**U.S Traffic Sign Detection Using Deep Learning Networks**

A Thesis

Submitted to

the Temple University Graduate Board

In Partial Fulfillment

of the Requirements for the Degree

MASTERS OF SCIENCE

IN ELECTRICAL ENGINEERING

by

Peter Dane Mollica

April, 2015

Examining Committee Members:

Dr. Joseph Picone, Advisory Committee Chair, Department of ECE

Dr. Iyad Obeid, Department of ECE

Dr. Haibin Ling, Department of Computer Science

Dr. Yimin Daniel Zhang, Department of ECE

**ABSTRACT**

Traffic Sign Recognition (TSR) from video images is an integral part of the driver support functions needed to make intelligent vehicles a reality. TSR is composed of two components: detection and classification. The focus of the proposed research is detection of U.S traffic signs based on the LISA dataset, the largest publicly available U.S traffic sign dataset in the world, comprising over 9,000 images.  
Detection methods involving Integral Channel Features and Aggregate Channel features have achieved state-of-the art performance. Our proposed research consists of a comparative study of the performance of  
Integral Channel Features and Aggregate Channel features versus using Convolutional Neural Networks (CNN). Our aim is to explore the detection performance of the CNN by varying the convolutional layers, max-pooling layers, and the fully-connected layers. We will evaluate our detection performance by using the PASCAL measure, which is a standard metric for this application.

**TABLE OF CONTENTS**

[**ABSTRACT** ii](#_Toc446024858)

[**TABLE OF CONTENTS** iii](#_Toc446024859)

[**LIST OF TABLES** iv](#_Toc446024860)

[**LIST OF FIGURES** v](#_Toc446024861)

[Introduction 1](#_Toc446024862)

[1.1 Traffic Sign Recognition Overview 3](#_Toc446024863)

[1.2 Related Work 5](#_Toc446024864)

[1.3 Thesis Overview 7](#_Toc446024865)

[Background 8](#_Toc446024866)

[2.1 Sliding Window Detection 8](#_Toc446024867)

[2.2 Current State-of-the-Art Performance 11](#_Toc446024868)

[2.2.1 Integral Images 11](#_Toc446024869)

[2.2.2 Integral Channel Features 12](#_Toc446024870)

[2.2.3 Aggregate Channel Features 14](#_Toc446024871)

[2.2.4 AdaBoost Classifier 15](#_Toc446024872)

[2.3 Deep Learning 17](#_Toc446024873)

[2.3.1 Feed Forward Neural Network 17](#_Toc446024874)

[2.3.2 Convolutional Neural Network 20](#_Toc446024875)

[References 23](#_Toc446024876)

**LIST OF TABLES**

**Table 1.1 Summary of Surveyed TSD Techniques until 2012…………………………………………6**

**LIST OF FIGURES**

**Figure 1.1: Examples of Vienna Convention signs and U.S signs……………………………….…….. 2**

**Figure 1.2: Block Diagram of TSR System ……………………………….…………………………… 3**

**Figure 1.3 Basic Block Diagram of Detection Task ...…………………….…………………………… 4**

**Figure 2.1 Sliding Window Process ...…………………….…………………………………………..… 9**

**Figure 2.2 Example of Non-Maximum Suppression .…….………………………………………..…. 10**

**Figure 2.3 Sliding Window Detection Algorithm ……….………………………………………….… 10**

**Figure 2.4 Integral Image Example ...…………………….…………………………………………… 12**

**Figure 2.5 Example of Image Channels ………………….…………………………………………… 13**

**Figure 2.6 Aggregate Channel Framework …..………….…………………………………………… 14**

**Figure 2.7 AdaBoost Algorithm ………………….……….…………………………………………… 15**

**Figure 2.8 Biological Neuron (left) and Computational Model (right)……………………………… 17**

**Figure 2.9 Fully Connected Feedforward Neural Network ………….……………………………… 18**

**Figure 2.10 Neural Network (left) and Basic Neuron Arrangement of CNN (right) ……….……… 20**

**Figure 2.11 Pooling Process ………………………………………………………………….………… 21**

**Figure 2.12 Full Convolutional Neural Network Architecture ...………………………….………… 22**

Chapter 1

Introduction

For the past several years there has been significant research interest in self-driving vehicles and Advanced Driver Assistance Systems (ADAS), most notably Google’s Self-Driving Car Project. In order for these systems to become more and more autonomous, it is essential for the integration of Traffic Sign Recognition (TSR) technology. TSR systems are composed of two main components, classification and detection. The classification component focuses strictly on classifying the type of traffic sign, after the sign has be detected. Whereas, the detection component focuses on locating the traffic sign in a sequence of video images. This work focuses exclusively on the detection task of TSR systems.

Traffic sign detection has been heavily researched on European traffic signs, or more specifically traffic signs that follow the Vienna Convention of Road Signs and Signals (United Nations et al., 1978). This is largely due to the introduction of the German Traffic Sign Detection Benchmark (GTSDB) (Houben 2013) competition. This competition has shown promising results, however the performances of the top performing models of the GTSDB have not been able to translate as well when used for U.S traffic signs (Mogelmose 2015). One of the main reasons that European models have not be able to translate as well is that, U.S signs and European signs, which have very similar meanings can look significantly different as shown in Figure 1.1 (Mogelmose 2015). Figure 1.1 (a)-(c) are example of Vienna Convention signs and (d)-(f) are examples of U.S traffic signs. Figure 1.1(a)(d) are both Keep Right signs, figure 1.1(b)(e) are both Left Turn signs, and figure 1.1 (c)(f) are both Speed Limit signs. From figure 1.1 we can clearly see there are significant difference, in geometry and color, for the Vienna Convention and U.S traffic signs that have the same meaning. In order to close this gap in performance, the largest U.S traffic sign dataset in the world, the Laboratory for Intelligent and Safe Automobiles (LISA) dataset (Mogelmose 2012), was created. The LISA dataset contains approximately 10,000 images containing U.S traffic signs and approximately 11,000 images that have similar scenes but do not encompass any traffic signs in the images.



