Road marking condition monitoring and classification using deep learning for City of Helsinki

Dana Tokmurzina

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 26.09.2020

Thesis supervisor:

Prof. Alexander Jung

Thesis advisor:

M.Sc. Saska Lohi

Aalto University
School of Science

AALTO UNIVERSITY
SCHOOL OF SCIENCE

ABSTRACT OF THE
MASTER'S THESIS

Author: Dana Tokmurzina

Title: Road marking condition monitoring and classification using deep learning for City of Helsinki

Date: 26.09.2020 Language: English Number of pages: 6+60

ICT Innovation EIT Digital Master School

Professorship: Data Science SCI3095

Supervisor: Prof. Alexander Jung

Advisor: M.Sc. Saska Lohi

The thesis explores application of deep learning on detection and classification of road markings in the city of Helsinki. The need for maintaining the infrastructure is the essential part for smart cities. City of Helsinki is thriving towards the digitization of the city, providing geo spatial information on one of their open geoinformatics service, https://kartta.hel.fi. That was utilized as the main data source. Based on the satellite images obtained from kartta.hel, the road markings were extracted.

Further, previous work on zebra crossings was studied, both using traditional ways and deep learning (DL) based ones. Deep Learning were favoured over traditional due to ability to capture deeper abstract concepts and hierarchical features. Several recent DL based object detection algorithms, their training process, hyperparameter tuning, results are described in depth. In addition history of computer vision, especially object detectors, their benchmarks, disadvantages, and advantages are studied extensively. Taking into account the specifics of the dataset such as low resolution, small size and noise, data augmentation and transfer learning were applied. After the comparison between various object detection algorithms and also taking into account requirements for the performance as accuracy, robustness to noise, shadows, state of the art algorithms were chosen, such as Retina Net and YOLO5. YOLO5 outperformed in all desired metrics. It achieved mAP_0.5 of 0.68, inference time of 0.017 seconds with relatively low (compared to RetineNet) time for training. In addition it produced good visual results on the test dataset.

Keywords: Deep Learning, object detection, Retina Net, YOLO5, data augmentation

Preface

I want to thank my family, my boyfriend, friends for continuous support and motivation and their genuine interest in my work! I am glad that I could use the knowledge gained in 3 different universities Nazarbayev University, University Cote d'Azur, Aalto University and learn the perks of conducting an extensive research. These 6 month I learnt how to work on something big, read huge amount of scientific papers and dive into the world of computer vision. I realized that many discoveries in deep learning (but in general, everywhere) were because of brave curiosity. This sparked in me a wish to someday create something that can be called an innovation!

I am grateful that I met professor Alexander Jung who actively spreads Machine Learning in a very engaging way for people. I want to thanks all such people that are willing to share their knowledge and put their highest effort for it. I hope that one day I can continue sharing the knowledge too. I also grateful for the City of Helsinki for the provided opportunity.

Otaniemi, 26.09.2020

Dana Tokmurzina

Contents

Symbols and abbreviations

Abbreviations

ML	machine learning
CNN	convolutional neural networks
AI	Artificial intelligence
DCNN	deep convolutional neural networks
IoT	Internet of Things
MSER	Maximally Stable Extremal Regions
FAST	Features from Accelerated Segment Test
HOG	Histogram of oriented gradients
POI	point of interest
RoI	region of interest
SIFT	Scale Invariant feature Transform
YOLO	You only look once
GPU	graphical processing unit
RELU	Rectified Linear Unit
SSE	Sum of Squared Errors
LBPH	Local Binary Pattern Histograms
FL	focal loss
LLBPH	Local Binary Pattern Histograms
DPMs	Deformable Part-Based machines
SVM	support vector machines
GLCM	gray level co-occurrence matrix
COCO	Common Objects In Context
AP	Average Precision
FPS	Frame per Second
SPP	spatial pyramid pooling
R-FCN	region-based fully convolutional network
FPN	feature pyramid networks
SSD	singleShot MultiBox
DSSD	deconvolutional single shot detector
DSOD	deeply supervised object detectors

1 Introduction

Smart cities are the term that constantly appears in media. The meaning of that term is vague as everything that more technologically advanced is perceived as smart. However, the main goal in designing smart cities should be concentrated near advantages it will bring to the citizens and how the city in general will increase the overall level of people's life quality. Safety of citizens is one of the main concerns that prevents adaption of some technologies, for example autonomous driving cars. However, that leads to drastic improvements in that area, motivating computer vision, IoT, AI sensor technologies to evolve rapidly.

The City of Helsinki is rapidly adapting new smart technology to meet the needs of increased urbanization, tourism and ease the life of its citizens. The road infrastructure is one of the main problems of any modern and especially capital city. Rigorous planning of road markings location prevents traffic accidents, by maintaining the traffic flow order. Moreover it assists pedestrians, vehicle drivers and autonomous cars.

Past few decades Helsinki has drastic decrease in traffic accidents. According to City of Helsinki traffic engineer Jussi Yli-Seppäl improvement in traffic safety is the result of several factors and one of the most important is the development of technology [1]. The advancement in sensors, computer vision now allows better urban planning.

The City of Helsinki developed open map service https://kartta.hel [2] that allows to access to aerial photographers of the Helsinki to view city plans, the location of the buildings and roads. Having that information helps in management. As the important part in management is to be aware of the current situation, monitor and take corresponding steps towards improving the planning. For example scheduling in advance the road maintenance.

The goal of this paper is to help with the maintenance of road marking in the City of Helsinki. That problem was approached with proposing a classification algorithm according to the [9] that will help to monitor the conditions of the roads from satellite images, by classifying them into 5 states and then accordingly plan their maintenance.

That could reduce the costs for the city planning and maintaining, avoid overdue repair. Lack of proper infrastructure leads to cause of potentially dangerous situations on the roads. In addition the opportunity to predict time for maintenance by investigating what regions are likely to have in future road markings that are, fading, peeling. Since road markings that are prone for damages mostly reside on parts of the road with the heaviest traffic flow, that areas can be prioritised for maintenance in advance. Also the valuable information about traffic flow assists in overall city planning. In general, knowing what road markings are prone to lose quality can prevent hazards and accidents. The main reasons for fading road markings are high traffic areas, snow plowing and de-icing, low quality of the asphalt, extreme temperatures, snow melting salts [10].

The algorithm for road markings classification should be robust as the consequences are the safety of the pedestrians and all actors involved in traffic movement.

The images obtained from the satellite imagery do not have a good resolution. Moreover, they are occluded by cars, pedestrians, shadows. However, the recent developments in the machine learning showed considerable good results in the general object detection.

This paper will investigate different methods that can be applied to detection and classification of the road markings. The related literature is reviewed, pre-processing of the images and both traditional methods and deep learning-based methods are studied. The classification of road markings into 5 classes was not implemented before. The feasibility and ability of neural network to properly learn features required for classification are accessed. The data was labeled manually, however several adjustments in labeling process were made after testing the algorithms. As one of the future scenarios is deployment of the proposed machine learning algorithm, the possible way is reviewed. As the deployment step favors the robust and fast algorithm, the paper will also focus on choosing the algorithm with highest possible speed, however prioritising accuracy. The training speed will also be one of the metrics. In general we are answering the question:

- The feasibility and ability of neural network to properly learn features required for classification of pedestrian road markings.
- The performance of the designed Deep learning-based methods based on mAP and speed and training time.

The remainder of this paper is structured in the following ways. Section 2 provides a background and overview of the fundamental concepts in image processing and machine learning fields, applied in the paper. It includes latterly conducted research with traditional methods, and deep learning based approaches, with the results they have achieved Section 3 provides information about the data we used, how it was gathered and prepared for further training. Further, Section 4 describes the experimental setup. The methods that we used and their results. Section 5 concludes the work done and suggests future work routes.

2 Background

2.1 Image

Digital image is the collection of pixels. The number of pixels in the image is called resolution. The black and white images are represented as 2D matrix with height and width and each value is proportional to the grey values or the brightness of the image at that pixel. The color is present as a third dimension, which is the depth. Usually 3 color channels and sometimes opacity are used to fully represent the color of the image. As the example, 3-megapixel camera that has 307, 200 pixels which represented by 640 by 480 display. If the color channels will be included, 9 million data points will be used per one image. That enhances the appearance of the image however as the input to the ML model, can cause serious performance problems as increased time for training and usage of more computer power. Computer vision is the ability of the computers to interpret images by reading their pixels. However if two images are same but one is rotated by 90 degrees, it will be totally different image for the computer. The same way, rotated, distorted, skewed, zoomed, shadowed images are all different for computer perception.

2.2 Properties of the road marking images

For the detection of pedestrian crossing images, various image processing techniques are used. The methods mostly relay on the color variation information and geometric properties as stripes, repetition of them. Moreover current high-resolution aerial images provide spectrum, texture and shape as the main features for traditional the zebra crossing detection [23]. According to [11, 23] zebra crossings are distinguishable from the surrounding background due to a sharp contrast, and in general are brighter. The second characteristic is a texture, meaning that the detection is simplified knowing that there is a periodic stripes. Shape of pedestrian crossings is parallelogram, however due to the distortions, and different angles at which images are taken, the shape has various sizes. In Singapore the road markings are squared and does not have repeated stripes.

After transforming image to the bird-eye view, the rectangle shape is preserved. Practice showed that the spectrum is not a reliable characteristics. That is due to the lighting condition, shadows and occlusions, that affect spectrum a lot. The texture is also prone to changes if significant part of the road marking is deteriorated or removed. Moreover if the image has low level of the view, then patterns of stripes can be perceived as vanishing. The same is true for a shape. However due to average stability of the images in texture and shape, they are used as main features in research works [23]. According to the literature review done on 81 resources on the Advanced Driver Assistance Systems, in [30] the research generally followed a common process: image preprocessing, object detection, object recognition and in video based datasets, video tracking. That is true for both traditional and deep learning based methods. The preprocessing steps involved in both methods mostly are the same. However the next steps for object detection, object recognition are different. This work will try

to give a brief overview of both approaches, based on traditional image processing techniques and deep learning ones.

2.3 Traditional methods

2.3.1 General steps in traditional object detection

Traditional methods are based on extraction of low level simple features as edge, color, and scale-invariant feature transforms [28].

The most common procedure for object recognition using traditional methods after preprocessing, are the following set of rules.

- Image Preprocessing. According to Gonzalez [31], main techniques for preprocessing stage are spatial and frequency filtering, intensity transformations. These steps are common also for deep learning based solutions.

- Proposal generation. It is likely, especially for high quality satellite images to contain redundant information. As the computation on the entire images is expensive and the noise can affect the algorithm performance, it is important to do a proposal generation. This stage involves locating the area where the object of interest is located. That specific regions is called: Region of Interest (ROI). That is implemented manually, using heuristics. According to Tang [30] most of the traditional methods are done in the frequency domain. Moreover there are three main approaches: sub-sampling, vanishing point detection and perspective analysis and projective modelling. However often many proposed regions does not incorporate any necessary information, resulting in many false positives. Many region proposal algorithms are computationally expensive.

- Feature vectors extraction is a second stage. Feature vectors try to incorporate discriminative semantic information about the ROI. That is obtained by using methods as SIFT (Scale Invariant feature Transform), Haar filters, HOG (Histogram of gradients), SURF (speeded up robust features).

- Region classification is the third step where the suitable ROIs are labeled. One of the traditional machine learning algorithms as Support Vector Machines (SVM) performs well on classification of small training data. The robust models that are based on the traditional methods are in need of well performing region classifiers and representative features. One of the successful implementation was done using PASCAL VOC dataset in 2009. The designed solution is called Deformable Part-Based machines (DPMs). Deformable loss is used on integrated multiple part models. Moreover it uses latent SVM in later stages.

These steps can be easily influenced by the changes in threshold values or color intensity. They are quite robust when working with same data however not adaptive to changing environments or new conditions. Algorithm which rely on edge detection, need a noticeable contrast between background and road marking. The location difference that can be due to regional design of road markings, quality of city

infrastructure or even illumination, prevents from adaptation of cross-regional models for road markings. For example from works of Riveiro [18] it was noticed that false negatives (missed crosswalks) mainly are due to poor quality of road markings, which is true for countries with less developed infrastructure. The next years did not have any significant gains from traditional models. However their complexity grew.

2.3.2 Related work

Pedestrian crossings detection is widely studied research area as numerous works were done already [11, 21, 19, 17]. The overall detection of road crossings is quite difficult due to the noise that is common for images that contain road data. Traffic scenes can have shadows, pedestrians, different kinds of vehicles, shadows from trees. The road markings moreover can vary region to region, and so making a zebra detection a difficult task.

The main motivations of many researchers were reduction in traffic dangers, according to Franke and Heinrich [15] and assisting people with visual impairments [6].

The work done in the field of pedestrian crossing detection using traditional methods, was mainly aimed in captioning the distribution patterns of pedestrian crossings using geometry and color characteristics.

Pedestrian crossings take various forms throughout the regions. In most of the cases they are well known as evenly spaced and parallel white stripes. The main geometry features are lines and edges which are repeatable and consistent [17]. Many researchers were using the images obtained from the camera attached to near road columns. They utilize methods specific for the images taken in such angle view as protective geometry constraints.

Couglan and Shen [19] proposed a method that grouped geometric features to detect zebra crossings, however their method was performing poorly on images with multiple crossings.

Chyi-Ren et al. [21] utilized the environmental parameters to construct environmental feature vectors and determine the crossing area. That made possible to detect well one crossing per image. In order to overcome problem of detecting zebra crossing in Fig 3 that is in front of the camera and missing partial crossings located far from centre. Fan [17] used coarse-to-fine framework that is able to detect all crossings in the image. Their method is purely based on traditional methods of image processing. To preserve the discriminative information for detection, they use visual cues and apriori constraint information. Firstly, in the coarse stage, the vanishing point is detected. Next, line and edges in combination with vanishing points. That results in RoIs candidates for the fine stage. Fine stage incorporates spatiotemporal information such as geometrical and statistical information specific for road markings. In order to locate pedestrian RoIs, refinement and fine localization are performed, including identification of the geometric and statistical constraints of the pedestrian crossings. Below is related work done in some specific areas:

2.3.3 Edge detector

Li et al. [16] used both color and shape features, that are local adaptive threshold for color and Canny edge detection. Their results were mostly dependant on accuracy of the Canny edge detector.

2.3.4 Texture

Sun et al. utilized mainly the texture feature of road markings [23]. JointBoost classifier is suitable for specific gray level changing texture. Gray level co-occurrence matrix (GLCM) with several 2D Gabor filters. Filters with low frequency enhance globally image. High frequency enhances the details. The latter are used due to distinguishable responses in frequency domain due to frequent occurrences of stripes. GLCM features represents information about direction, distance and gray scale variation of pair of pixels. Five features as: angular second moment, correlation, contrast, homogeneity, entropy. The statistics obtained from it acts as features for rotation invariable features. After features are extracted, Joint Boost classifier is applied, Joint Boost increases the weights for misclassified samples from the previous iteration step. This makes the training procedure more effective.

2.3.5 Preprocessing

Sun et. al. utilized image enhancement technique in order to reduce noise and optimize contrast for feature detection [23]. They applied the Wallis filter that acts as a local smoother operator. It is able to increase the gray contrast in the parts with small contrast and on the contrary decrease where the contrast is large. It especially efficient when the contrast between zebra crossing and background is not strong.

2.3.6 IPM

On of the popular ways in preprocessing is to transform the image from the front view to bird's-eye-view, which is mainly done with inverse perspective mapping (IPM). The use of IPM, as the preprocessing helps in dealing with noise, removal of redundant obstacles on the road, focusing on the road; transforming non linear lines into straight lines. According to [28] IPM transformations has some limitations. As on of the examples, while vehicle is moving, it is unavoidably changes its pitch or roll and and meets sudden obstacles on the road. That causes distortions for IPM's algorithm. In addition, the predetermined projection matrix is calculated beforehand.

2.3.7 Lane detection

Lane detection that includes detection of specific, for road markings lines, edges. Ahmetovic et. al. in 2014 devised a ZebraLocalizer algorithm for zebra crossing recognition [6]. Authors searched for line segments by adapting EDLines algorithm developed by Akinlar and Topal [24]. The resulting line segments are grouped according to their characteristics as a horizontal distance, vertical distance, and

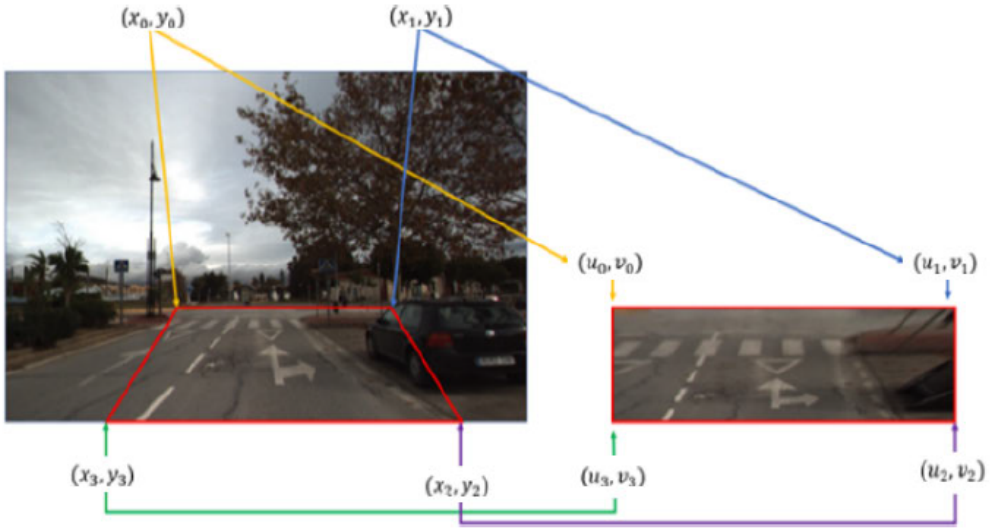

Figure 1: Example of transforming image to bird's eye view [28]

parallelism. Next candidate crossings are proposed. They go through discarding procedure based on the amount of stripes they contain and the gray scale intensity. The validated zebra crossing is put in the set of other validated zebra crossings. Each zebra crossing in that set has 2 features: orientation of a line that is orthogonal to all the parallel stripes and the smallest 4 corner bounding box, incorporating all the stripes.

