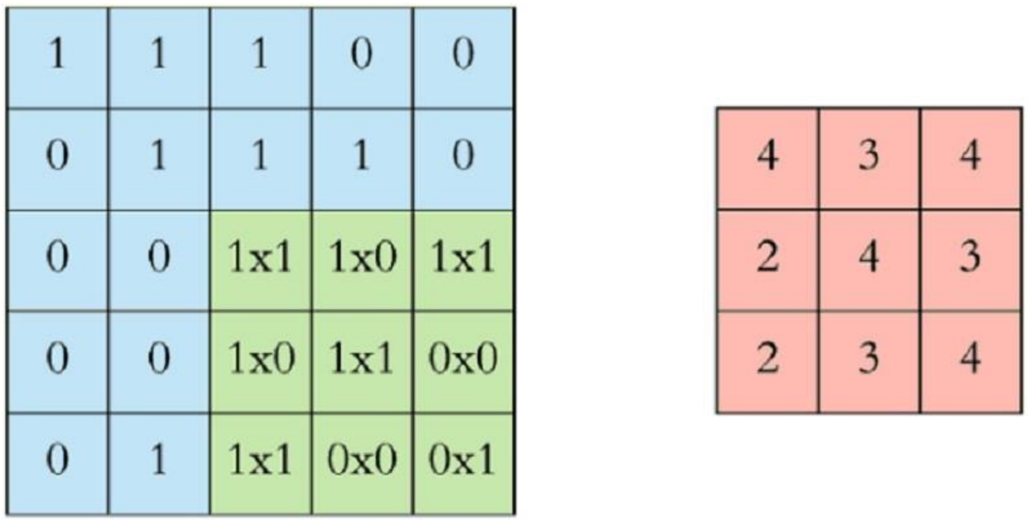


Figure 2.11 Feature map after finishing scanning

Figure 2.11 shows the feature map after finishing scanning.

In feedforward neural network model, each input neural gives each output neural in sub-sequent layers. This model called fully connected neural network. In CNNs model, layers connected through convolution. The following layer is the convolution result from the previous layer, so that we have local connections. Hence, each neuron within the following layer comes about from the channel applied to a local picture area of the previous neuron.

Each class uses distinctive filters. There are numerous such filters combined with others comes about. There are also a number of other layers, just like the pooling layer utilized to filter out valuable information.

2.1.12.Stride and Padding

Stride denotes how many steps we are moving in each steps in convolution. By default it is one. Figure 2.12 shows the stride = 1 example.

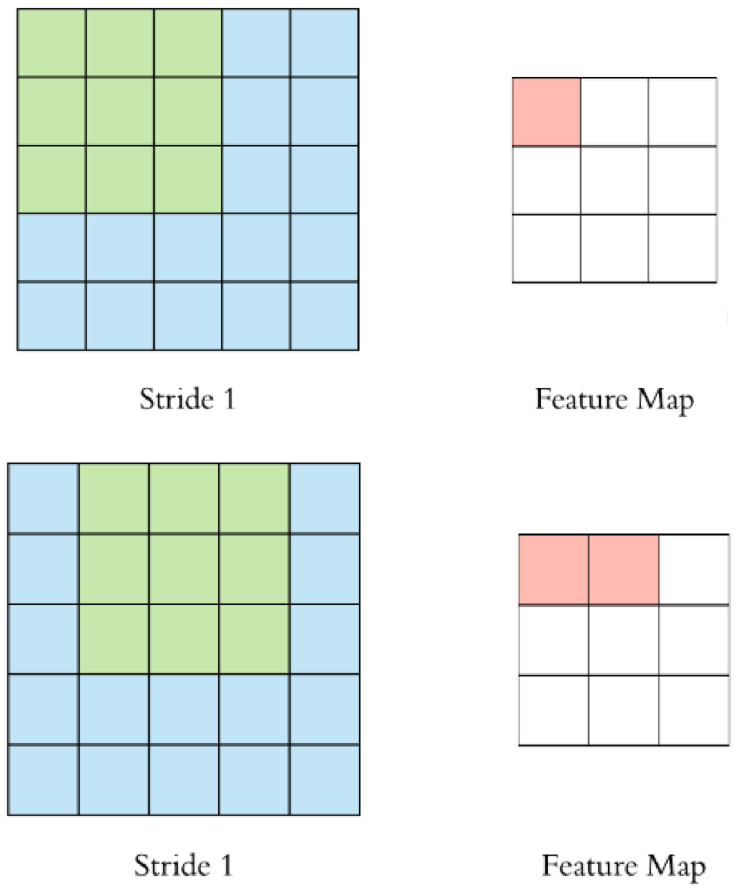
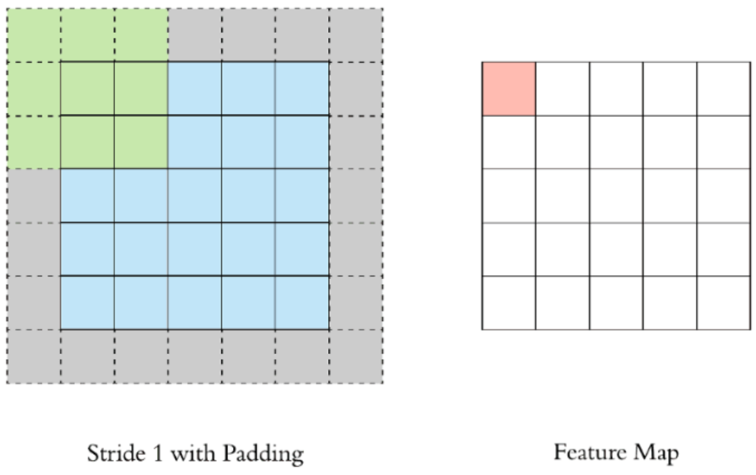


Figure 2.12 Stride = 1 example

To maintain the dimension of output as in input, we use padding. Padding may be a process of adding zeros to the input network symmetrically. Within the following example, the additional grey blocks indicate the padding. It is utilized to form the dimension of output same as input. Figure 2.13 shows the padding example.



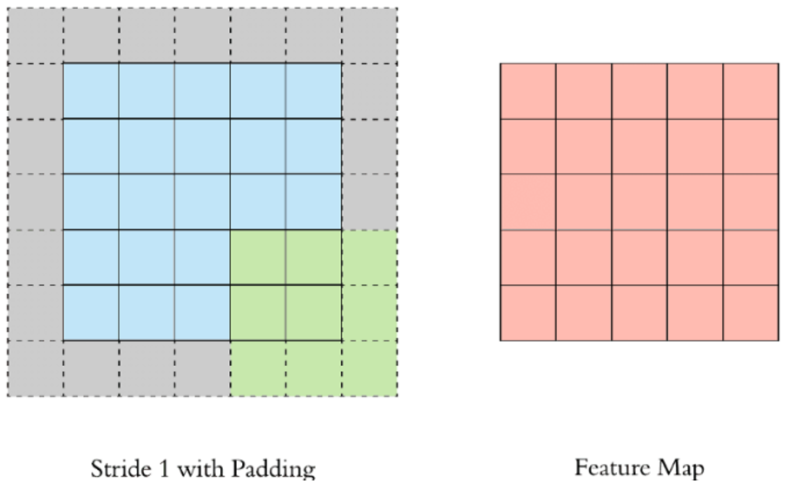


Figure 2.13 Padding example

2.1.13.Local Receptive Fields

Not at all like in a fully connected neural network, CNNs do not have each neuron in one layer connected to every neuron within the following layer. Instead, we only make connections in little 2D localized regions of the input image called the local receptive field. For each responsive field, there's a distinctive hidden neuron in its to begin with hidden layer. This incredibly decreases the number of weights to prepare and the computational complexity of the network.

We are able to pad the input matrix around the border by symmetrically including zeroes. Padding the input makes computation helpful by preserving input volume and avoids fast data loss at border. This padding process controlled by a hyperparameter called zero padding.

The sliding of the local receptive field controlled by another hyperparameter called stride length, which is the number of pixels the field moved at a time. Both these hyperparameters control the spatial size of the output volume. Figure 2.14 shows the local receptive fields.

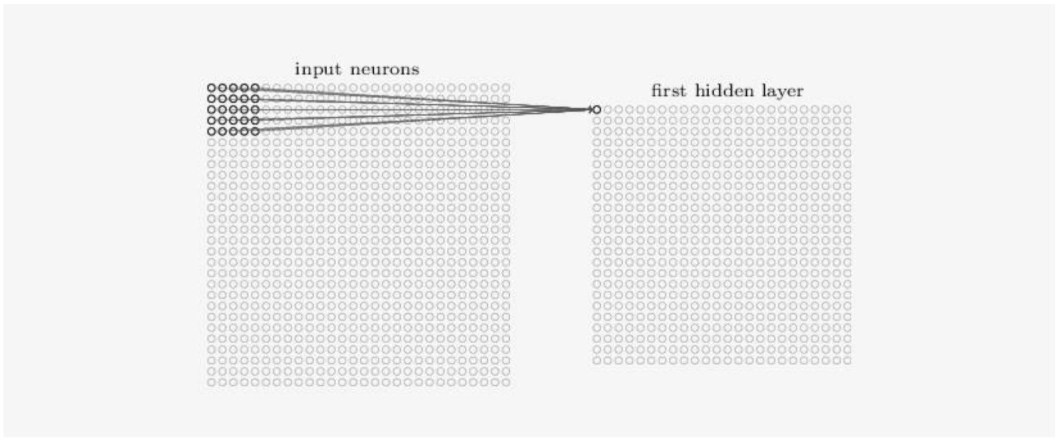


Figure 2.14 Local receptive fields [7]

2.1.14.Shared Weights and Biases

Another specific properties of CNNs is that it uses the same weights and biases for each of the hidden neurons. By sharing the weights and biases, the network is compelled to learn the weights to identify a feature at distinctive parts of the image. This implies that all the neurons within the layer distinguish exactly the same feature, just at distinctive areas within the input [8].

This makes CNNs well adjusted to interpretation invariance of images, which suggests the organize tries to identify a include within the image, and once it recognizes the highlight, the area of the include gets to be unessential.

The weights that define the feature map also called kernel or filter. To perform image recognition, we need more than one feature map. Thus, a convolutional layer consists of several different feature maps.

Another huge advantage of sharing weights and biases is that it moreover incredibly diminishes number of parameters that the network should learn, making a difference in speeding up the learning process. Less number of parameters moreover implies less chance of overfitting the input information. Figure 2.15 shows the share weights example.

[7] https://chsasank.github.io/deep-learning-crash-course-2.html
[8] https://deepnotes.io/intro (read on December 20th, 2019)

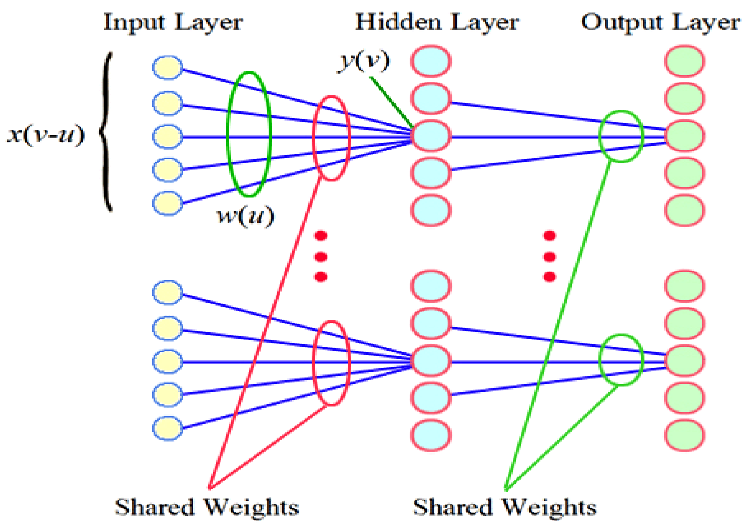


Figure 2.15 Share weights example [9]

2.1.15.Pooling Layers

Pooling layers are another sort of layers regularly utilized after convolutional layers. They disentangle or summarize the data from the convolution layer by performing a measurable total like average or max by taking each include map and creating a down sampled feature map [10].

Most common form of pooling layer used is max pooling, which simply outputs the maximum activation of region of the convolution layer output. Figure 2.16 shows pooling layer example.

