TRAFFIC SIGN RECOGNITION AND CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORKS

DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF:

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY(ISM), DHANBAD

Submitted by:

Syed Danish Haider
Adm No. 2013JE0124
8th B.Tech ECE
Date: 4/05/2017

Under the supervision of:

Dr. Debjani Mitra
Professor
Dept. of Electronics and
communication Engineering

INDEX

INDIAN INSTITUTE OF TECHNOLOGY(ISM), DHANBAD

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

TO WHOM IT MAY CONCERN

This is to certify that the project titled “**Traffic sign recognition and classification using convolutional neural nets”** was successfully developed by **Syed Danish Haider**(**2013JE0124**) under the guidance of Department of Electronics and Communication Engineering, Indian Institute of Technology (ISM), Dhanbad in 8th Semester of Bachelor of Technology of Academic year 2016-2017. This work is accepted for consideration towards partial fulfillment of the requirements for the Degree of Bachelor of Technology in Electronics and Communication Engineering.

Dr. Debjani Mitra
Head of Department
Department of Electronics
and communication Engineering
Indian Institute of Technology
Dhanbad

Dr. Debjani Mitra
Head of Department
Department of Electronics
and communication Engineering
Indian Institute of Technology
Dhanbad

ACKNOWLEDGEMENT

I would like to express the deepest appreciation to my project mentor **Dr. Debjani Mitra**, who has the attribute and the substance of a genius. She continually and convincingly conveyed a spirit of adventure in regard to the project, and an excitement in regard to the teaching. Without her guidance and persistent help this project would not have been possible. Finally, we must say that no height is ever achieved without some sacrifices made at some end and it is here where we owe our special department to our parents and our friends for showing their generous love and care throughout the entire period of time. Finally, I thank each and every one who helped me to complete my project work with their cordial support.

Syed Danish Haider
Adm No. 2013JE0124
Department of Electronics and communication Engineering,
Indian Institute of Technology
Dhanbad

CHAPTER 1: INTRODUCTION

Traffic sign recognition(TSR) has been a major problem faced by computer vision experts. We have been dreaming about smart vehicles. Cracking the problem of TSR would give us a major leap in this direction. Pioneer companies like Google, Tesla, Apple and Uber are working on this problem and we could soon see smart cars roaming on the streets.

One of the significant step in computer vision are Convolutional neural nets which were introduced in ImageNet challenge in the year 2012. Traffic sign detection and classification could be taken as a combined problem where we have to detect where is the traffic sign in a picture and what does that particular sign represent. A large number of data sets like German Traffic Sign benchmark, LISA dataset etc. which have been released are a major step in this direction.

Before convolutional neural networks, human crafted techniques were used for image segmentation and then classifying the traffic sign. This task was more of a hit and trial approach and was depended highly on the specific type of image. Convolutional neural networks are more robust in their approach as they extract the features of the image and then use these features to comment about the traffic sign that is in the picture. The specific architecture of CNNs is reason of its very high accuracy in computer vision tasks.

Applying deep learning to the traffic recognition task would give us better results than the conventional methods. The problem statement is not restricted to the recognition and classification of traffic signs but could be used to detect and classify other objects with the required changes. The similar kind of strategy was used in detecting fishes in a Kaggle competition and the results were pretty much good.

If implemented the system would give us a bounding box where the traffic sign is located and what type of sign it is.

PROBLEM DEFINITION

The inspiration of the topic came from the HackFest’17 in which a similar kind of problem statement was given by the company. The basic problem statement deals with classification and recognition of traffic signs from the input images.

The recognition part deals with where is the traffic sign located in an image and classification part deals with telling what sign is present in that image.

The state-of-the-art method is the convolutional neural networks which is being used in here. An architecture using selective searching approach has been used in this thesis for recognition of traffic signs which is more efficient than normal window searching approach. While doing traffic sign recognition the major problem which comes is that the network sometimes detect the traffic signs on the opposite road. It might also happen that due to less data the network starts making bias decisions which are towards the end of the road. This could be controlled up to some extent by cropping the image, attention modelling and bringing in sufficient data for the classification task.

The final goal would be to take in the images and detecting traffic signs present in it. Here we have detected only one traffic sign per image. The work following this would consist of extracting multiple traffic signs from an image.

CHAPTER 2: SURVEY OF TRAFFIC SIGN RECOGNITION PROBLEM

2.1 LITERATURE REVIEW

With the accelerating process of modernization and the increasing number of car ownership, road traffic safety has become more and more crucial around the world, especially in the developed countries. In fact, there are considerable amount of people who lose their lives due to traffic accidents every year.
Thus the area of Traffic Sign Recognition (TSR) systems has been met with growing research interest in the past decade, but the task of recognizing American signs is fairly unexplored so far. TSR is a task with various well defined applications:

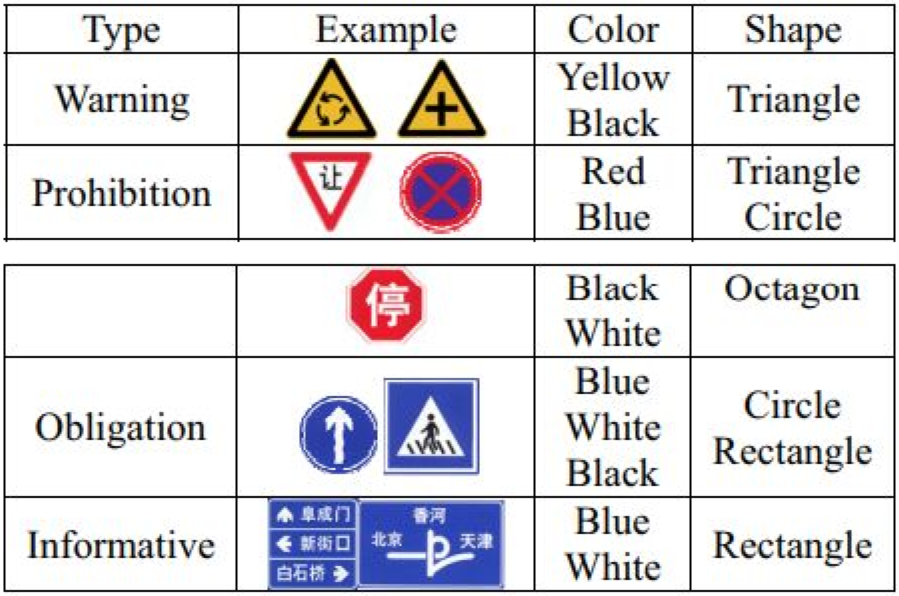
1. **Highway maintenance:** Check the presence and condition of signs along major roads.
2. **Sign inventory:** Similarly to the above task, creating an inventory of signs in city environments.
3. **Driver support systems:** Assist the driver by informing of current restrictions, limits, and warnings.
4. **Intelligent autonomous vehicles:** Any autonomous car that is to drive on public roads must have a means of obtaining the current traffic regulations. This can be done through TSR.

2.2 TRAFFIC SIGN DESCRIPTION

There are various types of traffic signs that contains a lot of information of current traffic environment. They are designed to regulate flow of the vehicles, to indicate the danger and difficulties around the drivers, to issue warnings to them, and to help them navigate well, and thus make the driving safe and convenient.

There exist two basic groups of traffic signs: ideogram-based and text-based signs. While the first group uses simple ideographs to express the sign meaning, the second one contains texts, arrows and other symbols. Most existing research work focuses on the first group of traffic signs.