**(a) (b) (c) (d) (e) (f)**

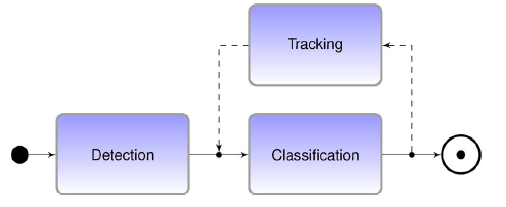
**Figure 1.1 Examples of Vienna Convention signs and U.S signs**

Two of the most promising methods used for traffic sign recognition include, Integral channel Features and Aggregate Channel Features. Integral Channel and Aggregate Channel features were originally applied to Pedestrian Detection (Dollar 2009, 2014) and later adopted for TSR. Where they have shown to achieve state-of-the-art performance for both European and U.S traffic signs (Mogelmose 2015). Although these methods have demonstrated state-of-the-art performance there is still room for improvement. To our knowledge, deep learning architectures, specifically Convolutional Neural Networks (CNN), have not been explored for TSR. CNNs have shown to be very powerful models for image recognition tasks, as was shown in the ImageNet Challenge (Krizhevsky 2012). Motived by these results, our proposed research consists of a comparative study of the performance of Integral Channel Features and Aggregate Channel features versus using Convolutional Neural Networks (CNN). Our aim is to explore the detection performance of the CNN by varying the convolutional layers, max-pooling layers, and the fully-connected layers. We will evaluate our detection performance by using the PASCAL measure (Everingham 2010), which is a standard metric for this application.

This chapter will focus first on a simple overview of traffic sign recognition systems, followed by a discussion on the related work on both the German Traffic Sign Detection Benchmark (GTSDB) and on the LISA dataset. We will conclude this chapter with an overview of the entire thesis proposal.

## Traffic Sign Recognition Overview

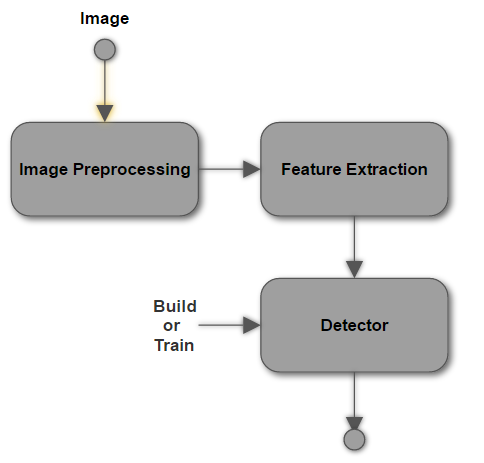
Although in this work we will be examining the detection aspect of Traffic Sign Recognition (TSR) systems, it will be useful to describe the whole system. Typically when we authors are referring to TSR systems they are referring to the detection of the traffic signs, the classification of that traffic sign, once it has been detected, and the tracking of the traffic sign. Figure 2 (Mogelmose 2012) shows a basic block diagram of the TSR system.



**Figure 1.2 Block Diagram of TSR System**

When given an image or video sequence as an input, the detection task is only concerned with discovering where at in the image the traffic sign is located. Whereas, classification is only concerned with deciding what type of traffic sign the detected traffic sign is. The final component is tracking, which tracks the detected sign from frame to frame. Each component can be approached separately in different ways. However for a full TSR system to work each component will depend on each other.

Since our work is focused on the detection task we will only look at this task at a closer level. A simple model of the Detection task can be seen below in figure 3. Figure 3 is a very simplistic block diagram of the detection process, however most method will look like this in some form or another. They will typically have an image preprocessing stage, then a feature extraction stage, and then a detection stage. The image preprocessing stage is used to help clean the data or transform the image in order to prepare it for the feature extraction stage. In the feature extraction stage, features such as edges, Harr-like features, and color features are taken from the image and sent to the detector. The detector depends on the type of method that we use. Typically there are two methods used, one is a model-based approach and the other is a learning-based approach, both will be discussed in more detail in the next chapter. The purpose of the detector is to take the features as input and make a decision on where the traffic sign is located in the image. Although in this work the focus is on the detection step it is important to note that without the classification and tracking components, a TSR system is useless.

