STUDY OF MACHINE LEARNING ALGORITHMS FOR VEHICLE DETECTION ON INDIAN ROADS

N Jayanth Aditya [1], Piranav kumar K [2]. Karnth M[3], Praveen Sai Kumar Vj[4]
Bachelor Of Engineering In Electronics And Communication Engineering, Sri Venkateswara College Of Engineering

ABSTRACT

The aim of this project is to study machine learning algorithms used for vehicle detection on Indian Roads.

The state of art models that are used for vehicle detection are YOLO-v3 and F-RCNN. YOLO-v3 is a model created by Darknet, This model contains 106 layers (75 convolutional layers and 31 layers for shortcut, route, and up sample).YOLO-v3 does detection at three different scales which are at layers 82, 94, and 106.This model uses pre-trained weights that have been created by training on MS COCO dataset which contains 123,287 images that are categorized into 80 different classes. A test image (Indian road) of input size 416x416 is fed into the pre-trained model. This model divides the image into grids and then converts it into a NumPy array. The model then detects the elements present in the images with respect to four coordinates xmin, xmax, ymin, ymax.

F-RCNN is a model derived from R-CNN. R-CNN uses selective search algorithm to find regions of interest and passes the RoIs to a convolution network. 2,000 proposed areas of interests are passed to the convolution network from the selective search algorithm. In F-RCNN the selective search algorithm is replaced by a region of proposal network. The region proposal network produces feature maps that are fed to a convolutional neural network for object detection.

YOLO v3 and F-RCNN were tested on images captured on Indian roads. The models were also tested on seven types of distorted images namely Gaussian, local variance, salt, pepper, salt and pepper, and speckle. The final outcome of the project is a detailed comparison between F-RCNN and YOLO v3. The comparison is based on performance, inference time and accuracy of the model.

INTRODUCTION

IMPORTANCE OF VEHICLE DETECTION

Autonomous driving in any society is poised to change life. Recent incidents, however, indicate that it is not yet clear how a manmade perception device can avoid even seemingly obvious errors when a driving system is implemented in the real world. A system's ability to identify a

vehicle accurately when driving in and around is known as vehicle detection. Various algorithms can be embedded to implement vehicle detection.

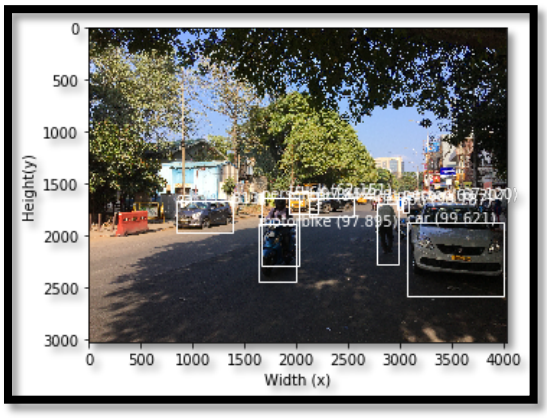


Fig 1.1 Example Image of Random Vehicle Detection

Vehicle identification and statistics in video highway surveillance scenes are of significant value for sophisticated road traffic management and highway regulation. Intelligent vehicle detection and counting in the field of motorway management are becoming

increasingly relevant. However, their identification remains a problem, due to the various sizes of vehicles, which directly affects the accuracy of vehicle counts.

The object detection subtask is itself one of the most critical prerequisites for autonomous navigation in many autonomous driving systems, as this function is what allows the car controller to compensate for obstacles while considering potential future trajectories. Therefore, we desire object detection algorithms that are as accurate as possible. Many high-quality object detectors have seen astounding improvements in recent years.

REAL-TIME DETECTION

A real-time vision system was developed based on research that analyses videos in colour, images taken from a forward-looking video camera. The device uses a combination of information on colour, edge and motion to identify and track the vehicles on the route. Cars are recognized by matching models from the image dataset, and by detecting features of the highway scene and determining how they relate to each other. Cars are often identified by temporal differentiation and by monitoring parameters of motion that are typical of cars. Experimental findings show robust auto detection and monitoring in real-time over thousands of image frames. The following data includes images taken under difficult visibility conditions.

The pictures show the environment in front of the car-next to the lane, other vehicles, bridges, pedestrians and trees. The system's primary role is to differentiate the cars in the images from other stationary and moving objects, and identify them as vehicles. This is a difficult activity since it is not known in advance about the constantly changing terrain along the road and the various lighting conditions that depend on the time of day and environment. It's difficult to recognize vehicles which suddenly enter the scene. Cars and trucks come up with very different speeds, sizes and looks.

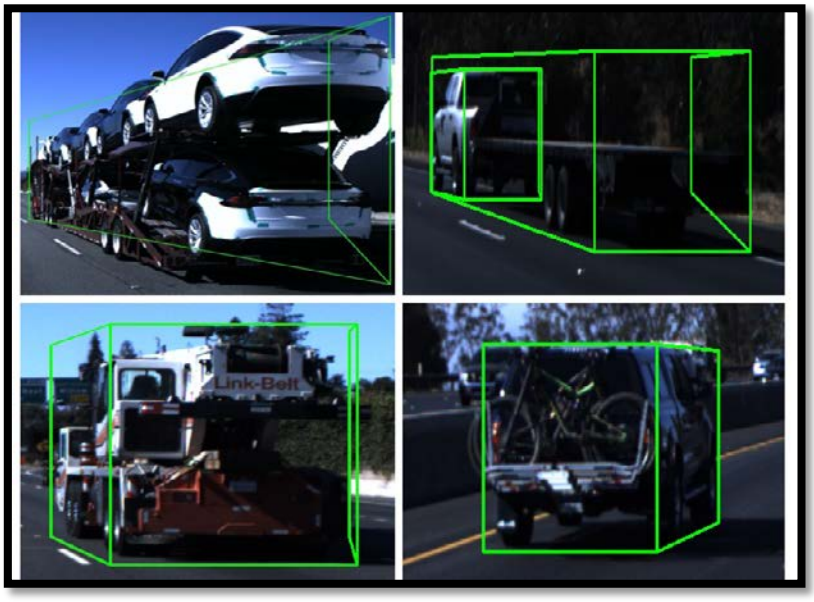


Fig 1.2 Example of vehicle detection using bounding boxes

In videos taken from a car driving on Indian roads, we have built and implemented a hard real-time vision system that recognizes and monitors lanes, road boundaries and multiple vehicles. Our device can run on easy, low cost hardware in real time. An ordinary video camera and a Laptop are what we count on. The vision algorithms use a combination of information about light, hue, and saturation to analyse the image. Highway lanes and boundaries are identified using an algorithm and are monitored. The road is segmented into regions of interest, the "bounding boxes," from which on-line vehicle models are generated and evaluated in real time for symmetry.

The machine inferences that a vehicle is identified and monitored from the detection and motion background of these screens, the identified features and the effects of correlation and symmetry. Experimental tests show reliable, real-time car identification and tracking over thousands of picture frames, unless the device experiences uncooperative conditions, e.g. too little contrast of brightness between the

cars and the background, which often happens at daytime, but mostly at night, and in highly congested traffic. The system works well under reduced visibility conditions, even when the backdrop is uniformly dark and some noise is added.

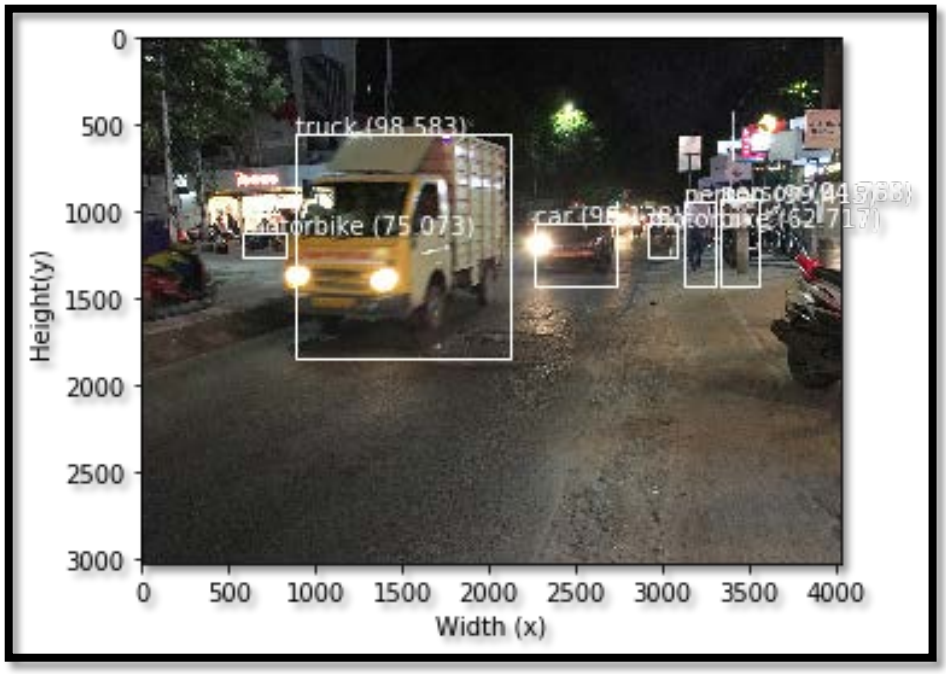


Fig 1.3 Example of vehicle detection using bounding boxes on Indian roads

Also in these extremely difficult situations, the machine normally recognizes and monitors the cars immediately in front of the camera-assisted car and only ignores the cars or misidentifies the car sizes in adjacent lanes. If an accurate 3D site model of the adjacent lanes and the history of the highway could be obtained and integrated into the method, more consistent results could be anticipated in these difficult conditions.

LITERATURE SURVEY

INTRODUCTION

The various data collected from the below literature survey was useful in various ways in developing this project. Each literature had its own advantages and disadvantages. Literature survey has helped in analysing the characteristics of various vehicle detection techniques, datasets that are available.

LITERATURE REVIEW

• J Janai, F Güney, A Behl, A Geiger. “Computer Vision for Autonomous Vehicles: Problems, Datasets and State-of-the-Art”, 18 Apr 2017, arXiv:1704.05519.

This paper attempts to narrow this gap by providing a survey on the state- of-the-art datasets and techniques. The survey includes both the historically most relevant literature as well as the current state of the art on several specific topics, including recognition, reconstruction, motion estimation, tracking, scene understanding, and end-to-end learning for autonomous driving. Towards this goal, the paper analyzes the performance of the state of the art on several challenging benchmarking datasets, including KITTI, MOT, and Cityscapes. Besides, we discuss open problems and current research challenges. This paper acts as a comprehensive guide on problems, datasets, and methods in computer vision for autonomous vehicles.

• Redmon, Joseph (2016). "You only look once: Unified, real-time object detection". Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. arXiv:1506.02640.

This paper presents YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, the authors of the paper frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single

neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance. The proposed unified architecture is extremely fast and the base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors.

• Redmon, Joseph (2017). "YOLO9000: better, faster, stronger". arXiv:1612.08242.

The paper introduces YOLO9000, a state-of-the-art, real-time object detection system that can detect over 9000 object categories. The paper proposes various improvements to the YOLO detection method, both novel and drawn from prior work. The improved model, YOLOv2, is state-of- the-art on standard detection tasks like PASCAL VOC and COCO. At 67 Frames Per Second(FPS), YOLOv2 gets 76.8 mean average precision(mAP) on VOC 2007. At 40 FPS, YOLOv2 gets 78.6 mAP, outperforming state-of-the-art methods like Faster RCNN with ResNet and SSD while still running significantly faster. The authors also propose a method to jointly train on object detection and classification. Using this method we train YOLO9000 simultaneously on the COCO detection dataset and the ImageNet classification dataset. The joint training allows YOLO9000 to predict detections for object classes that don't have labelled detection data.

• Redmon, Joseph (2018). "Yolov3: An incremental improvement". arXiv:1804.02767.

This paper proposes an enhanced model of YOLOv2. It's a little bigger than the previous model but more accurate. At 320 × 320 YOLOv3 runs in 22 ms at 28.2 mAP, as accurate as SSD but three times faster. The model achieves 57.9 AP50 in 51 ms on a Titan X, compared to 57.5 AP50 in 198 ms by RetinaNet, similar performance but 3.8 times faster.

• Ajay Kumar Boyat, Brijendra Kumar Joshi (2015). " A REVIEW PAPER: NOISE MODELS IN DIGITAL IMAGE PROCESSING". Signal & Image Processing: An International Journal (SIPIJ) Vol.6, No.2, April 2015. arXiv:1505.03489v1.

This paper presents a review of noise models that are essential in the study of image de-noising techniques. This paper expresses a brief overview of five noise models respectively Speckle, Gaussian, Poisson, Salt and Pepper. These noise models can be selected by analysis of their origin. In this way, the paper gives a complete and quantitative analysis of noise models available in digital images.

• Geethapriya. S, N. Duraimurugan, S.P. Chokkalingam (2019). "Real-Time Object Detection with Yolo". International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-8, Issue-3S, February 2019.

The objective of this paper is to detect of objects using You Only Look Once (YOLO) approach. The authors explain in detail how the algorithm looks at the image completely and predicts the bounding boxes using convolutional network. The rate of detection of objects is higher than that of the other algorithms.

• Prabhishek Singh, Raj Shree (2016). "A Comparative Study to Noise Models and Image Restoration Techniques". International Journal of Computer Applications (0975 – 8887) Volume 149 – No.1, September 2016.

In this proposed work, a comparative study analysis of simple, fast technique is given to remove noise of an image which is mostly introduced due to environmental changes or due to other issues. Researchers focus on the noise issues that changes image pixels value either on or off. To get an enough efficient method to remove the noise from the images is a greater challenge for the researchers. Noise plays an important role in degrading the image at the time of capturing or transmission of the image. There are many algorithms and filtering techniques available which have their own assumptions, merits and demerits depending upon the prior knowledge of the noise. Image smoothening is one of the most significant and widely used procedure in the image processing. Here, apart from noise a model, the light is also

thrown on comparative analysis of noise removal techniques is done. This paper will present the different noise types to an image models and investigating the various noise reduction techniques and their advantages and disadvantages.

- Kohli, P., & Chadha, A. (2019). Enabling Pedestrian Safety Using Computer Vision Techniques: A Case Study of the 2018 Uber Inc. Self-driving Car Crash. Perspectives on Asian Tourism, 261– 279. doi:10.1007/978-3-030-12388-8_19.

In this paper, They apply state-of-the-art Computer Vision models to the pedestrian safety scenario. More generally, their experimental results are an evaluation of various image enhancement and object recognition techniques for enabling pedestrian safety in low-lighting conditions using the Uber crash as a case study.

- Pandian, A. P., Ntalianis, K., & Palanisamy, R. (Eds.). (2020). Intelligent Computing, Information and Control Systems. Advances in Intelligent Systems and Computing. doi: 10.1007/978-3-030-30465-2.

This book integrates the computational intelligence and intelligent control systems to provide a powerful methodology for a wide range of data analytics issues in industries and societal applications. The recent research advances in computational intelligence and control systems are addressed, which provide very promising results in various industry, business and societal studies. This book also presents the new algorithms and methodologies for promoting advances in common intelligent computing and control methodologies including evolutionary computation, artificial life, virtual infrastructures, fuzzy logic, artificial immune systems, neural networks and various neuro-hybrid methodologies.

- P. V. Babayan, M. D. Ershov and D. Y. Erokhin, "Neural Network-Based Vehicle and Pedestrian Detection for Video Analysis System," 2019 8th Mediterranean Conference on Embedded Computing (MECO), Budva, Montenegro, 2019, pp. 1- 5.

In this paper, the authors compare various neural network architectures that are used for object detection and recognition. In this work vehicles and pedestrians are considered objects of interest. Modern artificial neural networks are able to detect and localize objects of known classes. This allows them to be used in various technical vision systems and video analysis systems. In this paper comparison between three architectures (YOLO, Faster RCNN, and SSD) by the following criteria: processing speed, mAP, precision and recall.

- Ross Girshick Fast R-CNN Microsoft Research In Proc. of the ACM International Conf. on Multimedia, 2018

This paper proposes a Fast Region-based Convolutional Network method (Fast R-CNN) for object detection. Fast R-CNN is implemented in Python and C++ (using Caffe) and is available under the open-source MIT License

- Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

In this work, they introduce a Region Proposal Network (RPN) that shares full-image. Convolutional features with the detection network, thus enabling nearly cost-free region proposals. They further merge RPN and Fast R-CNN into a single network by sharing their convolutional features— using the recently popular terminology of neural networks with “attention” mechanisms, the RPN component tells the unified network where to look. For the very deep VGG-16 model the detection system has a frame rate of 5fps (including all steps) on a GPU, while achieving state-of-the-art object detection accuracy on PASCAL VOC 2007, 2012, and MS COCO datasets with only 300 proposals per image.

INFERENCE

YOLOv3 model with its high speed, and ease of implementation make the model more suitable for Real time vehicle detection on Indian roads, Although Faster R-CNN performs better in terms of accuracy when given additional time . The models performance can be tested using images that are distorted and depict a real time scenario.

ARCHITECTURES USED FOR VEHICLE DETECTION

- Single Shot detection
- Region proposal network (RCNN, FAST RCNN, FASTER RCNN)

- Support vector machines
- YOLO

Single Shot Detection

The architecture is called the Single Shot Multibox Detector (SSD). SSD solves the issue of low resolution in YOLO by making predictions based on feature maps taken at different stages of the convolution network, it is as accurate and, in some cases, more accurate than the state-of-the-art faster-RCNN. Because the layers closest to the image have greater resolution. A convolution layer was introduced to keep the number of bounding boxes manageable. Blank filters are added to subsample the data for faster calculations. The figure 4.1 below shows how the features from different convolutional blocks are collected to form a multiscale (or multibox) detector.

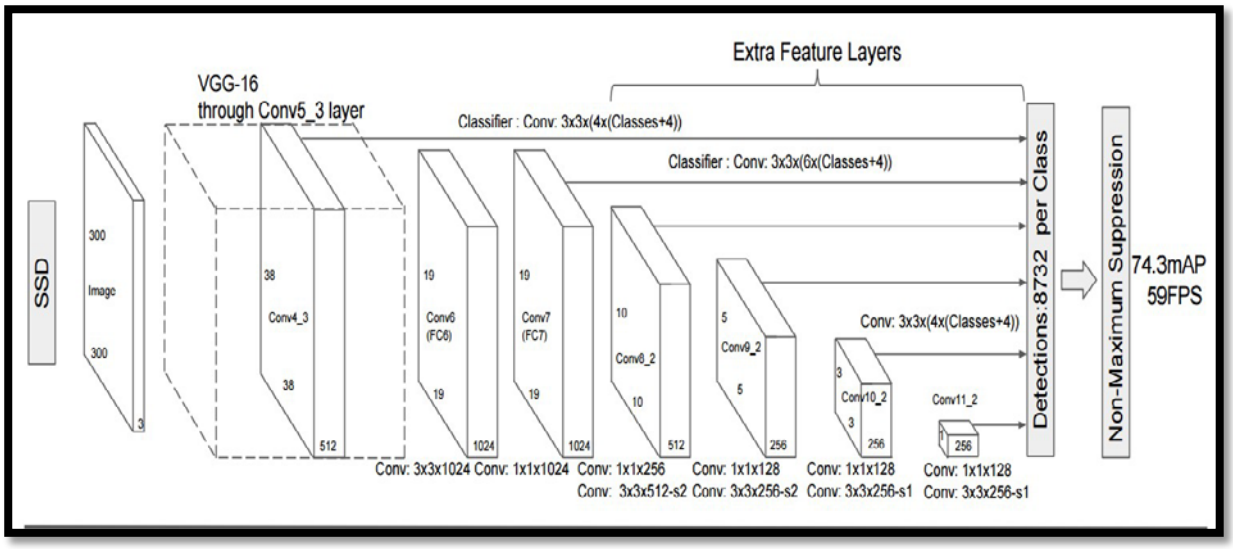


Fig 3.1 SSD Architecture

The benefit of using this approach is that the features are collected from different scales of the feature charts, and as a result the overall algorithm can identify artifacts of different scales and sizes and is more accurate than faster-RCNN. Furthermore, because all predictions are made in a single step, the SSD is much faster than the faster-RCNN

.