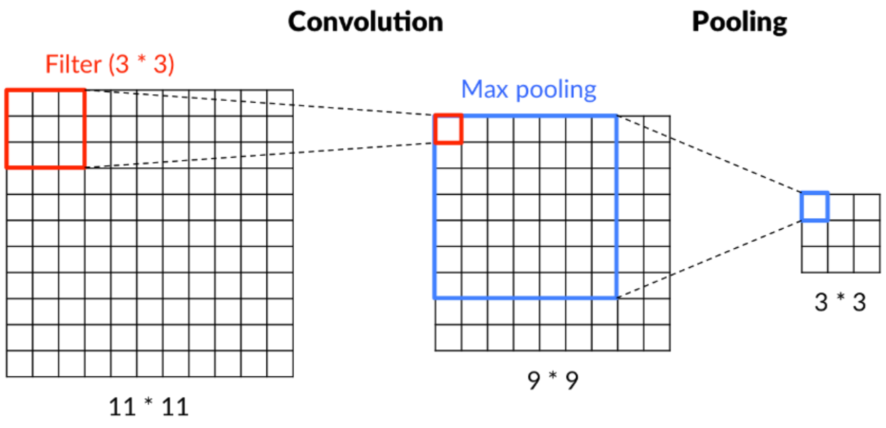


Figure 2.16 Pooling layer example [11]

2.2. Technologies Used

2.2.1. Anaconda

Anaconda is a free open source distribution of python serving for Data Science (DS), Machine Learning (ML) and similar applications such as Big Data processing, predictive analytics, scientific computing, etc. Anaconda towards to manage packages a simple way and suitable for users. The package management system of Anaconda is conda with more than 250 data science packages. Figure 2.17 shows the Anaconda logo.

Anaconda also includes a Graphical User Interface (GUI) called Anaconda Navigator. It is the graphical alternative for the Command Line Interface (CLI).

Anaconda helps user to manage libraries, environments and dependencies between libraries easily. User can also develop ML, Deep Learning (DL) models with scikit-learn, tensorflow and keras, high speed data processing with numpy, pandas, show results with matplotlib, bokeh.

At present, there are two installations of Anaconda: Anaconda2 for Python 2.x and Anaconda3 for Python 3.x. Both of them are compatible with Windows, Linux and MacOS.

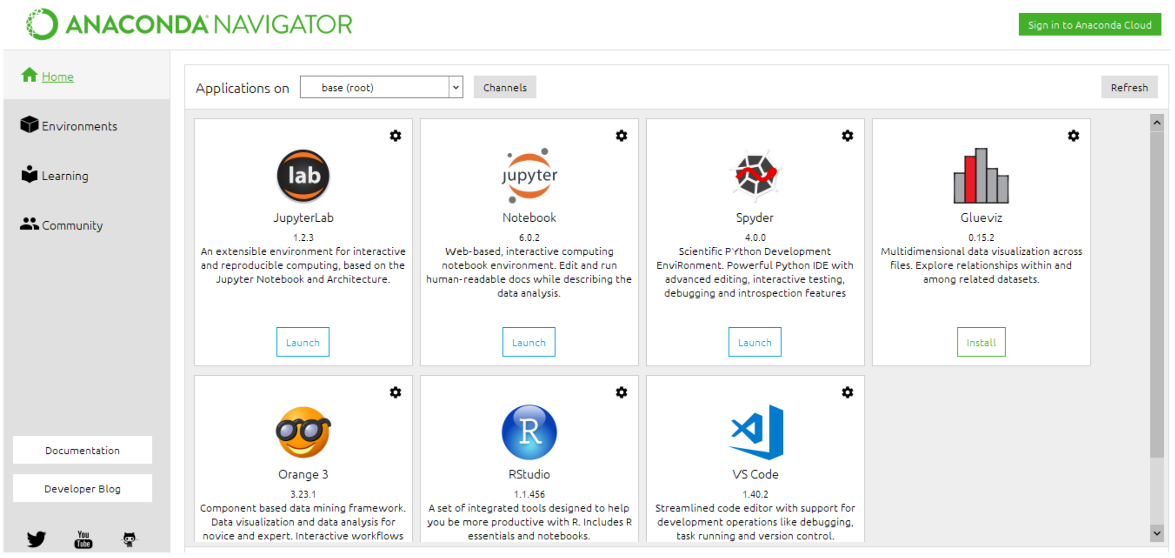
Figure 2.17 Anaconda logo

[9] https://www.researchgate.net/figure/Shared-Weight-Neural-Network-Edit-image-A-full-connected-Neural-Network-is-not-a-good_fig2_260704083
[10] https://deepnotes.io/intro (read on December 21st, 2019)
[11] https://saitoxu.io/2017/01/01/convolution-and-pooling.html

2.2.2. Anaconda Navigator

Anaconda Navigator allow user to launch applications and manage conda packages, environments and channels without using CLI. It can search for packages on Anaconda Cloud or in a local Anaconda Repository then install them in an environment, run the packages and update them.

These following applications are available by default in Anaconda Navigator: Jupyter Notebook, Spyder, Jupyter Lab, Glueviz, Orange, RStudio and Visual Studio Code. Figure 2.18 shows the Anaconda navigator UI.



JupyterLab
1.2.3
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook
6.0.2
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Spyder
4.0.0
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Glueviz
0.15.2
Multidimensional data visualization across files. Explore relationships within and among related datasets.

Orange 3
3.23.1
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows

RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

VS Code
1.40.2
Streamlined code editor with support for development operations like debugging, task running and version control.

Figure 2.18 Anaconda Navigator UI

2.2.3. Conda

Conda is an open-source package management system that allows user to install, run and upgrade packages and their dependencies. Conda not as it were package and disseminate computer program for Python but also for any other languages.

2.2.4. Anaconda Cloud

Anaconda Cloud could be a package management service that allows user to find, access, store, share public and private notebooks, environments, conda and PyPI packages. Anaconda Cloud also allows user to construct unused packages utilizing the Anaconda CLI at that point physically or naturally upload them to Cloud.

2.2.5. Jupyter Notebook

Jupyter Notebook is a web-based interactive computational environment for creating Jupyter Notebook documents which is a JSON document containing an ordered list of input/ouput cells that contain code, text (using Markdown), mathematics, plots and rich media. Figure 2.19 shows the Jupyter Notebook logo.

Major cloud computing suppliers (e.g. Amazon's SageMaker Notebooks, Google's Colaboratory, Microsoft's Azure Notebook...) have used Jupyter Notebook as the cloud user interface.

Figure 2.19 Jupyter Notebook logo

2.2.6. Spyder Environment

Spyder is an open source, cross-platform Integrated Development Environment (IDE) for scientific programming in the Python language designed by and for scientists, engineers and data analysts. It allows user to edit, analyze and debug when they are working about data exploration, interactive execution and deep inspection. Furthermore, Spyder offers built-in integration with many popular scientific packages, including numpy, scipy, pandas, matplotlib and more. Figure 2.20 shows the Spyder logo.

Figure 2.20 Spyder logo

2.2.7. Python Programming Language

Python is an integrated, high-level and general-purpose programming language created by Guido van Rossum and first released in 1991. It aims to readability and ease of writing with simple syntax. Figure 2.21 shows the Python programming language logo.

This language is dynamically typed, garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. It is also available for many operating systems.

Python is one of the most popular languages according to Stack Overflow with the huge developer communities that makes Python a worthwhile language to learn.

Python has a strong high-level data structure for programmers such as number, boolean, string, list, tuple and dictionary [12]:

- Integer: are positive or negative whole numbers with no decimal point.
- Long: are integers of unlimited size, written like integers and followed by an uppercase or lowercase L.
- Float: represent real numbers and are written with a decimal point dividing the integer and fractional parts.

[12] https://www.tutorialspoint.com/python/index.htm (read on December 22nd, 2019)

- Complex: are of the form a + bJ, where a and b are floats and J (or j) represents the square root of -1 (which is an imaginary number). The real part of the number is a, and the imaginary part is b. Complex numbers are not used much in Python programming.
- Boolean: represent one of two values: True or False.
- String: are the most popular types in Python. We can create them simply by enclosing characters in quotes. Python treats single quotes the same as double quotes. Creating strings is as simple as assigning a value to a variable.
- List: are the most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.
- Tuple: is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.
- Dictionary: Each key is separated from its value by a colon (:), the items are separated by commas, and the whole thing is enclosed in curly braces. An empty dictionary without any items is written with just two curly braces, like this: {}. Keys are unique within a dictionary while values may not be. The values of a dictionary can be of any type, but the keys must be of an immutable data type such as strings, numbers, or tuples.

These are applications of Python: web and internet development, scientific and numeric, education, desktop GUIs, software development and business applications.

Figure 2.21 Python programming language logo

2.2.8. OpenCV Library

OpenCV stands for Open Source Computer Vision. It is a free computer vision library that allows you to manipulate images and videos to accomplish a variety of tasks from displaying the feed of a webcam to potentially teaching a robot to recognize real-life objects [5]. Figure 2.22 shows the OpenCV logo.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies [13].

[13] https://opencv.org/about/ (read on December 23rd, 2019)

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV is written natively in C++ and has a template interface that works seamlessly with STL (Standard Template Library) containers.

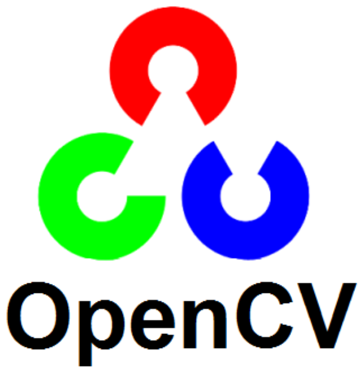


Figure 2.22 OpenCv logo

2.2.9. TensorFlow Library

TensorFlow is an open source software library that strongly supports mathematical operations for calculations in machine learning and deep learning. Its feature is to process all types of data represented as data flow graph. Figure 2.23 shows the TensorFlow logo.

TensorFlow is written in C ++, manipulating the interface in Python so the performance is very good, using both CPU and GPU so it can run on both regular personal computer (PC) and an extremely large server.

Figure 2.23 TensorFlow logo

Chapter 3. SYSTEM ANALYSIS AND DESIGN

3.1. Requirements

Functional requirements:

- User shall be able to train the traffic signs dataset by using Jupyter Notebook.
- User shall be able to run the tool by using the Spyder environment.
- User shall be able to input an image from dataset to test the accuracy.
- User shall be able to input a new image containing actual landscape and traffic sign.
- User shall be able to recognize the traffic sign in the input image.
- User shall be able to receive the result popped up when the tool finishes.
- User shall be able to receive notifications about errors, processes from CLI and message box.
- User shall be able to clear all input paths and quit the tool.

3.2. Use Case Diagram

In our tool, after generating the trained model, user can select the input image through the GUI and recognize the traffic sign in that image.

In addition, they can also select an image in the dataset to check the accuracy of the model.

Moreover, user also have other functions such as clearing all the paths or exit the tool. Figure 3.1 shows the use case diagram.

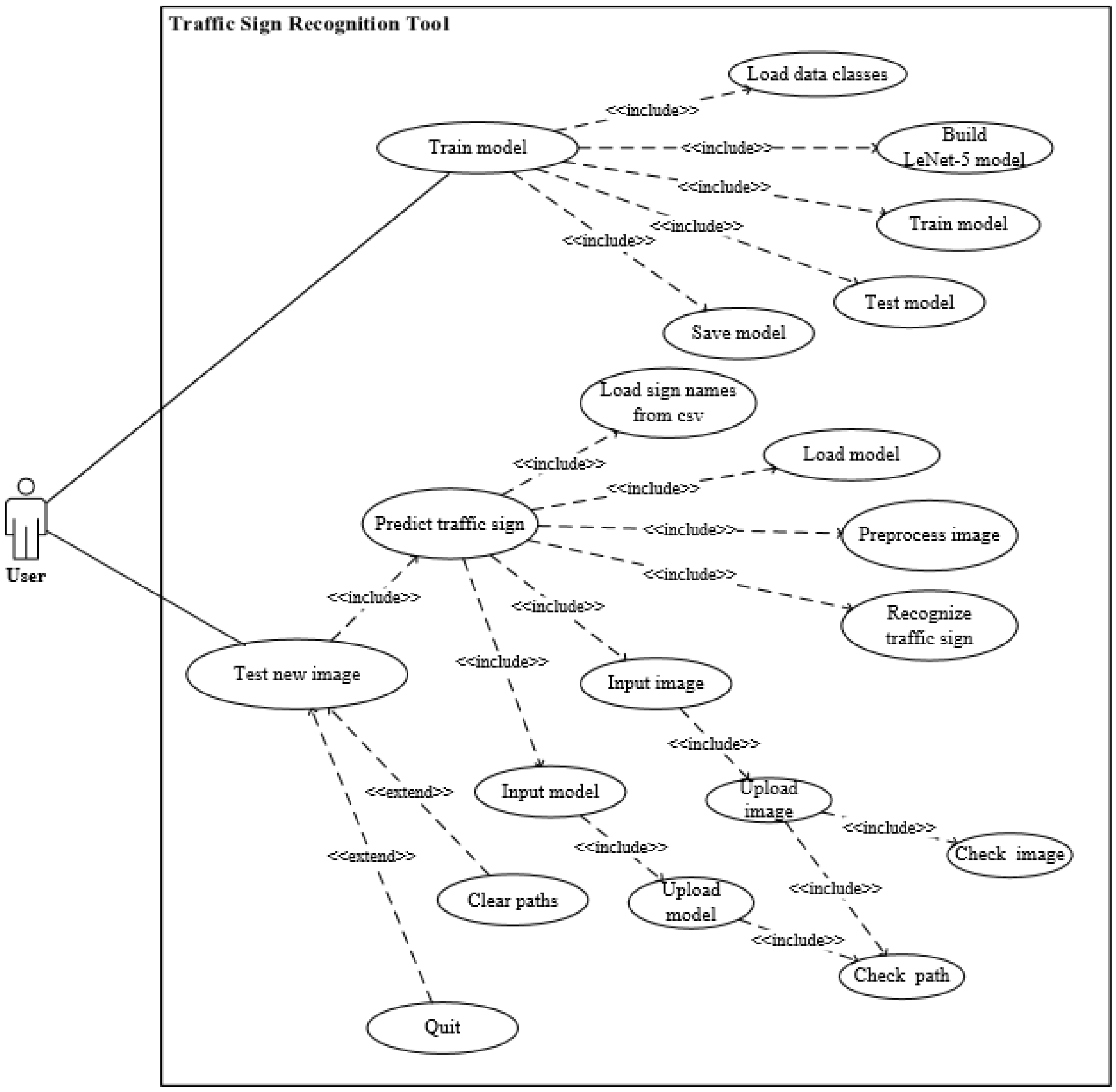


Figure 3.1 Use case diagram

3.3. Activity Diagram

3.3.1. Activity of Training Model

Figure 3.2 illustrates the flow of training model process. User can open the Jupyter Notebook, select the option “Restart and Run All” from the menu bar. The system will proceed to train, validate and test model. User can observe the processes through printed information on CLI.

