CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

Autonomous Traffic Sign Detection and Recognition using Deep Learning

A thesis submitted in partial fulfillment of the requirements

For the degree of Master of Science

In Computer Science

By

Rudri Oza

May 2021

The thesis of Rudri Oza is approved:

Dr. Robert McIlhenny	Date
Dr. Katya Mkrtychan	Date
Dr. Taehyung Wang, Chair	Date

California State University, Northridge

Acknowledgments

Above all, I want to express my gratitude to God, the Almighty, for blessing me with his blessings in order to complete this study. For me, writing this thesis has been an incredible experience. I'd like to express my heartfelt gratitude to everyone who has helped me tremendously during this study.

First and foremost, I'd like to express my heartfelt gratitude to my advisor, Dr. Taehyung Wang, Professor, California State University – Northridge's College of Engineering and Computer Science. In terms of research and graduate studies, Dr. Wang exuded a sense of adventure. This work would not have been possible without his encouragement and imaginative ideas. Dr. Robert McIlhenny and Dr. Katya Mkrtychan, members of my committee, have been excellent mentors and have encouraged me and shown trust in me throughout the journey.

A massive thank you to my family for all of the occasions you have stepped in to assist me throughout my academic career. When things got tough, it was because of your support that I was able to get through the difficulties, pain, and struggle. I am grateful to my father, Mr. Mahesh Oza, for his unwavering love, encouragement, and guidance throughout my life. I have no words to express my gratitude to my mother, Mrs. Asha Oza, who has always been a supportive and sympathetic ear to me. I am able to present this research as a result of both of their perseverance and nurturing. Your prompt inspirations, timely feedback, compassion, excitement, and dynamism have aided me throughout the process and brought me to this point of success. I would also like to take this opportunity to thank my uncle and aunt Dr. Jagdish Patel and Mrs. Saroj Patel for all your moral support and unconditional love. It is the result of your hearty blessings that I am able to submit this thesis today

Table of Contents

List of Figures

List of Tables

Abstract

Autonomous Traffic Sign Detection and Recognition using Deep Learning

By

Rudri Oza

Master of Science in Computer Science

The Advanced Driver Assistance System includes traffic sign identification and recognition. Traffic signs warn drivers of traffic laws, road conditions, and route directions, assisting them in driving more efficiently and safely. Traffic Sign Recognition is a technique for regulating traffic signals, warning drivers, and commanding or prohibiting specific acts. A quick real-time and reliable automated traffic sign detection and recognition system can assist and relieve the driver, improving driving safety and comfort significantly. For autonomous intelligent driving vehicles or driver assistance systems, automatic identification of traffic signals is also important. The aim of this study is to use Neural Networks to identify traffic sign patterns. Several image processing methods are used to pre-process the images. Then, to understand traffic sign patterns, Neural Networks stages are performed. To find the best network architecture, the system is trained and validated. The results of the experiments show that traffic sign patterns with complex backgrounds can be classified very accurately.

Chapter 1. Introduction

1.1 Problem Statement

Traffic signs are intended to safely control traffic flow by informing both drivers and pedestrians [1]. By providing detailed road and traffic statistics, traffic signs provide essential street instruction. Along with this technology has advanced at a breakneck pace, and it has become an integral part of every aspect of human life. The invention of self-driving cars is one of the most well-known examples of how technology has influenced transportation. To support a vehicle's operator, it incorporates both mechatronics and artificial intelligence. Today, autonomous driving functions for motorized road vehicles are a hot topic of study. Given the unpredictability of the open road, autonomous driving is now a possibility. Innovationists insist that conscience cars are now a "resilient problem," which Musk claims to be "almost ... a solved problem"[4].

Despite this, it still has trouble achieving accuracy and speed when detecting signs on the open road. As a result, identifying traffic signals will assist people in better understanding their surroundings when driving and walking down these streets. By providing rich road and traffic knowledge, traffic signs provide fundamental street instruction [2]. As a result, identifying traffic signals will assist people in better comprehending their surroundings when driving or walking down these streets. As a result, traffic sign identification and management are needed, if not critical, in order to improve traffic safety and efficiency [3]. In recent years, researchers in the Intelligent Transportation System (ITS) have looked into traffic sign identification. Recognizing these difficulties, I suggest testing a new paradigm that combines many techniques such as preprocessing, visualization, and the implementation of various combinations of Machine Learning Algorithms in an effort to improve accuracy.

1.2 Significance of Study

A modern race has begun with the new millennium, Of and so far it little was heard. A competition that has the potential to take over all of the world's drivers without warning. This means that the steering wheel would be controlled by an electronic "head" rather than another human. By the middle of the 20th century, the way people move should have changed substantially. By the middle of the 20th century, the way people move should have changed substantially.[5] The driver's function will be limited to issuing destination orders because cars will drive themselves from point A to point B.

Driving, It's also boring and boring, but it can be replaced by fun activities such as communiqué with acquaintances or intimate, Internet evaluation/watching broadcast, viewing a flick, etc. (Preparedness of a demonstration, teleconference or other business-related things).

In order for the charioteer to be interchanged, the microelectronic devices must control commands like acceleration, braking and direction. In addition, a driver must maintain your yeahs on the way and pay distributed devotion to insignias, as well as use intelligent cameras in traffic radar situations. More sensors are being used by technology, and information is being processed by machines.

One of the most significant components of self-driving cars and forward-looking driver succor systems is the classification of traffic signals [8], [5], [9], [4], [10]. Owing to various obstacles and a lack of attention, drivers often ignore traffic signals. The classification of traffic signs may be automated, which would help to reduce accidents. Traditional computer vision and machine learning-based methods for traffic sign classification were commonly used [8], [3], but deep learning-based classifiers quickly replaced them. Deep convolutional networks have recently exceeded traditional learning approaches in the classification of traffic signals. With the rapid advancements in deep learning algorithm structures and the possibility of high-performance implementation with graphical processing units (GPU), it is beneficial to rethink traffic sign classification problems from a deep learning perspective. Classification of traffic signs is a difficult task because photographs are subjected to adverse variations due to lighting, orientation, and vehicle speed variations, among other factors. ADAS usually uses a wide-angle camera mounted on the top of a vehicle to capture road signals and other visual features.

1.3 Target:

The method to create a system that is able to detect and identify real-time traffic signs requires a hybridization of many different methods of deep learning and image processing. The main question here was to how and which methods to implement to get the best results. The basic structure of the system will be as follows:

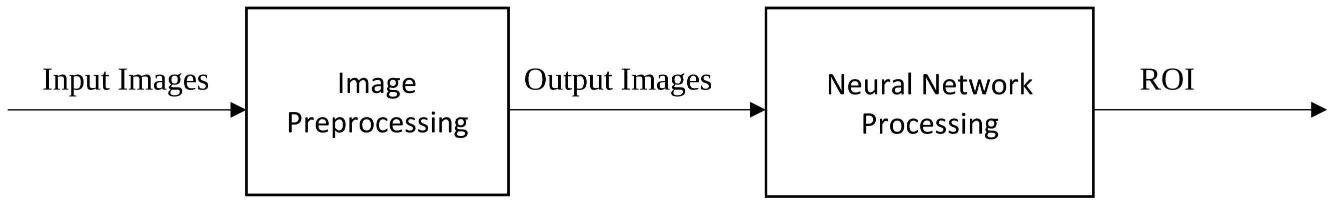


Figure 1: Structure of System

After identifying these methods, the main targets this thesis hopes to fulfill are that the system shall give better accuracy in order to identify traffic signs in real-time. The other goal fulfilled is the region of interest (ROI) to be identified and the signal to be verified following a wide search in an image for candidates.

1.4 Planning:

Course	**Period**	**Activity**	**Duration**
COMP 696 C	September To October	Identifying challenges and problems of the Autonomous driving car	0.25 Month
		Conducting a existing literature and technology review	0.25 Month
		Produce an objective and approaches	0.5 Month
		Write a proposal	0.5 Month
		Complete system requirements and specification	0.5 Month
		Get the database and understand and clean it	0.5 Month
		Design and implement a CNN System	0.5 Month
COMP 698 C	January To May	Evaluate and optimize and test the created system	0.5 Month
		Write a review paper	0.5 Month
		Write the thesis	2 Month
		Prepare the defense presentation	0.5 Month

Table 1: Project Plan

During the life cycle of the project, some tasks were executed prior to others. All the tasks processed for the COMP 696 were completed successfully. The report for the same was submitted to committee chair as well as committee members. After the final model, evaluation techniques were performed on the model. Finally, the thesis was formulated, and the defense preparation was undertaken.

Chapter 2. Literature Review

In late 2016, Tesla revealed that the next Model S generation would have "full hardware for itself." [1]. Bestowing to Elon Musk, the company's CEO, this will be a expertise efficient of implementing a long-held dream of motorized automation, with the required sensors and processing power to drive "all the way from LA to New York" without human intervention by the end of 2017[11]. This quintessential "smart" technology, on the other hand, was not born smart. The brain of this self-driving-car-in-development is still developing. This eponymous "smart" technology, on the other hand, was not born smart. This self-driving car's brain hasn't completely developed yet.

For contrast, Sermanet and LeCun [3] achieved a 95.17 percent accuracy using a much high capability convolutional neural network that is almost 100 times faster than the one presented in this study. Of course, this makes it difficult and expensive to incorporate these computer-complex algorithms into graphic implementations on-board [6]. This is particularly true if the device has already been installed and the "vendor" wants to add new features to existing platforms.

My potential alternative is different from today's state of the art frameworks in that it employs a modular, platform-independent approach. The proposed system of vision is based on a fully trainable CNN that can identify any form of traffic signals, including circular, rectangular or triangular signals [9]. CNN was programmed to detect 43 travel signals, but this number can be easily increased using a learning algorithm (continue with the addition of new traffic sign training data and use the current network after the output layer has been modified). The raw pixel values are the CNN's inputs, and the direct trust values reflecting the probability of a particular traffic sign are the CNN's outputs[13].

2.1 Image Processing

Image processing is a technique for applying image systems to enhance or extract significant image information. It is a form of modulation processing in which the image is created and the output is an image or its features[14]. The processing of images is one of today's fastest-moving technologies. It is also a critical field of research in engineering and informatics.

The three stages that make up image processing are as follows:

- Importing an image using image acquisition software
- Analyzing and manipulating the image and
- Producing an output that may be an altered image or a report based on image analysis.

The two types of image processing methods are analog signal and digital image processing. Analog image processing can also be used to provide hard copies such as printouts and images. Image analysts use a diverse range of analysis fundamentals when using these visual techniques. The computer-assisted handling of digital images is possible by digital photo processing methods [12]. The three overall phases that all types of data must take with digital techniques are pre-processing, improvement, display and the extraction.

Digital image processing is the method of manipulating digital images using a computer. It is a subfield of signals and systems that focuses on images in particular. DIP focuses on the development of a computer device capable of image processing[15]. The device takes a digital image as an input and processes it using powerful algorithms to produce an image as an output.

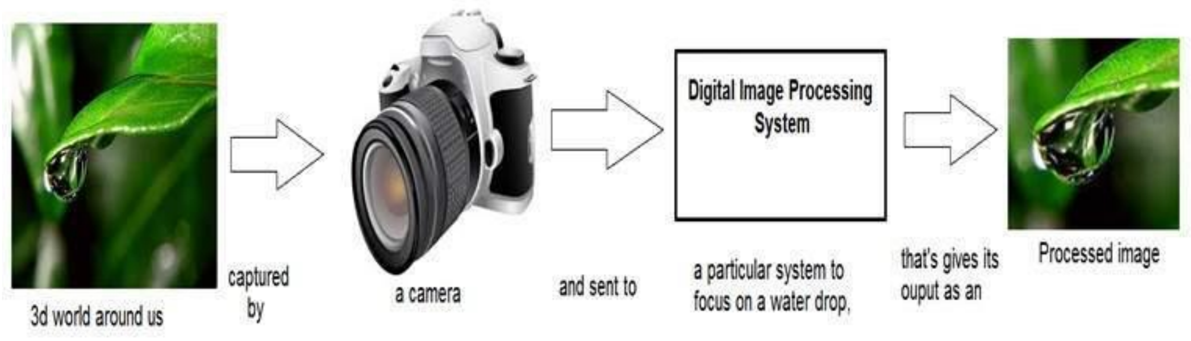

3d world around us
captured by
a camera
and sent to
Digital Image Processing System
a particular system to focus on a water drop,
that's gives its ouput as an
Processed image

Figure 2: How Image Processing Works

In the illustration above, an image was taken by a camera and sent to a digital device to delete all other information and concentrate solely on the water drop by zooming it in such a way that the image quality is maintained.

