SIGN BOARD RECOGNITION USING CONVOLUTIONAL NEURAL NETWORK

A PROJECT REPORT

Submitted by

G.MANIDEEP [RA1611004010101]
D.MOUNESH [RA1611004010361]
G.SAI TEJA [RA1611004010536]
V.SRAVAN [RA1611004010541]

Under the guidance of

Dr.C.T.MANIMEGALAI

(Associate Professor, Department of Telecommunication Engineering)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

ELECTRONICS AND COMMUNICATION ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be **University** *u/s 3 of UGC Act, 1956)*

S.R.M. Nagar, Kattankulathur-603203,Kancheepuram District,Tamil Nadu

APRIL 2020

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report titled "**SIGN BOARD RECOGNITION USING CONVOLUTIONAL NEURAL NETWORK**" is the bonafide work of "**G.MANIDEEP[RA1611004010101], D.MOUNESH[RA1611004010361], G.SAI TEJA [RA1611004010536] ,V.SRAVAN [RA1611004010541]** ", who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr.C.T.MANIMEGALAI
SUPERVISOR
Associate Professor
Department of Telecommunication Engineering

SIGNATURE

Dr.T.RAMA RAO,Professor
HEAD OF THE DEPARTMENT
Department of Electronics and Communication Engineering

Signature of the Internal Examiner

Signature of the External Examiner

DECLARATION

We here by declare that the Major Project(15EC496L) entitled “SIGN BOARD RECOGNITION USING CONVOLUTIONAL NEURAL NETWORK” to be submitted for the Degree of Bachelor of Technology is our original work in a team and the dissertation has not formed the basis for the award of any degree, diploma,associateship or fellowship of similar other titles.It has not been submitted to any other University or Institution for the award of any degree or diploma.

Date:

Place:

V.SRAVAN CHOWDARY

G.PENCHALA SAI TEJA

D.MOUNESH

G.MANIDEEP

ABSTRACT

Road injuries are a big drawback in society for a few time currently. Ignoring sign boards while moving on roads has significantly become a major cause for road accidents. Thus we came up with an approach to face this issue by detecting the sign board and recognition of sign board. At this moment there are several deep learning models for object detection using totally different algorithms like RCNN, faster RCNN, SPP-Net, etc. We prefer to use Yolo-3, which improves the speed and precision of object detection. This algorithm will increase the accuracy by utilizing residual units, skip connection and up-sampling. This algorithm uses a framework named Dark-net. This framework is intended specifically to create the neural network for training the Yolo algorithm. To thoroughly detect the sign board, we used this algorithm.

ACKNOWLEDGEMENTS

We wish to convey our profound gratitude to **Dr.T.P.GANESAN**,Pro Vice Chancellor(PD) for bestowing all the facilities and assistance in the implementation of this project.

We are grateful to **Dr.C.MUTHAMIZHCHELVAN**,Director (Engineering and Technology) for the instigation afforded.We are appreciative of **Dr.T.RAMA RAO**,Professor and Head for the Department Of Electronics and Communication Engineering for the assistance and stimulation provided throughout our work.

We whole-heartedly thank our project coordinator and panel members **Dr.P.VIJAY KUMAR, Dr.VIVEK MAIK,Dr.R.DAYANA**,Assistant Professors for their essential tips for betterment in project evaluation reviews.

In these respects,we are grateful to **Dr.C.T.MANIMEGALAI**,Associate Professor who has consistently inspired us offering eternal and worthy advice,guidance,recommendations and feedbacks in this project.

In regard to faculty and non teaching staff of the Department of the Electronics and Communications Engineering,we wish to thank them for their implicit and explicit assistance they have offered throughout our project work.

I would like to thank three other members in my group for helping to finish this project at the right period to achieve the desired outcomes.I always appreciate my parents and siblings for their continuous encouragement and their continuing confidence in my professional ability

V.SRAVAN CHOWDARY

G.PENCHALA SAI TEJA

D.MOUNESH

G.MANIDEEP

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

TSR	Traffic Sign Recognition
ADAS	Advanced Driver Assistance Systems
CNN	Convolution Neural Network
ReLu	Rectified Linear Unit Layer
VGG	Visual Geometry Group
HOG	Oriented Gradient Histogram
BB	Bounding Boxes
IoU	Intersection Over Union
R-CNN	Region-CNN
Yolo	You Only Look Once
mAP	Mean Average Precision
SPP-Net	Spatial Pyramid Pooling-Network
DPM	Deformable Parts Model
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
FPS	Frames Per Second
ITS	Intelligent Transportation Systems

CHAPTER 1

INTRODUCTION

1.1 TRAFFIC SIGN RECOGNITION

Traffic Sign Recognition (TSR) may well be a technology that permits a vehicle to identify traffic signs that ar denote on the trail e.g "speed-limit" or "baby" or "go-ahead".This is one in every of the normally referred to as Advanced Driver Assistance Systems (ADAS) apps.A number of automobile manufacturers are structuring the item.The traffic signs ar followed utilizing image handling techniques.The methods for identifying proof will as a rule be isolated into methods targeting shading, structure and comprehension.Traffic signs will be seen in various modern vehicles, transports and trucks utilizing front homeward cameras.A traffic sign recognizable device is one in every of the first cases in utilization for speed limits.

As per today's situation,if we are concerned about fatalities, road accidents can be contemplated as dominant cause for death. Especially in comparison to deaths related to suicides and injuries, etc., we concede that road accidents are indeed a significant cause of the loss of life, such as crashes, speeding, ignorance of sign boards. Our intention is to reduce deaths associated with ignorance.

Presently there are specific algorithms for the identification and classification of objects. These algorithms include RCNN. F-RCNN, Faster-RCNN. All these algorithms adopt a selective searching method to propose the region boundaries. When an input picture is sent to these algorithms, they define the boundaries and propose a particular number of regions. Once these regions and their boundaries are proposed, they are sent over to a convolutional neural network.

This network extracts the features you from which we generate a feature map. This feature map will be forwarded to the classifier.Classifier then distinguishes the weight-based images, compares them to the dataset extracted features, and finally classifies the objects in the image.But, these algorithms do not take the whole image for detecting objects. They define the regions in an input picture and define boundaries for objects i.e,, they take only specific number of regions say (two thousand).

For this reason these algorithms take a quite long time to test an image.This consequently lowers the speed and rapidness of detection. To improve the precision and rapidness, we tend to use Yolo algorithm.Yolo means You Look Only Once. The greatest advantage of this algorithm is that the whole picture is taken into account. It just uses a convolution neural network to derive features from pre-weights and contrast.

CHAPTER 2

LITERATURE SURVEY

2.1 Basics of Object detection

Traffic-signs have been carefully researched for the tracking and classification of texts and new ways of solving this problem have continued to be developed and developed for images collection technologies. In this research, we have introduced a tool to detect, portion and realize text-based traffic-signs from screening pictures and processing methods.[1]

2.2 Improving Accuracy

In this paper,the authors have made successful predictions by defining search regions, detecting probable road signs and reduce the number of regions based on contextual constraints.[3]

2.3 CNN based YOLO Framework

In this suggested system, they used the CNN-based multi-directional Yolo system in which license plate cars are recognized multidirectionally.In elegant real-time situations, this suggested method will tackle positional problems with particular orientation predictions and fast IoU evaluation. A spree of tests has been carried out to make sure the current strategy exceeds other modern strategies in terms of increased precision and reduced arithmetic cost.[5]

2.4 Detection of text on Sign Boards

In the other paper,in order to disintegrate the original work into two, i.e. the positioning of road signs and recognizing text on sign board, the authors developed a method utilizing divide

conquer strategy. By the use of a feature-tracking method, algorithms for both the tasks are merged into a single model.

This model offers a creative way to identify video text with two dimensional picture parameters, such as colour, edges and texture with a three dimensional geometric shape data of video sequences, in any video frame.[4]

2.5 Building a framework for detection,training

In this paper,authors came up with two phases: 1) detection 2) tracking. Two neural networks are developed during the detection process to derive traffic signal color and shape features from the input data. On the basis of the features obtained, traffic signs are then put in the photos. This approach is built in relation to the methodology of Fuzzy. Traffic signals from the previous process are tracked through image sequences using the Kalman filter during the tracking process.[2]

CHAPTER 3

RESEARCH AND EXISTING METHODOLOGIES

In this sector, a lot of research has already been completed. CNN was used in the classification of the objects to identify the different objects in the image. The CNN approach uses the convolution neural networks that are distributed in the form of layers. These layers must be conditioned by repeated backtracking in order to correct their bias. While this is a really simple explanation process, there's a lot going on inside the workings of neural networks if you create it from scratch. And the accuracy of such a network is not very high and should not be compared with the human brain. Yet that is also a very good prediction relative to other non-networks.

3.1 OBJECT DETECTION

Object recognition may be a procedure technology coupled to pc vision and image process that identifies instances of semiconductors of a specific kind (such as people, houses, or cars) in visual images and videos.Well-researched object detection domains involve face detection and pedestrian detection.

3.2 CONVOLUTIONAL NEURAL NETWORK

Convolution Neural Network (CNN) or ConvNet uses a special networking technique called an artificial neural network that uses a deep learning algorithm for guided learning (training with a given data set) to analyze data. A basic description of CNN procedure;
1) An input raw image will be divided into three channels R (red), G(green),B(blue) with each channel having pixels in n rows and m columns.
2) Then from the dataset we extract features in the form of m*m matrix.
3) Then this feature matrix will slide over each channel of the image and perform a convolution to produce a feature map.