After training successfully, the system will saved the model in model folder in source directory. This model is used to recognize the traffic sign from input image on the GUI.

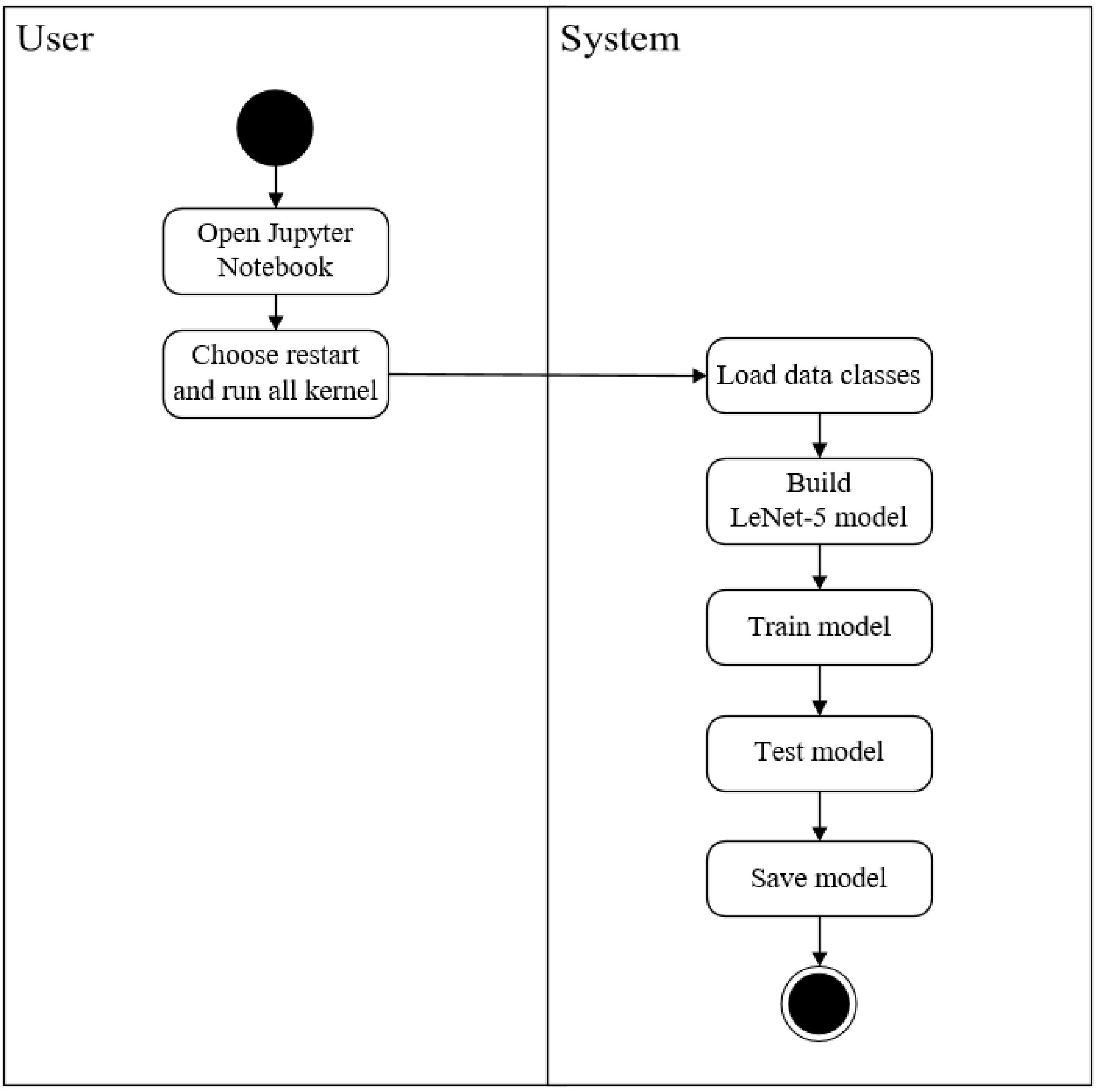


Figure 3.2 Training model activity diagram

3.3.2. Activity of Recognizing Traffic Sign

Figure 3.3 illustrates the flow of recognizing the traffic sign process. From the GUI, user can input the image and the model, then click "Predict" button. The system will proceed to preprocess image, detect the traffic sign contour and recognize it.

User can observe the processes from printed information in CLI and from message box in GUI.

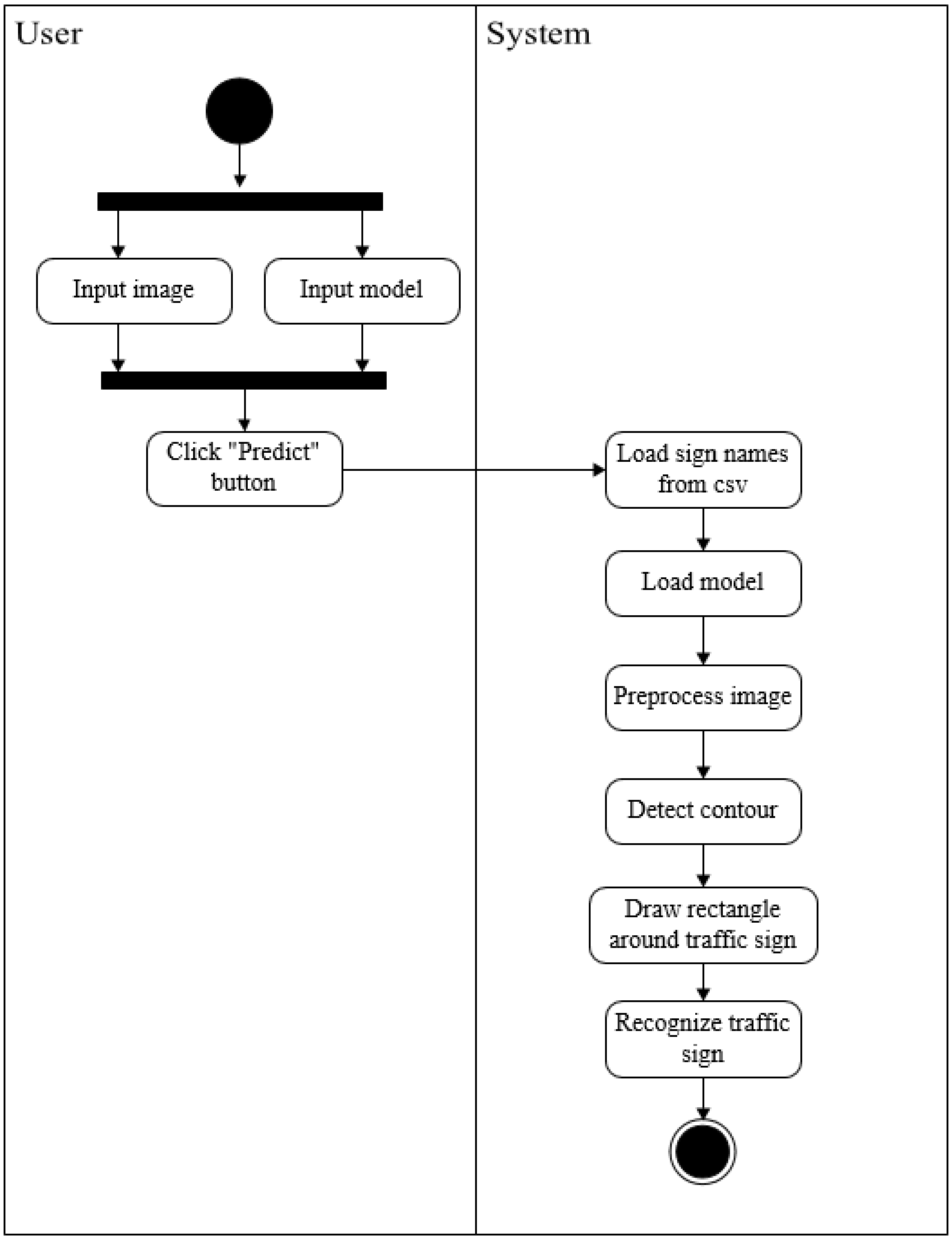


Figure 3.3 Recognizing Traffic Sign activity diagram

3.4. Sequence Diagram

3.4.1. Sequence of Training Model System

Figure 3.4 illustrates the high level of the sequences of training model system.

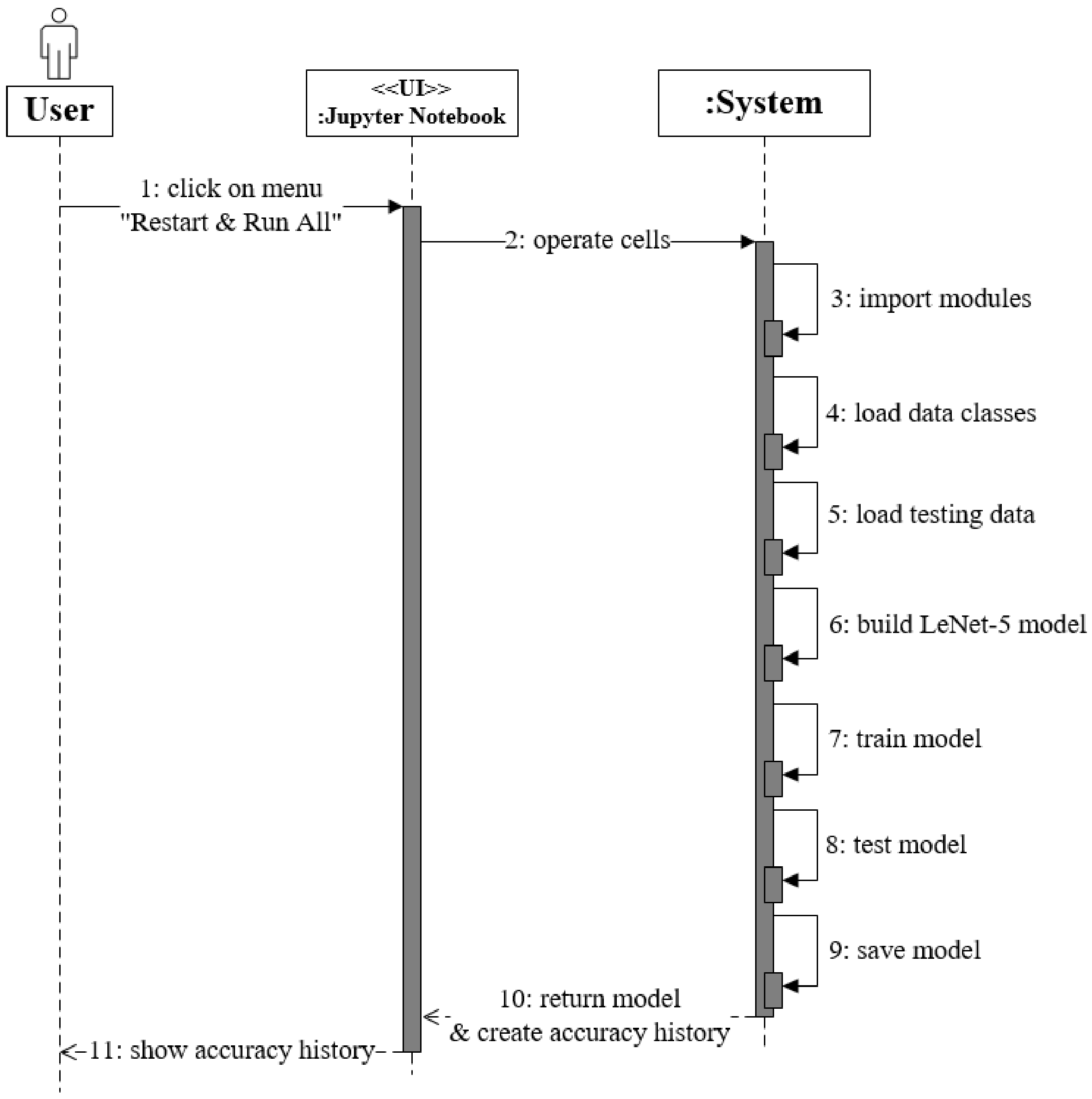


Figure 3.4 Training model sequence system

3.4.2. Sequence of Recognizing Traffic Sign System

Figure 3.5 illustrates the high level of the sequences of recognizing the traffic sign system.