2.2 Deep Learning

Deep learning is a method of engineering which covers artificial neural networks, which are brain-inspired algorithms [7]. Deep learning is defined by the pioneers and experts in the area, And these complex and nuanced insights shed a great deal of light on deeper learning.

Deep learning has advanced with the contemporary age, leading to an avalanche of data in all kinds of countries and around the globe. Large amounts of information are obtained from various outlets, such as social media and internet ad networks, e-commerce and online cinemas [9]. This huge volume of data, as well as other tools such as cloud computing, is easily available and can be shared.

However, since the data is typically unconstructed, people can understand and draw valuable knowledge for decades. Companies are progressively adopting automated support AI systems, because it recognizes the huge potential that this wealth of data can bring about.

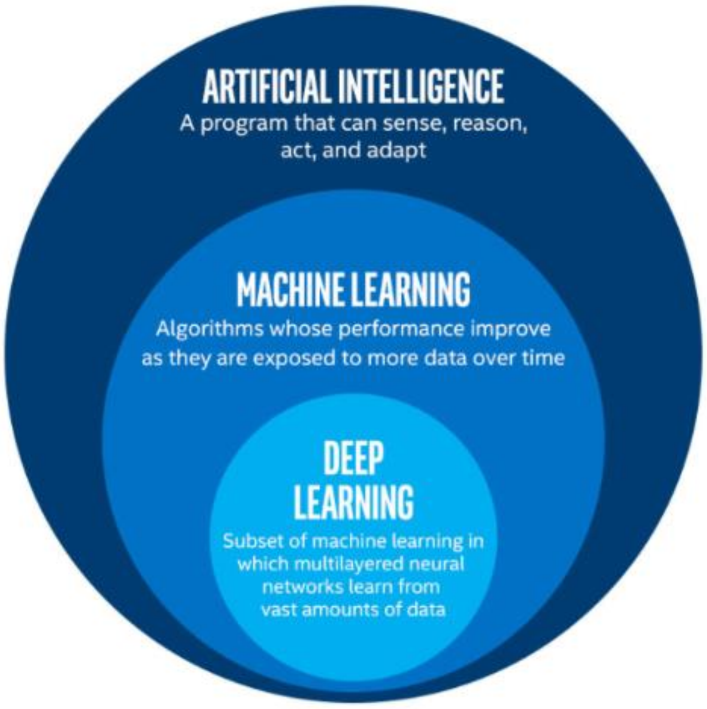

ARTIFICIAL INTELLIGENCE
A program that can sense, reason, act, and adapt
MACHINE LEARNING
Algorithms whose performance improve as they are exposed to more data over time
DEEP LEARNING
Subset of machine learning in which multilayered neural networks learn from vast amounts of data

Figure 3: Uses of Deep Learning vs. Machine Learning vs. Artificial Intelligence

2.3 Neural Networks

We may use neural networks to perform a variety of tasks, such as clustering, sorting, and regression. We can use neural networks to assemblage or character unlabeled data based on similarities between the samples [15] . Alternatively, We can train the network on the named data source for designation in order to categorize training datasets. Word embeddings can solve tasks that machine models cannot resolve since particularly unique capabilities are provided to artificial neural networks[13]. We might also argue that synthetic neuronic linkages and deep learning are bringing about a new industrial revolution today.

A neural network is an algorithm system which tries to identify the associations that are underlying a data set using a method to imitate the workings of the human brain. In this context, neural networks refer to neuronal structures that can be organic or artificial. As neural networks can adapt to change inputs, they can achieve the best result without having to redesign the power output [15]. Neural networks are rapidly gaining traction in the development of trade systems with artificially intelligent ideas.

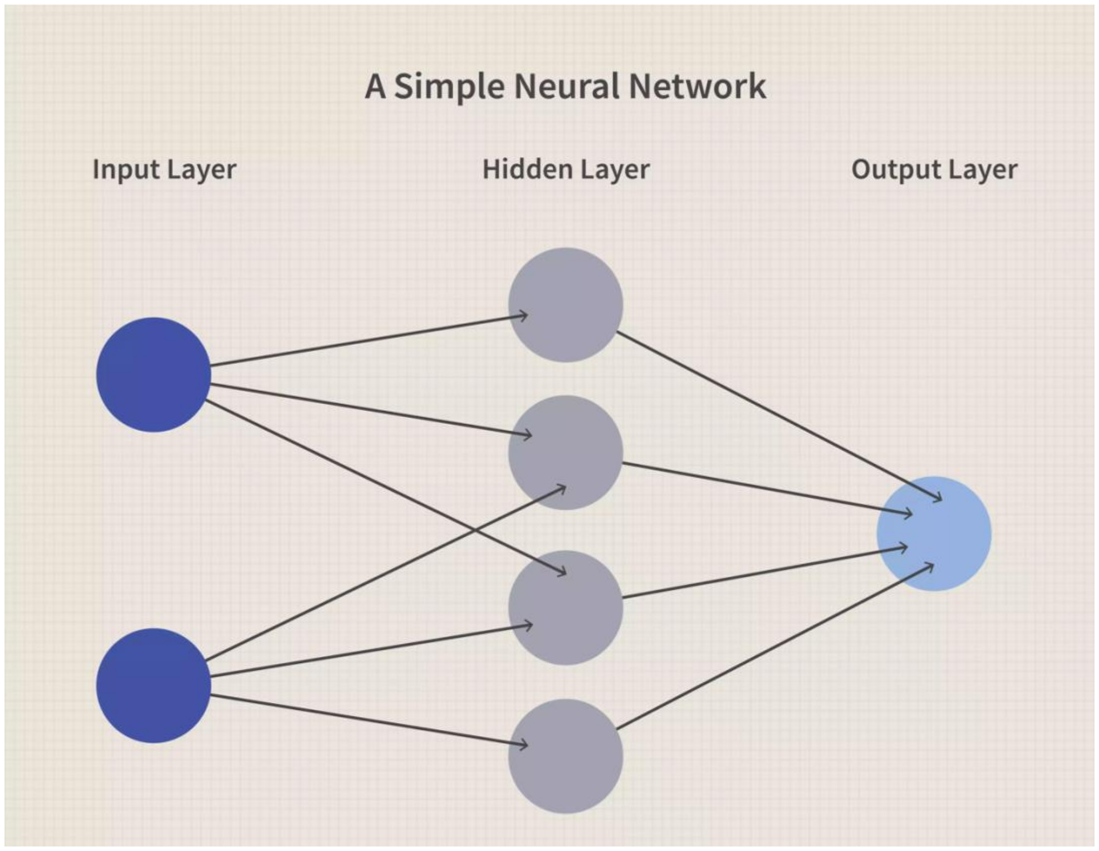


Figure 4: Structure of a Neural Network

A neural network is the same as the human brain's genetic algorithm. In a neural network, a "neuron" it is a mathematical process that uses a particular architecture to collect and categorize data [12]. Two approaches to mathematical adaptation and regression analysis closely resemble the network. Two approaches to mathematical adaptation and regression analysis closely resemble the network.

The genetic algorithm consists of layers of weighted connections. Each node is a sensor that functions as a multiple regression analysis [15]. The perceptron turns the signal into a non-linear component from a multiple linear regression.

Perceptron’s are arranged in convolutional nodes in a multi-coat perception (MLP) each input layer collects input patterns. The output layer includes classifications or output signals that can link input sequence to. For example, patterns might include a list of quantities for safety technical indicators; possible outputs may include "buy," "keep," or "sell."

The input weightings are fine-tuned by hidden layers until the neural network's margin of error is as small as possible. Hidden layers are supposed to extrapolate important characteristics from the input information that predict outputs. This is how the extraction of features works and how statistical methods such as the main analytical component work.

Chapter 3. Methodology

3.1 Dataset and Setup

There are 31,367 training images, 7,842 validation images, and 12,630 testing images in the GTSRB dataset. There are a total of 43 classes in which the photographs are distributed. In terms of similarity, the groups are divided into a category of six subclasses. All of the photographs were taken from a series of videos shot at various times of the day and in various lighting conditions. The footage was also shot from a moving vehicle, which eliminated motion blur.

The data set is divided into sets of training, verification and debugging with the following features:

- The dimensions of the Images are 32 pixels (bidth) x 32 pixels (height) (RGB color channels)
- There are 34799 pictures in the training collection.
- There are 4410 images in the validation package.
- There are 12630 pictures in the test range.
- There are 43 different classes (e.g. speed limit 20km/h, no input, bumpy lane etc.).

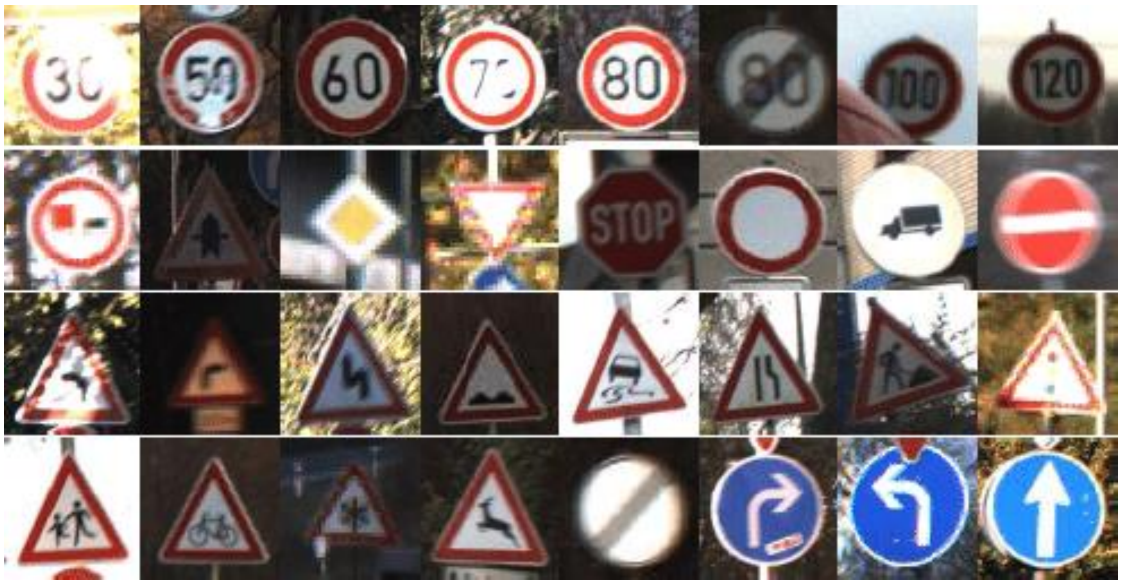


Figure 5: Images from the Dataset

NumPy, Pandas, OpenCV, and Matplotlib are the major tools used in this project. Sorting the data comes first. To get the distribution, NumPy's Unique function is used. After that, the Pandas Data Frame reads the CSB files with Class ID. Matplotlib is used to plot the data frame as a histogram after it has been expanded with more columns. Following the analysis of the dataset, data augmentation, model design, preparation, and testing begin.

