Contents

List of Figures

List of Tables

Part I

Background

Chapter 1

Introduction

The world is progressing fast enough on technology in the last years and computer science is not an area that falls behind. Computer science can be divided in many areas of study, but we could resume them in three big fields. Software development, information technology and security. We will be focusing on information technology such as artificial intelligence, machine learning and deep learning.

In computer science, Artificial Intelligence (AI) is the intelligence carried out by machines. Its goal is to build smart machines capable of performing tasks that normally are made by humans. AI is just software; it is an application written to do a task. For example, on Facebook with suggestions of names of people to tag on pictures, for fraud detection on credit cards or for self- driving cars that can manage their shelves to drive through the city.

Artificial intelligence can be divided into two categories:

- Weak AI: Sometimes referred to as "Narrow AI", is the type of AI that only learns to do one thing, It is going to be better than a human doing just a single thing such as predicting the temperature in your house to be warm but if you ask to detect a cat in a picture it will be terrible.

- Strong AI: Also known as General artificial intelligence (AGI) is created to learn to do anything. It is supposed to recreate the intelligence of a human, what does this mean? By this intelligence It could be possible to have a machine doing the same things as humans like thinking or being able to take decisions. Summarizing, it is an intelligence that can do all the tasks possible of doing by a human. That is why it has not been invented yet.

Today there is weak AI everywhere, Siri, facial recognition, Instagram filters, Google adds recommendations or Amazon recommendations and more examples that are taken for granted. Siri is a software capable of recognizing the human voice and is great at doing very few things. Therefore, it is not a strong AI, it is composed by a few Weak AI. This type of intelligence is better than a human at doing a task. For example, focusing on working out if a credit

card is good or a fraud, the software will be more efficient than a human just because it has processed a wide dataset of fraudulent or good cards. Nevertheless, the human will guide by its experience or knowledge.

Here is where Machine learning (ML) takes part, it is the ability of a machine to learn and predict by itself. It learns patterns from the data to make the prediction more accurate. These algorithms can make relevant conclusions from the dataset by creating a mathematical function that best fits the data. The main objective of an apprentice (learner) is to develop the ability to generalize and associate. When we translate this into a machine or computer, it means that they should be able to perform accurately. One of the machine learning fields that still has a long way to go is deep learning.

Deep learning (DL) is a subset of machine learning that is formed of layers recreating the human brain. It had a breakthrough many years ago, but it came to a standstill for some years and in the last few years has had a rebound. In terms of deep learning, the structure is called artificial neural network. DL plays with parameters, it trains a network with these parameters to learn on its own. By the way, after the network has been trained it will be able to recognize patterns in the input data to make the detection accurate.

In deep learning there is a very large and important study area that has gained a lot of strength in recent years, object detection in real time. It is mainly used in video vigilance cameras of highways or for autonomous cars cameras. It consists in the capacity of a machine to detect objects in a set of images in a very small processing time. The network's capacity of generalization must be very good because skipping the detection of an object in an image is a sign of loss of accuracy. This final degree project will be focusing on detecting vehicles (cars, trucks, motorcycles) in videos and then tracking those vehicles to be able to count them. Video processing must be done in record time to be in real-time.

1.1 References

Chapter 2

Introduction

A vehicle (from Latin: vehiculum[1]) is a machine that transports people or cargo. Vehicles include wagons, bicycles, motor vehicles (motorcycles, cars, trucks, buses), railed vehicles (trains, trams), watercraft (ships, boats), amphibious vehicles (screw-propelled vehicle, hovercraft), aircraft (airplanes, helicopters, aerostat) and spacecraft[2]. Land vehicles are classified broadly by what is used to apply steering and drive forces against the ground: wheeled, tracked, railed or skied. ISO 3833-1977 is the standard, also internationally used in legislation, for road vehicles types, terms and definitions. Vehicle detection and vehicle type recognition is a practical application of machine learning concepts and is directly applicable for various operations in a traffic surveillance system contributing to an intelligent traffic surveillance system. We will introduce the processing of automatic vehicle detection and recognition using static image datasets. Further using the same technique, we shall improvise vehicle detection by using live CCTV surveillance. The surveillance system includes detection of moving vehicles and recognizing them, counting number of vehicles and verification of their permit with the organization[3].

Intelligent vehicle detection and counting are becoming increasingly important in the field of highway management. However, due to the different sizes of vehicles, their detection remains a challenge that directly affects the accuracy of vehicle counts. To address this issue, this paper proposes a vision-based vehicle detection and counting system. In the proposed vehicle detection and counting system, the highway road surface in the image is first extracted and divided into a remote area; the method is crucial for improving vehicle detection. Then, the vehicle trajectories are obtained by the ORB algorithm. Finally, the above two areas are placed into the YOLOv5 network to detect the type and location of the vehicle. Several highway surveillance videos based on different scenes are used to verify the proposed methods. The experimental results verify that using the proposed segmentation method can provide higher detection accuracy, especially for the detection of small vehicle object[4].

Vehicle detection and statistics in highway monitoring video scenes are of considerable significance to intelligent traffic management and control of the

highway. With the popular installation of traffic surveillance cameras, a vast database of traffic video footage has been obtained for analysis. Generally, at a high viewing angle, a more-distant road surface can be considered. The object size of the vehicle changes greatly at this viewing angle, and the detection accuracy of a small object far away from the road is low. In the face of complex camera scenes, it is essential to effectively solve the above problems and further apply them [4].