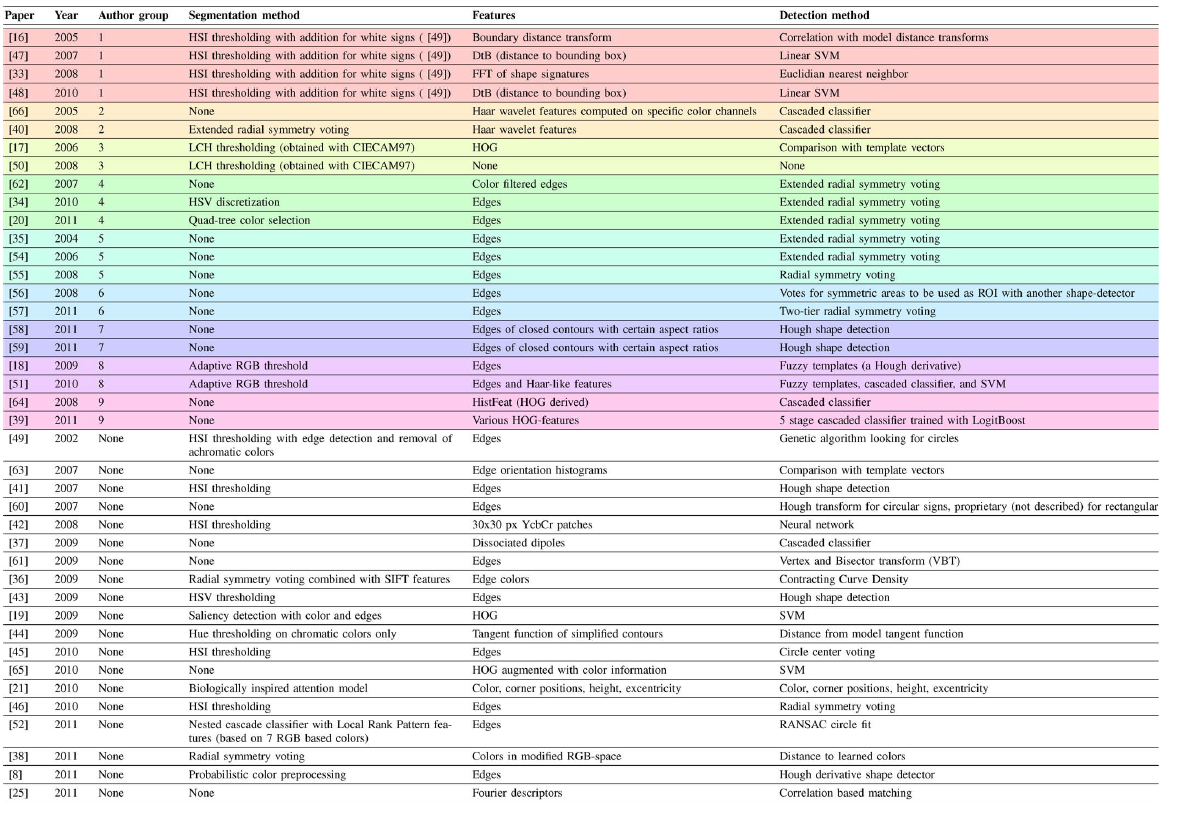


**Figure 1.3 Basic Block Diagram of Detection Task**

## Related Work

Traffic sign detection has been researched for a little over a decade. Some of the earliest methods tried to take advantage of the fact that traffic signs were designed to stand out in the environment, often through shape and color. The early shape-based methods searched for specific geometrical shapes such as rectangular or circular objects. For example Loy (2004) used edge features with a radial symmetry model for a dataset of approximately 50 images. Early color-based methods extracted colors from different color spaces of the input image, and used different thresholding techniques to extract the traffic sign. For example Vazquez-Reina (2005) converted the image into HIS color space and used thresholding in order to extract the traffic signs. While in theory, the color and shape of traffic signs can be very well defined, they can have some very practical issues, such as, illumination changes, damaged or worn signs, occlusions, and traffic signs blending in with background color. This lead to the implementation of learning-based methods that used more descriptive features such as Harr-like wavelet features, introduced by Viola and Jones (2001) for facial recognition, and HOG features, introduced by Dalal and Triggs (2005). Mogelmose (2012) surveyed the traffic sign detection field which illustrates both the early model-based methods and learning-based methods. A summary of the different techniques surveyed can be seen below in table 1 (Mogelmose 2012). For more specific details of each method please refer to Mogelmose (2012). However trying to compare the performances of each surveyed methods becomes very problematic due to the inconsistencies in types of traffic signs and the number of traffic signs used in each method.

The creation of the German Traffic Sign Detection Benchmark (GTSDB) competition (Houben 2013) has made the comparison of different TSD methods much easier and has significantly pushed state-of-the-art performances on European signs. Typically TSR research was split into two different methodologies, model-based and learning-based approaches. However more recently learning-based methods have continued to outperform model-based methods and are the preferred choice over model-based approaches. This was also observed in the GTSDB competition, where all of the frontrunners used learning-based approaches. The top three teams of the GTSDB competition where: Team VISICS (Houben 2013), Team Litsi (Liang 2013), and Team wgy@HIT501 (Wang 2013). Team VISICS proposed a method that uses Integral Channel Features (ICF), which was originally proposed by Dollar et al. (2009) for pedestrian detection. Team VISICS trained these features using a boosted decision forest. Team Litsi proposed a method that first segmented the image into regions of interest, through shape matching and color classification. They then used HOG features and color histograms to train a SVM. Team wgy@HIT501 also used HOG features, however found the best candidates with LDA. The best HOG feature candidates were trained using an IK-SVM. The GTSDB was one of the major catalysts that pushed the traffic sign detection performance to near perfection for European traffic signs.



**Table 1.1 Summary of Surveyed TSD Techniques until 2012**

However, the research activity in U.S traffic signs was significantly less than that of the European traffic signs. Staudenmaier (2012) proposed a method for U.S speed limit signs, which used a Bayesian Classifier Cascade with intensity features and tensor features. The worked showed to have a good detection rate above 90% but at the cost of multiple false positives per image, which in comparison to the European state-of-the-art methods was much worse. Liu (2012) also ran into a similar issue of multiple false positives with a good detection rate, but only for speed limit signs. One of the main problems for U.S traffic signs, was the fact that there wasn’t a large dataset that can be used as a benchmark. It wasn’t until Mogelmose (2012) that the LISA dataset became available. This dataset became the largest collection of U.S traffic signs in the world. With it, Mogelmose (2015) was able to achieve state-of-the-art performance by using Integral Channel features and Aggregate Channel features with a boosted decision tree forest. There was still a performance gap between their performance and the state-of-the-art performance of the European dataset, however this gap was significantly less than that of the methods used for U.S traffic signs. Lim (2014) also presented work on the LISA dataset with somewhat worse detection performance that Mogelmose (2015), but tries to address the issue of adverse weather conditions.

## Thesis Overview

The remainder of this thesis will be divided into the following sections. In Chapter 2 we will discuss all of the necessary background information. Including model-based approaches vs. learning-based approaches, the current state-of-the-art methods for the GTSDB and LISA datasets, and deep learning architectures. Chapter 3 will explain the proposed method and the data that we will be using in this work. Chapter 4 will include the preliminary experiments and the results of the experiments. Chapter 5 will discuss the expected outcomes and a timeline for the remaining tasks.