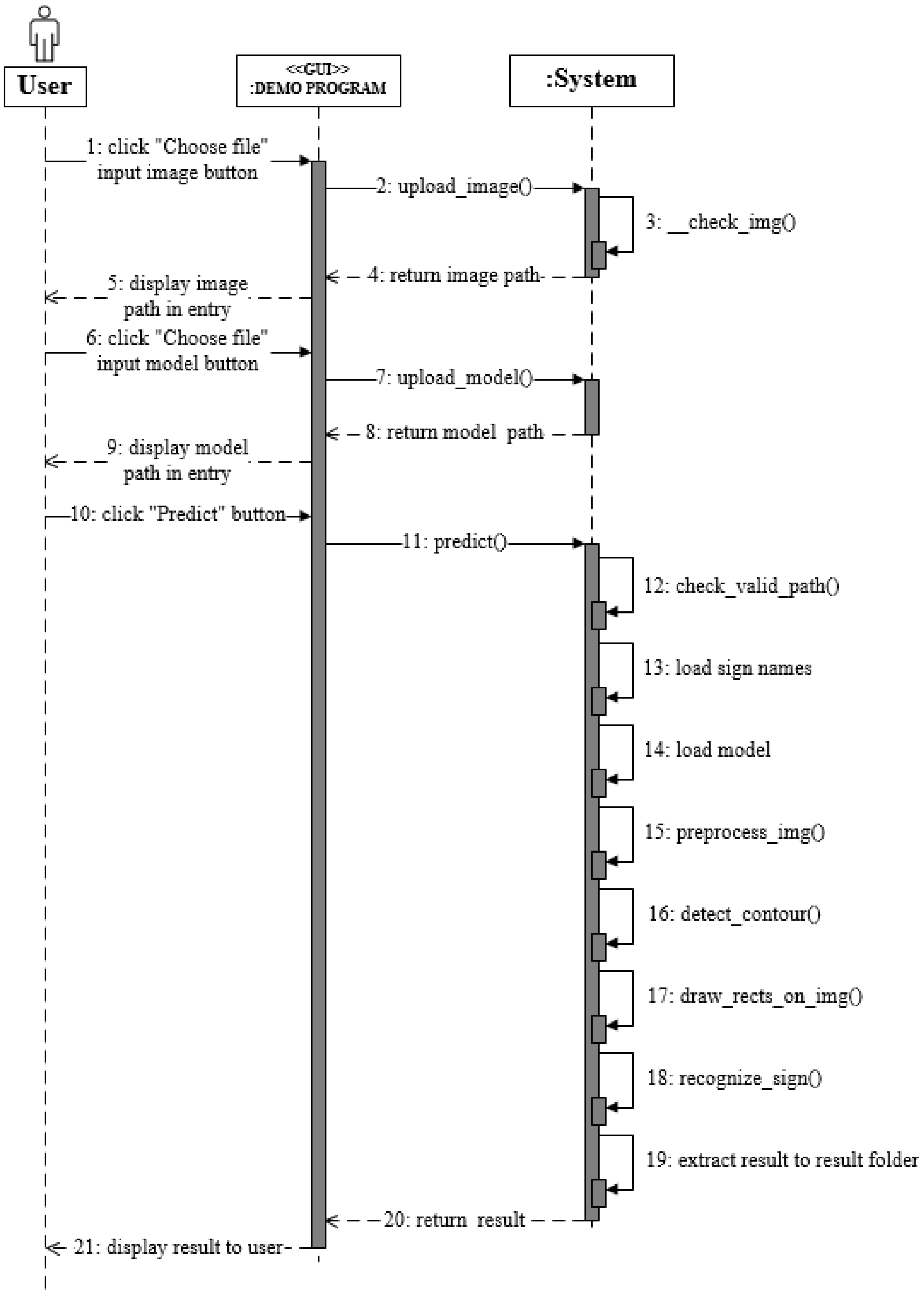


Figure 3.5 Recognizing traffic sign sequence system

Chapter 4. EXPERIMENT AND EVALUATION

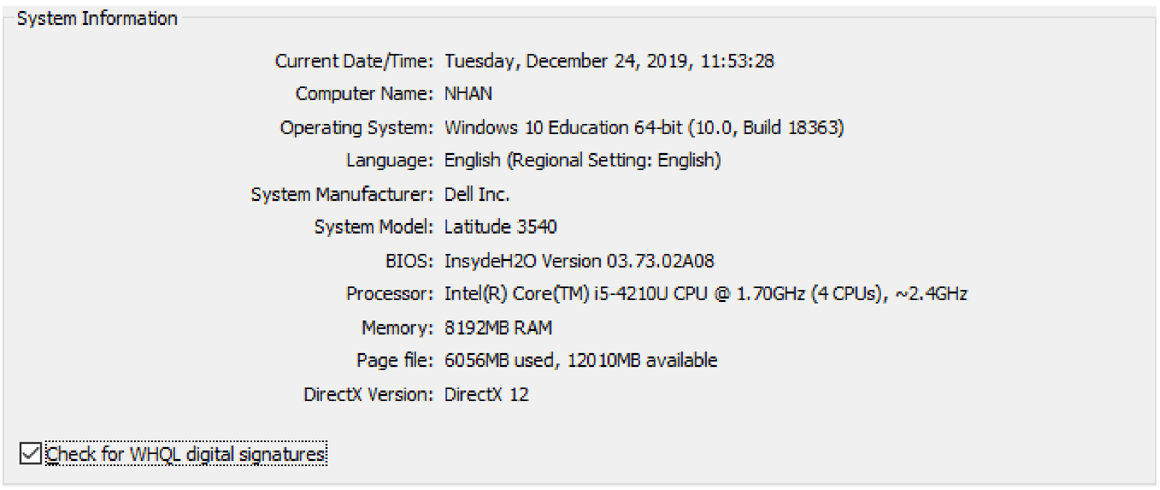
4.1. Experiment

4.1.1. Setting Environment

We set up this tool on:

- Dell Latitude 3540 2015 with an Intel(R) Core(TM) i5-4210U CPU 1.70GHz (4 CPUs) 2.40GHz, 64-bit operating system, x64-based processor, 500GB of HDD, 8GB of RAM.
- Windows 10 Education 64-bit (version 1909, build 18363.535).
- Anaconda Navigator 1.9.7, Jupyter Notebook 6.0.2, Spyder environment 4.0.0 and Python 3.7.5.
- Modules: TensorFlow 2.0.0, OpenCV 4.1.2, NumPy 1.17.4, Pandas 0.25.3, Matplotlib 3.1.2, Keras 2.3.1 and ski-learn 0.21.3.

Figure 4.1 shows the system information.



```
System Information

    Current Date/Time:  Tuesday, December 24, 2019, 11:53:28
        Computer Name:  NHAN
     Operating System:  Windows 10 Education 64-bit (10.0, Build 18363)
             Language:  English (Regional Setting: English)
  System Manufacturer:  Dell Inc.
         System Model:  Latitude 3540
                 BIOS:  InsydeH2O Version 03.73.02A08
            Processor:  Intel(R) Core(TM) i5-4210U CPU @ 1.70GHz (4 CPUs), ~2.4GHz
               Memory:  8192MB RAM
            Page file:  6056MB used, 12010MB available
      DirectX Version:  DirectX 12

☑ Check for WHQL digital signatures
```

Figure 4.1 System information

4.1.2. Dataset

Possessing a high-quality corpus is of significant importance when it comes to training a system. To have such a dataset, we need to get the suitable data in an appropriate format [14].

Firstly, getting the suitable data is to collect the data matching the requirements of the task. For example, in a task of identifying objects in a photo of sport, we collect photos portraying sports events rather than pictures of random topics. In other words, in the first of creating a high-quality dataset, creators need to gather the data that is aligned with the task. However, a high-quality dataset also has to be sizeable, which makes getting the right data become an incredibly challenging job as it requires a good deal of resources. As we try to increase the size of the dataset, the time needed increases. If we want to lessen the amount of time, we need more human resources to annotate the data properly. Also, finding the human resource is difficult because annotators possessing the required domain-specific knowledge are usually costly.

Secondly, we get to have the right data in the right format, which means the data must be processed to match the input representation of the system. Specifically, images, videos, or texts are converted into numerical forms as the input of most machine learning or deep learning systems nowadays is represented as vectors or tensor. Briefly, we need to normalize, standardize, and clean the newly collected raw data so that such systems can perform necessary operations and generate a good result.

4.1.3. K-fold Cross Validation Method

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample [15].

Being easy to understand and implement, k-fold cross validation is among the most popular methods used to evaluate the performance of machine learning systems. In this section, we describe how this procedure is applied.

[14] https://pathmind.com/wiki/datasets-ml (read on December 22nd, 2019)
[15] https://machinelearningmastery.com/k-fold-cross-validation/ (read on December 22nd, 2019)

When the number of samples in the dataset is limited, k-fold cross validation is implemented while the prediction is being made with four main phases. It begins with randomly shuffling the dataset then divides the dataset into k groups/folds, and this is the reason for its name as k-fold. The third phase of the procedure is where the dataset is cross-validated. In this phase, a group is chosen as the test set, while the rest of the corpus is treated as the training set. The training set is then fed into the model and evaluated by the current test set. The evaluation score is kept and summarized in the last step to present the final result of the process of evaluation.

Besides, it is crucial when choosing the value for k. When k is not large enough, for example, the group will not be able to represent the entire dataset in a statistical way. 10 is the recommended value for k, as it is proven by experiments to bring out the best results in most of the cases.

4.1.4. Overfitting Problem

When a model fits too well on the training data that it cannot perform learning on new data, overfitting occurs [16].

To be more specific, in the course of training, the model happens to learn the noise of the training data too much that it creates concepts and rules. When new data that does not follow these rules or concepts is fed into the system, the model refuses to learn new concepts, thereby reducing the performance of the system. In other words, the ability of a model to generalize is negatively impacted when an overfitting problem happens.

On the other hand, overfitting generally happens when using an overly complex model to train a dataset. Nonparametric and nonlinear architectures, for instance, are more likely to suffer from this kind of problem. This is because they can learn more, including detail and noise, than linear models due to their high flexibility.

[16] https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/ (read on December 23rd, 2019)

As overfitting is harmful to the performance of machine learning models, it is important to identify whether the algorithms being used generates overfitting results or not. There are two methods to do so: (1) utilize a resampling technique to predict accuracy and (2) use a validation set. For the first solution, we can choose k-fold cross validation, as described above, to identify overfitting. About the second solution, we divide the training dataset into training and validation sets. We then use the validation set to evaluate the results of training to understand how the model might perform on the test set or unseen data.

4.1.5. Rectified Linear Units Activation Function

Deep learning algorithms use activation functions as a means to transform the input into the output. While activation functions like sigmoid and tanh are inappropriate for neural networks with many layers because of vanishing gradient problem, Rectified Linear Units (ReLU) is the default activation function when implementing a multilayer network.

The ReLU activation function allows a network to easily obtain sparse representations. Apart from being more biologically plausible, sparsity also leads to mathematical advantages [6]. ReLU is designed to look like a linear function to utilize stochastic gradient descent with backpropagation to train deep architectures. Specifically, it is designed to output the positive input and 0 for other cases. In a mathematical, it can be presented as f(x) = max(0, x). Figure 4.2 shows ReLU activation function.

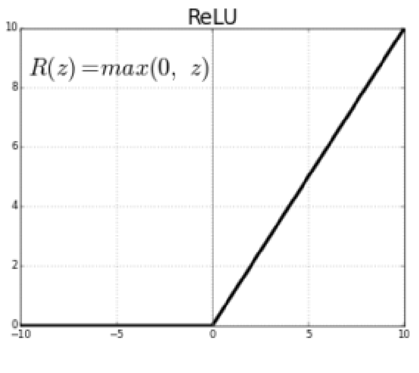


Figure 4.2 ReLU activation function [17]

4.1.6. AdamOptimizer Algorithm

In the course of training neural networks, we use optimizers to increase the performance. Adam optimizer is currently one of the most popular optimizers. This method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients [7].