2.3.8 HOG

Koester et al. [27] extracted road markings based on histogram of oriented gradients (HOG) and Local Binary Pattern Histograms LBPH. According to authors HOG features capture well edge directions (i. e. local intensity gradients) that are common for aerial photos of zebra crossings. And using HOG block normalization deals with shadows and deteriorated parts. Extraction of LBPH features was also done, but did not show improvements, however LBPH features provided slightly better performance. As the classifier, they trained SVM and radial based function for comparison. to classify road markings. The database they used had one regions specific road markings, that is why the recall dropped significantly, from 95.7% to 38.4% when their method was tested on another region [4]. This is another proof that many methods show low generalization ability with traditional methods. The comparison to line search algorithm [6] proposed by Akmetovic et. al. showed competitive results 97.2% to 95.7% on San-Francisco dataset.

2.3.9 Template Matching

Template matching technique was applied by Herumurti [25]. He used a circle mask template on aerial images and SURF for localizing zebra crossings. Achieved speed

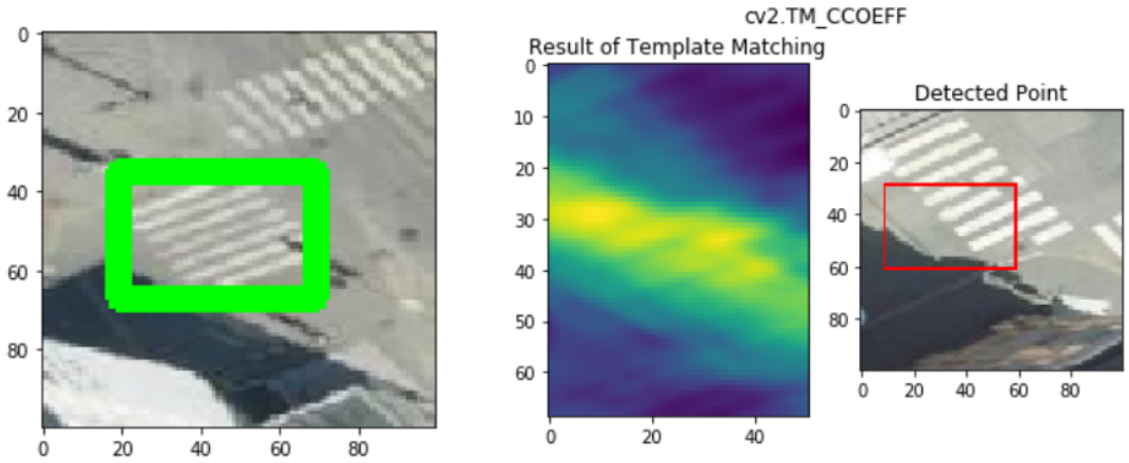

Figure 2: Example of Template Matching Correlation Coefficient applied for our dataset.

was 739.2 seconds to detect 306 crosswalks.

Li et al. [22] devised a shape-based method for detection of different road markers. After obtaining bird-eye view and binarizing image, the authors used the 8-neighbor chain codes and moment features to represent the shape of the road marking. The resulting step is a shape matching to detect road markings. Wu et al. [11] presented a system that uses images of road markings that are taken from in-car camera. Their system works even for images with multiple disconnected road markings components, despite the presence of other kinds of road markings. Further the system is able to learn the feature-based templates. That templates are made according to following procedure. The first step is an image rectification, as the images that are taken from a low viewpoint can have significant perspective distortion. The output of this algorithm is the bird's eye view image. The obstacles for inverse transform can be hills, roll movements. The authors [11] assume that RoIs of road markings are brighter that the surrounding environment. That motivated them to use MSER features which also assume that pixels inside RoI are either darker or brighter than outside. Next, sets of feature points, points of interest (PoI) are computed with Features from Accelerated Segment Test (FAST) corner detector, which is faster by 39 times than SIFT, according to authors.

Having bird's eye view and POIs allows to compute HOG histogram of oriented gradients descriptors, which is 128 dimensional feature vector. The HOG vector is computer using optimal scales and orientations, which is in this work for computational savings, were taken as default value of 3 for scales and 1 for orientation. That results in 384 dimensional vector for each POI. The above mentioned procedures are done for all template images and result in a source of templates. Next the same step of rectification, maximally stable extreme regions (MSER), FAST corner detection and obtaining the HOG descriptors are conducted for test images. Taking into account

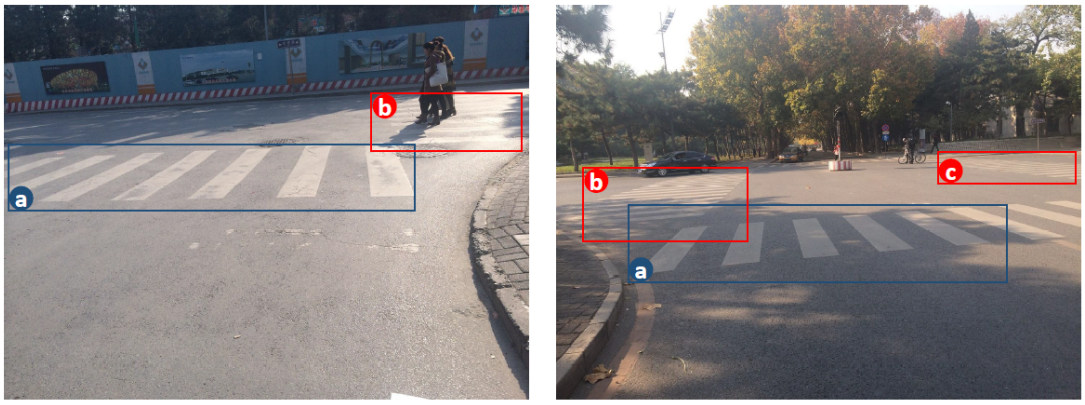

Figure 3: Problem of detecting only one crossing in front of camera (in blue box) and omitting partial ones (red box).

that there is possibility of having redundant RoIs or multiple patches can be matched for one template, the concluding refinement algorithm is necessary. Having edges and coordinates of POIs, structural matching algorithm computes the cost between two shapes. That shapes are formed by sets of POIs in test and template images. In general the proposed system works well in real time for road with lighting variations, limited occlusions. However, according to authors, better threshold for matching POIs and a tighter cost for computing similarity between shapes is needed.

FAST corner features are used as features that define needed templates. It works in real time, without using GPU. Those are MSER features that are highly covariant regions in the image. MSER method as shown from their work [11], is more stable towards motion blur, change of view points, illuminance, peturbations.

2.4 Deep Learning based methods

The traditional methods used for the object detection are mainly based on heuristics, hand crafted features. They are suitable when the training data is small, and when there is not enough computing power. It usually requires large amount of the manual coding for every type of the road marking for which a good understanding of underlying data and possible outliers is also needed. They should consist of carefully designed feature descriptors for every specific needs. However, it requires a lot of domain knowledge in OpenCV, image manipulations, filters. The anticipation of future images, meaning that all changes over time are known in advance. That can also be required if the system will be utilized in future. That methods does not scale for various kinds of road markings.

However the above mentioned deep understanding of the specific image features is not true for deep learning based models. In general deep learning models are black boxes and they lack explainability and theoretical support. They require more memory and speed as data quality increases, emergence of real time tracking, video data.

However the way that neural networks were able to learn hierarchical features,

starting from edges, corners, then going to patterns and even parts of the face means that they can represent more abstract concepts [13]. In other words, inputting just pure image pixel and getting as the output high level information. Deep learning based models capture dynamically, different scale information throughout different layers.

2.4.1 Deep Learning

Deep Learning models are also ML models, however their structure is made from stacked layers that extract features and with more layers (deeper models) the features become difficult and they are being learnt. That is why it is named Deep Learning. The first and basic deep learning model is artificial neural network (ANN) that consists of sequence of fully connected layers. Each layer holds specific amount of neurons and neurons of each consecutive layers are interconnected with each other, so creating connections. By introducing neurons, we assign weights for every connection. During training some connections are favored by assigning them bigger value. But they way how, some connections are favoured is the result of constant comparison of output produced by all the layers and weights. If some weights are contributing for the output meaning reducing the loss, difference between predicted and actual (to the one of the known, ground truth) is smaller, then that weight is favoured. And backpropagation occurs, which is a process of reassigning the weights according to the contributions of previous weights to the loss. That is the main summary of DL however now different models exists. They differ in architecture which is influenced by the desired outcome in terms of how underlying structure of the data should be handled and how rich are features [7].

2.4.2 Deep Convolutional Neural Networks

Neural networks were inspired by the idea of how might the brain perceive the images. The similar first attempts like 'neocognition' by Fukishima [5] that produce that was aimed to be a hierarchical and spatial invariant model. LeCun et al. [40] added backpropagation and optimization in form of stochastic gradient descent. In 1998 he published the paper that first introduced LeNet-5 architecture to recognize handwritten digits. The novelty was the introduction of convolutional layers and pooling layers. Further the limitations of small labeled datasets and computing power were eliminated in 2009, Jia et. al. were able to gather Imagenet, with 1.2 m labeled high resolution data and later 2012 Krizhevsky with advances in high-performance parallel computing systems, was able to train on ImagiNet for Large Scale Visual Challenge competition. The model, AlexNet, which is a CNN based [8] is able to extract features from raw RGB pixels in order to predict the label.

Compared to basic ANN model when applied to image will not consider the local connectivity of the pixels. However CNN were designed specifically to tackle that problem. In CNN, the neurons of previous are current layer are not fully connected, instead only the neurons in the current layer are connected to a small region of the previous layer (see Fig.4 a). These layers are known as feature maps. Instead of linear connections in ANN, CNN uses convolutional operations which are weights and

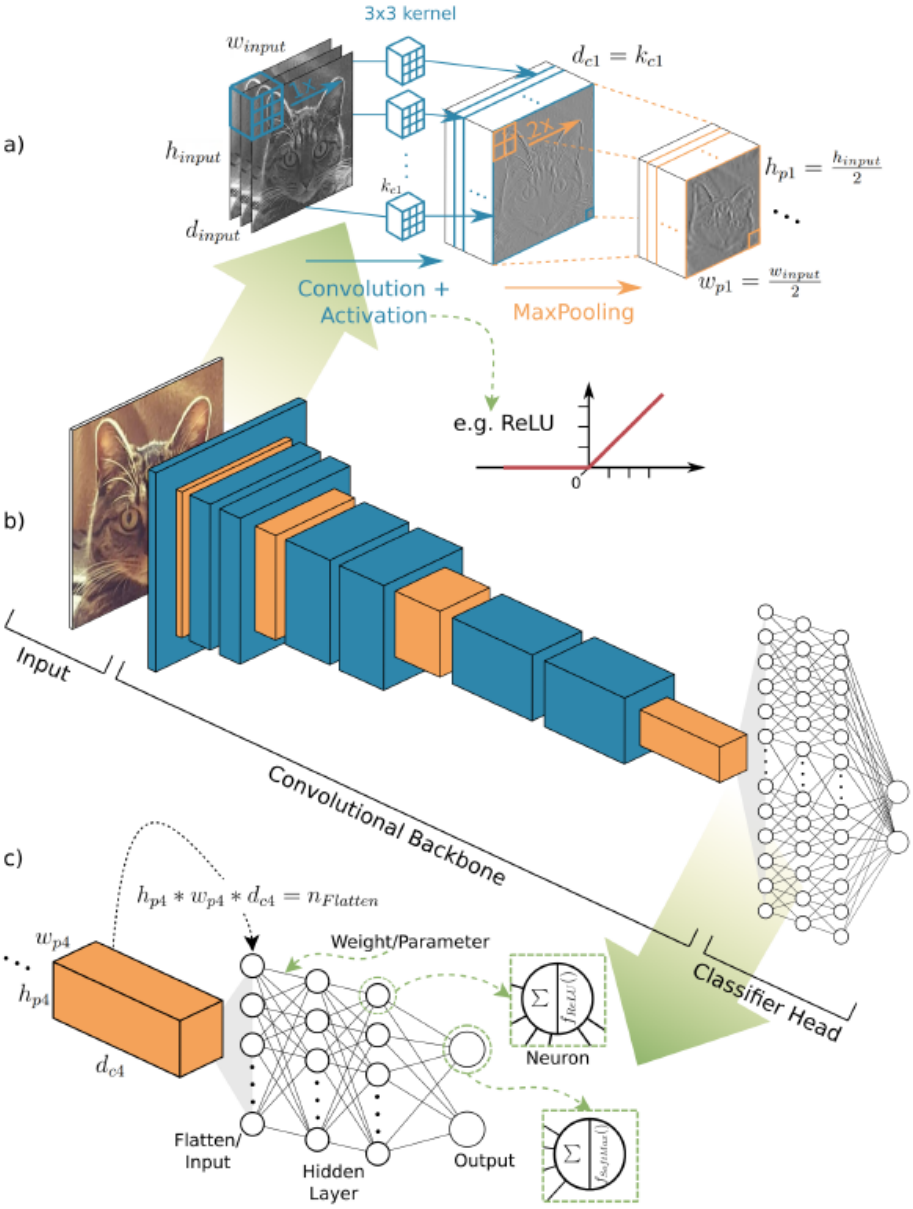

Figure 4: Hoeser et. al. CNN architecture [7]

also adjusted during training. So the features are not hand-crafted but are output of a learning process.

In Fig. 4 a. the input image of the cat is fed into the network. In the first convolutional layer, 3 by 3 kernel or filter is convolved with the input matrix. *Convolved* is this case means a dot product is performed on each filter sized patch of the input and the kernel (filter). Such convolution of an image with different kernels acts as image processing operations like edge detection, blurring, etc. Usually convolutional layer has several filters, that make it able to detect multiple features anywhere in the input. The filters are applied systematically across the pixels allowing to find suitable features across the entire image. That enables translation invariance, ability

to investigate not where the feature is but if it actually present. So changing the size, applying slight distortions will not affect the fact that feature will be detected.

After convolution, two dimensional array which is called feature map is produced, one feature map per filter. Feature maps emphasizes the areas in an image influencing filter the most [37]. In Fig. 4 a. a stack of d_{c1} feature maps which is the output of a linear kernel functions. As in ANN, we pass feature maps through non linearity functions, for example RELU in Fig. 4 a. ReLu is Rectified Linear Unit and the functions are given by $f(x) = max(0, x)$. Other functions as *sigmoid* and *tanh* are used less due to good performance of ReLU.

After, it is common in practice, to reduce number of parameters, especially when image size is large. For that reason pooling layer is used. The spatial pooling (subsampling or downsampling) that preserves important information but is able to shrink the sizes. There are several kinds depending on the type of aggregation such as max pooling, average pooling, sum pooling, l2 pooling. Max pooling is a common type of pooling [37]. Despite reducing number of parameters, computation it also introduces invariance to small translations such as scale, rotational [37]. Pooling also enables semantically low level features with bigger spatial dependence to be turned to several high level features with low spatial relevance [7]. By tweaking the layers in CNN, the model invariant to skew, color, thickness, brightness can be created [37]. In Fig. 4 b. the basic architecture for image classification is presented. There are 3 modules: Input, Convolutional backbone and classifier head. Convolutional backbone is comprised of feature maps. In Fig 4 c. the transition between classifier head and backbone are shown. Usually the last layer is fully connected ANN with soft-max output for probabilities. Classifier head contains number of neurons equal to number of output classes.

2.4.3 CNN Architectures for image recognition

Convolutional backbone comprises the most important part of the model. Throughout the history convolutional backbone, its architecture was changing depending on the target. In general, according to Hoeser et. al. [7] there are 4 families of CNN families: the Inception, ResNet, MobileNet families and Efficient design architectures. As the benchmarking different designs are evaluated according to their designs, number of parameters and top five accuracy (acc@5) which allows to access the performance of the algorithm, while number of parameters help to see the efficiency of the algorithm.

- Vintage Architectures

  Examples of vintage neural networks are AlexNet [8] with ZFNet [38] and VGG-19 [39]. They share similarity is structure: repeatable convolutions followed by non-linear activation and max pooling layers, with a classifier head in the end. AlexNet performance in acc@5 is 81.8% with 62M parameters, ZFNet 83.5% and also 62M parameters, improving accuracy up to 92% with 144 M, by using smaller kernel sizes and deepening their network. However the considerable improvements in Vintage Networks was achieved by VGG-19 using stacked convolutions of constant size and deepen overall the network using 19 layers.

- Inception Family.

  At the same time when VGG-19 was shown to the world, GoogleLeNet [41] or Inception V1, which was the upgrade of first model of LeCun et. al. The novelty is the introduction of the stacked Inception modules, which are parallel convolutional layers of different size and max pooling. Resulting in increase in variance of different kinds of feature representations. Also sparse max pool and bottlenecks layers are added to avoid an increase in number of parameters. Later works were trying to avoid problems of vanishing gradient, which occur when there are too many layers. To solve it, 2 additional (auxiliary classifiers) classifies were added in the middle of the network, which makes access for adding additional gradients to early layers, which due to vanishing gradients, are not trained effectively. That auxiliary classifiers are omitted while inference. The next novelty in a new adapted Inception V1 was an addition of a batch normalisation after convolution but before ReLu activation. It also helped in a problem of vanishing gradients and increase in depth of network. This neural network reached 92.18% in acc5 with only 11.5 million parameters. Further Inception V2 and v3 were advanced by applying a factorisation on the original Inception [7]. Inception V3 reached 94.9% with 23.6M. Next improvements in architecture was introduction of Xception by Chollet [42], where the depth-wise separable convolutions extract features. The model showed efficiency of factorisation in space and depth.

- ResNet Family

  Xception was utilizing residual connections, that lead to a totally a new architecture family, ResNet. ResNet resembles the main architecture with having bottleneck design, targeting parameter efficiency. However their novelty is in the introduction of the special shortcuts (residual connections). That feature allows to increase the number of layers as it is possible to transfer larger gradients to early layers. ResNet model has multiple architectures as ResNet-152 (acc@5 95.5% with 60.3M parameters). Later ResNetXt was invented that contained cardinality which is parallelism of convolutional operations. ResNeXt-101 reached 95.6% with 83.6 M parameters. A variant of ResNet, DenseNet utilizing intensively residual connections and instead of adding layers it uses concatenation to merge them in a deep feature map. DenseNet-264 uses considerably fewer parameters, just 34M and shows acc@5 of 93.9%. In general, ResNet and Inception showed that CNN are modular models [7]. Recently released plug in modules, Squeeze and Excitation (SE) module [20], which are used in SENet, shows that even Vintage Architectures can be enhanced. SENet uses feature recalibration between channels and emphasize that features in a feature map that are useful. Adding SENet blocks ResNet-50 improves Fig 5 by 1.3% which is a relative 5.2% on COCO's standard metric AP. And deeper architecture ResNet-101 by 0.7 by 0.7% (a relative 2.6%).