4) The feature is loaded into a activation function called ReLu (Rectified Linear Unit) which eliminates negative values to avoid zero summing.

5) After ReLu, the activation map is sent over to pooling to shrink the map where max pool technique is used to take the maximum value in the window.

6) The shrinked data is processed through the flattening layer to align the weights in a single row.

7)The flattened data is then classified based on weights in the fully connected layer.

The output is then detected upon comparing the weights for each image and then classified.Figure 3.1 illustrates how a convolution neural network works.

3.2.1 Design

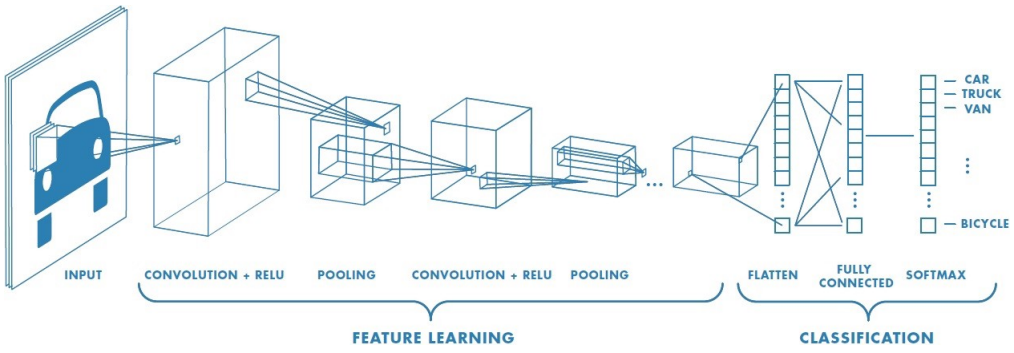

Figure 3.1: CNN Block Diagram

3.2.2 Convolution Layer

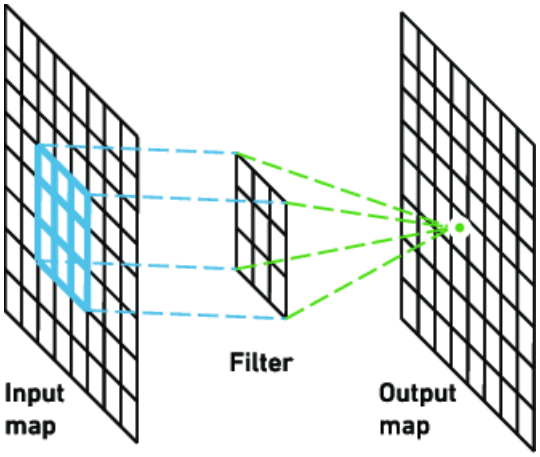

Figure 3.2: Convolutional Layer

The neural convolution is the first big component of CNN. The convolution layer extracts the features from the input image. The provided image pixels are converted to digital binary data. This layer helps to preserve the relationship between image pixels and binary data in the form of a series of small squares. In general, the convolution is a mathematical operation that takes place between two matrixes, such as the image matrix and the filter matrix. Convolution performs multiple operations such as detection edge, detection blur, sharpening, etc. Figure 3.3 explains the operation of convolution layer for an input image.

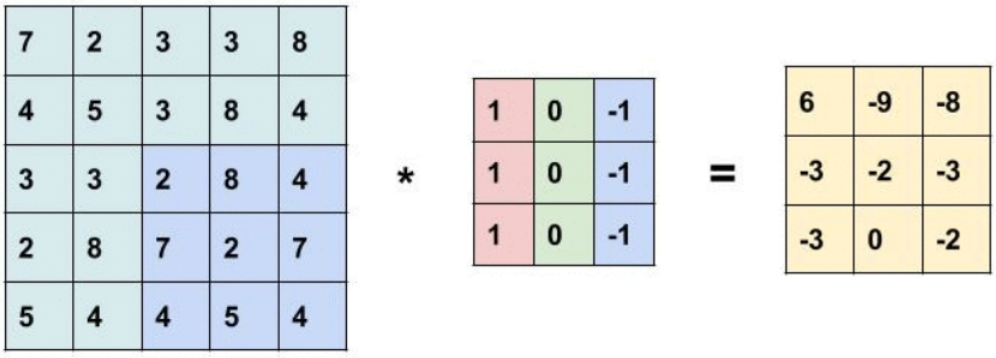

Figure 3.3: Conv Layer Example

3.2.3 Activation Function Layer

Second phase of the CNN includes the activation layer. We use the Rectified Linear Unit Layer (ReLu) activation function on the convolution layers that generate a sparse matrix. It also requires less computational power compared to the sigmoid function. It also reduces the risk of gradients disappearing. It is because when a > 0 gradient is constant, it contributes to faster learning. In the last row, we use softmax activation, which is used to evaluate the likelihood of each object class.

Mathematically, we can define the function ReLu as y = max (0, x) as shown in the Fig (1.4). It is one of the most widely used activation functions, particularly on CNN. The key aim behind using the Re-Lu Image Activation feature is to increase or add non-linearity.
Nonlinear features such as boundaries, colors and transitions between adjacent pixels will be inserted into the image after some kind of activation function has been applied. Based on the activation function, the neurons present in the network can be activated or deactivated.

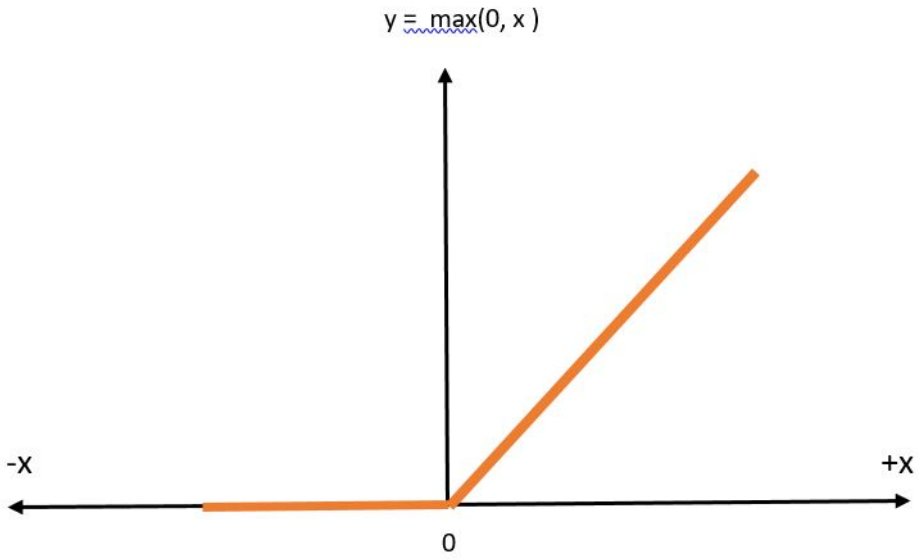

Figure 3.4: Re-Lu Activation Function

Along with this, rectifier is used to further break linearity in order to force additional non-linearity.The Rectified Linear Unit Activation Function or the Re Lu Activation Function is one of the best operations to eliminate non-linearities.Figure 3.5 describes the function of Re-Lu layer.

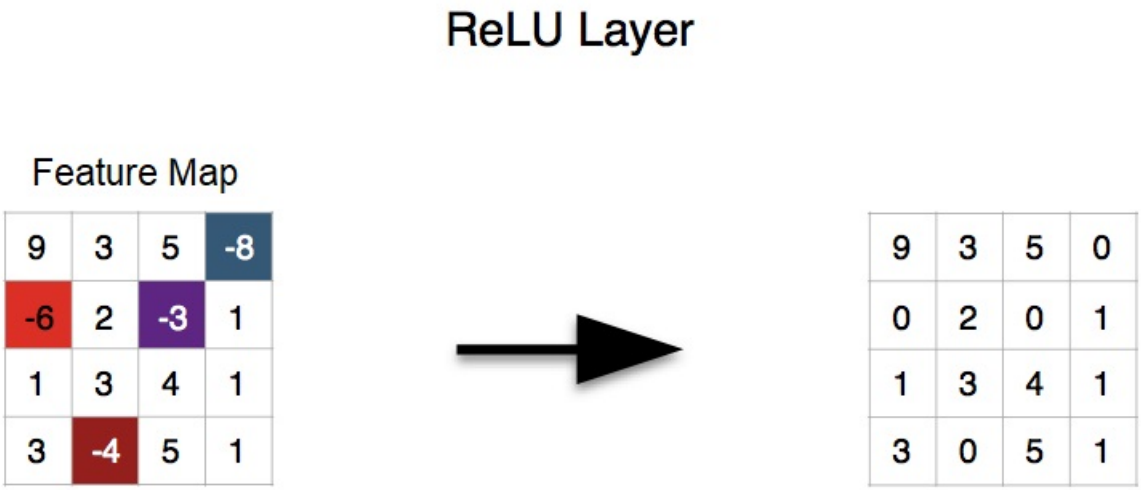

Figure 3.5: Re-Lu Layer Example

3.2.4 Pooling

The pooling layer reduces the number of parameters to be analysed in the image. Many types of pooling approaches may be used to lessen the parameters to be evaluated. The key category of methods for pooling are:

i)Mean pooling

ii)Max pooling

iii)Sum pooling

The method we used here is Max Pooling because, if the image is too large, Max pooling take only the largest parameters. The primary objective of the pooling is to down- sample the existing input representation by reducing its dimensions and other features etc. In this way Pooling helps in reducing the image quality into a great extent and thus by reduces the execution time. A typical max pool operation on a 2*2 window and stride size 2 can be seen in Figure 3.6.