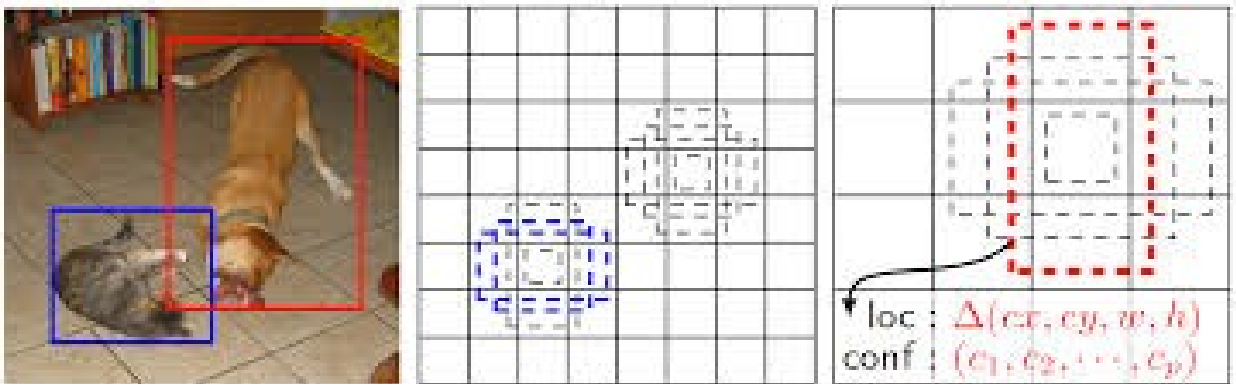


Fig 3.2 Image classification in SSD

Region proposal network (RCNN, FAST RCNN, FASTER RCNN)

RPN receives as an input image and generates a collection of object rectangles that all have score of objectivity. The RPN is planned with a network completely encompassed. As calculations are shared with a network for the detection of artifacts Faster R-CNN, it is believed that both networks share a similar collection of convolution layers. A mini network is moving through the last shared convolution layer to generate area proposals on the exit of the convolution property. It takes the convolutional property map space window n as

an input (used as n=3). All working sliding windows suit low dimension properties.

This function is composed of two fully bound box regression and layers of box classification. All of the fully connected layers in this mini network are shared in all spatial locations. This system is done by the Layer convolution and following the line of convolution of the two 1 x 1 boxes.

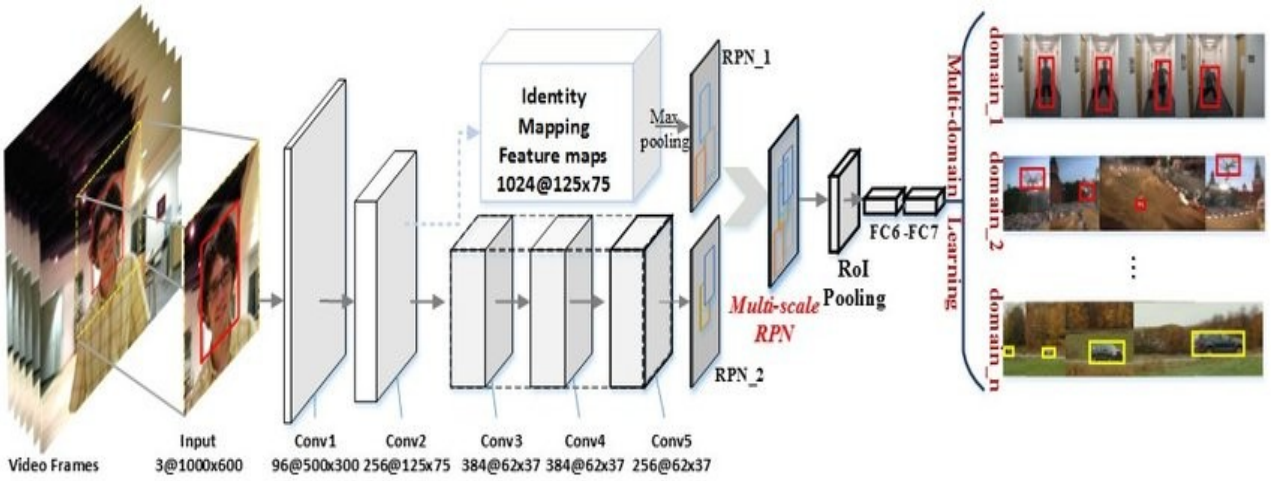


Fig 3.3 Region proposal network

Training of RPNs

In this analysis, RPNs with backpropagation and SGD are trained end-to-end. Image-centric sampling strategy is applied to train this network. All the batches in the study come from the images which include negative and positive sample anchors.

The orientation of the negative examples is realized when the missing functions of all the anchors are optimized here. For this purpose, instead it displays a random sample of anchors in an image. Accordingly, it is loaded with stacked samples when there are more positive samples.

Faster R-CNN

The R-CNN-Faster consists of two parts. The first component is a traditional network that is used to suggest zones called RPN, and the second component is the Faster R-CNN detector that uses regional propositions. The entire device comes from a single composite network, designed to detect artefacts. The first part is a traditional network which is used to propose RPN (Region Proposal

Network) regions.Fig 4.4 and 4.5 depict differences between Fast R-CNN and Faster R-CNN.

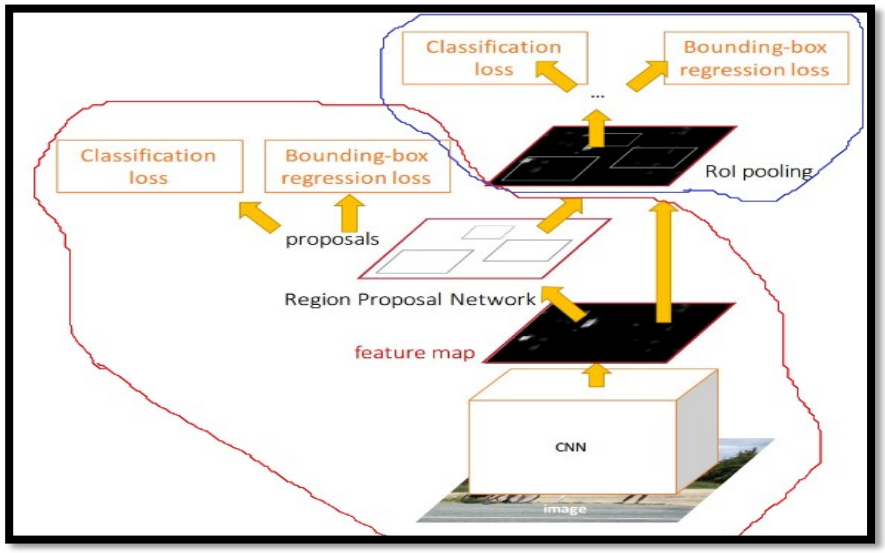
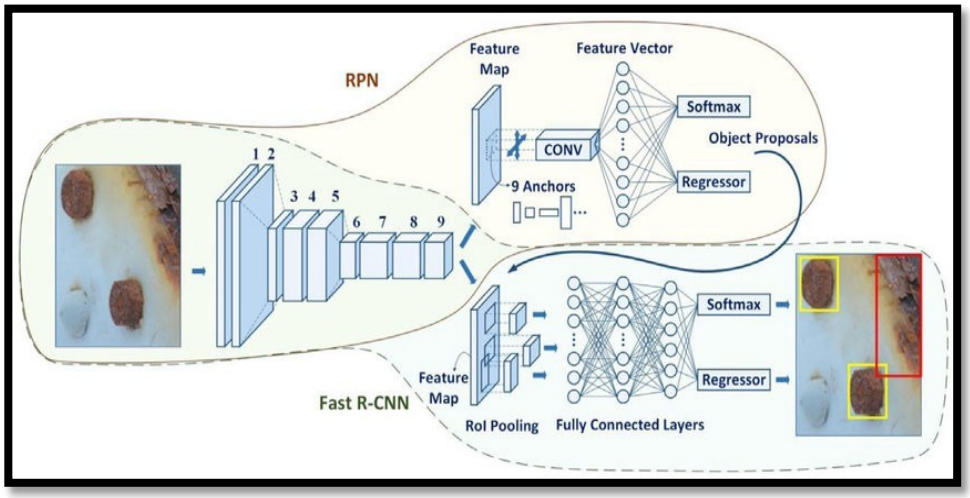


Fig 3.4 Faster RCNN



Comparing Faster R-CNN and R-CNN

Comparison of Faster R-CNN and R-CNN approaches which are advanced object detection networks focused on regional recommendation algorithms for object position definition and recognition. By increasing the training time of these detection networks in R-CNN, Faster R-CNN the produced results are more accurate. Unlike R-CNN, faster R-CNN and improved RPN need no external zone recommendations. In addition, the RPN improves the quality of the district proposal and thus improves the overall accuracy and speed of object detection.

Support Vector Machines

Using supervised learning with pre-categorized pictures, this approach uses the captured photos on real-time roads. There are photos of rears of vehicles and photos of vehicles in highways which are all 64x64 pixels.

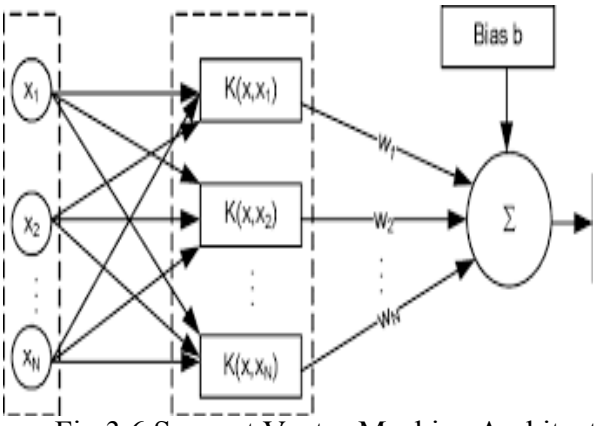


Fig 3.6 Support Vector Machine Architecture

First step is to prepare the images to extract HOG features for SVM classifier. HOG's principal aim is to classify the picture as a collection of local histograms. HOG is not invariant of size. All the training set images are 64x64 of the same dimension, so that the images can be directly used. The HOG technique records the frequency of gradient orientation of a picture's localized regions. First image is divided into small connected cells and directions are determined for each cell gradient. According to the gradient orientation each cell is divided into angular bins.

The pixel of each cell adds the gradient of the weight to the respective angular bin. Adjacent cell groups are called blocks. The clustering of cells into blocks is the basis of histogram normalization. This normalized histogram category produces the block histogram, and the descriptor represents a collection of block histograms.

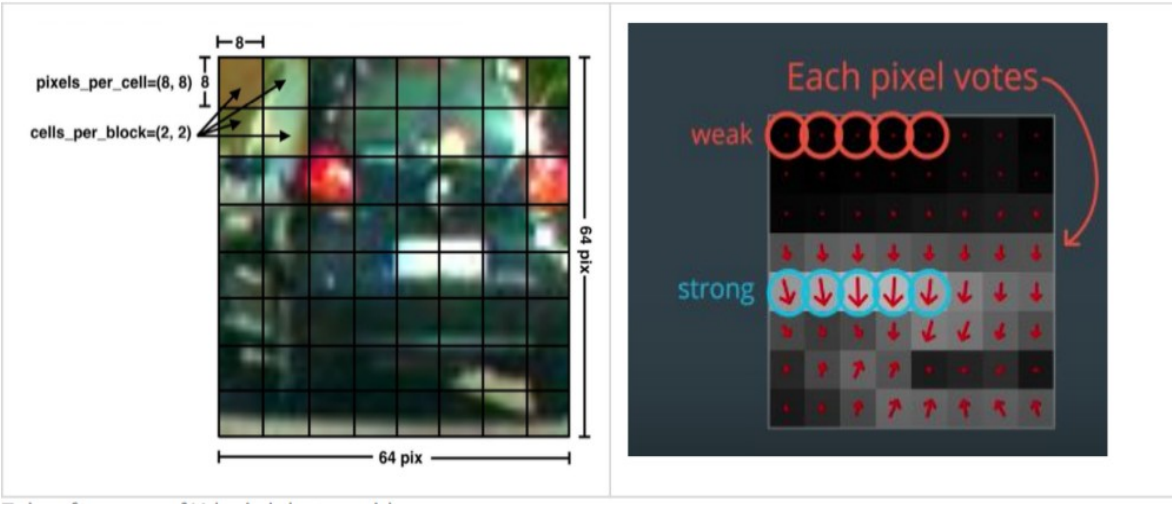


Fig 3.7 HOG and SVM on a Single Vehicle

The data is split into 2 parts with respect to 5-fold to train the model. As indicated by most learning systems, 80% is used for training and 20% is used for testing. To get better results, different parameters and colour spaces are used for the images.

Scan windows with a 64x64 pixel block is used to check the entire frame. For search windows, a 50 per cent variance is used. The SVM classifier is used for every window to detect whether there is a vehicle or not.

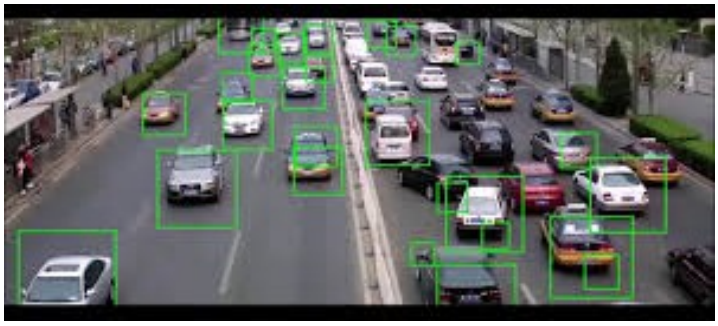


Fig 3.8 HOG and SVM approach on multiple vehicles

YOLO ALGORITHM-You Only Look Once

Introduction

Humans glance at an image and instantly know what objects are in the image, where they are, and how they interact. The human visual system is fast and accurate, allowing us to perform complex tasks like driving with little conscious thought. Fast, accurate algorithms for object detection would allow computers to drive cars without specialized sensors, enable assistive devices to convey real-time scene information to human users, and unlock the potential for general purpose, responsive robotic systems.

Current detection systems repurpose classifiers to perform detection. To detect an object, these systems take a classifier for that object and evaluate it at various locations and scales in a test image. Systems like deformable parts models (DPM) use a sliding window approach where the classifier is run at evenly spaced locations over the entire image.

More recent approaches like R-CNN use region proposal methods to first generate potential bounding boxes in an image and then run a classifier on these proposed boxes. After classification, post-processing is used to refine the bounding boxes, eliminate duplicate detections, and rescore the boxes based on other objects in the scene . These complex layers are slow and hard to optimize because each individual component must be trained separately.

We reframe object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. Using our system, you only look once (YOLO) at an image to predict what objects are present and where they are.

First, YOLO is extremely fast. Since we frame detection as a regression problem we don't need a complex pipline. We simply run our neural network on a new image at test time to detect vehicles . Furthermore, YOLO achieves more than twice the mean average precision of other real-time systems[3].

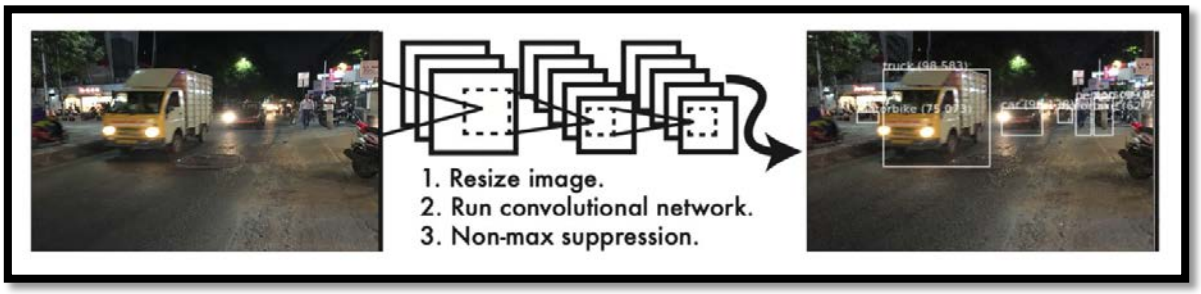


Fig 4.1 The YOLO Detection System-Processing images with YOLO

What is YOLO

It is a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

ARCHITECTURE

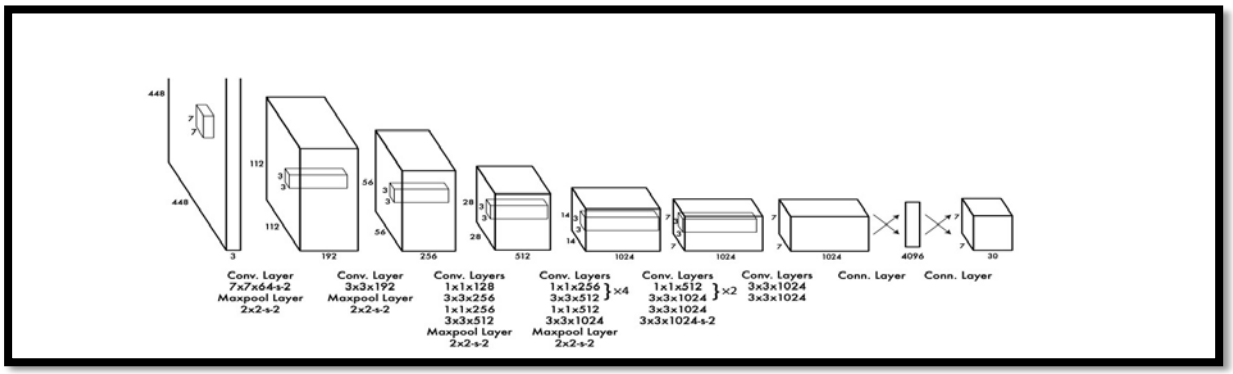


Fig 4.2 The Architecture.

The detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1 × 1 convolutional layer reduce the features space from preceding layers. [3]

The algorithm divides the image of the input into a grid of S x S. When an object's centre falls into a grid cell, then that grid cell is responsible for detecting the object. Each grid cell predicts bounding boxes B and scores trust for certain boxes. Such scores of confidence show how sure the model is that the box contains an item, and how accurate it thinks the box predicts.. Formally If no object exists in that cell, the confidence scores should be zero. Otherwise it is required that the confidence score need to be equal to the intersection over union (IOU) between the predicted box and the ground truth.

Each bounding box consists of 5 predictions: x, y, w, h, and confidence. The (x, y) coordinates represent the centre of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image. Finally the confidence prediction represents the IOU between the predicted box and any ground truth box [3].

Each grid cell also predicts C conditional class probabilities. These probabilities are conditioned on the grid cell containing an

object. We only predict one set of class probabilities per grid cell, regardless of the number of boxes B.

Consider the below example, an image is taken and divided into a 3 x 3 grid. Each grid is labelled and undergoes both image classification and objects localization techniques. The label is considered as Y and consists of 8 values.