The most essential types of traffic signs are warning, prohibition, obligation and informative signs as shown in table 2.1 [2.1]. Generally speaking, the traffic signs are well-designed, by using particular colors (yellow, red, blue, white and black) and shapes (triangle, rectangle, circle and octagon), to attract the driver's attention against the natural environment for the purpose of benefiting the problem of traffic sign recognition.


让
停
阜成门
新街口
白石桥
香河
北京
天津

Type	Example	Color	Shape
Warning		Yellow Black	Triangle
Prohibition		Red Blue	Triangle Circle
		Black White	Octagon
Obligation		Blue White Black	Circle Rectangle
Informative		Blue White	Rectangle

Table 2.1 COLOR AND SHAPE OF TRAFFIC SIGNS IN CHINA

2.3 RELATED WORK

It has been more than 20 years since the first idea about traffic sign recognition came out, and some significant progress has been made. In earlier study, the researchers mostly focus on one class of traffic signs (circular sign is the most concerned) or single frame images, moreover, some assumptions are made in order to simplify the problem. Nowadays, with the development of relevant technologies such as image processing and deep learning methods , a system capable of fast, accurate, automatic recognition of traffic signs in various conditions is demanded.

CHAPTER 3: BACKGROUND

This chapter explains fundamental concepts that are necessary for the reader to understand this thesis. It also presents existing tools and solutions related to Traffic Sign Recognition problem

3.1 TRAFFIC SIGN RECOGNITION

Road traffic assumes a major importance in modern society organization. To ensure that vehicular circulation flows in a harmonious and safe way, specific rules are established by every government which includes traffic signs that are displayed to the driver. This solution seems to be simple but sometimes the driver misses signs, which may result in road accidents. Modern vehicles already include many safety systems, but even two cars moving under the speed limit, their collision may lead to disastrous effect.

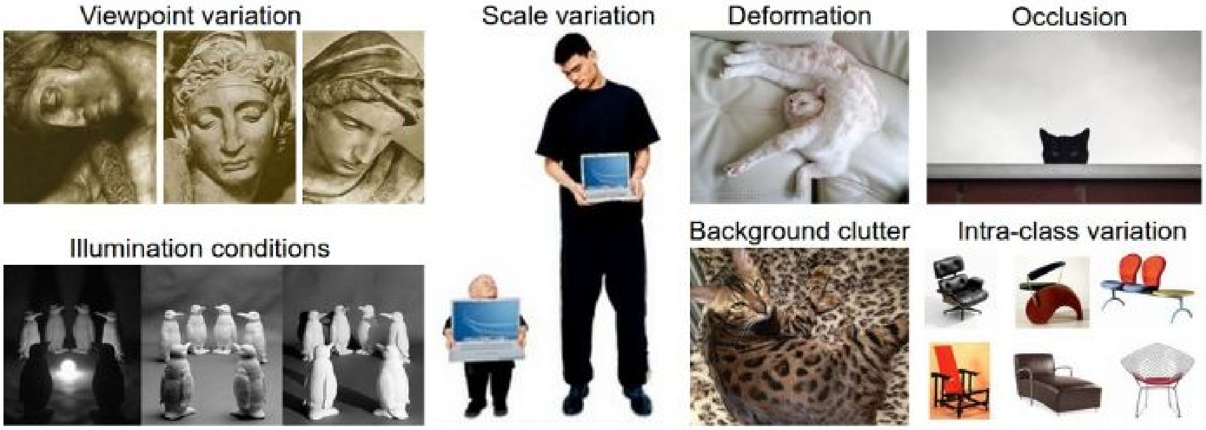
Although some drivers intentionally break the law not respecting traffic signs, an automatic system able to detect these signs can be a useful help to most drivers. One might consider a system taking advantage of the Global Positioning System (GPS). It could be almost flawless if an updated traffic sign location database would be available. Unfortunately, few cars have GPS installed and traffic sign localization databases are not available for download. Installing a low price "traffic sign information" receiver on a car could also be a good idea if traffic signs were able to transmit their information to cars. But, such system would be impractical, requiring a transmitter on each traffic sign.

In this thesis we focused on a system exploiting the already available visual feed through the camera installed on the car dashboard to detect and classify the traffic signs, we have tuned our system for US traffic signs, more specifically.

3.2 METHODS FOR FEATURE EXTRACTION

Feature extraction is an approach widely studied and applied to image recognition problems, it is a better representation of an image than raw pixel value and it is feed to machine learning algorithms to recognize a certain category of objects. There are various challenges that are faced during feature extraction and classification as listed below, keeping in mind the raw representation of images as a 3-D array of brightness values:

1. **Viewpoint variation**. A single instance of an object can be oriented in many ways with respect to the camera.
2. **Scale variation**. Visual classes often exhibit variation in their size (size in the real world, not only in terms of their extent in the image).
3. **Deformation**. Many objects of interest are not rigid bodies and can be deformed in extreme ways.
4. **Occlusion**. The objects of interest can be occluded. Sometimes only a small portion of an object (as little as few pixels) could be visible.
5. **Illumination conditions**. The effects of illumination are drastic on the pixel level.
6. **Background clutter**. The objects of interest may blend into their environment, making them hard to identify.
7. **Intra-class variation**. The classes of interest can often be relatively broad, such as chair. There are many different types of these objects, each with their own appearance.



A good image classification model must be invariant to the cross product of all these variations, while simultaneously retaining sensitivity to the inter-class variations. Different ways of extracting features will surely generate different recognition result, so we will go through relative works in this field.

3.2.1 NEAREST NEIGHBOR CLUSTERING

The principle behind nearest neighbor methods is to find a predefined number of training samples closest in distance to the new point, and predict the label from these. The number of samples can be a user-defined constant (k-nearest neighbor learning), or vary based on the local density of points (radius-based neighbor learning). The distance can, in general, be any metric measure: standard Euclidean distance is the most common choice. Neighbors-based methods are known as *non-generalizing* machine learning methods, since they simply “remember” all of its training data.

Despite its simplicity, nearest neighbors has been successful in a large number of classification and regression problems, including handwritten digits or satellite image scenes. Being a non-parametric method, it is often successful in classification situations where the decision boundary is very irregular.

3.2.2 GIST FEATURE EXTRACTION

This method of feature extraction is widely used in the field of computer vision, its approach is to the represent the mechanism of scene gist understanding, based on scene-centered, rather than object-centered primitives as mentioned in this paper[3.1].

3.2.3 SIFT FEATURE EXTRACTION

For the classification part of the task, a simple nearest neighbor classifier is applied and all the different feature descriptors were put to test for various illumination changes, scale variance, and image video classification. The result shows that the SIFT and color SIFT descriptor perform much better than histogram-based descriptor with much higher discriminative power. OpponentSIFT gives back the best recognition result among SIFT descriptors, but other SIFT descriptors may perform slighter better in other aspects depending on the circumstance.