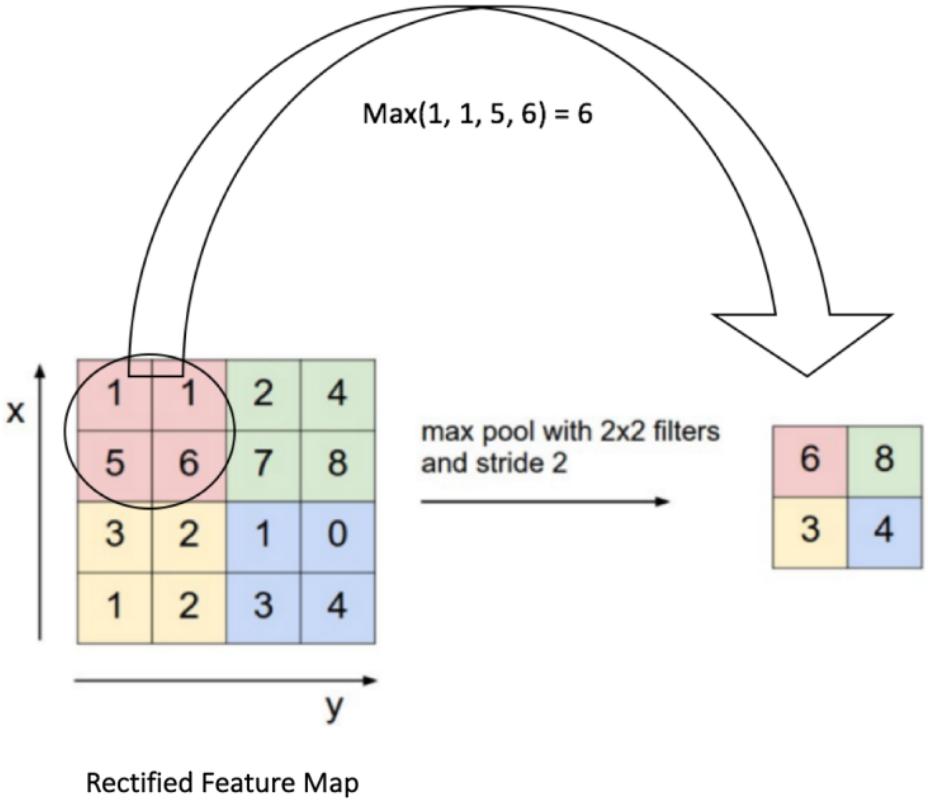

Figure 3.6: Pooled Map

3.2.5 Flattening

Flattening is that the process of converting all the resultant multi-dimensional arrays into one long continuous linear vector form. It gets the output from the earlier layer called pooling and flattens all its structure to make one long feature vector to be employed by the dense layer for the last classification.

3.2.6 Fully Connected Layer

The full connected layer present inside CNN represents the input vector function. The function vector present in the completely connected layer contains the information that is essential to the data. After the model is trained with a certain image data set, the feature vector is then useful for classification, regression, or input into another network, such as RNN, for translating into other output types, etc. It is often used as an encoded vector. During training, this feature vector is used to figure out the loss and help the network to induce training.

The image previously used in the Fully Connected layer holds certain information about local features within the input image, such as edges, blobs, shapes, etc., in the form of a vector. Each and every conv layer contains several types of filters that reflect all the features present in the image. The completely linked layer contains the composite and aggregated information from all the conv layers resulting from the first layer. Flattening and Classification processes are shown in Figure 3.7.

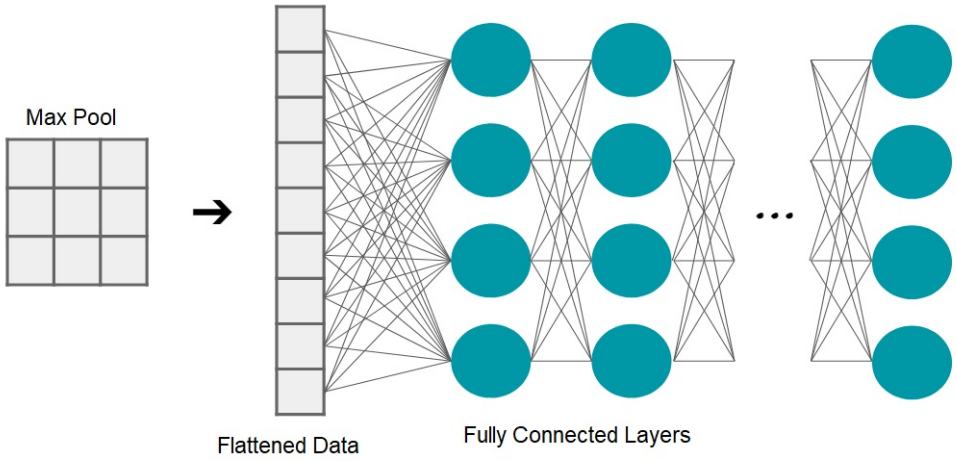

Figure 3.7: Flattening and Fully Connected Layer

3.3 TYPES OF ALGORITHMS FOR OBJECT DETECTION

3.3.1 HOG

For Oriented Gradient Histogram (HOG),the computational capacity is minimal and helpful in dealing with a few realistic issues.In each window that the pyramid sliding window is obtained, we have a bowed to figure Hoard choices that are taken care of to a SVM to make classificators.

3.3.2 R-CNN

Region-CNN (R-CNN) utilizes a factor proposition algorithm named selective-search to decrease the quantity of boundary boxes to the classifier to concerning 2 thousand proposals.Selective search utilizes native signals like type, speed, color and/or an interior indicator etc to induce all gettable item positions.We can cope with those boxes to our classifier subject to CNN.Keep as a main priority, CNN's altogether connected phase takes a collection size input.

3.3.3 SPP-Net

It is comprehensive to run CNN in two thousand with an explicit space search suggestion created by R-CNN.Spatial Pyramid Pooling-Network (SPP-Net) had been making an attempt to mend it.With SPP-Net,as a rule,the pattern of the entire image of CNN will only be realized once, and the CNN definition for each fix is calculated using a selective search.This is often often accomplished by enjoying a pooling activity sort on basically the amount of last conv layer work maps that compares to the territory.By anticipating the region upon conv-layer, the planar part of the conv-layer like an area will be remedied by revisiting the sample in the middle layers merely analytic the directions by sixteen just if there ought to arise an event of Visual Geometry Group (VGG).There was another test: they have to allow the illustrated input size to the CNN's completely associated layers and SPP includes another stunt.Being the last conv-layer, it uses fleeting pooling compared to conventional Maxpool.Pyramidal layer isolates locality of all outright size into a tenacious bin assortment and during this manner the utmost pool on every box is performed.Assuming that the live of bins stays a comparable, a vector of consistent size is generated.There was one vital disadvantage with SPP-Net in any case, guiding

back-propagation by abstraction pooling layer wasn't trifling. The system at that time basically tweaked the fully connected phase of the system.

3.3.4 Fast R-CNN

Fast-RCNN utilizes last thoughts and understands the primary issue of SPP-Net, for example they permitted start to finish teaching.To stretch the gradients by space-pooling, it uses a clear propogate check from back which is in a general sense equivalent to the estimation of the most extreme pooling tendency, of course, really the pooling territories spread and thusly gradients can be constituted in cell from a few parts of area.

It has ability to prepare itself for regressing of boundary boxes.Because of this perspective and ability Fast-RCNN is applied to neural network.Thus, the system presently had two heads, classification head and the head of regressing box.Fast-RCNN perform multiple tasks target is very well,but at this point needs singular system network for localisation and classification.Such two improvements rising the average preparation period and improve the quality of CNN's throughout testing in contrast to spatial pyramidal pooling network.

3.3.5 Faster R-CNN

The slowest element was selective-check in Fast-RCNN.Faster-RCNN substitutes selective-search by an exceptionally little, convolutionary network called Region Proposing Network so as to make esteem regions.

Faster-RCNN fuses the anchor-box concept to suit the distinctions in viewpoint proportion and item size.There are 3 kinds of such type at every area such as 128/128, 256/256 and 512/512. Furthermore it utilizes three perspective proportions 1:1,2:1 and 1:2 for viewpoint proportion. In this way, we have 9 boxes altogether at each spot, on which RPN figures its likelihood being either background or closer view. To support the boxes at each position we utilise regressing in boundary box. In this way,RPN provides the respective probabilities in boxes of different sizes.Through executing Spatial Pooling a lot of like Fast-RCNN, the particular sizes in bouncing boxes might be moved further.The rest of the system is like Fast-RCNN.Figure 3.8

illustrates this procedure.