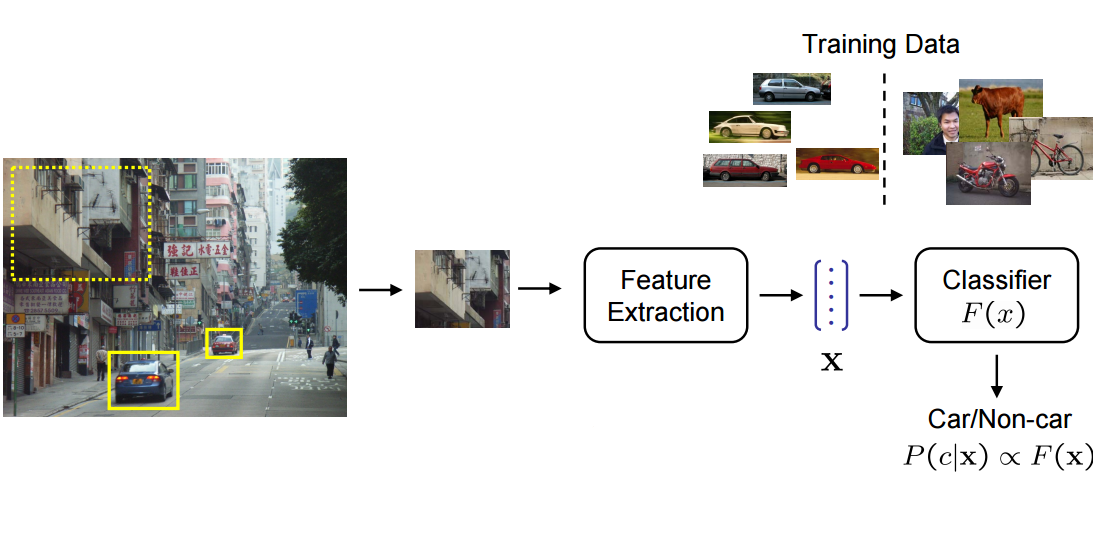
Chapter 2

Background

In this chapter we will describe all of the necessary background information for this work. We will first describe the sliding window detection method, which is implemented in most object detection/recognition systems. We will then transition into a detailed description of the current state-of-the-art method for traffic sign detection. Then the final section will discuss the basics of deep learning and conclude with a detailed description of Convolutional Neural Network (CNN) architectures, which will be our proposed method for traffic sign detection.

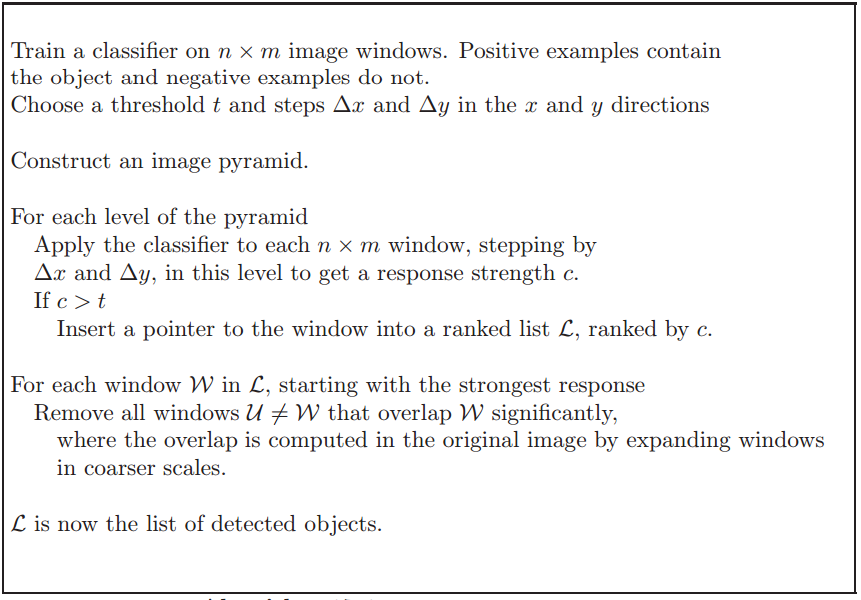
## Sliding Window Detection

Most object detection/recognition systems will typically implement a sliding window method. The first step of any learning-based object detection system, which implements a sliding window, is to train a classifier from a dataset of labeled images of size . The dataset should contain positive and negative image samples. Where the positive samples are large centered instances of the object we are looking for, of size . The negatives images in the dataset should contain images with scenery similar to that of the positive images however, they cannot contain the object in them. To obtain the negative samples, image patches of size are randomly extracted at different scales of the negative images. Once the dataset is organized a binary classifier needs to be trained on the positive and negative samples. Instead of training on the image patches themselves we train on features that are extracted from the image patches. Once the classifier is trained, we will test on test images by extracting an image patch, extract the features from the image patch, then pass the features into the binary classifier, which will determine whether it is or is not the object we are looking for. Figure 2.1 illustrates this process for a car detection system (Zimmerman 2012).



**Figure 2.1 Sliding Window Process**

There are two subtleties that need to be examined closer when applying the sliding window detector. The first being that not all of the objects in the image will be the same size as the image patches that the classifier was trained on. To remedy this issue, we can slide the windows across different scales of the same image (Forsyth 2012). The second subtlety deals with the response of the classifier, if the classifier response is above a specified threshold it will classify that window as a detected object. However depending on the distance between each window, we can have multiple windows that detect the same object, this can be visualized in figure 2.2. In order to handle this issue we would apply a method called Non-Maximum Suppression (Forsyth 2012). Non-Maximum Suppression is a method that looks at local responses of windows classified as a detected object. If we have multiple windows that are overlapping, where the overlapping area is above a certain threshold, we will only retain the window with the highest response. To summarize, figure 2.3 shows the sliding window detection algorithm (Forsyth 2012).



**Figure 2.3 Sliding Window Detection Algorithm**



**Figure 2.2 Example of Non-Maximum Suppression**

## Current State-of-the-Art Performance

Mogelmose (2015) has shown that Integral Channel Features (ICF) and Aggregate Channel Features (ACF) have achieved state-of-the-art performance for U.S traffic signs on the LISA dataset. In this section we will provide a detailed description of Mogelmose’s (2015) methodology, which we will use as the baseline measurements of this work. The main focus of Mogelmose’s (2015) methodology is in the feature extraction stage. The features used were Integral Channel features, which were first introduced by dollar (2009) for pedestrian detection. Dollar later introduced an extension of the Integral Channel features, which were the Aggregate Channel Features (Dollar 2014). The features were trained using an AdaBoost Classifier. This section will provide all of the necessary background information for these topics.

