****

**Master Thesis**

**Master of Science (M.sc)**

**Department of Tech and Software**

**Major: Data Science**

**Topic: A Safety Vehicle Detection System using YOLOv5**

Author: Girish Venugopalan Nair

Matrikel-Number: 10130716

First supervisor: Prof Dr. Raja Hashim Ali

Second supervisor: Prof Dr. Talha Ali Khan

Submitted on:

**Statutory Declaration:**

I hereby declare that I have developed and written the enclosed Masters Thesis completely by myself and have not used sources or means without declaration in the text. I clearly marked and separately listed all the literature and all the other sources which I employed when producing this academic work, either literally or in content. I am aware that the violation of this regulation will lead to failure of the thesis.

Berlin, 28-08-2022 ..............................................

Place, Date Signature

**ABSTRACT**

Autonomous vehicles are the trend nowadays and many of manufacturers introduce innovations to boast their intelligent vehicles. With the advent of science, many assistive technologies emerged to help the driver during long drives like hands-free driving, cruise control, auto adaptive headlights, etc. Artificial intelligence also takes its position in assisting the commutation to be safer and more effortless with technologies like self-driving, lane detection, collision detection, etc. Although collision detection has been there for a while in the automotive industry for many famous manufacturers like Mercedes, BMW, Audi, and Tesla generic danger prediction and warning have not been explored much. This research is proposing to predict threats in advance so that the driver is aware before it is too late to make a decision. The idea here is to use Ultralytics Yolov5 for quick object detection which is a Convolutional Neural Network designed by Ultralytics. Annotated road surface data RDD2020 was used for performing the tests. The data contains classifications like potholes, debris, and cracks which are used to train the model and then later used to warn the driver. Yolov5 has different models, and a comparison of the models was performed. An analysis of the hyperparameters was done to fine-tune the detection results.

Keywords: road damage; convolutional neural networks; semantic segmentation; Yolov5

Contents

[GLOSSARY 7](#_Toc112619127)

[LIST OF TABLES AND FIGURES 9](#_Toc112619128)

[CHAPTER 1: INTRODUCTION 10](#_Toc112619129)

[1.1 MOTIVATION 10](#_Toc112619130)

[1.2 OBJECTIVE 12](#_Toc112619131)

[1.3 SCOPE 13](#_Toc112619132)

[1.4 RESEARCH QUESTION 13](#_Toc112619133)

[CHAPTER 2: RELATED WORK 14](#_Toc112619134)

[2.1 AVAILABLE NEURAL NETWORK MODELS 15](#_Toc112619135)

[2.1.1 VISION TRANSFORMER VS RESNET 16](#_Toc112619136)

[2.1.2 EFFICIENTDET, YOLOV5 AND EFFICIENTNET 16](#_Toc112619137)

[2.1.3 RESMLP 18](#_Toc112619138)

[2.1.4 ResMLP WORKING: 18](#_Toc112619139)

[2.1.5 RESNET 19](#_Toc112619140)

[2.1.6 R-FCN 22](#_Toc112619141)

[2.1.7 SSD 22](#_Toc112619142)

[2.1.8 CRACKNET/CRACKNET-V 22](#_Toc112619143)

[2.1.9 YOLOV5 vs RCNN 24](#_Toc112619144)

[2.2 HAND-CRAFTED FEATURE ENGINEERING 27](#_Toc112619145)

[2.3 CHALLENGES IN COMPUTER VISION 27](#_Toc112619146)

[2.3.1 IMAGE CLASSIFICATION 27](#_Toc112619147)

[2.3.2 IMAGE LOCALIZATION 28](#_Toc112619148)

[2.3.3 OBJECT DETECTION 28](#_Toc112619149)

[2.3.4 IMAGE SEGMENTATION 29](#_Toc112619150)

[2.4 EDGE DETECTION 31](#_Toc112619151)

[2.4.1 HOLISTICALLY NESTED EDGE DETECTION 32](#_Toc112619152)

[2.5 TOP-DOWN FULLY CONVOLUTIONAL ENCODER-DECODER NETWORK 35](#_Toc112619153)

[2.6 SPATIALLY COHERENT RANDOM FORESTS 35](#_Toc112619154)

[2.7 DEEPEDGE: A MULTI-SCALE BIFURCATED DEEP NETWORK 37](#_Toc112619155)

[2.8 ACTIVE CONTOUR 38](#_Toc112619156)

[2.9 CONVOLUTIONAL NEURAL NETWORKS 38](#_Toc112619157)

[2.10 ROAD SIGN DETECTION 40](#_Toc112619158)

[2.11 ROAD DAMAGE DETECTION USING UNSUPERVISED DISPARITY MAP SEGMENTATION 41](#_Toc112619159)

[2.12 CONVENTIONAL TECHNIQUES FOR IMAGE PROCESSING 43](#_Toc112619160)

[2.12.1 YOLOv5 44](#_Toc112619161)

[2.12.2 ARCHITECTURAL DETAILS 44](#_Toc112619162)

[2.13 UNDERSTANDING FEATURE PYRAMID NETWORKS FOR OBJECT DETECTION (FPN) 50](#_Toc112619163)

[2.13.1 ACTIVATION FUNCTION 51](#_Toc112619164)

[2.13.2 OPTIMIZATION FUNCTION 51](#_Toc112619165)

[2.13.3 COST FUNCTION OR LOSS FUNCTION 51](#_Toc112619166)

[2.13.4 HYPERPARAMETERS IN YOLOV5 51](#_Toc112619167)

[2.13.6 TYPES OF YOLOv5 52](#_Toc112619168)

[CHAPTER3: METHOD 54](#_Toc112619169)

[3.1 CONVOLUTION NEURAL NETWORK USED 54](#_Toc112619170)

[3.1.1 YOLOv5 54](#_Toc112619171)

[3.1.2 RELEASES 55](#_Toc112619172)

[3.2 CUSTOM OBJECT DETECTION WITH YOLOV5 57](#_Toc112619173)

[3.2.1 PREPARE THE DATASET 57](#_Toc112619174)

[3.2.2 SETUP THE ENVIRONMENT 59](#_Toc112619175)

[3.2.3 YOLO ARCHITECTURE AND WORKING 60](#_Toc112619176)

[3.2.4 HYPER PARAMETER EVOLUTION OF YOLOV5 60](#_Toc112619177)

[3.3 PRUNE AND SPARSE YOLOv5 63](#_Toc112619178)

[CHAPTER 4: RESULTS AND DISCUSSIONS 66](#_Toc112619179)

[4.1 TRAINING ON MS-COCO TO GENERATE BASELINE METRICS 66](#_Toc112619180)

[4.2 TRAINING ON ROAD SIGNS 68](#_Toc112619181)

[4.3 TRAINING ON ROAD DEFECTS 71](#_Toc112619182)

[CHAPTER 5: CONCLUSION AND FUTURESCOPE 75](#_Toc112619183)

[ACKNOWLEDGMENT 81](#_Toc112619184)

[REFERENCES 82](#_Toc112619185)

# GLOSSARY

1. AIP Analog Image Processing
2. API Application Programming Interface
3. ANN Artificial Neural Network
4. BSDS Berkeley Segmentation Dataset
5. CNN Convolution Neural Network
6. CSP Cross Stage Partial
7. COCO Common Objects in Context
8. CPU Central Processing Unit
9. CGAN Conditional Generative Adversarial Network
10. CV Computer Vision
11. DIP Digital Image Processing
12. FC Fully Connected
13. FCN Fully Convolutional Network
14. FLOP Floating Point Operation
15. FPN Feature Pyramid Network
16. GA Genetic Algorithm
17. GAN Generative Adversarial Network
18. GPU Graphical Processing Unit
19. HED Holistically-nested Edge Detector
20. HOG Histogram of Orientated Gradients
21. IOU Intersection over Union
22. LBP Local Binary Pattern
23. LIDAR
24. mAP Mean Average Precision
25. MS- Microsoft
26. NLP Natural Language Processing
27. PANet
28. RCF Richer Convolutional Features
29. RDD Road Defect Detection
30. RF- Random Forest
31. ReLU – Rectified Linear Unit
32. SCRF - Spatially Coherent Random Forest
33. SGD stochastic gradient descent
34. SSD - Single Shot Multibox Detector
35. SVM Support Vector machines
36. TD-CEDN Top-down fully Convolutional Encoder-Decoder Network
37. VC-VC Video-based Vehicles Classification
38. ViT – Vision Transformer
39. YOLO - You Only Look Once

# LIST OF TABLES AND FIGURES

[Figure 0.1 RESNET 20](#_Toc112621740)

[Figure 0.2 Plain Network vs ResNet 21](#_Toc112621741)

[Figure 0.3 Parameters of Neural Networks 45](#_Toc112621742)

[Figure 2.0.4 DenseNet 46](#_Toc112621743)

[Figure 0.5 DenseNet 46](#_Toc112621744)

[Figure 0.6 DenseNet 47](#_Toc112621745)

[Figure 0.7 EfficientNet 48](#_Toc112621746)

[Figure 0.8 YOLOv5 Models 53](#_Toc112621747)

[Figure 0.1 YOLO Performance Comparison 56](#_Toc112621748)

[Figure 0.2 YOLO Models Performance 56](#_Toc112621749)

[Figure 0.3 Performance Comparison Metrics 57](#_Toc112621750)

[Figure 0.4 RDD Dataset Folder Structure 58](#_Toc112621751)

[Figure 0.5 YOLO Architecture 60](#_Toc112621752)

[Figure 0.6 Yolo Evolution 63](#_Toc112621753)

[Figure 0.1 Performance Metrics for MS-COCO 66](#_Toc112621754)

[Figure 0.2 Performance Metric for Road Signs 68](#_Toc112621755)

[Figure 0.3 Confusion matrix for LISA Road Signs 69](#_Toc112621756)

[Figure 0.4 F1 Curve 69](#_Toc112621757)

[Figure 0.5 Precision-Recall Curve 70](#_Toc112621758)

[Figure 0.6 Mosaic for Road Defect Detection 71](#_Toc112621759)

[Figure 0.7 mAP 0.5:0.95 for RDD 72](#_Toc112621760)

[Figure 0.8 mAP 0.5 72](#_Toc112621761)

[Figure 0.9 Recall 73](#_Toc112621762)

[Table 1 Yolov5 Layers 51](#_Toc112621763)

[Table 2 val.py 65](#_Toc112621764)

[Table 3 Hyperparameters 67](#_Toc112621765)

[Table 4 Confusion Matric for MS-COCO 67](#_Toc112621766)

# CHAPTER 1: INTRODUCTION

## 1.1 MOTIVATION

Commutation has always been a need of man to achieve his goals, to be productive, to socialize, to engage in business and many more. Various modes of travel also have been invented, yet the road remains the most widely used as per ‘Determinants of Mode Choice between Road and Shipping for Freight Transport: Evidence for Four Spanish Exporting Sectors’[1]. During a span of 3 years in US around 200,000 accidents occurred due to road debris which could have been avoided if a proper warning system was available in the vehicle[2]. The safety of vehicles is also regulated by the governments across the world and manufacturers comply with guidelines and standards Innovations like seat belts and driver assistance technologies have greatly improved road safety. However, human errors are still prone, and the time is ripe to discuss how we can reduce manual errors by timely warning using intelligent driver-assistive technologies.

There are two categories of problems addressed in this thesis. The first one is the defects on road surfaces which can be detected, and the driver be warned from a distance so that the driver can avoid the danger that may otherwise arise. The second category is to inform the driver about the road signboards so that it doesn’t go unnoticed even if the driver fails to notice it himself.

Inadequately maintained roads are a problem that can cause severe harm to moving vehicles without immediately apparent warning indications. On roadways, potholes are a potential hazard. Potholes frequently arise out of nowhere and get larger over time. Strong tired vehicles, such as cars and trucks, are particularly affected by potholes. When there is a sudden change in the road's state, such as when a rock falls into an old pothole or when construction workers remove pavement materials close to an already compromised portion of roadway, the problem is referred to as a pothole. Most motorists perceive a "pothole" as a rapid change in state since it wasn't visible until after a vehicle struck it, which is something nobody would do if they were aware of the flaw beforehand. Road cracks are frequently reported by motorists to the appropriate authorities so that these flaws can be fixed.

Over time, a number of techniques have been created for identifying these flaws and fixing any issues with roads before more significant harm occurs. Physical road flaws can be found by observing the road from several angles, including above, below, and on the sides. For instance, a road surface fissure can resemble a lattice or a spider web. Smaller cracks may be present all around a huge pothole. Before the flaw becomes a severe risk to the safety of the vehicles below it, magnetic micro-sensing technologies can find fractures on bridges. Car dashboard cameras have the ability to detect fissures in the road's surface before vehicles do. The location of the cracks and whether they endanger drivers are then determined by examining the camera footage.

Drivers who go for long drives often fall prey to highway hypnosis otherwise known as white line fever. It is a mental state in which drivers conscious mind is elsewhere and he keeps on mechanically responding to external events in the safe and ideal way. Although it is claimed to be safe, since it is an unpredictable state of mind, it is better to have a warning system which can alert the driver of road signs and other critical situations like a crossing of an animal or person.

Drivers can stay safe while driving by paying great attention to the condition of their roads whenever they drive, despite the fact that they have no way of predicting which parts of their trip will develop into an unanticipated "pothole." By observing the state of the road and avoiding any potholes they notice, drivers can avoid hitting potholes. Drivers can utilize the proper instruments to remedy problems when they are discovered early enough, either through visual inspection or system analysis, to prevent serious damage or dangerous accidents brought on by cracked wheel bearings from uneven terrain.

## 1.2 OBJECTIVE

There have been various attempts of solving this problem of vehicle safety using different sensors, radars, etc. The challenges are the detections should be real-time and should be efficient enough to detect irrespective of the size of the threat being detected. A camera system can identify speed, lane position, and signs from a traffic accident. Depending on how fast the cars were moving at the time of accident or which lane they were in, different drivers will react differently. By instantly alerting drivers to their present speed, position, and road sign condition, camera systems prevent traffic accidents.

When using such technology, drivers quickly learn how to adjust their driving style without putting themselves or others in risk. A camera system can also identify a collision based on the state of the road, such as whether it is wet, dry, or icy. Drivers frequently exceed the posted speed limit on wet roads because it seems more comfortable than when they encounter a rough dry section. There is no way for drivers to know if their present driving scenario matches what they observed on the road previously, therefore such factors influence whether an accident occurs or not. He might have been partially responsible for any accidents that occurred while he was driving in these dangerous conditions because of his limited sight on the wet roadways. In order to prevent endangering themselves and others even more, drivers who dislike these unsafe driving conditions—such as those who are fatigued—should refrain from operating a motor vehicle during these times. A method well discussed is semantic segmentation which classifies objects at the pixel level.

## 1.3 SCOPE

In this research we are exploring YOLOv5 object detection-based vehicle safety system which will be trained using IEEE RDD dataset and LISA Road Signs dataset. The performance of YOLOv5s on the above said datasets will be evaluated based on the benchmark set by YOLOv5s on MS-COCO128 dataset. Since the research will be for shorter period, other YOLOv5 models are skipped and any neural network modifications other than the ones achievable by hyperparameter tuning is out of scope for this research.

The vehicle safety system that is proposed in this research uses visual data collected from high-tech cameras. As with a better understanding of road landscapes, drivers on the side of the road will have a better sense of the driving space available to them. A self-driving car equipped with this technology can detect dangers such as objects in its path and defects in the road. It is possible that sophisticated sensors, such as cameras, can provide visual input that allows it to identify one or more traffic lights. It's the same with road scenes: a better understanding of the surrounding environment helps drivers better utilize the side-of-the-road terrain for vehicle manoeuvrings.