3.3 CNN FOR TRAFFIC SIGN RECOGNITION

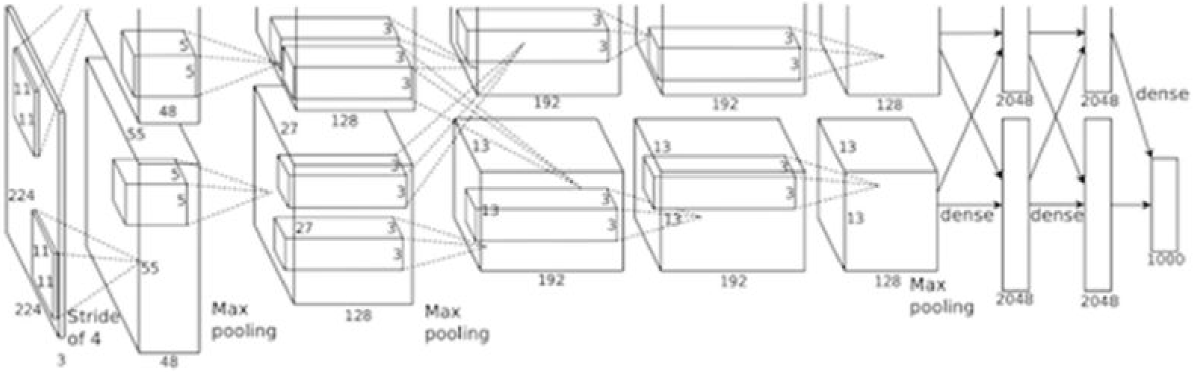
In recent years nearly all the breakthroughs in the field of Image recognition is due to the use of convolution layer in neural networks, with the advancement in technology of GPU computation it is possible to create and deploy deeper neural networks on large databases. Thus, going through the development of convolutional neural network is must.

The very idea of Convolutional Neural Network was introduced in the early 80s, it follows the discovery of visual mechanisms in living organisms. It is a form of feed forward artificial neural network where each individual neuron responds to an overlapping region in the visual field before its own layer.

3.3.1 AlexNet (2012)

The one that started it all (Though some may say that Yann LeCun’s paper[3.2] in 1998 was the real pioneering publication). This paper, titled “ImageNet Classification with Deep Convolutional Networks”, has been cited a total of 6,184 times and is widely regarded as one of the most influential publications in the field. Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton created a “large, deep convolutional neural network” that was used to win the 2012 ILSVRC (ImageNet Large-Scale Visual Recognition Challenge). 2012 marked the first year where a CNN was used to achieve a top 5 test error rate of 15.4% (Top 5 error is the rate at which, given an image, the model does not output the correct label with its top 5 predictions). The next best entry achieved an error of 26.2%, which was an astounding improvement that pretty much shocked the computer vision community. Safe to say, CNNs became household names in the competition from then on out.

In the paper[3.3], the group discussed the architecture of the network (which was called AlexNet). They used a relatively simple layout, compared to modern architectures. The network was made up of 5 conv layers, max-pooling layers, dropout layers, and 3 fully connected layers. The network they designed was used for classification with 1000 possible categories.

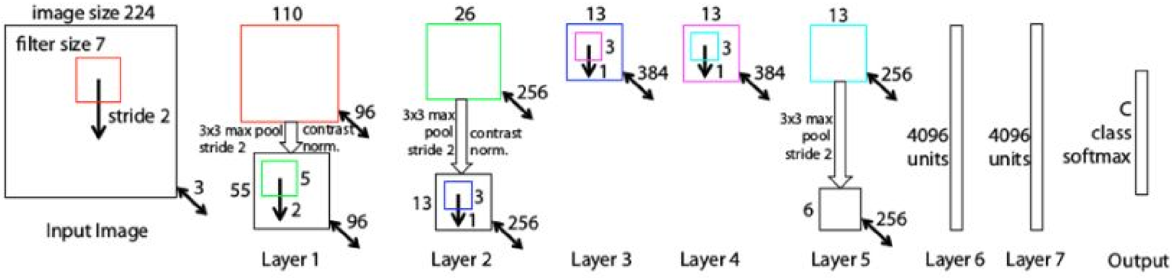


AlexNet architecture (May look weird because there are two different "streams". This is because the training process was so computationally expensive that they had to split the training onto 2 GPUs)

The neural network developed by Krizhevsky, Sutskever, and Hinton in 2012 was the coming out party for CNNs in the computer vision community. This was the first time a model performed so well on a historically difficult ImageNet dataset. Utilizing techniques that are still used today, such as data augmentation and dropout, this paper really illustrated the benefits of CNNs and backed them up with record breaking performance in the competition.

3.3.2 ZF Net (2013)

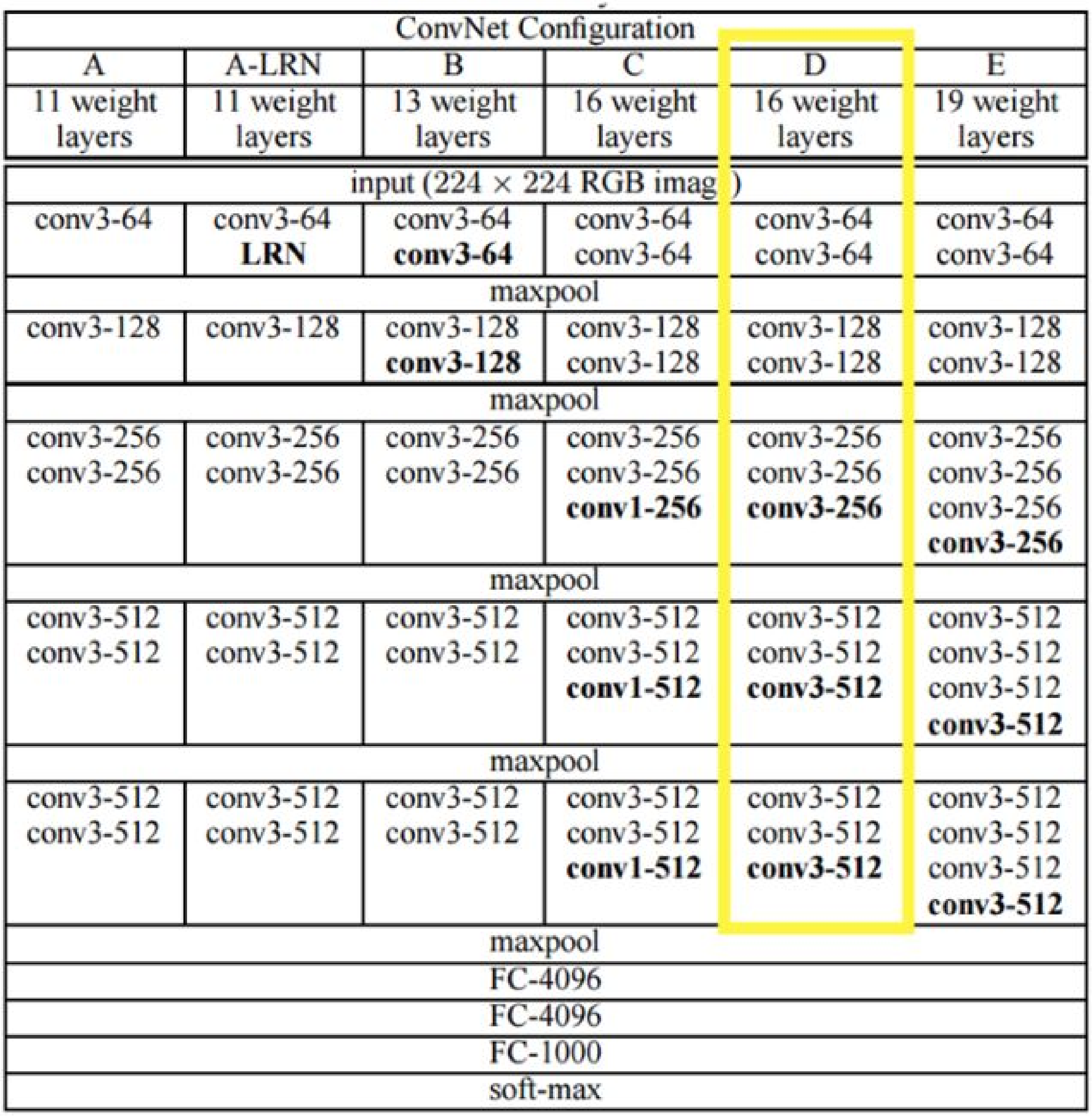
With AlexNet stealing the show in 2012, there was a large increase in the number of CNN models submitted to ILSVRC 2013. The winner of the competition that year was a network built by Matthew Zeiler and Rob Fergus from NYU. Named ZF Net [3.4], this model achieved an 11.2% error rate. This architecture was more of a fine tuning to the previous AlexNet structure, but still developed some very keys ideas about improving performance. Another reason this was such a great paper is that the authors spent a good amount of time explaining a lot of the intuition behind ConvNets and showing how to visualize the filters and weights correctly. In this paper titled “Visualizing and Understanding Convolutional Neural Networks”, Zeiler and Fergus begin by discussing the idea that this renewed interest in CNNs is due to the accessibility of large training sets and increased computational power with the usage of GPUs. They also talk about the limited knowledge that researchers had on inner mechanisms of these models, saying that without this insight, the “development of better models is reduced to trial and error”. While we do currently have a better understanding than 3 years ago, this still remains an issue for a lot of researchers! The main contributions of this paper are details of a slightly modified AlexNet model and a very interesting way of visualizing feature maps.