## Integral Images

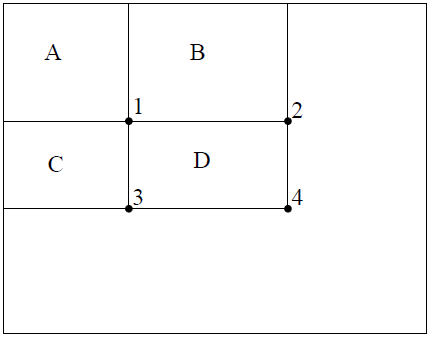
In order to understand the specific features used by Mogelmose (2015), we must give some background information on integral images, which were first introduced by Viola and Jones (2001). An integral image is a quick way of calculating rectangular features by summing the pixel values in a given rectangular region of an image. An integral image is defined at each pixel by the following equation (Viola and Jones 2001):

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

Where *ii(x,y)* is the integral image and *i(x,y)* is the original image. Equation 1 tells us that the integral image at pixel location *x* and *y* is the sum of the all of the pixel values above and to the left of the original image. We can calculate the entire integral image in one pass over of the original image by using the following two equations:

|  |  |  |
| --- | --- | --- |
|  |  | (2) |
|  |  | (3) |

Where *s(x,y)* is the cumulative row sum of the original image. Using an integral image we can compute and sum or difference of any rectangular area, this is illustrated in figure 2.4 (Viola and Jones 2001).



**Figure 2.4 Integral Image Example**

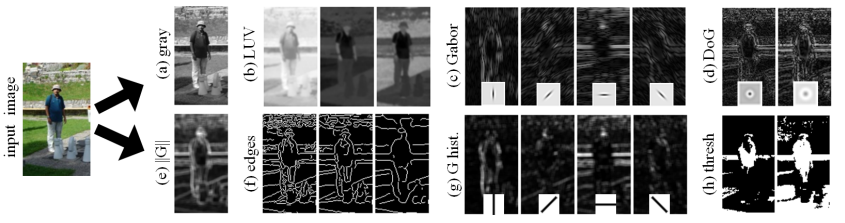
If we want to find the sum of the pixels in the rectangle D, in figure 2.4, we only need to use the four reference points: 1, 2, 3, and 4. Where the value at reference point 1 is the sum of the pixels in rectangle A. The value at reference point 2 is the sum of rectangles A and B. At reference point 3 the value is A+C. We can then compute the sum within D as . This gives us the ability to calculate the sum in constant time, O(1). With the knowledge of integral images we can now examine, in more detail, the Integral Channel Features used in Mogelmose (2015).

## Integral Channel Features

The idea behind ICF is to create multiple image channels by making linear and non-linear transformations of the original input image. The channels are then transformed into integral image. This allows for more efficient feature extraction through the integral images. For a given input image *I*, a channel can be viewed as a new image where the original input pixels are mapped to the new image channel, *C*. These mappings can be either linear or non-linear transformations and can be represented by the channel generating function Ω. The transformation can be written:

|  |  |  |
| --- | --- | --- |
|  |  | (4) |

There are multiple channel features that can be extracted from a transformed image channel, some that are more complex and longer to compute than others. One of the fastest to compute is a first-order channel feature, which is the sum of pixels in a fixed rectangular region. We have shown in the previous section that if the integral image is generated we can calculate the sum of a rectangular region in constant time, O(1), as it only requires operations on 4 reference values. A higher-order channel feature, is defined as a feature that is computed using multiple first-order features. An example of a higher-order feature is a Haar feature (Viola Jones 2001), which involves a sum of 2-4 rectangles arranged in patters that compute first and second order image derivatives at multiple scales.



**Figure 2.5 Example of Image Channels**

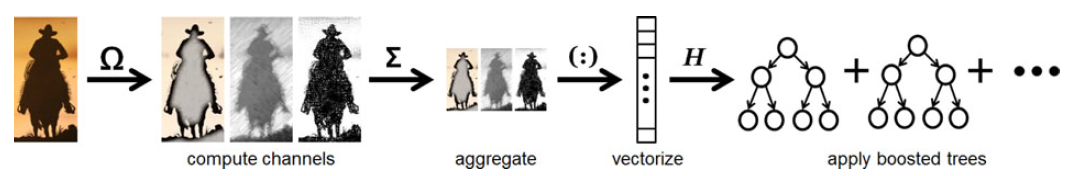
Before we extract the first-order or higher-order channel features, we must first transform out image into channels. Figure 2.5 shows an example of different types of channels that were transformed from the original input image. Each channel contains different pieces of information from the input image. For example the Gabor channels, in figure 2.5, are a convolution of 4 oriented Gabor filters with the input image. There are many different types of transformations that can performed on the input image. One such transform is a linear filter where we convolve a linear filter, such as the Gabor filter, with the input image. We also have non-linear transformations, such as gradient magnitudes, which looks at specific edge strengths in the image, see figure 2.5(e). There are also gradient histogram channels, which is a weighted histogram where the gradient angle is the index of the bin and the gradient magnitude at that angle is the magnitude of the bin. We can express the channels mathematically as:

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

is the gradient histogram channel for angle , where and are the gradient magnitude and quantized gradient angle of , respectively. In chapter 3 we will discuss the specific channels and features used for our baseline results.

## Aggregate Channel Features

Aggregate Channel Features (ACF) were proposed as an extension of ICF by dollar (2014). They were designed to provide a faster alternative to ICF, but they have also shown to have slightly better detection performance when compared to ICF, in some cases. ACF uses the same principle idea of ICF, which is computing features across different image channels and in this work the channels are actually the same channels used in ICF. The ACF method proposed by dollar (2014) for pedestrian detection can be visualized in figure 2.6.