- Efficient Design

  Current advances in Edge computing and the fact that there is a need to

run some models on devices with less computing power as mobile phones, demanded efficient models. The first products of this demand was MobileNet-224 with 4.2 M parameters and acc@5 89.9%. Next model MobileNet-V2 used inverted depth-wise convolutions with residual connections and was able to achieve 92.5%. After MobileNet introduction NASNet, a totally new view on defining an architecture was developed. The idea is that the best child network architecture is searched in space of the building blocks with the use of Recurrent Neural Network, which is in this case acts as a controller and searches for maximum performance accuracy. The training of every child is expensive, however when trained on small dataset, the new architecture can be scaled up to match larger dataset. The latest version NASNet-A performed at 96.2% with 88.9M. Further designs were concerning how search space is used and childnetwork is handled. Due to that PNASNet-5 (96.2% and parameters 86.1M) and AmoebaNet (96.3%, 155.3 M) were derived. MobileNASNet (MnasNet) uses more complex layer design which results in richer feature representations. This model outperformed MobileNet-V2 that uses handcrafted features. In late 2019, Tan and Lee [43] proposed EfficientNets that have scaling advantage. They use compound scaling that is able to uniformly scale in depth, width and resolution of the network. Achieved 97.1% acc@5 with 66M parameters.

	AP@IoU=0.5	AP
ResNet-50	45.2	25.1
SE-ResNet-50	46.8	26.4
ResNet-101	48.4	27.2
SE-ResNet-101	49.2	27.9

Figure 5: Object detection results on the COCO 40k validation set [20]

In general CNN enable consideration of representational features during learning. This gave a push towards using DCNN as the new standard solution for computer vision problems, as they learn better discriminative features in large and complex datasets. Regarding Earth observation research, DL reached it with a delay of 3 years [7]. In 2020 the number of publication involving remote sensing and DL doubled compared to 2019. Current applications span from weather forecasting, super-resolution data to object detection with different kind of sensors (optical, hyper-spectral, multi spectral) [7].

2.4.4 Evaluation metrics

Pattern Analysis Statistical modelling and Computational Learning Visual Project Classes Challenge (PASCAL VOC Challenge) is the international computer vision

challenge. They provide criteria for evaluating the performance of the algorithm. The formula states that accuracy for detection is directly proportional to ratio of number of t_p true positive examples (correctly detected) divided by sum of number false negatives f_n, and number of true positives.

$$Recall = \frac{t_p}{t_p + f_n} \tag{1}$$

$$Precision = \frac{t_p}{t_p + f_p} \tag{2}$$

2.5 Deep learning based Object detection

Current computer vision enhanced with neural networks is tackling problems in 4 fundamental fields: image classification, object detection, instance segmentation and semantic segmentation [12]. Image classification aims to classify objects into semantic categories. In addition to the obtained class, if the bounding box, that locates the object is searched for, then the problem becomes an object detection one. Further semantic segmentation deals with assigning a precise pixel level mask for each specific category. Instance segmentation goes more granular, than semantic segmentation. It separates objects of the same label. The problem that we are trying to solve is an object detection problem. We will describe the history of object detection and what algorithms are currently state of art.

As was described overall object detection problem consists of two tasks, regression and classification ones. Both image recognition and object localization of the object. So the targeted object should be distinguished form the background and other objects and assigned a correct categorical *label* and a bounding box that localize the prediction. Usually the bounding box is specified a set of coordinates for COCO dataset top left x position,top left y position, width, and height or for Pascal-VOC: bbox is x_{min} top left, y_{min} top left, x_{max} bottom right, y_{max} bottom right. And it is a regression problem. Given N number of labeled images with $x_1, x_2, ..., x_N$, their labels are in the form of $y_i = (label_1, bbox_1), (label_2, bbox_2), ..., (label_i, bbox_i)$ where the label is specified class and bbox will give coordinates of the bounding box, depending on the input formal (either PASCAL-VOC or COCO).

Different losses are used such as softmax, focal loss. The loss should give a measure of how different the distance between the expected and predicted bounding box is and whether the label was given correctly. For the evaluation metrics Intersection-over-union (IoU) applied objects and their predictions is used. In the below formula Eq. 3, b_{gt} stands for ground truth (gt) bbox or mask. The special threshold denoted as Ω is used to measure how bounding box matches true one. For that *Area of Intersection* is divided by area of *union*. Researchers usually use 0.5 for threshold [12].

$$IoU(b_{pred}, b_{gt}) = \frac{Area(b_{pred} \cap b_{gt})}{Area(b_{pred} \cup b_{gt})} \tag{3}$$

So in general prediction is considered positive if both label and ground truth box meet the criteria for IoU. The common benchmark dataset used is Microsoft Common

Objects (MS-COCO) test-dev set [7]. And MS-COCO's AP which is average over 10 IoU levels is commonly used metrics. AP is calculated using formulas for *Precision* = p (Eq. 2) and *Recall* = r (Eq. 1).

$$AP = \frac{1}{101} \sum_{r \in 0.0,\ldots,1.0} p_{interp}(r) \tag{4}$$

where

$$p_{interp}(r) = \max_{\tilde{r}:\tilde{r} \geq r} p(\tilde{r}) \tag{5}$$

Other metric used is mAP, which was used before for object detection tasks on PASCAL-VOC 2007 as MS-COCO appeared in 2014. Additionally for accessing the performance, inference speed measured in Frame per Second (FPS) is used. That is crucial for real-time object detection which should reach 20 FPS.

Object detection can be divided into two main categories: two stage detectors and one stage. Most common methods for two stage detectors are region proposal-based R-CNN, spatial pyramid pooling SPP, Fast R-CNN, Faster R-CNN, region-based fully convolutional network R-FCN, feature pyramid networks FPN and Mask R-CNN. One stage detectors are MultiBox, AttentionNet, YOLO (currently 5 versions), Single Shot MultiBox (SSD), YOLOv2, deconvolutional single shot detector (DSSD), deeply supervised object detectors (DSOD) [47].

2.5.1 Two-stage object detection algorithm

This method uses two stages to perform object detection. Set of proposals is generated, which is identified by algorithm as regions that can potentially contain objects. And in the second stage prediction is made on these proposals. In addition algorithm can change the proposed region or refine generator for proposal regions. After long usage of traditional methods for object detection there were no longer any drastic improvements but on contrary only increase in complexity. SegDPM [45] was the leader in object detection with 40.4% on Pascal VOC2010, before R-CNN came.

R-CNN R-CNN was designed by Girshick et. al. [44]. It improved mAP to 53.3% [47] on PASCAL VOC 2012. The idea was to divide the region into small areas, about 2000 for each image as in Fig. 6.

That is done using selective search [46]. However other region generation methods are utilized. Selective search is able to reject obvious background regions. Each left region is transformed by cropping and warping it into a fixed size region. After a 4095 dimensional feature vector is extracted by CNN based Deep Feature Extraction method. At this stage we get robust, high-level and semantic features for every proposed region [47]. After getting the regions, we train one binary SVM for each class independently. So for N classes, N binary SVM processes on each region. The localization errors are reduced by training a regression model to correct the predicted bounding box coordinates with the help of CNN features. Having predicted box coordinates as a center point, width and height $p = (p_x, p_y, p_w, p_h)$ and ground truth

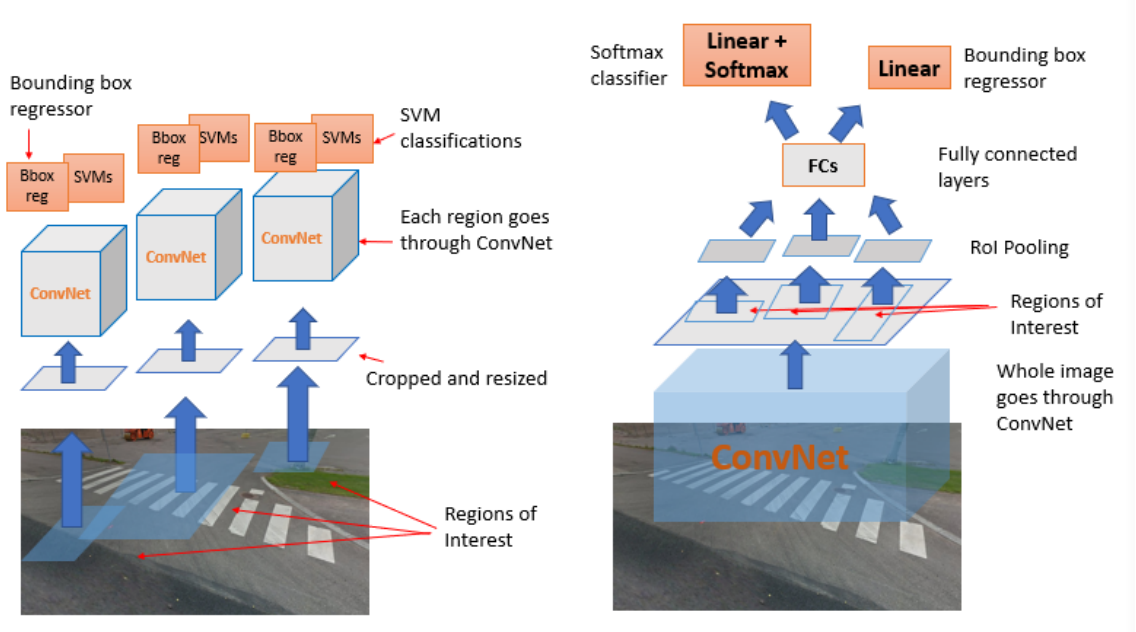

Figure 6: Stages of R-CNN (on the left) and Fast R-CNN

box coordinates $g = (g_x, g_y, g_w, g_h)$, the regressor learns scale-invariant transformations between center coordinates and log-scale between width and height. Then minimizing the Sum of Squared Errors (SSE):

$$L_{reg} = \sum((t_i - d_i(\mathbf{p}))^2 + \lambda\|\mathbf{w}\|) \tag{6}$$

where $d_i(p)$ is a box correction function, as an example, where $\hat{g}$ is a predicted bounding box:

$\hat{g}_x = p_w d_x(p) + p_x$
$\hat{g}_w = p_w exp(d_w(p))$
and
$t_x = (g_x - p_x)/p_w$
$t_w = log(g_w/p_w)$

Another technique used in RCNN and other detection techniques is non maximum suppression. Greedy non maximum suppression avoids repetitive detection of the same object. Greedily selecting the best boxes with highest confidence score and if there are several boxes left, removal of those that have big overlap (i.e. IoU > 0.5) with the current best box is performed and repeat again.

The limitations for the RCNN are the speed, as all the steps are applied for each picture, 3 separate models CNN, SVM, regression which do not share computation. Whole process requires in average 47 seconds for each test image [48]. Especially for the deep networks as VGG16. Storage utilized for saving feature vectors is relatively high. Selective search does not eliminate all redundant regions.

SPP-Net The inputs to CNN in RCNN are fixed sized. That leads to potential loss of information and distortions, lowering the performance value as some meaningful

features of the objects can get geometrical changes. This problem was tackled with the introduction of SPP-net [49] which is based on the spatial pyramid matching (SPM) theory [47]. The algorithm reached mAP 59.2% on VOC-2007 [48]. In general getting a fixed sized output should only be done before fully connected layers, which are located deep in the CNN. SPP-net uses a new kind of layer - Spatial Pyramid Pooling layer (SPP) that is situated after the convolutional layer and right before the fully connected ones. Using SSP-net, the feature maps are computed once from the entire image.

Spatial Pyramid Pooling partitions the feature map into different sized grids and then extracts fixed length representations from feature map. Still this methods needs memory to save extracted features which also means it is harder to optimize the algorithm (cannot use fine-tuning algorithm) as parameters are frozen before SPP layer. This lowers the performance for deep networks. However SPP-Net is still faster 24-102x than R-CNN on test images [49]. SPP-net still is a multi stage approach and speed is still an issue. In Fig. 7 a) featurized image pyramids are seen which help with improving scale invariance [47].

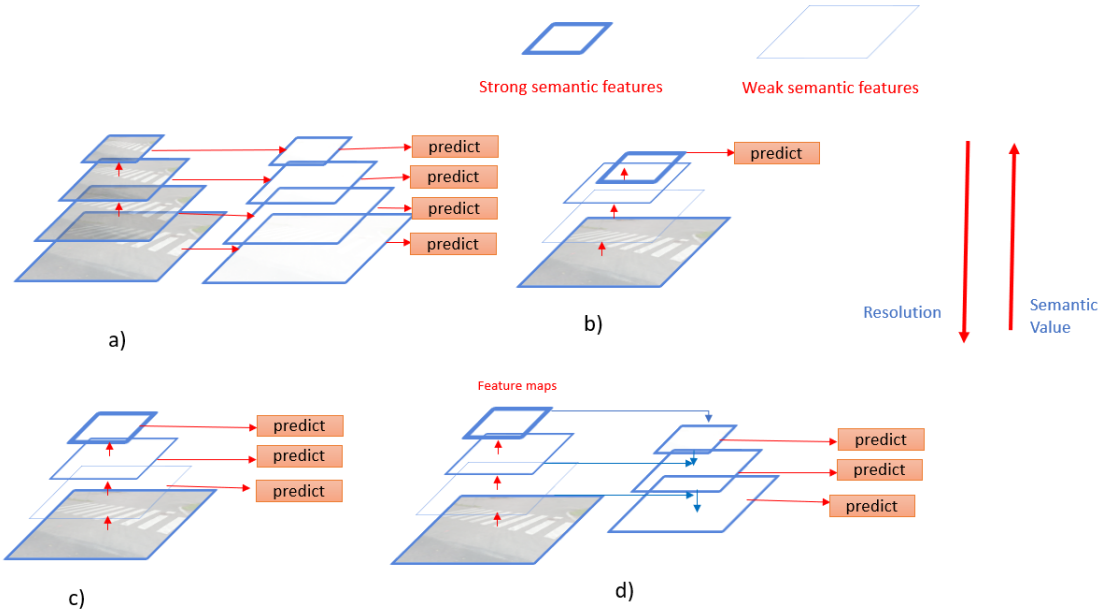

Figure 7: Feature pyramids. a) Featurized image pyramids b) Only last layer (strong semantic features) used to derive output c) All feature maps are considered for prediction d) Combining b) and c) low-resolution and semantically strong features (deep layers) with semantically weak features (shallow layers).

Fast R-CNN In 2015, some improvements were made for R-CNN. The three separate stages extraction, region classification and bounding box regression are now joint into one algorithm and trained (see Fig. 6). Everything is now suitable for end to end optimization, avoiding extra step for storing the feature vectors. That gives advantage in speed over previous algorithms. However the Regions of Interest (RoIs)

are generated using proposal methods as Selective Search and Edge Boxes. These methods utilize traditional approaches based on low level features and accuracy of them affects the final output. This step slows down the overall algorithm.

CNN feature vectors are not longer extracted from separate regions, but as in SPP-Net, convolution map computed once. The difference between SPP-net that there is a specific layer called RoI (Region of Interest) pooling layer. It is the same as SPP's spatial pyramid pooling layer but with one pyramid level. Having that makes training easier compared to SPP-Net as computing derivative during backpropagation is not as difficult now. This enables fine tuning of the convolutional layers. RoI pooling layer is used to output fixed size vector. After is sequence of fully connected layers which are followed by two output layers. One for outputting softmax probabilities for N classes + 1 class for background. And second layer for refining bounding box coordinates.

The method showed 70% mAP accuracy on the VOC-2007 dataset [48].

Faster R-CNN To eliminate the need of using separate region proposal algorithms, Region Proposal Network (RPN) was introduced, which is a fully convolutional network. The new object detection method using RPN was called Faster R-CNN [50]. First image passed to the ConvNet and feature map is returned. On the obtained feature map, RPN slides with a n by n spatial window over feature map generates for each window (center of the window) it outputs multiple anchor boxes with different size and shape (usually 3 sizes and 3 scales are used). Anchor is combination of centre of sliding window, scale and ratio. For each anchor, RPN predicts if the anchor is an object or not and predicts bounding box (see Fig. 8). That way it generates proposals. That proposal are adjusted to one equal size by a RoI pooling layer. In the final step proposals are passed to a fully connected layer with two outputs, regression and classification ones. That gives opportunity to optimize end to end training of the whole algorithm.

It takes 0.2 seconds for each image, which is drastic improvements compared to previous algorithms and makes it almost real-time detection. The limitations of the methods is that each feature vector still need to go separately through a sequence of the fully connected layers. It is also difficult to detect small objects as the method uses a single deep layer feature map for prediction. That makes detection of objects at different scales a complex task [12]. Despite limitations, Faster R-CNN achieved mAP = 73.2 % on PASCAL VOC 2007 [50]. Faster R-CNN with Inception ResNet as backbone and 300 proposals give highest accuracy [12].