ZF Net Architecture

3.3.3 VGG Net (2014)

Simplicity and depth. That's what a model created in 2014 (weren't the winners of ILSVRC 2014) best utilized with its 7.3% error rate. Karen Simonyan and Andrew Zisserman of the University of Oxford created a 19 layer CNN that strictly used 3x3 filters with stride and pad of 1, along with 2x2 maxpooling layers with stride 2.



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

The 6 different architecures of VGG Net. Configuration D produced the best results

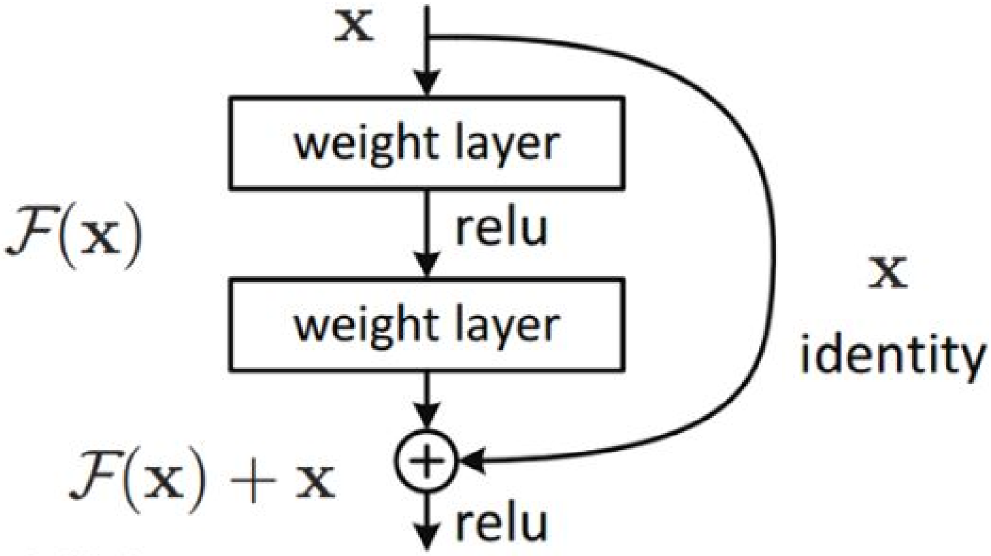
The use of only 3x3 sized filters is quite different from AlexNet's 11x11 filters in the first layer and ZF Net's 7x7 filters. The authors' reasoning is that the combination of two 3x3 conv layers has an effective receptive field of 5x5. This in turn simulates a larger filter while keeping the benefits of smaller filter sizes. One of the benefits is a decrease in the number of parameters. Also, with two conv layers, we're able to use two ReLU layers instead of one. As the spatial size of the input volumes at each layer decrease (result of the conv and pool layers), the depth of the volumes increase due to the increased number of filters as you go down the network.

VGG Net is one of the most influential papers in my mind because it reinforced the notion that convolutional neural networks have to have a deep network of layers in order for this hierarchical representation of visual data to work. Keep it deep. Keep it simple.

3.3.4 Microsoft ResNet (2015)

Imagine a deep CNN architecture. Take that, double the number of layers, add a couple more, and it still probably isn't as deep as the ResNet architecture[3.5] that Microsoft Research Asia came up with in late 2015. ResNet is a new 152 layer network architecture that set new records in classification, detection, and localization through one incredible architecture. Aside from the new record in terms of number of layers, ResNet won ILSVRC 2015 with an incredible error rate of 3.6%.

The idea behind a residual block is that you have your input x go through conv-relu-conv series. This will give you some F(x). That result is then added to the original input x. Let's call that H(x) = F(x) + x. In traditional CNNs, your H(x) would just be equal to F(x) right? So, instead of just computing that transformation (straight from x to F(x)), we're computing the term that you have to add, F(x), to your input, x. Basically, the mini module shown below is computing a "delta" or a slight change to the original input x to get a slightly altered representation (When we think of traditional CNNs, we go from x to F(x) which is a completely new representation that doesn't keep any information about the original x). The authors believe that “it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping”.


x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

A residual block

Another reason for why this residual block might be effective is that during the backward pass of back propagation, the gradient will flow easily through the graph because we have addition operations, which distributes the gradient.