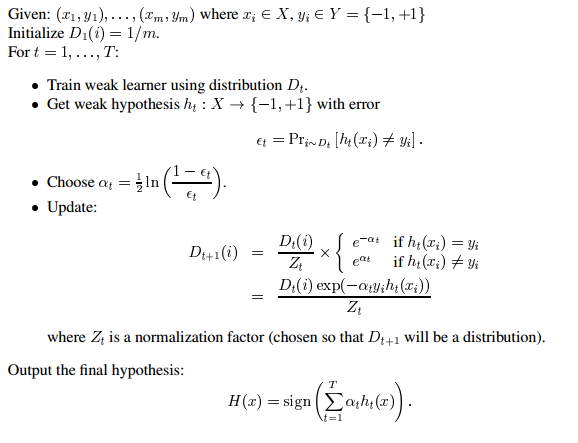


**Figure 2.6 Aggregate Channel Framework**

The first step to this process is the same as the Integral Channel Features, where we map the image into channels with some mapping function,. Then next step is to sum each block of pixels into an integral image. The features are even simpler than ICF, they are just summed blocks of pixels at various scales and stored in a feature vector. The classifier is learned from the feature vectors using AdaBoost.

## AdaBoost Classifier

The final part of Mogelmose’s (2015) method is the classifier. Their method uses machine learning technique called, AdaBoost. This section will give a description of the AdaBoost technique. Freund and Schapire (1996) discovered that it was possible to build a strong classifier from a weighted sum of weak classifiers, which they called boosting. They defined a weak classifier as a classifier that is just better than random guessing, meaning that the accuracy of the classifier is greater than 50%. The weights of each classifier are learned by adjusting the weights of the wrongly classified samples. Each sample that is incorrectly classified, by the current weak classifier, will have a higher weight on the next iteration of the algorithm. The pseudocode for the AdaBoost algorithm can be viewed in figure 2.7.



**Figure 2.7 AdaBoost Algorithm**

We can see from figure 2.7 that we have *m* labeled examples . Where the label values , where -1 will represent negative samples and +1 will represent positive samples. The first step is to initialize our distribution Dt of our samples, which is first initialized as each sample equally weight and the sum of the weights of all the samples are equal to 1. In the algorithm we iterate from t=1,…,T, where T is the number of weak classifiers. On each iteration we are trying to find the weak classifier, , that is going to give us the smallest error rate. This is typically done by using a linear decision stump. Also in the same iteration we want to find the new distribution and the weight of the weak classifier. The following equation is used to calculate the weak classifier weight, αt:

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

Where,

|  |  |  |
| --- | --- | --- |
|  |  | (7) |

Equation 7 is just stating that the error rate of the weak classifier is equal to the sum of the misclassified sample weights. The next step of the algorithm is to update all of the sample weights by using the following equations:

|  |  |  |
| --- | --- | --- |
|  |  | (8) |

Where Zt is the normalization factor, this will result in the final strong classifier H(x) seen below:

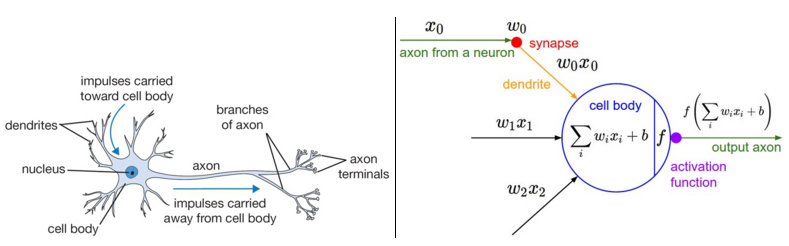
|  |  |  |
| --- | --- | --- |
|  |  | (8) |

## Deep Learning

In this section we will discuss, in detail, the main components of deep learning. We will start with the basic neuron and feed forward neural networks. We will then proceed to discuss the different components of Convolutional Neural Networks (CNNs), which will be the architecture proposed in this work.

## Feed Forward Neural Network

The original inspiration for Neural Networks came from the attempt to model the biological neural systems. The neuron is the basic computational unit of the nervous system. The nervous system is composed of billions of interconnected neurons. A single neuron receives many input signals from the dendrites and sends a single output signal along the axon. Where the axon branches to multiple synapses that connect to other dendrites, of other neurons. Based off of this biological knowledge, the computational model was created. Where the signal () coming from the axiom, of a previous neuron, is multiplied to the weights () of the synapses. The signals of all the synapses are summed and sent to an activation function (), which then sends a single signal back out the axiom. What the model tries to learn are the weights of the synapses. We can visualize the biological neuron and the computational model of the neuron below in figure 2.8 (Li 2015).



**Figure 2.8 Biological Neuron (left) and Computational Model (right)**

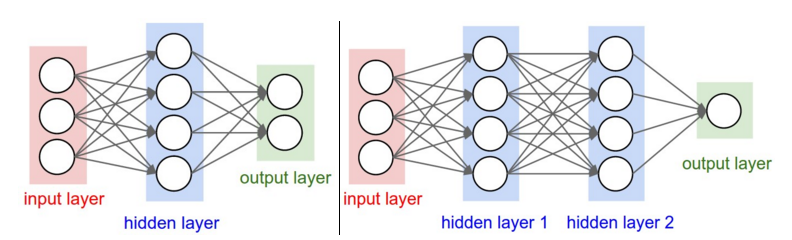
The computational model output of a single neuron can be expressed mathematically as:

|  |  |  |
| --- | --- | --- |
|  |  | (9) |

There are many different types of activations functions that can be used, as the output axiom, but the one that has become the most popular is the Rectified Linear Unit (ReLU). The ReLU is simply a ramp function and expressed as:

|  |  |  |
| --- | --- | --- |
|  |  | (10) |

A complete Neural Network is an interconnection of the modeled neurons. Where the output of one neuron will be the input of another. If we have multiple neurons then all of these outputs can be inputs of a single neuron. However, this should not be interpreted as a big blob of neurons, but as neurons connected in layers. We can visualize this in figure 2.9 (Li 2015).