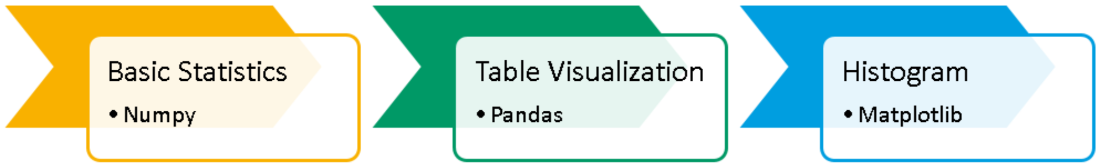


Figure 6: Stages of the dataset

3.2 Pre-Processing

A data set consists of data items known as records, points, vectors, patterns, events, instances, examinations, observations or entities. The basic features of an object, like the mass of an object or the time of an object occurrence happened, are captured by a variety of features that identify data objects. Variables, characteristics, fields, attributes, and measurements are all terms used to describe features.

Color, mileage, and strength, for example, can all be called car features. When working with data, we can come across a variety of different features.

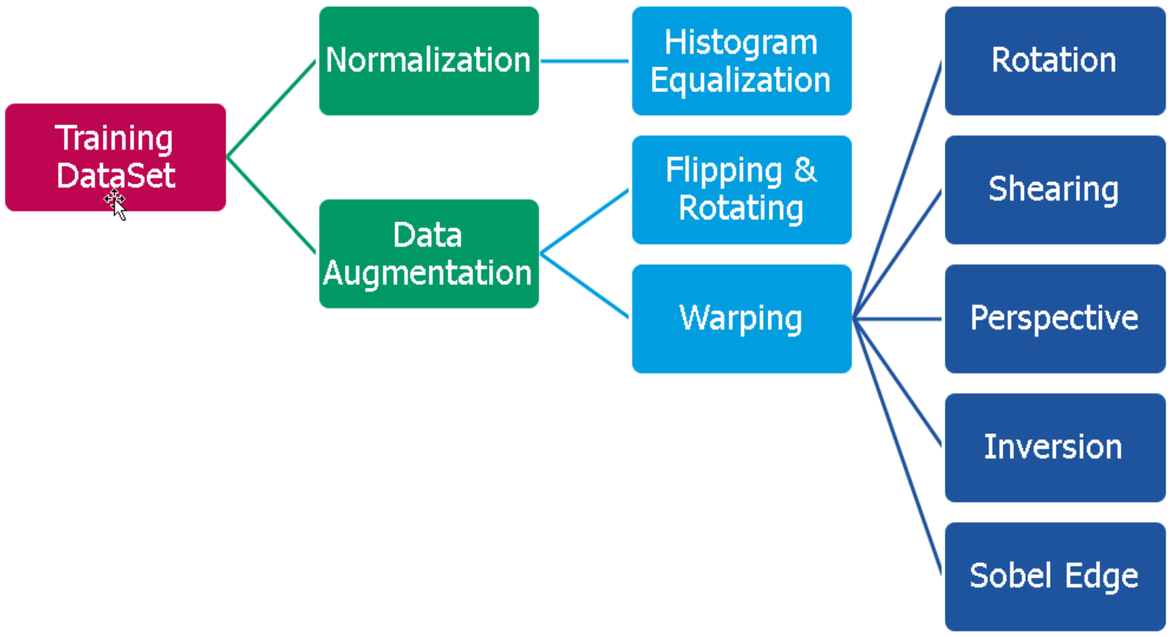


Figure 7: Data Preprocessing techniques

Data pre-processing is a method of data collection that requires raw data to be converted into an understandable format. Statistical data are often inaccurate, unreliable, and/or deficient in specific habits or patterns, as well as containing numerous errors. Preprocessing data is a tried-and-true way of addressing such problems.

In the real world, data is often incomplete: it lacks attribute values, some attributes of interest are missing, or it only contains aggregate data. Errors or outliers make the data noisy. Inconsistent: having inconsistencies in codes or names.

Techniques for pre-processing include:

- focusing on the global mean
- Using the mean to focus the picture locally
- Using Standard Deviation to Normalize Equalization of histograms Training data must be scaled and normalized, as well as extended, supplemented, and balanced.
- Scaling and normalizing the validation and test data is appropriate.

3.3 Augment Data

The volume and diversity of data collected in recent years has been primarily due to recent developments in deep learning models. Without actually gathering new data, data augmentation allows clinicians to greatly expand the diversity of data available for training models. Cropping, padding, and horizontal flipping are popular data augmentation techniques used to train large neural networks.

However, most neural network training methods only employ basic forms of augmentation. Although neural network architectures have received a lot of attention, finding strong types of policies for increase and increase in data which have received less attention when capturing data invariances.

Since certain classes had more images than others and vice versa, I had to perform data augmentation on the dataset, such as zoom, rotation, shear, brightness disruption, gaussian noise, color inversion, and so on. Before the augmentation, the data visualization looked like this:

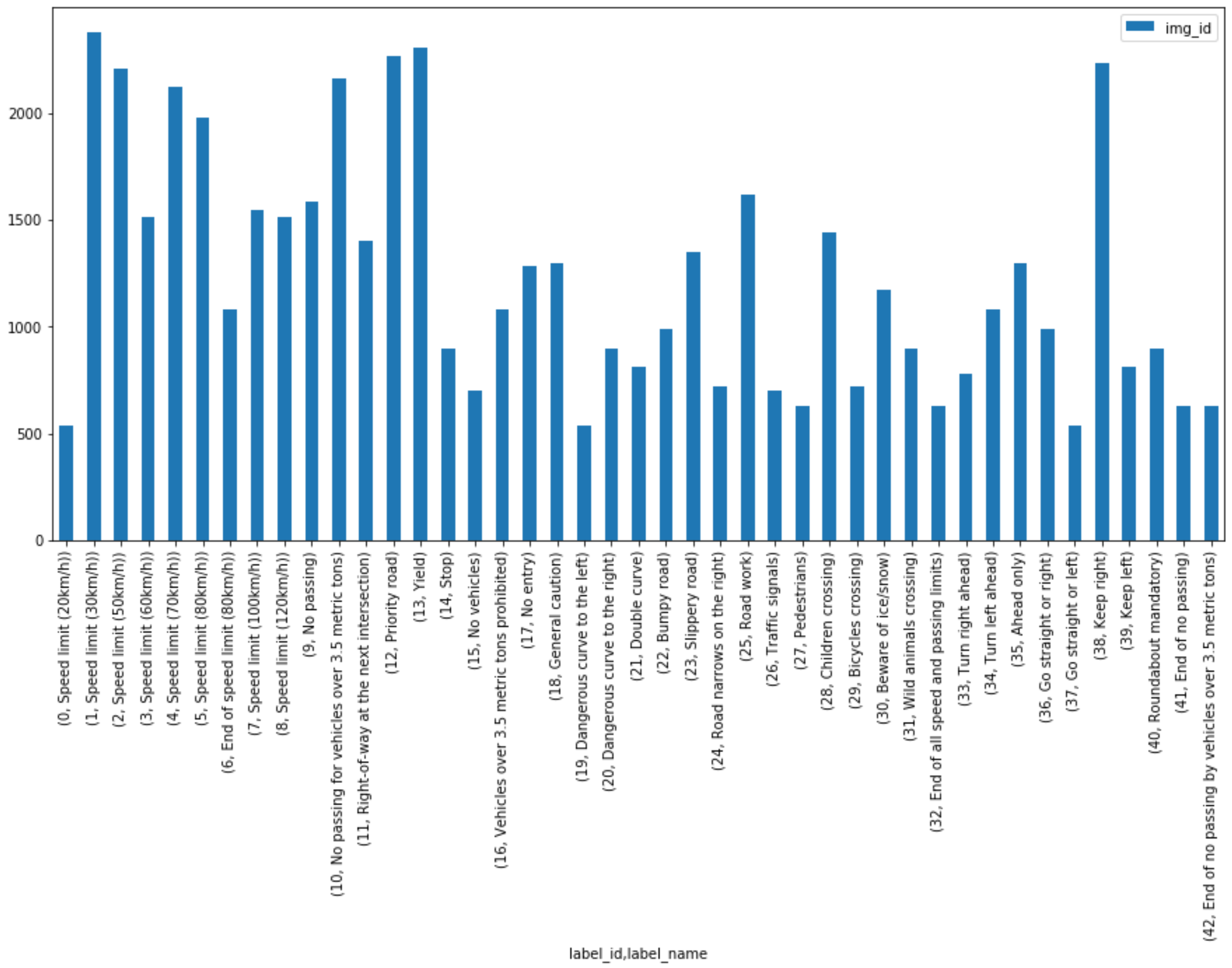


Figure 8: Data visualization of the dataset before augmentation

1) Zoom: With the zoom Augmentation, you can zoom in or out on the image. A single float value or a list of two values may be passed to the Image Data Generator class:

The zoom range is [1-value, 1+value] if only a single value is given. If a list is given, one value will be used as the lower limit and the other will be used as the upper limit.

Within the given range, the picture is zoomed in or out at random.

2) Rotation: By applying an angle to the image, we can rotate it. Each rotated image is a one-of-a-kind representation of the model. Depending on the object in the picture, you can rotate it up to 360 degrees. We're using rotation range = 50 in the example above, which means the Image Generator treats it as a range [-50,50] and applies a random angle from the range to the image.

3) Shear: Another bounding box transformation that can be performed with the aid of the transformation matrix is shearing. Shearing is often used to adjust the image's orientation.

4) Brightness: Brightness is added to images in order to improve their quality and yield better results.