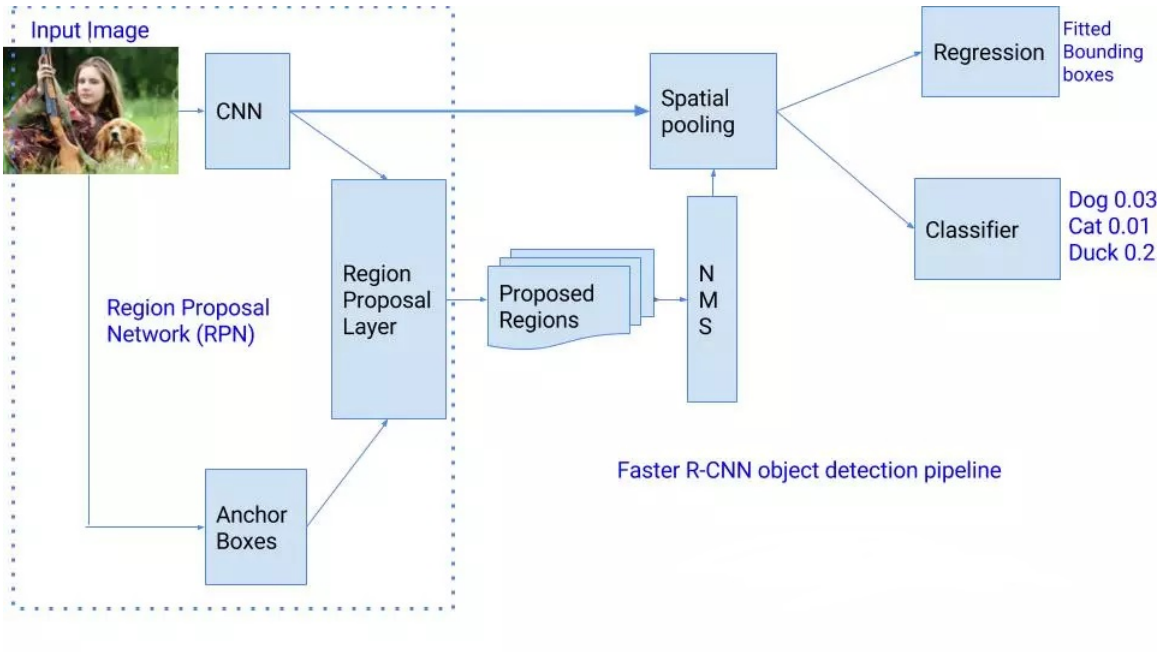

Figure 3.8: Faster RCNN

Faster-R-CNN is significantly ten times speedier than Fast-R-CNN, with comparable data set performance to VOC-2007 which is why Faster-RCNN became one of the most powerful algorithms for target detection.

3.3.6 Yolo

You Only Look Once (Yolo) utilizing CNN to identify objects may be an approximation for a constant item recognition, one of the leading spectacular entity location estimates that also involves a wide range of significant advances in computer vision created by established researchers.

In a scale sight, you can configure the two events and figure out how often the s*s*k size class is open. A large number of these circumstances, in any occasion, include low paces of trust, and on the off chance that you set an objective, express an assurance of 30 percent, you should avoid others.It is in every case snappy and will work endlessly.This rule sees the entire picture and in this manner forestalls bogus useful issues, not exclusively in past methodologies in any case while not estimation.

Yolo/Yolo-1 Algorithm

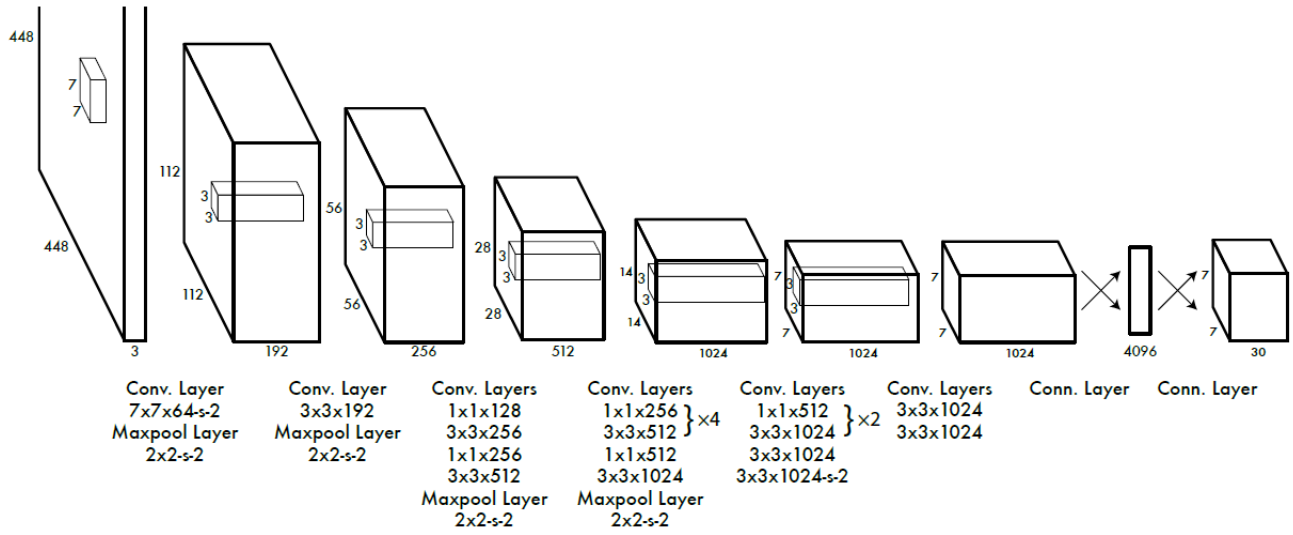

Figure 3.9: Yolo Architecture

The other On-Time Detectors have double the everyday amount of double of avg accuracy.Yolo permits tons of localizing errors relative to the new intelligence activity versions, but do not appear to be likely to induce false positives.Yolo's speed is at frames of 45 per second.In period, Yolo has an out of this world frame rate of 1 hundred 45 per second.In distinction to the window and so the national set up bases, the whole image is utilized for grouping, chase and associated class information and operations practice the foremost customary RCNN proof technologies.Botches unit of measurement created as a attainment in conjunction with various feeling services like DPM and RCNN.The principal identification of Yolo is that it depends on actual footage and managed functions for a selection, like Deformable Parts Model (DPM) and RCNN.Since Yolo repeats apace once it enters new environments or sudden positions, it's less liable to malfunction.This architecture is depicted in Figure 3.9.

Limitations

Much as it is important to attempt cell unit cases for each cell and other types, Yolo places daunting field limits on the box specifications. The design does not generalize into artifacts of several or exceptional size unless data is sufficient to display boundary frames. These spatial constraints should be extended to a limited amount of entities in our model. Seeing that our solution comprises of a broad downstream input stage, the bounding boxes are always tested in a very specific way.

Yolo-2 Algorithm

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Figure 3.10: Yolo-2 Architecture

For distinct events like Pascal Visual Object Classes Objects in Context, the foremost recent Yolo-2 platform is innovative e.g Yolo-9000 might be a multi-scaling app that classifies over 9 thousand object categories in real time with this multi-scaling frameworks.Yolo-9000 is as of presently structuring the coco data and so the Image-Net grouping information bundle. It structure is applied. With over nine,000 example characteristic targets, we tend to tend to support our Image-Net acknowledgment program in extra than 200 associations.Yolo-9000 is nice for re-establishing relationship non-detectoral discoveries.Figure 3.10 explains Yolo-2 architecture.

The alterations from Yolo 1 to 2 are:

Batch Normalization:It normalizes the input layer with the activation gradually changed and reduced. Batch normalization reduces unit value changes in the hidden layer, thus increasing neural network stability. In the higher resolution classifier Yolo-2, the input size has risen from 224/224 to 448/448.This has also helped regulate design and overriding generally. The input size of the mean avg precision architecture has increased to 2 percent. Changing the image input size increased the mean avg precision by up to 4 percent. This increase in input size will be applied during the Yolo-2 DarkNet-19 training on the Image-Net data set.

Anchor Boxes: The addition of anchor boxes is one of the major improvements to Yolo-2. A common system is used to group and estimate Yolo-2. These anchor boxes predict boundaries, use K-means clusters and are built for a given dataset.

Fine-Grained Features:The identification of smaller objects is the main issue of the Yolo-1. The figure in 13/13 grid cells has been solved by the Yolo-2, which is smaller than the previous iteration. This helps Yolo-2 to accurately identify, identify and identify smaller objects in the image.

Multi-Scale Training:Items of various yolo-1 input sizes are restricted to detection. If the pictures of a specific item are little, the same object can not be detected by Yolo in a larger image. It was mostly solved in Yolo-2, where several measurements have been made for random images ranging from 320/30 to 608/608. The network thus enables the understanding and simulation of events in different input dimensions.

Darknet-19:Yolo-2 uses group sorting Darknet-19 software, with nineteen layers, five full bundling layers and one layer of Softmax. As shown below, this design. It is a C-language and CUDA neural network system. Objects that are very useful for predicting in real time are detected quickly.

CHAPTER 4

METHODOLOGY

4.1 DESIGN METHODOLOGY

4.1.1 Yolo Algorithm Architecture

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 4.1: Architecture

This algorithm uses darknet framework for building the structure required for Yolo. It consists of 53 layers each for pre-trained dataset and the dataset we used. Figure 4.1 reveals the architecture of Yolo-3.