R-FCN Continuing the tendency to increase speed by sharing computation Region-based Fully Convolutional Net (R-FCN) by Dai et. al. was introduced [51]. Faster RCNN network consists of two subnetworks and computation is not shared in region classification step and still each vector separately inputted in fully connected layers. That slows down the algorithm especially with images that have a lot of proposals [12]. R-FCN solves that problem. The last convolutional layer of R-FCN outputs for each class position-sensitive maps. The position sensitive score map encodes information of one category about it's relative position. That score maps are able to recognize the

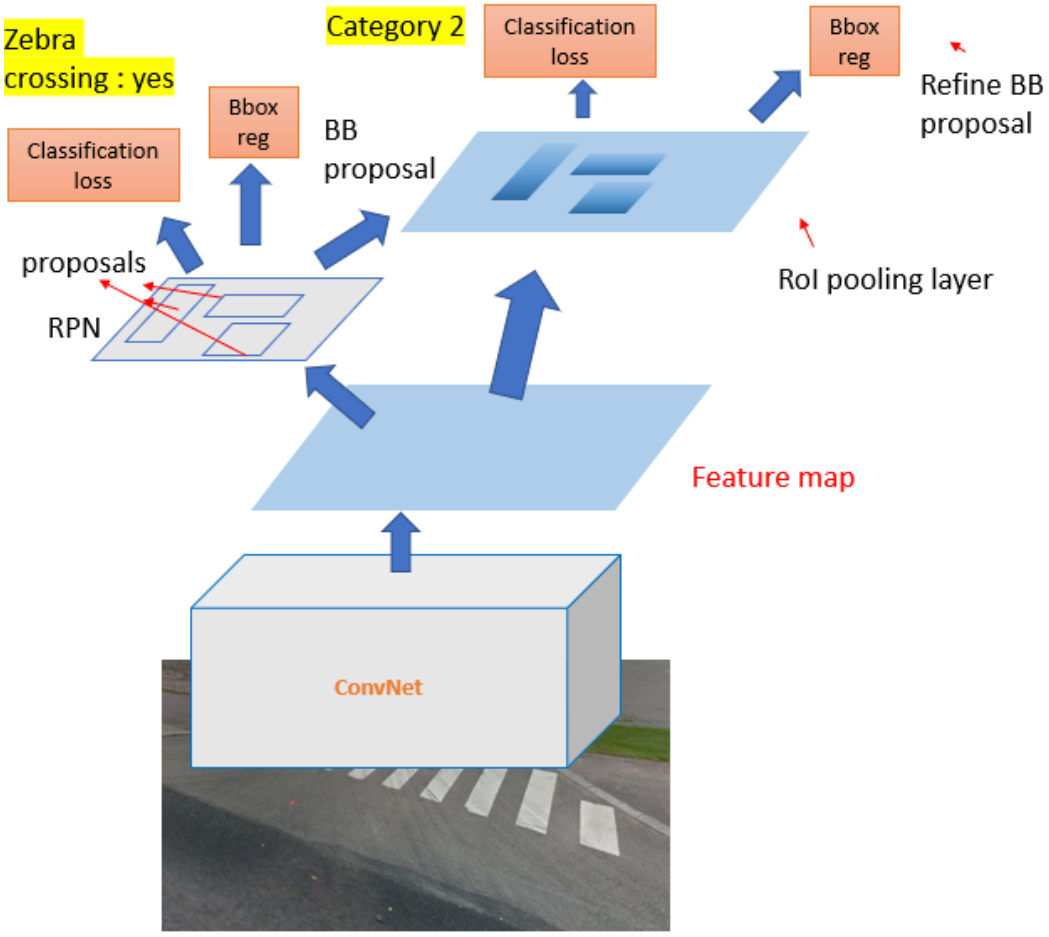

Figure 8: Stages of Faster RCNN

particular parts of the target. That is possible with the help of a Position Sensitive ROI Pooling layer (PSROI Pooling) which can extract spatial-aware region features. In other words R-FCN takes one regions proposal scans, divides it into sub regions. Then goes through each of that sub regions, making sure that the sub region is a part of the target object. If enough sub region will be considered as separate parts of the whole object that that whole region is classified as the target class. That way also the problem of the location variance and invariance are solved. Location variance is needed when the exact location of object is needed and location invariance, regardless where is the object, still classify as object is found. R-FCN is faster that Faster R-CNN however for the mAP of Faster R-CNN and R-FCN, both with ResNet101 backbone then latter has 83.6% while Faster R-CNN has 83.8%. And time per image is 0.03 and 2.24 seconds. With such a drastic save of time and almost same accuracy R-FCNN is favourable

FPN Feature Pyramid Networks were implemented by Lin et.al. [56]. The object detectors before FPN used a single deep layer feature map to classify (see Fig. 7 b)) which lose weak semantic and high resolution features. Deep Convolutional Networks

have difficulties with feature representation as the features that are situated deeply in the layers are strong in their semantic representation however they are spatially weak and that is reverse for features in the shallow layers. That is why there was a need to combine both features, from deep and shallow layers so that strong spatial of shallow layers are joint with powerful semantic features from deep layers [12] and so high level semantics is constructed on all scales (see Fig. 7 d)). That is beneficial for small objects as spatial information is weak for them. If pyramid of images as in Fig. 7 a) is used in order to solve the above mentioned problem then memory and time will be an issue. If instead pyramid of feature maps is used, then that can lead to importance of feature maps that are closer to shallow layers. That feature maps consist of low level features which are not accurate for prediction purposes. If feature maps of different spatial resolutions are used separately for prediction purposes as in Fig. 7 c) then there is a possibility of semantics gaps due to a variation of feature maps at different depths [47]. The FPN produces multi-scale feature maps (see Fig. 7 d). The architecture of FPN is composed of a bottom-up pathway which produces feature maps at different scale, with smallest spatial resolution at the top. That is done with a basic CNN based backbone. Then there is a Top Down pathway, where higher resolution layers are built from a layer with strong semantic features. The lateral connection are added between these pathways to maintain the location information. FPN should be fed into some object detector, as by itself it is just a feature extractor. FPN in combination with RPN improves average recall AR by 8 points on COCO (comparing with RPN baseline). And 12.9 point for small objects [56]. The result for mAP for Faster RCNN on COCO is 42.7% and for FPN+ Faster R-CNN is 59.1%.

2.5.2 One-stage object detection algorithms

As the name suggest they have only one stage, dropping a proposal generation step. They consider all regions as relevant ones. They are mostly lightweight and outperforming two stage algorithms in speed. That is crucial for real-time applications. The first such detector was developed by Sermanet et. al. and is called Overfeat [53]. The architecture is similar to AlexNet, however the top classifier layers are replaced by a regression network to predict bounding boxes.

YOLO The above mentioned object detection algorithm were first partitioning the image into the parts and then only performed localization. However Redmon et. al. designed YOLO that makes the use of the whole image at once [54]. It divides image into a fixed grid of cells. Then each cell acts as a proposal for detecting if the object is there or not. The first implementations assumed that these cells contain one or two center of the object. Then each cell undergoes prediction of having only one category of object (Fig. 9) then certain number (specified by B) of bounding boxes, their corresponding confidence scores, defined as $Pr(Object) * IOU_{pred}^{truth}$.

This confidence score shows how likely that there is an object and how accurate is this prediction. And further, not taking into account number of boxes (B), cell also predicts conditional class probability for each C classes, so the probabilities of this

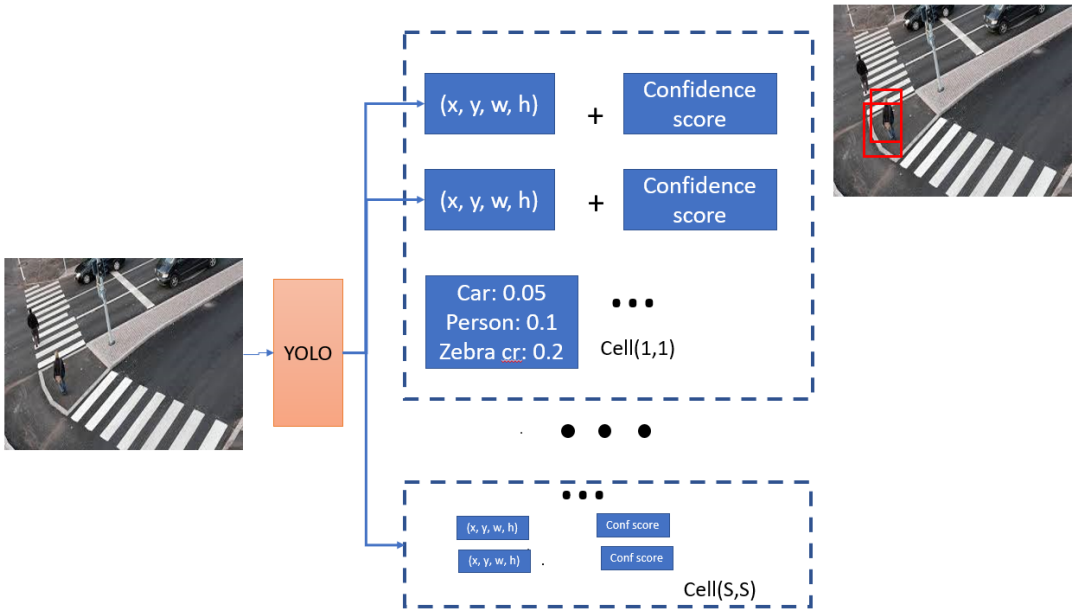

Figure 9: YOLO with prediction of boundary boxes

object belonging to each class. $Pr(Class_i|Object))$ but only for cells that contain object. Then the class-specific confidence scores for each box are calculated by taking predictions of individual box confidence scores and conditional class probabilities.

$$Pr(Object) * IOU_{pred}^{truth} * Pr(Class_i|Object)) = Pr(Class_i) * IOU_{pred}^{truth} \tag{7}$$

This equation measures the confidence of the algorithm on classification and regression.

YOLO outputs multiple prediction boxes. In general if grid was S by S, and there were C number of classes then the YOLO gives = **(S, S, B * 5 + C)**

YOLO then selects right boxes that are the most responsible for the object. That box has biggest IoU with the ground truth. The loss function is comprised of 3 parts: classification loss 9, localization loss 8 and confidence loss 10.

$$\begin{aligned} &\lambda_{coord} \sum_{i=o}^{S^2} \sum_{j=o}^{B} \mathbb{1}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\ &+\lambda_{coord} \sum_{i=o}^{S^2} \sum_{j=o}^{B} \mathbb{1}_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \end{aligned} \tag{8}$$

$$\begin{aligned} &\lambda_{noobj} \sum_{i=o}^{S^2} \sum_{j=o}^{B} \mathbb{1}_{ij}^{noobj} (C_i - \hat{C}_i)^2 + \\ &\sum_{i=o}^{S^2} \sum_{j=o}^{B} \mathbb{1}_{ij}^{obj} (p_i(c) - \hat{p}_i(c))^2 \end{aligned} \tag{9}$$

$$\sum_{i=o}^{S^2}\sum_{j=o}^{B}\mathbb{1}_{ij}^{obj}(C_i-\hat{C}_i)^2 \tag{10}$$

where x_i and y_i denote the center of the boxes w_i and h_i stand for the weight and height normalized relative to the image size. The sum of squared error is estimated only when object is present. C_i shows the confidence scores for each class and to show the existence of the object, indicator function is used $\mathbb{1}_{ij}^{obj}$. Then if object is present, $\hat{p}_i(c)$ represents for cell i,class c, the corresponding conditional class probability. The λ_{coord} introduced is emphasis on the boundary box accuracy. For boxes that do not have object λ_{noobj} (usually default is 0.5) pushes the confidence score to decrease. That is helpful as usually the image will have less boxes that contain objects and that causes class imbalance problem. And with λ_{noobj} makes the loss due to non-objects smaller. Final **loss function** in the sum of all three losses.

In order to tackle the duplicate predictions for same object, **Non-maximal suppression** is used. It sorts the predictions according to their confidence scores and takes top predictions, compare them within same class. If the same class predictions IoU is bigger than 0.5, then it is ignored. Then repeat again with top predictions. Usually it adds 2-3% in mAP [54].

You Only look once (YOLO) a real-time detector YOLO's speed varies from 45 FPS and accuracy mAP = 63.4% to Fast YOLO with 155 FPS and 52.7% on VOC-2007 dataset depending on the backbone. Yolo uses only one feature map, the last one, which as was mentioned previously, losses information, especially for objects at multiple scales and ratios and for unusual and new aspect rations. As it only can detect up to two objects, the usefulness declined for tasks with multiple object detection [12]. Localisation is less accurate compared to two stage detectors. However YOLO's performance is good when it learns general representations of the object, for example generalizing from natural images or artwork. In addition False Positive rate is low for YOLO [54]. All the drawbacks were addressed with YOLO's versions, currently there is YOLO v5.

YOLOv2 The difference between YOLOv1 is the addition of a batch normalization in convolution layers which eliminates the necessity of dropouts. In general it improved mAP till 2%. It was improvements to first version and was designed by Redmon et. al. [57]. The methods use a more powerful deep convolutional backbone architecture.

YOLO prediction on the shape and size of the boundary box is more random. In early stages of training shapes, model is struggling to pick up right sizes. However the real life shapes are commonly of one ratio, for example for pedestrians it is 0.41. That was helps to initialize the initial guesses with available knowledge from real world. This initial guess box is called an anchor and usually the offsets for each anchor box are predicted instead of generating random bounding boxes. The offsets are constrained which enables the diversity for the different shapes. After training a CNN classifier network, the fully connected layers (used for boundary box outputs) are replaced with a convolutional layer to predict boundary box. That way the category prediction is now implemented not on the cell level but on the boundary

box level. There are now 4 parameters for dimensions (center, width and height) of boundary box, 1 box confidence score and C class probabilities.

In order to identify the common bounding box patterns, K-means clustering algorithm to find K top bounding boxes is used. Instead of measuring regular spatial distance, IoU is used. The offsets are also constrained to eliminate randomness. YOLOv2 uses sigma function to constraint the offset range. That additions increases mAP in 5% [57]. Another new improvements in YOLOv2 are multi scale training and pass through. In latter method, the input is reshaped and then again concatenated with the input. Thus learned weights are more robust for detecting fine-grained patterns. And convolution filters on these new concatenated input applied. But as the layers downsample the input, there is subsequent loss of fine-grained features. Multi Scale training acts as a data augmentation technique, by randomly selecting after specified number of batches, other sizes for training. That makes model robust for various scales and sizes. YOLOv2 is able to take different resolution images and thus balancing between speed and accuracy. With low resolution YOLOv2 has enough accuracy for fair detection (see Fig. 10). When resolution is 288 by 288, and running at 91 FPS, YOLOv2 shows 69.0 mAP similar to Fast R-CNN (70 mAP and 0.5 FPS). Higher resolutions shows improvements in mAP and especially at 544 by 544, mAP for YOLOv2 is 78.6 with 40 FPS [57]

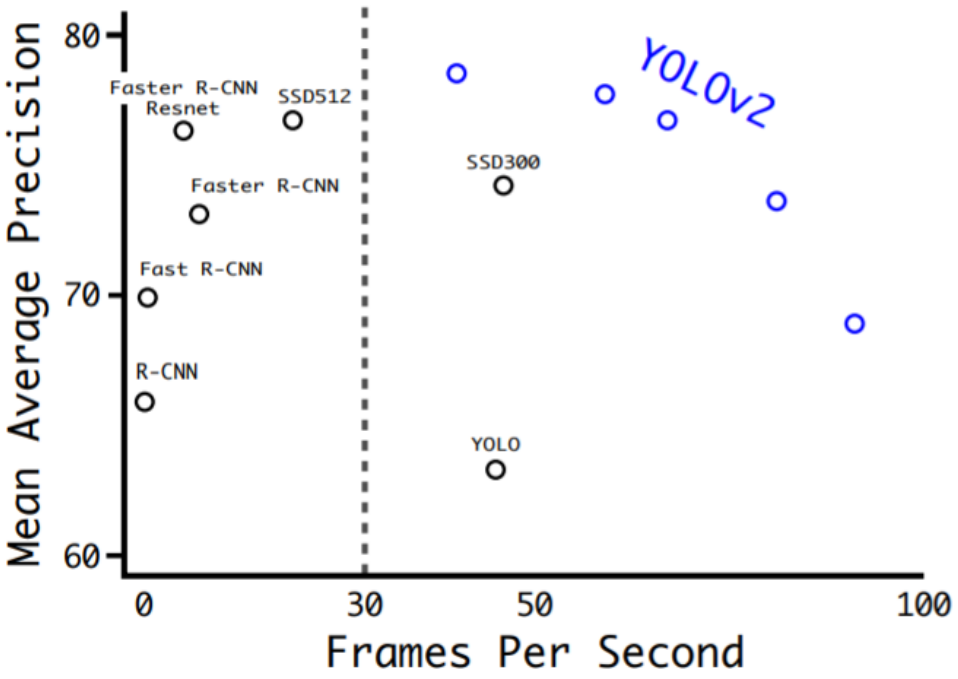

Figure 10: mAP and speed on VOC-2007 [57]

YOLOv3 One of the most important feature in YOLOv3 is a new underlying architecture network, Darknet-53 that has 53 layers trained on ImageNet. The

YOLOv3 network incorporates Darknet-53 and as the object detector another 53 layers, so giving a network of 106.

Darknet-53 mainly consists of 3 by 3 and 1 by 1 filters with skip connections. It has less billion floating point operations than ResNet-152 however does not lose performance accuracy while maintaining 2x speed. Essential elements for Darknet are Residual Blocks, Skip Connections ad Up-Sampling. The CNN layers use Batch Normalisation and Leaky ReLu.

YOLOv3 improved much in detecting small objects by preventing the loss of low level features. It does not use pooling layers, only convolutional. The difference of YOLOv3 is that it now considers that there are non exclusive classes like *Person* and *Girl*, where the sum of output probabilities can be greater than 1. To achieve it, YOLOv3 changes the softmax function to independent logistic classifiers. They compute the likeliness of assigning a specific label to an input. As the loss function, YOLOv3 utilizes binary cross-entropy loss for every category.

The other novelty is that only one boundary box prior is is associated for each ground truth object. Corresponding objectness score (how likely each candidate box may contain an object) is 1 only for the greatest overlap, others are assigned to zero.

Also it uses three different scales from which to extract features.

SSD SingleShot Multibox Detector (SSD) proposed in 2016 by Liu [55]. SSD tries to address the problems of YOLO.

SSD partitions as YOLO, the image into a grid of cells. The difference comes with the introduction of the set of anchors that are used instead of fixed grid of cells (that anchors were already explained in YOLOv2). These anchors are of several scales and ratios and assist in predicting bounding boxes. Each anchor has corresponding anchors with 4 value offsets. Offset differences are learnt by the regressors. Moreover it classifiers output **C+1** class probabilities [12].

One of the important feature of SSD is that multiple feature maps were used to predict an object. Some convolutional feature maps were added to backbone to improve detection of large objects. The loss is a weighted sum of localization loss and classification loss over all prediction maps. The final prediction is joined result that is obtained from different feature maps. Hard negative mining is added to avoid domination of negative proposals during training.

Data augmentation applied to SSD showed improvements in detection accuracy. SSD reaches Faster RCNN performance (see Fig. 10) with higher FPS. SSD with input size 500 by 500 is called SSD500 achieves on VOC-2007 mAP 76.9% at 22 FPS, outperforming Faster R-CNN that has mAP of 73.2% at 7 FPS only. The limitations of SSD decreasing accuracy with increased speed. The bigger number of default boxes seem to increase detection performance. Compared to RCNN, YOLO has lower localization error for detecting similar classes. That is in reverse for classification errors. SSD still presents a problem of a class imbalance. When the background classes outnumbers foreground, it creates many negative samples that negatively affect training accuracy.

Retina Net RetinaNet was designed by Lin et. al. [56]. Retina Net uses Feature Pyramid Network (FPN) introduced before 2.5.1 and Focal loss. RetinaNet allows to deal with images which objects located densely and in case of large number of background classes compared to foreground.

Feature maps are produced sequentially in a pyramidal shape (see. Fig. 7), due to applied pooling. However as was also mentioned, there is possibility that information is lost in between layers of features, i. e large semantic gaps that are produced. And that was approached with FPN (see Fig. 11 b). Feature Pyramid Network advantages have been discussed in section FPN (2.5.1). The novelty of this algorithm is incorporation of a focal loss. With that, RetinaNet approaches the class foreground-background imbalance problem presented in SSD using the focal loss. Before Retina Net, the class imbalance problem was the main concern that one-stage detector will not be able to reach the performance of the two-stage detectors.