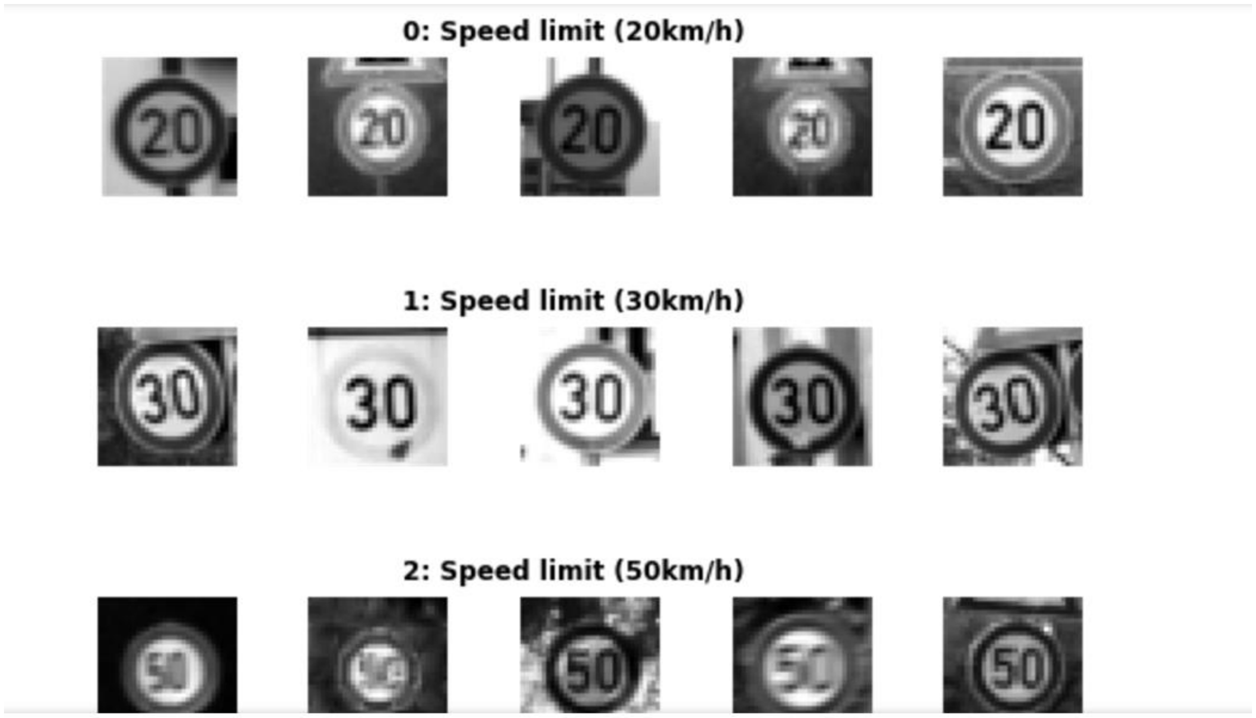


Figure 9: Results of Data Augmentation-1

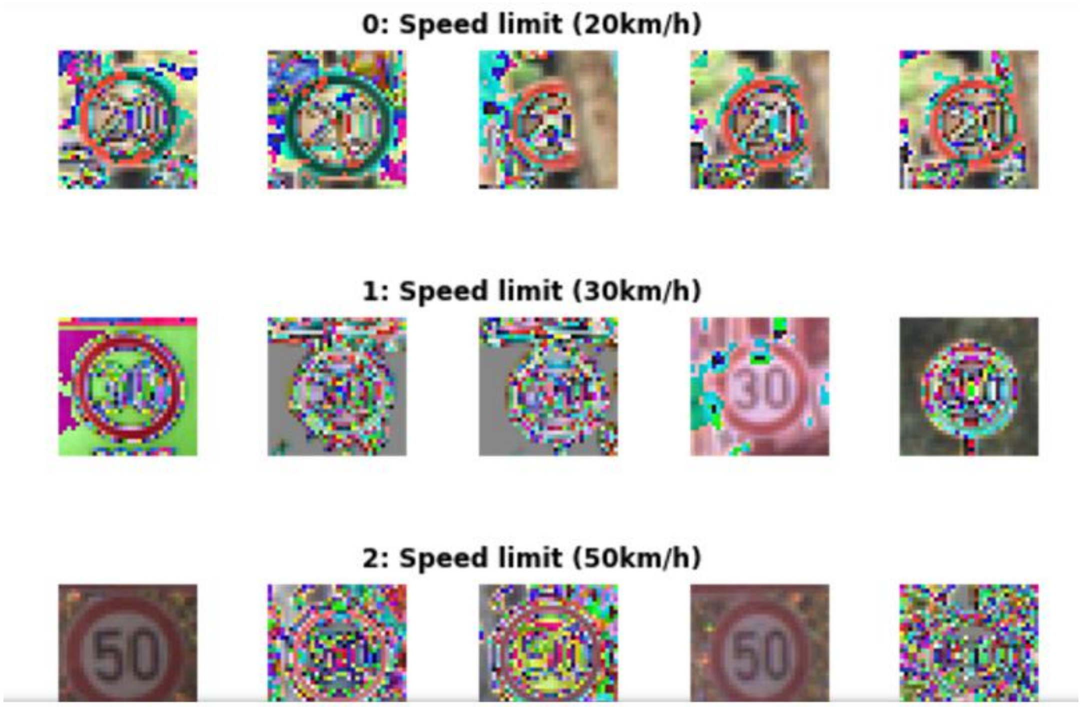


Figure 10: Results of Data Augmentation-2

3.4 Model Architecture

Yann Le Cun's paper on traffic sign classification influenced the proposed architecture. I made a scarce twists and built a prefabricated codebase that allows us to experiment with distinctive filter sizes, depth, and number of convolution layers, as well as completely connected layer dimensions. For the first convolutional layer, I mostly tried 5x5 and 3x3 filter (aka kernel) sizes, starting with a depth of 32. The network comprises three convergent layers, each of which has a kernel size of 3x3 and a depth at the next node. Each operation is followed by a 2x2 max pooling operation, with ReLU as the activation function. The final three layers are completely connected, with the final layer yielding 43 results (the total number of potential labels) when the SoftMax activation function is used. The network is trained using the Adam optimizer and mini-batch stochastic gradient descent.

Most classification deep learning nets have a model architecture that is very similar to this one. In order to achieve scale invariance, Spatial Transformers have been used. For the classification mission, we used a CNN for feature extraction and a linear Classifier with SoftMax activation:

Localization Modules -> Spatial Transformer Module -> CNN ->Linear Classifier VGG1 -> VGG2 -> VGG3 -> VGG4 -> CONCAT - VGG1_VGG2_VGG3_VGG4 -> FC1 -> FC2 -> Logits

The neural network used in this case is the VGG neural network. The Visual Geometric Group of Oxford University is an acronym for VGG-16, a network of 16 layers suggested by the Visual

Geometric Group. There are other layers like the Max pool layer with no trainable parameters but the trained parameters are contained in those 16 layers.

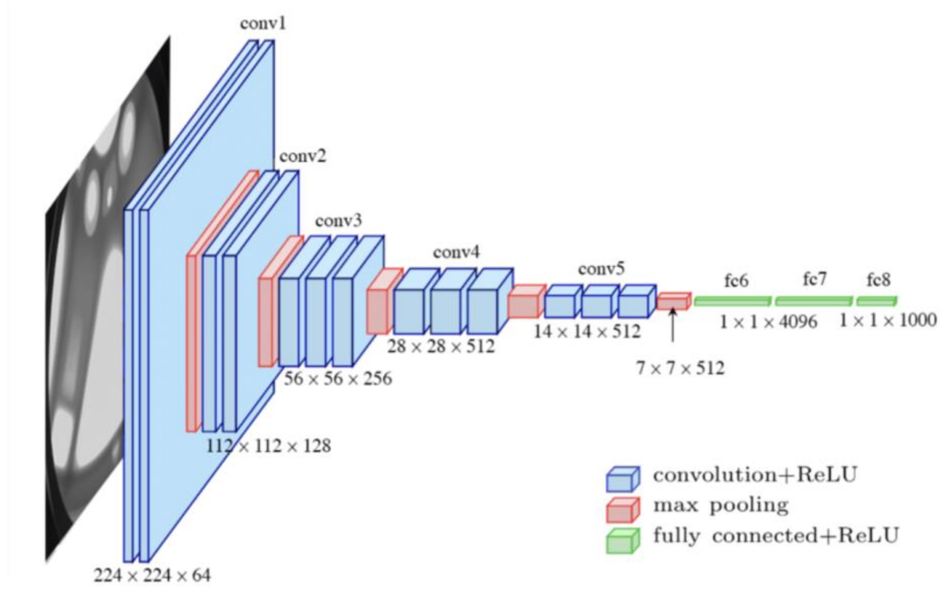
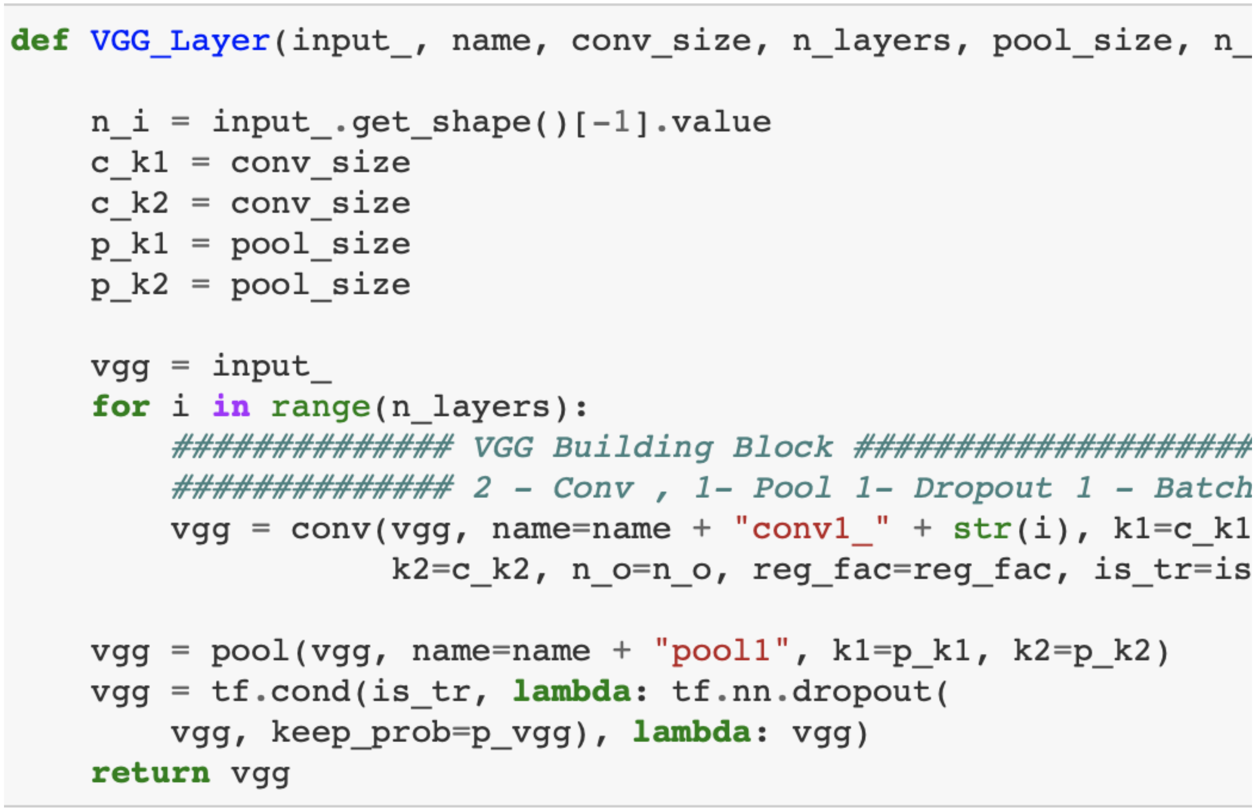


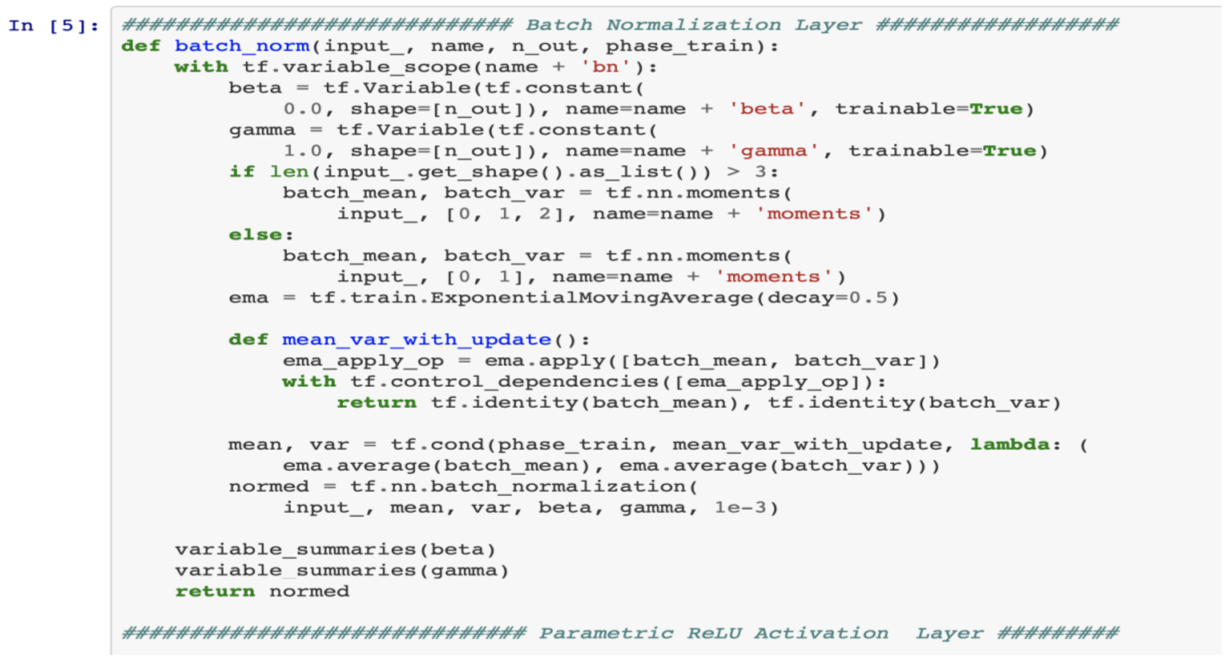
Figure 11: VGG Network Structure



```
def VGG_Layer(input_, name, conv_size, n_layers, pool_size, n_

    n_i = input_.get_shape()[-1].value
    c_k1 = conv_size
    c_k2 = conv_size
    p_k1 = pool_size
    p_k2 = pool_size

    vgg = input_
    for i in range(n_layers):
        ############### VGG Building Block ###################
        ############### 2 - Conv , 1- Pool 1- Dropout 1 - Batch
        vgg = conv(vgg, name=name + "conv1_" + str(i), k1=c_k1
                   k2=c_k2, n_o=n_o, reg_fac=reg_fac, is_tr=is

    vgg = pool(vgg, name=name + "pool1", k1=p_k1, k2=p_k2)
    vgg = tf.cond(is_tr, lambda: tf.nn.dropout(
        vgg, keep_prob=p_vgg), lambda: vgg)
    return vgg
```

Figure 12: VGG Layer

The Batch Normalization Layer is the next layer in the neuronic linkage. Batch Normalization Layer is a tactic that provides a consistent the inputs of each mini batch to a layer. This stabilizes the learning process and significantly reduces the number of training epochs required to train deep networks. Batch normalization, also known as batch standard, is a technique for coordinating the update of multiple layers in a model. It can also function as a regularizes, reducing generalization error in the same way that activation regularization does.