There are 2 Conv layers with 32 filters of sizes 3*3 and 1*1, 3 Conv layers with 64 filters of sizes 3*3/2 and 1*1, 3 Conv layers with 128 filters of sizes 3*3/2 and 1*1, 3 Conv layers with 256 filters of sizes 3*3/2 and 1*1, 3 Conv layers with 512 filters of sizes 3*3/2 and 1*1, 3 Conv layers with 1024 filters of sizes 3*3/2 and 1*1.Some of the convolutional layers are followed by a residual block to accommodate skip connections. There are at total 23 residual units. So, there are at total 106 layers in Yolo-3.Of these 106 layers, 75 layers are utilized for the purpose of convolution and the remaining 21 layers are used by activation function, up-sampling, pooling etc.

4.1.2 Working

Algorithms such as RCNN, F-and Faster RCNN are currently being used. In such approaches, the model implements a selective search algorithm along with a pipelining structure. This algorithm is used to extract regions of objects in an image that is given as input to them. It then proposes regions for each object inside the input image and then passes through a series of neural layers of the network. So to improve accuracy, we need to increase the depth of the neural network, i.e. as the number of layers increases. But these algorithms take a long time to test the picture. So they succeed with the Yolo algorithm.

We used Yolo approach to detect and recognize the sign board. The main difference between yolo and the previously described algorithms is that it takes whole image into account, checks every portion of the image. Then it posts boxes for each object in the image. Unlike previous CNN algorithms, yolo scans each and every part of object in the image and posts boxes for objects in the image having value above a threshold.

Basically Yolo divides the image into regions, then predicts the boundary boxes and probability classes for every such region. Generally, yolo divides the visual image into an array of n rows and n columns of grid units, each grid divides into k boundary boxes and the trust value. Trust score implies that the accuracy of the boundary box actually holds the object. Scores for each box will be estimated by yolo for each training session.

Finally we can combine the both classes to calculate the probability of each class in the boundary boxes. So, total n*n*k boxes are predicted. However, the boxes which have confidence score less than a threshold, we can remove them.

Yolo2 and Yolo3 are advanced versions of base yolo network with several enhancements to increase the detection speed and accuracy. The primary benefit of using Yolo-3 over its previous ones is its ability to detect both smaller objects and distinct objects in a single picture. Then by using regression scores for each classes are determined.

YOLO-3 uses a variant of Dark-net, which consists of 53 layer network trained on ImageNet. Yolo3 network considers the image as an n*n grid of sub-images and predicts the class probabilities and boundary boxes for sub-image. The boundary boxes constitutes of four parameters. Such parameters are location of the center of the object along the X axis, the location of the center of the object along the Y axis, the breadth of the object and the length of the object.Yolo-3 can be used effectively for the identification of objects on real time basis. Since modern days applications need everything rapidly, we can utilize Yolo-3.

4.1.3 Block Diagram

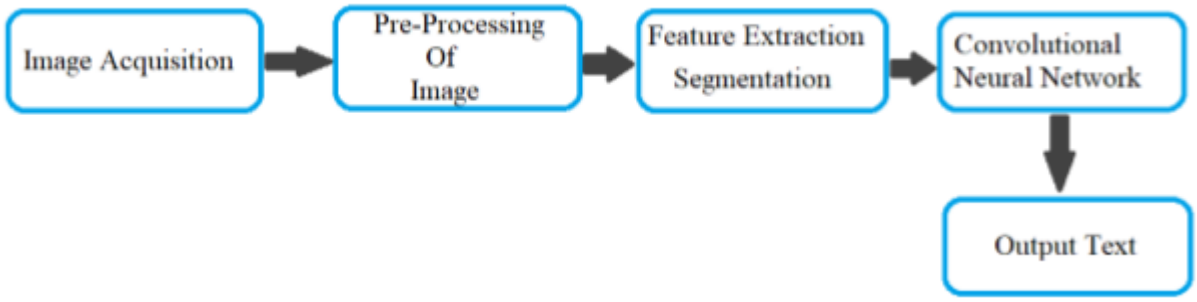

Figure 4.2: Block diagram

A convolutionary neural network can be programmed using the Deep Learning techniques to capture and know such predetermined traffic signs. In conjunction, the neural network utilizes Image detection and Machine learning to facilitate the network to execute. Self drive auto parts firms like Waymo and Uber will create and relocate datasets for traffic signaling together with

infographic and route planning providers as well as Tom Tom. The qualified neural network will be used to detect traffic signal in live time.

This work is incredibly successful and sustainable in real time with sophisticated computer vision and neural network technology.The most popular are those which are based on the sign-board form.There are numerous sign-board detection algorithms.Typical forms of sign-board shapes such as circles, rectangles and hexagons identify different styles to make a distinction between them.The fibrous features, the Free-man chain coding, Ada-Boost recognition and the cognitive networking strategies for machine learning are few appropriate text detection algorithms.

The convolution layer computes the yield of neurons that are linked to regional areas or responsive areas, each calculating a point product in the volume input to which they are connected, among their weights and a low receptive field. In other words, presumes that you have a picture outlined as a 5/5, vector value matrix so you take a 3/3 matrix, so that the window or the kernel passes via the picture. In each and every location in the matrix you deduct the 3/3 window values by a presently secured glass image values.

4.1.4 Flow Chart

1)Extract borders of characters i.e.Contours may be defined clearly as a curve that connects all the continuous points (along the boundary). The contours are a useful tool for analysis of shape and for detection and recognition of objects. Here Contours demonstrated how to distinguish each human character in an image by using the technique of contour expanding method.

2)With the open-cv method, an image boundary is developed for each character. Open-cv character identification technique. Eight layers with 2/4 layer residual feed back in the convolutionary network in order to reminisce about key images of patterns.

3)The first model must train specifically assorted individual text photos in order to measure the soft-max category classification(2nd model must have information about receipt picture function);the last predictor is the same model as the predictor for measurement of embedding

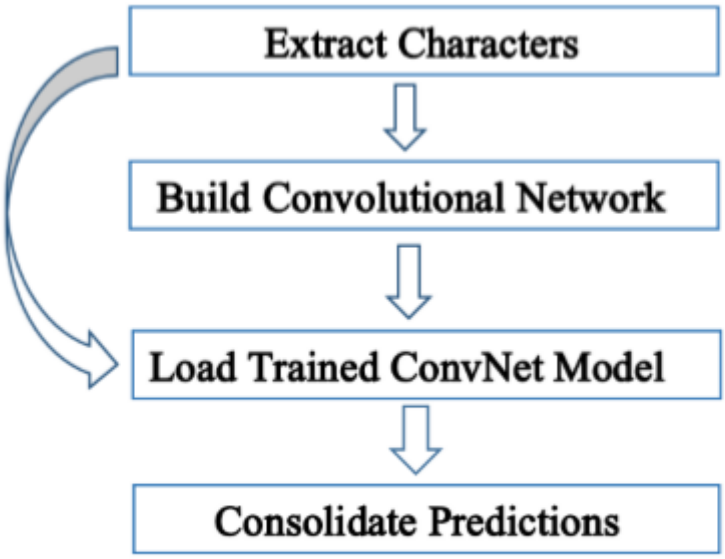

Figure 4.3: Flow Chart

defined flat neuron..

4)The final step in the OCR involves the pre-processing of a picture in specific word contour, accompanied by an estimation and consolidation of contours by letter and word in a picture.

5)Consolidating the estimates, each term contour is assigned a pervasive identity along the line aligned with that term in the picture and all predictions merged in a type pattern of various word shapes and positioned letters.

4.2 REALISTIC CONSTRAINTS

Requirement of huge amount of data for training YOLO-3 CNN model:Training a dataset is the exhaustive work.We will be having vast amount of data and we need to train the data for obtaining weights.We should be vigilant while setting the batch and sub-division numbers which determine the accuracy of a model's prediction.The last and most arduous task is determining weights with suitable mAP for predictions.

Processor Speed:The processor speed defines the frame rate i.e,number of frames per second.Training with a CPU is exhaustive since it uses PC's processor.Rather we use GPU of PC

to boost the training speed.So processor speed is what that determines good accuracy.

Resolution of the Camera:Some times due to bad weather conditions,detection may be arduous.So we should have a high resolution camera to detect the objects or sign boards to avoid blurring.

4.3 MULTIDISCIPLINARY ASPECTS

Machine Learning:We need to have a sheer knowledge of machine learning models(Deep Learning) for object detection.Once we know the depth of these models,we can know the pros and cons of a model and can avoid them easily while using them somewhere else and we can subsequently develop further models.

Image Processing:We should have an absolute idea about image processing and the parameters involved in it.There are certain parameters that define an image and they should be reproduced while detecting an object inside an image.

Programming Language:At present,there are many programming languages in use for machine learning.But today most of the machine learning models are implemented using Python because of its ease of interpretation.So one should have thorough knowledge in it and its dependent libraries.

4.4 ENGINEERING STANDARDS

4.4.1 IEEE Standards for ITS

P2020 Standard for Automotive System Image Quality:It outlines the basic characteristics which relate to automobiles image resolution for ADAS uses and recognizes available metrics and other helpful information linked to such features. This establishes a standard set of qualitative and quantitative assays to measure picture resolution attributes of automobile camera. In addition, resources and evaluation techniques are specified to enhance standard oriented interaction and correlation of automobile image quality between manufacturers and suppliers of structure former and unit suppliers.It defines strategies and measurement metrics and evaluating techniques to assure durability and inter-entity refer points automobile picture standard.