A CCTV camera is a very essential part of an intelligent traffic surveillance framework. It is simply the automated process of monitoring the traffic in a particular area and detecting vehicles for further action, as shown in the diagram. The captured images can provide valuable clues to the cops and other public essential tracking services, such as vehicle's license plate number, time and motion of vehicle, details associated with the driver, etc.. which all may lead to evidence of some crime or any unforeseen or unfortunate incidents. Earlier people used to process images manually. In fact, this system is still going on in India, whereas countries like the USA also have implemented automated machines- CCTVs that function 24x7 and take immediate action via signaling too. Manual work has always been proven slower and less efficient due to human errors and many other factors that affect living beings. Keeping these points in mind and moving with the advancement of technologies, many innovative thinkers have developed certain intelligent traffic control systems using various techniques. This research is based upon the combination of two prior-made researches by scholars whose works have been published [5].

2.1 References

[1] Merriam Webster. Definition of vehicle (2021). URL https://www.merriam-webster.com/dictionary/vehicle. Cited on page/s 5.

[2] Wikipedia. Vehicle (2021). URL https://en.wikipedia.org/wiki/Vehicle. Cited on page/s 5.

[3] Sriashika Addala. Research paper on vehicle detection and recognition. (05 2020). doi: 10.13140/RG.2.2.34908.82561. Cited on page/s 5.

[4] Liang H. Li H. et al. Song, H. Vision-based vehicle detection and counting system using deep learning in highway scenes. (2019). doi: https://doi.org/10.1186/s12544-019-0390-4. Cited on page/s 5, 6.

[5] Rusc T. & Fornalski P. Baran, R. A smart camera for the surveillance of vehicles in intelligent transportation systems. (2016). doi: https://doi.org/10.1007/s11042-015-3151-y. Cited on page/s 6.

Chapter 3

Related Works

At present, vision-based vehicle object detection is divided into traditional machine vision methods and complex deep learning methods. Traditional machine vision methods use the motion of a vehicle to separate it from a fixed background image. This method can be divided into three categories: the method of using background subtraction, the method of using continuous video frame difference, and the method of using optical flow. Using the video frame difference method, the variance is calculated according to the pixel values of two or three consecutive video frames. Moreover, the moving foreground region is separated by the threshold. By using this method and suppressing noise, the stopping of the vehicle can also be detected. When the background image in the video is fixed, the background information is used to establish the background model. Then, each frame image is compared with the background model, and the moving object can also be segmented. The method of using optical flow can detect the motion region in the video. The generated optical flow field represents each pixel's direction of motion and pixel speed. Vehicle detection methods using vehicle features, such as the Scale Invariant Feature Transform (SIFT) and Speeded Up Robust Features (SURF) methods, have been widely used. For example, 3D models have been used to complete vehicle detection and classification tasks. Using the correlation curves of 3D ridges on the outer surface of the vehicle, the vehicles are divided into three categories: cars, SUVs, and minibuses. The use of deep convolutional networks (CNNs) has achieved amazing success in the field of vehicle object detection. CNNs have a strong ability to learn image features and can perform multiple related tasks, such as classification and bounding box regression. The detection method can be generally divided into two categories. The two-stage method generates a candidate box of the object via various algorithms and then classifies the object by a convolutional neural network. The one-stage method does not generate a candidate box but directly converts the positioning problem of the object bounding box into a regression problem for processing. In the two-stage method, Region-CNN (R-CNN) uses selective region search in the image. The image input to the convolutional network must be fixed-size, and the deeper structure of the network requires a long training time and con-

sumes a large amount of storage memory. Drawing on the idea of spatial pyramid matching, SPP NET allows the network to input images of various sizes and to have fixed outputs. R-FCN, FPN, and Mask RCNN have improved the feature extraction methods, feature selection, and classification capabilities of convolutional networks in different ways. Among the one-stage methods, the most important are the Single Shot Multibox Detector (SSD) and You Only Look Once (YOLO) frameworks. The MutiBox, Region Proposal Network (RPN) and multi-scale representation methods are used in SSD, which uses a default set of anchor boxes with different aspect ratios to more accurately position the object. Unlike SSD, the YOLO network divides the image into a fixed number of grids. Each grid is responsible for predicting objects whose centre points are within the grid. YOLOv2 added the BN (Batch Normalization) layer, which makes the network normalize the input of each layer and accelerate the network convergence speed. YOLOv2 uses a multi-scale training method to randomly select a new image size for every ten batches. Our vehicle object detection uses the YOLOv3 network. Based on YOLOv2, YOLOv3 uses logistic regression for the object category. The category loss method is two-class cross-entropy loss, which can handle multiple label problems for the same object. Moreover, logistic regression is used to regress the box confidence to determine if the IOU of the a priori box and the actual box is greater than 0.5. If more than one priority box satisfies the condition, only the largest prior box of the IOU is taken. In the final object prediction, YOLOv3 uses three different scales to predict the object in the image. The traditional machine vision method has a faster speed when detecting the vehicle but does not produce a good result when the image changes in brightness, there is periodic motion in the background, and where there are slow moving vehicles or complex scenes. Advanced CNN has achieved good results in object detection; however, CNN is sensitive to scale changes in object detection. The one stage method uses grids to predict objects, and the grid's spatial constraints make it impossible to have higher precision with the two-stage approach, especially for small objects. The two stage method uses region of interest pooling to segment candidate regions into blocks according to given parameters, and if the candidate region is smaller than the size of the given parameters, the candidate region is padded to the size of the given parameters. In this way, the characteristic structure of a small object is destroyed and its detection accuracy is low. The existing methods do not distinguish if large and small objects belong to the same category. The same method is used to deal with the same type of object, which will also lead to inaccurate detection. The use of image pyramids or multi-scale input images can solve the above problems, although the calculation requirements are large.