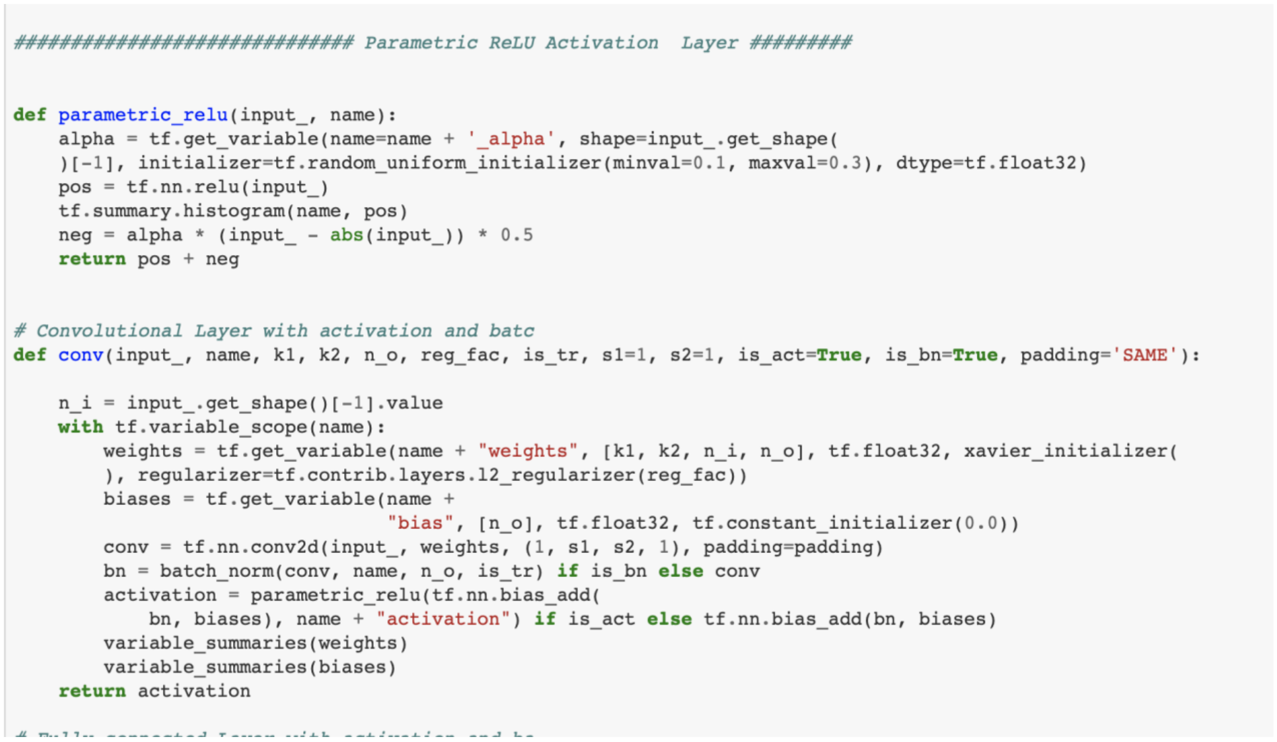


```
In [5]: ############################## Batch Normalization Layer ##################
        def batch_norm(input_, name, n_out, phase_train):
            with tf.variable_scope(name + 'bn'):
                beta = tf.Variable(tf.constant(
                    0.0, shape=[n_out]), name=name + 'beta', trainable=True)
                gamma = tf.Variable(tf.constant(
                    1.0, shape=[n_out]), name=name + 'gamma', trainable=True)
                if len(input_.get_shape().as_list()) > 3:
                    batch_mean, batch_var = tf.nn.moments(
                        input_, [0, 1, 2], name=name + 'moments')
                else:
                    batch_mean, batch_var = tf.nn.moments(
                        input_, [0, 1], name=name + 'moments')
                ema = tf.train.ExponentialMovingAverage(decay=0.5)

                def mean_var_with_update():
                    ema_apply_op = ema.apply([batch_mean, batch_var])
                    with tf.control_dependencies([ema_apply_op]):
                        return tf.identity(batch_mean), tf.identity(batch_var)

                mean, var = tf.cond(phase_train, mean_var_with_update, lambda: (
                    ema.average(batch_mean), ema.average(batch_var)))
                normed = tf.nn.batch_normalization(
                    input_, mean, var, beta, gamma, 1e-3)

            variable_summaries(beta)
            variable_summaries(gamma)
            return normed

        ############################## Parametric ReLU Activation  Layer #########
```

Figure 13: VGG Layer

Following the efficient completion of the batch normalization layer, it is necessary to enable the newly transformed data in order to push it into the next layer and begin the model. The ReLu Activation Layer is the next layer. The corrected linear active function, or short ReLU, is a piece by frame linear function, which directly produces an output the input if the input is positive; anything other than that, the input is zero. Since a model is easier to train and consequently better results have been obtained, many forms of neural networks have become standard activator. There are many advantages of using this as an activation layer:

- Computational Simplicity
- Representational Sparsity
- Linear Behavior
- Train Deep Networks

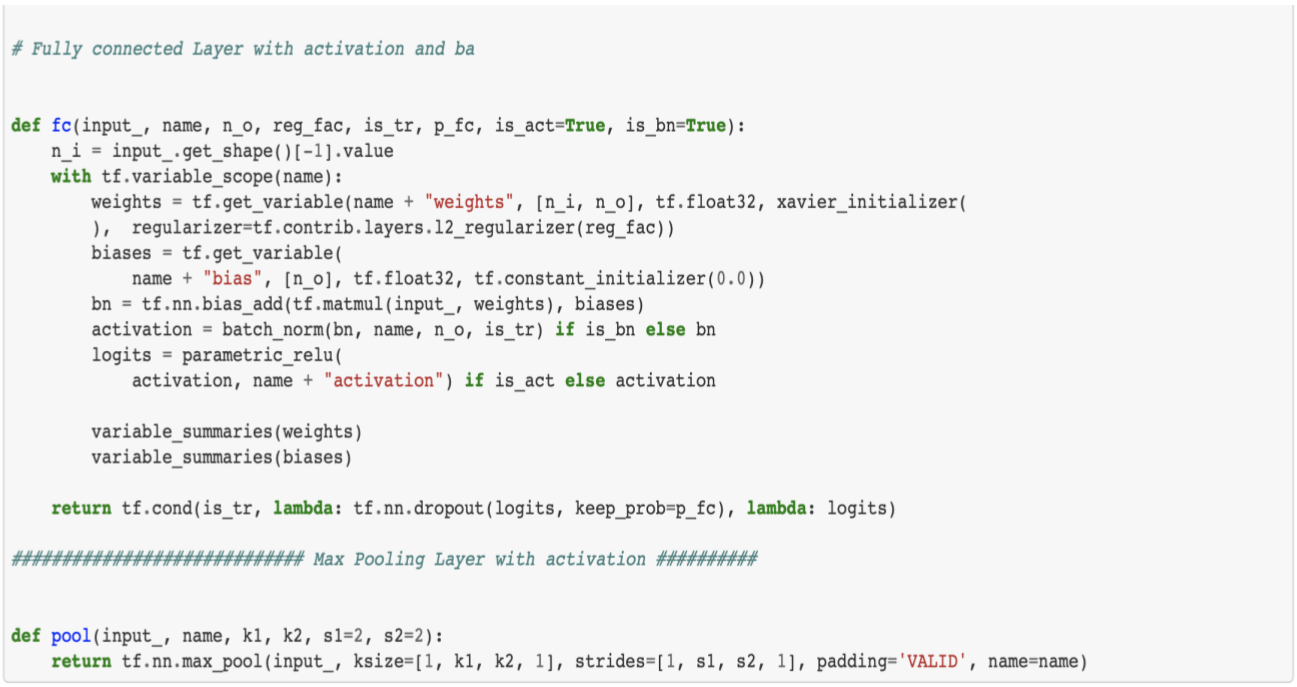


```
############################# Parametric ReLU Activation  Layer #########


def parametric_relu(input_, name):
    alpha = tf.get_variable(name=name + '_alpha', shape=input_.get_shape(
    )[-1], initializer=tf.random_uniform_initializer(minval=0.1, maxval=0.3), dtype=tf.float32)
    pos = tf.nn.relu(input_)
    tf.summary.histogram(name, pos)
    neg = alpha * (input_ - abs(input_)) * 0.5
    return pos + neg


# Convolutional Layer with activation and batc
def conv(input_, name, k1, k2, n_o, reg_fac, is_tr, s1=1, s2=1, is_act=True, is_bn=True, padding='SAME'):

    n_i = input_.get_shape()[-1].value
    with tf.variable_scope(name):
        weights = tf.get_variable(name + "weights", [k1, k2, n_i, n_o], tf.float32, xavier_initializer(
        ), regularizer=tf.contrib.layers.l2_regularizer(reg_fac))
        biases = tf.get_variable(name +
                                 "bias", [n_o], tf.float32, tf.constant_initializer(0.0))
        conv = tf.nn.conv2d(input_, weights, (1, s1, s2, 1), padding=padding)
        bn = batch_norm(conv, name, n_o, is_tr) if is_bn else conv
        activation = parametric_relu(tf.nn.bias_add(
            bn, biases), name + "activation") if is_act else tf.nn.bias_add(bn, biases)
        variable_summaries(weights)
        variable_summaries(biases)
    return activation
```

Figure 14: ReLu Activation Layer

The next step after activating the dataset is to link it to the next layer, which necessitates the addition of a new layer to the architecture called the convolutional layer. At the heart of the convolutionary neural network are the convolutionary layer that gives its name to the network. This layer conducts a process known as "convolution." A convolution, like a conventional neural network, is a linear operation that involves multiplying a collection of weights with the input. The replication takes place in this technique dsesigned for two-dimensional input between the data array provided and the two-dimensional weight array referred to as the filter or kernel. Each filter is convolved with the input volume to produce a neuron-based activation map. A convolution is the basic process of applying a filter to an input to produce an activation.