**Figure 2.9 Fully Connected Feedforward Neural Network**

On the left of figure 2.9 we have a 2-layer Neural Network and on the right we have a 3-layered Neural Network, typically when discussing an N-layer Neural Network the input layer is excluded. The architecture or size of a Neural Network is defined by two measures, how many layers we have and how many neurons are in that layer.

Now that we have a general idea of the architecture we need to examine how to optimize the parameters for our data. Once the architecture is decided upon we need to decide which parameters will minimize the loss (or error) function. Typically the loss function is defined by the mean square error (Bishop 2007):

|  |  |  |
| --- | --- | --- |
|  |  | (11) |

Where , is a vector of the parameters of the Neural Network architecture ( ); and are the feature vector input and the feature vector labels respectively; and is the number of training examples used. As stated before the goal is to find the parameters, , that will minimize the MSE:

|  |  |  |
| --- | --- | --- |
|  |  | (12) |

In order to minimize the mean square error there are many different numerical methods we can choose however the most popular is the gradient decent algorithm. The idea of gradient decent is to find the direction of steepest decent in a local area. The first step is to initialize all of our parameters. After they are initialized, we will update the each parameter by taking the derivate of the mean squared error with respect to each parameter and subtracting it from the original parameter value. The parameter update equation can be seen below:

|  |  |  |
| --- | --- | --- |
|  |  | (13) |

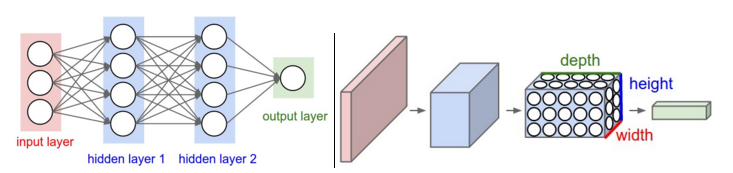
Where is called the learning rate and can be thought of as the size of the step we want to take in our local area. Also to note, one update of the parameters is called an epoch, and when implementing it practically we will limit the number of epochs or stop updating the parameters if it converges, meaning:

|  |  |  |
| --- | --- | --- |
|  |  | (14) |

Where is a very small value. By using gradient decent we will be able to learn all of the parameters, which will optimize our model to minimize the model’s MSE.

## Convolutional Neural Network

Similar to Neural Networks, Convolutional Neural Networks (CNNs) are made up of neurons that have learnable weights. They have a loss function and even the last layer of the CNN can be thought of as a fully connected neural network. However Convolutional Neural Networks are specialized for processing data with a grid-like topology, such as images. The difference between the CNN and a Neural Network is that the layers of a CNN have neurons arranged in 3 dimensions, width, height, and depth. The high dimensionality of the image data makes it impossible to use a single fully connected neural network. We will see that through the architecture of the CNN we will be able to reduce the full image into a single neuron. A visualization of the neuron arrangement of a CNN can be seen below in figure 2.10 (Li 2015).

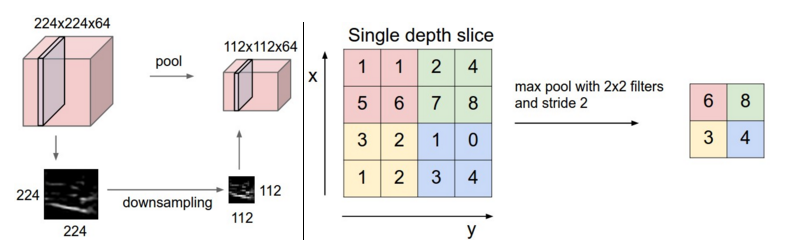


**Figure 2.10 Neural Network (left) and Basic Neuron Arrangement of CNN (right)**

The CNN’s architecture has three main layers: the Convolutional Layer, the Pooling Layer, and the Fully-Connected Layer. A basic CNN architecture may be structured with the input image connected to the Convolutional layer which is connected to a Pooling layer and finally connected to a Fully-Connected layer. In the remainder of this section we will give a description of each layer.

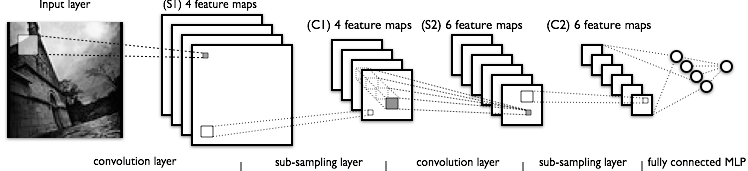
The first layer we are going to discuss is the convolutional layer, which is the main building block of the CNN. The convolutional layer as the name suggests, applies multiple convolutions to the input image with filters that are learned. The filters will slide across the image’s width, height and then depth, which generates a new image. The filters are stacked on top of each other giving the depth dimension that we see in figure 2.10. Intuitively, we want to learn these filters so that when they see a specific feature they will activate a response. The reason for these filters is so the network can create a local connectivity of neurons instead of having learnable weights at each pixel. For example if we have a 32x32x3 image, where the depth is 3 for the RGB channel, if we assigned learnable weights to each pixel we would have a 3072 parameters that need to be learned. However if we have a convolutional filter that is 5x5 now we only have to learn 5x5x3=75 parameters. Note that the depth is three because we must apply the filter to each channel. This also gives us the ability to use stack multiple filters at each convolutional layer and still have less parameters to learn. The number of filters, or the depth, that we use can be analogous to the number of neurons in a hidden layer of a Feedforward Neural Network.

The next layer we will discuss is the Pooling/Sub-sampling layer. The Pooling layer will be found between successive Convolutional layers. The Pooling layer is needed to reduce the spatial size of the output of the convolutional layer. The Pooling layer is applied to every depth image generated from the Convolutional layer. The most common form of pooling is the max 2x2 filter, which downsamples the input image by taking the max value of every 2x2 block as it slides along the height and width of the image. There are two specific hyperparameters associated with the Pooling layer. The first is the spatial extent, F, which is the size of the downsampling filter. The second is the stride, which is the distance between the filters as it moves from left to right and top to bottom of the image. The pooling process can be seen below in figure 2.11 (Li 2015).