Detection of specific objects in images is difficult, due to the nature of ob-

jects in images that are often of different sizes, different orientations, and overlapping objects that causes occlusion of the object of interest to be detected. These problems require a detection algorithm that has several properties, such as translation invariance(invariant to different locations of object of interest in the image), rotation invariance (invariant to the rotation of the object in the image), and scale invariance (invariant to the size of the objects in the image). A common approach is to use machine learning methods that learn a representation directly from the available data to train a model. Popular Methods use low level features such as SIFT [12], HOG [3], and Haar [17] combining them with a machine learning method to classify the objects. This approach is known as the "Feature +Classifier" approach.

3.1 Vehicle detection research in Europe

Vision-based vehicle detection methods in Europe have achieved abundant results. In, between the "Hofolding" and "Weyern" sections of the A8 motorway in Munich, Germany, the Multivariate Alteration Detection (MAD) method was used to detect the change of two images with a short time lag. The moving vehicles are highlighted in a change image, which is used to estimate the vehicle density of the road. Using the motorways A95 and A96 near Munich, the A4 near Dresden, and the "Mittlere Ring" in Munich as the test environments, the Canny edge algorithm is applied to the road image, and the histogram of the edge steepness is calculated. Then, using the k-means algorithm, the edge steepness statistics are divided into three parts, and a closed vehicle model is detected based on the steepness. A contrast-based approach was used to create a colour model to identify and remove vehicle shadow areas, which eliminates interference caused by movement in the scene. After eliminating the shadow area, the vehicle detection performance can be significantly improved. The experiment was conducted on Italian and French highways. The HOG and Haar-like features were compared in, and the two features were merged to construct a detector for vehicle detection that was tested on French vehicle images. However, when the above method is used for vehicle detection, the type of vehicle cannot be detected. Additionally, when the illumination is insufficient, it is difficult to extract the edge of the vehicle or detect the moving car, which causes problems in low vehicle detection accuracy and affects the detection results for further use. Pictures of aerial view angles were used by but cannot clearly capture the characteristics of each car and produce false vehicle detections. Nonetheless, with the development of deep learning technology, vehicle detection based on CNN has been successfully applied in Europe. In Fast R-CNN was used for vehicle detection in traffic scenes in the city of Karlsruhe,

Germany. Fast R-CNN uses a selective search strategy to find all candidate frames, which is notably time-consuming, and the vehicle detection speed is slow. In short, research on vision-based vehicle detection is still progressing, and major challenges are gradually being overcome, which will make a significant contribution to the development of European traffic construction.

Chapter 4

Object Detection With YOLO

YOLO (You Only Look Once) is a state-of-art algorithm devoted to object detection, as the name implies it can predict objects just by looking once to the image in a clever way. YOLO makes the prediction by classifying the object and locating it in the image. It uses deep learning and CNN techniques to detect objects, and distinguishes itself from its competitors because, as its name indicates, it requires to see the image only once, allowing it to be the fastest of all although it sacrifices a little accuracy. This speed allows users to easily detect objects in real time in videos (up to 30 FPS). In other words, YOLO takes as input value an image and passes through the neural network that looks like a CNN and returns a vector of bounding boxes and class prediction. To understand better how YOLO makes the prediction we will explain it with an example of an image as it can be seen in Figure Figure 4.1.

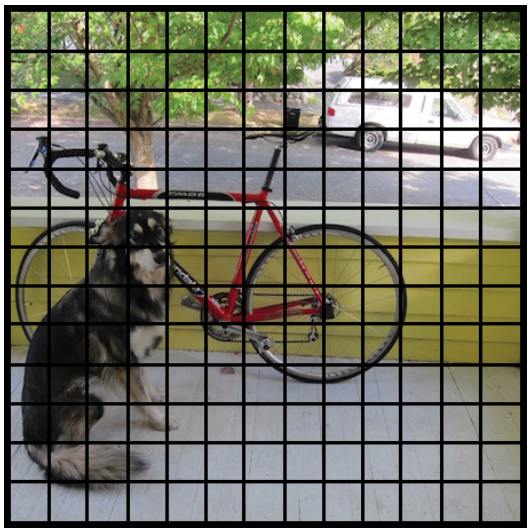

FIGURE 4.1. YOLO divides up the image into a grid of 13 by 13 cells: Each of these cells is responsible for predicting 5 bounding boxes. A bounding box describes the rectangle that encloses an object.

4.1 YOLO in brief

To perform de detection YOLO first divides the image into an S x S grid of cells, in Figure Figure 4.1 we can see that the image is divided into a 13 x 13 grid of cells. In each of the cells it predicts N possible bounding boxes and calculates the level of certainty (probability) of each of them (shown in Figure 13), that is, S x S x N different bounding boxes are calculated in the image, the vast majority of them with a very low level of certainty. The score of each BB does not classify what type of object is or it does not know what object is fitting the bounding box, it just gives a confidence value or a probability value of how good the BB is surrounding an object in each cell. In case of Figure Figure 4.1 for each grid cells 5 bounding boxes are predicted and are graphically shown in Figure 4.2 where the higher the probability is the fatter the box is. Each cell makes N bounding box predictions and M class probabilities.

FIGURE 4.2. The predicted bounding boxes may look something like this, the higher the confidence score, the fatter the box is drawn

The bounding box prediction is formed by 5 components: x, y, width, height, confidence score.

- The (x, y) components are like a mathematical graph coordinates that refer to the centre of the box, relative to the grid cell location. If the centre of the bounding box does not fall inside the grid cell this means that the grid cell is not responsible of the object that has predicted the bounding box. This happens because each object that appears in the

image is related to a single grid cell that is responsible for predicting the object. (x, y) are normalized between 0 and 1.

- (Width, height) represent the dimension of the box that contains an object, they are also normalized to values from 0 to 1 and they are fundamental to locate the object in the image.
- Confidence score is a real value that represents the assurance the algorithm has that the box contains an object of any class. The method how the confidence is calculated will be explained later.