The completely linked layer follows the convolution layer. It takes the previous layers' output and "flattens" it into a single vector that can be used as an input for the next level. The first completely connected layer applies weights to the inputs from the function analysis to predict the correct mark. The final probabilities for each mark are provided by the fully connected output layer. The purpose of a fully - connected layers is to classify images into a label based on the results of the process.



```python
# Fully connected Layer with activation and ba


def fc(input_, name, n_o, reg_fac, is_tr, p_fc, is_act=True, is_bn=True):
    n_i = input_.get_shape()[-1].value
    with tf.variable_scope(name):
        weights = tf.get_variable(name + "weights", [n_i, n_o], tf.float32, xavier_initializer(
        ),  regularizer=tf.contrib.layers.l2_regularizer(reg_fac))
        biases = tf.get_variable(
            name + "bias", [n_o], tf.float32, tf.constant_initializer(0.0))
        bn = tf.nn.bias_add(tf.matmul(input_, weights), biases)
        activation = batch_norm(bn, name, n_o, is_tr) if is_bn else bn
        logits = parametric_relu(
            activation, name + "activation") if is_act else activation

        variable_summaries(weights)
        variable_summaries(biases)

    return tf.cond(is_tr, lambda: tf.nn.dropout(logits, keep_prob=p_fc), lambda: logits)

############################ Max Pooling Layer with activation #########


def pool(input_, name, k1, k2, s1=2, s2=2):
    return tf.nn.max_pool(input_, ksize=[1, k1, k2, 1], strides=[1, s1, s2, 1], padding='VALID', name=name)
```

Figure 15: Fully Connected Layer

The pooling layer comes next. Pooling layers provide a way to draw down sampling characteristic maps by summarizing the characteristics present in feature map patches. Average bundling and max bundling are two common bundling methods which synonymize the average presence of a feature with its most activated presence. We use a max pooling system in this case. The results are shown or grouped functional maps that emphasize the most current characteristic of the patch rather than its average presence with the average grouping. This works better in practice than average pooling for vision tasks such as image classification.

Hyperparameters: These can be adjusted to ensure an optimum performance in the model. In this case, we apply parameters for each layer of the VGG network created and have an optimum network.

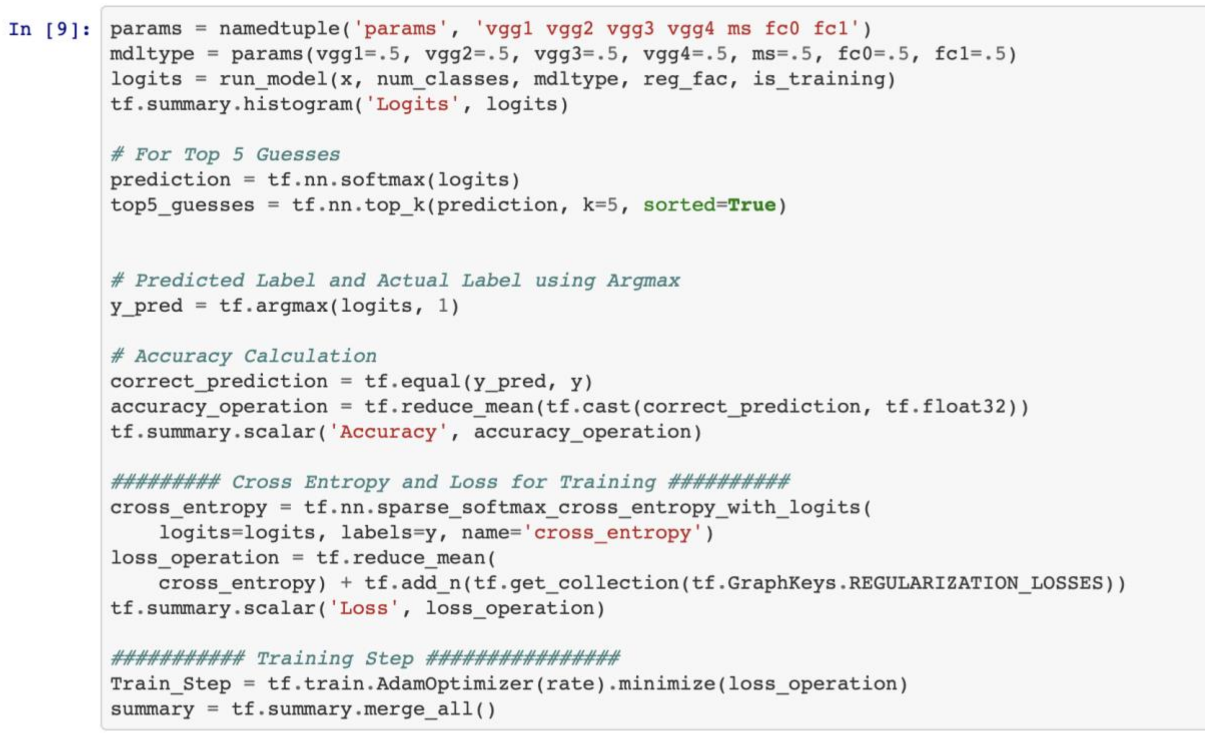
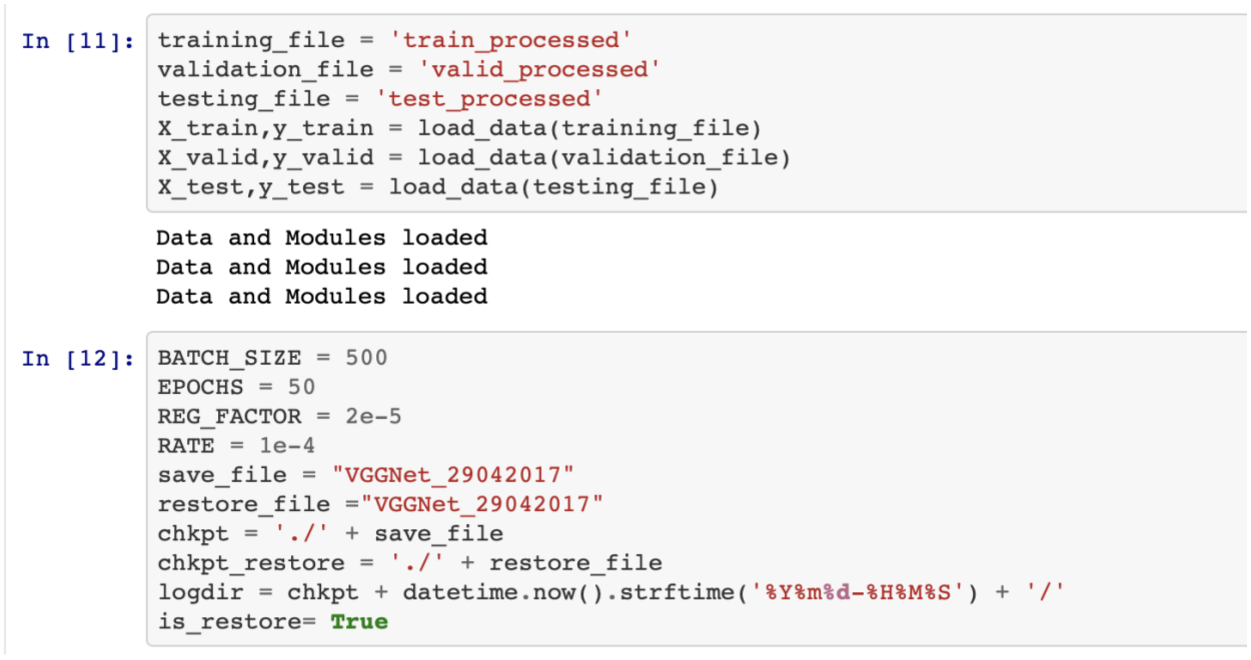

```
In [9]: params = namedtuple('params', 'vgg1 vgg2 vgg3 vgg4 ms fc0 fc1')
        mdltype = params(vgg1=.5, vgg2=.5, vgg3=.5, vgg4=.5, ms=.5, fc0=.5, fc1=.5)
        logits = run_model(x, num_classes, mdltype, reg_fac, is_training)
        tf.summary.histogram('Logits', logits)

        # For Top 5 Guesses
        prediction = tf.nn.softmax(logits)
        top5_guesses = tf.nn.top_k(prediction, k=5, sorted=True)


        # Predicted Label and Actual Label using Argmax
        y_pred = tf.argmax(logits, 1)

        # Accuracy Calculation
        correct_prediction = tf.equal(y_pred, y)
        accuracy_operation = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
        tf.summary.scalar('Accuracy', accuracy_operation)

        ######### Cross Entropy and Loss for Training ##########
        cross_entropy = tf.nn.sparse_softmax_cross_entropy_with_logits(
            logits=logits, labels=y, name='cross_entropy')
        loss_operation = tf.reduce_mean(
            cross_entropy) + tf.add_n(tf.get_collection(tf.GraphKeys.REGULARIZATION_LOSSES))
        tf.summary.scalar('Loss', loss_operation)

        ########### Training Step ################
        Train_Step = tf.train.AdamOptimizer(rate).minimize(loss_operation)
        summary = tf.summary.merge_all()
```

Figure 16: Hyperparameter layer

3.5 Model Training

Training, validation and model testing are next steps. To reassess a kernel function the performance of the model. Low training truthfulness and rationale implies poor performance. High precision in the training set but low precision in the validation set means overcasting. Matrix of Confusion to understand class confusion. Plot a random size confusion matrix to understand confusion in the model among classes.



```
In [11]: training_file = 'train_processed'
         validation_file = 'valid_processed'
         testing_file = 'test_processed'
         X_train,y_train = load_data(training_file)
         X_valid,y_valid = load_data(validation_file)
         X_test,y_test = load_data(testing_file)

         Data and Modules loaded
         Data and Modules loaded
         Data and Modules loaded

In [12]: BATCH_SIZE = 500
         EPOCHS = 50
         REG_FACTOR = 2e-5
         RATE = 1e-4
         save_file = "VGGNet_29042017"
         restore_file ="VGGNet_29042017"
         chkpt = './' + save_file
         chkpt_restore = './' + restore_file
         logdir = chkpt + datetime.now().strftime('%Y%m%d-%H%M%S') + '/'
         is_restore= True
```

Figure 17: Running the model

3.6 Model Testing

A confusion matrix is a summary of a categorization problem prediction results. The performance of a classification algorithm is summarized in a confusion matrix. Calculating a confusion matrix can give you a better understanding of the correctness of your model and the types of errors it makes. For this project, prior and after processing, I have created a confusion matrix for the dataset and the images. The results showed very clearly that the data set did not show the required results prior to image processing. At first, the rates were well below the ROI and the corresponding rates were much lower.