## 1.4 RESEARCH QUESTION

In short, this paper’s contributions can be summarized as the following:

(1) Study the road surface detection and danger prediction capabilities using RDD Dataset and Road signs Dataset.

(2) To understand if YOLOv5 is sufficient to be embedded into Advanced Driver Assistance Systems to warn the driver of any anticipated threat during driving.

(3) Using customized dataset estimate the metrics for YOLOv5 based predictions and study if the hyper parameter tuning for the different models give any significant changes in threat detection accuracy or speed.

# CHAPTER 2: RELATED WORK

It is now common practice to use an image classifier for an object detection task; these methods involve changing the object's size and location in the test image before using the classifier to identify the object. One well-known illustration is the sliding window strategy Felzenszwalb et al[3] [4]. Because it requires more crops and results in large duplicate calculation from overlapping crops, the R-CNN approach can be time-consuming. Using a Fast R-CNN, this calculation redundancy was eliminated (Girshick, 2015) The You Look Only Once (YOLO) system (Redmon et al., 2016; Redmon and Farhadi, 2016),[5] the Region-based Fully Convolutional Networks (R-FCN) system (Dai et al., 2016),[6] and the Single Shot Multibox Detector (SSD) system are four current object identification systems (Liu et al., 2016).

As part of the EU Road Safety Atlas project, Google has been using ANN CatNet-2 since 2008 for real-time roadside pothole detection on its Google Maps service in Finland and Sweden. By delivering precise information about roadside hazards through real-time monitoring utilizing CNNs and dashcams put on volunteer drivers' cars, the project hoped to increase road safety. The program was so effective that it prevented significant damage to both cars and public property in these volunteer zones by reducing the frequency of potholes by 2-3% annually. However, this system has some drawbacks: Drivers must be aware of potential hazards when changing lanes or crossing railroad lines so they may take appropriate action to avoid them; otherwise, their cars may get damaged or even toppled by these hazards without them recognizing it until after the harm has already been done. By examining the images captured by conventional video cameras, CNN may be used to identify the state of the road's surface. By calculating the brightness, texture, and color saturation of an image, the current condition of the road is examined. Dashcams and CNNs have a great deal of promise for preventing accidents, but drivers using these systems must always be extremely cautious because they are still prone to mishaps even while according to the recommended safety procedures.

Historically, the identification of traffic signs has been based on color and shape patterns, with two connected phases: detection and classification. Traffic signs are detected in a picture after numerous pre-processing procedures, such as data transformation and normalization, which consists of defining regions of interest (ROI) based on color segmentation and "sliding window" approach. Following the step of pattern recognition, the classification phase classifies each sign feature into categories such as "speed limits" and "pedestrian crossing." The likely traffic indicators are then classified using a shallow neural network (a multilayer perceptron, or MLP). Some researchers have employed shallow classifiers, like support vector machines (SVMs) or random forests, in conjunction with local descriptors, like the histogram of oriented gradient (HOG), for successful feature extraction and classification. Hmida et al. demonstrated a traffic sign identification system based on linear SVMs and the MNIST dataset, for instance. Gecer et al. developed a high-performance technique for identifying traffic signs based on blob detectors and SVM classifiers, which boosted the model's color discriminating power by achieving a 98.94% accuracy rate. Due to the wide diversity of road signs in unexpected locations, hidden and diminutive road signs, and varying weather conditions (e.g., shadows and lightning), it is difficult to discern them using conventional methods; therefore, deep learning techniques are employed.

## 2.1 AVAILABLE NEURAL NETWORK MODELS

During this research, we will be doing a comparison study of the following Computer vision CNN models based on several published research papers:

1. **Vision Transformer**
2. **OpenAI Clip**
3. **ResNet**
4. **EfficientNet**
5. **EfficientDET**
6. **Mask RCNN**
7. **Faster RCNN**
8. **Detectron2**
9. **Unet**
10. **Deeplab**

### 2.1.1 VISION TRANSFORMER VS RESNET

The Vision Transformer or ViT is a machine learning model for image classification.[7] Transformer architecture can be considered the de facto standard for natural language processing since it was proposed by Vaswani et al. (2017). Inspired by the success of Transformer Architecture for NLP, Alexey et al. (2021) propose Transformers for Image Recognition at Scale.

The approach mentioned is to directly apply Transformer architecture with minimal changes to train it with supervision. The dataset chosen was ImageNet and it yielded moderate accuracy less than ResNets of similar size. The advantage of CNN noted in comparison with the above was the inductive biases such as translation equivariance and locality. However, when trained on large dataset (14M-300M images) it yields high accuracy from 77 to 94.5% because inductive bias becomes insignificant. The larger models are comparable with the state-of-the-art CNN as claimed by the authors. Also, the study that has been conducted was using generic image datasets not optimized for any specific purpose.

### 2.1.2 EFFICIENTDET, YOLOV5 AND EFFICIENTNET

Renjie Xu et al. (2021) describes a study conducted to detect forest fires, an ensemble learning using Yolov5, EfficientDET and EfficientNET. Contemporary studies involving RCNN, and SSD are cited by the authors. Characteristics and requirements of forest fire detection and road hazard detection are of different in nature. However, the study performs a critical analysis of different neural networks and enumerates the pros and cons of each. In the case of road hazard detection, it requires only to identify objects fast enough and the target image dataset is of limited dispersion. Hence Yolov5 will be the necessary and sufficient neural network for our case.

The authors have selected Yolov5 as it is a real-time object detector. It has cross stage partial network (CSPNet) built into Darknet making CSP Darknet. It solves the problem of repeated gradient information. It thus captures gradient changes into feature map. This will in effect improve the speed by reducing FLOPS (floating-point operations per second).

Yolov5 has path aggregation network (PANet) in its neck. This incorporates a feature pyramid network (FPN) which allows for propagation of low-level features. FPN improves the location accuracy of the detected objects. The head of the Yolov5 generates 3 types of feature maps (18 x 18, 36 x 36, 72 x 72) and achieves multiscale prediction.

EfficientDET was developed by Google and has excellent performance when processing Pascal VOC and Microsoft COCO datasets. It has the capacity to learn complex features and employees a better PANet called bi-directional feature pyramid network (Bi-FPN) which enables fast feature fusion. EfficientDet was proposed by M. Tan et al. in 2020. D7x configuration of EfficientDet (AP of 55.1 on MS-COCO dataset) is new state-of-the-art average precision (AP). It introduces learnable weights which helps to identify the relevance of input features. In comparison with Yolov5’s neck PANet, Bi-FPN gives performances with less parameters and FLOPS. Also, it embeds compound scaling that uniformly scales resolution, depth, and width for prediction network as well as feature network. This helps to improve the accuracy and efficiency.

EfficientNet is also a network proposed by Google which introduced compound scaling as discussed above. This has made it outperform ResNet, DenseNet and ResNeXt in image classification task. This network is a candidate when we can trade-off between accuracy and efficiency.

CNNs depend on the translational invariance. This means that when a car is identified there could be many positions where the wheels could be. So, at low-level translational invariant features are captured and higher levels, high level feature and or a combination of low-level features.

The impressive deep learning technique of real-time convolutional neural networks (RCNN) is used for image segmentation and object detection. They've been specifically engineered to pick up the presence of an object. Selective Search is used in both RCNN and Fast-RCNN neural networks. Greediness is at the heart of Selective Search's design. Best Result is not often guaranteed by Greedy Algorithms. In addition, it must be repeated numerous times. RCNN, performs about 2000 iterations of selective search on the image. By running the CNN only once per image, Ross Girshick (the creator of RCNN) came up with the idea of sharing this computational burden between the 2,000 regions of the image. The R-CNN architecture uses a selective search process to generate a region proposal network for bounding boxes. In order to generate a feature vector map, a CNN is used to warp these region proposals into standard squares. Features from the image are used to create an output dense layer that is then fed into a classification algorithm to help identify the objects located within the region proposal network and assign them a class. Aside from predicting precision gains, the algorithm also predicts offset values that will be used to improve the region proposal. Convolutional feature maps are generated by feeding the input image to the CNN, which in turn generates them. These maps are used to identify the regions where proposals have been submitted. It is then possible to feed all of the proposed regions into the network by using a ROI pooling layer to reshape them into a fixed size. Fast-RCNN extracts all the regions first, then performs a selective search once for each of them. This has the effect of greatly reducing the complexity of time. The final bottleneck, Selective Search, is eliminated by FRCNN. As an alternative, it utilizes the Region Proposal Network (RPN). The regions are fixed in RPN as a n x n grid. It takes less time to run than a selective search. When it comes to computer vision, object detection has long been a vital part of the field. Bounding boxes are a useful tool for describing and identifying the objects in an image and their relative locations. In today's world, there are a variety of ways to accomplish this task. (2) Using a Region Proposal Network to find objects in an image and a second CNN backbone network to fine-tune the generated proposals to make predictions are two of the most common approaches: (1) Single Shot Detection (architectures such as RetinaNet, YOLOv3, and so on); and (2) (two-stage networks such as RCNN, Faster RCNN).

### 2.1.3 RESMLP

The convolution Neural Network uses sum of multiplied matrices by a filter. It has a weight sharing mechanism so that it has fewer number of parameters that deep neural networks. CNNs are not the only mechanism by which we can perform computer vision tasks. The transformer architecture used for Natural Language Processing can also do image classification and has been proved on ImageNet dataset.

Fully connected Layer based image classification was proposed by Touvron et al. (2021) which is called Residual Multilayer Perceptrons(ResMLP). It could perform well on the ImageNet-1k data. ResMLP is based on ViT architecture based on the transformers model.

### 2.1.4 ResMLP WORKING:

ResMLP divides the image into N X N patches where N is 16. Each of those patches is flattened into a vector and those are input into the ResMLP layer independently. The ResMLP will take a matrix X of size d x (N x N) where d is the vector dimension and N x N is the number of patches. The matrix will then undergo several transformations until a matrix Y of the same size as X is obtained.

Company name

Description automatically generated with low confidence

Text

Description automatically generated

GELU is the activation layer and aff is the operator that transforms the column of the input by shifting and rescaling and A B C matrix is the weight learned by the model.

Once the matrix Y is obtained, it will be averaged, and a d-dimensional vector will be derived which will be used as a feature for the linear classifier. Although Highway Network introduced gated shortcut connections and the solution space contains ResNet, ResNet performs better in comparison.

Diagram

Description automatically generated

Figure: 2.1 ResMLP Architecture

2.1.5 RESNET

Universal approximation theorem states that a single layer feedforward network with enough capacity is sufficient to represent any function. The two problems associated with this approach is that the layer can become huge, and overfitting can happen. In order to solve this problem, the number of layers in the neural network is increased. Increasing the depth by just stacking up the layers doesn’t give required results. Deep Networks inherently pose a threat of vanishing gradient problem becoming infinitively small as it is back propagated to the earlier layers. Due to this as the network grows deeper and deeper the performance degrades.

ResNet proposes a solution called “identity shortcut connection”. It skips one or more layers as follows[8]

Diagram

Description automatically generated

Figure 0.1 RESNET

[8]

The shortcut connections turn the network into residual network. The following equation can be used when input and output are the same dimensions.

Text

Description automatically generated with medium confidence

Where x and y are the input and output vectors of the layers. The function n F(x, {Wi}) represents the residual mapping. We have two approaches when dimension increases.

1. The shortcut will perform identity mapping with extra zeros padded for increasing dimensions. This doesn’t require additional parameters.
2. The projection shortcut in the following equation is used to match dimension (1x1 convolutions)

A picture containing text, watch, gauge

Description automatically generated

Where Ws is the linear projection by the shortcut connections to match the dimensions.

For both above-mentioned approaches, when the shortcuts go across feature maps of two sizes, they are performed with a stride of two. The comparison table for plain network vs ResNet is shown below:

Table

Description automatically generated

Figure 0.2 Plain Network vs ResNet

The findings from the comparison performed by[8] as per the table above are the following :

1. 34-layer ResNet is better than 18-layer ResNet by 2.8% and also has lower training errors and thus the degradation problem is well addressed, and accuracy gain is obtained from increased depths.
2. Compared to plain network, the ResNet reduced the error by 3.5% which justifies the effectiveness of residual learning on extremely deep systems.
3. 18-layer ResNet converges faster and thus eases optimization.

A quantitative determination based on extensive road inspection, such as employing a mobile measurement system (MMS) (KOKUSAI KOGYO CO., 2016) or laser-scanning approach (Yu and Salari, 2011), is proposed in Road Damage Detection Using Deep Neural Networks with Images Captured Through a Smartphone. In order to more effectively evaluate a road surface, several attempts have been made to establish a system for assessing road features using a blend of recordings from in-vehicle cameras and image processing technology. An automated method for detecting cracks in asphalt pavement has been developed using image processing techniques and a naive Bayes-based machine-learning approach (Chun et al., 2015), while a system for detecting potholes has been previously proposed using a commercial black-box camera (Jo and Ryu, 2015). Deep neural networks have recently made it feasible to analyze the damage to road surfaces pretty accurately (Zhang et al., 2016; Maeda et al., 2016; Zhang et al., 2017). For instance, CrackNet, which forecasts class scores for all pixels, was introduced by Zhang et al. (Zhang et al., 2017). Such road damage detecting techniques, however, just consider whether there is damage. Although some studies categorize damage according to types—for instance, Zalama et al. (Zalama et al., 2014) classified damage types vertically and horizontally, and Akarsu et al. (Akarsu et al., 2016) divided damage into three types—the majority of studies primarily concentrate on categorizing damages between a few types. Therefore, it is essential to properly classify and detect various forms of road damages in order to create a useful damage detection model for usage by driver assistance systems.

### 2.1.6 R-FCN

Another framework for object detection that Dai et al. suggested is R-FCN (Dai et al., 2016). For precise and effective object detection, it has a region-based, fully convolutional network architecture. Faster R-CNN is significantly faster than Fast R-CNN, but each image needs to have the region-specific component applied hundreds of times. The R-FCN approach takes crops from the final layer of the features before prediction, as opposed to the Faster R-CNN method, which takes crops from the layer where the region suggestions are predicted. This method of deferring cropping till the final layer reduces the amount of per-region computation required. The R-FCN model (using Resnet 101) was demonstrated by Dai et al. (Dai et al., 2016) to frequently attain accuracy comparable to Faster R-CNN at faster running speeds.

### 2.1.7 SSD

A single feed-forward convolutional network is used by SSD (Liu et al., 2016),[9] an object identification system, to forecast classes and anchor offsets without the need for a separate per-proposal classification procedure. The utilization of multi-scale convolutional bounding box outputs coupled to numerous feature mappings at the network's top is the framework's distinguishing characteristic.

### 2.1.8 CRACKNET/CRACKNET-V

Various algorithms and their capabilities to identify cracks have been discussed in ‘Advances in deep learning methods for pavement surface crack detection and identification with visible light visual images’ [10]

Deep learning and machine learning are both learning techniques that rely on data sets. The distinction is that the former's mathematical statement is explicit and understandable, whereas the latter's is implicit. Machine learning is focused on lower-dimensional features, whereas deep learning (deeper neural networks) is focused on higher-dimensional features. As a result, machine learning falls halfway between deep learning techniques and manually created feature engineering.

Common machine learning methods include the Support Vector Machine (SVM), Decision Tree, and Random Forest. Accordingly, the primary algorithms for classifying and identifying cracks are CrackIT, CrackTree, CrackForest, etc.