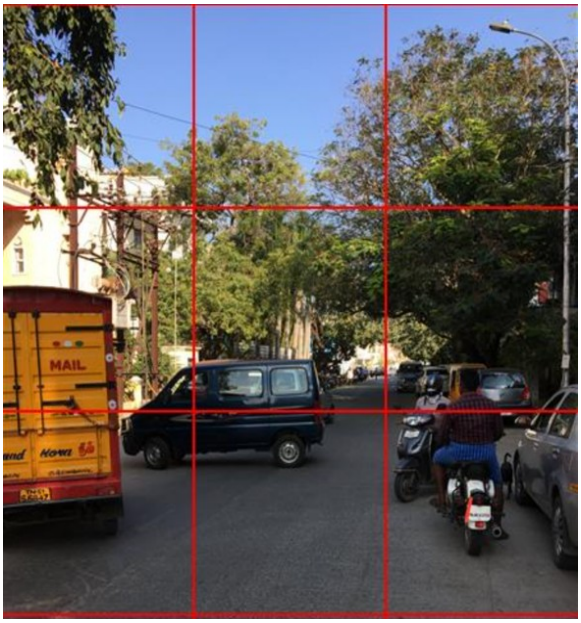


Fig 4.3 Example Image with 3x3 Grids

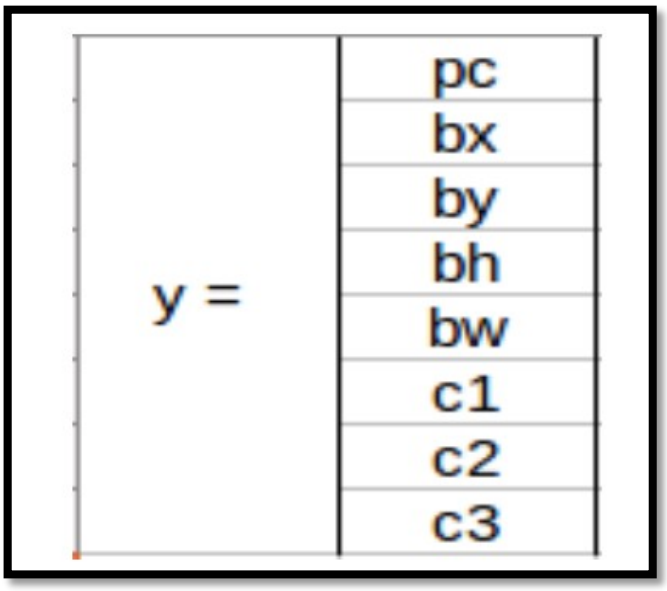


Fig 4.4 Bounding box and Class values of grid

Pc – Represents whether an object is present in the grid or not. If present pc=1 else 0. bx, by, bh, bw – are the bounding boxes of the objects (if present). c1, c2, c3 – are the classes. If the object is a car then c1 and c3 will be 0 and c2 will be 1.In the fig 5.3, the first grid contains no proper object. So it the pc value is 0. [6]

In fig 5.5, 1 represents the presence of an object. And bx, by, bh, bw are the bounding boxes of the object in the 6th grid. The object in that grid is a car so the classes are (0, 1, 0). The matrix form of Y in this is Y=3x3x8.

Fig 4.5 Bounding box and Class values of grid 6.

For the 5th grid also the matrix will be similar with different bounding boxes depending upon the objects position in the corresponding grid. If two or more grids have the same object, then the object's centre point will be identified and the grid that has that point will be taken. In IoU, the real and expected bounding box value is taken and the IoU of two boxes is determined using the formulae.

IoU = Area of Intersection / Area of Union.

If the value of IoU is more than or equal to the threshold value (0.5) then it's a good prediction. The threshold value is just an assumption. We can also take greater threshold value to increase the accuracy or for better prediction of the object.

The other method is Non-max suppression, the high probability boxes are taken and the boxes with high IoU are suppressed. Repeat this until a box is selected and consider that as the bounding box for that object. [6]

ANCHOR BOX

By using Bounding boxes for object detection, only one object can be identified by a grid. So, for detecting more than one object we go for an Anchor box.

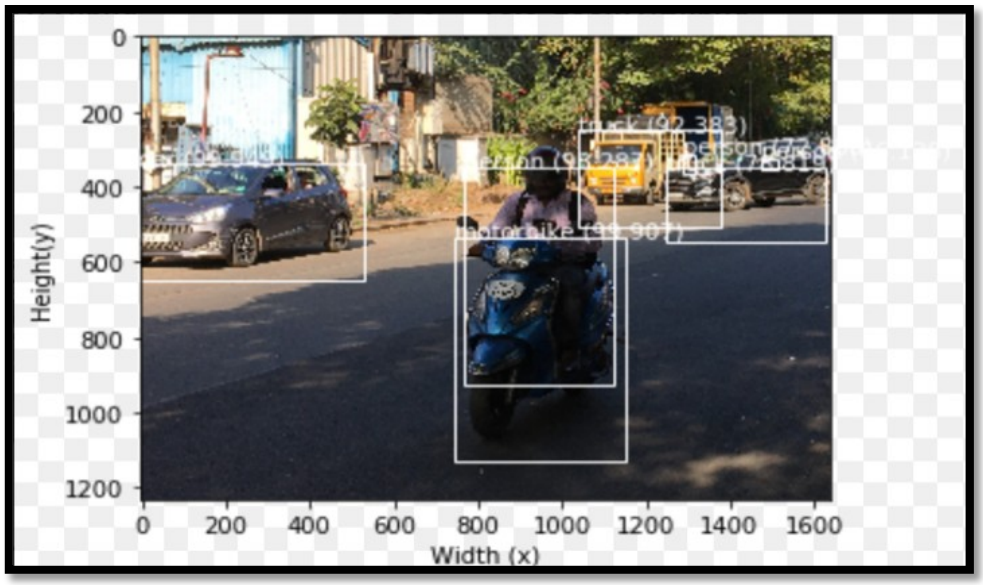


Fig 4.6 Example Image for Anchor Box

Consider a scenario in which both the human and the car's midpoint come under the same grid cell. For this case, we use the anchor box method. The red colour grid cells are the two anchor boxes for those objects. Any number of anchor boxes can be used for a single image to detect multiple objects. In our case, we have taken two of the anchor boxes.

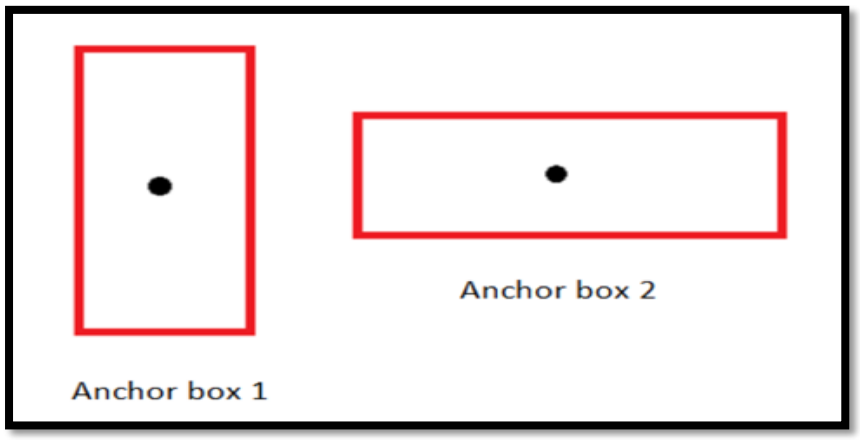


Fig 4.7 Anchor Boxes

The fig 5.7 represents the image's anchor box which we considered. For this method of identification of overlapping objects, label Y includes 16 values i.e. the values in both anchor boxes Pc in both anchor boxes reflect the object's presence. The bx, by, bh, bw in both anchor box reveals their respective boundary box metrics.[6]

Limitations of YOLO

YOLO imposes strong spatial constraints on bounding box predictions since each grid cell only predicts two boxes and can only have one class. This spatial constraint limits the number of nearby objects that our model can predict. The model performs with lower accuracy on small objects that appear in groups, such as flocks of birds.

The model also uses relatively coarse features for predicting bounding boxes since our architecture has multiple down sampling layers from the input image.

Finally, while training on a loss function that approximates detection performance, The loss function treats errors the same as in small bounding boxes versus large bounding boxes. A small error in a large box is generally benign but a small error in a small box has a much greater effect on IOU. The main source of error is incorrect localizations. [3]

FASTER RCNN

Introduction

Recently, deep ConvNets have significantly improved image classification and object detection accuracy. Compared to image classification, object detection is a more challenging task that requires more complex methods to solve. Due to this complexity, current approaches train models in multi-stage pipelines that are slow and inelegant. Complexity arises because detection requires the accurate localization of objects, creating two primary challenges. First, numerous candidate object locations (often called “proposals”) must be processed. Second, these candidates provide only rough localization that must be refined to achieve precise localization. Solutions to these problems often compromise speed, accuracy, or simplicity.

The proposed single-stage training algorithm that jointly learns to classify object proposals and refine their spatial locations, can train a very deep detection network (VGG16) 9 times faster than R-CNN and 3 times faster than Spatial pyramid pooling network (SPPnet). At runtime, the detection network processes images in 0.3s (excluding object proposal time).

The Region-based Convolutional Network method (R- CNN) [9] achieves excellent object detection accuracy by using a deep ConvNet to classify object proposals. R-CNN, however, has notable drawbacks:

1. Training is a multi-stage pipeline. R-CNN first fine-tunes a ConvNet on object proposals using log loss. Then, it fits SVMs to ConvNet features. These SVMs act as object detectors, replacing the softmax classifier learnt by fine-tuning. In the third training stage, bounding-box regressors are learned.

2. Training is expensive in space and time. For SVM and bounding-box regressor training, features are extracted from each object proposal in each image and written to disk. With very deep networks, such as VGG16, this process takes 2.5 GPU-days for the 5k images of the VOC07 train/val set. These features require hundreds of gigabytes of storage.

3. Object detection is slow. At test-time, features are extracted from each object proposal in each test image. Detection with VGG16 takes 47s / image (on a GPU).

R-CNN is slow because it performs a ConvNet forward pass for each object proposal, without sharing computation. Spatial pyramid pooling networks (SPPnets) were proposed to speed up R-CNN by sharing computation.

FAST RCNN

A Fast Region-based Convolutional Network method (F-RCNN) for object detection. Fast R-CNN builds on previous work to efficiently classify object proposals using deep convolutional networks. Fast R-CNN trains the very deep VGG16 network 9 times faster than R-CNN and achieves a higher MAP on PASCAL VOC 2012. Compared to SPPnet, Fast R-CNN trains VGG16 3 times faster, tests 10 faster, and is more accurate.

ARCHITECTURE

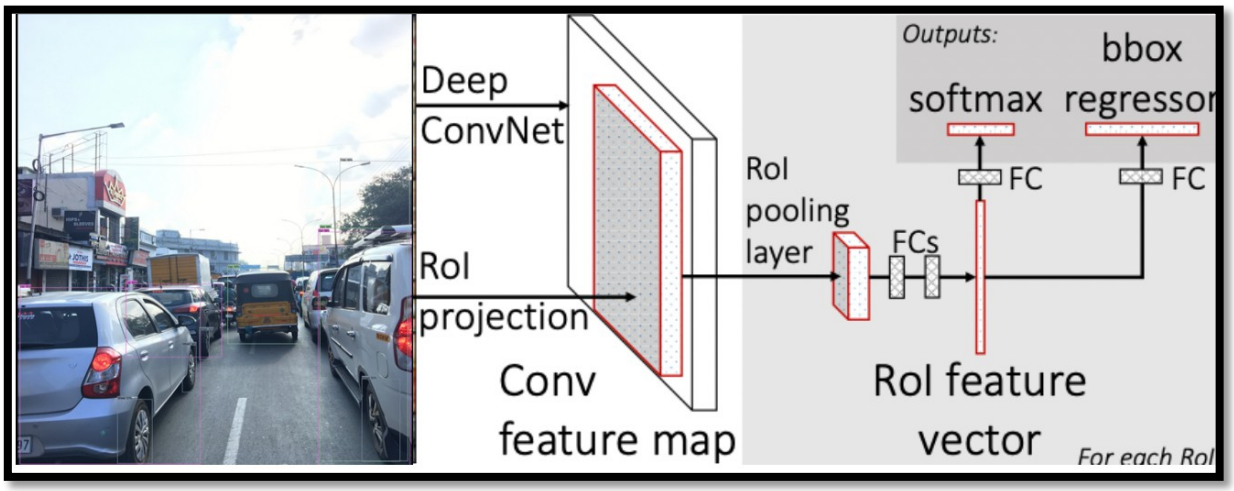


Fig 5.1 F-RCNN architecture.

An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss. The RoI pooling layer uses max pooling to convert the features inside any valid region of interest into a small feature map with a fixed spatial extent of H × W (e.g., 7 × 7), where H and W are layer hyper-parameters that are independent of any particular RoI. A RoI is a rectangular window into a conv feature map. Each RoI is defined by a tuple (r, c, h, w) that specifies its top-left corner (r, c) and its height and width (h, w). RoI max pooling works by dividing the h × w RoI window into an H × W grid of sub-windows of approximate size (h/H) × (w/W) and then max-pooling the values in each sub-window into the corresponding output grid cell. Max pooling is applied independently to each feature map channel. The RoI layer is simply the special-case of the spatial pyramid pooling layer used in SPPnets in which there is only one pyramid level.

First, the last max pooling layer is replaced by a RoI pooling layer that is configured by setting H and W to be compatible with the net's first fully connected layer (e.g., H =W =7forVGG16).

Second, the network's last fully connected layer and soft- max (which were trained for 1000-way ImageNet classification) are replaced with the two sibling layers described earlier (a fully connected layer and softmax over K + 1 categories and category-specific bounding-box regressors).

Third, the network is modified to take two data inputs: a list of images and a list of RoIs in those images.

Training all network weights with back-propagation is an important capability of F-RCNN.

The root cause is that back-propagation through the SPP layer is highly inefficient when each training sample (i.e. RoI) comes from a different image, which is exactly how R-CNN and SPPnet networks are trained. The inefficiency stems from the fact that each RoI may have a very large receptive field, often spanning the entire input image. Since the forward pass must process the entire receptive field, the training inputs are large (often the entire image).

In F-RCNN training, stochastic gradient descent (SGD) mini- batches are sampled hierarchically, first by sampling N images and then by sampling R/N RoIs from each image. Critically, RoIs from the same image share computation and memory in the forward and backward passes. Making N small decreases mini-batch computation. For example, when using N = 2

and R = 128, the proposed training scheme is roughly 64× faster than sampling one RoI from 128 different images (i.e., the F-RCNN and SPPnet strategy).

One concern over this strategy is it may cause slow training convergence because RoIs from the same image are correlated. This concern does not appear to be a practical issue and we achieve good results with N = 2 and R = 128 using fewer SGD iterations than R-CNN.

In addition to hierarchical sampling, Fast R-CNN uses a streamlined training process with one fine-tuning stage that jointly optimizes a softmax classifier and bounding-box regressors, rather than training a softmax classifier, SVMs, and regressors in three separate stages [9, 11].

Once a F-RCNN network is fine-tuned, detection amounts to little more than running a forward pass (assuming object proposals are pre-computed). The network takes as input an image (or an image pyramid, encoded as a list of images) and a list of R object proposals to score.

Test-time, R is typically around 2000, although we will consider cases in which it is larger (≈ 45k). When using an image pyramid, each RoI is assigned to the scale such that the scaled RoI is closest to 2242 pixels in area.
For each test RoI r, the forward pass outputs a class posterior probability distribution p and a set of predicted bounding-box offsets relative to r (each of the K classes gets its own refined bounding-box prediction).

For whole-image classification, the time spent computing the fully connected layers is small compared to the conv layers. On the contrary, for detection the number of RoIs to process is large and nearly half of the forward pass time is spent computing the fully connected layers (see Fig. 2). Large fully connected layers are easily accelerated by compressing them with truncated SVD.

In this technique, a layer parameterized by the u × v weight matrix W is approximately factorized as

$W \approx U\Sigma_t V^T$

Using SVD. In this factorization, U is a u × t matrix comprising the first t left- singular vectors of W , Σt is a t × t diagonal matrix containing the top t singular values of W , and V is v × t matrix comprising the first t right-singular vectors of W . Truncated SVD reduces the parameter count from uv to t (u + v), which can be significant if is much smaller than min (u, v). To compress a network, the single fully connected layer corresponding to W is replaced by two fully connected layers, without a non-linearity between them. The first of these layers uses the weight matrix $\Sigma_t V^T$ (and no biases) and the second uses U (with the original biases associated with W). This simple compression method gives good speedups when the number of RoIs is large.

Anchors

At each sliding-window location, multiple region proposals are predicted. So the reg layer has 4k outputs encoding the coordinates of k boxes, and the cls layer outputs 2k scores that estimate probability of object or not object for each proposal. The k proposals are parameterized relative to k reference boxes, which are called anchors. An anchor is centered at the sliding window , and is associated with a scale and aspect ratio. By default, 3 scales and 3 aspect ratios are used, yielding k = 9 anchors at each sliding position. For a convolutional feature map of a size W × H (typically ~2,400), there are W x H x k anchors in total.

Limitations of F-RCNN

Training an RCNN model is expensive and slow due to the following:

- Extracting 2,000 regions for each image based on selective search
- Extracting features using CNN for every image region. Suppose there are N images, then the number of CNN features will be N*2,000
- The entire process of object detection using RCNN has three models:

1. CNN for feature extraction
2. Linear SVM classifier for identifying objects
3. Regression model for accurately detecting the bounding boxes.

All these processes combine to make RCNN very slow. It takes around 40-50 seconds to make predictions for each new image, which

essentially makes the model cumbersome and practically impossible to build when faced with a gigantic dataset.

COCO Dataset

Introduction

Coco is a dataset with the goal of advancing the state-of-the-art in object recognition by putting the topic of object recognition within the framework of the wider scene comprehension. This is accomplished by capturing photographs of complex daily scenes in their natural context, including common objects. Artifacts are marked using segmentations per instance to aid in specific object localization. The dataset contains photos of 91 objects types. With a total of 2.5 million labelled instances in 328k images.

Importance of COCO Dataset

One of computer vision's key goals is to understand the visual scenes. Scene comprehension includes various tasks including identifying what objects are present, locating objects in 2D and 3D, assessing the characteristics of the objects and the environment, characterizing object relationships, and providing a descriptive explanation of the situation. The real description and identification of objects and detection datasets help in exploring the first challenges related to scene understanding. For instance the ImageNet dataset , which contains an unprecedented number of images, has recently enabled breakthroughs in both object classification and detection research. The community has also created datasets containing object attributes, scene attributes, key points, and 3D scene information.

Features of COCO Dataset

The Microsoft Common Objects in Context (MS COCO) dataset contains 91 common object categories with 82 of them having more than 5,000 labelled instances, In total the dataset has 2,500,000 labelled instances in 328,000 images. In contrast to the popular ImageNet dataset, COCO has fewer categories but more instances per category. This can aid in learning detailed object models capable of precise 2D localization. The dataset is also significantly larger in number of in- stances per category than the PASCAL VOC and SUN datasets. Additionally, a critical distinction between our dataset and others is the number of labelled instances per image which may aid in learning contextual information. MS COCO contains considerably more object instances per image as compared to ImageNet (3.0) and PASCAL (2.3). In contrast, the SUN dataset, which contains significant contextual information, has over 17 objects per image but considerably fewer object instances overall.