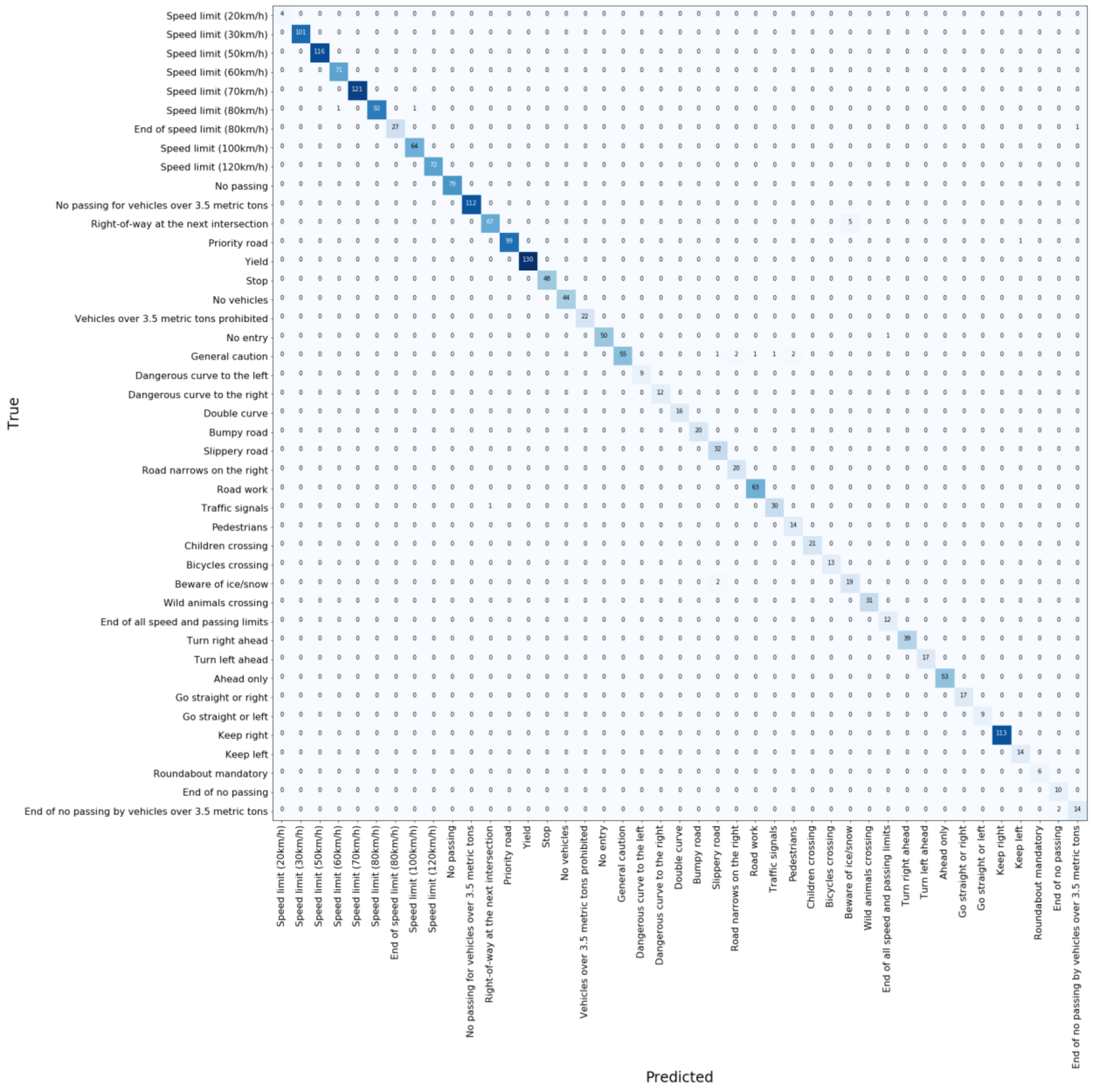


Figure 18: Confusion Matrix before Image Preprocessing

The image above clearly shows that there are a lot of false predictions when running the model with unprocessed images, but the following is the confusion matrix with the pre-processed pictures, with an accuracy of 98 per cent higher than the previous one showing quite well improving the true positive.

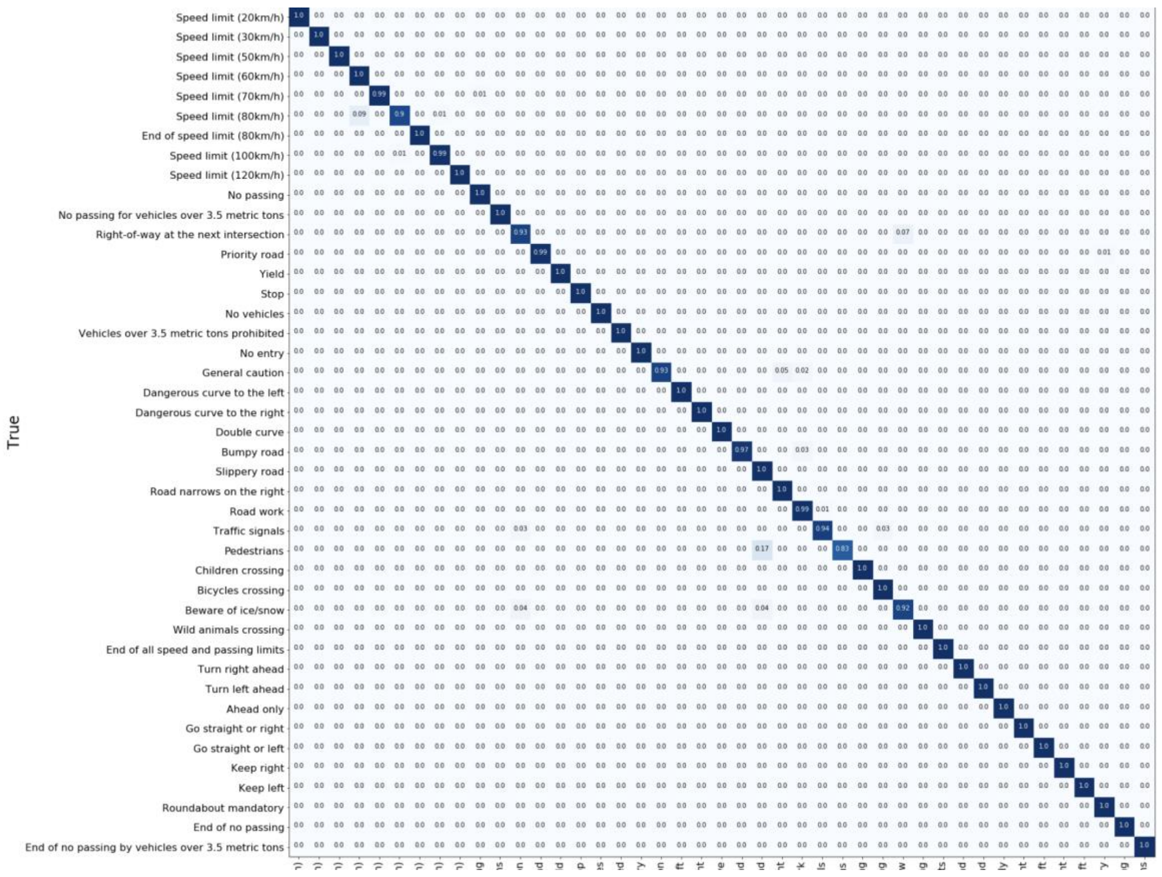


Figure 19: Confusion Matrix after Image Preprocessing

I obtained 97.86 percent precision on the original test set. I could explore how these changes in the future are managed by the model by blurring/distorting new images or modifying contrast. However, the precision of 98,67 percent for new images. Annealing of the learning rate, increasing dropping, increasing the batch size with increasing accuracy. Because my model does not have the challenge of classifying the image so highly, my images are not quite intuitively part of the GTSRB. Example: Elderly travel Looks like the crossing of children and the model predicts it well. Color invariance is quite good, the forecast of turning left works well.