Since 2012, deep learning has become more popular. However, the "black box" and interpretability issues in the deep learning method's mathematical theory prevent it from being fully understood. However, it is obvious that technology has advanced to SOTA (State-of-the-Art) levels in computer vision (CV), natural language processing (NLP), and reinforcement learning, and has even surpassed the performance of humans in some areas. Scientists are currently working to find theoretical solutions, and engineers are using visualization technology to make the deep learning process easier to understand. Convolutional Neural Network (CNN) learning produces a pattern that is translation invariant and contains spatial hierarchies, two characteristics that are fundamental to higher animal vision (such as that of cats and humans). As a result, algorithms utilizing these two attributes have led to advancements in the field of computer vision identification since 1998, particularly since 2012. CNN-based image recognition algorithms benefit from automatic feature extraction, good generalization, and high precision. It is superior to hand-crafted features and machine learning in various ways. Both in terms of accuracy measurement and recognition impact, ConvNet is unquestionably better than conventional machine learning techniques. ConvNet was a preliminary investigation of the deep learning method for crack recognition that only examined accuracy and left out test speed performance.

When predicting original CFD images using the NVIDIA GTX1080Ti GPU, Structured-Prediction CNN could reach 2.6 frames per second. Structured-Prediction CNN used a small-size input of 27 by 27, and both CFD and AigleRN annotated the cracks at the pixel level (2-pixel mistakes). So, compared to ConvNet, the fracture detecting effect is considerably superior.

3D photographs of the asphalt pavement produced by laser scanning were utilized as input for CrackNet/CrackNet-V. Laser scanning photographs offer a greater resolution than photos taken with visible light cameras, and they also have the ability to filter out some noise and random interference, which is similar to preparing an image. The outcome is a crisper image with accurate labeling and segmentation of the pixel-level cracks.

CrackNet-model V's design includes preprocessing layers (median filtering and Z Normalization). A 1515 convolutional layer, two 11 convolutional layers, and 33 convolutional layers—all of which allude to the VGG architecture—were also often utilized. Because there are no fully-connected layers and the convolution kernel is substantially smaller, CrackNet-V is deeper than CrackNet but lighter, with only 64,113 parameters. Because neither CrackNet nor CrackNet-V have a maximum pooling layer, the input and output picture resolutions are kept constant, allowing for flawless pixel-level segmentation.

CrackNet-V outperforms CrackNet by a small margin. Regardless of the training and testing times (including forward propagation and reverse propagation), the former is around one-fourth as fast as the latter, demonstrating the optimization effect of the CrackNet-V design.

### 2.1.9 YOLOV5 vs RCNN

As part of IEEE BigData 2020, the Global Road Damage Detection Challenge (GRDDC) 2020 was organized, and the dataset RDD was once more expanded. Building road damage detection systems based on data from the Czech Republic, India, and Japan involved 121 teams in this competition. The goal was to execute as accurately as possible, measured by the F1-Score on two specified test data sets, and the photos were gathered from front-facing smartphones put in vehicles. There are 21,041 photos in the training set and 5,295 samples in the test set

The challenges are summarized by Arya et al.[11] The top-12 teams adopted some data augmentation and used proven DL object detection frameworks with Transfer Learning (TL). In order to improve the performance of an ensemble of models at the time of inference, Zhang et al.[12] trained a Conditional GAN (CGAN) to augment the data with artificially created road damage.

The FasterRCNN techniques (rank #9–#12) underperformed the YOLO-based models (rank #1, #2, #4–#6, #8). Results from Test Time Augmentation (TTA) were inconsistent. To concentrate on the road area, many teams used semantic segmentation based on DL.

In the 1980s, automatic road damage detection first became available. It employs SVM or AdaBoost to implement classification after finishing image processing, which mostly relies on conventional edge detection and filtering. These early techniques can't be used in real scenarios since they are sensitive to data diversity. With the quick advancement of artificial intelligence in the area of computer vision, deep learning-based road damage detection has taken center stage in ADRD. Although this has significantly aided the development of ADRD, there are still shortcomings in its accuracy and applicability.

The CNRDD dataset uses the same evaluation methodology as previous detection datasets. As assessment indicators, it employs F1-Score, Precision, Recall, mAP@0.5, and F1-Score. Precision is calculated as the total number of forecasts divided by the number of right predictions. Equation defines it as having a connection to true positives (TP) and false positives (FP). Recall, which is connected to true positives (TP) and false negatives (FN), reflects the correctly predicted fraction in all annotation samples (FN). Equation Precision = TP /(TP + FP) and Recall = TP(TP + FN) can be used to determine it. TP, FN, and FP are connected to Intersection over Union (IoU). The overlap area and union area are divided between the predicted box and the ground-truth (GT) box by the IoU formula. The projected instance is regarded as a TP when the IoU between the expected bounding box and GT exceeds 0.5. A forecast box is referred to be an FP if it only overlaps the GT by 0.5 IoU or less. Additionally, the outcome is counted as an FP if the expected result is not in GT at all or if the predicted damage label differs from the GT label. The number FN represents the instances of road damage that the model either fails to predict or incorrectly predicts.

The detection of road damage is much more challenging than the detection of ordinary objects. This is primarily due to the sparseness of the damage foreground pixels and the low pixel density of the related bounding box. While the bounding box of other example items (such as animals or buildings) has large and dense pixel areas. As a result, there is a significant similarity between the various damage kinds indicated by bounding boxes. This issue can be solved by changing the bounding box to pixel-level annotation and changing the detection task into a segmentation task.

In order to investigate the relative benefits and drawbacks of using one- and two-stage detectors for the job at hand, respectively, they applied the YOLO ("You Only Look Once") and Faster R-CNN computer vision frameworks on the GRDC dataset.

One stage detector like YOLO were previously developed to get around some of these aforementioned inference time bottlenecks. Fast R-CNN and R-CNN models, as measured on popular image dataset benchmarks such as MS COCO and PASCAL VOC 2007, enable real-time detection through approximately 200ms per image test times using GPUs versus 47.0s in the case of R-CNN.

In order to partition an input image into a SxS grid of anchor boxes that are then sent through a number of convolutional layers to produce a set of n bounding boxes with labels, YOLO may be described as "one-stage" due to its bypass of Faster R-CNN's region proposal stage. To remove bounding boxes with areas of overlap greater than a predetermined Intersection over Union ("IoU") threshold, each of these n bounding boxes is then subjected to a time-effective non-maximum suppression ("NMS") technique. With the most recent YOLO version of ultralytics-YOLO ("YOLOv5"), which uses GPUs to provide per-image prediction timings in the 7-10ms context, this one-stage technique allows for significantly faster inference speeds. They investigated YOLOv5 in its different model varieties and discovered that both the YOLOv5-x and l model versions outperformed Faster R-CNN in F1-score and inference time. YOLOv5 was subsequently employed in this strategy as the foundation model architecture.

Much like with traditional tree-based horizontal or vertical ensembling methods like Random Forests or Gradient Boosting, this has the effect of reducing model prediction variance so that improved accuracy may be reached. YOLOv5 models are trained with distinct batch size, learning rate, optimizer, and other hyperparameters, with each model's different kernel patterns learned under its unique set of hyperparameters supplementing those of other included models. Since no single model would be in charge of the final predictions, the price for this greater accuracy would be longer inference times and a less interpretable model. Similar to the first strategy, this second Test Time Augmentation approach aggregates individual model predictions on several augmented image versions created by horizontally flipping and scaling the base test image to resolutions of 1.30x, 0.83x, and 0.67x. Then, using a chosen IoU threshold and a comparison of the bounding box confidence scores, this technique filters these five separate bounding box prediction sets—corresponding to one base and four enhanced images—through the NMS procedure. As a result, this TTA procedure's multiple prediction ensembling allows for less generalization error. In order to theoretically enable real-time detection in the field, these TTA and EM approaches can be combined such that each k set of base and augmented test images produced through TTA can be fed to each of I EM models in order to yield k \* I bounding box prediction sets. Several variations of these YOLOv5x and YOLOv5l configured with different batch size and other hyperparameter values were trained. Following this methodology, an ensemble of six YOLOv5x and YOLOv5l models each trained with 32, 16 and 8 batch sizes for 150 epochs was shown empirically to yield significant improvement over these prior single-model experiments with an F1 score of 0.57, leading to the selection of this ensemble structure as the approach's central component. Given the additional observation that per image inference times increased linearly with the number of models included in this ensemble, the six-model approach was chosen to satisfy this self-imposed 0.5s inference time constraint, producing maximum 0.42ms per image inference times with the vast majority of predictions times falling in the 0.21-0.40ms range.

## 2.2 HAND-CRAFTED FEATURE ENGINEERING

‘Advances in deep learning methods for pavement surface crack detection and identification with visible light visual images’ [10] discusses Edge/morphology/feature detection techniques and many feature transformation (or filtering) algorithms used in hand-crafted feature engineering methods. For instance, the former category includes Canny, Sobel, the Histogram of Oriented Gradient (HOG), Local Binary Pattern (LBP), and the latter category includes the Gabor filter and intensity thresholding. These algorithms use mathematical calculations to gradually extract or analyze the edge, morphology, and other aspects of objects in images. They do not rely on a data source and are not learning methods. Additionally, the majority of mathematical calculations use analytical formulas, which are low on computation and quick. These algorithms' inability to adapt to different random variable components is one of their weaknesses.

## 2.3 CHALLENGES IN COMPUTER VISION

Typically, any computer vision problem falls into any of the following six categories or a combination of them.

1. **Classification**: To identify which class out of the given k class an image belongs to.
2. **Localization**: To identify the bounding boxes for all instances for a given class k
3. **Detection**: To identify the bounding box of all instances of k classes
4. **Semantic Segmentation**: Pixel classification into k classes
5. **Instance Segmentation**: Classify pixels into k classes and identify the instance.
6. **Content-based Image Retrieval**: Identify u images like a given image x from a list of n images.

### 2.3.1 IMAGE CLASSIFICATION

Image classification is a classic machine learning problem. Although it is very easy for humans to recognize objects like handwriting, warning signs, people, and things, it is a complex process to make computers recognize objects from mere videos or images. There are a variety of techniques that have been applied to solve this problem, like Harr Cascade Classifier, Template Matching, State Vector Machines, etc. However, Convolution Neural Networks (CNNs) generally have been very effective in such complex tasks involving unseen images. The image selected for CNNs processing will be analysed based on its height, width, and channel (grayscale or RGB). The height and width of the image together represent the number of pixels of the image. Convolutional layers in the network process the image and generate a smaller set of features which will be passed to the next layer. Pooling layers perform the down sampling along the spatial dimensions. The fully connected layer does the classification by assigning scores by assessing the features extracted from the previous layer.

Image classification algorithms determine the class of the image while image detection algorithms draw a bounding box around the object found in the image. There can be many bounding boxes based on the number of objects identified in the image. To cater to this, the neural network needs a variable-size output layer. This is not in accordance with the normal convolutional neural network. The R-CNN algorithm by Girshick et al. [13]extracts 2000 regions from the image which is fed and then proposed regions are resized and passed into a CNN. The CNN would then classify the input regions. A modified form of this algorithm is the Fast R-CNN.

### 2.3.2 IMAGE LOCALIZATION

The process of locating an image within an image is known as image localization. By comparing an object's current pixel arrangement to a previously recorded one, a computer may determine where it is. Robot hands and navigational systems are two applications that use image localization. To determine a user's position and direct them to their destination, navigation systems use picture localization. Robots can grab delicate things via image localisation without endangering themselves or the objects. Using image localization, Google's prototype Bristlebot, for instance, paints with bristles rather than human fingers.

### 2.3.3 OBJECT DETECTION

Identifying items in an image or collection of photos is the process of object detection. To detect objects, numerous computer vision algorithms have been created. Stereo correspondence, dense feature extraction, stereo matching, Kohonen neural networks, and variant-based algorithms are a few examples of object detecting techniques. The development of object detection systems has made use of numerous machine learning methods, however these algorithms are not without their own complications. Numerous object detection systems have been created for a variety of uses, including, to mention a few, traffic monitoring, surveillance, and augmented reality.

### 2.3.4 IMAGE SEGMENTATION

Segmentation is the process of classifying the pixels of an input image into k classes. While semantic segmentation just identifies the k classes, instance segmentation identifies every instance of a class. It uses Fully Convolutional Network (FCN) which is a CNN without fully connected layers (FC). It can be designed as an encoder follower by a decoder and in Yolo both are FCN. Yolo is regression based and image is split into ‘s x s‘ grid cells .An object whose centre matches with a gird cell is predicted. The CNN predicts as follows;

* B number of bounding boxes (x, y, w, h). (x, y) is identified as the centre of the bounding box relative to the cell position. Confidence score is calculated as IOU of predicted bounding box with ground truth bounding box.
* C conditional probabilities are also predicted for each grid cell (one per class)

Prediction of CNN is (S, S, (B \*5+ C)); since there are 5 elements for boundary box. Object detection draws a bounding box around the objects detected in a processed image. However, image segmentation goes one step further to identify the outline of the object. Modelling for this is very difficult and should be applied only it is necessary.

Different segmentation techniques are explored in the paper Nevriyanto et al. [14]. Using the segmentation technique known as "Region Growing," pixels in a given region that share a seed point are grouped together. For instance, if the threshold is exceeded by the size of similarity of one or more adjacent pixels, those pixels are equal. They'll be grouped together as well [3]. Until there are no more identical pixels, surrounding pixels are grouped together. But in this study, the growing region process begins with the initial initialization of the region, which is derived through the binary segmentation, making the time complexity of the growing region process faster.

The most effective technique for intensity-based segmentation is called adaptive thresholding. In addition to being more common, thresholding may also be simply implemented computationally faster in picture improvement.

Starting with the pixels, the region-expansion approach is developed. Additionally, if the resulting zone continues to match the homogeneity requirements, it will expand the surrounding area. the segmentation method that works from the bottom up, starting with a single pixel (also known as a "seed") and ending with a segmented area. The following are the main driving forces in the expanding area: A similarity criterion's selection, a seed point's choice, and a stopping rule's definition.

Intelligent Thresholding Adaptive Thresholding uses pixel-by-pixel threshold block processing. Block sizes are often user-specific, avoiding two extreme situations: too small blocks, which may take a lot of processing time to calculate, and huge blocks, which may yield results that aren't significantly better than those produced with global thresholding. For block processing in Matlab, use the blkproc function.

A morphologic-based transformation function for picture segmentation is the watershed transformation. Because it has the advantages of speed and simplicity, the usage of watershed is acknowledged as a powerful technique. The result of the watershed transformation is a portion of the image made up of watershed pixels and a region (a group of pixels related to a local minimum). Without any additional processing, the watershed transformation of a gradient image typically produces an over-segmented image. We typically use the original image or gradient function (image filtered) or can also use a marker to decrease the smallest local value that is filtered in order to solve this problem. Additionally, the issue of over-segmentation can be resolved by correcting some images after the process, such as combining the neighbouring region adaptive thresholding segmentation, which shows segmentation based on light intensity. In other words, it cannot be used optimally for the introduction between background and foreground (object), even though it is a very quick computation processing method, because it does not meet satisfactory results. While the expanding region exhibits segmentation, the nearby pixel (neighbour) still meets the inhomogeneity condition, and as a result, the result is sufficient to introduce the background and foreground. Unfortunately, the computation procedure takes a very long time. Because of slow-moving computational barriers, image number 5 cannot be displayed. The identification of neighbouring pixels keeps repeating itself. Watershed demonstrates that segmentation results only take the region and boundary (according to the ridge line) of the image and cannot take the object of the image (foreground and background are not sufficiently distinct).

A soft computing algorithm is employed for the research, and it is an existing algorithm. It is concluded that the adaptive threshold method is not faster in terms of processing time. Regrettably, the outcome has to be optimized. Although the Region needs processing, adaptive yields superior outcomes. Watershed is fast and has good segmentation, although it struggles with handling detailed objects.