The basis for COCO Dataset Creation

Image Classification

The task of object classification requires binary labels indicating whether objects are present in an image. Early datasets of this type comprised images containing a single object with blank backgrounds, such as the MNIST handwritten digits or COIL household objects. Popular datasets in the machine learning community due to the larger number of training examples, CIFAR-10 and CIFAR-100 offered 10 and 100 categories from a dataset of tiny 32 × 32 images. While these datasets contained up to 60,000 images and hundreds of categories, they still only captured a small fraction of our visual world [4].

Common Object Categories

The selection of object categories is a non-trivial exercise. The categories must form a representative set of all categories, be relevant to practical applications and occur with high enough frequency to enable the collection of a large dataset. Other important decisions are whether to include both “thing” and “stuff” categories and whether fine-grained and object-part categories should be included. “Thing” categories include objects for which individual instances may be easily labelled (person, chair, and car) where “stuff” categories include materials and objects with no clear boundaries (sky, street, grass). For precise localization of object instances, developers decided to only include “thing” categories and not “stuff.” The specificity of object categories can vary significantly. For instance, a dog could be a member of the “mammal”, “dog”, or “German shepherd” categories. To enable the practical collection of a significant number of instances per category, developers chose to limit our dataset to entry- level categories, i.e. category labels that are commonly used by humans when describing objects (dog, chair, and person). It is also possible that some object categories may

be parts of other object categories. For instance, a face may be part of a person

Object Detection

Detecting an object entails both stating that an object belonging to a specified class is present, and localizing it in the image. The location of an object is typically represented by a bounding box. Early algorithms focused on face detection using various ad hoc datasets. Later, more realistic and challenging face detection datasets were created. Another popular challenge is the detection of pedestrians for which several datasets have been created.. Since the detection of many objects such as sunglasses, cell phones or chairs is highly dependent on contextual information, it is important that detection datasets contain objects in their natural environments. The use of bounding boxes also limits the accuracy for which detection algorithms may be evaluated.

Semantic Scene Labelling

The task of labelling semantic objects in a scene requires that each pixel of an image be labelled as belonging to a category, such as sky, chair, floor, street, etc. In contrast to the detection task, individual instances of objects do not need to be segmented. This enables the labelling of objects for which individual instances are hard to define, such as grass, streets, or walls

TESTING OF IMAGES BEFORE DISTORTION

Fig 7.1 IMG_1

Fig 7.2 IMG_1 after Detection

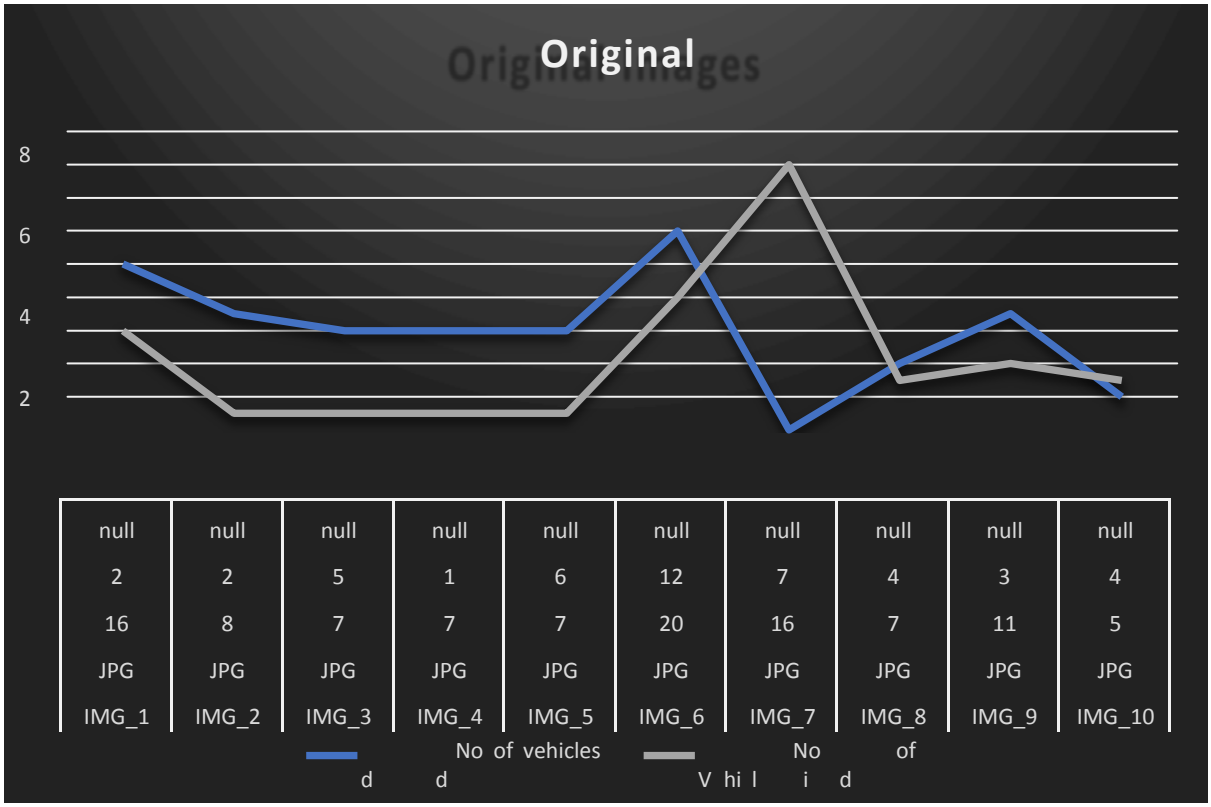


Fig 7.3 Graph for vehicles detected and Missed

NOISE IN IMAGE PROCESSING

Image noise is the random variation of the brightness or colour in the captured images. It is image signal degradation caused by external sources. Images containing multiplicative noise have the characteristic that the brighter the area, the noisier it is. But this is mostly an additive. A noisy image can be modelled as A (x, y) = H (x, y) + B (x, y)

Where, A (x, y) = function of noisy image, H (x, y) = function of image noise, B (x, y) = function of original image.

Noise in digital images tells the unwanted information. Noise produces undesirable effects such as artifacts, unrealistic edges, unseen lines, corners, blurred objects and disturbs background scenes. Prior learning of noise models is essential to further processing to reduce these undesirable effects. Digital noise can come from various sources, such as Charge Coupled Device (CCD) and Complementary Semiconductor Metal Oxide (CMOS) sensors. For timely, complete, and quantitative analysis of noise models, points spreading function (PSF) and modulation transfer function (MTF) have been used in some context.

TYPES OF NOISES USED

During the testing period, Seven types of noises were induced to the test images. The types of Noises that were used are Gaussian, Local Variance, Pepper, Poisson, Salt, Salt and Pepper and Speckle Noise models. The noise induced images were converted the into greyscale images and were also quantised into 4- bit and 6-bit Quantised images to understand their effects on the detection accuracy of our model during unconventional conditions.

GAUSSIAN NOISE MODEL

Gaussian Noise is also referred to as electronic noise because it occurs in amplifiers or sensors. Gaussian noise caused by natural sources like thermal vibration of atoms and the discrete nature of warm object radiation. Gaussian noise in digital images generally disturbs the grey values, Which is why Gaussian noise model is designed by its probability distribution function (PDF) or standardises histogram in terms of grey value.

The Probability Distribution Function (PDF) of Gaussian Noise is represented by the following mathematical equation,

$$P(z) = \frac{1}{2\pi\sigma_{„}\,2\sigma^2}(z-\mu)^2\ e$$

In this noise model, the mean value is zero, variance is 0.1 and 256 grey levels in terms of its Probability Distribution Function (PDF), which can be interpreted form the graph below.

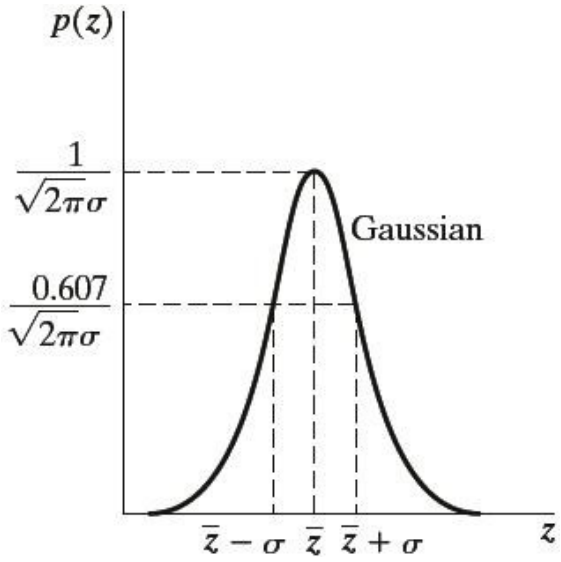


Fig 8.1 PDF of Gaussian Noise

Because of this equal randomness, the normalised Gaussian noise curve appears in bell-shaped form. The PDF of this noise model shows that 70 to 90 per cent noisy pixel values of degraded image in the range between μ - σ and μ + σ Normalized histogram shape in spectral domain is almost the same.

Fig 8.2 Image Before and after applying Gaussian Noise

LOCAL VARIANCE

Local variance may be used to assess characteristics of the local image. Local variance is used to assess contrast with the local image. In the image filtering process, local variance and mean approach can be used to control filtering degree. Regional heterogeneity can also be used for study of image texture. Average local variance (ALV) method, where ALV for a moving window is the mean of the standard deviation. Local variance is mainly affected in the technique of lossy compression hence the technique of loss compression is considered in this paper. Wavelet transform coding for image compression is used as image compression technique. A local variance transition due to JPEG and wavelet compression is where DCT and wavelet coefficients are believed to be accompanied by Laplacian distribution. Due to various compression

methods, the ratio between the expected value of local variances is used before and after compression for analysis of local variance variations. Blurring effect is also analysed by applying a blurred image to the algorithm. Local Variance is calculated using the mathematical equation given below,

$$\sigma^2 = E[X^2] - (E[X])^2$$

Fig 8.3 Image after applying Local Variance

POISSON NOISE MODEL

The noise's presence is seen because of the statistical existence of electromagnetic waves such as x-rays, visible lights, and gamma rays. The origins of x-rays and gamma rays released count of photons per unit time. From its source, these rays are absorbed into the patient's body, into medical x rays and imaging devices for gamma rays. These sources have random photons fluctuation. The result has spatial and temporal randomness in the image collected. Also, this noise is referred to as quantum (photon) noise or shot noise. The Probability density function of the Poisson noise is,

P(

λ is the shape parameter which indicates the average number of events in the given time interval.

Th41e following is the plot of the Poisson probability density function for four values of λ.

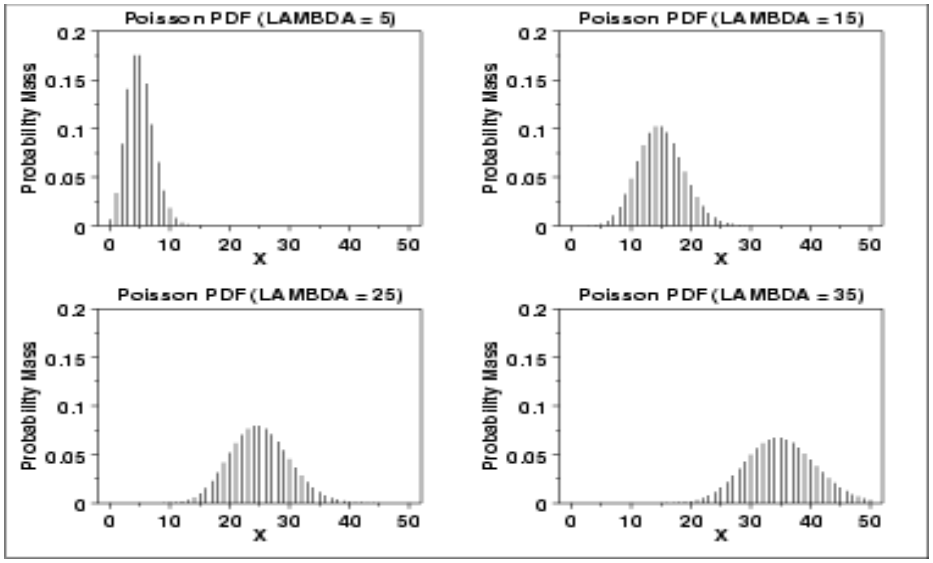


Fig 8.4The PDF of Poisson Noise

Fig 8.5 Image after applying Poisson Noise

SALT AND PEPPER NOISE MODEL

Salt and pepper noise are also called noise data drop, because the original data values are dropped statistically. This noise is also known as the noise from salt and pepper. However, the image isn't completely corrupted by salt and pepper noise instead of changing some pixel values in the image.

Although there are possibilities of some neighbours not changing in the noisy image this noise is seen in the transmission of data. Image pixel values are replaced by corrupted pixel values either maximum or minimum pixel value i.e., 255 or 0 respectively, if the number of bits is 8 for transmission. In this connection, the dead pixels are either dark or bright.

So gradually dark pixel values are present in a bright region in a salt and pepper noise and vice versa. The probability density function of Salt and pepper noise is given by the mathematical equation,

$$P(z) = \begin{cases} Pa \; for \; z = a \\ +Pb \; for \; z = b8 \\ 0 \; otherwise \end{cases}$$

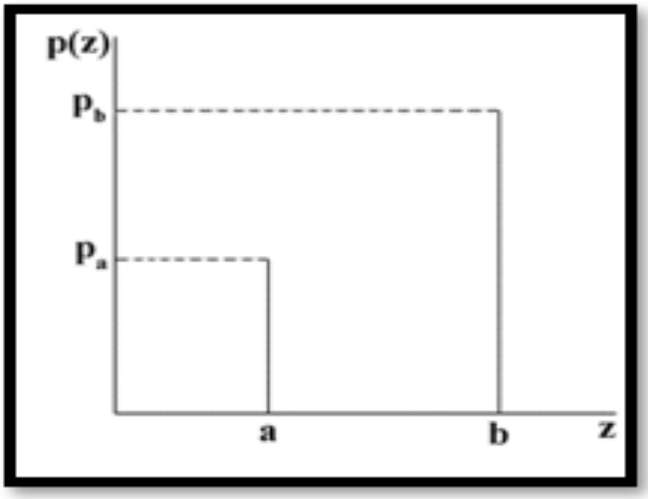


Fig 8.6 The PDF of Salt and Pepper Noise

If zero is the mean, and 0.05 is the variance. In fig 9.6 there are two spikes, one for the bright region (where the grey level is lower) called 'region a' and the other one for the dark region (where the grey level is large) called 'region b,' The PDF values in 'region a' and 'region b' are minimum and maximum.

Fig 8.7 Image after applying (a) Salt and (b) Pepper Noise

Fig 8.8 Image after applying Salt and Pepper Noise

SPECKLE NOISE MODEL

Speckle Noise can be seen in a coherent imaging system like laser, radar, and acoustics etc. Speckle noise may be similar to the Gaussian noise in an image. Its function with probability density follows the gamma distribution which is given below,

$F(g) = \%g\alpha"\$e"a$

$\alpha - 1!\ a\alpha$

The probability distribution graph for gamma distribution function is given below,

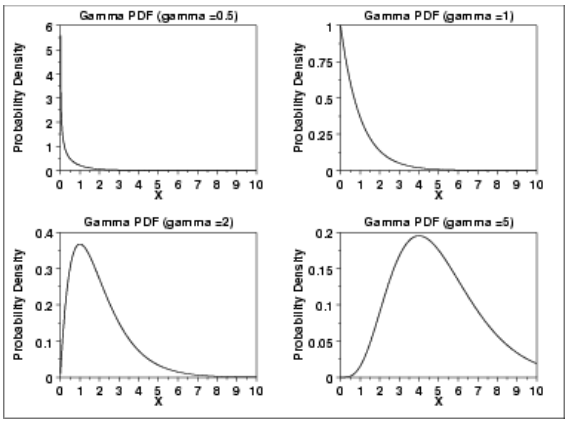


Fig 8.9 The PDF of Speckle Noise i.e. Gamma Distribution for various values of gamma

Fig 8.10 Image after applying Speckle Noise

QUANTISATION IN IMAGE PROCESSING

In image processing, quantization is a loss compression technique that is achieved by compressing a range of values to a single quantity. When decreasing the number of discrete symbols in a given stream, the stream becomes more compressible.

For example, reducing the number of colours required to represent a digital image allows its file size to be reduced.

The simple explanation is to convert an image to its digital form, we have to sample the function in both coordinates and in amplitude. Digitizing the coordinate values is called sampling. Digitizing the amplitude values is called quantization. Quantization, involved in image processing, is a lossy compression technique achieved by compressing a range of values to a single quantum value. When the number of discrete symbols in a given stream is reduced, the stream becomes more compressible. For example, reducing the number of colors required to represent a digital image makes it possible to reduce its file size. Specific applications include DCT data quantization in JPEG and DWT data quantization in JPEG 2000

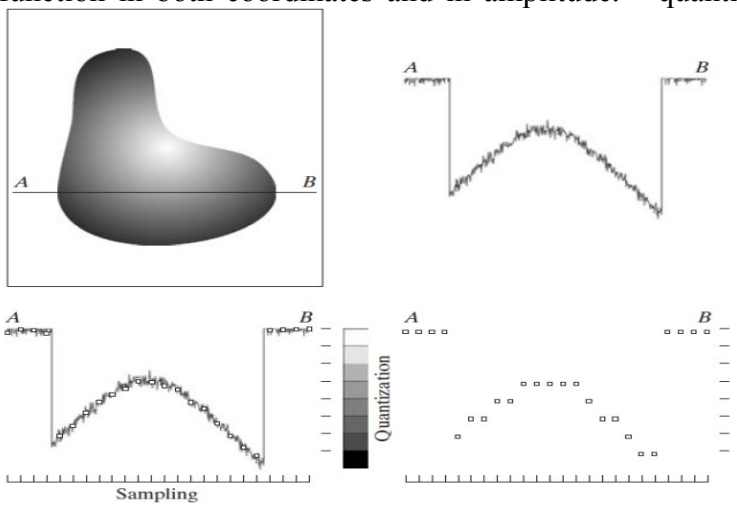


Fig 8.11 Generating digital image. (a) Continuous image. (b) illustrates the concepts of sampling and quantization. (c) Sampling and quantization. (d) Digital scan line. Ref: Digital Image Processing by Gonzalez.

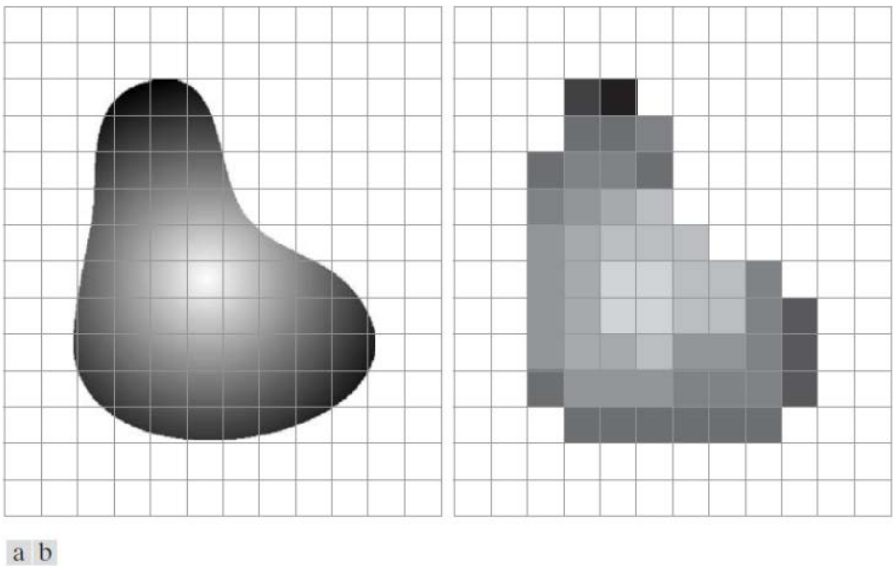


Fig 8.12 (a) Continuous image projected onto a sensor array. (b) Result of image sampling and quantization.