4.2 The Predictions Vector

The first step to understanding YOLO is how it encodes its output. The input image is divided into an S x S grid of cells. For each object that is present on the image, one grid cell is said to be "responsible" for predicting it. That is the cell where the center of the object falls into.

Each grid cell predicts B bounding boxes as well as C class probabilities. The bounding box prediction has 5 components: (x, y, w, h, confidence). The (x, y) coordinates represent the center of the box, relative to the grid cell location (remember that, if the center of the box does not fall inside the grid cell, than this cell is not responsible for it). These coordinates are normalized to fall between 0 and 1. The (w, h) box dimensions are also normalized to [0, 1], relative to the image size.

Chapter 5

Methodology

The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1 ×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224 ×224 input image) and then double the resolution for detection.

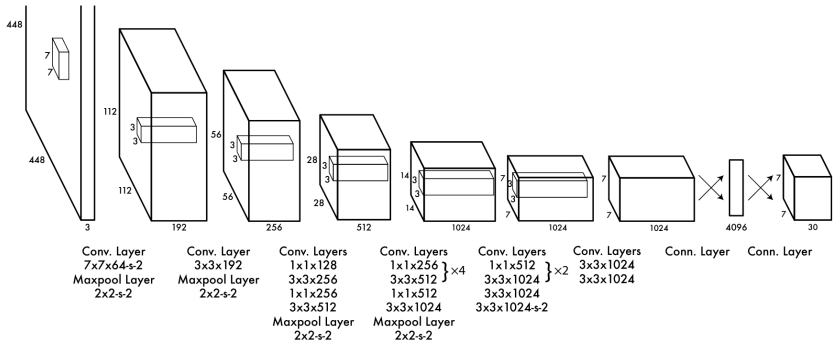

FIGURE 5.1. The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

5.1 The Network

The network structure looks like a normal CNN, with convolutional and max pooling layers, followed by 2 fully connected layers in the end:

Note that the architecture was crafted for use in the Pascal VOC dataset, where the authors used $S = 7$, $B = 2$ and $C = 20$. This explains why the final feature maps are 7×7, and also explains the size of the output ($7x7x(2 \times 5 + 20)$). Use of this network with a different grid size or different number of classes might require tuning of the layer dimensions. The authors mention

TABLE 5.1. YOLO Network Structure

Name	Filters	Output Dimension
Conv 1	7 x 7 x 64, stride=2	224 x 224 x 64
Max Pool 1	2 x 2, stride=2	112 x 112 x 64
Conv 2	3 x 3 x 192	112 x 112 x 192
Max Pool 2	2 x 2, stride=2	56 x 56 x 192
Conv 3	1 x 1 x 128	56 x 56 x 128
Conv 4	3 x 3 x 256	56 x 56 x 256
Conv 5	1 x 1 x 256	56 x 56 x 256
Conv 6	1 x 1 x 512	56 x 56 x 512
Max Pool 3	2 x 2, stride=2	28 x 28 x 512
Conv 7	1 x 1 x 256	28 x 28 x 256
Conv 8	3 x 3 x 512	28 x 28 x 512
Conv 9	1 x 1 x 256	28 x 28 x 256
Conv 10	3 x 3 x 512	28 x 28 x 512
Conv 11	1 x 1 x 256	28 x 28 x 256
Conv 12	3 x 3 x 512	28 x 28 x 512
Conv 13	1 x 1 x 256	28 x 28 x 256
Conv 14	3 x 3 x 512	28 x 28 x 512
Conv 15	1 x 1 x 512	28 x 28 x 512
Conv 16	3 x 3 x 1024	28 x 28 x 1024
Max Pool 4	2 x 2, stride=2	14 x 14 x 1024
Conv 17	1 x 1 x 512	14 x 14 x 512
Conv 18	3 x 3 x 1024	14 x 14 x 1024
Conv 19	1 x 1 x 512	14 x 14 x 512
Conv 20	3 x 3 x 1024	14 x 14 x 1024
Conv 21	3 x 3 x 1024	14 x 14 x 1024
Conv 22	3 x 3 x 1024, stride=2	7 x 7 x 1024
Conv 23	3 x 3 x 1024	7 x 7 x 1024
Conv 24	3 x 3 x 1024	7 x 7 x 1024
FC 1	-	4096
FC 2	-	7 x 7 x 30 (1470)

that there is a fast version of YOLO, with fewer convolutional layers. The table Table 9.1, however, display the full version. The sequences of 1x1 reduction layers and 3x3 convolutional layers were inspired by the GoogLeNet (Inception) model. The final layer uses a linear activation function. All other layers use a leaky RELU ($\phi(x) = x, if x > 0; 0.1x\ otherwise$).

5.2 The Loss Function

The loss function start like this:

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \tag{5.1}$$

This equation computes the loss related to the predicted bounding box position (x, y). Don't worry about λ for now, just consider it a given constant. The function computes a sum over each bounding box predictor ($j = 0..B$) of each grid cell ($i = 0..S^2$). $\mathbb{1}obj$ is defined as follows:

- 1, If an object is present in grid cell i and the jth bounding box predictor is "responsible" for that prediction
- 0, otherwise

But how do we know which predictor is responsible for the object? Quoting the original paper:

> YOLO predicts multiple bounding boxes per grid cell. At training time we only want one bounding box predictor to be responsible for each object. We assign one predictor to be "responsible" for predicting an object based on which prediction has the highest current IOU with the ground truth.

The other terms in the equation should be easy to understand: (x, y) are the predicted bounding box position and $(\hat{x}, \hat{y})$ hat are the actual position from the training data.

Let's move on to the second part:

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \tag{5.2}$$

This is the loss related to the predicted box width / height. The equation looks similar to the first one, except for the square root. Quoting the paper again:

> Our error metric should reflect that small deviations in large boxes matter less than in small boxes. To partially address this we predict the square root of the bounding box width and height instead of the width and height directly.