3.4 COMPARISON

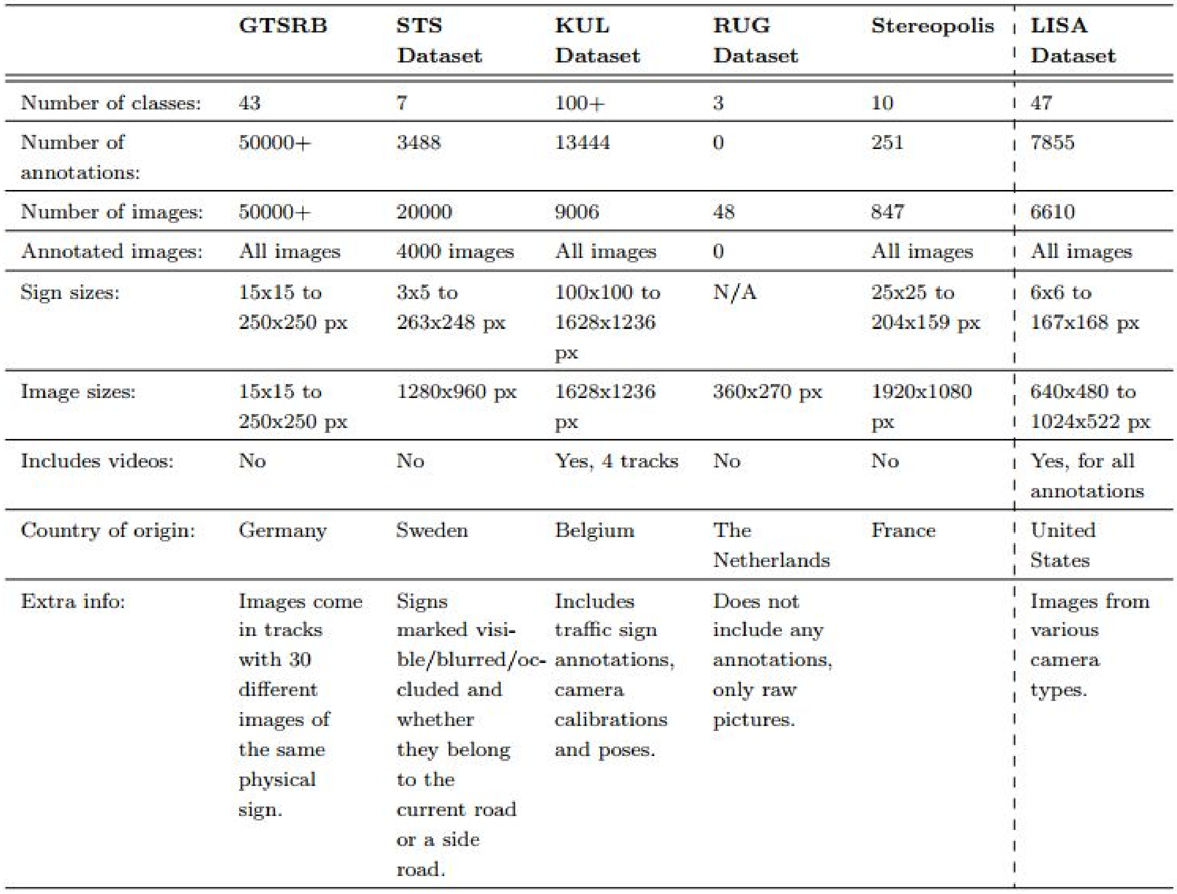
After a thorough examination of related research in the field of image recognition and segmentation, the conclusion we reached is that the Deep Learning methods in particular convolution neural network is going to be applied to this thesis task. The main reason is that cutting edge CNNs are generating much better classification result than any other methods developed in the past, also it has been proven that learning features automatically and directly from pixel images can replace handcrafted feature extractors.

3.5 TRAFFIC SIGN DATABASES

There are few publicly available datasets present :

1. German Traffic Sign Recognition Benchmark (GTSRB) [3.6]
2. KUL Belgium Traffic Signs Dataset (KUL Dataset) [3.7]
3. Swedish Traffic Signs Dataset (STS Dataset) [3.8]
4. RUG Traffic Sign Image Database (RUG Dataset) [3.9]
5. Stereopolis Database [3.10]
6. LISA traffic sign Dataset

LISA Dataset is used in this thesis, a comparison between the above mentioned datasets are given in this paper[3.11] is as shown below in the table.



	GTSRB	STS Dataset	KUL Dataset	RUG Dataset	Stereopolis	LISA Dataset
Number of classes:	43	7	100+	3	10	47
Number of annotations:	50000+	3488	13444	0	251	7855
Number of images:	50000+	20000	9006	48	847	6610
Annotated images:	All images	4000 images	All images	0	All images	All images
Sign sizes:	15x15 to 250x250 px	3x5 to 263x248 px	100x100 to 1628x1236 px	N/A	25x25 to 204x159 px	6x6 to 167x168 px
Image sizes:	15x15 to 250x250 px	1280x960 px	1628x1236 px	360x270 px	1920x1080 px	640x480 to 1024x522 px
Includes videos:	No	No	Yes, 4 tracks	No	No	Yes, for all annotations
Country of origin:	Germany	Sweden	Belgium	The Netherlands	France	United States
Extra info:	Images come in tracks with 30 different images of the same physical sign.	Signs marked visible/blurred/occluded and whether they belong to the current road or a side road.	Includes traffic sign annotations, camera calibrations and poses.	Does not include any annotations, only raw pictures.		Images from various camera types.

Information on the publicly available sign databases.

3.6 DEEP LEARNING FRAMEWORK

There are various deep learning frameworks available having various pros and cons.

- **TensorFlow** is the most popular one, TensorFlow is an open source software library for numerical computation using data flow graphs. Tensorflow supports Python and C++, along to allow computing distribution among CPU, GPU (many simultaneous) and even horizontal scaling using gRPC. There is a lot of code to write, and you need to reinvent the wheel over and over again.
- **Theano** is low-level library, following Tensorflow style. And as it, it its not properly for Deep Learning, but for numerical computations optimization. It allows automatic function gradient computations, which together with its Python interface and it’s integration with Numpy, made this library in it’s beginning in one of the most used for general purpose Deep Learning.
- **Keras** is a very high-level library that works on top of Theano or Tensorflow (it’s configurable). Also, Keras reinforces minimalism, you can build a Neural Network in just a few lines of code as it is modular.
- **Lasagne** has emerged as a library that works on top of Theano. Its mission was to abstract a bit the complex computation underlying to Deep Learning algorithms and also provide a more friendly interface but it is losing speed in favor of Keras as it rose a little bit ago.
- **Caffe** is one of the most veteran frameworks, it focuses only in computer vision, but it does it really well. The drawbacks are that it’s not flexible. If you want to introduce new changes you need to program in C++ and CUDA, but for less novel changes you can use its Python or Matlab interfaces. One of it’s bigger drawbacks is its installation. It has a lot of dependencies to solve but as a tool for put in production computer vision systems is the undisputed leader. It’s robust and very fast.

CHAPTER 4: METHODOLOGY

This chapter deals with the whole pipelining which is used in the thesis. It will consist of the details of training and making the network. Also, along with the architectures used, and the data preparation strategy that is being used in the project has also been discussed here.

4.1 A DETAIL ABOUT CNNs :

Here a close look on the CNN architecture has been made. The inspiration behind this particular architecture starting right from ANNs has been made here.

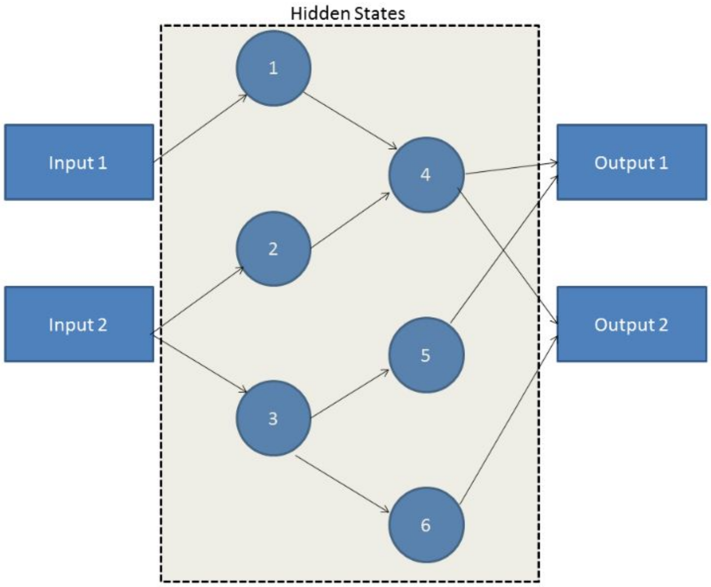
4.1.1 BASIC ANNs

The working of ANN takes its roots from the neural network residing in human brain. ANN operates on something referred to as Hidden State. These hidden states are similar to neurons. Each of these hidden state is a transient form which has a probabilistic behavior. A grid of such hidden state act as a bridge between the input and the output. A typical structure of an ANN (Figure 3.1) is to put layers of hidden neurons between input data and output data, neurons of different layers are linked together by weights. By constantly feeding the network with input data and output targets we update the weight vector, and this is called training of the network.