## 2.4 EDGE DETECTION

Edge detection can be thought of as a technique for extracting visually appealing object borders and edges from natural photos. Because of its extensive use in numerous high-level applications, including as object detection, object proposal generation, and picture segmentation. A fundamental low-level issue in computer vision is edge detection. The edge and non-edge pixels are distinguished using complex learning paradigms after traditional approaches have extracted the local cues of brightness, color, gradient, and texture or other manually generated features like Pb and gPb. This field has advanced greatly thanks to certain well-known CNN-based techniques including DeepEdge, N4 -Fields, DeepContour, and HED.

In order to do pixel-wise prediction for edge detection in an image-to-image manner, Liu et al. [15] proposes richer convolutional features (RCF), a novel deep structure fully utilizing the CNN features from all the convolution layers. RCF has the ability to automatically learn to incorporate complementary data from all CNN layers, resulting in accurate representations of objects or object portions at various scales. The evaluation's findings show that RCF excels at edge identification.

CNNs and the closest neighbour search are combined in the N 4-Fields that were proposed by Ganin et al. Shen et al. divided the subclasses of the contour data and then fitted each subclass by learning the model's parameters. A recent edge detector called HED, developed by Xie et al., does image-to-image training and prediction. Their side output layers, which are made up of one conv layer with kernel size 1, one deconv layer, and one softmax layer, are connected to the final conv layer of each stage in VGG16 by this holistically nested architecture. Additionally, Liu et al. exploited the loose labels produced by bottom-up edges to direct the training of HED. Wang et al. successfully learned sharp boundaries by employing a top-down backward refinement route. A hierarchical deep model was developed by Xu et al. to effectively combine edge representations that were learned at various scales. By concurrently detecting and recognizing the semantic categories of edge pixels, Yu et al. expanded the success of edge detection to semantic edge detection.

A unified approach to contour detection and image segmentation is covered in a different study called Contour Detection and Hierarchical Image Segmentation. They have made the following contributions: a high-performance contour detector that combines local and global image information; a technique to convert any contour signal into a hierarchy of regions while maintaining contour quality; substantial quantitative evaluation; and the publication of a new annotated dataset. employing the precision-recall methodology described in, the assessment of several contour detection and picture segmentation methods on the Berkeley Segmentation Dataset (BSDS300). By treating region borders as contours, this benchmark permits evaluation of segmentations within the same framework by comparing machine-generated contours to human ground truth data.

### 2.4.1 HOLISTICALLY NESTED EDGE DETECTION

Early contour detection methods seek to quantify the presence of a boundary by local measurements at a specific image location. Edges are found using the Roberts, Sobel, and Prewitt operators by combining local derivative filters with a grayscale image. Zero crossings of the Gaussian operator's Laplacian are used by Marr and Hildreth. Additionally, the Canny detector incorporates non-maximum suppression and hysteresis thresholding stages, modelling edges as sharp discontinuities in the brightness channel. By taking into account how the image responds to a family of filters with various scales and orientations, a deeper description can be obtained. The Oriented Energy method, which employs quadrature pairs of even and odd symmetric filters, is one example. A filter-based approach with an automatic scale selection mechanism is what Lindeberg suggests. More recent local approaches use learning techniques for cue mixing and take into account color and texture information. In order to forecast edge strength, Martin et al. create gradient operators for the brightness, color, and texture channels. These operators are then used as input to a logistic regression classifier. Dollar et al. provide a Boosted Edge Learning (BEL) approach that aims to build an edge classifier in the form of a probabilistic boosting tree from thousands of simple features computed on image patches rather than relying on such hand-crafted features. This method has the benefit of perhaps handling cues like parallelism and completion during the initial classification stage. In order to develop edge detectors, Mairal et al. learn discriminative sparse representations of local image patches. They develop a discriminative dictionary for each class and feed the reconstruction error they receive from each dictionary as a feature into a final classifier. For these contemporary local techniques, the wide range of scales at which objects may appear in the image remains a challenge. Combining data from various scales of the local operators created by Ren is advantageous. The boundary classifier is given more localization and relative contrast cues, which are described in terms of the multiscale detector output. The localization cue measures the separation between a pixel and the closest peak response for each scale. Each pixel is normalized in terms of the local neighbourhood by the relative contrast cue. An orthogonal area of research in contour detection concentrates mostly on globalization, another level of processing that makes use of local detector output. The simplest of these methods identifies extended, smooth contours by connecting high-gradient edge segments. Several of the most recent high-performance approaches benchmarked have more advanced stages of globalization, which can be distinguished by their usage of local edge detection operators of In order to enforce curvilinear continuity of contours, Ren et al. employ the Conditional Random Fields (CRF) framework. On top of locally discovered contours, they compute a constrained Delaunay triangulation (CDT), which results in a graph that includes the detected contours and any new "completion" edges that the triangulation introduces. The CDT fills in small gaps in the detected contours and is scale-invariant. They build a CRF with edge potentials in terms of detector response and vertex potentials in terms of junction type and continuation smoothness by associating a random variable with each contour and completion edge. To calculate expectations, they employ loopy belief propagation. For the purpose of separating conspicuous smooth curves from the output of a local contour detector, Felzenszwalb and McAllester employ a different method. They take into account the collection of brief oriented line segments that link adjacent pixels in the image. Each of these segments either a backdrop segment or a curve segment. They make the following assumptions: detector responses are conditionally independent given the labelling of line segments; curves are derived from a Markov process; the prior distribution on curves favours a small number of curves per scene. Finding the best line segment labelling becomes a generic weighted min-cover issue where the objects covering the line segments are chosen from a set of all feasible curves and background line segments. The elements being covered are the line segments themselves. An approximate solution to this NP-hard problem is discovered using a greedy "cost per pixel" heuristic. Beginning with the output, Zhu et al. also construct a weighted edgel graph, where the weights represent directed collinearity between adjacent edgels. They propose using the intricate eigenvectors of the normalized random walk matrix to identify closed topological cycles in this graph. Since edgel chains are permitted to loop back at their termination points, this approach recovers both closed contours and smooth curves.

Statistical Edges, Pb, and gPb are examples of approaches driven by II: information theory on top of features obtained through meticulous manual design. BEL, Multi-scale, Sketch Tokens, and Structured Edges are examples of learning-based systems that continue to rely on aspects of human design. Additionally, a recent wave of developments using convolutional neural networks, such as N4 -Fields, DeepContour, DeepEdge, and CSCNN, underline the significance of autonomous hierarchical feature learning. Due to its cutting-edge performance on the BSD500 dataset (with, for example, an F-score of.746) and its practically significant speed of 2.5 frames per second, the Structured Edges method (typically abbreviated SE) arose as one of the most renowned systems for edge detection prior to this aggressive innovation in deep learning. Recent CNN-based techniques have shown encouraging F-score performance gains over SE. The F-score performance and speed of these CNN-based algorithms can still be significantly improved, as the current time to create a forecast varies from a few seconds to a few hours (even when using modern GPUs).

If we want to come close to the human ability to resolve ambiguity in natural picture edge and object boundary recognition, we need a method called holistically nested edge detection (HED), which automatically learns the type of rich hierarchical features that are essential. The term "holistic" is employed because HED tries to train and forecast edges in an image-to-image manner even if it does not explicitly model organized output. The study emphasizes the inherited and gradually enhanced edge maps created as side outputs while using the word "nested."

Each of these edge maps follows the same path along which each prediction is made, with subsequent edge maps being shorter. In contrast to earlier multi-scale techniques, where scale-space edge fields are neither automatically learnt nor hierarchically coupled, this incorporated learning of hierarchical characteristics.

The proposed holistically-nested edge detector (HED) addresses two pressing problems: (1) holistic image training and prediction for image-to-image classification, inspired by fully convolutional neural networks (the system accepts an input image and immediately outputs an edge map image); and (2) nested multi-scale feature learning, inspired by deeply-supervised nets, which applies deep layer supervision to "guide" early classification results. It was discovered to be precise and computationally effective.

A single-stream deep network with numerous side outputs makes up the architecture. The deeply-supervised net technique, in which the authors demonstrate how hidden layer supervision can enhance both optimization and generalization for image classification problems, is the most similar to multiple prior publications. If a unified output is wanted, the numerous side outputs also provide us the freedom to build another fusion layer.

The framework is implemented using the publicly accessible Caffe Library and is built upon the publicly accessible FCN and DSN implementations. As a result, hardly much engineering hacking is needed. The entire network is tuned in the proposed HED system starting from an initialization using a pre-trained VGG-16 Net model.

## 2.5 TOP-DOWN FULLY CONVOLUTIONAL ENCODER-DECODER NETWORK

Two significant problems with this low-level vision problem are resolved by the suggested strategy, known as TD-CEDN: (1) learning multi-scale and multi-level features; and (2) implementing a successful top-down refined approach in the networks. TD-CEDN executes the pixel-wise prediction by utilizing features at all network tiers. A coarse feature map in a feedforward pass is trained following the encoder stage, in contrast to skip connections and earlier encoder-decoder approaches, and then modify this feature map in a top-down manner during the decoder stage using features at increasingly lower layers. As a result, the deconvolutional process is carried out in stages under the direction of Deeply-Monitoring Net, which offers integrated direct supervision. The aforementioned suggested technologies result in a clearer and more accurate prognosis. On the BSDS500 dataset (ODS F-score of 0.788), the PASCAL VOC2012 dataset (ODS F-score of 0.588), and the NYU Depth dataset (ODS F-score of 0.588), the suggested approach attained the state-of-the-art (ODS F-score of 0.735).

## 2.6 SPATIALLY COHERENT RANDOM FORESTS

RFs that produce spatially coherent clusters are known as spatially coherent random forests (SCRFs). The manner in which each node assesses its split function sets them apart from conventional RFs. We suggest adding a regularization term that represents the spatial coherency of the resulting split in the image plane. Whereas in regular RFs, a split function is simply selected based on an information gain, this is not the case in our proposal. The split functions selected at each node are influenced by the weighting of the spatial coherency term and the information gain. The selection will be skewed toward split functions that produce finer splits in the image plane, and consequently smoother pixel labeling or image segmentations, as the weight is increased.

For Contour Estimation, SCRF Each tree cuts the image into pieces, and we want to combine all T of the trees in the forest to create a single, coherent contour map of the picture. Estimating the contour map of a single tree segmentation is the initial stage. Then, by averaging the maps of the forest's T trees, we can create a contour map of the forest. The estimation is made more reliable and is kept from being significantly impacted by the randomness of each tree by averaging the contour maps. All of the information that can be gleaned from a single tree should be taken into consideration when attempting to produce a contour map of an image utilizing that tree. For instance, a flat contour map is produced when only the segmentation defined by the tree's leaves is used, which downplays the significance of the various contours. Our fundamental premise is that we may infer information about the picture contour from each split in the tree and from its depth.

Random Forest was extended to accommodate spatial reasoning. Spatially coherent labeling is encouraged by the new forest, known as Spatially Coherent Random Forest (SCRF). In SCRF, every split function is assessed using a conventional information gain metric that has been regularized by a spatial coherency term. We have employed SCRF as a stand-in for image segmentation to detect contours in real-world photos. The image plane is segmented by each tree in the forest, and the edges of all of the segmentations are combined to create a final hierarchical contour map. The suggested improvement is simple to apply and has no negative effects on the advantages of Random Forests that have made them so well-liked. The standard Berkeley Segmentation Datasets were used to analyze the technique SCRF, and it was discovered that it performed roughly 10% better than conventional Random Forests.

## 2.7 DEEPEDGE: A MULTI-SCALE BIFURCATED DEEP NETWORK

It can be demonstrated that, in contrast to conventional methods, we are able to flip the pipeline that has long been in place. Specifically, we use object-related features as high-level contour detection cues rather than detecting contours with low-level cues for a higher-level identification task. We do this using a multi-scale deep network made up of a bifurcated fully connected sub-network and five convolutional layers. The component of the pre-trained network that is fixed and taken directly from a large-scale object classification challenge is the section from the input layer to the fifth convolutional layer. Four alternative scales of the picture input are applied to this component of the network. Then, a bifurcated sub-network made up of two independently trained branches is connected to these four identical parallel streams. With a classification aim, one branch learns to forecast the likelihood of a contour, whilst the other branch is trained to learn the percentage of human labelers who agree that a contour is present at a particular site.

In the paper, a front-to-end convolutional architecture is shown in which contours are immediately learned from unprocessed pixels. The first five convolutional layers of the network created by Krizhevsky et al. are reused in the proposed deep learning architecture to produce the features. This network is referred to as the KNet. Reusing the KNet's features allows our method to integrate object-level data for contour prediction because the KNet has been trained for object classification. It is demonstrated in the experimental procedure how the results of contour detection are significantly enhanced by this high-level object information. Additionally, the defined architecture integrates local and global information from the image while operating on multiple scales simultaneously, which results in much higher contour identification accuracy rates. A trained subnetwork that splits into two branches connects the features calculated by the KNet's convolutional layers at four different input sizes (the architecture of our model is illustrated in Fig. 2). What should the goal of the lesson be? A human observer uses object level reasoning and a variety of supporting evidence to determine if a pixel represents a boundary edge. Despite the fact that it is impossible to record this data, we do have the percentage of observers that agree for each pixel. It stands to reason that a learning objective that forecasts the percentage of human labelers who will agree will better emulate human thinking. As a result, we optimize the two branches in the bifurcated sub-network using various learning objectives. When the other branch is trained to anticipate the percentage of human labelers in agreement using a regression criterion, the weights in one branch are optimized using an edge classification target. It can be demonstrated that while the outcomes of the regression branch have high accuracy, forecasts from the classification branch produce predictions with high edge recall. Thus, combining these two outputs results in cutting-edge F-score and average precision as well as good results for both measures.

## 2.8 ACTIVE CONTOUR

The active contour method is suggested by the Video-based Vehicles Classification (VC-VC) system by contrasting the current image with a background image. In order to segment an image, it is utilized to obtain deformable models or structures with restrictions and forces. In order to create a parametric curve or contour, contour models explain the edges of objects or any other aspects of an image. A segment with a width of one pixel and a length of one or more pixels is referred to as an active contour. Contours and boundary detection techniques provide the essential picture data for object representation and identification. They used the length and shape of their contour pixels, for instance, to compute item sizes, classify forms, and identify the feature points of objects. They also used these contour pixels to distinguish objects from their backgrounds. The shape of objects can also be preserved and restored to its original shapes using contour information in graphics and vision for a range of purposes. As a result, a great deal of research has been done on contour-tracing algorithms for extracting and tracking an object's contour. The four criteria that are used to evaluate a contour tracing method are traceability, processing time, data size to maintain the traced contour information, and the capacity to accurately restore and enlarge the original contour using the recorded data.

## 2.9 CONVOLUTIONAL NEURAL NETWORKS

Artificial Neural Networks are used to simulate the decision process of biological neurons using computational networks. They have multiple layers (at least three). All the neurons in a layer are connected to every neuron in the adjacent layers. This is comparable to weighted graphs in which nodes are neurons and connections between the neurons are the directed edges of the graph. Depending upon the prediction validations the weights are adjusted to make further predictions more accurate. This process is called learning and is repeated several times until the errors in the prediction is very small or acceptable after a certain number of iterations called epochs.

Convolution Neural Networks (CNNs) are a category of deep learning containing layers which extract feature maps from images using different kernels. A convolution layer processes the input image using a defined window. The window can be parametrized to extract features. This window is called a filter. It gives output of only the features it has identified in the input image. This output is referred to as a feature map. Typically, multiple such filters are applied. The filters are not explicitly defined but the network automatically learns the parameters during training. Once the feature mappings are created, the depth of the feature map stack decided by the number of layers of filter it has. The filters could have different with and height, but the depth should match with the input.