COLOUR QUANTISATION

Colour quantization or colour image quantization is applied to colour spaces; it is a method that decreases the number of distinct colours used in a picture, usually with the goal that the new picture will be as visually identical as possible to the original picture. Colour quantization is important for viewing images of several colours on computers that can only show a small number of colours, typically due to memory constraints, which allows for efficient compression of some image types. Most standard techniques treat colour quantization as a clustering point problem in three-dimensional space, where the dots represent colours contained in the original image, and the three axes represent the three colour channels.

The quantization of colours can be extended to almost every three-dimensional clustering algorithm, and vice versa. Usually the points in each cluster are combined after the clusters are positioned to get the representative colour to which all colours in that cluster are mapped. The three colour channels are typically red, green, and blue, but the Lab colour space is another common option, where the Euclidean distance is more compatible with the perceptual difference.

Fig 8.13 4-bit and 6-bit quantisation applied on Image.

TESTING YOLO WITH THE DISTORTED IMAGES

To test the performance of the YoloV3 algorithm in un-natural circumstances we applied the above-mentioned distortions and noises to our ten test images. We recorded the observation. First, to find the accuracy of vehicle detection we considered two metrics which were number of vehicles that were originally present in the image and the number of vehicles that were detected by Yolo, similarly we considered the same metrics for human detection. Without applying any sort of distortion, we were able to achieve an accuracy of 61% .We zoomed into two random sections of the original to image to observe any change in the detection accuracy. In the case of zoomed images we were able to achieve an accuracy of 67%.

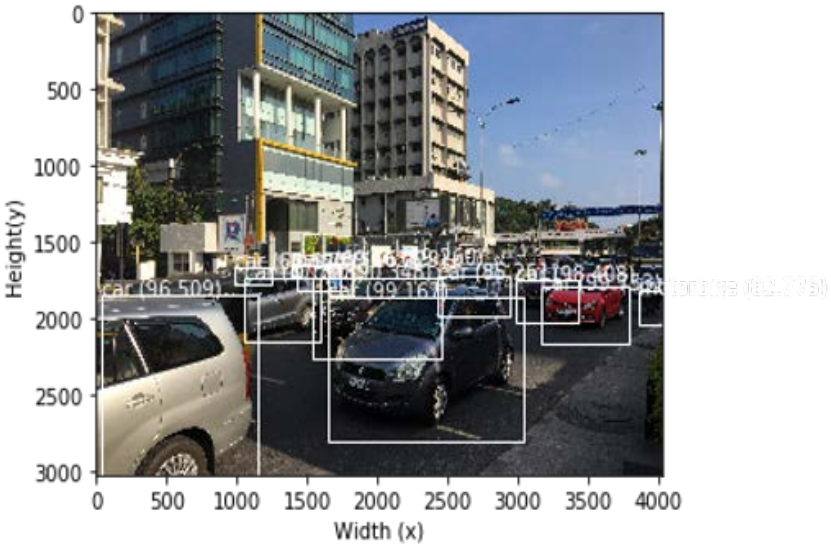


Fig 9.1 Detection using YOLO on Distorted Image

YOLO ON GAUSSIAN NOISE

The Performance of YOLO algorithm on Gaussian noise induced images is comparatively less as the more the noise is induced, the less the vehicles are detected because of the degradation of the quality of image.

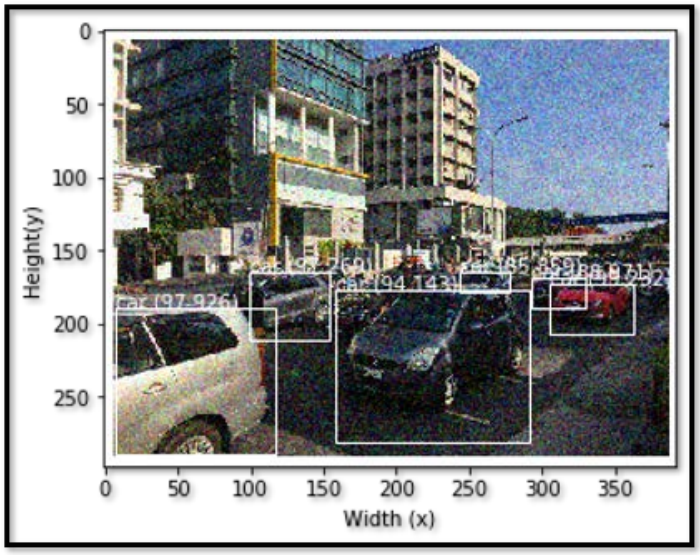


Fig 9.2 YOLO on Gaussian Noise Induced Image

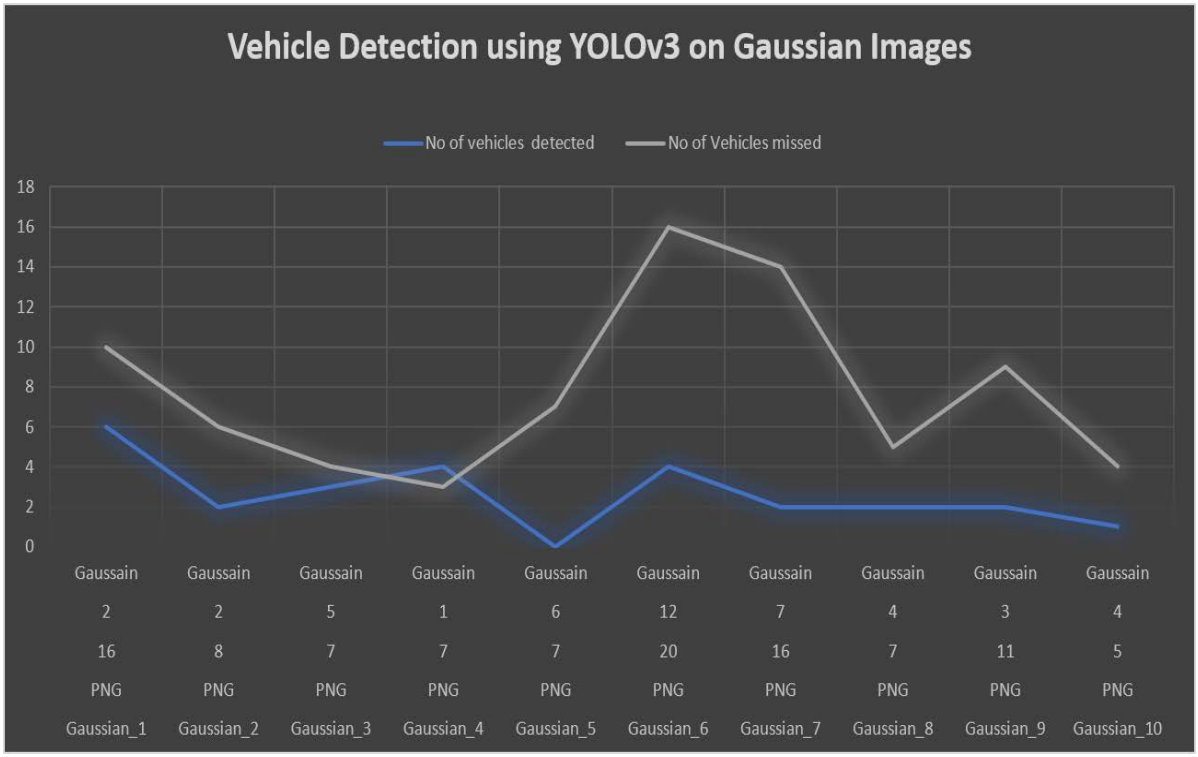


Fig 9.3 Performance of YOLO on our Gaussian induced test set

YOLO ON LOCAL VARIANCE

The performance of the YOLO algorithm on images induced by Local Variance noise is comparatively less as the more induced the noise, the less detected the vehicles due to the deterioration of the image quality.

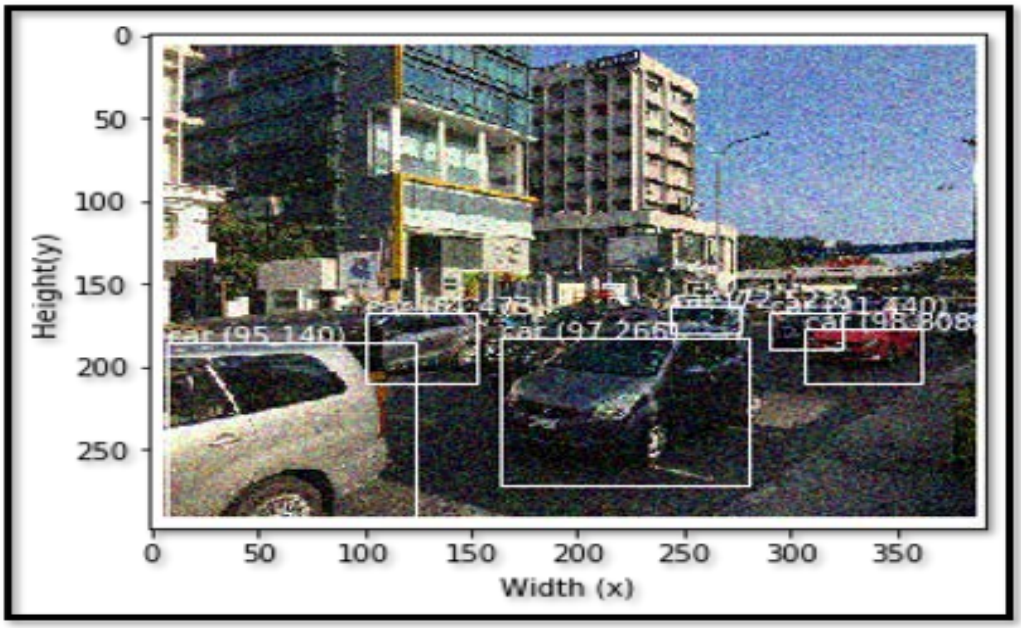


Fig 9.4 YOLO on Local Variance induced Image

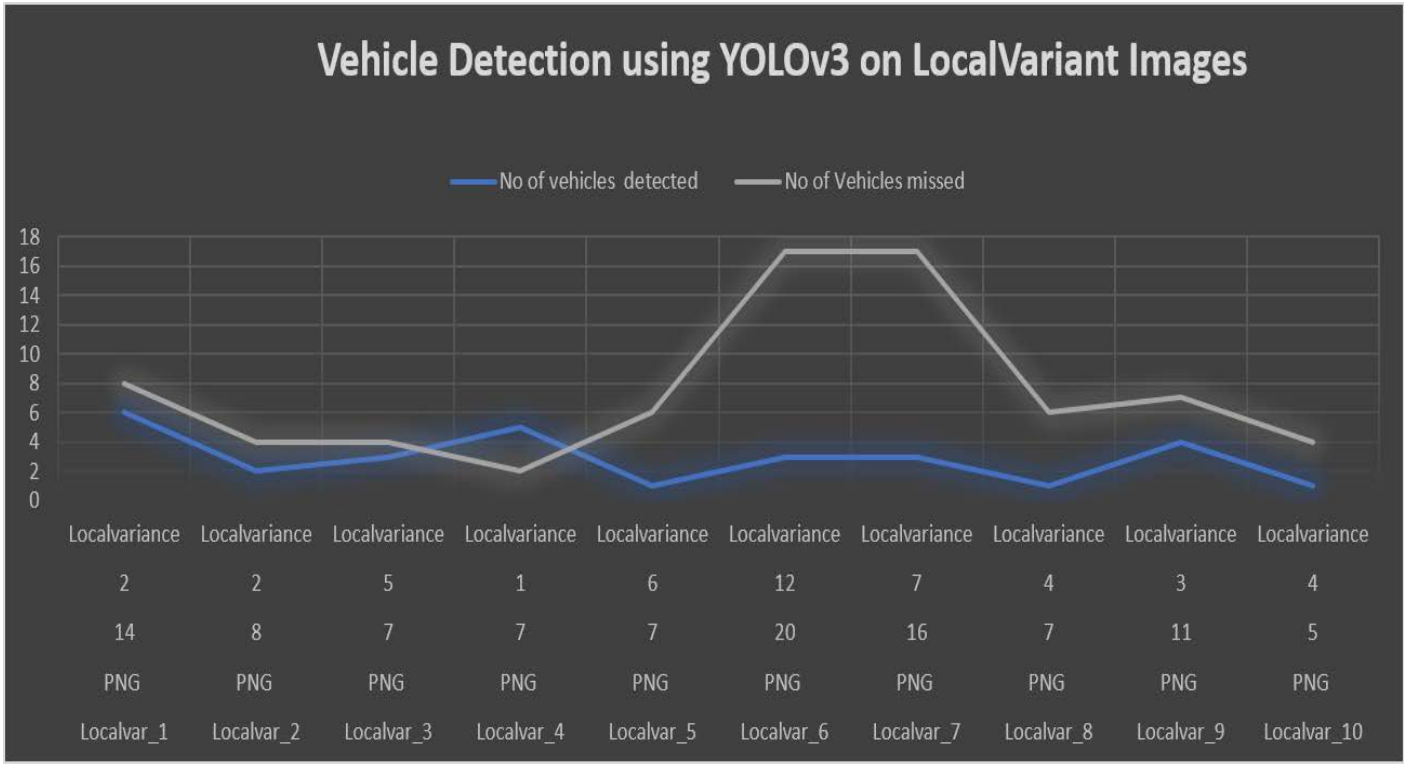


Fig 9.5 Performance of YOLO on our Local Variance induced test set

8+89YOLO ON POISSON NOISE

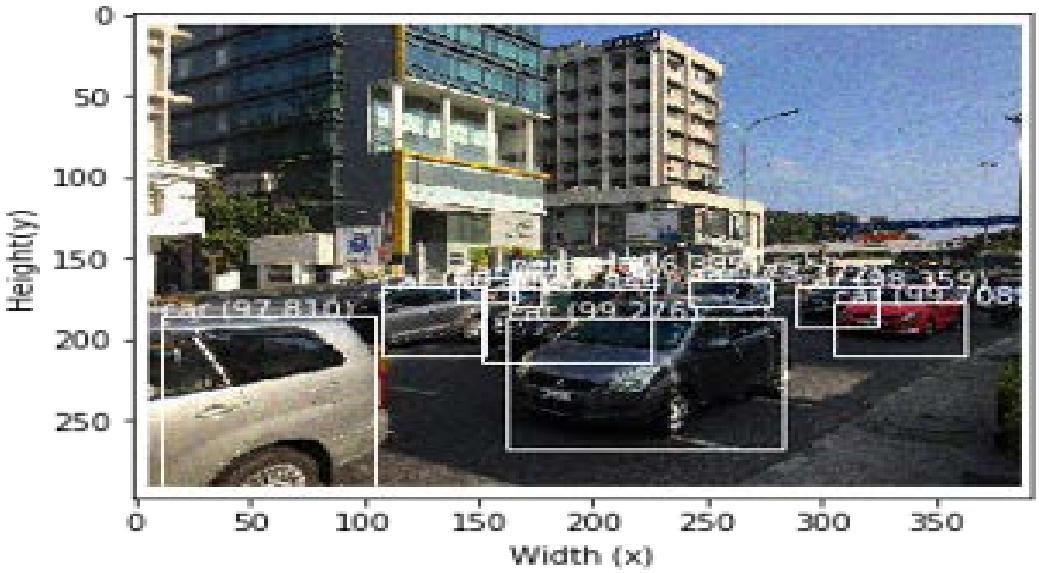


Fig 9.6 YOLO on Poisson induced Image

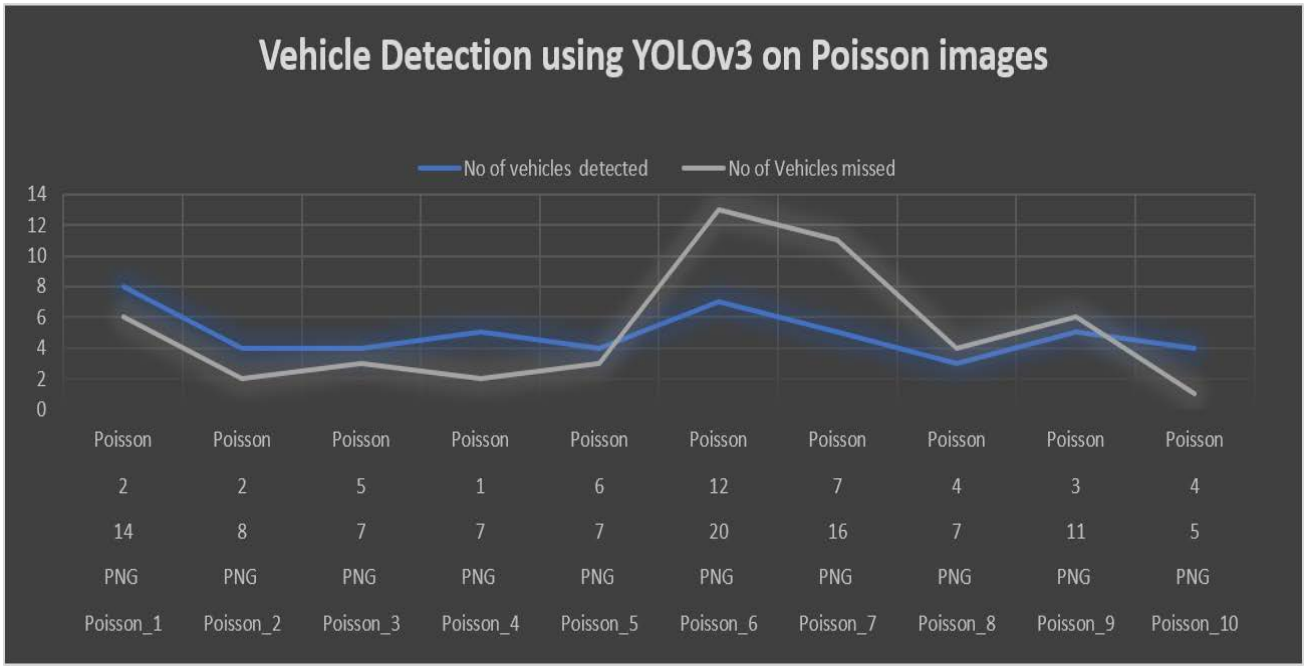


Fig 9.7 Performance of YOLO on our Poisson induced test set.