Moving on to the third part:

$$\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{obj}(C_i-\hat{C}_i)^2+\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{noobj}(C_i-\hat{C}_i)^2 \tag{5.3}$$

Here we compute the loss associated with the confidence score for each bounding box predictor. C is the confidence score and $\hat{C}$ is the intersection over union of the predicted bounding box with the ground truth. $\mathbb{1}obj$ is equal to one when there is an object in the cell, and 0 otherwise. $\mathbb{1}noobj$ is the opposite.

The λ parameters that appear here and also in the first part are used to differently weight parts of the loss functions. This is necessary to increase model stability. The highest penalty is for coordinate predictions ($\lambda coord = 5$) and the lowest for confidence predictions when no object is present ($\lambda noobj = 0.5$).

The last part of the loss function is the classification loss:

$$\sum_{i=0}^{S^2}\mathbb{1}_{i}^{obj}\sum_{c\in classes}(p_i(c)-\hat{p}_i(c))^2 \tag{5.4}$$

It looks similar to a normal sum-squared error for classification, except for the $\mathbb{1}obj$ term. This term is used because so we don't penalize classification error when no object is present on the cell (hence the conditional class probability discussed earlier).

Note that the loss function only penalizes classification error if an object is present in that grid cell (hence the con-ditional class probability discussed earlier). It also only pe-nalizes bounding box coordinate error if that predictor is "responsible" for the ground truth box (i.e. has the highest IOU of any predictor in that grid cell).**?**

Chapter 6

Dataset

Our new benchmark has 100000 cropped images after discarding some of the images only containing background. Of these, 10000 contain 30000 traffic signs in total. Although our source images cover much of Bangladesh and India, an imbalance still exists between different classes of traffic sign in our benchmark. This is unavoidable: classes such as signs. It can be from different viewpoints, have different ground sampling distances (gsd), different image sizes, aspect ratios, color, etc. Vehicles in different data may be significantly different in size and appearance. Figure **??** shows different images from four different datasets. It is obvious how different the VOC data is from any of the elevated data, while the VEDAI and AFVID data are somewhat similar, and the AF Build-ing Camera data is more similar to the aerial and satellite data. A more detailed look at the aerial datasets is available in TableNumber of classesIn the configuration file that defines the YOLO net model there is a 'classes' definition that is used to define the number of classes. To change the number of classes used, more than just the classes setting will need to be changed; the number of filters in the last convolutional layer must be altered to reflect the changed number of classes. The number of filters is set by $num(classes + coords + 1)$, where num isnumber of anchor boxes, and coords is four corresponding to the four coordinates used to define a bounding box.

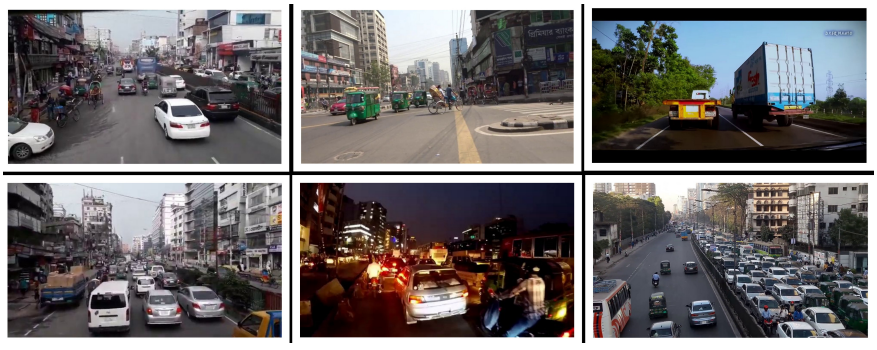

FIGURE 6.1. Dataset Collage. Some sample data from Dhaka AI dataset.

6.1 COCO Dataset Details

- **Images per class** ≥ 1500 images per class recommended

- **Instances per class** ≥ 10000 instances (labeled objects) per class recommended

- **Image variety** Must be representative of deployed environment. For real-world use cases we recommend images from different times of day, different seasons, different weather, different lighting, different angles, different sources (scraped online, collected locally, different cameras) etc.

- **Label consistency** All instances of all classes in all images must be labelled. Partial labelling will not work.

- textbfLabel accuracy Labels must closely enclose each object. No space should exist between an object and it's bounding box. No objects should be missing a label.

- **Background images** Background images are images with no objects that are added to a dataset to reduce False Positives (FP). We recommend about 0-10% background images to help reduce FPs (COCO has 1000 background images for reference, 1% of the total). No labels are required for background images.