Then there are the pooling layers which does the dimensionality reduction. Pooling layers compresses the spatial information. There are different types of pooling called average pooling, min pooling and max pooling. The max pooling technique simply yields the maximum value in a window per scanned location. This is the most widely used pooling technique. Another technique called global pooling matches the input dimensions and thus compresses the size of feature mappings into a single value. Strided convolution is a learned down sampling than explicitly mentioning the technique. The stride length defines the shifting unit size.

The purpose of layers can also differ such as dropout and dense layers. In convolution layer the connection of a neuron is only to a local area of input neurons to reduce the learning parameters while going into deeper layers. Usually, the architecture would consist of a stack of many convolution layers, pooling layers followed by fully connected layers. Passing of the layer from the input to the output is called forward propagation. They are very effective and the layers in the network which are closer to the input learn low level features while layers deeper learn high order-features. They learn the spatial hierarchies of features with the help of backward propagation adaptively.

Typically, a pooling layer follows one or two convolution layers which forms a block together and it is repeated until the feature map becomes small enough add traditional layers. The reshaping of a multidimensional layer to a one dimensional fully connected layer is called flattening.

## 2.10 ROAD SIGN DETECTION

Road sign detection and recognition using digital image processing and neural network-based techniques are covered in A Study on Traditional and CNN Based Computer Vision Sensors for Detection and Recognition of Road Signs with Realization for ADAS. The procedure is typically broken down into three steps, namely the detection of road signs to find prospective candidates for road signs, the verification of the candidates found during the detection of road signs, and the final step. In order to create the actual information from the detected and validated indicators, traffic signs must first be recognized. Road sign detection and recognition for ADAS can be done using both CNN-based and DIP-based techniques.

To detect and identify the speed limit signs in Norway, Torresen et al.[16] provide a red-colored circular speed limit signs detection method. A reliable visual speed limit sign identification and recognition system for American and European speed limit signs is presented by Moutarde et al.[17] In order to detect and identify speed restriction signs in the United States of America, Keller et al.[18] describe a rectangular speed limit sign identification technique (U.S.A.). Liu et al.[9] employ a different strategy, combining the Fast Radial Symmetric Transform methodology with the de-noising method based on the histogram of oriented gradients (HOG) to identify circular speed limit signs. Color segmentation is used by both Zumra et al.[19] and Vavilin et al. [20]before using additional digital processing techniques. To classify the traffic signs, Lipo et al. [21]describe an approach that fuses camera and LIDAR data, followed by the HOG and linear SVM. The evaluation of the identification of traffic signs in actual situations is presented by Sebastian et al. [22] The Viola-Jones detector, which is based on the Haar features and Histogram of Orientated Gradients (HOG), relies on linear classifiers to recognize traffic signs. On the benchmark GTSDB, model-based Hough-like voting techniques are tested. Additionally, it addresses various approaches put out by Ming et al.[23] which employ two distinct supervised modules for detection and recognition, respectively. Gangyi et al.[24] offer the approach that uses the HOG and a coarse-to-fine sliding window scheme for the detection and recognition of traffic signs, respectively. Markus et al.[25] use contemporary variations of HOG features for detection and sparse representations for classification. Supreeth et al.[26] provide a color and shape based detection technique for the auto associative neural network-based recognition of red-colored traffic signs. A traffic sign detection and identification technique is presented by Nadra Ben and colleagues with the goal of identifying and following prohibitory signs. Then Traffic signs are recognized using feature vector extraction and the Support vector mechanism (SVM), and the identified traffic signs are then tracked using the Lucas-Kanade tracker's optical-flow based technique. To find rectangular patterns, Y. Chang et al.[27] used a modified radial symmetric transform, and to filter out false positives, they used an AdaBoost detector based on Haar-like features. For the detection and recognition of North American speed limit signs, Abdelhamid Mammeri et al. [28]suggested an algorithm. There are many cutting-edge studies based on various CNN models to find and identify traffic signs, including some hybrid methods.

## 2.11 ROAD DAMAGE DETECTION USING UNSUPERVISED DISPARITY MAP SEGMENTATION

Road Damage Detection Based on Unsupervised Disparity Map Segmentation[29]discusses a technique to more clearly discern between sections of damaged and undamaged roads, road disparity maps were modified. We immediately obtain the numerical solution for the energy minimization problem given in rather than estimating the transformation parameters using nonlinear optimization techniques like GSS-DP and GD. Without using any parameters, the suggested approach can segment dense disparity maps for identifying road damage. Furthermore, the method may be applied for estimating vehicle state because the stereo rig roll angle can be precisely predicted from disparity maps. Additionally, this method can be used to automatically label training data for the detection of road damage. The suggested technique for detecting road damage computes the precision, recall, Fscore, IoU, and accuracy, among other metrics, at the pixel level. It is evident that the suggested technique for detecting road damage works well. The detected road damage regions have a pixel-level accuracy of about 97.56%.

Dongjun Jeong in his paper ‘Road Damage Detection Using YOLO with Smartphone Images’ [30] discusses road damage detection using YOLOv5 using the pictures captured using smartphone camera.

The model which he uses for this purpose is YOLOv5x and the dataset is same as the one proposed in this thesis which is IEEE BigData Cup Challenge 2020 for road damage detection. In the Japanese dataset, he removed the bounding boxes for pointless classes, and each image is examined to see if an object is there. He could use 1,072 photographs for the Czech Republic, 3,223 images for India, and 7,900 images for Japan to train our model after pre-processing the data. He constructed numerous scenarios to test the performance between merely feeding single nation images and using two different countries together in the Czech Republic, India, and Japan, building on prior research with the conjugation of multiple source road photographs. In order to train our models, we divided the input photos into training datasets (80%) and validation datasets (20%). We then employed several data augmentation techniques, such as 5-Fold cross-validation, image translation, image scale, mosaic, and hue, saturation, and HSV values. The SGD optimizer, learning rate 0.01, momentum 0.937, and weight decay 0.0005 were used as the default hyper-parameters.

With 50 epochs and 32 batch sizes, the model is claimed to be trained stably and showed steady performance. The experimental results were performed with a confidence threshold of 0.4, non-maximum suppression (NMS) with the intersection of union (IoU) 0.5. The average performance claimed to have achieved is due to the combination of datasets he tried among the Czech, India and Japan road images. The inference speed for the same are listed as shown in the table.

Table

Description automatically generated

Figure 2.4 Dataset Ensembles

Source [30]

## 2.12 CONVENTIONAL TECHNIQUES FOR IMAGE PROCESSING

Image processing is the procedure used to perform certain operations on images with the goal of obtaining an upgraded image or extracting relevant, comprehensible information. With the exception that the input is an image and the output is either an image or attributes related to that picture, it is comparable to signal processing. The two main categories of image processing techniques are analogue and digital image processing. Analog image processing (AIP) is the process of using printouts and images for basic interpretation and analysis. However, digital image processing (DIP) approaches are methods for manipulating images digitally through the use of computers. The fundamental, common procedures that all the data in DIP goes through are pre-processing, augmentation, information extraction, and display.

The process of detecting and recognizing speed restriction road signs can be broadly divided into three stages, including (i) detecting speed limit signs, (ii) segmenting digital data, (iii) recognizing digital data, and (iv) recognizing digits. The process of detecting and recognizing speed regulatory road signs can also be broadly divided into three stages, including (i) detecting speed regulatory signs, (ii) feature extraction, and (iv) feature matching.

The axes of radial symmetry are used in the idea of a radial symmetric transform. The radial symmetric transform operates based on the several symmetry axes that the typical polygenes of n-sides feature. Each pixel's gradient is used to determine how each vote is cast. Voting is based on the gradient's direction.

It also compares YOLO to a more recent CNN model called "CSPJacinto-SSD" for the detection and recognition of traffic signs. JacintoNet, a straightforward, lightweight model made of of convolution, group convolution, and max-pooling layers, now includes the CSPNet capabilities. It has been demonstrated that the Cross Stage Partial (CSP) feature increases accuracy while lowering model complexity and parameter requirements. At the input of each stage, the role of CSP is to simply divide the feature maps into two parts along the channels; one part sends into the convolution block as usual, while the other half skips all layers and concatenates with the output convolution block as the final block output. Prior to the convolution block, the feature channels are increased using 1 x 1 convolution, and the context of features from the CSP layer is combined using 1 x 1 convolution following the convolution block. With a few changes to the anchor boxes based on the multi-head SSD concept suggested in [60], the dense heads used in the planned CSPJacinto-SSD are referred to as those in SSD. There is an additional position of anchor boxes with offset 0, rather than just the previous offset 0.5, at thick head levels 2 to 4. Particularly useful for light-weight SSD models that require more anchor boxes to direct the objects' potential appearance locations, this feature can enhance the density of anchor boxes while boosting the recall of object identification. When compared to the original SSD model, the anchor box settings have changed slightly. Because it might make the anchor borders tighter and keep 1:3 anchors, the anchors 1:2 are altered to 1:1.5. Compared to the original SSD, the anchor boxes' base size has changed.

### 2.12.1 YOLOv5

YOLO in comparison to the above-mentioned networks comes with many versions as YOLO, YOLOv2, YOLOv3 and YOLOv4, YOLOv5, YOLOv6 and YOLOv7 so far. The mechanism and architecture are different. It is a single pass which is defined as ‘You Only Look Once’. During training the input images along with ground truth (segments and labels) is fed into it. Outputs are also bounding boxes which is faster to train and predict. This is due to the tradeoff with the accuracy compared to Faster RCNN.

### 2.12.2 ARCHITECTURAL DETAILS

The Models backbone extracts relevant features from the image threat is fed into it. YOLOv5 makes use of CSP- Cross stage Partial Networks as its backbone to extract feature rich information. It has excellent results with respect to processing time compared to other deep networks.

The YOLOv5 object detector's backbone network is often pretrained using ImageNet classification. Pretraining refers to the network's weights being modified for the new task of object recognition even when they have already been trained to recognize important features in an image.

For the YOLOv5 object detector, the authors took into account the following neural backbones.

* CSPResNext50
* CSPDarknet53
* EfficientNet-B3

Table

Description automatically generated

Figure 0.3 Parameters of Neural Networks

DenseNet serves as the foundation for both the CSPResNext50 and the CSPDarknet53. It was created to connect layers in convolutional neural networks with the following goals in mind: to improve feature propagation, encourage the network to reuse features, and decrease the number of network parameters; to address the vanishing gradient problem (it is difficult to backprop loss signals through an extremely deep network).

Diagram, engineering drawing

Description automatically generated

Figure 2.0.4 DenseNet

Source: [31]

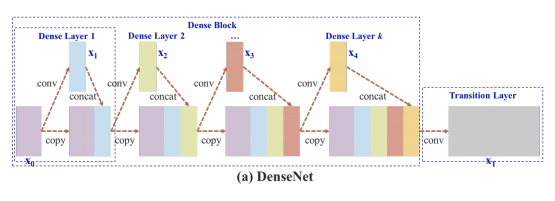


Figure 0.5 DenseNet

Source: [31]

DenseNet has been modified in CSPResNext50 and CSPDarknet53 so that the feature map of the base layer is separated by copying it, passing one copy through the dense block, and passing another directly to the next layer. By transmitting an unedited version of the feature map, the CSPResNext50 and CSPDarknet53 aim to reduce processing bottlenecks in the DenseNet and enhance learning.

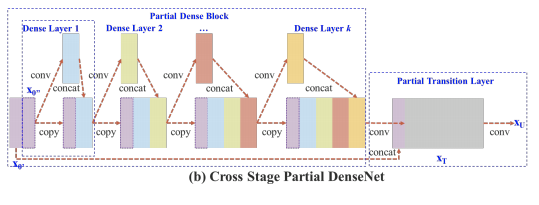


Figure 0.6 DenseNet

Source: [32]

Google Brain created EfficientNet particularly to research the scaling issue with convolutional neural networks. While scaling up the ConvNet, you have a variety of options, including input size, width scaling, depth scaling, and scaling all of the aforementioned options. According to the EfficientNet study, all of these have an optimal value, which they locate through search.

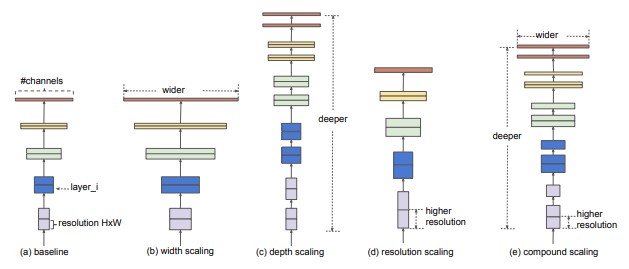


Figure 0.7 EfficientNet

Source: [33]

EfficientNet has better performance compared to other similar networks, however, the authors of YOLO decided to try out other networks as well to ensure that which works best in the case of object detection. After a lot of experimentation, evaluation and comparison of results, finally it was decided that YOLOv5 will have **CSPDarknet53** as its backbone.

A picture containing text, receipt

Description automatically generated

**Fig: Darknet53**

|  |
| --- |
| # YOLOv5 🚀 by Ultralytics, GPL-3.0 license  # Parameters  nc: 80 # number of classes  depth\_multiple: 0.33 # model depth multiple  width\_multiple: 0.50 # layer channel multiple  anchors:  - [10,13, 16,30, 33,23] # P3/8  - [30,61, 62,45, 59,119] # P4/16  - [116,90, 156,198, 373,326] # P5/32  # YOLOv5 v6.0 backbone  backbone:  # [from, number, module, args]  [[-1, 1, Conv, [64, 6, 2, 2]], # 0-P1/2  [-1, 1, Conv, [128, 3, 2]], # 1-P2/4  [-1, 3, C3, [128]],  [-1, 1, Conv, [256, 3, 2]], # 3-P3/8  [-1, 6, C3, [256]],  [-1, 1, Conv, [512, 3, 2]], # 5-P4/16  [-1, 9, C3, [512]],  [-1, 1, Conv, [1024, 3, 2]], # 7-P5/32  [-1, 3, C3, [1024]],  [-1, 1, SPPF, [1024, 5]], # 9  ]  # YOLOv5 v6.0 head  head:  [[-1, 1, Conv, [512, 1, 1]],  [-1, 1, nn.Upsample, [None, 2, 'nearest']],  [[-1, 6], 1, Concat, [1]], # cat backbone P4  [-1, 3, C3, [512, False]], # 13  [-1, 1, Conv, [256, 1, 1]],  [-1, 1, nn.Upsample, [None, 2, 'nearest']],  [[-1, 4], 1, Concat, [1]], # cat backbone P3  [-1, 3, C3, [256, False]], # 17 (P3/8-small)  [-1, 1, Conv, [256, 3, 2]],  [[-1, 14], 1, Concat, [1]], # cat head P4  [-1, 3, C3, [512, False]], # 20 (P4/16-medium)  [-1, 1, Conv, [512, 3, 2]],  [[-1, 10], 1, Concat, [1]], # cat head P5  [-1, 3, C3, [1024, False]], # 23 (P5/32-large)  [[17, 20, 23], 1, Detect, [nc, anchors]], # Detect(P3, P4, P5)  ] |

Table 1 Yolov5 Layers

We can see the difference of YOLOv5 backbone with Darknet-53. YOLOv5 backbone and head have slowly evolved over many releases and contain C3 modules which are reduced CSP modules. The Focus layer introduced in the beginning appears to be equivalent to a simple 2d-convolutional layer without the need for the space-to-depth operation. For example, a Focus layer with kernel size 3 can be expressed as a Conv layer with kernel size 6 and stride 2.