YOLO ON SALT AND PEPPER

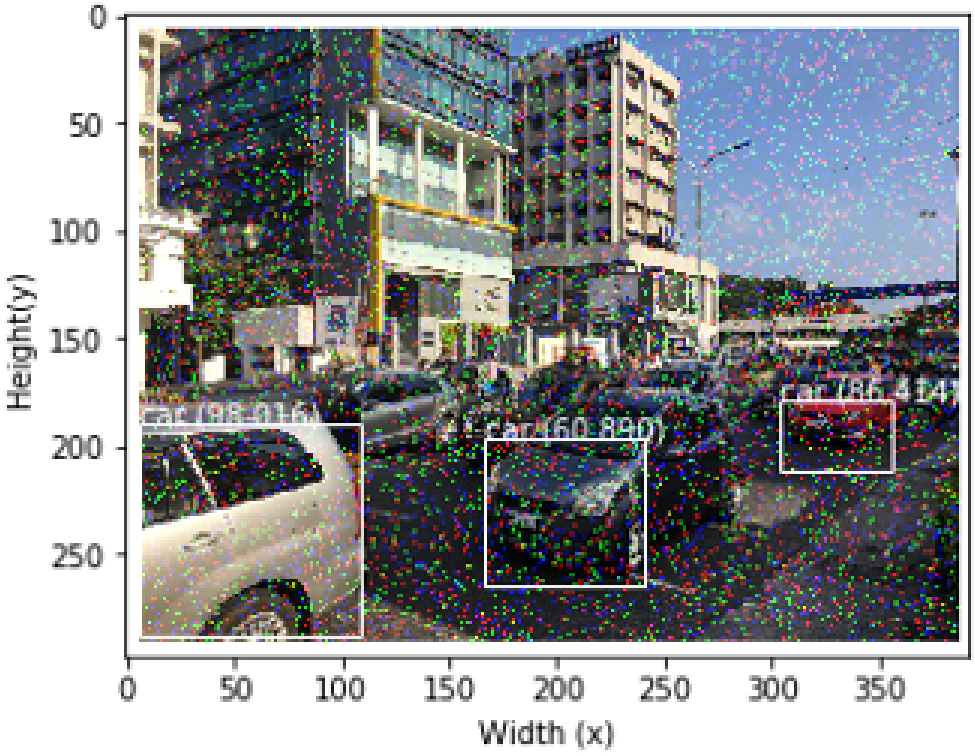


Fig 9.8 YOLO on Salt Noise induced IMG_NO

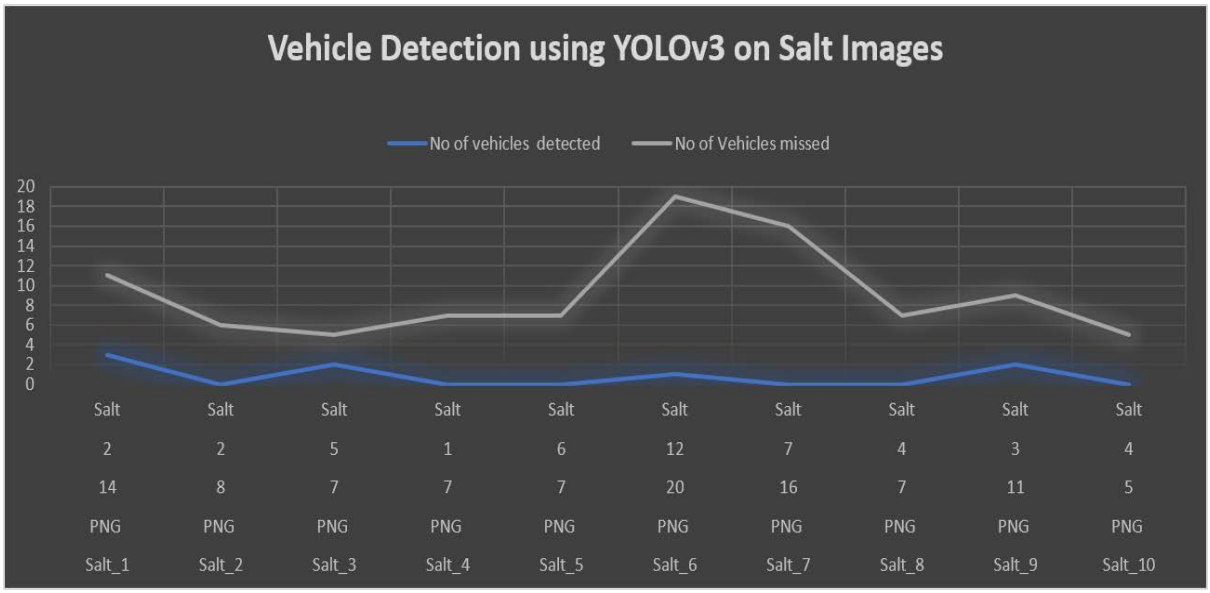


Fig 9.9 Performance of YOLO on Slat Noise induced Images

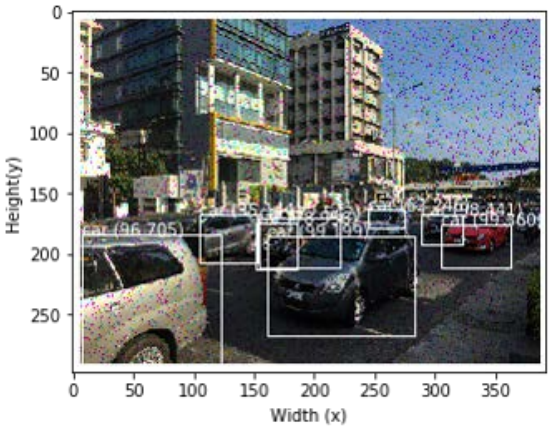


Fig 9.10 YOLO on Pepper Noise Induced Image

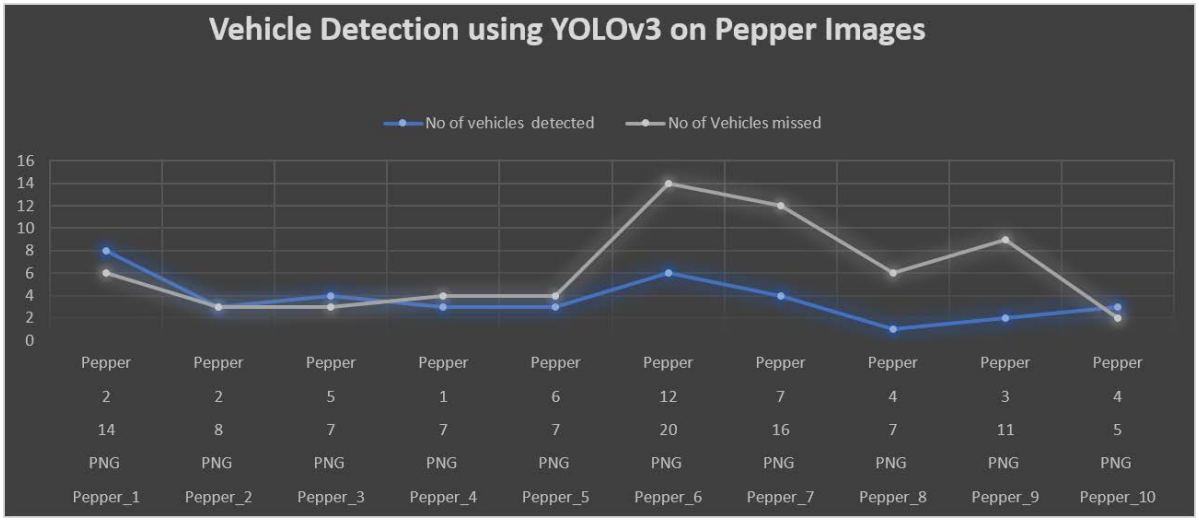


Fig 9.11 Performance of YOLO on Pepper Induced test set images.

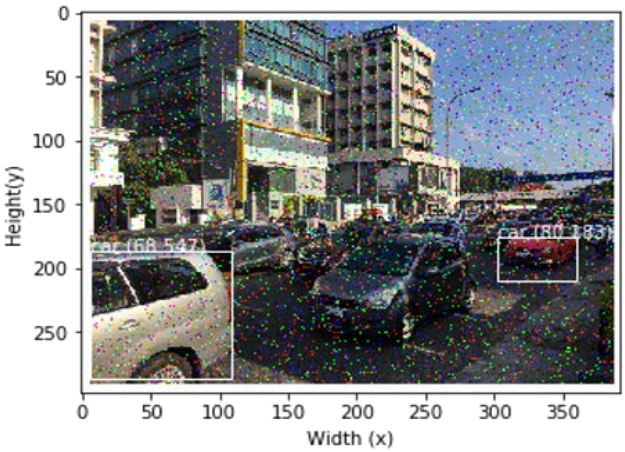


Fig 9.12 YOLO on Salt and Pepper noise induced Image

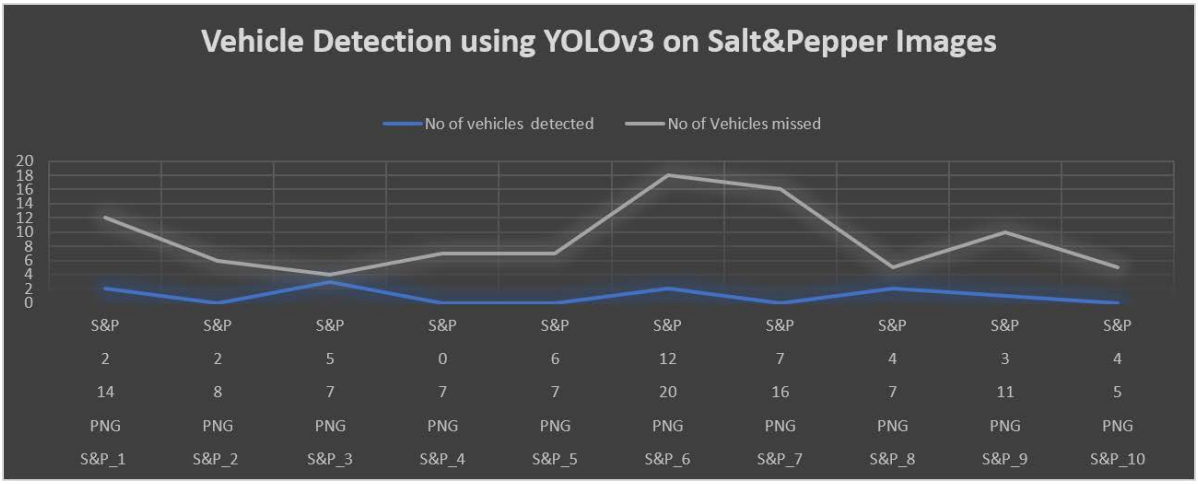


Fig 9.13 Performance of YOLO on Salt and Pepper noise induced test set.

YOLO ON SPECKLE NOISE

Fig 9.14 YOLO on Speckle Noise induced Image

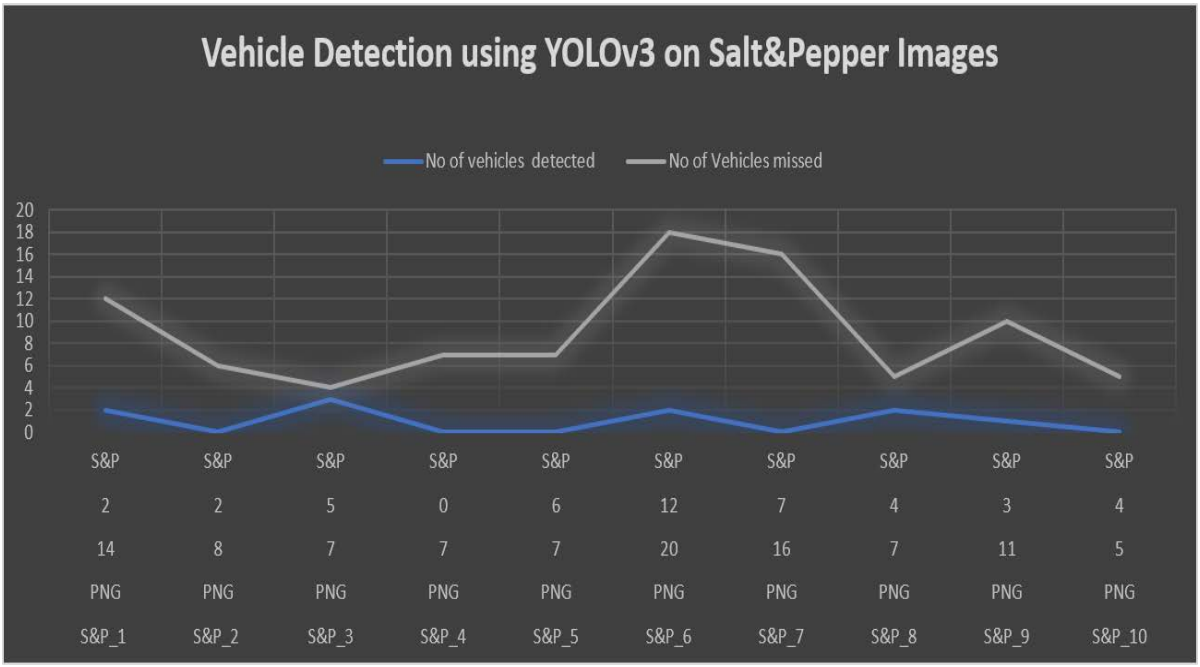
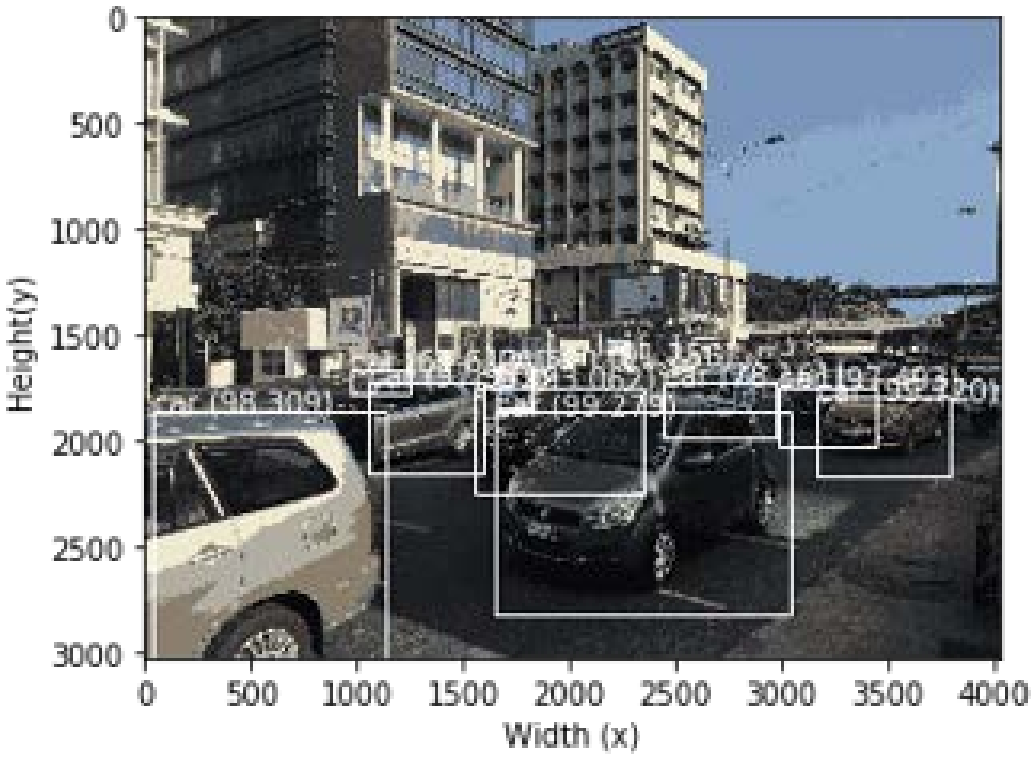


Fig 9.15 Performance of YOLO on Salt and Pepper noise induced test set.

YOLO ON 4BIT QUANTIZED IMAGES



10 Fig 9.16 YOLO on 4BIT Quantised Image

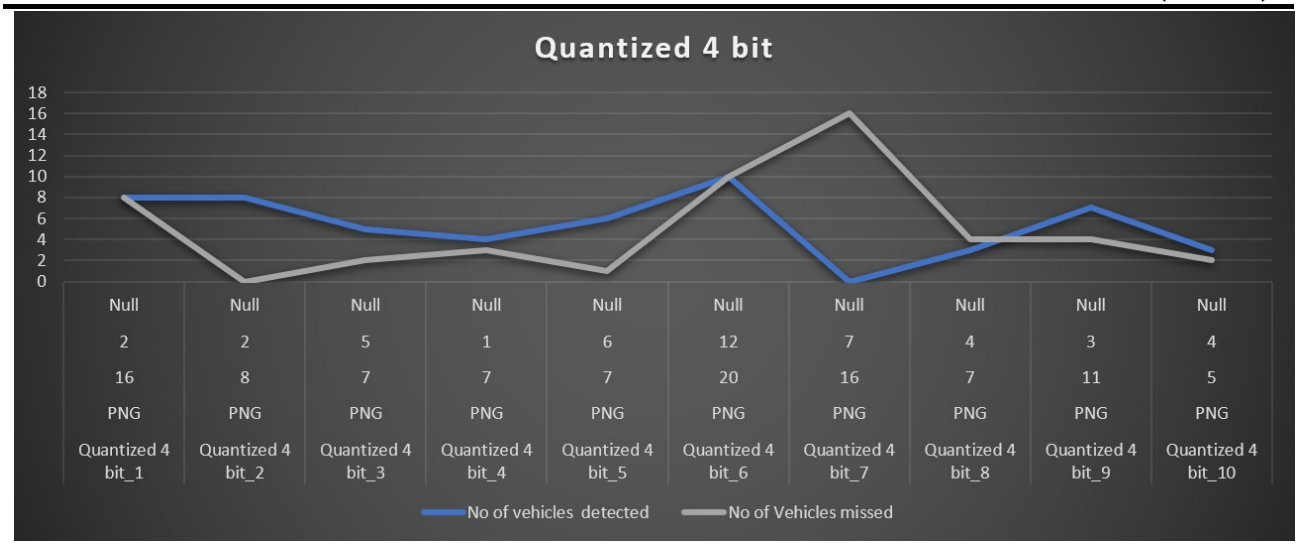


Fig 9.17 Performance of YOLO on 4bit Quantised test set.

YOLO ON 6BIT QUANTISED IMAGES

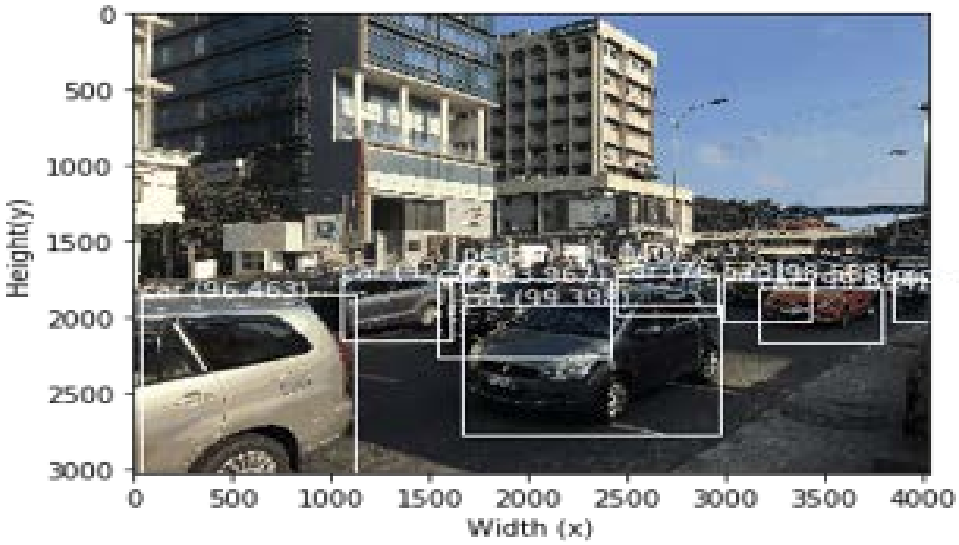


Fig 9.18 YOLO on 6bit quantised Image

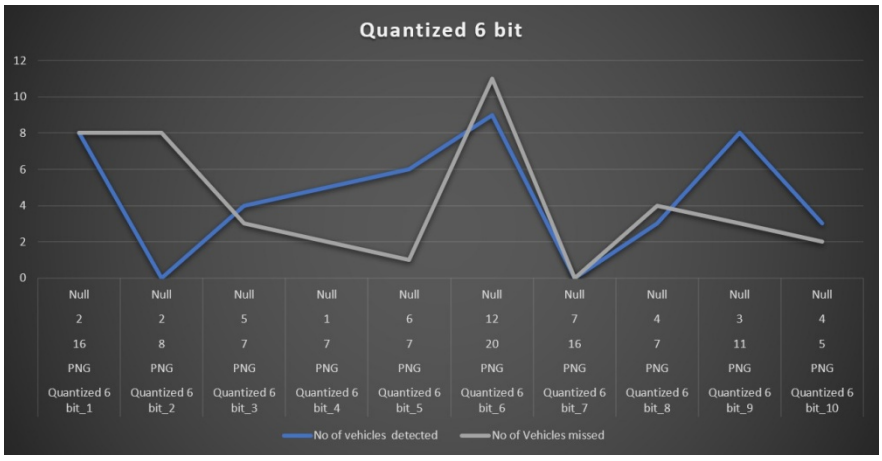


Fig 9.19 Performance of YOLO on 6bit Quantised test set.