Adam optimizer earns its ubiquity from its ability to take advantage of the Adaptive Gradient algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). Specifically, instead of changing the learning rate according to the mean, Adam calculates an exponential moving average of the gradient and the mean square and controls the decaying rate of these moves.

4.1.7. German Traffic Sign Recognition Benchmark Dataset

In this thesis, we use the German Traffic Sign Recognition Benchmark (GTSRB), which is a multi-class image classification benchmark in the domain of advanced driver assistance systems and autonomous driving [18].

4.1.7.1. Overview

GTSRB is held at the International Joint Conference on Neural Networks (IJCNN) 2011. This dataset contains single image with more than 40 classes. Each class represents for one type of traffic sign. The total number of images is more than 50,000. GTSRB dataset is often used for multi-classes classification problem with the high reliability due to semi-automatic annotation. Moreover, all the physic traffic sign instances in the dataset are unique.

Figure 4.3 shows the GTSRB dataset illustration.

[17] https://ailephant.com/glossary/relu-function/

[18] http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset (read on December 22nd, 2019)

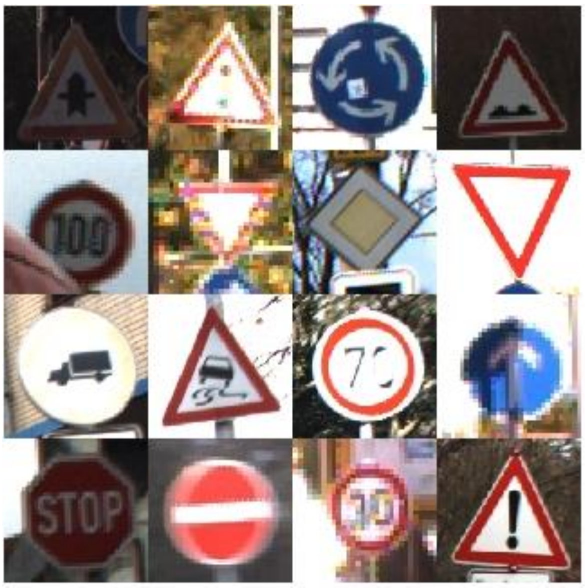


Figure 4.3 GTSRB dataset illustration [19]

4.1.7.2. Structure

The structure of the dataset is described in detail in the homepage. Specifically, one directory for one class. Each directory contains one CSV file named "GT-<ClassID>.csv and the training images. These training images are grouped by tracks. Each track contains 30 images that represents for one single physical traffic sign.

4.1.7.3. Image Format

The GTSRB dataset contains numerous of images, which are formatted clearly. These images are all stored in PPM form and just contain only one traffic sign per image with the size is from 15x15 to 250x250 pixels. The traffic sign is not necessary to place at the center of the image and the bounding box of the traffic sign is a part of the annotations.

[19] http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset

4.1.7.4. Annotation Format

As mentioned above, the GTSRB dataset also contains the annotations format for the images. There are 9 annotations format for each image (Filename, Width, Height, ROI.x1, ROI.y1, ROI.x2, ROI.y2, ClassID and Path). Specifically, the Filename refers to the image name, the Width and Height refer to the width and the height of the image. The ROI (region of interest) x1, y1, x2, y2 refer to the coordinate of the traffic sign in the image. The ClassID refers to the type of that image and the Path refers to the path to the location of the image stored in the input dataset.

Figure 4.4 shows the annotation format example.

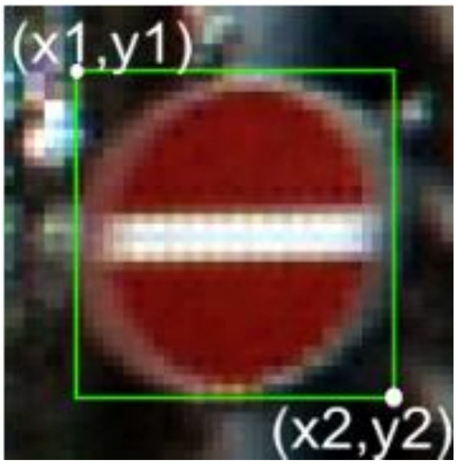


Figure 4.4 Annotation format example [20]

4.1.8. LeNet-5 Architecture

LeNet-5 is a CNN model. It comprises 7 layers, not counting the input, all of which contain trainable parameters (weights). The input is a 32x32 pixel image. [8]

Figure 4.5 shows the LeNet-5 architecture.

[20] http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset

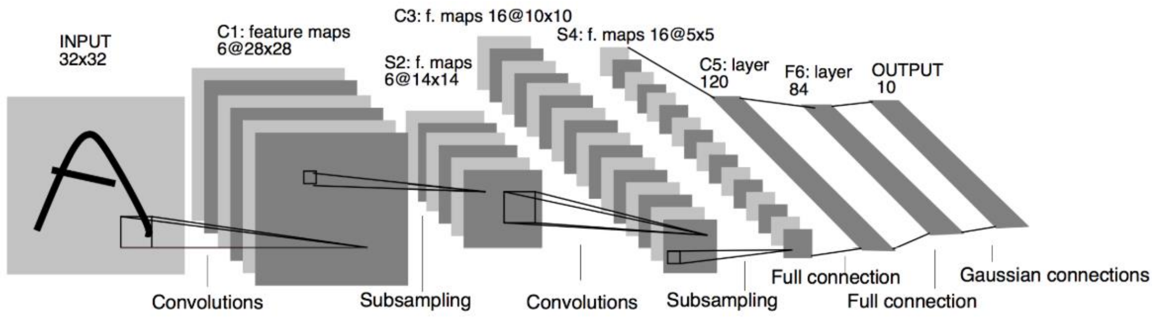


Figure 4.5 LeNet-5 model [8]

In the following, convolutional layers are labeled Cx, subsampling layers are labeled Sx, and fully connected layers are labeled Fx, where x is that the layer index.

Layer C1 may be a convolutional layer with 6 feature maps. Each unit in each feature map is connected to a 5x5 neighborhood within the input. The dimensions of the feature maps are 28x28, which prevents connections from the input falloff the boundary. C1 contains 156 trainable parameters, and 122,304 connections.

Layer S2 may be a subsampling layer with 6 feature maps of size 14x14. Each unit in each feature map is connected to a 2x2 neighborhood within the corresponding feature map in C1. The four inputs to a unit in S2 are added, multiplied by a trainable coefficient. Then they are added to a trainable bias - the result older a sigmoidal function. The 2x2 receptive fields are non-overlapping, therefore feature maps in S2 have a half number of rows and columns as feature maps in C1. Layer S2 and 12 trainable parameters and 5,880 connections.

Layer C3 may be a convolutional layer with 16 feature maps. Each unit in each feature map is connected to many 5x5 neighborhoods at identical locations during a subset of S2's feature maps. the primary six C3 feature maps take inputs from every contigous subsets of three feature maps in S2. The subsequent six take input from every contiguous subset of 4. The subsequent three take input from some discontinuous subsets of 4. Finally, the last one takes input from all S2 feature maps. Layer C3 has 1,516 trainable parameters and 151,600 connections. Table 1 shows the set of S2 feature maps combined with each C3 feature map.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	X				X	X	X			X	X	X	X		X	X
1	X	X				X	X	X			X	X	X	X		X
2	X	X	X				X	X	X			X		X	X	X
3		X	X	X			X	X	X	X			X		X	X
4			X	X	X			X	X	X	X		X	X		X
5				X	X	X			X	X	X	X		X	X	X

Table 1 S2 and S3 feature maps combination [8]

Layer S4 could be a subsampling layer with 16 feature maps of size 5x5. Each unit in each feature map is connected to a 2x2 neighborhood within the corresponding feature map in C3, in a very similar way as C1 and S2. Layer S4 has 32 trainable parameters and a couple of 2,000 connections.

Layer C5 could be a convolutional layer with 120 feature maps. Each unit is connected to a 5x5 neighborhood on all 16 of S4's feature maps. C5 is labeled as a convolutional layer, rather than a totally connected layer. Layer C5 has 48,120 trainable connections.

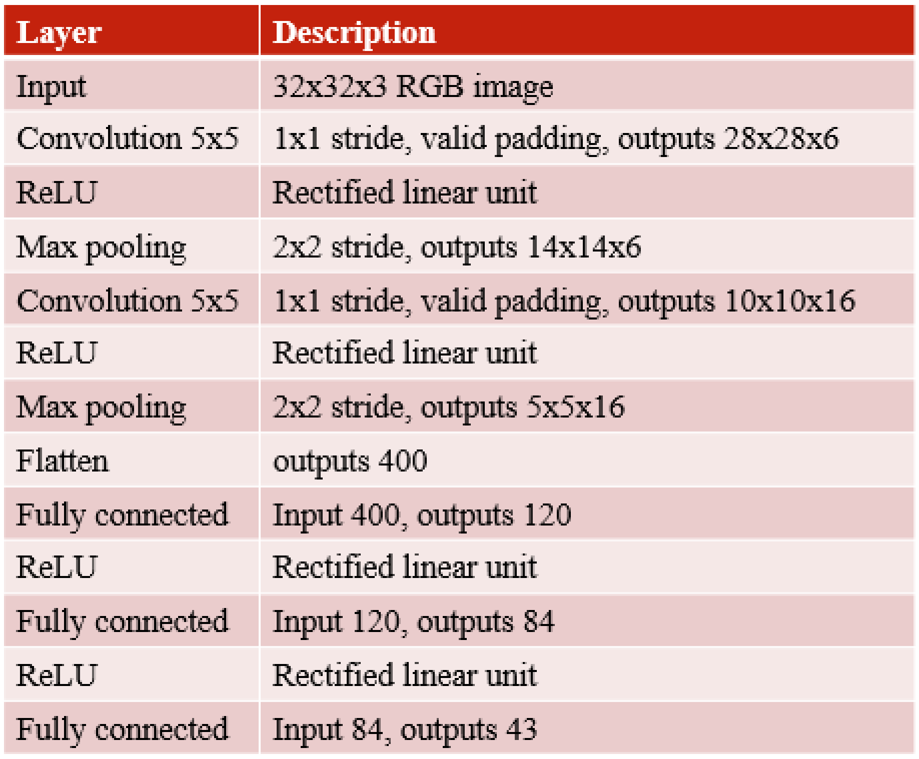
Layer F6 contains 84 units and is fully connected to C5. It has 10,164 trainable parameters.

Finally, the output layer consists of Euclidean Radial Basis Function units (RBF), one for every class, with 84 inputs each. The outputs of every RBF unit yi is computed as Formula 1:

$$y_i = \sum_j (x_j - w_{ij})^2 \qquad (8)$$

Formula 1

In this thesis, we set up LeNet-5 model as Table 2:



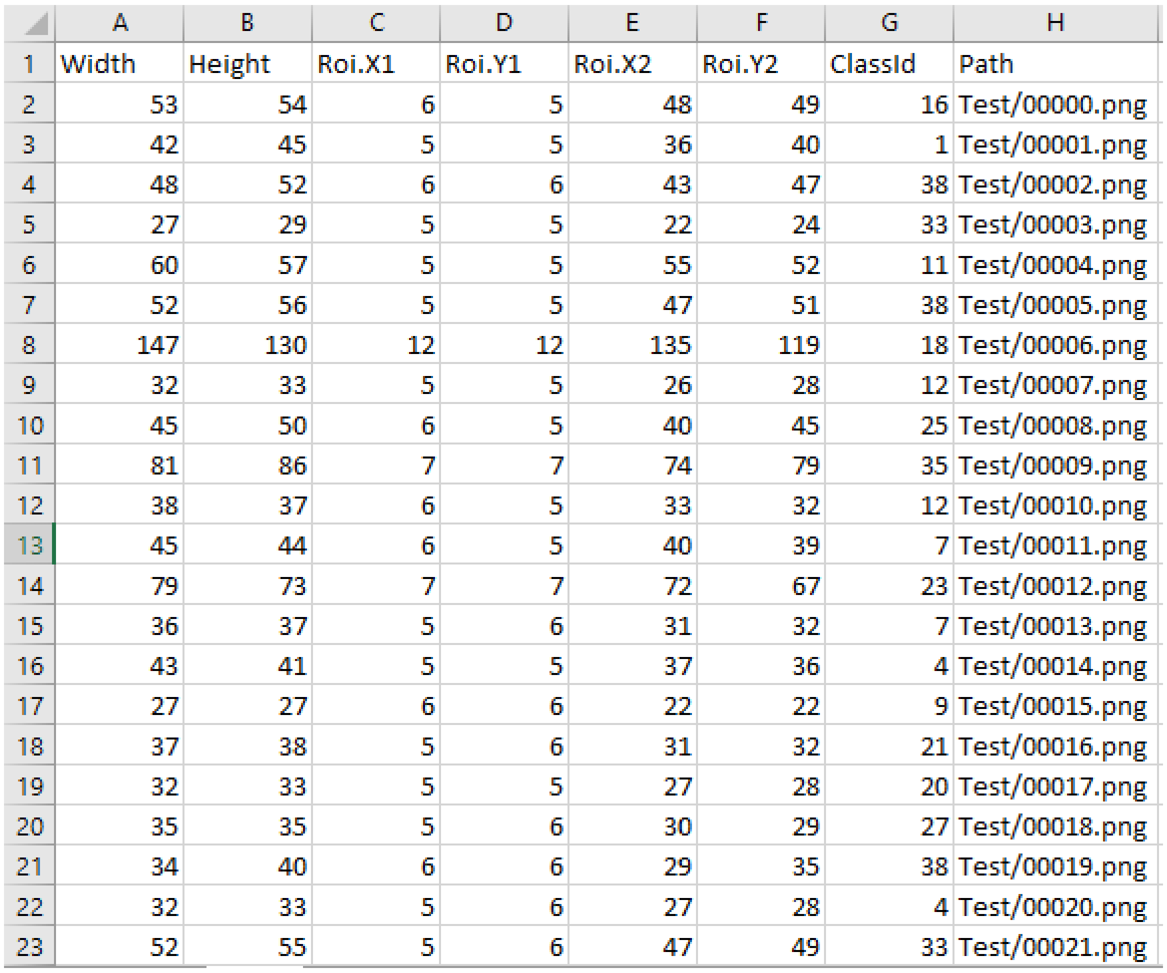
Layer	Description
Input	32x32x3 RGB image
Convolution 5x5	1x1 stride, valid padding, outputs 28x28x6
ReLU	Rectified linear unit
Max pooling	2x2 stride, outputs 14x14x6
Convolution 5x5	1x1 stride, valid padding, outputs 10x10x16
ReLU	Rectified linear unit
Max pooling	2x2 stride, outputs 5x5x16
Flatten	outputs 400
Fully connected	Input 400, outputs 120
ReLU	Rectified linear unit
Fully connected	Input 120, outputs 84
ReLU	Rectified linear unit
Fully connected	Input 84, outputs 43