Structure of Retina Net In the (see Fig. 11 a.) as the backbone, usually ResNet50 or ResNet101 is used, however the choice is not limited to only those two. In the Fig. 11 b. FPN that outputs class and box subnets. Next as in Fig. 11 c and d, further subnetwork for object classification and subnetwork for object regression are used to predict final output. Class subnet outputs probability if the object is present for each of anchors and object classes. It applies FL at the loss function. Box regression Subnet is similar to class subnet however there is no sharing of parameters. It predicts the location of the object relative to anchor. As the loss function it uses *smooth_l1_sigma*.

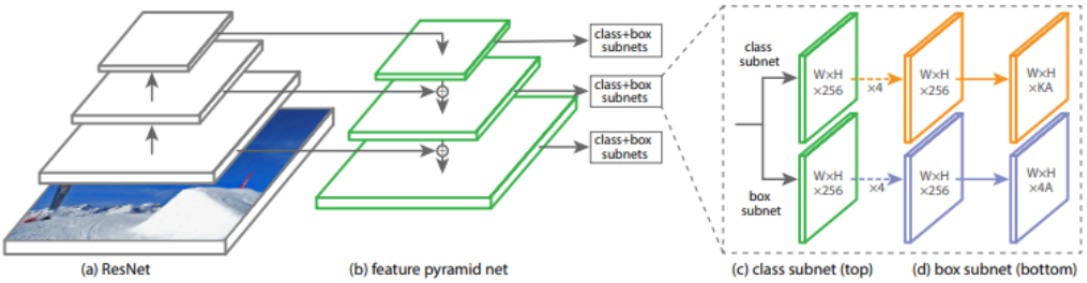

Figure 11: Retina Net architecture [56]

Focal loss During training RetinaNet uses focal loss which emphasize misclassified examples. Instead of simply discarding easy negative examples, their contribution to the loss is reduced. Considering Cross Entropy (CE) Loss

$$CE(p_t) = -log(p_t) \tag{11}$$

In case of large imbalance problem, as the example if ratio of easy/hard examples is 100.000/100. The loss for each easy example is 0.1 and each hard is 2.3. The total

loss from easy, 10000 still bigger than from hard, which is only 230. To tackle this, α balance CE Loss (Eq. 12) is introduced.

$$CE(p_t) = -\alpha_t log(p_t) \tag{12}$$

Adding a weighting factor α for positive class, for negative (1-α). The weighting factor can also be the inverse frequency.

However with the introduction of the focal loss (FL) Eq. 13

$$FL(p_t) = -(1 - p_t)^{\gamma} log(p_t) \tag{13}$$

now the weights for easy examples are reduced.

$-(1 - p_t)^{\gamma}$ lies between 0 and 5.

We take y = 0 for background and y = 1 for foreground.

When y = 1, p_t = probability, otherwise it is (1-probability).

Considering the following cases, we will see how the focal loss changes:

First case:if we have easy foreground object with probability = 0.9, then

CE(foreground) = -log(0.9) = 0.1053

For easy classified background with p = 0.1

CE(background) = -log(1–0.1) = 0.1053 Calculating FL for both cases with Eq. 13 and α=0.25 and $\gamma = 2$ gives the same value for both, 0.00026, which is a very small number.

In the second case, the same FL will be for misclassified foreground object with p=0.1 and background object with p=0.9, α, γ, but this time 0.4667.

From that it can be concluded that FL adds huge weight for misclassified and on the other hand ignores the well classified ones. In general α=0.25 and $\gamma = 2$ values are well used and it is advisable to decrease α and γ on the contrary increased.

The focal loss outperformed naive hard negative mining strategy. During training RetinaNet considers about 100 thousand anchors, compared to SSD that ranges from 8 to 26 thousand. And the FL is calculated as the sum of the FL throughout all 100 thousand anchors. Then it is normalized by the amount of anchors which were assigned to a ground-truth. RetinaNet achieved mAP 40.8 with backbone ResNeXt-101-FPN on MS-COCO [56].

YOLO4 YOLO4 was introduced by Alexey Bochkovskiy in April 2020 [59]. It addresses the problems of a long training by proposing fast, smaller mini-batch training on a single GPU. On the MS COCO dataset it reached 43.5% mAP, at 65 FPS. The authors of the algorithm claim that they introduced universal features that work well for all computer vision problems. They are Weighted-Residual-Connections (WRC), Cross-Stage-Partial-connections (CSP), Cross mini-Batch Normalization (CmBN), Self-adversarial-training (SAT), Mish-activation, Mosaic data augmentation, DropBlock regularization, and CIoU loss. The authors divide the methods in two categories: bag of freebies (BoF) and Bag of specials (BoS). Both methods contribute for the better accuracy without increasing the inference cost, but affect training cost. They are data augmentation (more information in 2.5.3), cutOut, randomly masking

some parts of the image. DropBlock [60] tries not to randomly drop activations as the valuable semantic information can be lost between nearby pixels. However, it drops them in a continuous manner, for example whole head or whole door, so still leaving without semantic information but in a way that network will be able to learn what to learn other features. The next improvement was the loss function. Authors assumed that simple mean squared error considers that points are not related to each other and so does not take into account that possible relationships. That can be solved with the GIoU loss. Another improvement, CIoU loss includes the shape and orientation of the object, considers the center and the aspect ratio. YOLO4 uses CIoU as the loss function for the bounding boxes. BoS has post-processing methods, that increase the inference cost but contribute a lot in the performance of the algorithm in a positive way. The attention mechanism is one of such improvements. Fundamental idea of a channel attention is to emphasize the feature maps that have higher importance. For example feature map that contain the edges are more important than those that contain the background. Squeeze-Excitation Networks do it by scaling specific important channel by a higher value [61].

In general attention mechanisms try to emphasize the regions that we are most interesting. YOLO4 uses modified Spatial Attention Module that is capable of increasing accuracy by 0.5%, and losing only 0.1% in calculations [59]. The CSP [64] helps to deal with duplicate gradient problems that common for huge ConvNet backbones. The usage of CSP gives less parameters.

As the activation function YOLO4 uses Mish in the backbone [62]. YOLOv4 uses SPP and modified PAN that concatenates, instead of addition. The new data augmentation technique that was proposed by the author of YOLO5 Glenn Jocher, Mosaic was used in the model. Mosaic combines four images into one in certain ratios. That reduces the size of mini batch used. Moreover different scales combined in one, can help in identifying objects of smaller scales than usual [65]. The next novelty is the use of Self-Adversarial Training (SAT). This technique applies modifications on the image and attacks itself by fooling that there is no object. Then in the next stage, network tries to find object without fooling itself.

EfficientDet Was designed by Google Research, Brain Team [63]. EfficientDet is based on a new bi-directional feature network that more efficient than just one directional pyramid network. This feature allows to perform multiscale feature fusion. Second advantage is a compound scaling that allows to scale uniformly dimensions for all backbones. Using that advantages, EfficientDet reached state of the art, 55.1 AP on COCO dataset (see Fig. 12). Flops used in this picture indicate floating-point operations, additions and multiplications. Authors used this metrics to access the training time.

YOLO5 Glenn Jocher released YOLOv5 after a short time. There are still discussions of the name of the method, but in this work we will use the name "YOLO5". YOLO5 outperforms EfficientDet COCO AP and FPS (see Fig. 13). However the comparison here is quite ambiguous as authors did not provide the specifications about the training of different models. Such as the batch sizes, and if batch size

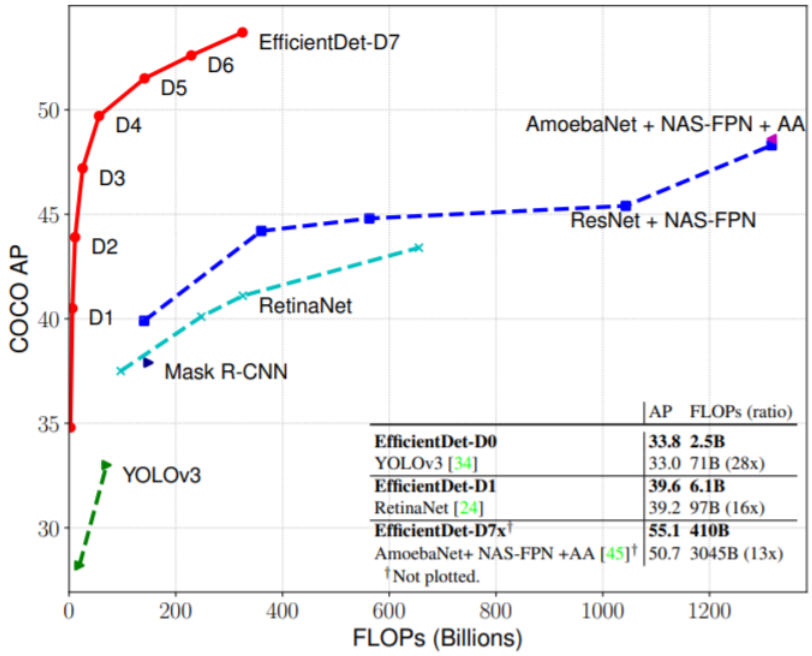

Figure 12: FLOPs vs. AP COCO. All values are for single-model single-scale [63].

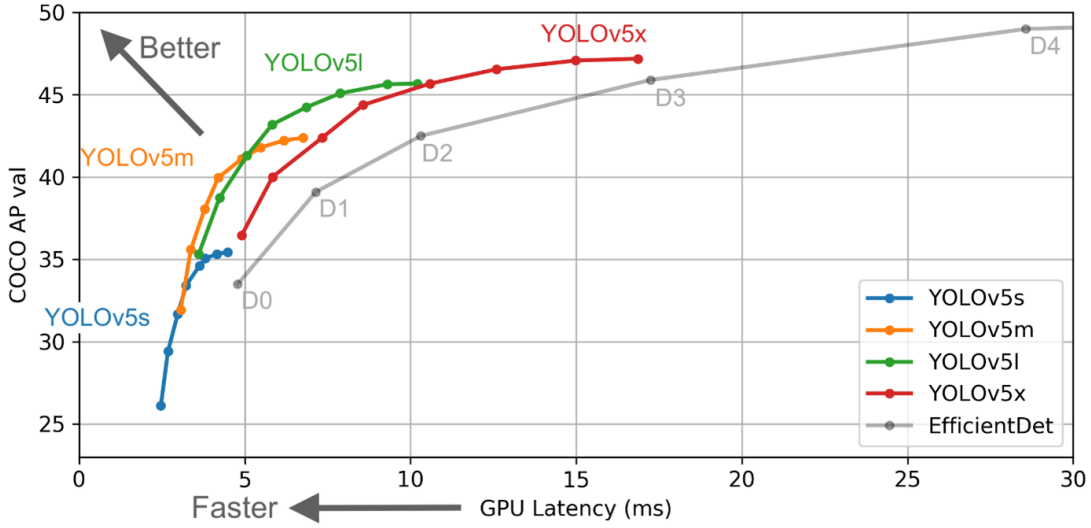

Figure 13: Performance of YOLO5 models and comparison to EfficientDet [65]

is not one, according to the author of YOLO4, they cannot measure the latency [3]. YOLOv5 first from YOLO family was written in PyTorch and not in Darknet research framework, which is mainly in C language and relatively harder to use. Moreover PyTorch allows to use 16 Bit Floating Point Precision (on V100, T4 GPUs) instead of 32 which speeds time for inference [65].

YOLOv5 stores model configurations in yaml file, which specifies the different layers in the network and how many of those layers in total in network. The backbone

for YOLO5 is same as for YOLO4 with CSP bottleneck to decrease parameters. PA-NET neck for feature aggregation as it preserves spatial information.

Features of YOLO5 For each training batch, YOLO5 performs online data augmentation: scaling, color space adjustments, and mosaic augmentation (introduced in YOLO4 2.5.2). YOLO5 attempts to learn automatically bounding box anchors. As author mentioned, real data sometimes does not fit the COCO datasets, there are cases when bounding box are extremes. For example for giraffe, which will have very tall bounding box. Or manta rays with flat and wide boxes [65]. The anchors are predicted using kmeans and evolutionary algorithms based on their distribution in the dataset.

2.5.3 Comparison between different algorithms

The following are the standard metrics used for evaluating the performance of general object detection algorithms: average precision (AP), mean average precision (mAP), speed (FPS), flops, true positive, false positive, IOU threshold, recall, and confidence threshold. Mean average precision (mAP) is widely used as a performance evaluation metric for object The comparison between object detectors presented above is not a trivial task. Most importantly it depends on the target that is aimed with the detection algorithm, like speed or accuracy or balance between them. However the characteristics of the algorithm are also affected by the choice of features that construct or are part of the object detector. These are:

- Backbone choice (VGG16, ResNet, Inception, etc.)
- Image input resolution
- Data augmentation
- transfer learning
- Output strides for the extractor,
- IoU thresholds, non maximum suppression,
- Choice of anchors
- Hard mining ration
- Feature extractor
- Loss function for regression and for classification
- Deep learning software platform
- Training specifications: batch size, learning rate and etc. ...

Nowadays object detectors have advanced and contain a lot of features and use different techniques, which makes the comparison hard. In this work we relied on the performance mAP on famous training dataset such as COCO.

Image Augmentation The deep learning based models learn better from the bigger datasets. However it is not easy to get large manually labeled dataset. Bigger dataset usually provide different possible scenarios. So that features inside dataset increases probability of the model to be robust towards unseen data. Overfitting is tackled as large dataset are less prone to overfit. The class imbalance problem can also be solved by augmenting specific classes. as the risk of having imbalanced dataset is reduced.

2.5.4 Related work

Line detection Li et al. [34] in order to extract the road boundaries implemented a model based on both a CNN with a recurrent neural network (RNN). Without prior knowledge with CNN, geometrical structure of lane is extracted and, further RNN is able to detect it. Another work in done in detection of road lane boundaries is done by He et. al. [35]. The extraction of deep features was done using dual view DVCNN dual view CNN. Two image of the same scene however with different angles of the view, front and top, are inputted. Second image one is obtained using ROI and inverse perspective transform. Disturbances from other objects on the road such as vehicles and shadows are excluded from the front-view image, while the cylindrical and enlarging gradually toward the end structures are still present in the top-view image. But there is a decrease in accuracy when the road marking is covered by other things on the road or when the image is too bright.

Berriel et. al. presented a crosswalk classification system [4] using the the crowd sourcing platforms to train automatically ConvNet. Their work is applied to challenging dataset, Brazil roads, which present additional challenges that are not present in many developed countries. The authors claim that model will be more robust when applied to other country-models as it incorporates wide range of features. The designed solution was based on VGG architecture and consists of 16 layers, 13 convolutional and 3 fully connected, where the last has 2 classes: yes or no for crosswalk. the input images were re scaled by the needs of the model to 256 by 256. Two augmentation techniques are used. With 50% being mirrored horizontally and cropping 224 by 224 pixels randomly. Xavier initialization for network weights. The authors used 0.01 as initial learning rate and decreased it by a factor of 10, 3 times during the whole training. As evaluation metrics they utilized global accuracy and F1 score, moreover they used an instance based metric, which considers correct guesses in a sequence of the images. Their models on average achieved accuracy of 96.3% with 92.78% F1 score and 0.21 as standard deviation, implying that models are robust to noise and to initialization of weights. The common misclassified examples were results of rare type (red or blue background), aging crosswalks, crosswalks not in the traffic direction and situated very far. In addition authors admit that annotation by one human was done which increases the subjectivity and bias.

Another similar work was done on road markings that included arrows, bike markings. The work provides useful insights as it also classifies to categories road markings. The classification is performed only for bike arrows, for example distinguishing between seven types of road markings, or example forward (F), forward-left-right

(FLR). Authors' method was based on the adaptive region of interest (ROI) and deep learning bases feature pyramid network (FPN) that serves simultaneously as the detector and classifier [28]. In order to enhance the performance of the model, authors decided to do pre processing of the image. The previous work which also utilized deep-CNN-based road marking detection but inputting entire image, did show performance improvements. So in in this work, authors decided to include extraction of vanishing point with consequent creation of ROI image. That step removes unnecessary information such as occlusion, illumination. Authors chose to extract vanishing point not with deep a learning based VPGNet, where the annotation should be done manually and computer power should be sufficient enough. But with traditional mage processing using Line Segment Detector (LSD) designed by von Gioi et.al LSD [36]. LSD is used to detect locally straight contours on images. After that the detector and classifier network built on four components is proposed. The one stage Retina Net was favoured. Despite the advantages that were already mentioned in section, Retina Net is outperforms in memory, power and speed. The hyperparameters taken for RetinaNet: 1000 iteration in each epoch, epoch amount is 50 (40 min for each epoch), learning rate initial is 0.0001 with 0.1 reduction. Loss for classification used is a focal loss, for regression smooth L1. Training was done on dataset that includes three different geographical regions under different illumination conditions, weather. The images are taken from the front view. Data augmentation was applied by shifting ±4 pixels and also horizontally flipping. In order to access the performance of the model, two fold cross validation is done, moreover authors tested model on mixed dataset of three regions. According to the results obtained

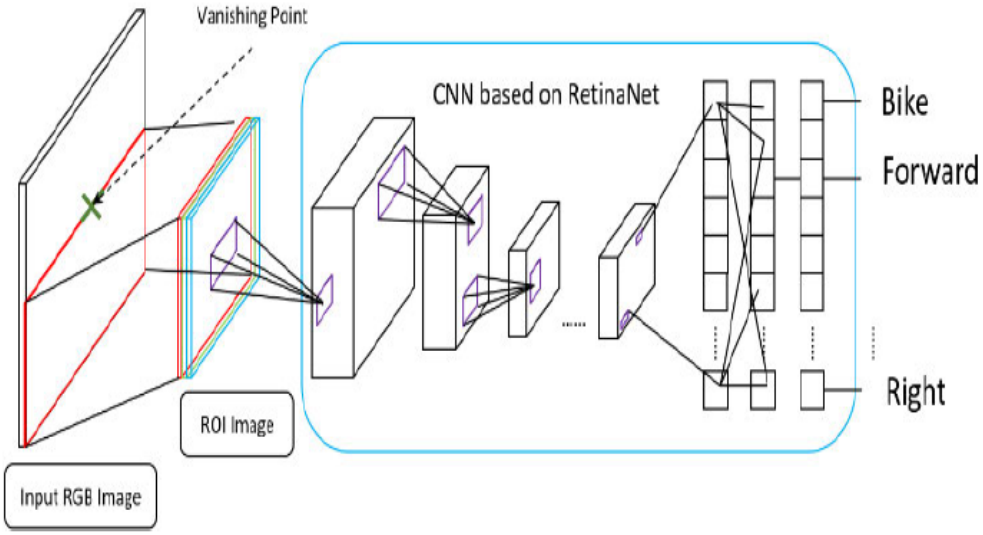

Figure 14: Hoang et. al. proposed model

with training on model with a Resnet-50 backbone and without pre-trained weight accuracy in average was 0.969, with pre trained on ImageNet 0.964. Using the backbone VGG-16 showed the best F! score compared to previous and accuracy

almost same as without pretrained and ResNet-50 backbone 0.968.