IMAGE STATISTICS

Name	Total Images
Training Dataset (Coco Dataset)	123,287 images with 80 different classes
Number of Images used for Testing	307 images

Table 9.1 Image Statistics

ACCURACY OF YOLO ON TESTED IMAGES

Type of Image used for Testing	**ACCURACY USING YOLO-V3**
1. Original Image (Images without noise)	63%
2. Zoomed Images	68%
3. Greyscale Images	59%
4. Distorted Images (Salt, Pepper, Salt and Pepper, Poisson, Gaussian, Speckle, Local-Variance)	41%
5. Quantized Images (4-bit Quantized, 6-bit Quantized)	49% ,53%

Table 9.2 Accuracy on YOLO

PERFORMANCE OF FAST-RCNN ON DISTORTED IMAGES

To test the performance of the Fast R-CNN algorithm in un-natural circumstances we applied the above-mentioned distortions and noises to our ten test images and recorded the observations. First to find the accuracy of vehicle detection we considered two metrics which were number of vehicles that were originally present in the image and the number of vehicles that were detected by Fast- RCNN, similarly we considered the same metrics for human detection. Without applying any sort of distortion, we were able to achieve an accuracy of 75%. We Zoomed into two random sections of the original to image to observe any change in the detection accuracy. In the case of zoomed images, we were able to achieve an accuracy of 77%

FAST-RCNN ON GAUSSIAN NOISE

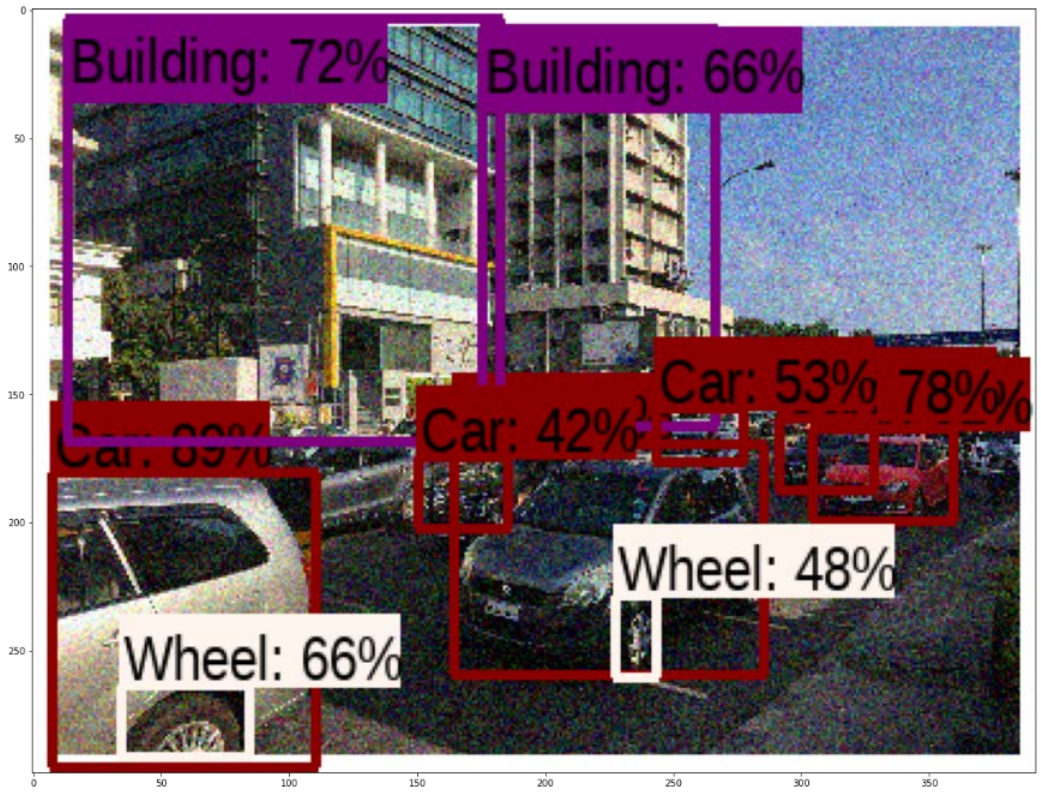


Fig 10.1 Fast RCNN on Gaussian Noise induced IMG_NO

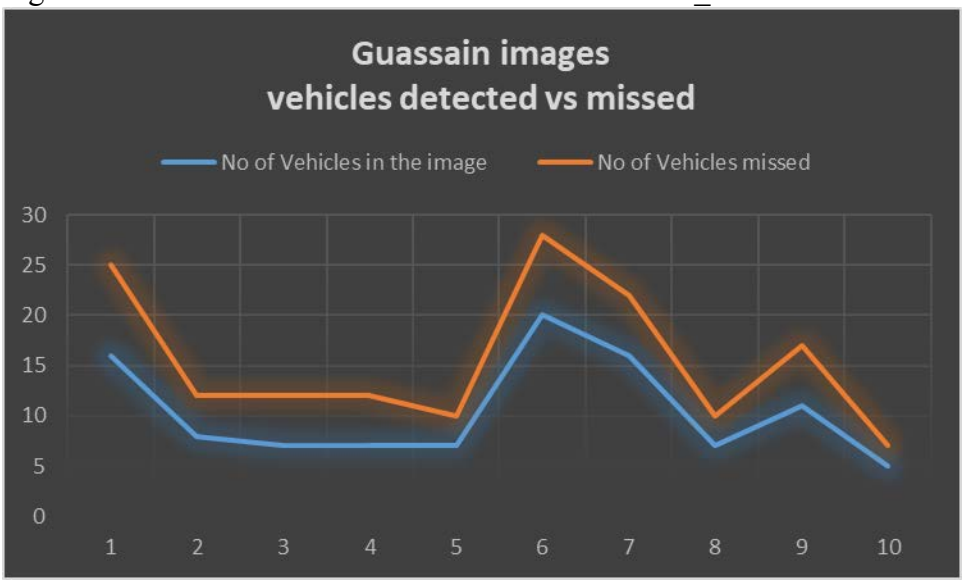


Fig 10.2 Performance of F-RCNN on our Gaussian Noise induced test set.

FAST-RCNN ON LOCAL VARIANCE

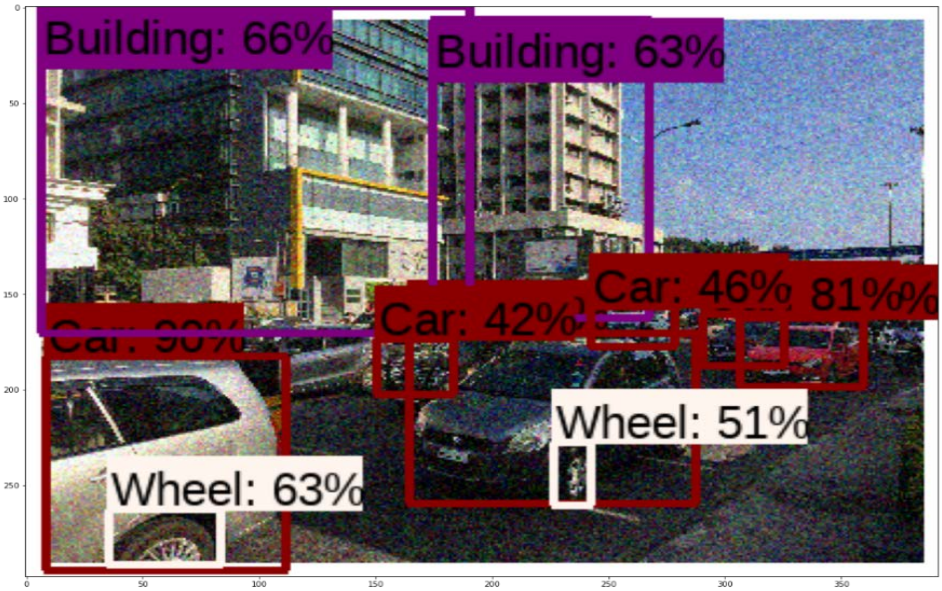


Fig 10.3 F-RCNN on Local Variance induced

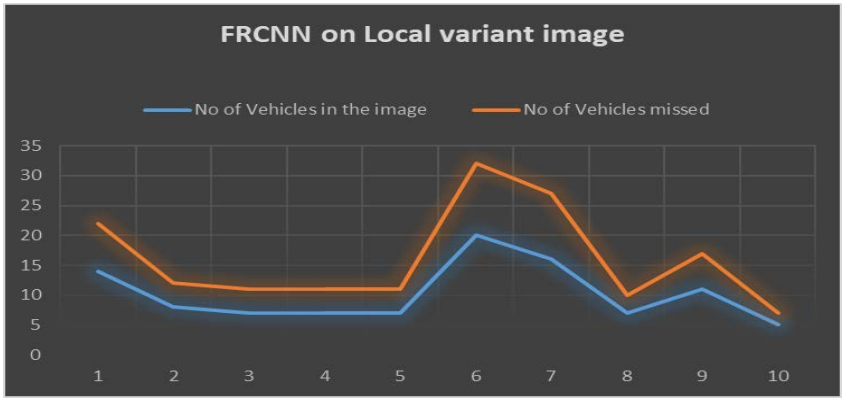


Fig 10.4 Performance of F-RCNN on our Local variance induced test set

FAST-RCNN ON POISSON NOISE

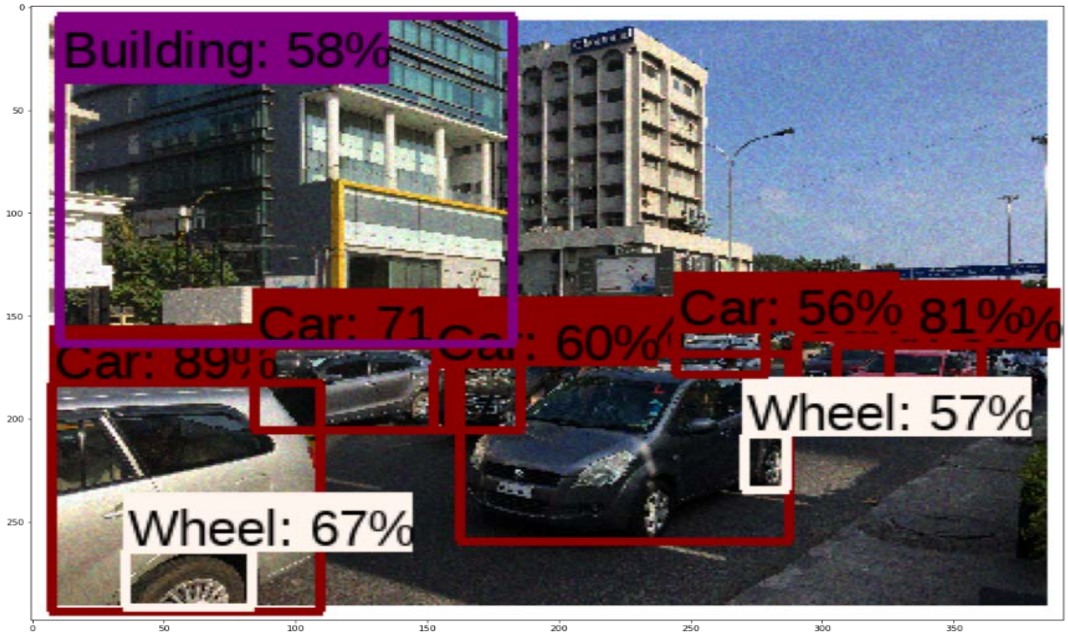


Fig 10.5 F-RCNN on Poisson Noise induced

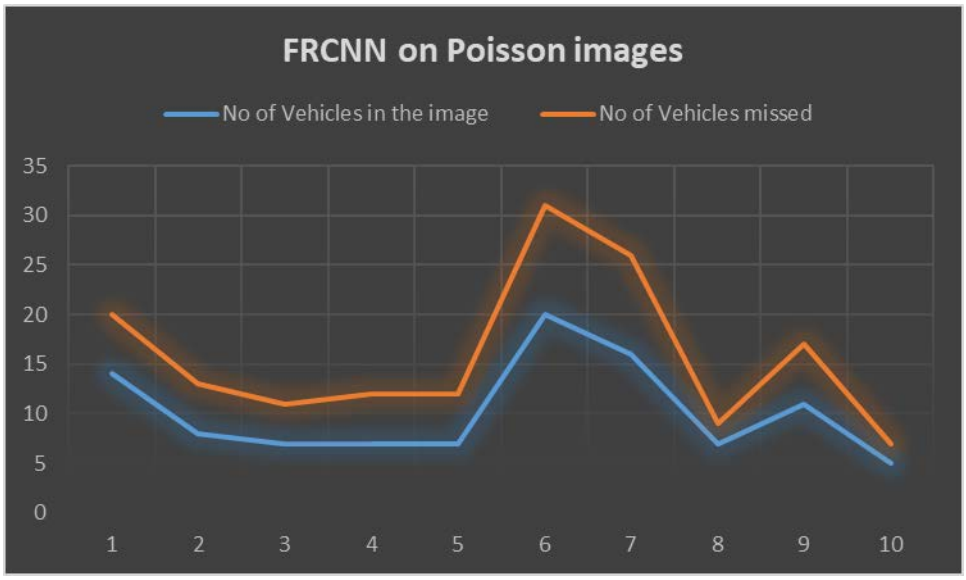


Fig 10.6 Performance of F-RCNN on our Poisson Noise induced test set.

FAST-RCNN ON SALT AND PEPPER

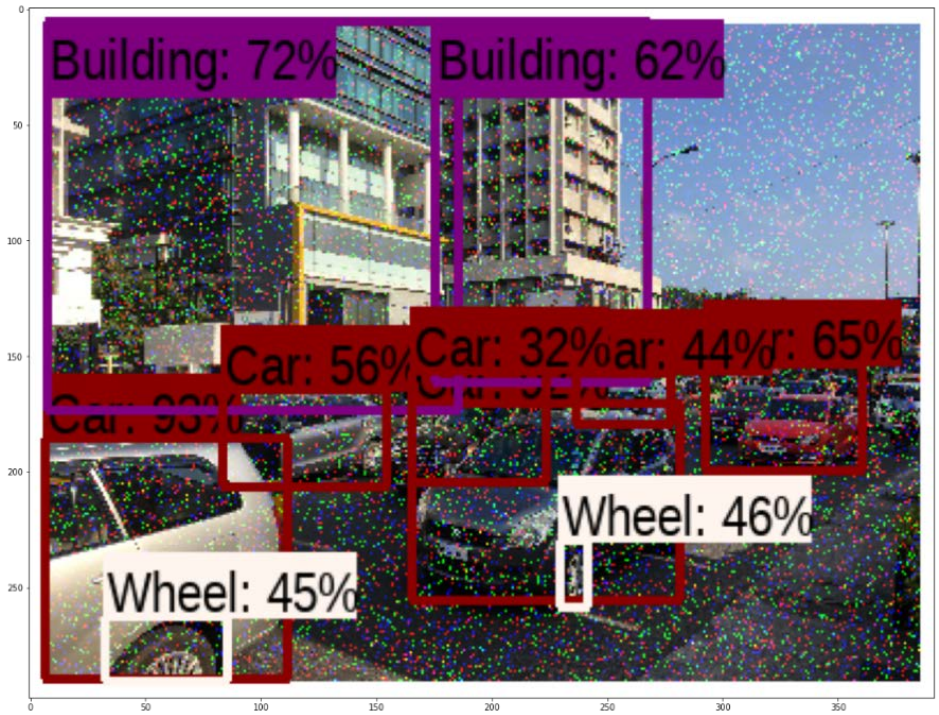


Fig 10.7 F-RCNN on Salt noise induced

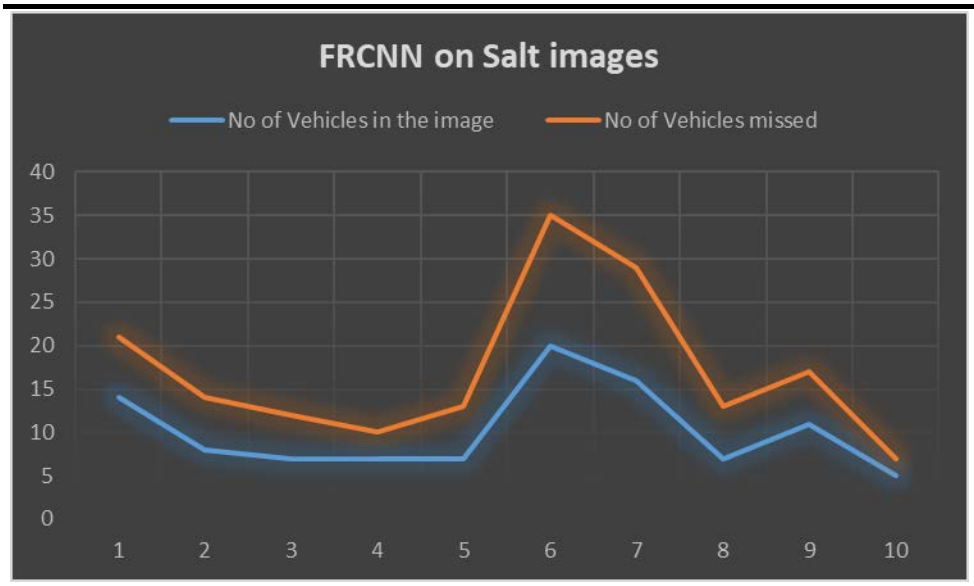


Fig 10.8 Performance of F-RCNN on our Salt noise induced test set.