Here, output of the network is a function of the inputs and the weights of hidden and input layers. This could be mathematically represented as:

$$Y = XW' = sum(X.W)$$

Where Y is the output vector, X is the input vector and W is the weight vector, sum() function denotes the sum of n values.



4.1.2 The learning algorithm:

A lots of learning algorithms have been developed so far, but gradient descent remains to be one of the top most algorithm that is used in neural networks for training. It has a simple intuitional structure which makes it easy to implement and fast. In gradient descent we compute the error between the output for certain input variables, i.e. *Y = f(X,W)* where X is a vector of inputs and W are the weights, with the actual output *t,* i.e the target. Then, this error *E* is minimised using differential calculus approach. Here,

$$E = D(t,f(X,W))$$

In this function X and t will be obtained from the input data and W is the only variable.

Here we have an error function Etr associated with training and Eval associated with validation set and Etest associated with the test error. Now our task is to update W so as to minimize Etr

The major trade off that we face here is between training and test error. At the end, we have to make the final calculation on the basis of Etest. While training we should be careful not to have a large difference between Etr and Etest which give rises to the problem of overfitting the the deep convolutional nets. The relationship between these two errors have been studied by Yann LeCun in 4.1 The second problem which arises in deep neural nets is that calculation of the derivatives is highly computation intensive. As in a normal deep neural net there are millions of parameters(weights), so it is required to differentiate with respect to individual weights. To tackle this problem of intensive computation we use backpropagation algorithm.

4.1.3 Update Weight Matrix:

We have already established before that it is possible to adjust weight matrix to minimize the error between target value and the output of last neuron layer, here we are going through how it is done and how some of the parameters can affect the network.

Back Propagation starts at the output layer with the calculation of error in Equation 1, the update of the weight matrix follows the rule of Equation 2.

$$E = Y (1 - Y)(t - Y)$$
$$wij = w'ij + \alpha .E.X$$

For the ith input of the jth neuron in the output layer, the weight wij is adjusted by adding to the previous weight value, w'ij , a term determined by the product of a learning rate, α , an error term, ej , and the value of the ith input, Xi .
The error vector, E , is determined by the product of the actual output, Y , its complement, 1 − Y , and the difference between the desired output, T , and the actual output. Once the output layer is updated, we will keep going back and adjust the next layer back.

One more thing to keep in mind is the choice of learning rate. A bigger learning rate gives a faster training process adjustment but make the network unstable and sometimes leads to no convergence. On the contrary, a smaller learning rate may guarantee a convergence but it might just be too slow and too long time waiting according to the research of 4.2.

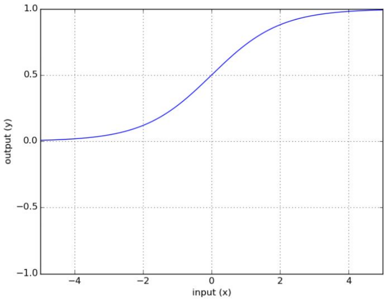
4.2 DIFFERENT ACTIVATION FUNCTIONS:

Below are some of the most popular activation function used in Deep Neural Networks.

Hyperbolic Tangent Function

The hyperbolic tangent activation function is in the form of , it is a continuous activation function with rotational symmetric property, which means that it produce negative value for negative input and positive value for positive input. It is also a saturating nonlinear activation function.

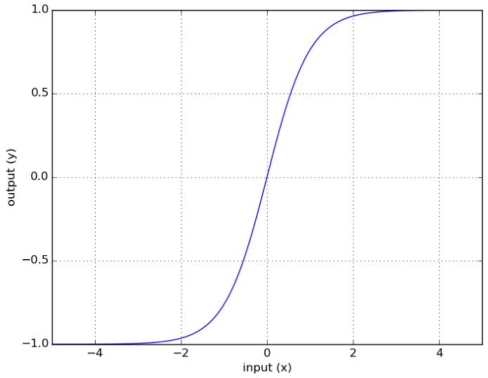
f(x) = tanh(x)



Sigmoid Activation Function

The Sigmoid activation function is in the form of , it is a continuous activation function that is also a saturating nonlinear activation function. The output can not be negative .

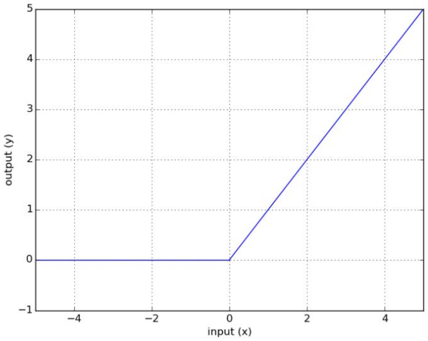
f(x) = 1 (1 + e−x)



Rectified Linear Unit (ReLU)

Comparing with the saturating nonlinearities listed above, ReLU is a non-saturating nonlinearity that is much faster in terms of training with gradient descent. Introduced by Krizhevsky in [15], ReLU nonlinearity is used on CIFAR-10 dataset for a particular four-layer convolutional network. It took takes tanh units 35 epochs to reach 25 error rate and only 5 epochs for a ReLU activation. Faster training actually makes it possible to train bigger and deeper networks in the future.

$$f(x) = \max(0, x).$$



4.3 CONVOLUTIONAL LAYER & SUB-SAMPLING METHOD

A convolutional layer is fundamentally different from a Multilayer Perception since there is no full connection between the input and output of the same layer, this is also called sparse connectivity. This sparse connection exploit the spatial correlation by only having connection between neurons of adjacent layers [32]. By doing this the neuron in the convolutional layer only respond to a subset of neurons in previous layer and thus can only be activated by this subset neurons. Another important point to make is that the convolutional filter in a typical CNN is replicated across the entire visual field, which means that both the weight vector and the bias remain the same to form one feature map. The reason we replicate the filter is to make sure that the we can extract the same feature no matter what position it is extracted. Also by replicating the filter, we minimize the number of parameters that we have to learn and in turn make the learning easier. Once a feature map is drawn, the global location of that feature is not as important, but rather the relative location of each detected features is of concern here. Take a detected 90 degree corner for example, four of those corners indicates a square but the global position of these corners are irrelevant, sometimes this precise position can be harmful to a classification. It is out of this concern that we do sub-sampling of feature map to reduce the precision of the global position and in turn reduce the spatial resolution of the feature map. In the study by [17], they discover that a sub-sampling layer performs a local averaging and a sub-sampling. It reduces the resolution of the feature map and reduces the sensitivity of the output to shift and distortions. A typical subsampling layer has a fixed receptive field like in the paper of [6, 13, 18] . Like the famous LeNet-5 network has a 2×2 kernel size, each unit computes the average of the four inputs and multiply it with a trainable coefficient then add a trainable bias. At this stage the pooling is non-overlapping, which means the feature map after the pooling layer has half the rows and columns. With the sub-sampling after different convolutional layers, the resolution decreases thus we can extract more feature maps in a certain layer. Recently a technique of overlapping pooling is gaining popularity due to its ability of preventing over-fitting. Here are two key parameters in play, the pooling kernel size z and the kernel stride s. The kernel size indicates how many inputs are going into the pooling process and the stride means how far away two pooling units. if s = z then it is our regular pooling, but if s < z, then it is overlapping pooling