Table 2 LeNet-5 model

4.1.9. Experiment Results

4.1.9.1. Train Model

In this thesis, we use dataset from GTSRB as input. The input folder includes a folder named "train" containing subfolders corresponding to 43 classes representing 43 types of traffic signs, a folder named "test" consisting of traffic signs used to check the accuracy of the trained model, 1 excel/csv file named "signnames" saves ID and corresponding sign name and 1 excel/csv file named "Test" used to store information of an image in the test folder such as height, width , ROI.X1, ROI.X2, ROI.Y1, ROI.Y2, ClassId and Path. Figure 4.6 shows the Test.csv file content.

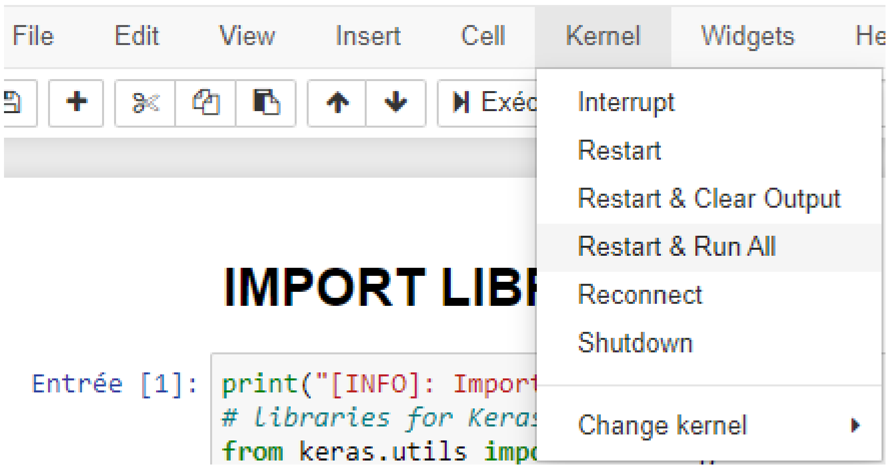


	A	B	C	D	E	F	G	H
1	Width	Height	Roi.X1	Roi.Y1	Roi.X2	Roi.Y2	ClassId	Path
2	53	54	6	5	48	49	16	Test/00000.png
3	42	45	5	5	36	40	1	Test/00001.png
4	48	52	6	6	43	47	38	Test/00002.png
5	27	29	5	5	22	24	33	Test/00003.png
6	60	57	5	5	55	52	11	Test/00004.png
7	52	56	5	5	47	51	38	Test/00005.png
8	147	130	12	12	135	119	18	Test/00006.png
9	32	33	5	5	26	28	12	Test/00007.png
10	45	50	6	5	40	45	25	Test/00008.png
11	81	86	7	7	74	79	35	Test/00009.png
12	38	37	6	5	33	32	12	Test/00010.png
13	45	44	6	5	40	39	7	Test/00011.png
14	79	73	7	7	72	67	23	Test/00012.png
15	36	37	5	6	31	32	7	Test/00013.png
16	43	41	5	5	37	36	4	Test/00014.png
17	27	27	6	6	22	22	9	Test/00015.png
18	37	38	5	6	31	32	21	Test/00016.png
19	32	33	5	5	27	28	20	Test/00017.png
20	35	35	5	6	30	29	27	Test/00018.png
21	34	40	6	6	29	35	38	Test/00019.png
22	32	33	5	6	27	28	4	Test/00020.png
23	52	55	5	6	47	49	33	Test/00021.png

Figure 4.6 Test.csv file content

At this point, we will perform data training just like the activity diagram in Figure 3.2.

From Jupyter Notebook, we select "Restart & Run All" (Figure 4.7) from the Kernel menu then the system will do tasks such as loading data classes, building LeNet-5 models, training with the loaded data classes and testing the trained model with a test set to measure the model's accuracy.



```
Entrée [1]: print("[INFO]: Import
            # libraries for Kera
            from keras.utils imp
```

Figure 4.7 Restart & Run All

The pixel data of each image is normalized before it is fed into the neural network. The output of the neural network is also normalized using the softmax function to produce logits in the range [0, 1]. Our choice for an activation function is ReLU.

The AdamOptimizer algorithm is used to optimize the objective function, instead of the gradient descent algorithm. The Adam algorithm uses momentum to zone-in on the ideal learning-rate during training, unlike the gradient descent algorithm where the learning rate hyperparameter will have to be manually tuned at the start and does not change during the training process.

K-fold cross validation will be the method to split into training sets and validation sets. Specifically, we set k = 5, meaning that the training set is divided into 5 episodes: D1, D2, D3, D4 and D5. The system will train 5 times, each time choose 1 for a validation set, the remaining 4 episodes will be training sets. For example, the first time, D1 is the validation set, D2, D3, D4, D5 are the training sets, the second, D2 is the validation set, D1, D3, D4, D5 are the training sets. Once the training is completed, the model will use the test set to check accuracy. The model with the highest accuracy will be selected and saved in the model folder. Training by this method will help objectively assess the accuracy of the model through the different training sets and validation sets.

Besides, as mentioned above, identifying training set, validation set as well as test set and data training method are very important. If we determine how to train and divide the data set unreasonably, we may face overfitting.

Finally, we save the model with the highest accuracy to the "model" folder in the source directory.

Figure 4.8 shows the accuracy of 5 trained models.

```
[INFO]: SUMMARY:
[INFO]: Model 1 accuracy: 0.9159144893111639
[INFO]: Model 2 accuracy: 0.8558194774346793
[INFO]: Model 3 accuracy: 0.872921615201900З
[INFO]: Model 4 accuracy: 0.8453681710213776
[INFO]: Model 5 accuracy: 0.8832937450514647
--------------------
[INFO]: Average accuracy: 0.8746634996041172
[INFO]: Highest accuracy: 0.9159144893111639

Text(0.5, 0, 'Accuracy')
```

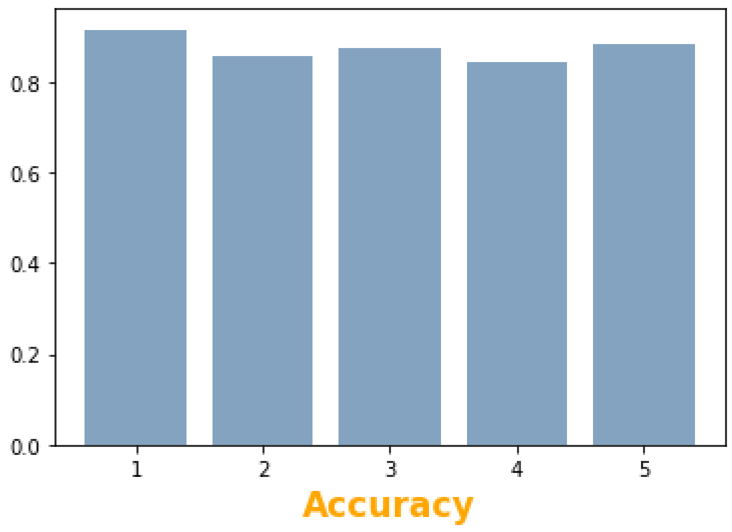


Figure 4.8 Trained models accuracy

In this case, the average accuracy of 5 trained models is 0.87% and the highest accuracy is 0.92%. Thus, the system will select the model 1 by the result.

4.1.9.2. Recognize Traffic Sign

After successful model training, we will implement the traffic sign recognition according to operation diagram in Figure 3.3.

From Spyder, we run the program and the tool GUI appears. (Figure 4.9)

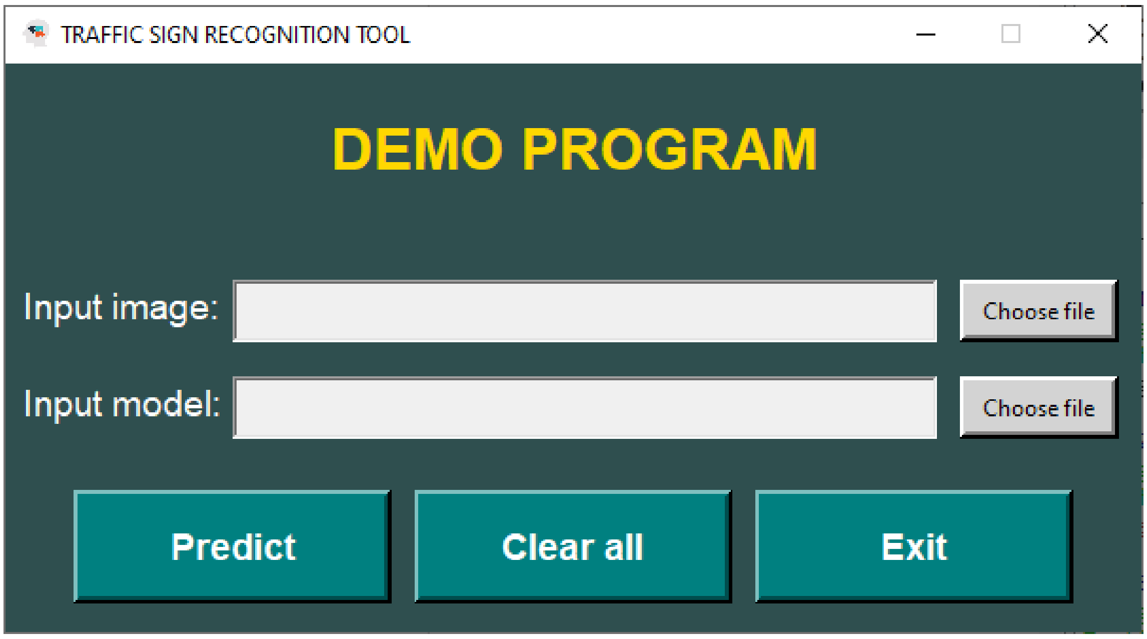
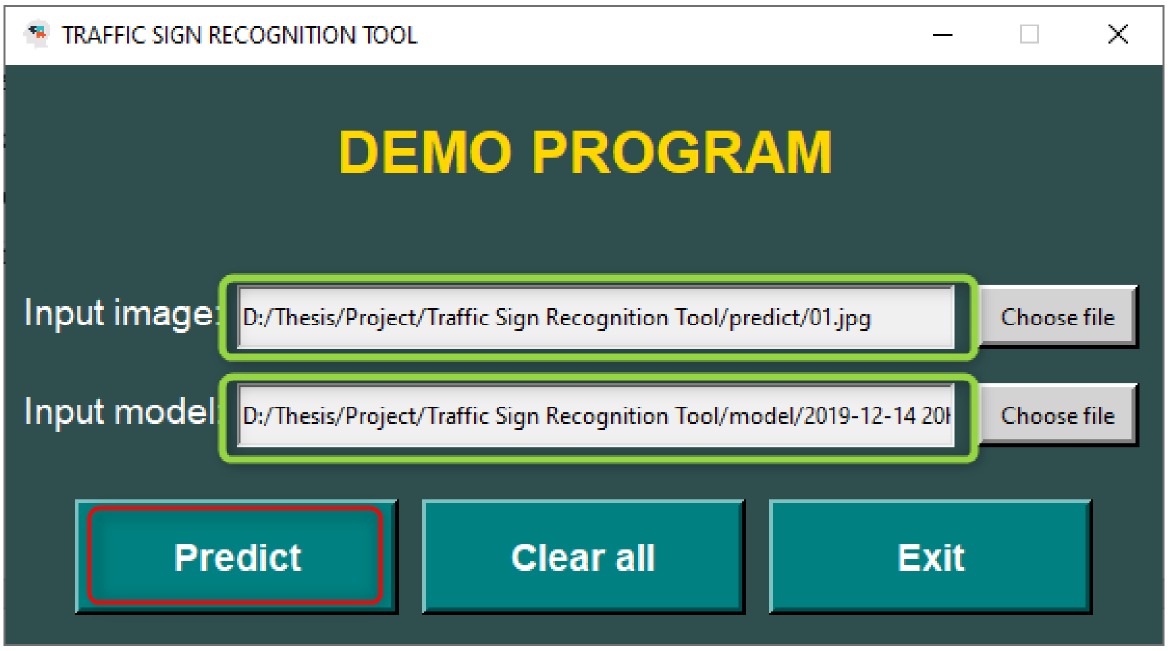


Figure 4.9 Traffic sign recognition tool GUI

In this interface, we will enter the input image path and the model already trained and then click the Predict button. At this time, the system will carry out the image prediction process including steps such as loading sign names file, loading model, pre-processing image, identifying the location and framing the traffic sign included in the input image, putting that frame into the model and recognizing the traffic sign.