The primary purpose of Model Neck is to produce feature pyramids. Pyramids of features enable models to scale objects successfully in general. The ability to recognize the same thing in various sizes and scales is helpful. Models that use feature pyramids perform well on unobserved data. Other models, such as FPN, BiFPN, and PANet, employ various other feature pyramid methodologies.[34] [35] [36]

PANet architecture is used for generating feature pyramids as neck in YOLOv5. [36]

## 2.13 UNDERSTANDING FEATURE PYRAMID NETWORKS FOR OBJECT DETECTION (FPN)

The final detecting step is generally carried out using the model Head. It's using anchor boxes on the features and produced final output vectors that included bounding boxes, objectness scores, and class probabilities. The head of the YOLOv5 model is the same as that of the earlier generations, V3 and V4.

### 2.13.1 ACTIVATION FUNCTION

Any deep neural network's selection of activation functions is extremely important. Many activation functions, such Leaky ReLU, mish, and swish, have recently been introduced. The Leaky ReLU and Sigmoid activation function was chosen by the authors of YOLO v5. mish swish

In YOLO v5, the final detection layer uses the sigmoid activation function whereas the middle/hidden layers use the Leaky ReLU activation function.

### 2.13.2 OPTIMIZATION FUNCTION

There are two options given by YOLOv5 for optimization.

* Stochastic Gradient Descent [37]
* Adam [38]

By default, SGD is used as optimization function. However, it can be changed to Adam by using the command –adam as the commandline argument.

### 2.13.3 COST FUNCTION OR LOSS FUNCTION

The Loss function is evaluated as a compound loss calculated using the objectness score, bounding box regression score and class probability score. Also, another choice of using Focal Loss function is given.

### ****2.13.4 HYPERPARAMETERS IN YOLOV5****

**The kind of parameters whose values govern a deep learning model's entire learning process are called hyperparameters. While not a part of a deep learning model, it also impacts the values of model parameters that a learning algorithm ultimately learns.**

**About thirty (30) different hyperparameters are employed in YOLO-v5 depending on the training environment. We can optimize the results if we wish to get better ones. By optimizing the parameters competitive results can be obtained based on the target of optimization. default values from training on the COCO dataset can be used if optimization is not a major concern.**

Important hyper parameters explored during the research are given below:

* **Learning-rate start (lr0):** It defines the step size for each iteration which means how fast the network learns from training.
* **Learning-rate end (lr1):** It checks if the current learning rate is greater than or equal to the value of this parameter then it ends training.
* **Momentum: This parameter is used for the Stochastic Gradient Descent optimization. This parameter helps to replace the gradient with an aggregate of gradient.**
* **Mosaic: This parameter helps to create a new image from multiple input images and thus the new image which has been created is used for training. This parameter is useful in the case of data augmentation and optimal feature techniques.**
* **Degree: This parameter rotates input images by certain angles and uses it for training so that the accuracy is improved.**
* **Scaling: The images are resized for optimization as well as matching it with the grid size.**
* **Flipud: This parameter helps to flip the images from the input dataset randomly up and down so as to get better results. Again, this also falls under data augmentation technique.**
* **weight-decay: This is the penalty term impacting the cost function during back propagation which helps to shrink/compress the weight.**

**Although there are many other hyper parameters that can be tuned for better results like warmup epochs, warmup momentum this research focuses mainly on the above said parameters alone.**

### 2.13.6 TYPES OF YOLOv5

Diagram

Description automatically generated with low confidence

Figure 0.8 YOLOv5 Models

* **YOLOv5n**

With 75% fewer parameters (from 7.5M to 1.9M), the nano model maintains the YOLOv5s depth multiple of 0.33 but reduces the YOLOv5s width multiple from 0.50 to 0.25, making it suited for mobile and CPU solutions.

* **YOLOv5s**

YOLOv5s has 224 layers and 7.2 million trainable parameters and is as fast as 2 milliseconds (FLOPs or floating-point operations around 17 billion in number) with an mAP of 36.7. The predictions are saved in ‘runs/exp’ directory.

* **YOLOv5m**

YOLOv5m has 308 layers, 21 million parameters, a mean average precision of 44.5, and an average speed of inference of 2.7ms(FLOPs value at 51.3 billion).

* **YOLOv5l**

The YOLOv5l version has 47 million parameters and 392 layers. It recorded an mAP of 48.2 percent with an inference speed of 3.8ms on average(since FLOPs value is 115.4 billion).

* **YOLOv5x**

This is the biggest model referred to as the YOLOv5x and it has 476 layers and 87 million parameters along with a FLOPs value of 218.6 Billion in value. It recorded an inference speed of 6.1ms and an mAP of 50.4 percent.

# CHAPTER 3: METHODOLOGY

## 3.1 CONVOLUTION NEURAL NETWORK USED

Using the YOLOv5 model, a vehicle safety system with an effective obstacle detection mechanism and increased speed is created. The video of the driver's vision is captured by a camera and sent to the model for precise and rapid object recognition on urban roadways. Each frame of the supplied video is processed as an input to the object recognition and detection algorithm (YOLOv5). In the algorithm, each frame is processed in three stages: backbone, neck, and head.

(i)Backbone: CSPDarknet

(ii)Neck: PANet with spatial pyramid pooling (SPP)

(iii) Head: YOLO layer

Different features of Driver Assistance Systems and Autonomous vehicles are the subject of numerous studies. Using a laser scanner, Chen et al.[39] developed an IoT-based occlusion technique termed multiple targets tracking in occlusion region with interactive object models in urban contexts to solve the challenge of object recognition for autonomous vehicles. The difficulty in identifying the object resulted from the various observed shapes on each laser scan. Consequently, the proposed system is created using a machine learning approach and YOLOv5 to mitigate the occlusion problem.

Yolov5 will be trained with different annotated data sets for road surface detection and road signs and used for predictions and generating warnings. During this research, the performance of Yolov5 is evaluated by gathering various metrics and compared.

### 3.1.1 YOLOv5

The algorithms mentioned in the previous sections have the drawback that they consume too much time for pre-processing. In the case of R-CNN, it defines regions, and then the regions are sent for classification. YOLOv5 (You only look once) on the contrary inputs an image split into SxS default regions and all the regions are processed at the same time. The regions with a confidence level above a certain threshold will be candidates to identify objects within them.

### 3.1.2 RELEASES

YOLO was first released in May 2016 by Joseph Redmon. In 2017, a better version named YOLO 9000 was released. The most popular and stable version named YOLOv3 was released in 2018 with the paper “YOLOv3: An Incremental Improvement”.[40] In April 2020, Alexey Bochkoviskiy introduced YOLOv4: Optimal Speed and Accuracy of Object Detection. YOLOv4 outperformed YOLOv3 by a high margin.

Chart, line chart

Description automatically generated

Figure 0.1 YOLO Performance Comparison

On 9th June 2020, Glenn Jocher, an unofficial author, released YOLOv5 based on PyTorch with exceptional improvements. YOLOv5 is so far the best compared to the previous versions.

Chart

Description automatically generated

Figure 0.2 YOLO Models Performance

Calendar

Description automatically generated

Figure 0.3 Performance Comparison Metrics

**Source:** [41]

## 3.2 CUSTOM OBJECT DETECTION WITH YOLOV5

Training YOLOv5 can be defined in the following steps:

1. Prepare the Dataset
2. Setup the Environment
3. Configuring the parameters in the model
4. Train
5. Evaluate

### 3.2.1 PREPARE THE DATASET

Initially, MS COCO dataset was used to do the validations after the environment was set up. The documentation for YOLOv5 gives pretrained checkpoints given in the above figure based on validation against MS COCO val2017 [42]dataset. Therefore, the initial runs were executed to validate the environment with MS COCO dataset. Since YOLOv5 has different models namely, YOLOv5x, YOLOv5n6, YOLOv5m6, etc each of those models was validated with the MS COCO Dataset. YOLO documentation gives results by training each model to 300 epochs.

Since our intent is to predict threats and hazards during driving, we have categorized two types of data that can be used to train YOLOv5. The first one is RDD2020 [11]which can be used for training YOLO on road conditions. The dataset contains 26,336 road images collected from India, Japan, and the Czech Republic with more than 31000 instances of road damage. There are four classes mainly longitudinal cracks D00, Transverse Cracks D10, Alligator Cracks D20, and Potholes.

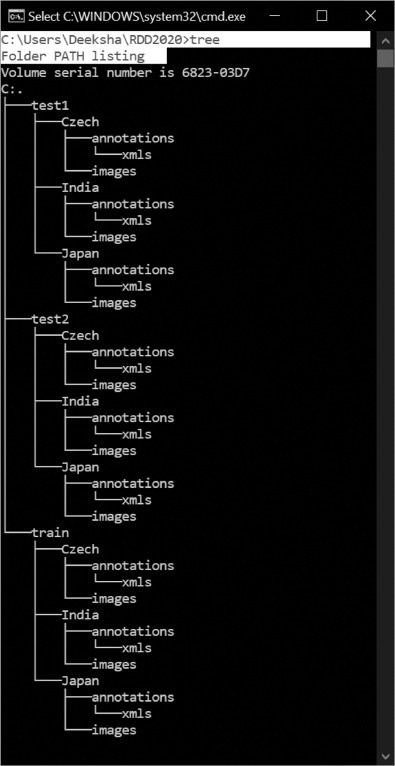
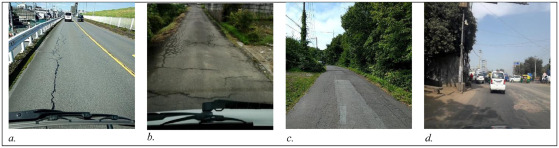


Figure 0.4 RDD Dataset Folder Structure

Source: [11]



Sample images for road damage categories considered in the data. a. Longitudinal Crack (D00) b. Transverse Crack (D10) c. Alligator Crack(D20) d. Pothole(D40).

The Second category dataset is a small one called LISA Dataset [43]. This dataset contains four image classes. It contains only 877 images and is a very small one but suits the purpose of this research.

1. Traffic Light
2. Stop
3. Speed Limit
4. Crosswalk

Both the datasets had to be converted into a YOLO-supported format to ensure the training and detection process. The online tool named Roboflow was used to annotate and generate augmented images to train both categories, Road Surface, and Sign Board warnings. The generated annotated dataset contains more than 12k images which is comparatively very less quantity but sufficient for the purpose of this research.

### 3.2.2 SETUP THE ENVIRONMENT

The environment consists of:

1. Server hosted by Paperspace.com datacentre in California [8Core CPU, 30 GB Memory, 8 GB GPU, Storage 100GB]
2. PyTorch
3. Jupyter Notebook
4. Wandb.ai [For tracking the metrics during training and testing]

### 3.2.3 YOLO ARCHITECTURE AND WORKING

Yolov5 architecture consists of three parts:

1. Backbone: CSPDarknet
2. Neck: PANet
3. Head: Yolo Layer

The data is initially fed into the CSPDarknet for feature extraction. The extracted features are then fed into PANet for feature fusion. The last stage Yolo layer detects yields the results (class, score, location, size)

Diagram

Description automatically generated

Figure 0.5 YOLO Architecture

Source: [44]

### ****3.2.4 HYPER PARAMETER EVOLUTION OF YOLOV5****

**Hyper parameter evolution is the method of parameter optimization by using Genetic algorithm for optimization.** Various aspects of training are controlled by hyperparameters in machine learning and determining their ideal values can be difficult. Grid searches and other conventional techniques can easily become unworkable because of 1) the high dimensional search space 2) unknown relationships across the dimensions, 2) costly nature of assessing the fitness for each point. GA is a good choice for hyperparameter searches due to the above-mentioned concerns.

**INITIALIZE HYPERPARAMETERS**

The hyperparameters of YOLOv5 are defined in a yaml file in /data directory. There are around 30 such parameters which can be tuned to improve the performance of the neural network. The better the initial guesses of the parameters are, the more better final values will be. Therefore, it is very crucial that the parameters are properly set before evolving. However, the evolving process also could be started with the optimized value obtained from COCO training from scratch.

**DEFINE FITNESS**

**YOLOv5 has defined the fitness function as a combination of the following:**

* mAP@0.5 contributes 10% of the weight
* mAP@0.5:0.95 contributes the remaining 90%

It can be adjusted as per requirement as it is the value which is maximized during the training.

**EVOLVE**

Parameters are evolved by starting from a base scenario and then fine tuning is done. For example, to fine tune on COCO128 dataset for 10 epochs using pretrained YOLOv5s, the command is as follows:

|  |
| --- |
| python train.py --epochs 10 --data coco128.yaml --weights yolov5s.pt –cache --evolve |

By default, the number of iterations is 300. A minimum of 300 generations is recommended as it is required for best results. Evolution is an expensive and time taking process requiring a lot of GPU hours for execution.

|  |
| --- |
| # Hyperparameter Evolution Results  # Generations: 1000  # P R mAP.5 mAP.5:.95 box obj cls  # Metrics: 0.4761 0.79 0.763 0.4951 0.01926 0.03286 0.003559  lr0: 0.01 # initial learning rate (SGD=1E-2, Adam=1E-3)  lrf: 0.2 # final OneCycleLR learning rate (lr0 \* lrf)  momentum: 0.937 # SGD momentum/Adam beta1  weight\_decay: 0.0005 # optimizer weight decay 5e-4  warmup\_epochs: 3.0 # warmup epochs (fractions ok)  warmup\_momentum: 0.8 # warmup initial momentum  warmup\_bias\_lr: 0.1 # warmup initial bias lr  box: 0.05 # box loss gain  cls: 0.5 # cls loss gain  cls\_pw: 1.0 # cls BCELoss positive\_weight  obj: 1.0 # obj loss gain (scale with pixels)  obj\_pw: 1.0 # obj BCELoss positive\_weight  iou\_t: 0.20 # IoU training threshold  anchor\_t: 4.0 # anchor-multiple threshold  anchors: 0 # anchors per output grid (0 to ignore)  fl\_gamma: 0.0 # focal loss gamma (efficientDet default gamma=1.5)  hsv\_h: 0.015 # image HSV-Hue augmentation (fraction)  hsv\_s: 0.7 # image HSV-Saturation augmentation (fraction)  hsv\_v: 0.4 # image HSV-Value augmentation (fraction)  degrees: 0.0 # image rotation (+/- deg)  translate: 0.1 # image translation (+/- fraction)  scale: 0.5 # image scale (+/- gain)  shear: 0.0 # image shear (+/- deg)  perspective: 0.0 # image perspective (+/- fraction), range 0-0.001  flipud: 0.0 # image flip up-down (probability)  fliplr: 0.5 # image flip left-right (probability)  mosaic: 1.0 # image mosaic (probability)  mixup: 0.0 # image mixup (probability) |

**VISUALIZE**

One plot is stored for each hyperparameter in the results file, yolov5/evolve.png. The x axis shows values, and the y axis shows fitness. Yellow denotes concentrations that are higher. A parameter has been disabled and is not changing when there are vertical lines next to it. This can be used to fix parameters and stop them from evolving and is user selectable in the meta dictionary in train.py.