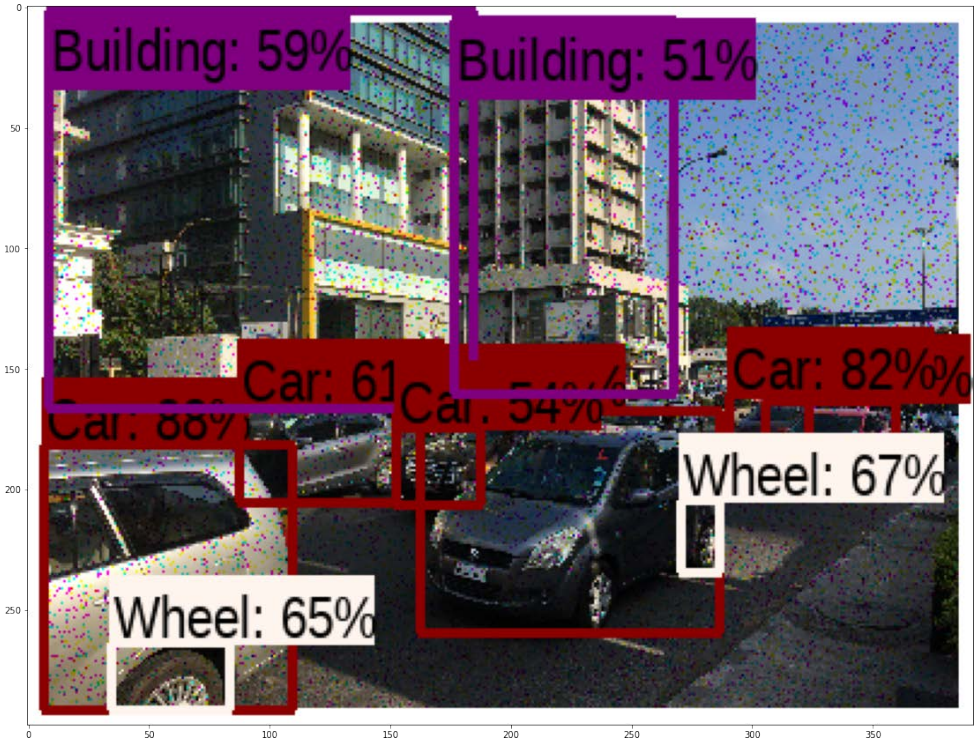


Fig 10.9 F-RCNN on Pepper Noise induced

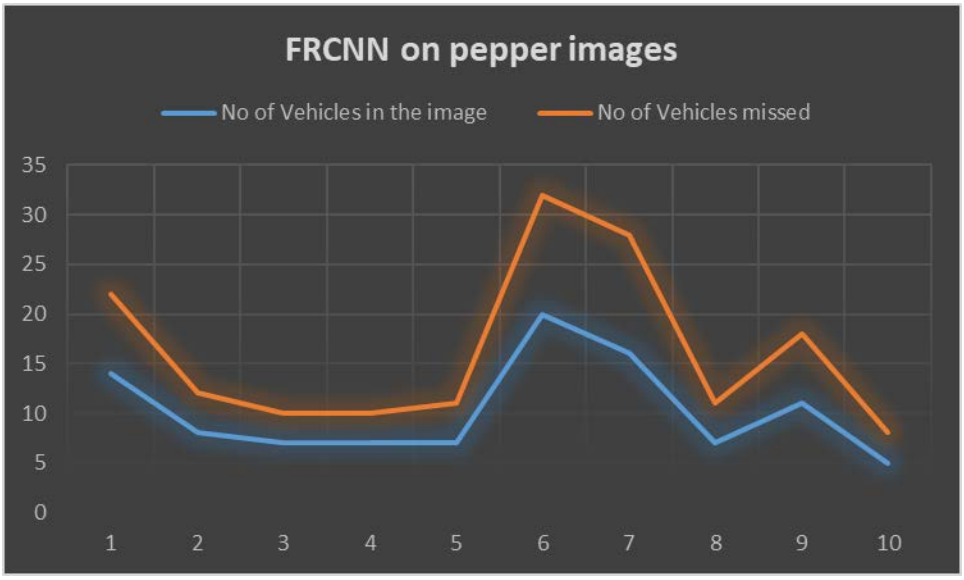


Fig 10.10 Performance of F-RCNN our Pepper Noise induced test set.

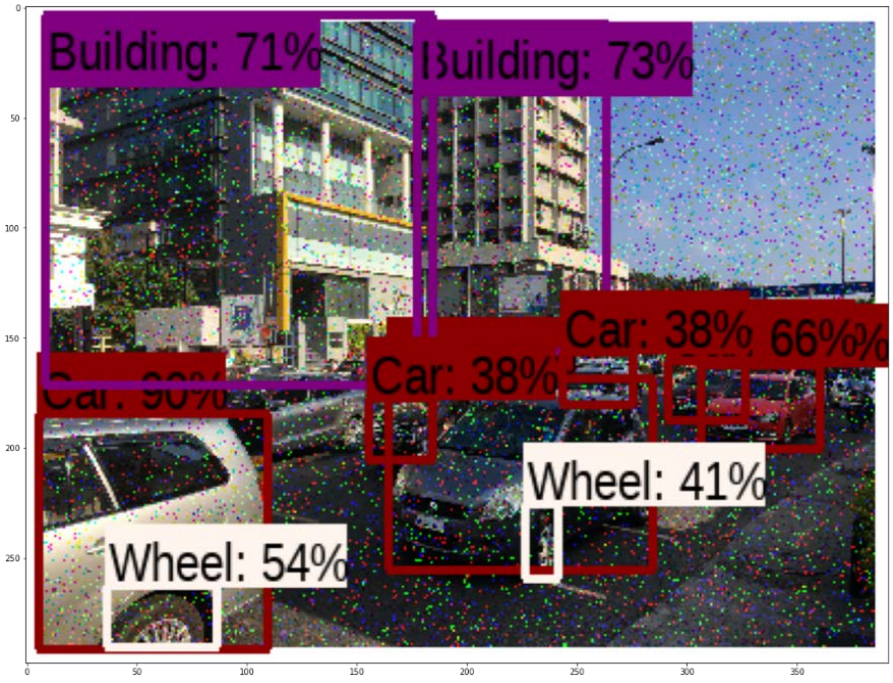


Fig 10.11 Fast-RCNN on Salt and Pepper noise induced

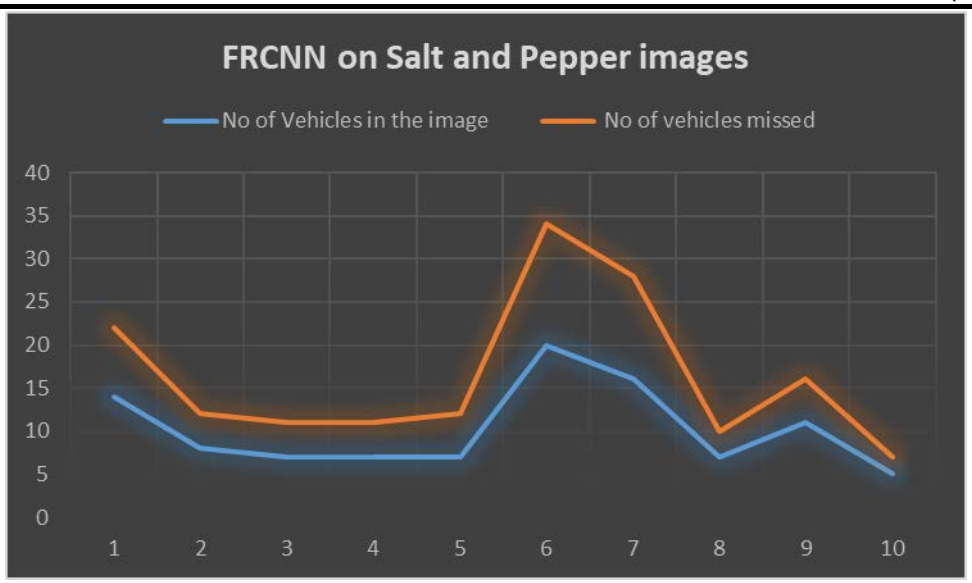


Fig 10.12 Performance of Fast-RCNN on our Salt and Pepper noise induced test set.

FAST-RCNN ON SPECKLE NOISE.

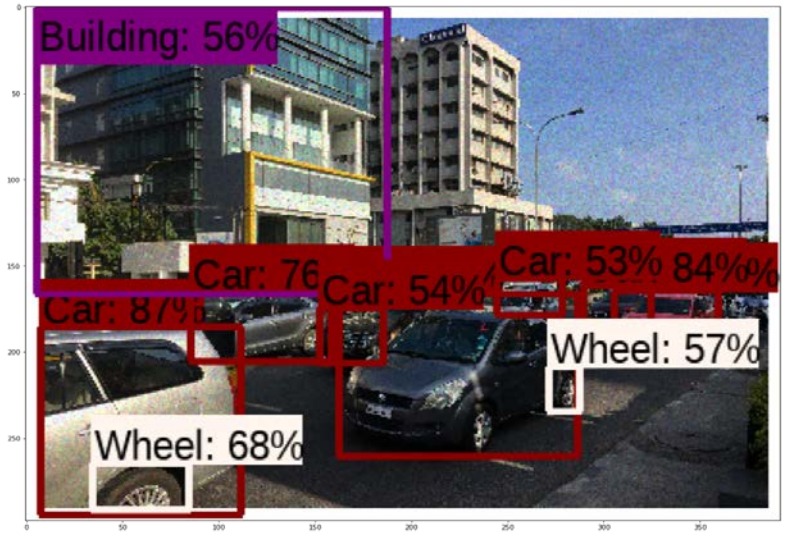


Fig 10.13 Fast-RCNN on Speckle Noise induced

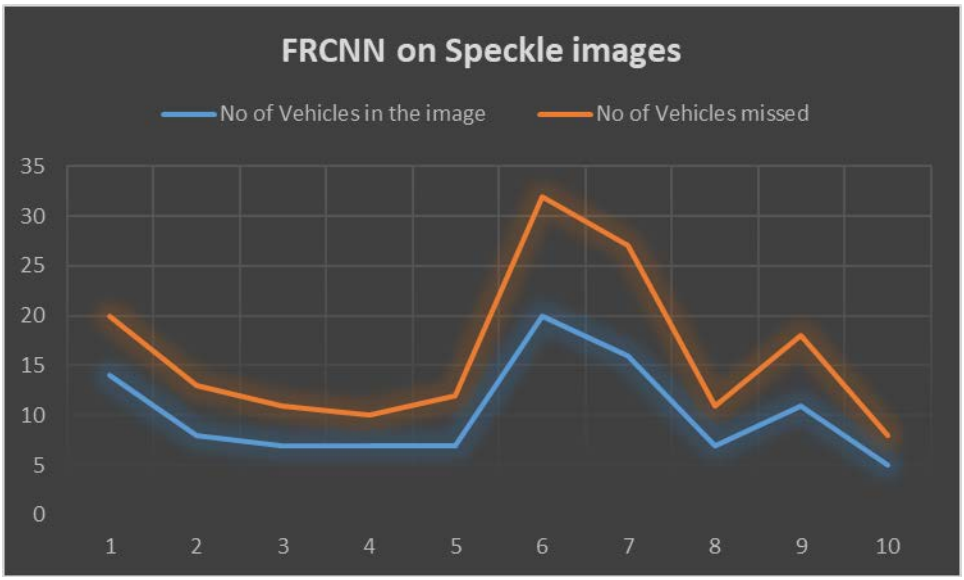


Fig 10.14 Performance of Fast-RCNN on our Speckle Noise induced test set.

F-RCNN ON 4BIT QUANTISED IMAGES

Fig 10.15 Fast-RCNN on 4bit Quantised

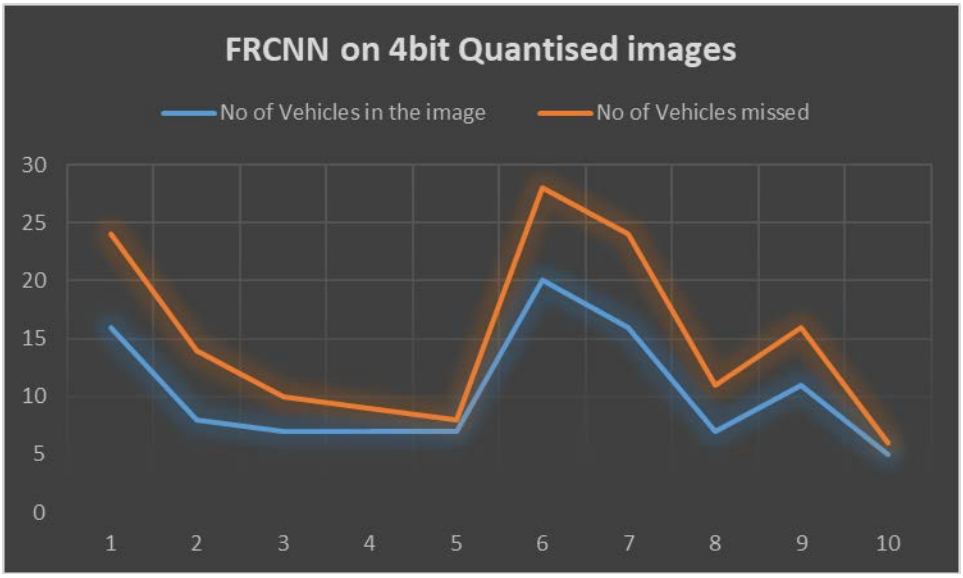


Fig 10.16 Performance of F-RCNN on our 4bit quantised test set.

FASTER-RCNN ON 6BIT QUANTISED IMAGES

Fig 10.17 F-RCNN on 6bit Quantised

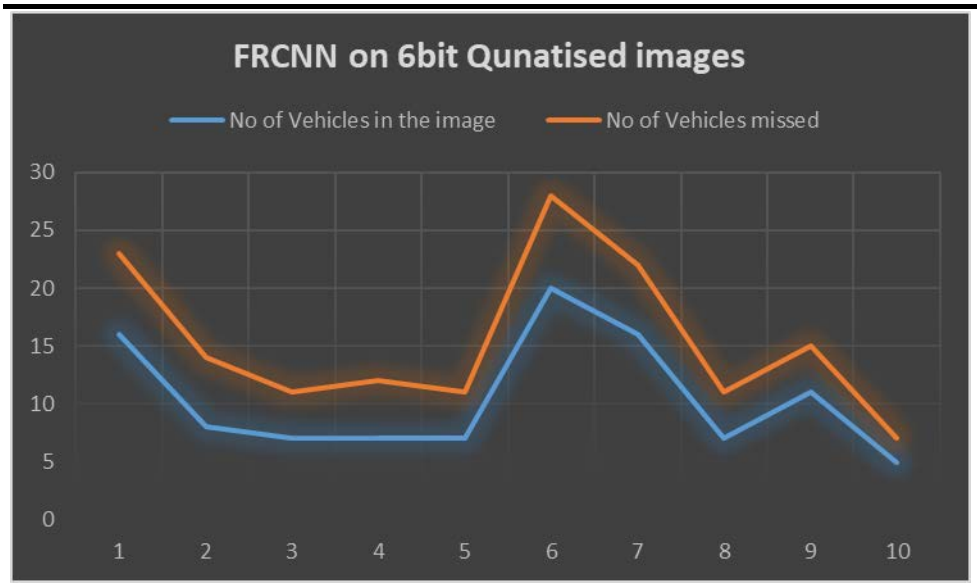
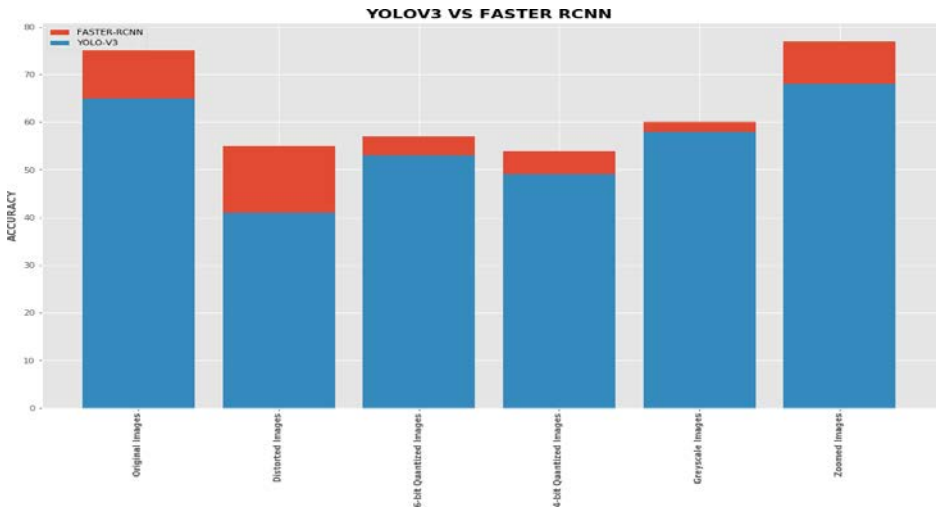


Fig 10.18 Performance of F-RCNN on our 6bit Quantised test set.

PERFORMANCE STUDY

	Fast RCNN	**YOLO v3**
Original Images (*Accuracy*)	75 %	63 %
Distorted Images *(Accuracy*)	55 %	41 %
Quantized 6-bit Images *(Accuracy)*	57 %	53 %
Quantized 4-bit Images *(Accuracy)*	54 %	49 %
Greyscale Images *(Accuracy)*	60 %	59 %
Zoomed Images *(Accuracy)*	77 %	68 %
Inference Time (*seconds to execute*)	3.2 s	0.42 s

Table 11.1 Performance Study



CONCLUSION

A comparison between two state of the art vehicle detection algorithms namely, Yolo-v3 and Fast RCNN with respect to Indian roads was performed. The algorithms were put to a test with the help of Indian road images using Nvidia k80 GPU and taking into account the following metrics - accuracy, inference time and performance on inducing distortion. On observing the following metrics for each of the algorithms, we observe that Fast RCNN performs slightly better than Yolo v3.

Although, for Realtime vehicle detection applications we would recommend using Yolo-v3. The algorithm takes far lesser amount of time to detect vehicles when compared to Fast RCNN.

REFERENCES

1. J Janai, F Güney, A Behl, A Geiger .“Computer Vision for Autonomous Vehicles: Problems,Datasets and State-of-the-Art”, 18 Apr 2017, arXiv:1704.05519
2. R Girshick, J Donahue, T Darrell, J Malik .“Rich feature hierarchies for accurate object detection and semantic segmentation”. 22 Oct 2018, arXiv:1811.2524
3. Joseph Redmon, Ali Farhadi.”YOLOv3: An Incremental Improvement”. 8 Apr 2018, arXiv:1808:028672.
4. Tsung-Yi,Lin Michael,Maire Serge,Belongie Lubomir,Bourdev Ross,Girshick James Hays,Pietro Perona ,Deva Ramanan ,C. Lawrence Zitnick ,Piotr Dollar.“Microsoft COCO: Common Objects in Context”. 21 Feb 2018, arXiv:1804.05519.
5. Ajay Kumar, Boyat, Brijendra Kumar Joshi (2015). ” A REVIEW PAPER: NOISE MODELS IN DIGITAL IMAGE PROCESSING”. Signal & Image Processing: An International Journal (SIPIJ) Vol.6, No.2, April 2015. arXiv:1505.03489v1.
6. Geethapriya. S, N. Duraimurugan, S.P. Chokkalingam (2019). “Real- Time Object Detection with Yolo”. International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-8, Issue-3S, February 2019.
7. Prabhishek Singh, Raj Shree (2016). “A Comparative Study to Noise Models and Image Restoration Techniques”. International Journal of Computer Applications (0975 – 8887) Volume 149 – No.1, September 2016.
8. Kohli, P., & Chadha, A. (2019). Enabling Pedestrian Safety Using Computer Vision Techniques: A Case Study of the 2018 Uber Inc. Self-driving Car Crash. Perspectives on Asian Tourism, 261– 279. doi:10.1007/978-3-030-12388-8_19.
9. Pandian, A. P., Ntalianis, K., & Palanisamy, R. (Eds.). (2020). Intelligent Computing, Information and Control Systems. Advances in Intelligent Systems and Computing. doi:10.1007/978-3-030-30465-2.
10. P. V. Babayan, M. D. Ershov and D. Y. Erokhin, "Neural Network- Based Vehicle and Pedestrian Detection for Video Analysis System," 2019 8th Mediterranean Conference on Embedded Computing (MECO), Budva, Montenegro, 2019, pp. 1-5.
11. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks
12. PYTHON v3.6- https://www.python.org/downloads/release/python-360/
13. Tensorflow 2.0- https://www.tensorflow.org/
14. Keras- https://keras.io