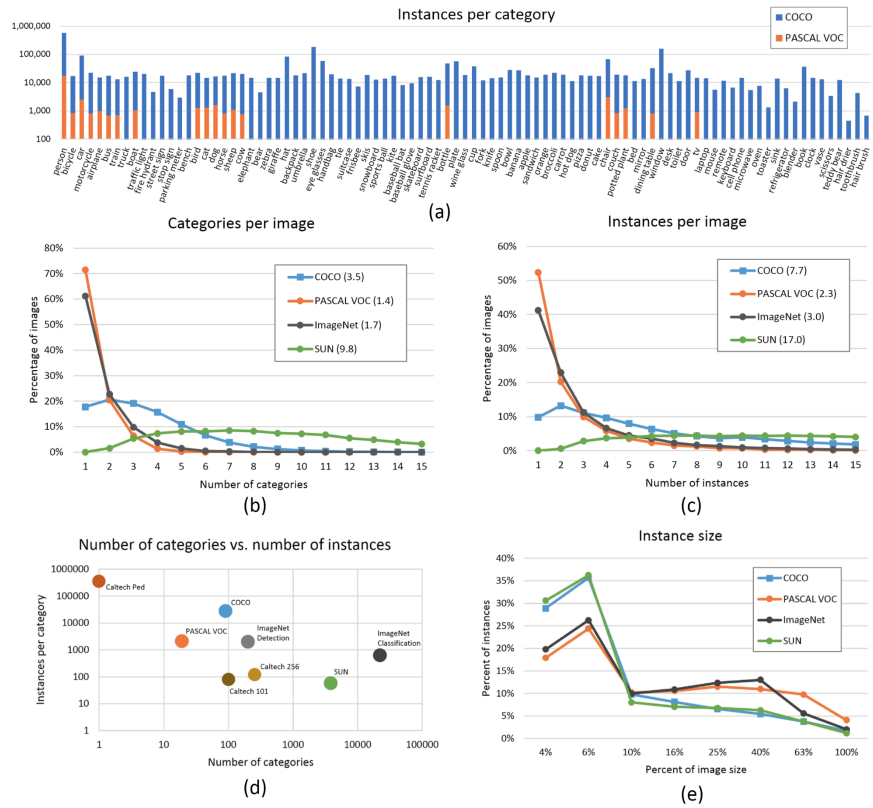

FIGURE 6.2. Dataset Collage. Some sample data from Dhaka AI dataset.

6.2 Dhaka AI Dataset

We have used a mixed dataset from Indian Driving Dataset and Dhaka AI Traffic Detection Challenge Dataset.

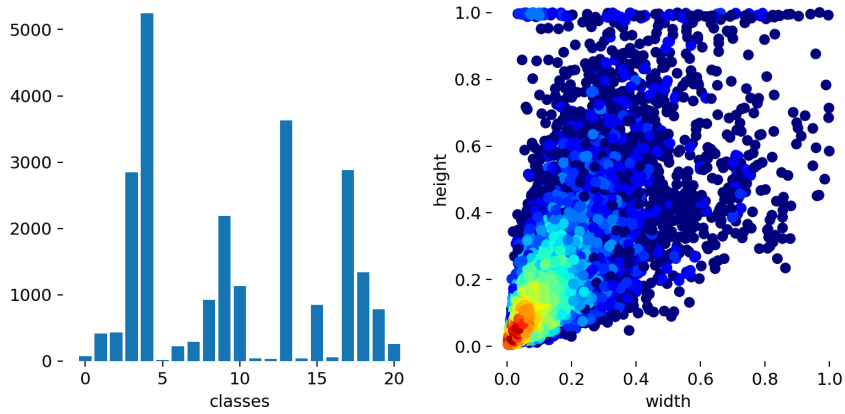

FIGURE 6.3. Dataset Collage. Some sample data from Dhaka AI dataset.

6.2.1 List of classes

- Ambulance
- Auto Rickshaw
- Bicycle
- Bus
- Car
- Garbage Van
- Human Hauler
- Minibus
- Minivan
- Motorbike
- Pickup
- Army Vehicle
- Police Car
- Rickshaw
- Scooter
- SUV
- Taxi
- Three Wheelers (CNG)
- Truck
- Van
- Wheelbarrow

6.3 Training Settings

Before modifying anything, first train with default settings to establish a performance baseline. A full list of train.py settings can be found in the train.py

argparser.

- Epochs. Start with 300 epochs. If this overfits early then you can reduce epochs. If overfitting does not occur after 300 epochs, train longer, i.e. 600, 1200 etc epochs.

- Image size. COCO trains at native resolution of `--img 640`, though due to the high amount of small objects in the dataset it can benefit from training at higher resolutions such as `--img 1280`. If there are many small objects then custom datasets will benefit from training at native or higher resolution. Best inference results are obtained at the same `--img` as the training was run at, i.e. if you train at `--img 1280` you should also test and detect at `--img 1280`.

- Batch size. Use the largest `--batch-size` that your hardware allows for. Small batch sizes produce poor batchnorm statistics and should be avoided.

- Hyperparameters. Default hyperparameters are in `hyp.scratch.yaml`. We recommend you train with default hyperparameters first before thinking of modifying any. In general, increasing augmentation hyperparameters will reduce and delay overfitting, allowing for longer trainings and higher final mAP. Reduction in loss component gain hyperparameters like `hyp['obj']` will help reduce overfitting in those specific loss components.

The Training
In brief the trainng process is:

- First, pretrain the first 20 convolutional layers using the ImageNet 1000-class competition dataset, using a input size of 224x224

- Then, increase the input resolution to 448x448

- Train the full network for about 135 epochs using a batch size of 64, momentum of 0.9 and decay of 0.0005

- Learning rate schedule: for the first epochs, the learning rate was slowly raised from 0.001 to 0.01. Train for about 75 epochs and then start decreasing it.

- Use data augmentation with random scaling and translations, and randomly adjusting exposure and saturation.

6.4 Ensemble

Ensemble modeling is a process where multiple diverse models are created to predict an outcome, either by using many different modeling algorithms or using different training data sets. The ensemble model then aggregates the prediction of each base model and results in once final prediction for the unseen data. The motivation for using ensemble models is to reduce the generalization error of the prediction. As long as the base models are diverse and independent, the prediction error of the model decreases when the ensemble approach is used. The approach seeks the wisdom of crowds in making a prediction. Even though the ensemble model has multiple base models within the model, it acts and performs as a single model. Most of the practical data mining solutions utilize ensemble modeling techniques.[?]

Chapter 7

The Training

In brief the trainng process is:

- First, pretrain the first 20 convolutional layers using the ImageNet 1000-class competition dataset, using a input size of 224x224
- Then, increase the input resolution to 448x448
- Train the full network for about 135 epochs using a batch size of 64, momentum of 0.9 and decay of 0.0005
- Learning rate schedule: for the first epochs, the learning rate was slowly raised from 0.001 to 0.01. Train for about 75 epochs and then start decreasing it.
- Use data augmentation with random scaling and translations, and randomly adjusting exposure and saturation.