4.4 BATTLE WITH OVERFITTING

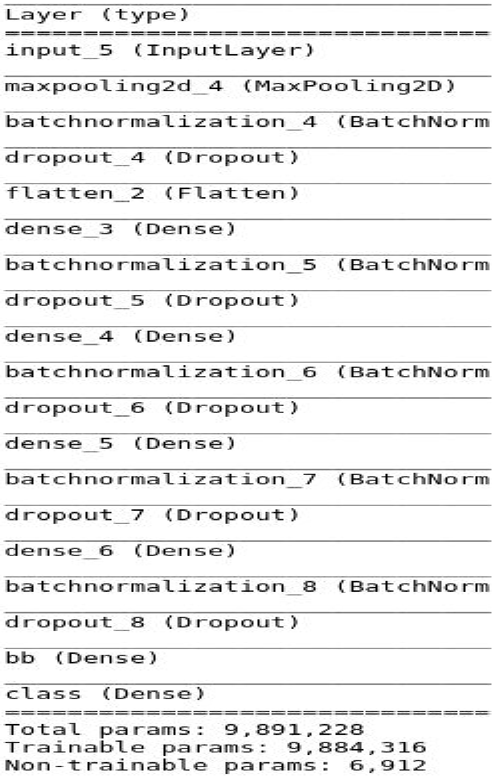
One of the biggest problem when training deep artificial neural networks is the tendency of overfitting. Overfitting happens when the network is excessively complex by either having too many parameters or iterating for too many circles. It is well illustrated in the Figure 3.5 that the more complicated the network gets, the bigger a difference there is between the training error and the testing error. It so happens that a deep convolutional network is very complicated with many parameters, so logically it is vulnerable to overfitting, thus weakens its ability of generalization. A lot of methods are proposed with the development of CNN to battle overfitting, for example adding random noise[20], Dropout [12], performing data augmentation [15, 28, 7], performing dimension reduction [3, 31] and cross validation. Data augmentation might be the most intuitive way of avoiding overfitting since by enriching the dataset, the training image can better represent the true pattern of each class, thus in turn gives back a better generalization result. In the research done by [15], they propose a way of data augmentation that require no extra disk space and no extra computational time by CPU code generating image when GPU is running the last batch training. First, random patches of the size 224×224 is extracted from the original 256×256 image, then translation and reflection is done to make more of the dataset. Secondly, alternating the intensities of RGB channels in training image. In short, each time a hidden neuron has a 0.5 possibility of getting activated, and those ones that is activated is going to contribute to forward pass and back propagation. In this way each time a input pattern is present, a new architecture of the network is sampled but at the same time they share weights. This technique reduce the correlation between neurons, thus gives better performance on overfitting. Another technique is carried out in the paper [20], they propose to corrupt training examples with noise from known distributions within the exponential family and present a novel learning algorithm, called marginalized corrupted features (MCF), that trains robust predictors by minimizing the expected value of the loss function under the corrupting distribution, essentially learning with finitely many (corrupted) training examples.

4.5 LEARNING IN DETAIL

4.5.1 Overall Architecture

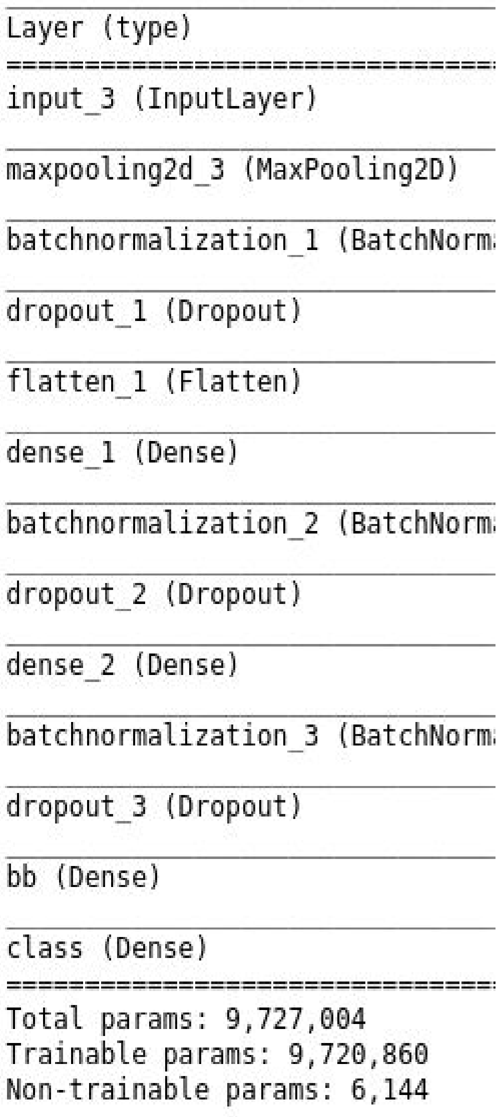
Here two cascading architectures have been used for the recognition and detection of the traffic sign. We firstly use a VGGNet block for extracting features and then use these features to extract the required traffic sign and the bounding box around it.

The overall architecture is as,



```
________________________________
Layer (type)
================================
input_5 (InputLayer)
________________________________
maxpooling2d_4 (MaxPooling2D)
________________________________
batchnormalization_4 (BatchNorm
________________________________
dropout_4 (Dropout)
________________________________
flatten_2 (Flatten)
________________________________
dense_3 (Dense)
________________________________
batchnormalization_5 (BatchNorm
________________________________
dropout_5 (Dropout)
________________________________
dense_4 (Dense)
________________________________
batchnormalization_6 (BatchNorm
________________________________
dropout_6 (Dropout)
________________________________
dense_5 (Dense)
________________________________
batchnormalization_7 (BatchNorm
________________________________
dropout_7 (Dropout)
________________________________
dense_6 (Dense)
________________________________
batchnormalization_8 (BatchNorm
________________________________
dropout_8 (Dropout)
________________________________
bb (Dense)
________________________________
class (Dense)
================================
Total params: 9,891,228
Trainable params: 9,884,316
Non-trainable params: 6,912
```

Our second approach has used Resnet features, here we have placed the classification block and the bounding box block right after the dense layers which induces an effect of attention modeling in the network. The architecture is shown below:



```
________________________________
Layer (type)
================================
input_3 (InputLayer)
________________________________
maxpooling2d_3 (MaxPooling2D)
________________________________
batchnormalization_1 (BatchNorm
________________________________
dropout_1 (Dropout)
________________________________
flatten_1 (Flatten)
________________________________
dense_1 (Dense)
________________________________
batchnormalization_2 (BatchNorm
________________________________
dropout_2 (Dropout)
________________________________
dense_2 (Dense)
________________________________
batchnormalization_3 (BatchNorm
________________________________
dropout_3 (Dropout)
________________________________
bb (Dense)
________________________________
class (Dense)
================================
Total params: 9,727,004
Trainable params: 9,720,860
Non-trainable params: 6,144
```

CHAPTER 5: RESULT AND ANALYSIS

This chapter represents the comparative results of the two structures based on VGGNet and RESNet, both of them provides satisfactory results.