CHAPTER 5

CONCLUSION

5.1 DATASET

5.1.1 GTSDB Dataset

The two prominent datasets for signboard detection, training and assessment models are the German Traffic Sign Recognition Benchmark and the German Traffic Sign Detection Benchmark .In and German Traffic Sign Detection Benchmark, we have 900 pictures .Off these 900 pictures approximately 600 pictures are used for training process and 300 images are used for testing process. All these testing and training pictures have a default .ppm format. The main reason behind choosing German Traffic Sign Detection Benchmark over German Traffic Sign Recognition Benchmark is that German Traffic Sign Detection Benchmark has pictures with high resolution as 1360*800p.Below shown images from Figure 5.1 and Figure 5.2 are some of the images from German Traffic Sign Detection Benchmark.Table 5.1 lists the GTSDB classes.

5.1.2 ImageNet Dataset

Another dataset we used for this process is Image-Net as shown compositely in Figure 5.3.ImageNet is an open source computer vision research database which primarily re-posits images for the research and educational purposes. It has over 1.4 crores of images. This dataset is loaded already in yolo algorithm. Then we get a pre-trained weights. These pre-trained weights are then fed into the neural network. The network then extracts the features from pre-trained weights.

S.No	Class	22	Bend
1	Speed limit 20	23	Uneven road
2	Speed limit 30	24	Slippery road
3	Speed limit 50	25	Road narrows
4	Speed limit 60	26	Construction
5	Speed limit 70	27	Traffic signal
6	Speed limit 80	28	Pedestrian crossing
7	Restriction ends 80	29	School crossing
8	Speed limit 100	30	Cycles crossing
9	Speed limit 120	31	Snow
10	No overtaking	32	Animals
11	No overtaking (trucks)	33	Restriction ends
12	Priority at next intersection	34	Go right
13	Priority road	35	Go left
14	Give way	36	Go straight
15	Stop	37	Go right or straight
16	No traffic both ways	38	Go left or straight
17	No trucks	39	Keep right
18	No entry	40	Keep left
19	Danger	41	Roundabout
20	Bend left	42	Restriction ends (overtaking)
21	Bend right	43	Restriction ends (overtaking (trucks))

Table 5.1: GTSDB classes

Figure 5.1: GTSDB Dataset Images

Figure 5.2: GTSDB Dataset Images

Figure 5.3: ImageNet Dataset

5.2 Parameters Involved in Training Dataset

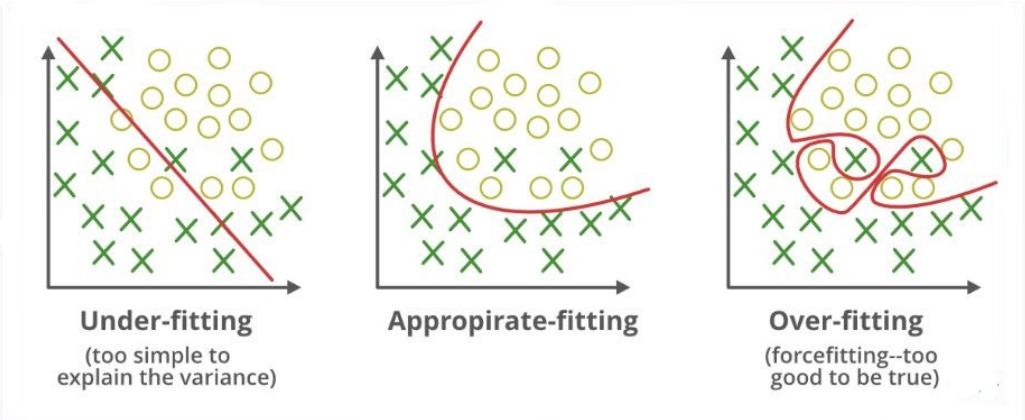

Figure 5.4: Over-fitting,Appropriate-fitting,Under-fitting

Over-fitting:If a model has a lot of input, it begins to learn from untrustworthy data lists. If a model is over-fitted, the data collection is used. The model does not properly describe the data because of too much detail and noise.Over-fitting sources include non-parametric and non-linear strategies, because the model can make the profound learning algorithms more resilient and possibly unreasonable. A linear algorithm can be used for avoiding over-fitting when using vector data or parameters such as the highest depth trees.

It occurs when approximately 100 percent and approximately 0 percent of validation data or other actual photos may be found on a test dataset after the testing pattern.

This can be graphically interpreted because the loss of both training and validation data sets from the start decreases. At some stage, however, there is a decrease in data set losses, which is still growing. At the same time, the loss for training data continues to decrease to almost 0, which is almost 100 percent correct detections and no errors. The objective is to define this

specific point where the loss of validation data set is limited. The approaches commonly used to prevent over-fitting are:
(1)Cross-Validation:A standard way of figuring out-sample forecast mistakes is to use five-fold cross confirmation.
(2)Early Stopping:The principles give us guidance on how many changes should be made before the trainee is unduly fit.

(3)Pruning:It is widely used in similar models during development. The nodes that give the question less analytical power are literally removed.
(4)Regularization:This adds a price notion with the aim of adding more features. Thus, the coefficients for various variables are relocated to zero and therefore the cost term boosts.

Under-fitting:An algorithm or numerical model for computer vision is considered to be inconceivably inappropriate if it can not capture the intrinsic trends of results. The existence of our model or algorithm simply implies our results don't work reasonably well, normally if we have fewer data to create a particular model and even try to build a linear model with non-linear info. The rules of the computer vision algorithm are too easy to implement and are robust for these simple data. The algorithm is like that.

Good-Fit Statistical Model:The situation is generally supposed to correspond to the data, as the model makes zero error predictions. This condition should be done at a time where it is over-fitting and under-fitting. They need to stop at one point until the error intensifies in order to match properly. It is said that the model possesses deep understanding on the training data set and on our unidentified scientific dataset.

Mean Average Precision (mAP):In this scenario, mAP is a measure used to examine accuracy for Target Identification tasks. The average precision for each class in the individual model is generally established to measure mAP for a bespoke model that is first trained in the function of recognition. Then there is an average plot of all groups of such measures.

Threshold:Threshold is used to assess whether the estimate Bounding Boxes (BB) is correct or incorrect. Typically, the standard is founded at an extent, that is 50, 75, and 95 percent.

Intersection Over Union (IoU):It reveals how many of the projected BB overlaps with the so-called Ground Reality BB with the actual object. It is used to determine overlaps between two BB. , While contrasts between IoU and threshold, it can also be determined by overlaps between BB expected and Ground Reality BB separated into two Bounding Box territories, it can be determined whether the anticipated BB is TP or FP.

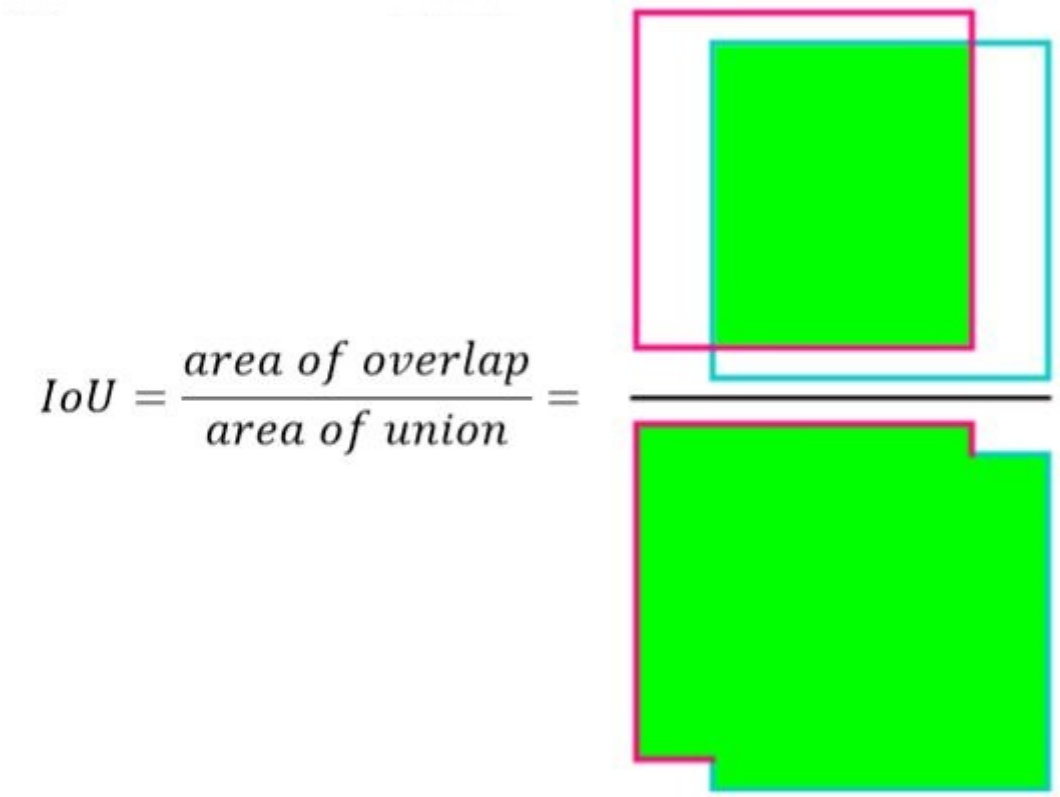

Figure 5.5: IoU between Ground Truth BB and predicted BB

To compute,fully comprehend models, the coming new definitions are used:

•True Positive (TP) is no. of BB with correct predictions,IoU>=threshold

•False Positive (FP) is no. of BB with wrong predictions,IoU<=threshold

•False Negative (FN) is no. of Ground Truth BB which aren't identified.