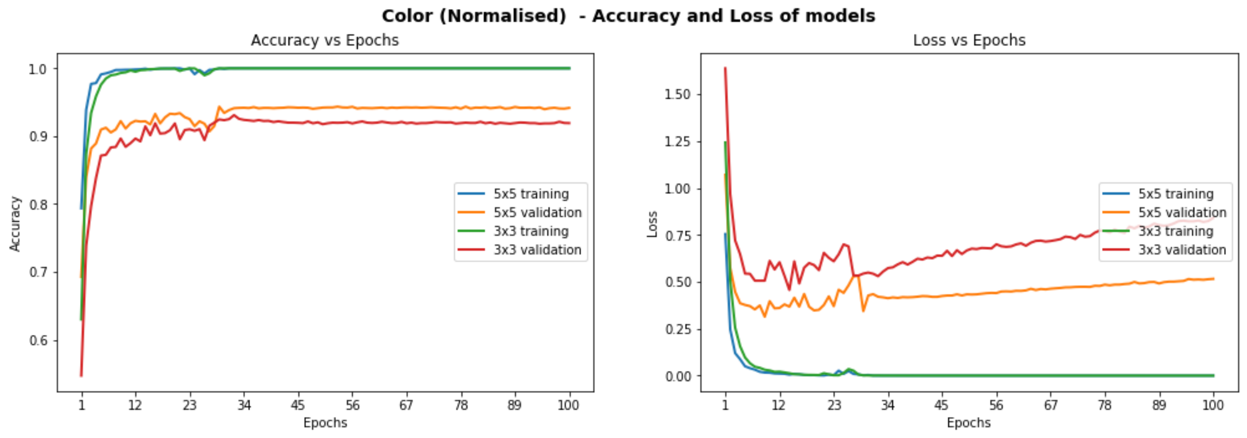


Figure 20: Graphs with RGB Normalized Images

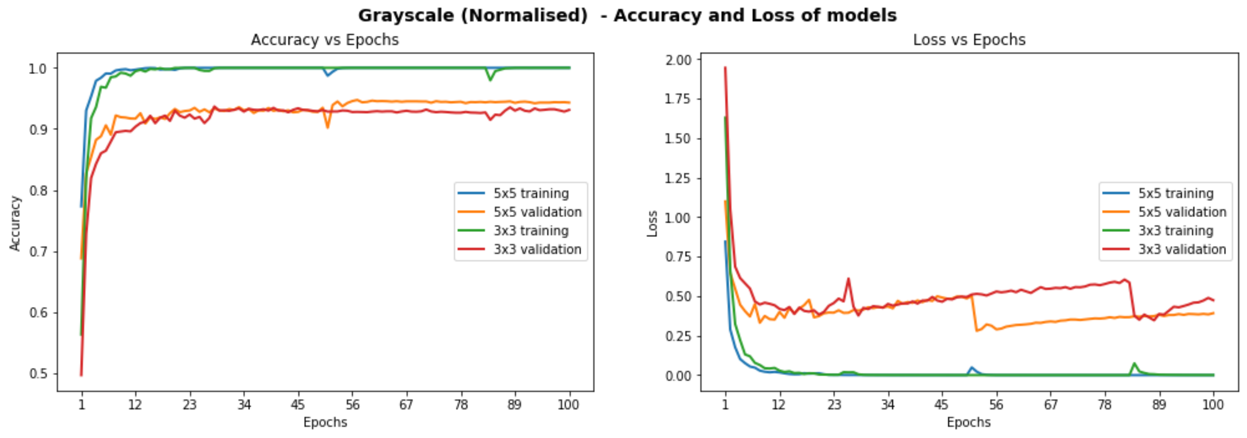


Figure 21: Graphs with Grayscale Normalized Images

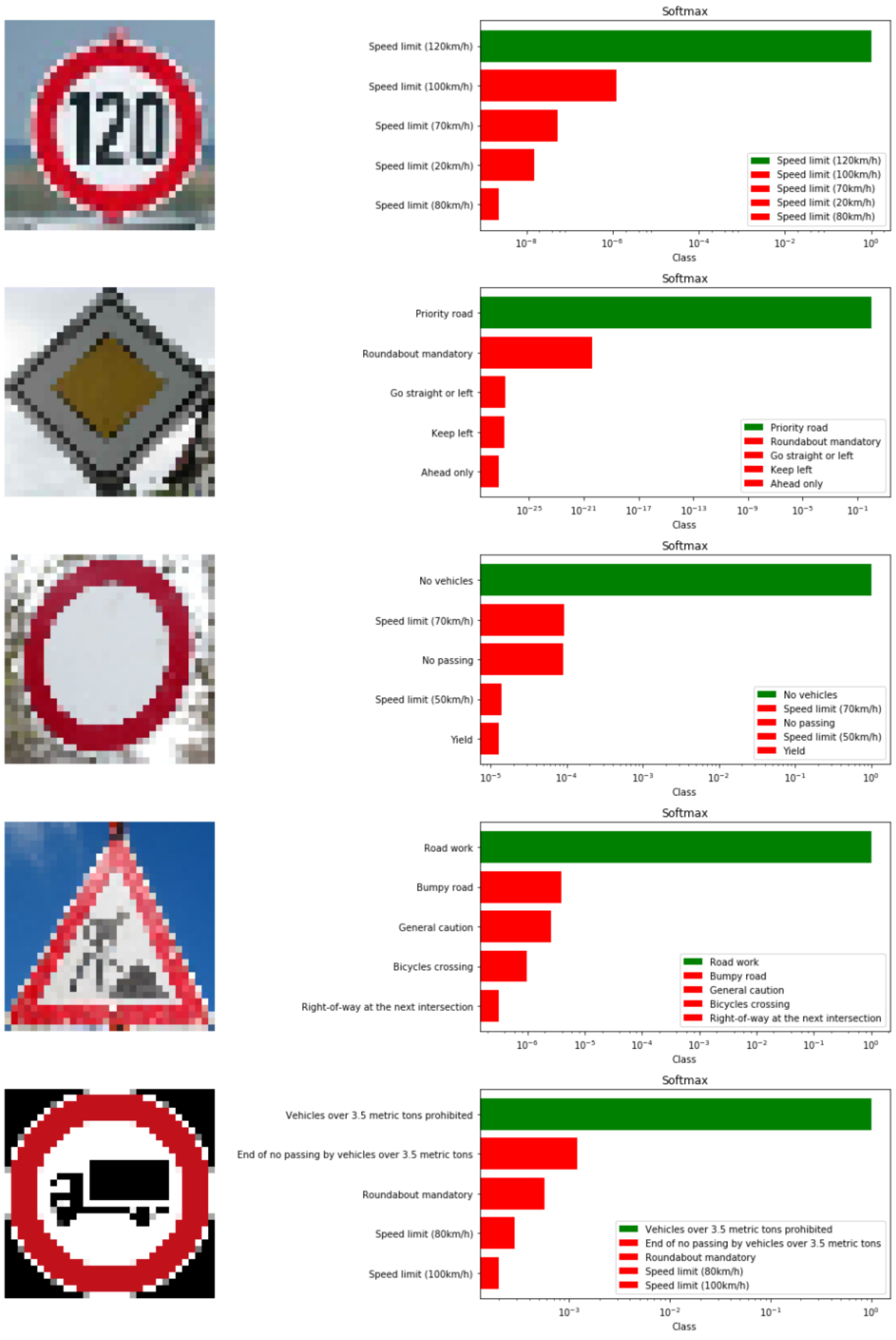


Figure 22: The ROI plotted as bar graphs of new images

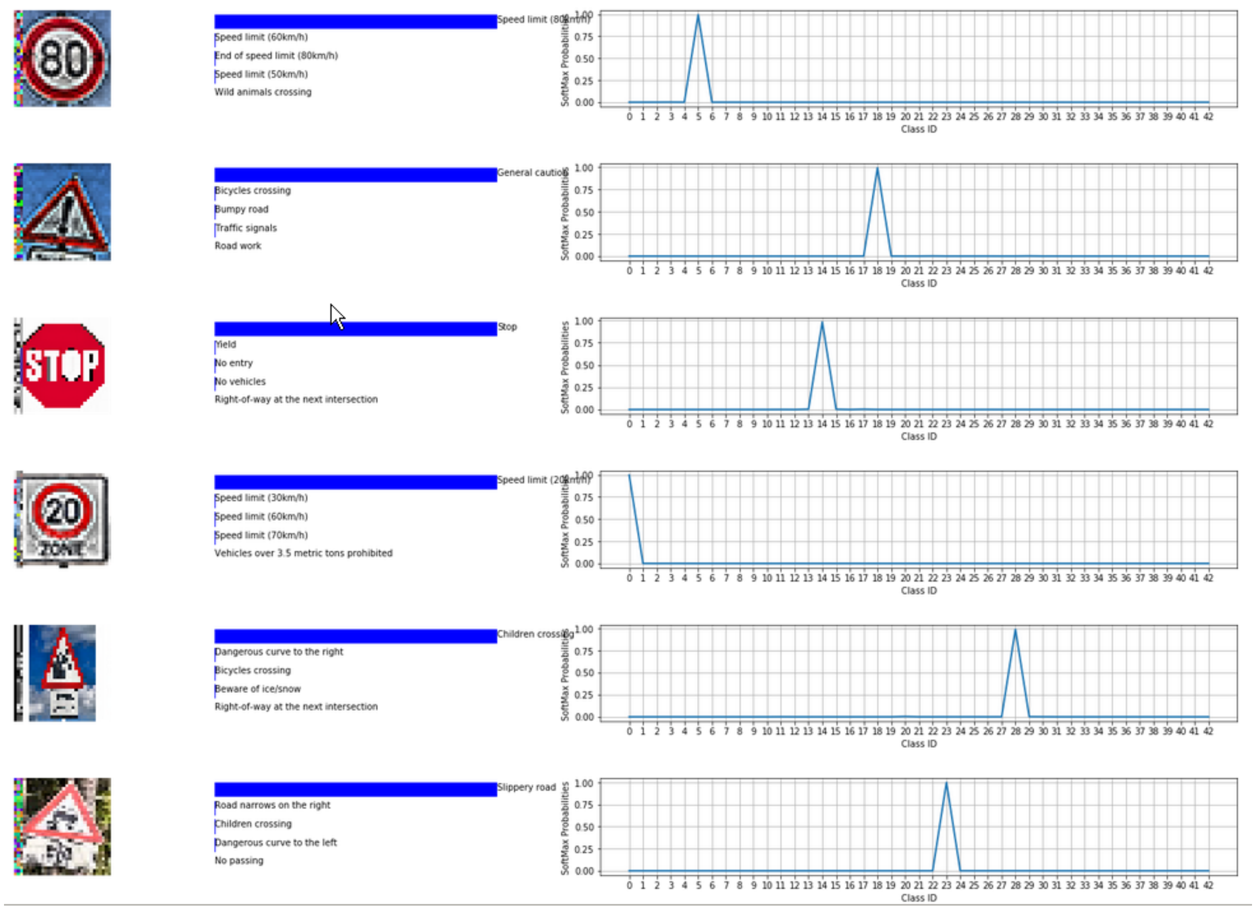


Figure 23: The ROI plotted as line graphs of new images

Chapter 5. Discussion and Future Work

There was a great deal of success in the results of this project. The model works well with existing and new images and gives more than 98 per cent success rate. However, as it always has a place for improvement with everything. Some things have worked not like 32*32*3 for the pictures. How accurate prediction is determined by the resize method order. The resize method tends to destroy image appearance ratios, which deteriorate the model's performance.

The improvements and better results can be achieved in the future as follows:

- Higher framerate better performance
- Use other methods such as YOLO or SSD
- Processing dynamic image
- Classification by using CNN
- Make the Dataset

Chapter 6. Conclusion

Following evaluation of the test set, the model is tested with a number of images. The following is a brief summary on this:

The layer is completely connected to the overlayer layer. It translates the results in one vector from previous layers and can be used as a next level input. The first fully connected layer is used to predict the proper mark by weighing the input in the function analysis. Electrically connected output layer provides the final probabilities for every mark. A fully connected layer is intended to classify images into a label based on the results of the process. No cars with a sign in front of you with something to write, but writing isn't in the 43 classes. Elderly Crossing sign looks like a crossing for children but not in the GTSRB. A slightly rotated sign for general caution, slippery road and snow warning, a simple children's sign for crossing, etc. The SoftMax probability is standardized, i.e. with a SoftMax function that produces a range of probability, the logistical regression output is activated. These are designed and standardized (0-1). The top five assumptions are in the next bar chart.

Since our model is not as challenging to classify images, it even predicts images not quite intuitively part of the GTSRB. For example - Elderly Crossing Looks like the crossing of kids and the model predicts it well. Color invariance is quite good, the prediction of turning left works well.

References

[1] Abdi, L. & A. Meddeb. 2017. Deep learning traffic sign detection, recognition and augmentation. In Proceedings of the Symposium on Applied Computing, 131-136. ACM

[2] Benallal, M. & J. Meunier. 2003. Real-time color segmentation of road signs. In CCECE 2003-Canadian Conference on Electrical and Computer Engineering. Toward a Caring and Humane Technology (Cat. No. 03CH37436), 1823-1826. IEEE.

[3] Carrivick, J. L., M. W. Smith & D. J. Quincey. 2016. Structure from Motion in the Geosciences. John Wiley & Sons.

[4] Kiran, C., L. V. Prabhu & K. Rajeev. 2009. Traffic sign detection and pattern recognition using support vector machine. In Advances in Pattern Recognition, 2009. ICAPR'09. Seventh International Conference on, 87-90. IEEE.

[5] Houben, S., J. Stallkamp, J. Salmen, M. Schlipsing & C. Igel. 2013. Detection of traffic signs in realworld images: The German Traffic Sign Detection Benchmark. In The 2013 international joint conference on neural networks (IJCNN), 1-8. IEEE.

[6] Scott, P., S. Cooper, P. Ingram, H. Chalmers & S. Miller. 2011. Road Asset Management Systems. In Eighth International Conference on Managing Pavement AssetsFugroFederal Highway AdministrationIntervial ChileCAF-Banco de Desarrollo de America Latina.

[7] Gu, T., B. Dolan-Gavitt & S. Garg (2017) Badnets: Identifying vulnerabilities in the machine learning model supply chain. arXiv preprint arXiv:1708.06733.

[8] Stein, G. P., O. Shachar, Y. Taieb & U. Wolfovitz. 2011. Detecting and recognizing traffic signs. Google Patents.

[9] Rasdorf, W., J. E. Hummer, E. A. Harris & W. E. Sitzabee (2009) IT issues for the management of highquantity, low-cost assets. Journal of Computing in Civil Engineering, 23, 91-99.

[10] Houben, S., J. Stallkamp, J. Salmen, M. Schlipsing & C. Igel. 2013. Detection of traffic signs in realworld images: The German Traffic Sign Detection Benchmark. In The 2013 international joint conference on neural networks (IJCNN), 1-8. IEEE.

[11] R. Vicen-Bueno, R. Gil-Pita, M.P. Jarabo-Amores and F. L´opez-Ferreras, “Complexity Reduction in Neural Networks Applied to Traffic Sign Recognition”, Proceedings of the 13th European Signal Processing Conference, Antalya, Turkey, September 4-8, 2005.

[12] Lopez-Ferreras, “Multilayer Perceptrons Applied to Traffic Sign Recognition Tasks”, LNCS 3512, IWANN 2005, J. Cabestany, A. Prieto, and D.F. Sandoval (Eds.), Springer-Verlag Berlin Heidelberg 2005, pp. 865-872.

[13] H. X. Liu, and B. Ran, "Vision-Based Stop Sign Detection and Recognition System for Intelligent Vehicle", Transportation Research Board (TRB) Annual Meeting 2001, Washington, D.C., USA, January 7-11, 2001.

[14] H. Fleyeh, and M. Dougherty, "Road And Traffic Sign Detection And Recognition", Proceedings of the 16th Mini - EURO Conference and 10th Meeting of EWGT, pp. 644-653.

[15] D. S. Kang, N. C. Griswold, and N. Kehtarnavaz, "An Invariant Traffic Sign Recognition System Based on Sequential Color Processing and Geometrical Transformation", Proceedings of the IEEE Southwest Symposium on Image Analysis and Interpretation Volume , Issue , 21-24 Apr 1994, pp. 88 – 93.

[16] M. Rincon, S. Lafuente-Arroyo, and S. Maldonado-Bascon, "Knowledge Modeling for the Traffic Sign Recognition Task", Springer Berlin / Heidelberg Volume 3561/2005, pp. 508-517.

[17] C. Y. Fang, C. S. Fuh, P. S. Yen, S. Cherng, and S. W. Chen, "An Automatic Road Sign Recognition System based on a Computational Model of Human Recognition Processing", Computer Vision and Image Understanding, Vol. 96 , Issue 2 (November 2004), pp. 237 – 268.

[18] C. Bahlmann, Y. Zhu, V. Ramesh, M. Pellkofer, T. Koehler, "A System for Traffic Sign Detection, Tracking, and Recognition Using Color, Shape, and Motion Information", Proceedings of the 2005 IEEE Intelligent Vehicles Symposium , Las Vegas, USA., June 6 - 8, 2005.

[19] L. Pacheco, J. Batlle, X. Cufi, "A new approach to real time traffic sign recognition based on colour information", Proceedings of the Intelligent Vehicles Symposium, Paris, 1994, pp. 339–344.

[20] Y. Aoyagi, T. Asakura, "A study on traffic sign recognition in scene image using genetic algorithms and neural networks", Proceedings of the IEEE IECON Int. Conf. on Industrial Electronics, Control, and Instrumentation, Taipei, Taiwan, vol. 3, 1996, pp. 1838–1843.

[21] M. Lalonde, Y. Li, Detection of Road Signs Using Color Indexing, Technical Report CRIM-IT-95/12-49, Centre de Recherche Informatique de Montreal. Available from: publications.html, 1995.