7.1 Training Settings

Before modifying anything, first train with default settings to establish a performance baseline. A full list of train.py settings can be found in the train.py argparser.

- Epochs. Start with 300 epochs. If this overfits early then you can reduce epochs. If overfitting does not occur after 300 epochs, train longer, i.e. 600, 1200 etc epochs.
- Image size. COCO trains at native resolution of `--img 640`, though due to the high amount of small objects in the dataset it can benefit from training at higher resolutions such as `--img 1280`. If there are many small objects then custom datasets will benefit from training at native or higher resolution. Best inference results are obtained at the same `--img` as the training was run at, i.e. if you train at `--img 1280` you should also test and detect at `--img 1280`.

- Batch size. Use the largest `--batch-size` that your hardware allows for. Small batch sizes produce poor batchnorm statistics and should be avoided.
- Hyperparameters. Default hyperparameters are in `hyp.scratch.yaml`. We recommend you train with default hyperparameters first before thinking of modifying any. In general, increasing augmentation hyperparameters will reduce and delay overfitting, allowing for longer trainings and higher final mAP. Reduction in loss component gain hyperparameters like `hyp['obj']` will help reduce overfitting in those specific loss components.

7.2 Ensemble

Ensemble modeling is a process where multiple diverse models are created to predict an outcome, either by using many different modeling algorithms or using different training data sets. The ensemble model then aggregates the prediction of each base model and results in once final prediction for the unseen data. The motivation for using ensemble models is to reduce the generalization error of the prediction. As long as the base models are diverse and independent, the prediction error of the model decreases when the ensemble approach is used. The approach seeks the wisdom of crowds in making a prediction. Even though the ensemble model has multiple base models within the model, it acts and performs as a single model. Most of the practical data mining solutions utilize ensemble modeling techniques.[?]

Chapter 8

Limitations

YOLO imposes strong spatial constraints on boundingbox predictions since each grid cell only predicts two boxes and can only have one class. This spatial constraint limits the number of nearby objects that our model can predict. Our model struggles with small objects that appear ingroups, such as flocks of birds.Since our model learns to predict bounding boxes fromdata, it struggles to generalize to objects in new or unusual aspect ratios or configurations. Our model also uses relatively coarse features for predicting bounding boxes since our architecture has multiple downsampling layers from the input image.Finally, while we train on a loss function that approximates detection performance, our loss function treats errors the same in small bounding boxes versus large bounding boxes. A small error in a large box is generally benign but a small error in a small box has a much greater effect on IOU.Our main source of error is incorrect localizations.

Chapter 9

Result and Analysis

Here we will compare our model with existing models and our proposed model result. We tried this dataset in 5 different models:

Table 9.1. YOLO Network Structure

Model Name	mAP (our dataset)	COCO mAP	Speed (ms)	Detect Small object and less overlapping
EfficientDet D4 1024x1024	12.27	48.5	133	YES
SSD ResNet152 V1 FPN 1024x1024 (RetinaNet152)	8.75	38.3	104	NO
Faster R-CNN ResNet101 V1 1024x1024	11.56	37.1	72	NO
Faster R-CNN Inception ResNet V2 1024x1024	11.89	38.7	236	NO
YOLO v5(1024x1024)	12.06	49.7	65	YES
YOLO v5(1024x1024) Modified (Proposed Method)	13.46	49.8	70	YES

After benchmarking with above model we found that our model do best both in time and accuracy.

The dataset in the start is labelled for 21 classes Ambulance, Auto Rickshaw, Bicycle, Bus, Car, Garbage Van, Human Hauler, Minibus, Minivan, Motorbike, Pickup, Army Vehicle, Police Car, Rickshaw, Scooter, SUV, Taxi, Three Wheelers (CNG), Truck, Van, Wheelbarrow and has been trained for 60 epochs using the network configuration of Table 9.2. The network has been trained for 60 epochs that are 20000 iterations (4 epochs are 1328 iterations) because darknet framework trains a maximum of 2000 iterations for each class and calculates the mAP@0.5 every 4 epochs for the validation set and every 3 epochs for the test set unless the difference between the cost of an iteration and the cost of the next iteration is lower than 0.0005.

For evaluating which weight file is the best one to use for detecting vehicles in new images we will guide by the mAP value computed in the validation and

Table 9.2. Initial values for the YOLO network parameters.

Parameter	Value
Batch size	64
Subdivisions = sub batches: number of mini batches it is split up the batch	32
Width × height of the input images	416 × 416
IoU minimum threshold between predicted BB and ground-truth	0.5
Data augmentation	Saturation and exposure=1.5 angle=0 hue=0.1
Learning rate	0.001
Max_batches	Number class × 2000
Momentum	0.9
Decay	0.0005

Table 9.3. Evaluation of the model on the validation set with an IoU threshold of 50Area-Under-Curve for each unique Recall.

Epochs	Ap Car	Ap Rickshaw	Ap Bus	Ap SUV	Ap Truck
3	39.35%	09.33%	39.57%	16.76%	27.82%
6	41.65%	13.68%	41.73%	18.74%	30.76%
9	44.15%	15.01%	43.53%	19.02%	32.07%
12	48.35%	15.93%	46.98%	19.38%	33.43%
15	49.89%	16.45%	50.13%	20.31%	34.76%

test set. In the Table 9.3 we can see which weights files give best value on the mAP. The training has been done for 15 epochs, but it can happen that the weights computed in previous epochs give a better value on the precision and the mAP. This may happen due to overfitting; the model is very good detecting on the training set but falls to detect objects in new images. As it can be seen in Table 4, the weight file stored in the epoch 15 and the epoch 18 obtain the best mAP value of 37.20% and in the Table 3 the weight file obtained in the epoch 16 gets a 35.52% mAP best value.