•True Negative (TN) is no. of BB's not adequately anticipated (in just an image as possible, but which do not overlap any ground truth BB).

Class	GTSDB Dataset Sign Boards
Prohibitory	Speed Limit(20,30,50,60,70,80,100,120), No Overtaking, No Overtaking (trucks), No traffic both ways, No trucks
Mandatory	Go Right, Go Left, Keep Left, Keep Right, Go Straight, Go Left or Straight, Go Right or Straight, Roundabout
Danger	Bend left, Bend right, Bend, Uneven road, Slippery road, Road narrows, Construction, Traffic Signal, Pedestrian Crossing, School Crossing, Cycles Crossing, Snow, Animals, Priority at next intersection, Danger
Other	Priority road, Give Way, Stop, Restriction Ends for 80kmph, Restriction Ends, No Entry, Restriction ends (overtaking), Restriction ends (overtaking (trucks))

Table 5.2: GTSDB classes division into four classes

5.3 RESULTS AND DISCUSSION

Sometimes due to obscure images, the detector might not predict the sign board accurately. So based on the features like shape, texture we divided the 43 classes further into four categories. We are able to detect 43 classes divided into 4 categories (prohibitory, mandatory, danger, other) which are defined in the dataset of German Traffic Sign Detection Benchmark. Some samples of our detected signs are shown in Figures 5.6,5.7,5.8 and 5.9.Table 5.2 lists the division of GTSDB classes into four classes.Out of the 43 classes 12 classes are assorted to prohibitory, 8 classes assorted to mandatory, 15 classes assorted for danger and remaining classes to other.We are able to detect sign-boards in a video at **9 Frames Per Second (FPS)**.

Figure 5.6: Detected sign board for sample with Pedestrian Crossing Sign

Figure 5.7: Detected sign board for sample with Speed Limit 80 Kmph sign

Figure 5.8: Detected sign board for sample with Give Way and Stop sign

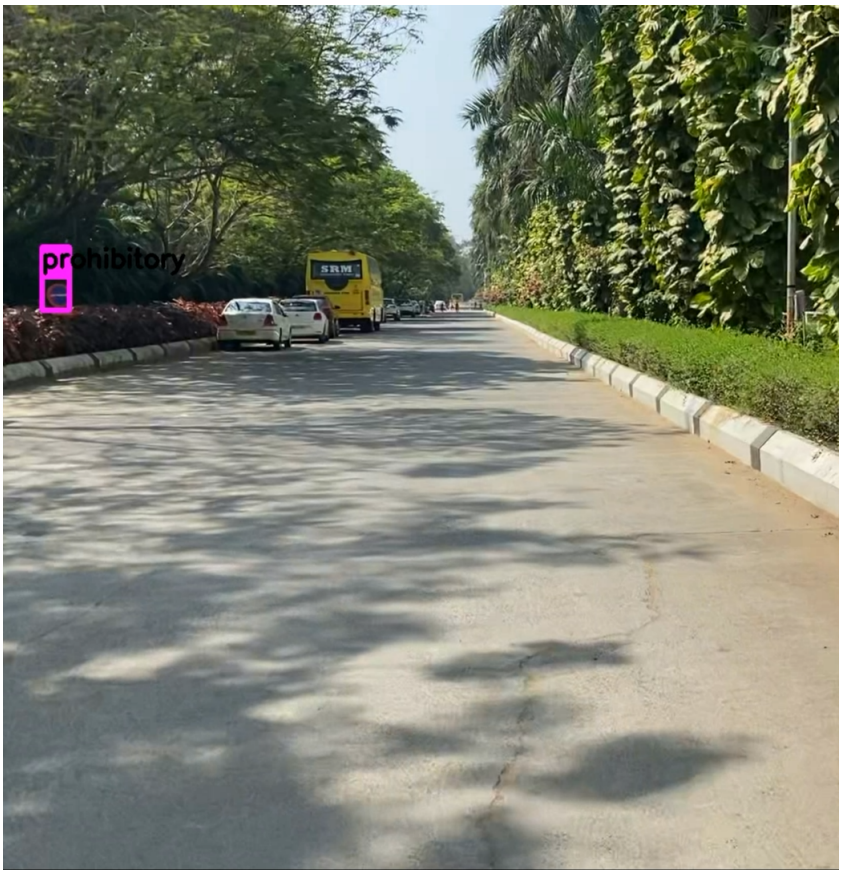

Figure 5.9: Detected sign board for sample with No Entry sign

5.4 CONCLUSION

The project is intended to preserve valuable lives by preventing deaths incurred by traffic signals being ignored. The system mostly focuses on the bulk of the population, particularly those who used to be driving at night, and also helps the traffic police to alleviate problems. This initiative's main concept is the traffic accidents sustained by the neglect of road signals by the driver.

5.5 FUTURE ENHANCEMENT

We can implement the newest item-detection architectures like a feature pyramid network and multi scale training to provide greater precision or pace of detection.Since the ground network is a decisive factor in the return of speed-accuracy compensation, it is worth investigating instead of using the existing ground network for generic object detections the development of a ground network specially trained for sign-board identification.Finally,not only traffic signs but other flat objects of standard form can also be utilized with the technique suggested.

PUBLICATION

Our paper "SIGN BOARD RECOGNITION BASED ON CNN USING YOLO-3" has been accepted for inclusion in the Journal of Computational and Theoretical Nanoscience (Scopus Indexed) at National Conference on Advances in Big data and Cloud Computing,2020 which is to be in between 27th-28th April 2020 at Sathyabama Institute of Science and Technology,Chennai under the special issue of "Internet of Things: Principles, Applications and Security Concerns".

REFERENCES

[1] de Oliveira, G. H., da Silva, F. A., Pereira, D. R., de Almeida, L. L., Artero, A. O., Bonora, A. F., and de Albuquerque, V. H. C. (2018). "Automatic detection and recognition of text-based traffic signs from images." *IEEE Latin America Transactions*, 16(12), 2947–2953.

[2] Fang, C.-Y., Chen, S.-W., and Fuh, C.-S. (2003). "Road-sign detection and tracking." *IEEE transactions on vehicular technology*, 52(5), 1329–1341.

[3] Greenhalgh, J. and Mirmehdi, M. (2014). "Recognizing text-based traffic signs." *IEEE Transactions on Intelligent Transportation Systems*, 16(3), 1360–1369.

[4] Wu, W., Chen, X., and Yang, J. (2005). "Detection of text on road signs from video." *IEEE Transactions on Intelligent Transportation Systems*, 6(4), 378–390.

[5] Xie, L., Ahmad, T., Jin, L., Liu, Y., and Zhang, S. (2018). "A new cnn-based method for multi-directional car license plate detection." *IEEE Transactions on Intelligent Transportation Systems*, 19(2), 507–517.