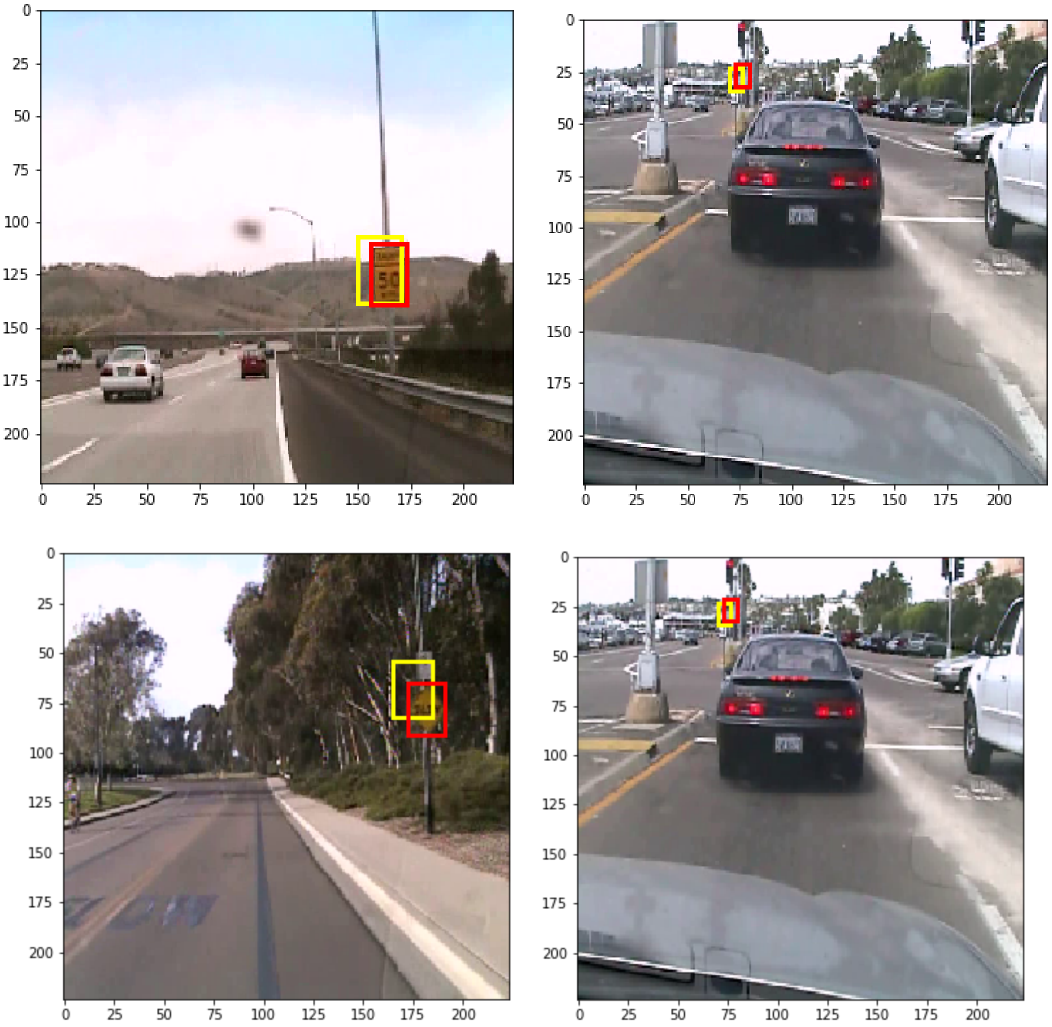
5.1 CLASSIFICATION RESULTS

VGGNet provides a training accuracy of 99.42% and a validation accuracy of 98.60% with a log loss error of 0.0810 whereas RESNet provides a training accuracy of 99.65% and a validation accuracy of 97.90% with a log loss error of 0.1018 thus we can say that for our validation data set VGGNet model have better generalizing property for classification.

5.2 SEGMENTATION RESULTS

VGGNet provides a training mse loss on the bounding box of 0.1965 and validation mse loss of 57.2074 whereas RESNet provides a training mse loss on the bounding box of 0.1220
and validation mse loss of 96.6257, thus VGGNet performing better for both the task

Below we can see some good segmentation results :



Good Segmentation Results

5.3 CONCLUSION

This thesis provides an overview of the state of sign detection. Instead of treating the entire TSR flow, focus has been solely on the detection of signs. During recent years, a large effort has gone into TSR, mainly from Europe, Japan, and Australia and the developments have been described. The detection process has been split into segmentation, feature extraction, and detection. Many segmentation approaches exist, mostly based on evaluating colors in various color spaces. For features there are also a wealth of options. The choice is made in conjunction with the choice of detection method. By far the most popular features are edges and gradients, but other options such as HOG and Haar wavelets have been investigated. The detection stage is dominated by the Hough transform and its derivatives, but for HOG and Haar wavelet features, SVMs, neural networks, and cascaded classifiers have also been used. Arguably, the biggest issue with sign detection as it is currently is the lack of use of public image databases to train and test systems. Currently, every new approach presented uses a new dataset for testing, making comparisons between papers hard. This gives the TSR effort a somewhat scattered look. Recently, a few databases have been made available, but they are still not widely used, and cover only Vienna Convention compliant signs. We have contributed with a new database, the LISA Dataset, which contains US traffic signs. This issue leads to the main unanswered question in sign detection: Is a model based shape detector superior to a learned approach, or vice versa? Systems using both approaches exist, but are hard to compare, since they all use different data sets.

REFERENCES

[2.1] MENG-YIN FU, YUAN-SHUI HUANG.
A SURVEY OF TRAFFIC SIGN RECOGNITION

[3.1] Aude Oliva, and Antonio Torralba.
Building the gist of a scene: the role of global image features in recognition

[3.2] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner.
Gradient Based Learning Applied to Document Recognition

[3.3] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton.
ImageNet Classification with Deep Convolutional Neural Networks

[3.4] Matthew D. Zeiler, Rob Fergus
Visualizing and Understanding Convolutional Networks.

[3.5] Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun.
Deep Residual Learning for Image Recognition

[3.6] Johannes Stallkamp, Marc Schlipsing, Jan Salmen,Christian Igel.
The German Traffic Sign Recognition Benchmark: A multi-class classification competition

[3.7] Radu Timofte, Karel Zimmermann, Luc Van Gool
Multi-view traffic sign detection, recognition, and 3D localisation

[3.8] Fredrik Larsson and Michael Felsberg
Using Fourier Descriptors and Spatial Models for Traffic Sign Recognition

[3.9] Cosmin Grigorescu, Student Member, IEEE, and Nicolai Petkov
Distance Sets for Shape Filters and Shape Recognition

[3.10] R. Belaroussi, P. Foucher, J.-P. Tarel, B. Soheilian, P. Charbonnier, N. Paparoditis. Road Sign Detection in Images: A Case Study

[4.1] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner.
Gradientbased learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[4.2] Stephen Marsland. Machine learning: an algorithmic perspective. CRC press,2014.

[4.3] Laurens Maaten, Minmin Chen, Stephen Tyree, and Kilian Q Weinberger.
Learning with marginalized corrupted features. In Proceedings of the 30th International Conference on Machine Learning (ICML-13), pages 410–418, 2013.