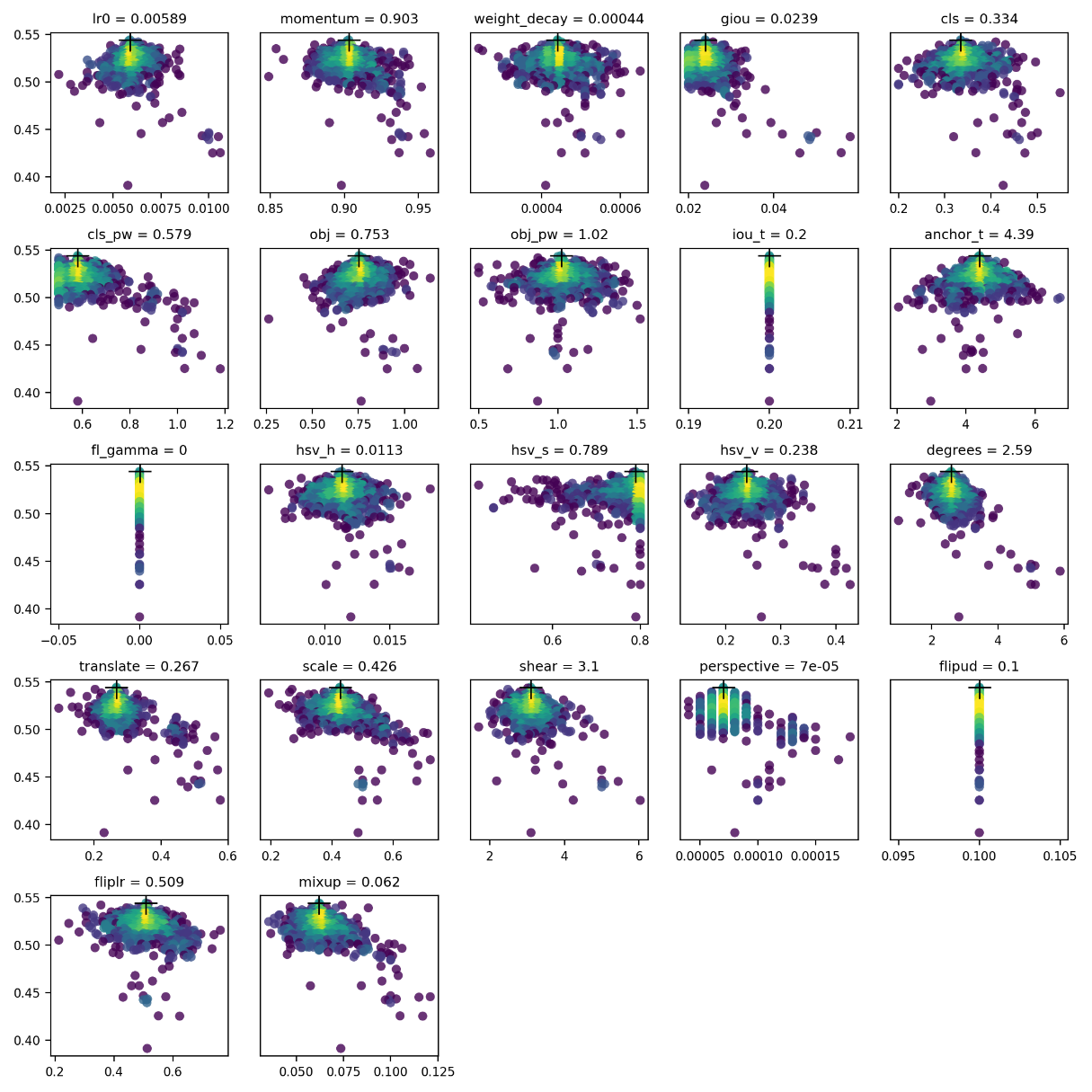


Figure 0.6 Yolo Evolution

## 3.3 PRUNE AND SPARSE YOLOv5

Modification of weights by reducing the weights parameters, that does not influence on classification of classes is known as "pruning." Yolov5 employs a pruning strategy that "converts to zeros randomly in some proportion of weights parameters, nn.conv2d layers whose weights are virtually zero". Yolov5 must first be trained on the custom data, after which it must validate your trained model using validation data before invoking the prune function to allow the trained model to be pruned. Custom-trained weight values can be specified in best.pt file. Then the following code must be specified in val.py file.

|  |
| --- |
| from utils.torch\_utils import prune prune(model,0.3) |

|  |
| --- |
| # Directories  save\_dir = increment\_path(Path(project) / name, exist\_ok=exist\_ok) # increment run  (save\_dir / 'labels' if save\_txt else save\_dir).mkdir(parents=True, exist\_ok=True) # make dir  # Load model  model = DetectMultiBackend(weights, device=device, dnn=dnn, data=data, fp16=half)  stride, pt, jit, engine = model.stride, model.pt, model.jit, model.engine  imgsz = check\_img\_size(imgsz, s=stride) # check image size  half = model.fp16 # FP16 supported on limited backends with CUDA  if engine:  batch\_size = model.batch\_size  else:  device = model.device  if not (pt or jit):  batch\_size = 1 # export.py models default to batch-size 1  LOGGER.info(f'Forcing --batch-size 1 square inference (1,3,{imgsz},{imgsz}) for non-PyTorch models')  # Data  data = check\_dataset(data) # check  **#pruning**  **from utils.torch\_utils import prune**  **prune(model,0.3)**  # Configure  model.eval()  cuda = device.type != 'cpu'  is\_coco = isinstance(data.get('val'), str) and data['val'].endswith(f'coco{os.sep}val2017.txt') # COCO dataset  nc = 1 if single\_cls else int(data['nc']) # number of classes  iouv = torch.linspace(0.5, 0.95, 10, device=device) # iou vector for mAP@0.5:0.95  niou = iouv.numel()  # Dataloader  if not training:  if pt and not single\_cls: # check --weights are trained on --data  ncm = model.model.nc  assert ncm == nc, f'{weights[0]} ({ncm} classes) trained on different --data than what you passed ({nc} ' \  f'classes). Pass correct combination of --weights and --data that are trained together.'  model.warmup(imgsz=(1 if pt else batch\_size, 3, imgsz, imgsz)) # warmup  pad = 0.0 if task in ('speed', 'benchmark') else 0.5 |

Table 2 val.py

Once, the code is pasted into the val.py file then the following code can be executed to prune the model.

|  |
| --- |
| python3 val.py --data "coco.yaml or your custom data.yaml" --weights " custom file.pt" |

# 

# CHAPTER 4: RESULTS AND DISCUSSIONS

The training and performance evaluation were done as three sections. Initially baseline performance of YOLOv5 was validated by training and testing on MS-COCO128 dataset. In the second part, YOLOv5 performance on LISA dataset for road signboard detection and recognition was performed. In the third part, YOLOv5 was trained on road defects using RDD dataset and evaluated. The details of the results are discussed below.

## 4.1 TRAINING ON MS-COCO TO GENERATE BASELINE METRICS

The baseline results for YOLOv5s model using MS COCO128 dataset for 7 epochs were found to be as given below. With just 7 epochs itself YOLOv5s attained decent levels of precision.

Chart, line chart

Description automatically generated

Figure 0.1 Performance Metrics for MS-COCO

|  |
| --- |
| **hyperparameters:** lr0=0.01, lrf=0.01, momentum=0.937, weight\_decay=0.0005, warmup\_epochs=3.0, warmup\_momentum=0.8, warmup\_bias\_lr=0.1, box=0.05, cls=0.5, cls\_pw=1.0, obj=1.0, obj\_pw=1.0, iou\_t=0.2, anchor\_t=4.0, fl\_gamma=0.0, hsv\_h=0.015, hsv\_s=0.7, hsv\_v=0.4, degrees=0.0, translate=0.1, scale=0.5, shear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5, mosaic=1.0, mixup=0.0, copy\_paste=0.0 |

Table 3 Hyperparameters

|  |
| --- |
| python train.py --img 640 --batch 16 --epochs 7 --data coco128.yaml --weights yolov5s.pt --cache |

**x**

Chart, line chart

Description automatically generated

Table 4 Confusion Matric for MS-COCO

From the above given results, it can be seen that YOLOv5s performs well with the large number of classes of the dataset and with decent number of epochs for the coco dataset containing different categories of objects including people.

## 4.2 TRAINING ON ROAD SIGNS

The LISA road-sign dataset although had only around 800 images, it was augmented using roboflow and additional images generated. It was also converted into Yolo compatible format. The developed dataset was fed into the network, and it quickly got trained to yield around 80% precision as you can see from the below given chart. There are four classes into which the objects are classified which are traffic like, stop sign, speed limit, cross walk. We could add as many classes as required to possibly distinguish between the type of warnings that can be generated when it is detected. However, for the research purpose it is only required to identify the signboards.

**Graphical user interface, chart

Description automatically generated**

Figure 0.2 Performance Metric for Road Signs

**Chart, waterfall chart

Description automatically generated**

Figure 0.3 Confusion matrix for LISA Road Signs

**Chart, line chart

Description automatically generated**

Figure 0.4 F1 Curve

**Chart

Description automatically generated with medium confidence**

Figure 0.5 Precision-Recall Curve

## 4.3 TRAINING ON ROAD DEFECTS

The road defects detection using YOLOv5 was not an ideal case as per the results. The images were augmented, and additional images generated with transforms. This along with the captured image quality affected the training. Some of the images annotated decently is shown as a mosaic the picture given below. From the picture it appears that YOLO is able to identify the cracks and its types. However, the precision converges to below just 6% with the default hyper parameters.

**A picture containing text, different, various, multiple

Description automatically generated**

Figure 0.6 Mosaic for Road Defect Detection

Graphical user interface, chart, line chart

Description automatically generated

Figure 0.7 mAP 0.5:0.95 for RDD

Chart, line chart

Description automatically generated

Figure 0.8 mAP 0.5

Chart, line chart

Description automatically generated

Figure 0.9 Recall

Further, it was explored to optimize the neural network by tweaking the parameters like batch size, image resolution etc. However, this change in the execution broke the integration with the Pytorch library and threw error which was confirmed to be from a Pytorch library bug at the Ultralytics forum: <https://github.com/pytorch/pytorch/issues/74016>

|  |
| --- |
| AttributeError: 'NoneType' object has no attribute '\_free\_weak\_ref' · Issue #74016 · pytorch/pytorch |

The error was attempted to be fixed by running the following:

|  |
| --- |
| pip install torch==1.10.1+cu111 torchvision==0.11.2+cu111 torchaudio==0.10.1 -f <https://download.pytorch.org/whl/torch_stable.html> |

However, this didn’t fix the issue and further trials couldn’t be done since it had to be fixed in pytorch library developers and would take time to fix it.

# CHAPTER 5: CONCLUSION AND FUTURESCOPE

This paper put forward a YOLOv5-based approach to road defect detection and road signboard detection and recognition to enable driver assistance systems installed on vehicles. Two categories of Datasets explored the possibility of detecting road hazards with the timely identification of the road surface defects as well as the sign board warnings.

Due to model's limited resolution and access to context data, detecting small objects in photos was difficult. Especially if processing is to take place on the same device that takes the pictures, many modern systems that handle object identification need to do so at real-time speeds, creating specific requirements in computational resources. It is true for many autonomous vehicle systems, where the car takes pictures and analyses them in real time, frequently to guide what it will do next. Smaller objects in this context are those that can be detected at a greater distance from the car, allowing for early detection of such things and essentially extending the detection range of the car. If this one area were to be improved, the system would be better informed and able to make more solid choices. YOLOv5 needs better optimization in this context as it was found out from this study for detecting road defects. The road signs were detected with minimal dataset and minimal training. However, the road surface detection posed a challenge to the YOLO algorithm at least with the default hyperparameters. Although YOLOv5 is a powerful tool, it is not optimized to detect smaller items because it is intended to be a general-purpose object detector.

The Microsoft Common Objects in Context (COCO) API's metrics at three distinct object scales (bounding box areas) and Intersection over Unions (IOU) are compatible with the original implementation of YOLOv5, which is advantageous for the objectives of this study. While the calculation of values at particular scales can provide us with a decent indicator of the model's performance, it may not be correct for other datasets.

In fewer than 50 epochs, the yolov5s model's traffic sign detection attained 80% accuracy. Road defect detection using Yolov's model with standard hyperparameter values, however, did not produce sufficient results. The size of the input image, the quantity of epochs, the size of the batch, the learning rate, the momentum, and the activation function are all tuneable parameters that can enhance YOLO performance.

Finally, dashboard mounted cameras can be used to feed Yolov5 trained with the Road defects and signboard trained data to give alerts and warnings before it is too late so that it enhances the safety during as well as can assist autonomous vehicles to avoid damages to vehicle and enable situation aware driving. This can be further enhanced by adding sufficient data to predict various other situations like pedestrian safety, weather changes, unusual driving conditions such as oil spill or accident situations, fire hazards, lane changes etc. The method also enables quantifying and mapping quality of the road depending upon the frequency of the defects or warnings triggered or objects detected so as to enable drivers to choose the best route to their destinations.

Ultralytics has been working on YOLOv7 which has better capabilities to detect smaller objects and road defects compared to earlier versions. YOLOv7 outperforms all the other known object detectors in terms of both speed and accuracy in the range from 5 FPS to 160 FPS and has the greatest accuracy 56.8% AP among all known real-time object detection networks with 30 FPS or above[45, p. 7] on GPU V100. YOLOv7-E6 object detector (56 FPS V100, 55.9% AP) outperforms both transformer-based detector SWINL Cascade-Mask R-CNN (9.2 FPS A100, 53.9% AP) by 509% in speed and 2% in accuracy, and convolutional based detector ConvNeXt-XL Cascade-Mask R-CNN (8.6 FPS A100, 55.2% AP) by 551% in speed and 0.7% AP in accuracy, as well as YOLOv7 outperforms: YOLOR, YOLOX, Scaled-YOLOv4, YOLOv5, DETR, Deformable DETR, DINO-5scale-R50, ViT-Adapter-B and many other object detectors in speed and accuracy.

An experiment has been done by opencv [45]to perform multiscale object detection for road defects recently using yolov7. The experiments were run on GTX 1060 6GB machine. Tiny model was trained and used to evaluate performance on a driving video. This resulted in variable confidence levels for the same class for different instances. However, the same was repeated using multi resolution model to see if the performance would vary on the higher model. The results have remained the same as the previous experiment.

The same experiments were performed with YOLOv7 large model, and it started to look much better than when it was evaluated with the tiny model. The same was even better with multi resolution trained model with the large Yolov7 model an it was performing even better by detecting potholes from farther distances. There were still failure cases.

In comparison with other neural networks, Yolov7 is itself the best performing model to detect road defects. The dataset contained 1265 training images, 401 validation images and 118 test images.Also when the images were found to be greater that 1500 pixels, they were down sampled by 0.3. Thus, a normalized resolution was being used during the experiment. Although the experiments were run on GTX 1060 6GB machine, the trainings were done on 16 GB Tesla P100 GPU.