With fixed ROI average accuracy drop from 0.994 to 0.834. Comparison between adaptive ROI and IPM shows that accuracy drops from 0.969 to 0.681 (IPM worse performing). One of the most important results of these work is comparison between YOLOv3 that obtained general accuracy of accuracy 0.623 and Faster RCNN with accuracy 0.574. Comparing results of recall and precision, for recall, Faster-RCNN 0.877 outperformed YOLOv3, which had 0.814. In precision YOLOv3 had 0.759 whereas Faster RCNN only 0.636. From that it can be concluded that YOLOv3 is better in precision which is a good measure to determine when costs of False Positive (incorrectly recognizing as road marking) is higher. Recall is better when costs of incorrectly recognizing as False Negative (incorrectly recognizing as background) is higher. The time taken for processing one frame for algorithm by Hoang et. al takes in average 49 ms, for Faster RCNN 428 ms and 44 ms for YOLOv3. Authors revealed that distortion that occur when the image is transformed to a bird's-eye view negatively influences overall accuracy.

2.6 Deployment

Research done during last few decades in deep learning field, revealed enormous possibilities in the field of computer vision. Since 1970, when the image was first put into the digital form and today it is possible to visualize how computer perceives all those pixels. From diagnosis pneumonia from ultra ray images to classification of the quality of potatoes for baby food. The application field currently is vast and more importantly it does not require much of the knowledge in image processing domain. For example, the same company that recognizes the quality of potatoes require only the labeled data of the potatoes and further train them, using the recent robust models. With foundations of the deep learning and machine learning, one is able to use AI frameworks and pretrained models, tune hyperparameters and get sufficient results. This process of taking a trained model and output prediction is called AI serving . That also became possible with advent of computer processes, cloud technologies and special platforms designed to perform Machine learning.

Not many companies can adapt the new recent advances in machine learning. One of the reasons that whole infrastructure sometimes should be changed in order to be able to run that developed algorithm. The latter problems include the ingestion of the data, as for example the current and future algorithm use the data in different format or from different sources. That also implies that transformation phase is affected. Latency. Moreover if the more data will come in, the proposed architecture should be able to train on new data. So the easy maintenance is crucial. However despite all the pitfalls that are encountered on the way of automating machine learning models, the advantages as automating manual labour are far more meaningful. This all became is possible with cloud solutions that provide resources for storing, computing and networking. Moreover the autoscaling based on traffic and workload

In business environment the most work of machine learning is dedicated towards deployment of the model into production. The machine learning process itself takes the 5% of the work. Two most time consuming areas are data ingestion and cleaning

and deployment of the model into production. In some cases to overcome network latency, the Edge AI solution is implemented. The main point is to push AI serving models close to devices.

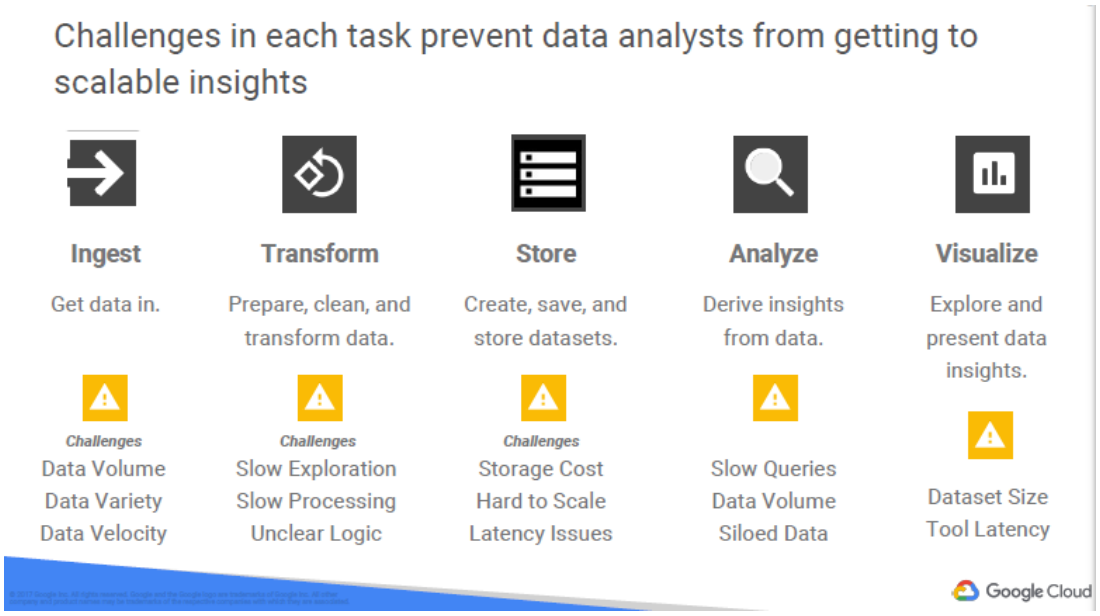

Figure 15: Google Cloud: challenges towards scalable insights.

The road marking classification developed in this paper can be used later on edge devices such as mobile phones or even installed in the monitoring service of Helsinki city center.

One of the possible ways to deploy the model is to use Google Cloud AutoML Vision API (see Fig. 16).

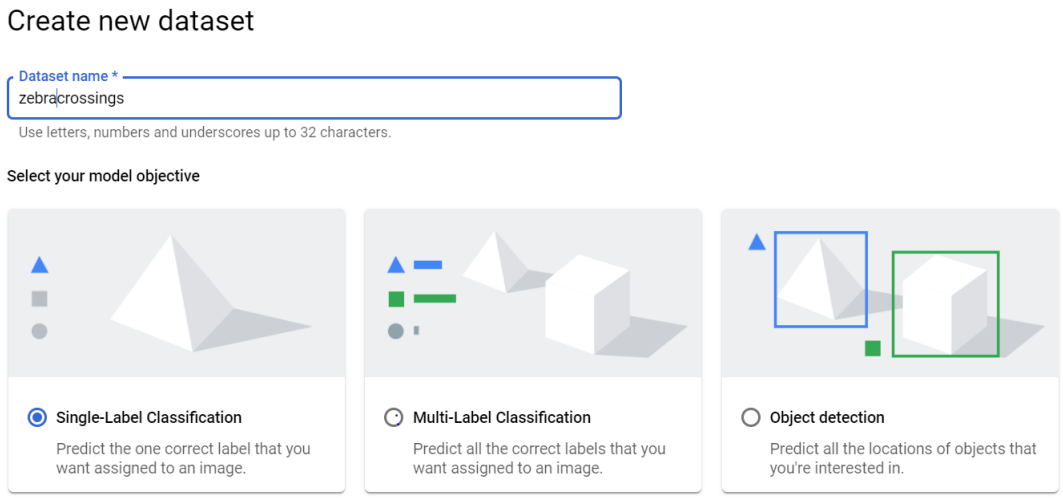

Figure 16: Example of creation model for AutoML Vision

3 Research material: data

In this section experimental part of the thesis is discussed. First, data gathering, preparation of the training data is presented. Moreover insights of the data are accessed such as possible problems with the data that can lead to poor performance. Next, the selected methods, the motivation to choose them, their adaptation to the current dataset are shown. Finally, results are visualized and their possible explanation, based on the knowledge of basics of deep learning and related work are given.

The main goal for the experimental part was to show the feasibility of the object detector on the custom dataset of zebra crossings, which is hard in general for deep learning models. the dataset resolution is small and it contains in total 716 training images for 5 categories. Taking it into account, data augmentation and transfer learning were applied. For accessing the model mAP was used. In addition the inference time was also measured as the one of the goals of the thesis is to use fast, real-time classifier.

3.1 Data

The City of Helsinki's uses open data WMS service, which is API for image tiles. The python library that helps with requests is OWSlib. The zoom level is important in order to get correct images. The most suitable was to use 5 cm accuracy. The corresponding WMS layer used: Ortoilmakuva_2019_5 cm. 20 cm model will be a good version for an ML model.

Further to get the pedestrian crossings, the data from the Map Service, that specifies the map layer corresponding to the crossings, should be downloaded. The file 'klinj_suojatie.gpkg'from kartta.hel.fi contains bounding boxes for pedestrian crossings. .gpkg is a data format for geographic information system. Image patches were stored as jpg files. in a subfolder of "PedestrianCrossingImg/". A separate subfolder is created for images obtained from a particular layers (such as aerial photographs or building maps).

The folder contains 1392 images. Some images were very bad quality, so we decided to discard them.

Data was labelled using a graphical image annotation tool *LabelImg* 17. https://github.com/tzutalin/labelImg.

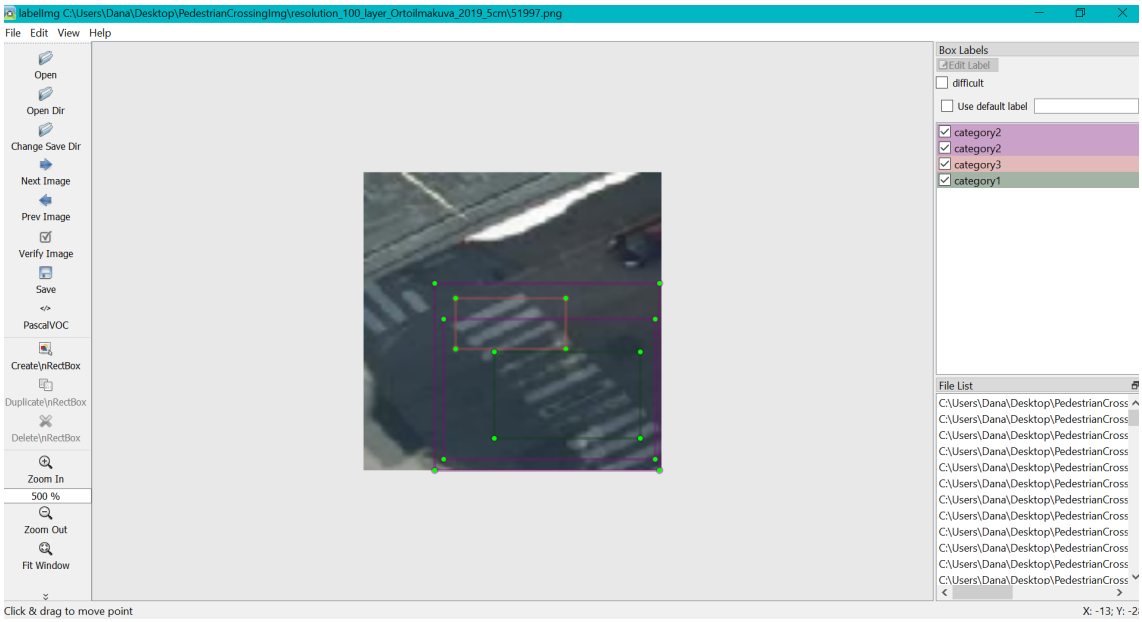

Figure 17: Labelling with LabelImg.

It is open source, and provides graphical interface for drawing bounding box around the object and providing corresponding label. The annotations are saved in PASCAL VOC format as XML files. Data labelling one of the most crucial part of machine learning. Without proper, consistent labelling, the model will fail at learning. During labelling we followed steps recommend by one of Stanford's course on computer vision Fig. 18.

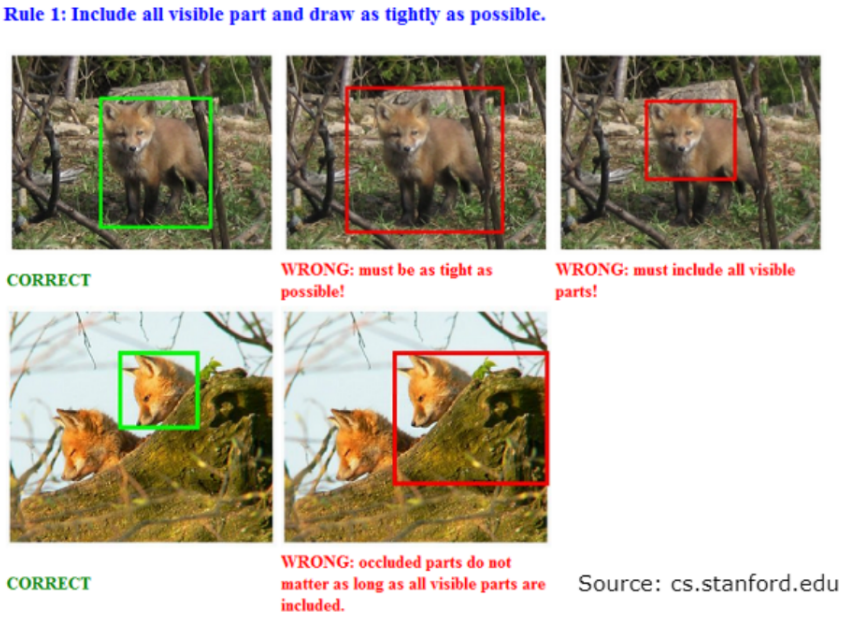

Figure 18: Labelling recommendations.

According to *Tiemerkintöjen kuntoluokitus* [9].There are five condition classes and the condition class is determined by the aforementioned criteria according to the weakest component:

- Condition class 5 (very good): Road markings are like new. Wear is hardly recognizable >95 %
- Condition class 4 (good): Minor wear of road markings. Also, if the marking is otherwise flawless, but isn't aligned with the old marking >80-95%
- Condition class 3 (satisfactory): About a quarter of the road markings has worn off and/or been covered. 70 – 80%
- Condition class 2 (bad): Almost half of the road markings have worn off and/or been covered 50-70%
- Condition class 1 (very bad): Over half of the road markings have worn off and/or been covered. <50% (see. Fig. 20)

The size of the images is 100 by 100. As the size influence the time for training neural network, 100 by 100 is perfect for us. Moreover most images are concentrated on road markings so background noise is minimum. After labelling we got 716 images and 958 annotations as some images contain more than one pedestrian crossing. In the Fig. 19 the spread of the data among classes is shown.

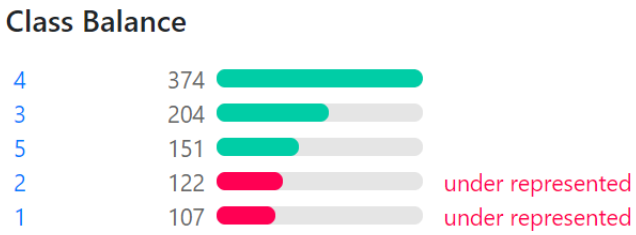

Figure 19: Distribution of class categories in our dataset

According to Google AutoML Vision, data preparation tutorial it is advisable that data has at least 1000 images per class and minimum 10 or 50 for advanced models. However for smaller datasets, transfer learning, image augmentation help a lot [28].

3.2 Problems with Data

During labelling we encountered some problems (see Fig. 21 and 24) as different shapes of zebra crossing, cars, shadows, occlusions. Some zebra markings were recognized as separate in highly damaged road markings.

4 Methods

This chapter provides information the usage of two object detection algorithms: Retina Net and YOLO5. Data preprocessing, data preparation, data augmentation, environment set up, implementation of the algorithms and their performance are discussed.

4.1 RetinaNet

The first object detection that we used was RetinaNet. It achieved great results compared to other detection algorithms on COCO dataset that is challenging for most object detection algorithm. As we see from (see Fig. 23) RetinaNet outperforms other algorithms and is robust to detect small objects. For the training, as the dataset was small, it was advised to do transfer learning. Transfer Learning is process of loading a pretrained model or weights files as the starting point, or using a previous checkpoint. This also boosts training performance and training speed, compared to training from scratch. We used weights from pretrained model ResNet50 trained on COCO dataset. That means that this weights are already good in recognizing common objects. Common backbones for Retina Net are ResNet50, ResNet101. We used ResNet50 backbone trained on COCO dataset. Initialization, according to [56]

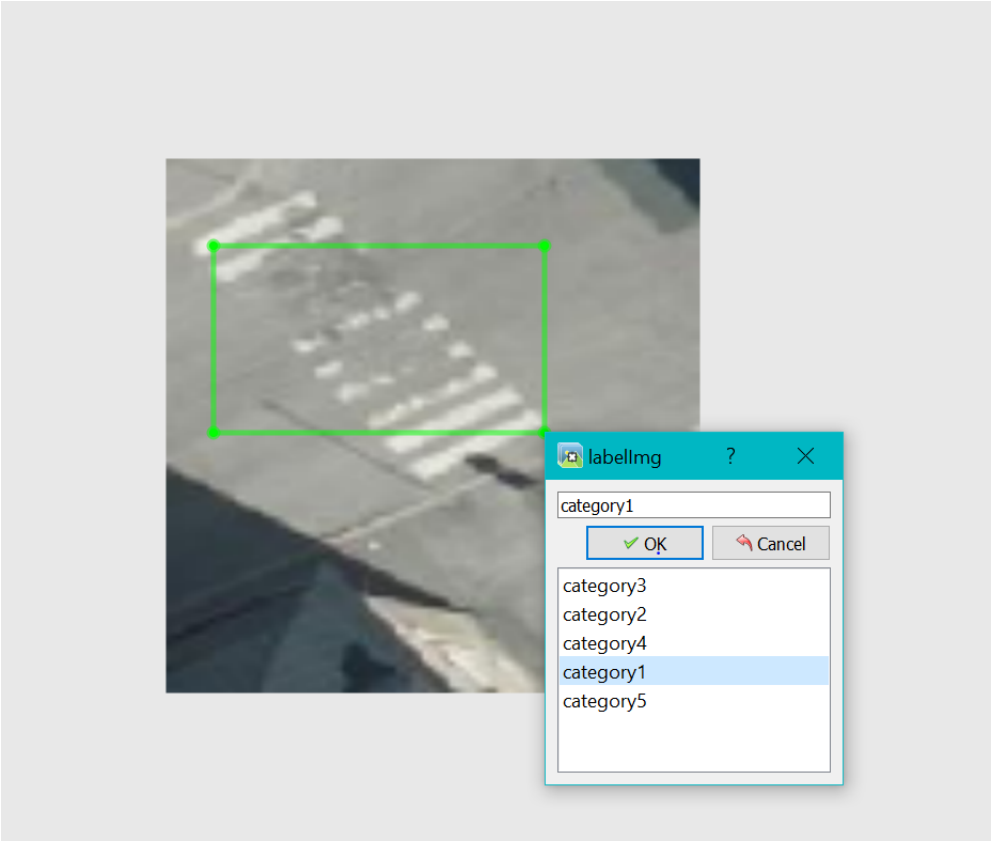

Figure 20: Example of category 1 zebra crossing.