APPENDICES

CODE

```
#DATA PART
# Detecting Objects on Video with
import numpy as np
import cv2
import time

"""
Start of:
Reading input video
"""
# 'videos\\traffic-cars.mp4'
video = cv2.VideoCapture('videos/traffic-cars.mp4')
# Preparing variable for writer
# that we will use to write processed frames
writer = None
# Preparing variables for spatial dimensions of the frames
h, w = None, None

"""
End of:
Reading input video
"""

"""
Start of:
Loading YOLO v3 network
"""
# Loading COCO class labels from file
# Opening file
```

```
# Pay attention! If you're using Windows, yours path might looks like:
# r'yolo-coco-data\coco.names'
# or:
# 'yolo-coco-data\\coco.names'
with open('yolo-coco-data/coco.names') as f:
    # Getting labels reading every line
    # and putting them into the list
    labels = [line.strip() for line in f]
# # Check point
# print('List with labels names:')
# print(labels)
# Loading trained YOLO v3 Objects Detector
# with the help of 'dnn' library from OpenCV
# Pay attention! If you're using Windows, yours paths might look like:
# r'yolo-coco-data\yolov3.cfg'
# 'yolo-coco-data\\yolov3.weights'
# Getting list with names of all layers from YOLO v3 network
layers_names_all = network.getLayerNames()
# # Check point
# print()
# print(layers_names_all)
# Getting only output layers' names that we need from YOLO v3 algorithm
# # Check point
# print()
# Setting minimum probability to eliminate weak predictions
probability_minimum = 0.5
# Setting threshold for filtering weak bounding boxes
# with non-maximum suppression
threshold = 0.3
# Generating colours for representing every detected object
# # Check point
# print()
```

```
# print(type(colours))  # <class 'numpy.ndarray'>
# print(colours.shape)  # (80, 3)
# print(colours[0])  # [172  10 127]

"""
End of:
Loading YOLO v3 network
"""
"""
Start of:
Reading frames in the loop
"""
# Defining variable for counting frames
# At the end we will show total amount of processed frames
f = 0
# Defining variable for counting total time
# At the end we will show time spent for processing all frames
t = 0
# Defining loop for catching frames
while True:
    # Capturing frame-by-frame
    ret, frame = video.read()

    # If the frame was not retrieved
    # e.g.: at the end of the video,
    # then we break the loop
    if not ret:
        break
    # Getting spatial dimensions of the frame
    # we do it only once from the very beginning
    # all other frames have the same dimension
    if w is None or h is None:
```

```
    # Slicing from tuple only first two elements

"""
Start of:
Getting blob from current frame
"""
# Getting blob from current frame
# The 'cv2.dnn.blobFromImage' function returns 4-dimensional blob fro
# frame after mean subtraction, normalizing, and RB channels swapping
# Resulted shape has number of frames, number of channels, width and
# E.G.:

"""
End of:
Getting blob from current frame
"""
"""
Start of:
Implementing Forward pass
"""
# Implementing forward pass with our blob and only through output lay
# Calculating at the same time, needed time for forward pass
network.setInput(blob)  # setting blob as input to the network
start = time.time()
output_from_network = network.forward(layers_names_output)
end = time.time()
# Increasing counters for frames and total time
f += 1
t += end - start
# Showing spent time for single current frame
print('Frame number {0} took {1:.5f} seconds'.format(f, end - start))
"""
```

```
End of:
Implementing Forward pass
"""
"""
Start of:
Getting bounding boxes
"""
bounding-boxe = []
# Going through all output layers after feed forward pass
for result in output_from_network:
    # Going through all detections from current output layer
    for detected_objects in result:
        # Getting 80 classes' probabilities for current detected obje
        scores = detected_objects[5:]
        # Getting index of the class with the maximum value of probab
        class_current = np.argmax(scores)
        # Getting value of probability for defined class
        confidence_current = scores[class_current]

        # # Check point
        # # Every 'detected_objects' numpy array has first 4 numbers
        # # bounding box coordinates and rest 80 with probabilities
        #  # for every class
        # print(detected_objects.shape)  # (85,)

        # Eliminating weak predictions with minimum probability
        if confidence_current > probability_minimum:
            # Scaling bounding box coordinates to the initial frame s
            # YOLO data format keeps coordinates for center of boundi
            # and its current width and height
            # That is why we can just multiply them elementwise
            # to the width and height
```

```
                # of the original frame and in this way get coordinates f
                # of bounding box, its width and height for original fram
                box_current = detected_objects[0:4] * np.array([w, h, w,

                # Now, from YOLO data format, we can get top left corner
                # that are x_min and y_min
                x_min = int(x_center - (box_width / 2))
                y_min = int(y_center - (box_height / 2))

                # Adding results into prepared lists
                bounding_boxes.append([x_min, y_min,
                                       int(box_width), int(box_height)])
                confidences.append(float(confidence_current))
                class_numbers.append(class_current)
"""
End of:
Getting bounding boxes
"""
"""
Start of:
Non-maximum suppression
"""
# Implementing non-maximum suppression of given bounding boxes
# With this technique we exclude some of bounding boxes if their
# corresponding confidences are low or there is another
# bounding box for this region with higher confidence

# It is needed to make sure that data type of the boxes is 'int'
# and data type of the confidences is 'float'
# https://github.com/opencv/opencv/issues/12789
results = cv2.dnn.NMSBoxes(bounding_boxes, confidences,
                           probability_minimum, threshold)
```

```
"""
End of:
Non-maximum suppression
"""
"""
Start of:
Drawing bounding boxes and labels
"""
# Checking if there is at least one detected object
# after non-maximum suppression
if len(results) > 0:
    # Going through indexes of results
    for i in results.flatten():
        # Getting current bounding box coordinates,
        # its width and height
        # Preparing colour for current bounding box
        # and converting from numpy array to list
        colour_box_current = colours[class_numbers[i]].tolist()
        # # # Check point
        # print(type(colour_box_current))  # <class 'list'>
        # print(colour_box_current)  # [172 , 10, 127]

        # Drawing bounding box on the original current frame
        cv2.rectangle(frame, (x_min, y_min),
                      (x_min + box_width, y_min + box_height),
                      colour_box_current, 2)

        # Preparing text with label and confidence for current boundi
        text_box_current = '{}: {:.4f}'.format(labels[int(class_numbe
                                               confidences[i])
```

```
        # Putting text with label and confidence on the original imag
        cv2.putText(frame, text_box_current, (x_min, y_min - 5),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, colour_box_current
"""
End of:
Drawing bounding boxes and labels
"""
"""
Start of:
Writing processed frame into the file
"""
# Initializing writer
# we do it only once from the very beginning
# when we get spatial dimensions of the frames
if writer is None:
    # Constructing code of the codec
    # to be used in the function VideoWriter
    fourcc = cv2.VideoWriter_fourcc(*'mp4v')

    # Writing current processed frame into the video file
    # Pay attention! If you're using Windows, yours path might looks
    # r'videos\result-traffic-cars.mp4'
    # or:
    # 'videos\\result-traffic-cars.mp4'
    writer = cv2.VideoWriter('videos/result-traffic-cars.mp4', fourcc
                             (frame.shape[1], frame.shape[0]), True)

# Write processed current frame to the file
writer.write(frame)
"""
End of:
Writing processed frame into the file
```

```
    """
"""
End of:
Reading frames in the loop
"""
# Printing final results
print()
print('Total number of frames', f)
print('Total amount of time {:.5f} seconds'.format(t))
print('FPS:', round((f / t), 1))
# Releasing video reader and writer
video.release()
writer.release()
```

TESTING CONFIGURATION

```
[net]
# Testing
batch=1
subdivisions=1
width=416
height=416
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1
learning_rate=0.001
burn_in=1000
max_batches = 8000
policy=steps
steps=6400,7200
scales=.1,.1
[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
```

```
pad=1
activation=leaky
# Downsample

[convolutional]
batch_normalize=1
filters=64
size=3
stride=2
pad=1
activation=leaky
[convolutional]
batch_normalize=1
filters=32
size=1
stride=1
pad=1
activation=leaky
[convolutional]
batch_normalize=1
filters=64
size=3
stride=1
pad=1
```

```
activation=leaky

[shortcut]

from=-3

activation=linear


# Downsample

[convolutional]

batch_normalize=1

filters=128

size=3

stride=2

pad=1

activation=leaky

[convolutional]

batch_normalize=1

filters=64

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch_normalize=1

filters=128

size=3
```

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch_normalize=1

filters=64

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch_normalize=1

filters=128

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

```
# Downsample

[convolutional]
batch_normalize=1
filters=256
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky
```

```
[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear
```

```
stride=1
pad=1
activation=leaky
[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky
[shortcut]
from=-3
activation=linear
[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky
[convolutional]
batch_normalize=1
filters=256
size=3
```

```
stride=1
pad=1
activation=leaky
[shortcut]
from=-3
activation=linear
# Downsample
[convolutional]
batch_normalize=1
filters=512
size=3
stride=2
pad=1
activation=leaky
[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky
[convolutional]
batch_normalize=1
filters=512
```

```
activation=leaky

[convolutional]

batch_normalize=1

filters=512

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch_normalize=1

size=3

stride=1

pad=1

filters=1024

activation=leaky

[convolutional]

batch_normalize=1

filters=512

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch_normalize=1
```

size=3

stride=1

pad=1

filters=1024

activation=leaky

[convolutional]

size=1

stride=1

pad=1

filters=27

activation=linear

[yolo]

mask = 6,7,8

anchors = 10,13, 16,30, 33,23, 30,61, 62,45, 59,119, 116,90, 156,198, 373,326

classes=4

num=9

jitter=.3

ignore_thresh = .7

truth_thresh = 1

random=1

[route]

layers = -4

[convolutional]

batch_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[upsample]

stride=2

[route]

layers = -1, 61

[convolutional]

batch_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch_normalize=1

size=3

stride=1

pad=1

filters=512

activation=leaky

[convolutional]

batch_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch_normalize=1

size=3

stride=1

pad=1

filters=512

activation=leaky

[convolutional]

batch_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch_normalize=1

size=3

stride=1

pad=1

filters=512

activation=leaky

[convolutional]

size=1

stride=1

pad=1

filters=27

activation=linear

[yolo]

mask = 3,4,5

anchors = 10,13, 16,30, 33,23, 30,61, 62,45, 59,119, 116,90, 156,198, 373,326

classes=4

num=9

jitter=.3

ignore_thresh = .7

truth_thresh = 1

random=1

[route]

layers = -4

[convolutional]

batch_normalize=1

filters=128

size=1

```
stride=1
pad=1
activation=leaky
[upsample]
stride=2
[route]
layers = -1, 36
[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky
[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=256
activation=leaky
[convolutional]
batch_normalize=1
filters=128
```

```
size=1
stride=1
pad=1
activation=leaky
[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=256
activation=leaky
[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky
[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=256
```

activation=leaky

[convolutional]

size=1

stride=1

pad=1

filters=27

activation=linear

[yolo]

mask = 0,1,2

anchors = 10,13, 16,30, 33,23, 30,61, 62,45, 59,119, 116,90, 156,198, 373,326

classes=4

num=9

jitter=.3

ignore_thresh = .7

truth_thresh = 1

random=1