In Figure 4.10, we input an image and a trained model.



D:/Thesis/Project/Traffic Sign Recognition Tool/predict/01.jpg

D:/Thesis/Project/Traffic Sign Recognition Tool/model/2019-12-14 20h

Figure 4.10 Input image and model paths

Figure 4.11 shows the original image we have input.

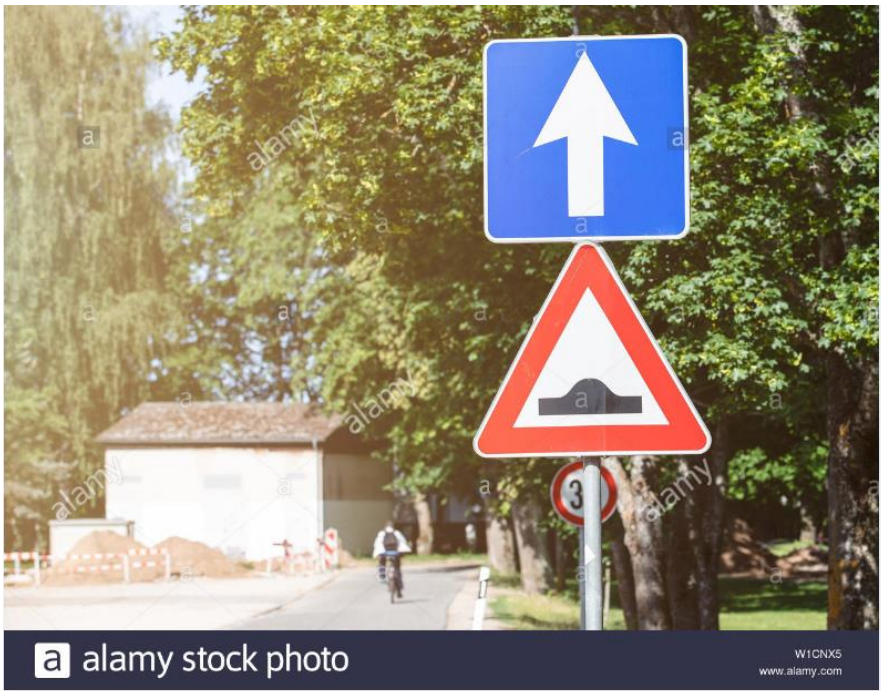


Figure 4.11 Original input image

Specifically, during image preprocessing, we use imread () function of OpenCV to read the input image. The input is a RGB image, we convert the image to HSV to exploited image color filtering. Because the R, G and B components of the object in a digital image are all correlated with the amount of light hitting the object, image descriptions of those components make the distinction become difficult. Descriptions of colors/brightness/shades or colors/brightness/saturation are usually more appropriate. After preprocessing image, we return a binary image in which blue and red colors are filtered (Figure 4.12).

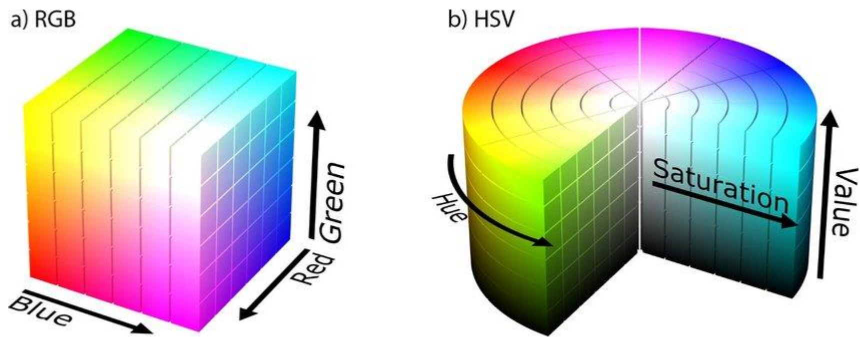


Figure 4.12 RGB and HSV [21]

Figure 4.13 shows the binary image.

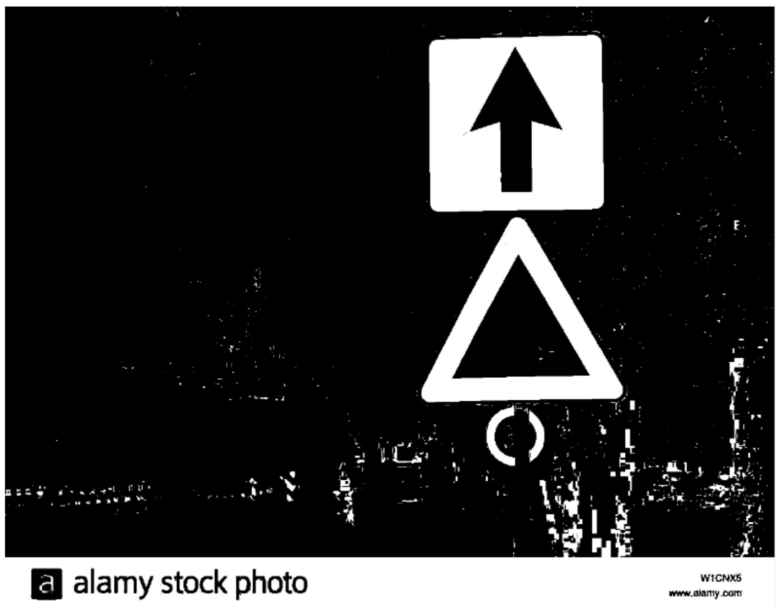


Figure 4.13 Binary image result

[21] https://observablehq.com/@anbnyc/color-on-the-web

From the binary image, we use findContours () function of OpenCV to detect contours of the traffic sign.

Figure 4.14 shows the image with contours around 2 traffic signs.

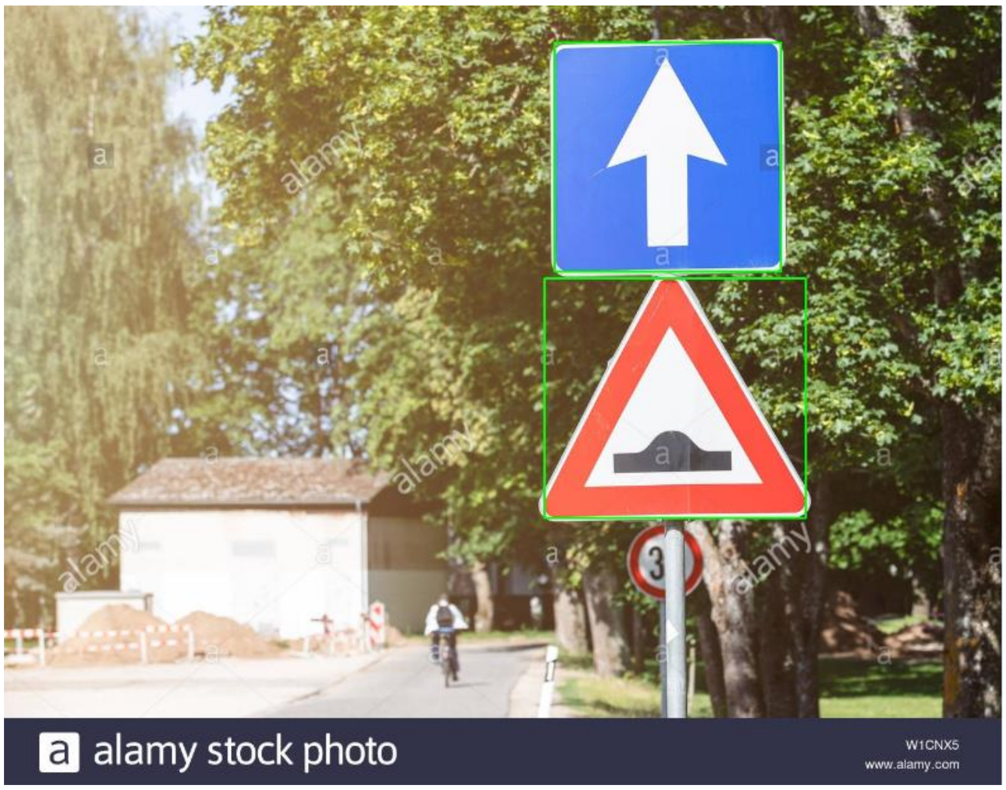


Figure 4.14 Image with contours around

After obtaining the location of the traffic sign in the input image, we resize that region into a 32x32 image and put it in the trained model. The model analyzes and outputs the name of the traffic sign.

In this case, the tool recognized correctly the bumpy road sign and ahead only sign. We use imwrite() function of OpenCV to save the binary image, the image with contours around and the result image in "result" folder.

Figure 4.15 shows the final recognition result of the tool.

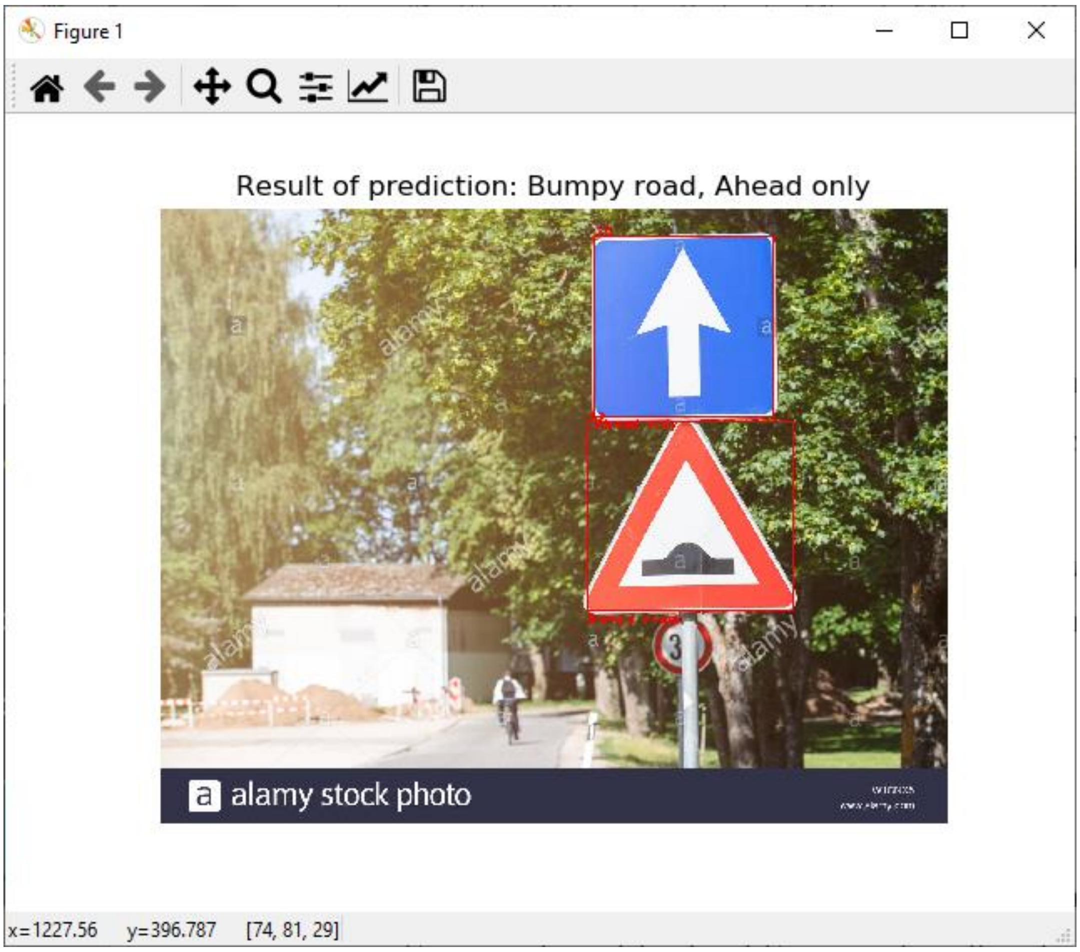


Figure 1
Result of prediction: Bumpy road, Ahead only
alamy stock photo
x=1227.56 y=396.787 [74, 81, 29]

Figure 4.15 Recognition result

4.2. Evaluation

To evaluate the tool performance, we used 40 sample images. In those images, there are 15 images from dataset and 25 images from the internet. Table 3 shows the input image information descriptions.