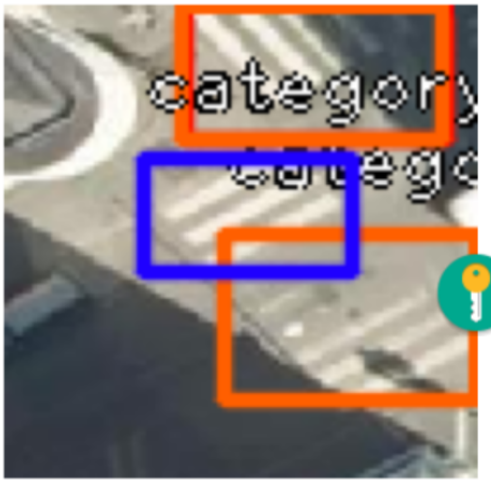

Figure 21: Example when one road marking was recognized as 2.

Figure 22: Problems

affects training a lot. The prior probability used by Lin et. al., for anchor boxes is 0.01 and same for bias of last convolutional layer of class subnet [56]. That was done due to fact that approximate ratio of foreground to background is about 0.01.

Below is the parameters that should be specified before the training.

- –**freeze-backbone** means to not train any of the weights that are part of the backbone. As we have small dataset, this is useful as the overfitting will be tackled. Freezing backbone - allows to freeze weights of the backbone and train purely feature pyramid network, which allows to spend less time on training.
- –**random transform** performs data augmentation.
- –**weights** the path to pretrained model is specified
- –**batch-size** depends on the memory restrictions, computing power however higher value gives a smoother learning curve. We take 8 as the batch-size. In general batch size 8 is a good starting point. And can be increased by 2, in

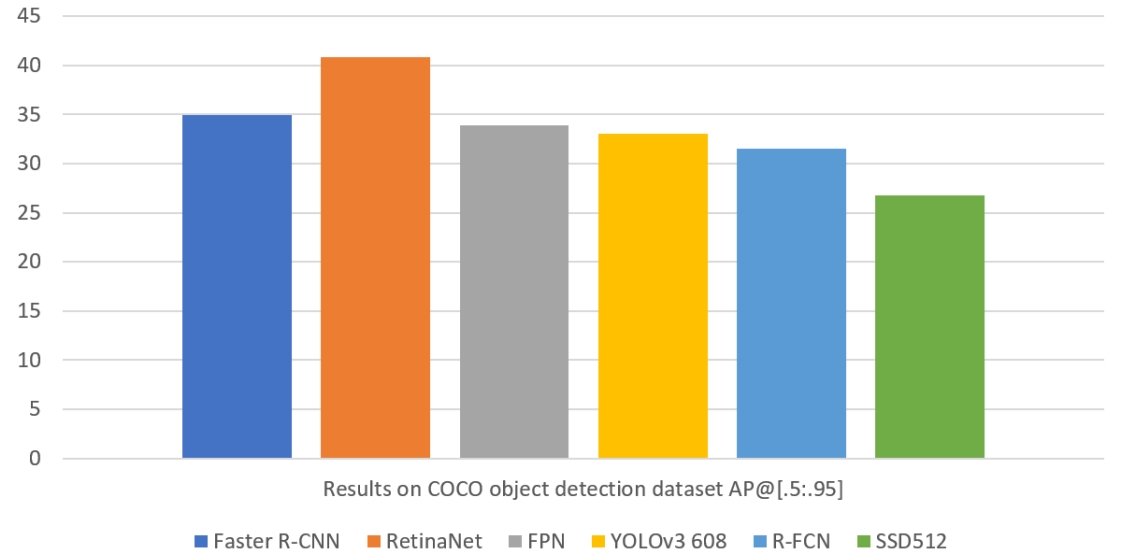

Figure 23: Comparison between object detection algorithms on COCO dataset

case of the successful start of the training process, i. e. loss starts to converge. The calculation for step size is simple: steps = count of rows in train.csv / batch-size. RetinaNet converges quickly, so a smaller number of epochs usually is sufficient.

- –**steps** number of step per epochs, usually it is equal = (count of rows in train.csv / batch-size)
- –**epochs** number of epoch we want to train for. Recommended at least 20 epochs.

According to related work done with RetinaNet, it is advisable to allow model to reach a total loss of at least 2. And the lower the loss, the better. However the low less can also be a sign of overfitting.But in general loss less that 2 is recommended to achieve good enough results.

Anchors selection. RetinaNet has 5 default anchor boxes 32, 64, 128, 256 and 512. For smaller objects the anchor of size 512 can be substituted by 16. We left the default anchors as bounding boxes were pretty big and rarely had small area to capture. As lower learning rate, non max suppression threshold and increase in image size did not show any improvements, they were not used [58].

Inference Training models only contain the parameters needed for training while inference models contain an entire model. So to get predictions on the test data, one should use inference model. Usually the last model trained at final epoch is taken, in case of YOLO5 we took only best model. Then it is converted to inference model. And we can make prediction on the new, unseen data.

Data preparation We utilized Google Colab to run the experiments. The data was uploaded as zip file and further was transformed into two csv files. The first one contains the path, bounding box and class name for each image. Each row corresponds to one labelling. The second file contains the class name and their mapping.

4.1.1 First implementation of RetinaNet with tf1.x

We utilized the implementation of object detection with RetinaNet by Fizyr https://github.com/fizyr/keras-retinanet

The first version was done in March 2020. We used Tensorflow version 1.x and Keras version 2.3x. With Tensorflow 1.x we had no issues inputting data of size less that *steps/batch size*. We used –batch-size 8 –steps 500 –epochs 10 parameters and ResNet50 as a backbone to train our model Fig. 24. Model was trained with

```
!keras-retinanet/keras_retinanet/bin/train.py --freeze-backbone \
  --random-transform \
  --weights {PRETRAINED_MODEL} \
  --batch-size 8 \
  --steps 500 \
  --epochs 15 \
  csv annotations.csv classes.csv
```

Figure 24: Training specification

checkpoint, 4 times and each model had 10 epochs and last one 15 epochs. The loss after first 10 epochs was 1.3840 with regression loss of 1.09 and classification loss of 0.2852. The final loss after 40 epochs was 0.8947.

```
Epoch 00011: saving model to ./snapshots/resnet50_csv_11.h5
Epoch 12/15
500/500 [==============================] - 254s 508ms/step - loss: 0.9117 - regression_loss: 0.7514 - classification_loss: 0.1603

Epoch 00012: saving model to ./snapshots/resnet50_csv_12.h5
Epoch 13/15
500/500 [==============================] - 253s 507ms/step - loss: 0.9038 - regression_loss: 0.7449 - classification_loss: 0.1589

Epoch 00013: saving model to ./snapshots/resnet50_csv_13.h5
Epoch 14/15
500/500 [==============================] - 253s 507ms/step - loss: 0.9035 - regression_loss: 0.7462 - classification_loss: 0.1573

Epoch 00014: saving model to ./snapshots/resnet50_csv_14.h5
Epoch 15/15
500/500 [==============================] - 254s 508ms/step - loss: 0.8947 - regression_loss: 0.7416 - classification_loss: 0.1531

Epoch 00015: saving model to ./snapshots/resnet50_csv_15.h5
```

Figure 25: The final loss after 40 epochs with tf1.x and bias labelling.

As was further investigated, the data labelling process, contained a bias. Especially class 3 and class 4 were incorrectly labeled. The bounding boxes were not tight (see Fig. 26) and many poor quality road markings were also labeled. However as the dataset is small sized, poor quality examples should be omitted as they can affect negatively the training process. Considering the mistakes in labeling, the second labelling was done. It tried to make all boxes tight, exclude poor quality images and try to rigorously distinguish between class 3 and class 4. The second model showed

Figure 26: Wrong labelling, the bounding box is not tight

better results. We further accessed the mAP which was 0.6343. As per class, average precision was highest for class 4, as it also contains more examples. Class 5 also has the highest AP. This can be explained by the similar, easily distinguishable features that category 5 class images shared. The worst results were for the class 3, as it very similar to class 4 and class 2. Manual labelling of those classes was hard and that can lead to a possible bias.

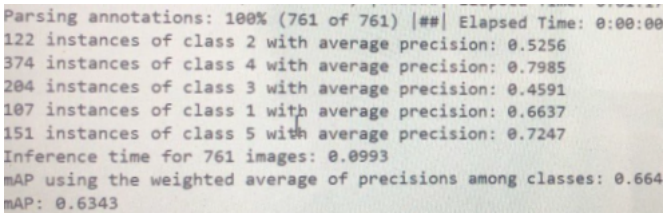

Figure 27: Evaluation of the Retina Net tf1.x on **train** data, using updated dataset

One example of the result from the implementation is shown below. The time for processing is 0.05 seconds.

4.1.2 Second implementation of RetinaNet with tf2.3

Requirements: tensorflow = 2.3.1, keras = 2.4.1. During training we encountered an error (see. Fig 30:

That made us to decrease the steps (Fig. 24) to 50. As we had dataset of 716 images and with batch size of 8, the *ceil*(718/8) was to take 50 steps. However we did not see any improvements, as the training reached Plateau. That can be possibly

```
Running inference on: 52566.png
processing time:  0.056893110275268555
category2 0.733415
```

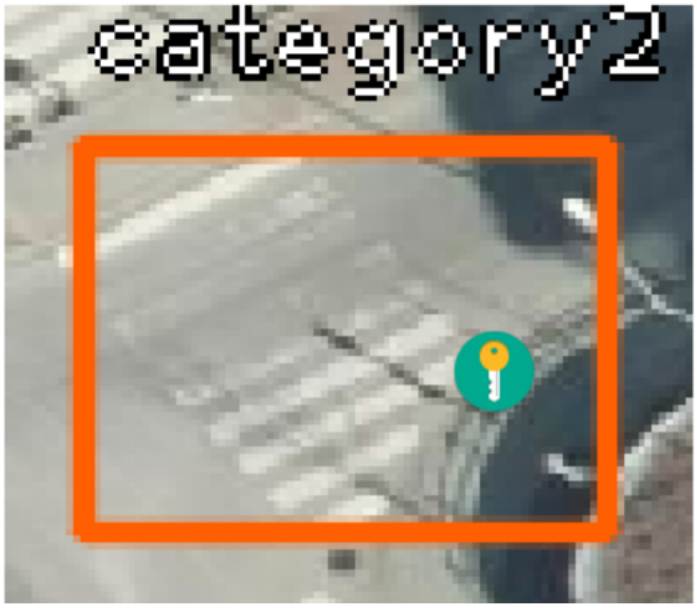

Figure 28: The performance of Retina Net with tf1.x and bias labelling.

> WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your dataset or generator can generate at least `steps_per_epoch * epochs` batches (in this case, 7 batches). You may need to use the repeat() function when building your dataset.

Figure 29: Tensorflow 2.x error, the image is taken from *Stackoverflow*

because number of steps was not enough to reach a global minimum and model was stack in a local minimum.

```
Epoch 00028: ReduceLROnPlateau reducing learning rate to 9.999999974752428e-08.
50/50 [==============================] - 96s 2s/step - loss: 1.9158 - regression_loss: 1.4516 - classification_loss: 0.4642
Epoch 29/30
50/50 [==============================] - ETA: 0s - loss: 1.9205 - regression_loss: 1.4552 - classification_loss: 0.4653
Epoch 00029: saving model to ./snapshots/resnet50_csv_29.h5
50/50 [==============================] - 96s 2s/step - loss: 1.9205 - regression_loss: 1.4552 - classification_loss: 0.4653
Epoch 30/30
50/50 [==============================] - ETA: 0s - loss: 1.9169 - regression_loss: 1.4595 - classification_loss: 0.4574
Epoch 00030: saving model to ./snapshots/resnet50_csv_30.h5

Epoch 00030: ReduceLROnPlateau reducing learning rate to 1.0000000116860975e-08.
50/50 [==============================] - 96s 2s/step - loss: 1.9169 - regression_loss: 1.4595 - classification_loss: 0.4574
```

Figure 30: Tensorflow 2.x error, the image is taken from *Stackoverflow*

After that we decided to increase the size of the dataset. We applied the data augmentation technique such as shear shear transformation $\pm$ 15 and blur 1 pixel. The data augmentation was done using *Roboflow* software [66].

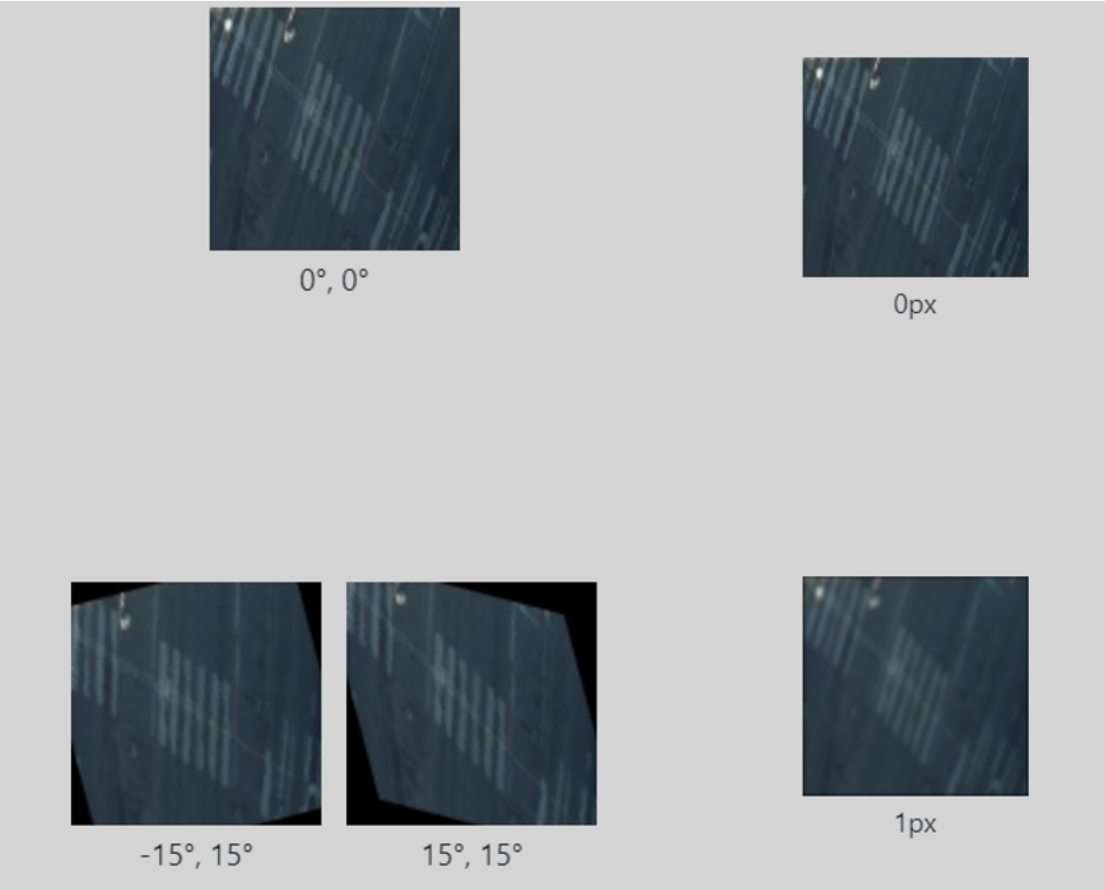

Figure 31: Image augmentation: shear transformation ± 15 and blur 1 pixel

The motivation to use blurring was that the objects in the wild are not usually in focus or model might be overfitting on hard edges. Shear transformation helps to change perspective, so the model is more robust towards change in pitch and yaw. Also we increased the size of the image to 416 by 416 and overall size of the dataset after augmentation was 3055 (94% of whole dataset) images. We took big percentage for training data as our data is small (will poorly generalize on small training dataset) and to increase steps. We left 76 for validation and 76 images with 100 annotations for testing. We set *steps* to 300. We trained with 25 epochs. The results were not good (see. Fig. 32).