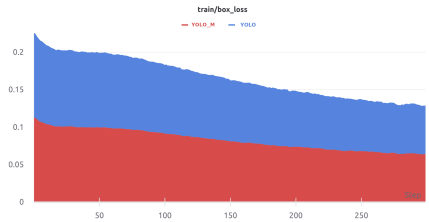

FIGURE 9.1. The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

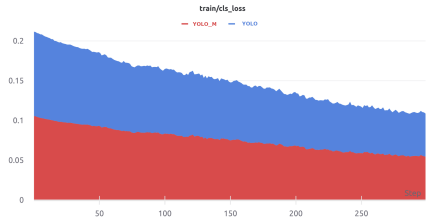

FIGURE 9.2. The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

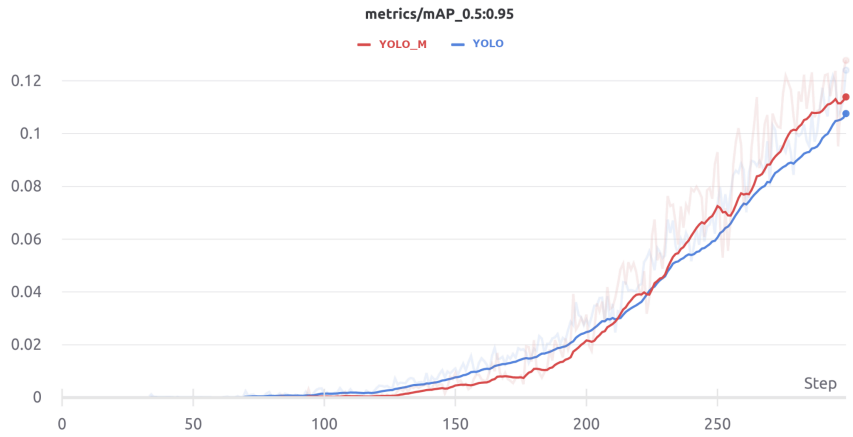

FIGURE 9.3. The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

In the Figure 9.3 we can see the graph that shows the average loss computed for every iteration and the mAP calculated for every 4 epochs (1 epoch = 332 iterations so 4 epochs = 1328 iterations). The value of the loss remains between 3.5 and 4 and the mAP has a value of 35% from the iteration 5000 (epoch 15) to the iteration 6000 (epoch 18). With the Figure 9.3 and the Table 9.3 we can conclude that the best weight file for detecting vehicles in new images is obtained in the epoch 15. Note that the dataset is quite complex as it has many labelled small vehicles in each images that are not visually easy to see, this complexity affects the mAP value by obtaining a relatively low one. This happens because the model fails to detect some vehicles in the validation set that it is difficult even for the human eye to detect them. On the next pages, to test the capability of generalizing of our model for this training, we will get one image containing cars, other image containing trucks, other image containing motors and a final image containing cars, trucks and SUVs and pass them from our model to see if it is able to detect all the vehicles in the images.

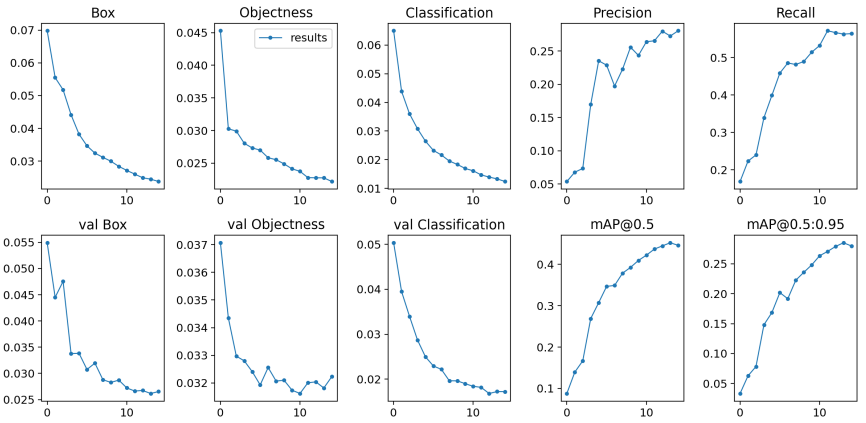

FIGURE 9.4. The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

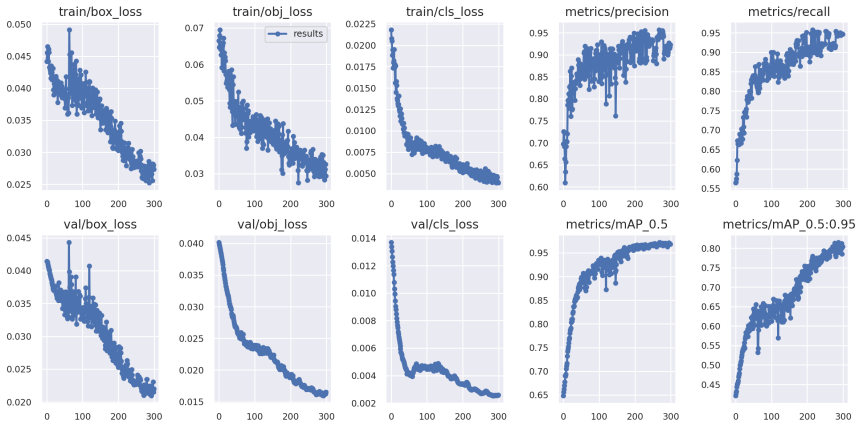

FIGURE 9.5. The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

Part II

Results

Part III

Epilogue