No.	Label	Width	Height	ROI .X1	ROI .Y1	ROI .X2	ROI .Y2	Class ID
1	Vehicles over 3.5 metric tons prohibited	53	54	6	5	48	49	16
2	Speed limit (30km/h)	59	65	5	6	54	60	1

3	Keep right	48	52	6	6	43	47	38
4	Speed limit (70km/h)	53	54	5	6	48	49	4
5	Turn right ahead	52	55	5	6	47	49	33
6	Keep right	52	56	5	5	47	51	38
7	General caution	147	130	12	12	135	119	18
8	Priority road	38	37	6	5	33	32	12
9	No vehicles	42	42	5	6	37	37	15
10	No entry	29	34	5	6	24	29	17
11	Speed limit (100km/h)	71	70	7	6	65	64	7
12	No passing	52	51	6	5	47	46	9
13	Road narrows on the right	75	66	7	6	68	60	24
14	Traffic signals	88	82	8	8	80	75	26
15	Roundabout mandatory	43	43	5	5	38	38	40
16	+ Bumpy road + Ahead only	1300	1018	N/A	N/A	N/A	N/A	N/A
17	Keep right	1300	953	N/A	N/A	N/A	N/A	N/A
18	Right-of-way at the next intersection	1000	750	N/A	N/A	N/A	N/A	N/A
19	Wild animal crossing	1300	867	N/A	N/A	N/A	N/A	N/A
20	Ahead only	500	334	N/A	N/A	N/A	N/A	N/A
21	Speed limit (30km/h)	270	194	N/A	N/A	N/A	N/A	N/A
22	Road work	1300	866	N/A	N/A	N/A	N/A	N/A

23	Stop	1200	1296	N/A	N/A	N/A	N/A	N/A
24	Children crossing	424	680	N/A	N/A	N/A	N/A	N/A
25	Speed limit (50km/h)	1300	956	N/A	N/A	N/A	N/A	N/A
26	No entry	800	533	N/A	N/A	N/A	N/A	N/A
27	General caution	866	1390	N/A	N/A	N/A	N/A	N/A
28	Bumpy road	1300	953	N/A	N/A	N/A	N/A	N/A
29	Roundabout mandatory	1024	1024	N/A	N/A	N/A	N/A	N/A
30	Slippery road	865	1390	N/A	N/A	N/A	N/A	N/A
31	+ Road work + Children crossing	910	480	N/A	N/A	N/A	N/A	N/A
32	Priority road	612	459	N/A	N/A	N/A	N/A	N/A
33	Traffic signals	1024	684	N/A	N/A	N/A	N/A	N/A
34	Stop	800	1067	N/A	N/A	N/A	N/A	N/A
35	General caution	768	1024	N/A	N/A	N/A	N/A	N/A
36	Stop	640	1280	N/A	N/A	N/A	N/A	N/A
37	Wild animal crossing	850	566	N/A	N/A	N/A	N/A	N/A
38	+ Speed limit (50km/h) + Speed limit (70km/h) + Speed limit (80km/h) + Speed limit (100km/h) + Speed limit (120km/h)	450	326	N/A	N/A	N/A	N/A	N/A

39	Speed limit (30km/h)	863	1390	N/A	N/A	N/A	N/A	N/A
40	No vehicles	500	334	N/A	N/A	N/A	N/A	N/A

Table 3 Input image information descriptions

Note that from number 1 to number 15, we take the images from the dataset so we have information such as width, height, ROIs and ClassID. From number 16 to number 40, we take the images from the internet so we only have information like width and height. ROIs and ClassID will be N/A (Not Available) because this is exactly what our tool will recognize.

Figure 4.16 shows the input images from the internet.

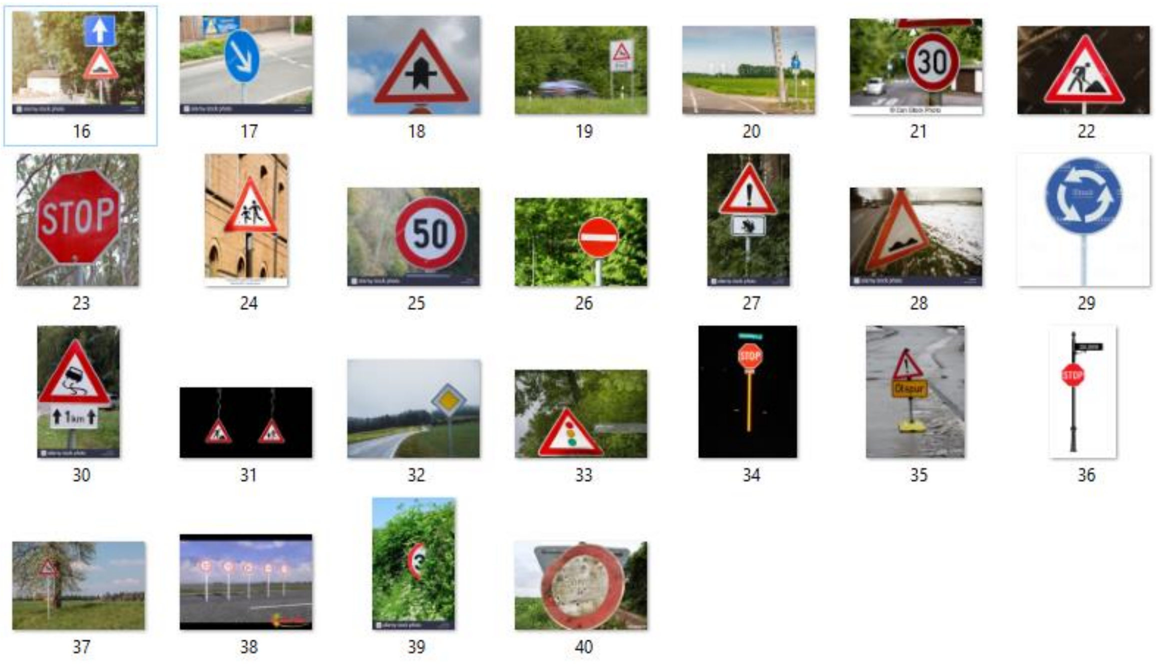


Figure 4.16 New input images from the internet

The tool can accept various input images from the internet. These images are much larger than the images in the dataset. Each of these images contains one or more traffic signs and landscape. They can be taken from digital cameras, phones, dashcams or even cut from video clips. In this case, we input types of images such as good quality, high-contrast images between the objects and the images with obscured or tilted or ruined traffic signs and used the saved model to recognize the traffic signs. The result is in the Table 4 below:

No.	Label	Correct Value	Incorrect Value
1	Vehicles over 3.5 metric tons prohibited	Vehicles over 3.5 metric tons prohibited	N/A
2	Speed limit (30km/h)	Speed limit (30km/h)	N/A
3	Keep right	Keep right	N/A
4	Speed limit (70km/h)	Speed limit (70km/h)	N/A
5	Turn right ahead	Turn right ahead	N/A
6	Keep right	Keep right	N/A
7	General caution	General caution	N/A
8	Priority road	Priority road	N/A
9	No vehicles	No vehicles	N/A
10	No entry	No entry	N/A
11	Speed limit (100km/h)	Speed limit (100km/h)	N/A
12	No passing	No passing	N/A
13	Road narrows on the right	Road narrows on the right	N/A
14	Traffic signals	Traffic signals	N/A
15	Roundabout mandatory	Roundabout mandatory	N/A
16	+ Bumpy road + Ahead only	+ Bumpy road + Ahead only	N/A
17	Keep right	Keep right	N/A

18	Right-of-way at the next intersection	Right-of-way at the next intersection	Yield
19	Wild animal crossing	Wild animal crossing	N/A
20	Ahead only	Ahead only	N/A
21	Speed limit (30km/h)	Speed limit (30km/h)	Unidentified
22	Road work	Road work	N/A
23	Stop	Stop	N/A
24	Children crossing	Children crossing	N/A
25	Speed limit (50km/h)	Speed limit (50km/h)	N/A
26	No entry	No entry	N/A
27	General caution	General caution	N/A
28	Bumpy road	Bumpy road	N/A
29	Roundabout mandatory	Roundabout mandatory	N/A
30	Slippery road	Slippery road	N/A
31	+ Road work + Children crossing	+ Road work + Children crossing	N/A
32	Priority road	Priority road	N/A
33	Traffic signals	Traffic signals	N/A
34	Stop	Stop	N/A
35	General caution	General caution	N/A
36	Stop	Stop	N/A
37	Wild animal crossing	Wild animal crossing	+ Go straight or left

			+ Unidentified
38	+ Speed limit (50km/h) + Speed limit (70km/h) + Speed limit (80km/h) + Speed limit (100km/h) + Speed limit (120km/h)	N/A	+ No entry + Keep right + Ahead only
39	Speed limit (30km/h)	N/A	Can not recognize
40	No vehicles	N/A	Unidentified

Table 4 Recognition result

As shown in Table 4, the average accuracy of recognizing traffic signs in above input images is 85%.

Note that if one of the 2 columns Correct Value and Incorrect Value has a value of N/A, it means that the tool predicted completely correct or incorrect. If both columns have the value of a traffic sign name, the tool may have recognized more than the expected number of traffic signs. This means that the tool has predicted correctly but it is accompanied by a residual prediction, such as predicting a region that is not a traffic sign.

After the experimental process, we realized that the tool can recognize traffic signs as the original target. However, the accuracy is still not absolute and needs to be improved more. With the analyzing the input images, we can see that most of the input images which are clear, no color overlap between objects, no noise or being obscured, the tool is able to recognize the traffic sign with significant accuracy. As for the input images that are blurred by the camera or due to weather conditions or the brightness contrast between objects is too high or the road signs are partially obstructed, the tool often misdiagnoses or it still recognize correctly the traffic signs but accompanied by the incorrect recognition of other positions in the image.

Chapter 5. CONCLUSION AND FUTURE WORKS

5.1. Conclusion

In this thesis, we study using the Convolutional Neural Network to recognize the traffic sign image. We also study the image processing technique to enhance the image quality and get the essential information in it. We have built a tool that can recognize the traffic sign from an image to support the driver. The tool can identify the traffic sign in the image, frame it with a red rectangle and export its name. This may help the driver avoid accidents related to traffic signs when they are out of sight or do not understand that sign. The tool just works well with clearly input image, the contrast between the objects is not too high, favorable weather conditions and no obstructions.

5.2. Future Works

Currently, the tool can only recognize the traffic sign from image and the accuracy is not really high. We need to improve the accuracy as well as develop the ability of recognizing from video. Beside, we also want to develop the ability of integrating with embedded system in the car and give the driver recommendations at that point of time.

REFERENCES

[1] Rafael C. Gonzalez, Richard E. Woods (2006). Digital Image Processing 3rd, NJ, USA.

[2] Russell, Stuart J., et al (2003). Artificial Intelligence - Modern Approach 2nd, Upper Saddle River, New Jersey: Prentice Hall.

[3] J. Jiang, P. Trundle, J. Ren (2010). Medical image analysis with artificial neural networks, Digital Media & Systems Research Institute, University of Bradford, United Kingdom.

[4] LeCun Y., Pierre Sermanet (2011). Traffic Sign Recognition with Multi-Scale Convolutional Networks, Proceedings of International Joint Conference on Neural Networks.

[5] Joe Minichino, Joseph Howse (2015). Learning OpenCV 3 Computer Vision with Python 2nd Edition, Packt Publishing Ltd, UK.

[6] Xavier Glorot, Antoine Bordes, Y. Bengio (2011). Deep Sparse Rectifier Neural Networks, Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, PMLR.

[7] Diederik P. Kingma, Jimmy Lei Ba (2015). Adam - A method for Stochastic Optimization, a conference paper at ICLR.

[8] LeCun Y., Bengio Y., et al (1998). Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE.

[9] Zsolf T. Kardkovacs, Adam Siegler, et al (2011). Real-time traffic sign recognition system, IEEE Xplore.

[10] Mrinal Haloi (2015), Traffic Sign Classification Using Deep Inception Based Convolutional Networks, arXiv.