**Figure 2.11 Pooling Process**

On the left of figure 2.11 we can see how the Convolutional Layer is downsampled to half the size of the original input image, through the pooling process. On the right of the figure we see how the down sampling is performed using max pooling with 2x2 filters with a stride of 2. A complete architecture can be visualized in figure 2.12.



**Figure 2.12 Full Convolutional Neural Network Architecture**

References

Bishop, C. (2007). "*Pattern Recognition and Machine Learning"* (2nd ed). New York, New York, USA: Springer.

Dalal, N. and Triggs, B. (2005) "Histograms of oriented gradients for human detection," *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, San Diego, CA, USA, 2005, pp. 886-893 vol. 1. doi: 10.1109/CVPR.2005.177

Dollar, P., Appel, R., Belongie, S., and Perona, P. (2014), "Fast Feature Pyramids for Object Detection," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 8, pp. 1532-1545, Aug. 2014. doi: 10.1109/TPAMI.2014.2300479

Dollár, P., Tu, Z., Perona, P., and Belongie S. (2009), “Integral channel features,” in Proc. BMVC, vol. 2, no. 3, p. 5, 2009

Everingham, M., Gool, L., Williams, C., Winn, J., and Zisserman, A. (2010). The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vision* 88, 2 (June 2010), 303-338. DOI=http://dx.doi.org/10.1007/s11263-009-0275-4

Freund, Y. and Schapire, R.E. (1996), ``Experiments with an new boosting algorithm, Machine Learning: Proceedings of the Thirteenth International Conference, Morgan Kauffman, SanFrancisco, pp.148-156 (1996). Introduced AdaBoost

Friedman, J., Hastie, T., and Tibshirani, R. (2000), “Additive LogisticRegression: A Statistical View of Boosting,” Annals of Statistics,vol. 38, no. 2, pp. 337-374, 2000.

Forsyth, D. and Ponce, J. (2012). *Computer Vision a Modern Approach* (2nd ed., p. 549). New York, New York, USA: Pearson

Houben, S., Stallkamp, J., Salmen, J., Schlipsing, M., and Igel, C. (2013), "Detection of traffic signs in real-world images: The German traffic sign detection benchmark," *Neural Networks (IJCNN), The 2013 International Joint Conference on*, Dallas, TX, 2013, pp. 1-8. doi: 10.1109/IJCNN.2013.6706807

Krizhevsky, A., Sutskever, I., and Hinton, G. (2012), Imagenet classification with deep convolutional neural networks. In NIPS, 2012. 1, 2, 3, 4

Li, F., Karpathy, A., and Johnson, J. (2015), "CS231n: Convolutional Neural Networks for Visual Recognition," www. http://cs231n.stanford.edu/

Liang, M., Yuan, M., Hu, X., Li, J., and Liu, H. (2013) “Traffic sign detection by ROI extraction and histogram features-based recognition,” in Proc. IEEE IJCNN, Aug. 2013, pp. 1–8.

Lim, K., Lee, T., Shin, C., Chung, S., Choi, Y., and Byun, H., (2014). Real-time illumination-invariant speed-limit sign recognition based on a modified census transform and support vector machines. In *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication* (ICUIMC '14). ACM, New York, NY, USA, , Article 92 , 5 pages. DOI=http://dx.doi.org/10.1145/2557977.2558090

Liu, W., Wu, Y., Lv, J., Yuan, H., and Zhao, H. (2012) "U.S. speed limit sign detection and recognition from image sequences," *Control Automation Robotics & Vision (ICARCV), 2012 12th International Conference on*, Guangzhou, 2012, pp. 1437-1442.  
doi: 10.1109/ICARCV.2012.6485388

Loy, G. and Barnes, N. (2004), “Fast shape-based road sign detection for a driver assistance system,” in *Proc. IEEE/RSJ Int. Conf. IROS*, 2004, vol. 1, pp. 70–75

Mogelmose, A., Liu, D., and Trivedi, M. M. (2015), "Detection of U.S. Traffic Signs," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 6, pp. 3116-3125, Dec. 2015. doi: 10.1109/TITS.2015.2433019

Mogelmose, A., Liu, D., and Trivedi, M. M. (2014), "Traffic sign detection for U.S. roads: Remaining challenges and a case for tracking," *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, Qingdao, 2014, pp. 1394-1399. doi: 10.1109/ITSC.2014.6957882

Mogelmose, A., Trivedi, M. M., and Moeslund, T. B. (2012), "Vision-Based Traffic Sign Detection and Analysis for Intelligent Driver Assistance Systems: Perspectives and Survey," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1484-1497, Dec. 2012.  
doi: 10.1109/TITS.2012.2209421

Staudenmaier, A., Klauck, U., Kreßel, U., Lindner, F., and Wöhler, C. (2012) “Confidence measurements for adaptive Bayes decision classifier cascades and their application to US speed limit detection,” in Proc. Pattern Recognit., ser. Lecture Notes in Computer Science, vol. 7476, pp. 478–487, 2012.

Vázquez-Reina, A., Lafuente-Arroyo, S., Siegmann, P., Maldonado-Bascón, S., and Acevedo-Rodríguez, F. (2005), “Traffic sign shape classification based on correlation techniques,” in *Proc. 5th WSEAS Int. Conf. Signal Process., Comput. Geometry Artif. Vis.*, 2005, pp. 149–154.

Viola, P. and Jones, M. (2001), "Rapid object detection using a boosted cascade of simple features," *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, 2001, pp. I-511-I-518 vol.1. doi: 10.1109/CVPR.2001.990517

Wang, G., Ren, G., Wu, Z., Zhao, Y., and Jiang, L. (2013), “A robust, coarse-to- fine traffic sign detection method,” in Proc. IEEE IJCNN, Aug. 2013, pp. 1–5.

United Nations (1978), "Vienna Convention on road signs and signals,"

Zimmerman, A (2012), "Category-level Localization", Visual Geometry Group University of Oxford, http://www.robots.ox.ac.uk/~vgg