The layers of large model of YOLOv7 are given as the following:

|  |
| --- |
| %%writefile cfg/training/yolov7\_pothole.yaml  # parameters  nc: 1 # number of classes  depth\_multiple: 1.0 # model depth multiple  width\_multiple: 1.0 # layer channel multiple  # anchors  anchors:  - [12,16, 19,36, 40,28] # P3/8  - [36,75, 76,55, 72,146] # P4/16  - [142,110, 192,243, 459,401] # P5/32  # yolov7 backbone  backbone:  # [from, number, module, args]  [[-1, 1, Conv, [32, 3, 1]], # 0  [-1, 1, Conv, [64, 3, 2]], # 1-P1/2  [-1, 1, Conv, [64, 3, 1]],  [-1, 1, Conv, [128, 3, 2]], # 3-P2/4  [-1, 1, Conv, [64, 1, 1]],  [-2, 1, Conv, [64, 1, 1]],  [-1, 1, Conv, [64, 3, 1]],  [-1, 1, Conv, [64, 3, 1]],  [-1, 1, Conv, [64, 3, 1]],  [-1, 1, Conv, [64, 3, 1]],  [[-1, -3, -5, -6], 1, Concat, [1]],  [-1, 1, Conv, [256, 1, 1]], # 11  [-1, 1, MP, []],  [-1, 1, Conv, [128, 1, 1]],  [-3, 1, Conv, [128, 1, 1]],  [-1, 1, Conv, [128, 3, 2]],  [[-1, -3], 1, Concat, [1]], # 16-P3/8  [-1, 1, Conv, [128, 1, 1]],  [-2, 1, Conv, [128, 1, 1]],  [-1, 1, Conv, [128, 3, 1]],  [-1, 1, Conv, [128, 3, 1]],  [-1, 1, Conv, [128, 3, 1]],  [-1, 1, Conv, [128, 3, 1]],  [[-1, -3, -5, -6], 1, Concat, [1]],  [-1, 1, Conv, [512, 1, 1]], # 24  [-1, 1, MP, []],  [-1, 1, Conv, [256, 1, 1]],  [-3, 1, Conv, [256, 1, 1]],  [-1, 1, Conv, [256, 3, 2]],  [[-1, -3], 1, Concat, [1]], # 29-P4/16  [-1, 1, Conv, [256, 1, 1]],  [-2, 1, Conv, [256, 1, 1]],  [-1, 1, Conv, [256, 3, 1]],  [-1, 1, Conv, [256, 3, 1]],  [-1, 1, Conv, [256, 3, 1]],  [-1, 1, Conv, [256, 3, 1]],  [[-1, -3, -5, -6], 1, Concat, [1]],  [-1, 1, Conv, [1024, 1, 1]], # 37  [-1, 1, MP, []],  [-1, 1, Conv, [512, 1, 1]],  [-3, 1, Conv, [512, 1, 1]],  [-1, 1, Conv, [512, 3, 2]],  [[-1, -3], 1, Concat, [1]], # 42-P5/32  [-1, 1, Conv, [256, 1, 1]],  [-2, 1, Conv, [256, 1, 1]],  [-1, 1, Conv, [256, 3, 1]],  [-1, 1, Conv, [256, 3, 1]],  [-1, 1, Conv, [256, 3, 1]],  [-1, 1, Conv, [256, 3, 1]],  [[-1, -3, -5, -6], 1, Concat, [1]],  [-1, 1, Conv, [1024, 1, 1]], # 50  ]  # yolov7 head  head:  [[-1, 1, SPPCSPC, [512]], # 51  [-1, 1, Conv, [256, 1, 1]],  [-1, 1, nn.Upsample, [None, 2, 'nearest']],  [37, 1, Conv, [256, 1, 1]], # route backbone P4  [[-1, -2], 1, Concat, [1]],  [-1, 1, Conv, [256, 1, 1]],  [-2, 1, Conv, [256, 1, 1]],  [-1, 1, Conv, [128, 3, 1]],  [-1, 1, Conv, [128, 3, 1]],  [-1, 1, Conv, [128, 3, 1]],  [-1, 1, Conv, [128, 3, 1]],  [[-1, -2, -3, -4, -5, -6], 1, Concat, [1]],  [-1, 1, Conv, [256, 1, 1]], # 63  [-1, 1, Conv, [128, 1, 1]],  [-1, 1, nn.Upsample, [None, 2, 'nearest']],  [24, 1, Conv, [128, 1, 1]], # route backbone P3  [[-1, -2], 1, Concat, [1]],  [-1, 1, Conv, [128, 1, 1]],  [-2, 1, Conv, [128, 1, 1]],  [-1, 1, Conv, [64, 3, 1]],  [-1, 1, Conv, [64, 3, 1]],  [-1, 1, Conv, [64, 3, 1]],  [-1, 1, Conv, [64, 3, 1]],  [[-1, -2, -3, -4, -5, -6], 1, Concat, [1]],  [-1, 1, Conv, [128, 1, 1]], # 75  [-1, 1, MP, []],  [-1, 1, Conv, [128, 1, 1]],  [-3, 1, Conv, [128, 1, 1]],  [-1, 1, Conv, [128, 3, 2]],  [[-1, -3, 63], 1, Concat, [1]],  [-1, 1, Conv, [256, 1, 1]],  [-2, 1, Conv, [256, 1, 1]],  [-1, 1, Conv, [128, 3, 1]],  [-1, 1, Conv, [128, 3, 1]],  [-1, 1, Conv, [128, 3, 1]],  [-1, 1, Conv, [128, 3, 1]],  [[-1, -2, -3, -4, -5, -6], 1, Concat, [1]],  [-1, 1, Conv, [256, 1, 1]], # 88  [-1, 1, MP, []],  [-1, 1, Conv, [256, 1, 1]],  [-3, 1, Conv, [256, 1, 1]],  [-1, 1, Conv, [256, 3, 2]],  [[-1, -3, 51], 1, Concat, [1]],  [-1, 1, Conv, [512, 1, 1]],  [-2, 1, Conv, [512, 1, 1]],  [-1, 1, Conv, [256, 3, 1]],  [-1, 1, Conv, [256, 3, 1]],  [-1, 1, Conv, [256, 3, 1]],  [-1, 1, Conv, [256, 3, 1]],  [[-1, -2, -3, -4, -5, -6], 1, Concat, [1]],  [-1, 1, Conv, [512, 1, 1]], # 101  [75, 1, RepConv, [256, 3, 1]],  [88, 1, RepConv, [512, 3, 1]],  [101, 1, RepConv, [1024, 3, 1]],  [[102,103,104], 1, IDetect, [nc, anchors]], # Detect(P3, P4, P5)  ] |

From the above it is evident that detecting potholes is a challenging problem and needs deeper neural network and as well as better quality of training dataset. It also needs higher computation resources and is a target for the future releases of YOLO.

## ACKNOWLEDGMENT

I would like to thank Prof Dr. Raja Hashim Ali, Prof Dr. Talha Ali Khan for the support in guiding me and reviewing paper contents from time to time. Their valuable suggestions have helped me in completing this work.

I would also like to thank my family who have supported me morally and financially in pursuing the long-cherished dream of learning Machine learning and Data Science which I strongly believe will pave the foundation for a successful contribution for the betterment of the society.

# REFERENCES

[1] L. García-Menéndez, I. Martínez-Zarzoso, and D. Miguel, “Determinants of Mode Choice between Road and Shipping for Freight Transport: Evidence for Four Spanish Exporting Sectors,” *J. Transp. Econ. Policy*, vol. 38, pp. 447–466, Sep. 2004.

[2] “RoadDebris\_FACTSHEET.pdf.” Accessed: Aug. 15, 2022. [Online]. Available: https://aaafoundation.org/wp-content/uploads/2017/12/RoadDebris\_FACTSHEET.pdf

[3] P. Felzenszwalb, D. McAllester, and D. Ramanan, “A discriminatively trained, multiscale, deformable part model,” in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, AK, USA, Jun. 2008, pp. 1–8. doi: 10.1109/CVPR.2008.4587597.

[4] F. Liang, D. Wang, Y. Liu, Y. Jiang, and S. Tang, “Fast Pedestrian Detection Based on Sliding Window Filtering,” in *Advances in Multimedia Information Processing – PCM 2012*, Berlin, Heidelberg, 2012, pp. 811–822. doi: 10.1007/978-3-642-34778-8\_76.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection.” arXiv, May 09, 2016. doi: 10.48550/arXiv.1506.02640.

[6] J. Dai, Y. Li, K. He, and J. Sun, “R-FCN: Object Detection via Region-based Fully Convolutional Networks.” arXiv, Jun. 21, 2016. doi: 10.48550/arXiv.1605.06409.

[7] A. Dosovitskiy *et al.*, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.” arXiv, Jun. 03, 2021. doi: 10.48550/arXiv.2010.11929.

[8] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[9] C. Liu, S. Li, F. Chang, and Y. Wang, “Machine Vision Based Traffic Sign Detection Methods: Review, Analyses and Perspectives,” *IEEE Access*, vol. 7, pp. 86578–86596, 2019, doi: 10.1109/ACCESS.2019.2924947.

[10] K. Lu, “Advances in deep learning methods for pavement surface crack detection and identification with visible light visual images.” arXiv, Dec. 02, 2021. doi: 10.48550/arXiv.2012.14704.

[11] D. Arya, H. Maeda, S. K. Ghosh, D. Toshniwal, and Y. Sekimoto, “RDD2020: An annotated image dataset for automatic road damage detection using deep learning,” *Data Brief*, vol. 36, p. 107133, Jun. 2021, doi: 10.1016/j.dib.2021.107133.

[12] X. Zhang, X. Xia, N. Li, M. Lin, J. Song, and N. Ding, “Exploring the Tricks for Road Damage Detection with A One-Stage Detector,” Dec. 2020, pp. 5616–5621. doi: 10.1109/BigData50022.2020.9377923.

[13] R. Girshick, “Fast R-CNN.” arXiv, Sep. 27, 2015. doi: 10.48550/arXiv.1504.08083.

[14] A. Nevriyanto and D. Purnamasari, “Performance Analysis of Comparison between Region Growing, Adaptive Threshold and Watershed Methods for Image Segmentation,” *Hong Kong*, p. 7, 2018.

[15] Y. Liu, M.-M. Cheng, X. Hu, K. Wang, and X. Bai, “Richer Convolutional Features for Edge Detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 8, pp. 1939–1946, Aug. 2019, doi: 10.1109/TPAMI.2018.2878849.

[16] L. Sekanina and J. Torresen, “Detection Of Norwegian Speed Limit Signs,” Aug. 2002.

[17] F. Moutarde, A. Bargeton, A. Herbin, and L. Chanussot, “Robust on-vehicle real-time visual detection of American and European speed limit signs, with a modular Traffic Signs Recognition system,” Jul. 2007, pp. 1122–1126. doi: 10.1109/IVS.2007.4290268.

[18] C. G. Keller, C. Sprunk, C. Bahlmann, J. Giebel, and G. Baratoff, “Real-time recognition of U.S. speed signs,” in *2008 IEEE Intelligent Vehicles Symposium*, Eindhoven, Netherlands, Jun. 2008, pp. 518–523. doi: 10.1109/IVS.2008.4621282.

[19] Z. Malik and I. Siddiqi, “Detection and Recognition of Traffic Signs from Road Scene Images,” Dec. 2014, pp. 330–335. doi: 10.1109/FIT.2014.68.

[20] K. Deb, A. Vavilin, J.-W. Kim, and K.-H. Jo, “Vehicle License Plate Tilt Correction Based on the Straight Line Fitting Method and Minimizing Variance of Coordinates of Projection Points,” *Int. J. Control Autom. Syst.*, vol. 8, pp. 975–984, Oct. 2010, doi: 10.1007/s12555-010-0506-z.

[21] L. Wang, “Support Vector Machines: Theory and Applications,” *Stud. Fuzziness Soft Comput. V 177*, vol. 302, Jan. 2005.

[22] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, “Detection of Traffic Signs in Real-World Images: The German Traffic Sign Detection Benchmark,” presented at the Proceedings of the International Joint Conference on Neural Networks, Aug. 2013. doi: 10.1109/IJCNN.2013.6706807.

[23] Q. Hou, M.-M. Cheng, X.-W. Hu, A. Borji, Z. Tu, and P. Torr, “Deeply supervised salient object detection with short connections,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 4, pp. 815–828, Apr. 2019, doi: 10.1109/TPAMI.2018.2815688.

[24] G. Wang, G. Ren, Z. Wu, Y. Zhao, and L. Jiang, “A robust, coarse-to-fine traffic sign detection method,” Aug. 2013, pp. 1–5. doi: 10.1109/IJCNN.2013.6706812.

[25] M. Mathias, R. Timofte, R. Benenson, and L. Van Gool, “Traffic sign recognition — How far are we from the solution?,” Aug. 2013, pp. 1–8. doi: 10.1109/IJCNN.2013.6707049.

[26] S. H.S.G and C. M. Patil, “An approach towards efficient detection and recognition of traffic signs in videos using neural networks,” in *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, Mar. 2016, pp. 456–459. doi: 10.1109/WiSPNET.2016.7566175.

[27] T. Zhang, J. Zou, and W. Jia, “Fast and robust road sign detection in driver assistance systems,” *Appl. Intell.*, vol. 48, no. 11, pp. 4113–4127, Nov. 2018, doi: 10.1007/s10489-018-1199-x.

[28] A. Mammeri, A. Boukerche, J. Feng, and R. Wang, “North-American speed limit sign detection and recognition for smart cars,” in *38th Annual IEEE Conference on Local Computer Networks - Workshops*, Oct. 2013, pp. 154–161. doi: 10.1109/LCNW.2013.6758513.

[29] R. Fan and M. Liu, “Road Damage Detection Based on Unsupervised Disparity Map Segmentation,” *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 11, pp. 4906–4911, Nov. 2020, doi: 10.1109/TITS.2019.2947206.

[30] D. Jeong, “Road Damage Detection Using YOLO with Smartphone Images,” in *2020 IEEE International Conference on Big Data (Big Data)*, Dec. 2020, pp. 5559–5562. doi: 10.1109/BigData50022.2020.9377847.

[31] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely Connected Convolutional Networks.” arXiv, Jan. 28, 2018. Accessed: Aug. 16, 2022. [Online]. Available: http://arxiv.org/abs/1608.06993

[32] C.-Y. Wang, H.-Y. M. Liao, I.-H. Yeh, Y.-H. Wu, P.-Y. Chen, and J.-W. Hsieh, “CSPNet: A New Backbone that can Enhance Learning Capability of CNN.” arXiv, Nov. 26, 2019. Accessed: Aug. 16, 2022. [Online]. Available: http://arxiv.org/abs/1911.11929

[33] M. Tan and Q. V. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks.” arXiv, Sep. 11, 2020. Accessed: Aug. 16, 2022. [Online]. Available: http://arxiv.org/abs/1905.11946

[34] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal Loss for Dense Object Detection.” arXiv, Feb. 07, 2018. doi: 10.48550/arXiv.1708.02002.

[35] M. Tan, R. Pang, and Q. V. Le, “EfficientDet: Scalable and Efficient Object Detection.” arXiv, Jul. 27, 2020. doi: 10.48550/arXiv.1911.09070.

[36] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, “Path Aggregation Network for Instance Segmentation.” arXiv, Sep. 18, 2018. doi: 10.48550/arXiv.1803.01534.

[37] S. Ruder, “An overview of gradient descent optimization algorithms.” arXiv, Jun. 15, 2017. Accessed: Aug. 16, 2022. [Online]. Available: http://arxiv.org/abs/1609.04747

[38] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization.” arXiv, Jan. 29, 2017. doi: 10.48550/arXiv.1412.6980.

[39] J.-F. Chen, C.-C. Wang, and C.-F. Chou, “Multiple target tracking in occlusion area with interacting object models in urban environments,” *Robot. Auton. Syst.*, vol. 103, Feb. 2018, doi: 10.1016/j.robot.2018.02.004.

[40] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement.” arXiv, Apr. 08, 2018. Accessed: Aug. 16, 2022. [Online]. Available: http://arxiv.org/abs/1804.02767

[41] “ultralytics/yolov5.” Ultralytics, Aug. 16, 2022. Accessed: Aug. 16, 2022. [Online]. Available: https://github.com/ultralytics/yolov5

[42] “COCO - Common Objects in Context.” https://cocodataset.org/#home (accessed Aug. 16, 2022).

[43] “Road Signs | MakeML - create neural network with ease.” https://makeml.app/datasets/road-signs (accessed Aug. 16, 2022).

[44] R. Xu, H. Lin, K. Lu, L. Cao, and Y. Liu, “A Forest Fire Detection System Based on Ensemble Learning,” *Forests*, vol. 12, no. 2, p. 217, Feb. 2021, doi: 10.3390/f12020217.

[45] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, “YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors.” arXiv, Jul. 06, 2022. Accessed: Aug. 25, 2022. [Online]. Available: http://arxiv.org/abs/2207.02696