After first 5 epochs we run inference on the test data to see how fast is RetinaNet learns (see Fig. 33).

```
Epoch 00019: saving model to ./snapshots/resnet50_csv_19.h5
300/300 [==============================] - 275s 918ms/step - loss: 1.6044 - regression_loss: 1.1970 - classification_loss: 0.4074
Epoch 20/25
300/300 [==============================] - ETA: 0s - loss: 1.5883 - regression_loss: 1.1827 - classification_loss: 0.4056
Epoch 00020: saving model to ./snapshots/resnet50_csv_20.h5
300/300 [==============================] - 275s 917ms/step - loss: 1.5883 - regression_loss: 1.1827 - classification_loss: 0.4056
Epoch 21/25
300/300 [==============================] - ETA: 0s - loss: 1.5829 - regression_loss: 1.1807 - classification_loss: 0.4022
Epoch 00021: saving model to ./snapshots/resnet50_csv_21.h5
300/300 [==============================] - 275s 917ms/step - loss: 1.5829 - regression_loss: 1.1807 - classification_loss: 0.4022
Epoch 22/25
300/300 [==============================] - ETA: 0s - loss: 1.5756 - regression_loss: 1.1734 - classification_loss: 0.4022
Epoch 00022: saving model to ./snapshots/resnet50_csv_22.h5
300/300 [==============================] - 275s 917ms/step - loss: 1.5756 - regression_loss: 1.1734 - classification_loss: 0.4022
Epoch 23/25
300/300 [==============================] - ETA: 0s - loss: 1.5523 - regression_loss: 1.1561 - classification_loss: 0.3962
Epoch 00023: saving model to ./snapshots/resnet50_csv_23.h5
300/300 [==============================] - 275s 917ms/step - loss: 1.5523 - regression_loss: 1.1561 - classification_loss: 0.3962
Epoch 24/25
300/300 [==============================] - ETA: 0s - loss: 1.5400 - regression_loss: 1.1461 - classification_loss: 0.3939
Epoch 00024: saving model to ./snapshots/resnet50_csv_24.h5
300/300 [==============================] - 275s 916ms/step - loss: 1.5400 - regression_loss: 1.1461 - classification_loss: 0.3939
Epoch 25/25
```

Figure 32: Training Retina Net for total of 25 epochs with 300 steps

```
Running network: 100% (76 of 76) |########| Elapsed Time: 0:00:22 Time:  0:00:22
Parsing annotations: 100% (76 of 76) |####| Elapsed Time: 0:00:00 Time:  0:00:00
13 instances of class 3 with average precision: 0.0974
10 instances of class 2 with average precision: 0.0653
45 instances of class 4 with average precision: 0.3721
23 instances of class 5 with average precision: 0.0807
10 instances of class 1 with average precision: 0.0350
Inference time for 76 images: 0.2729
mAP using the weighted average of precisions among classes: 0.2067
mAP: 0.1301
```

Figure 33: Testing Retina Net on 76 new images after 5 epochs with 300 steps

The results after first (see Fig. 33) inference, mAP using weighted average of precisions among classes, only 0.2067. All classes, except 5 are recognized poorly. Class 4 is the best, however it has more samples.

```
Running network: 100% (76 of 76) |########| Elapsed Time: 0:00:22 Time:  0:00:22
Parsing annotations: 100% (76 of 76) |####| Elapsed Time: 0:00:00 Time:  0:00:00
13 instances of class 3 with average precision: 0.0884
10 instances of class 2 with average precision: 0.1114
45 instances of class 4 with average precision: 0.4634
23 instances of class 5 with average precision: 0.2650
10 instances of class 1 with average precision: 0.1134
Inference time for 76 images: 0.2718
mAP using the weighted average of precisions among classes: 0.3004
mAP: 0.2083
```

Figure 34: Testing Retina Net on 76 new images after total of 20 epochs with 300 steps

The second test in Fig. 38, on total of 20 epochs showed improvements and increase of mAP to 0.3004. With improvements to all the classes except class 3. Third test included more 15 epochs run on previous checkpoints. So in total of 35 epochs. Total loss was 1.44 and mAP 0.31

From the images 36 and 37, the network predicts bounding boxes with the low confidence and errors. The performance slightly improved during 15 epochs, but not much. We decided to proceed with another object detection method.

```
Running network: 100% (76 of 76) |########| Elapsed Time: 0:00:22 Time:  0:00:22
Parsing annotations: 100% (76 of 76) |####| Elapsed Time: 0:00:00 Time:  0:00:00
13 instances of class 3 with average precision: 0.1346
10 instances of class 2 with average precision: 0.1343
45 instances of class 4 with average precision: 0.4417
23 instances of class 5 with average precision: 0.2571
10 instances of class 1 with average precision: 0.2432
Inference time for 76 images: 0.2704
mAP using the weighted average of precisions among classes: 0.3100
mAP: 0.2422
```

Figure 35: Testing Retina Net on 76 new images after total of 35 epochs with 300 steps

Figure 36: Testing Retina Net on the training images after total of 35 epochs with 300 steps

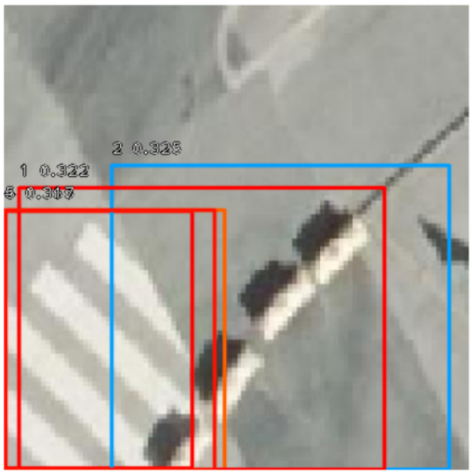

Figure 37: Testing Retina Net on the test images after total of 35 epochs with 300 steps

4.2 YOLO5

We chose YOLO5 to as our object detector to conduct a second experiment. More about YOLO5 can be read in section 2.5.2.

Data The original data was saved in Pascal format with .xml extension. However for YOLO, we needed a txt file containing class and coordinates of the bounding boxes and separate .yaml file for data. With the help of the Roboflow [66] we created 2 datasets. Both contained as training data 1827 images. First dataset had preprocessing applied. The following techniques were used: resizing to 416 by 416, grayscale (one color channel), autoadjustment of the contrast with adaptive equalization. For image augmentation we used shear ±15 as was used in RetinaNet dataset and we added noise. Second dataset had only shear ±15 and resize to 416 by 416 transformations. The benefits of using the above mentioned data manipulation techniques are provided below:

- Grayscale, merging color channels so model is faster and not sensitive to colors. That is important for illumination changes. As we want to make model treat equally the same objects under different illumination. Also for speed is the target then color is not of importance.
- Auto adjust the contrast. Based on the image's histogram, contrast is boosted. That is helpful as it improves normalization and line detection under different lighting conditions. We used the specific method of adaptive equalization that can better detect edges.
- Noise helps to prevent overfitting and adversial attacks, that can fool networks using special constructed noise and totally changing the prediction.

In Fig. 31, shear transformation is visualized.

4.3 Results of YOLO5

Training on 100 epochs, first dataset with yolo5s model. The training took 0.475 hours and best weights were of size 14.8 Mb. After first test, we see that mAP_0.5 was going up, starting from 60-70 it the slope 39 is less steep, however it is not flat. Which means there is a room for improvement and we can train for more epochs. The slope of the loss function is slowly going down but also have not converged yet 40. mAP[.5,.95] denotes average mAP over different IoU thresholds, from 0.5 to 0.95, step 0.05 (0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95).

Before running for longer epoch we wanted to compare the performance of the network on the second dataset. We also run it for 100 epochs, using yolo5s model. The training took 0.925 hours.

We see in Fig.41 that mAP can also be improved. Example of running one test image on obtained model in Fig. 44

Third test was conducted on first dataset but this time using number of epochs equal to 150 and yolo5l model. Time required for running model 2.077 hours. The

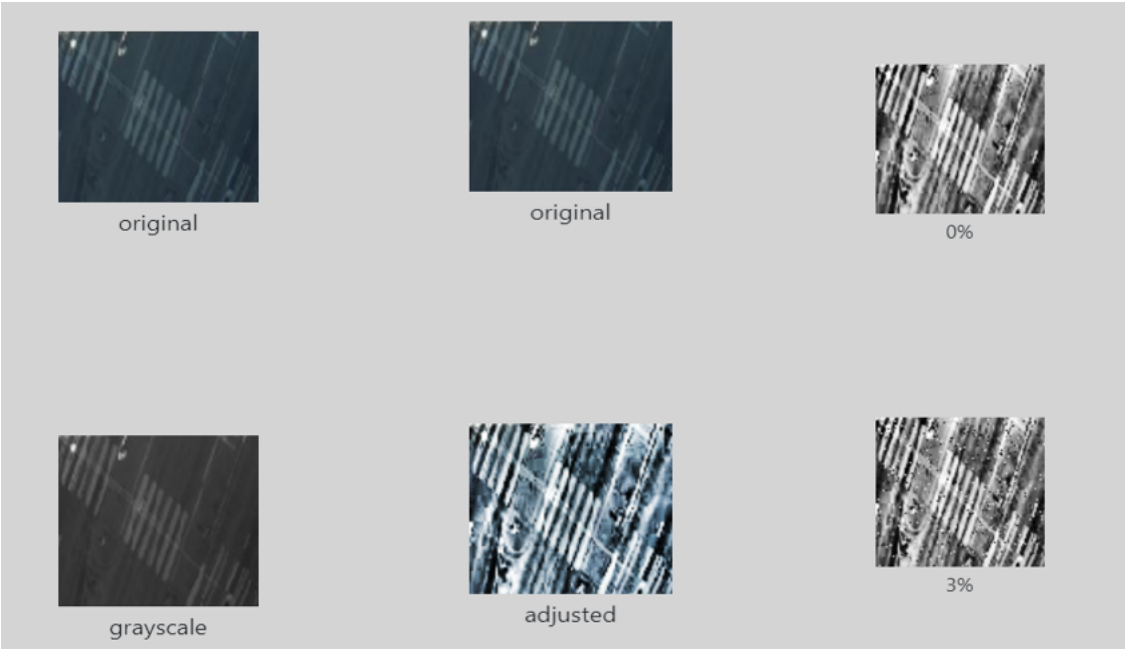

Figure 38: First dataset. Image preprocessing: grayscale, autoadjustment of the contast. Image augmentation: added noise 3%

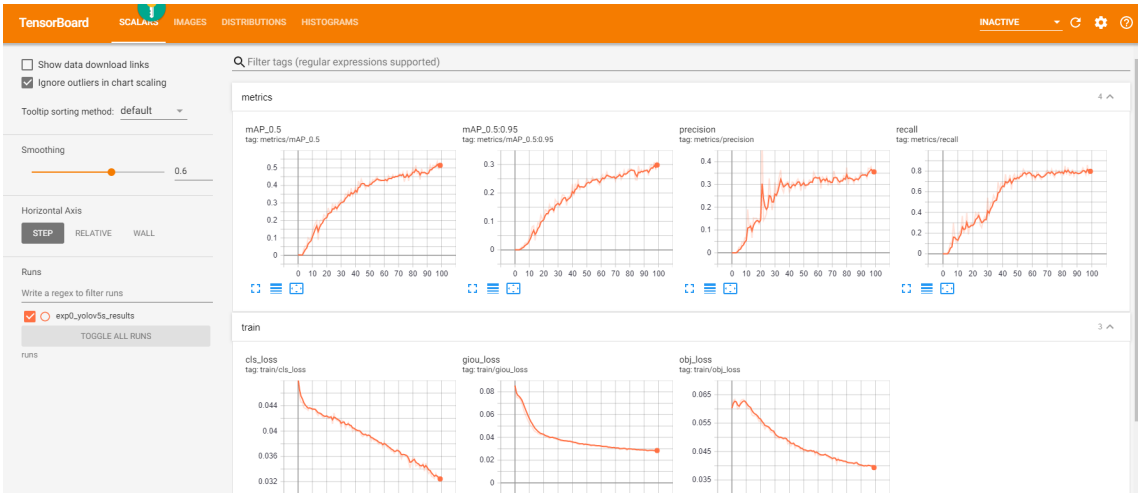

Figure 39: Tensorboard outputs on first dataset with 100 epochs, yolo5s model.

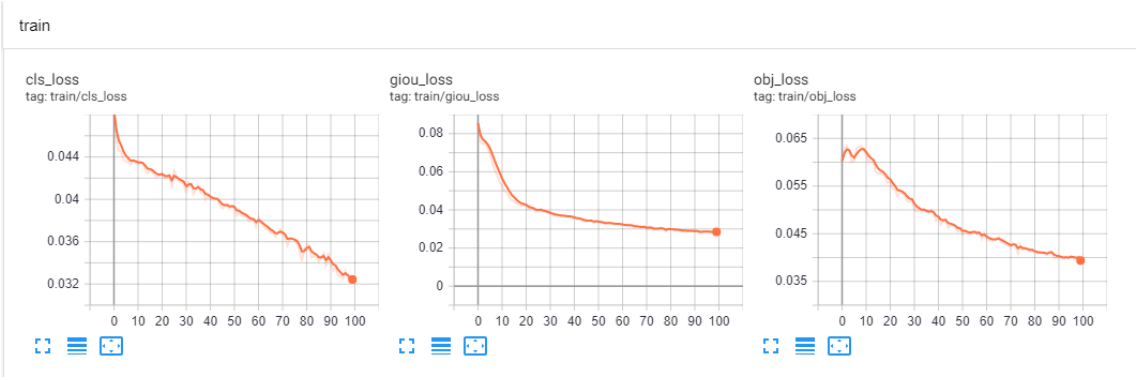

Figure 40: Tensorboard outputs on first dataset with 100 epochs, yolo5s model.

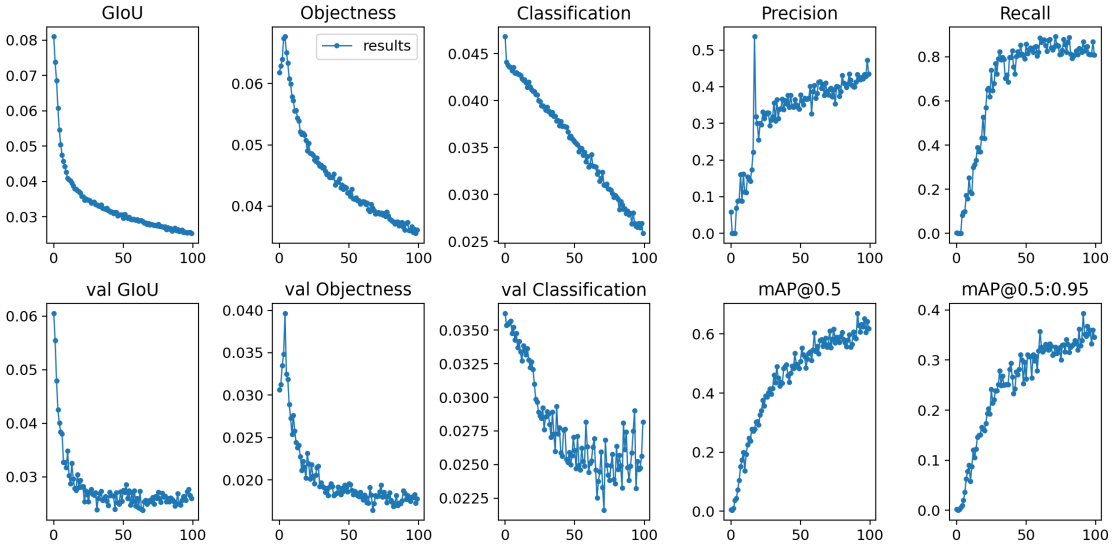

Figure 41: Results on second dataset. YOLO5s ,model with 100 epochs

.

Figure 42: Class 1 with 0.78 confidence. Inference on a second dataset. yolo5s model with 100 epochs

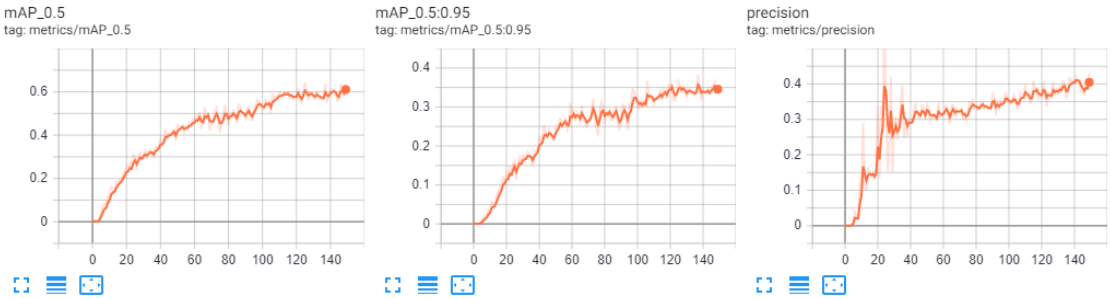

Figure 43: Training first dataset with yolo5l model on 150 epochs.

mAP curve is more flat in the end and achieves mAP_0.5 (PASCAL VOC metric)of 0.6 while first model on same data achieved 0.5 mAP_0.5. It took 4.121 hours to train. The forth test was to run same model as in third but on the second dataset. We are sure that the results of more advanced model and longer training time will be better. But it is interesting to compare how the same models performs on different datasets Fig. 44.

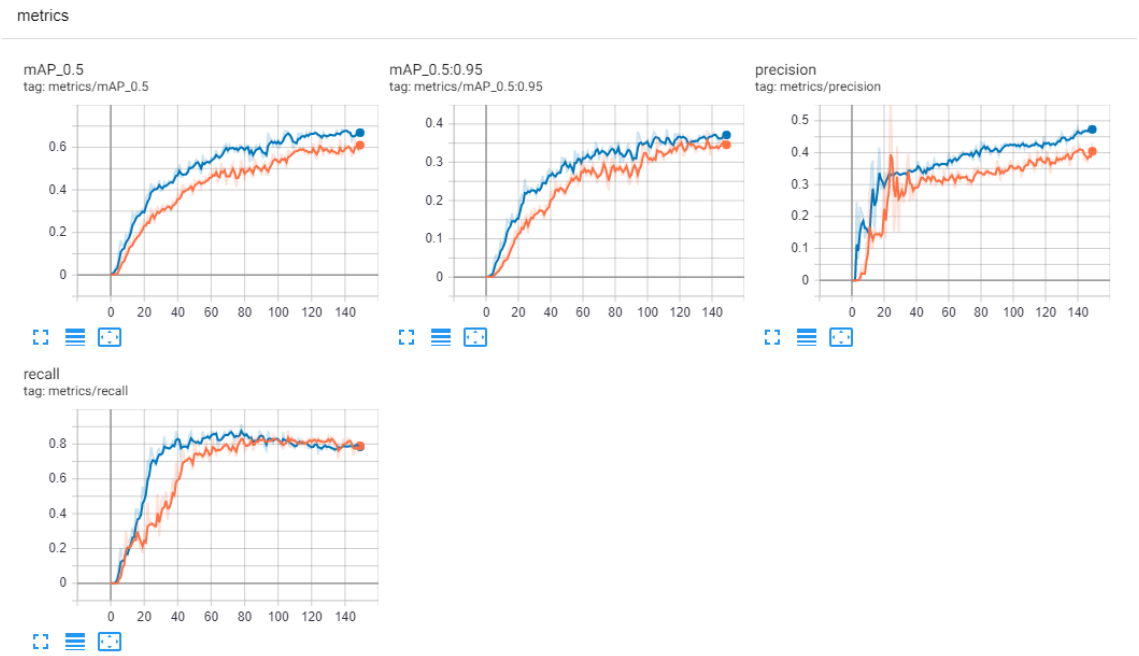

Figure 44: Comparison of the results of yolo5l model with 150 epochs on dataset1 (in red) and dataset 2 (in blue).

Some results on the test batch of forth test

4.4 Discussion

We evaluated the performance of the Retina Net and YOLO5. Retina Net with tf1.x version allowed us to use higher steps. We achieved mAP of 0.63 on the training data. However we could not use the same number of steps with tf2.x version and the model did not provide us sufficient results. Further work is to train longer or apply more augmentation on our data. So far mAP achieved on test data is 0.24. Visualizations showed that network outputs many bounding boxes with low confidence. However bounding boxes are tight which means that localization loss reached pretty good results.

YOLO5 performed well on our dataset. Especially on the data without preprocessing. That can be explained by losing color information during converting images to grayscale. Yolo5l model trained during 150 epochs showed highest mAP_0.5 of 0.6. YOLO5l trained on dataset2 slightly outperforms in mAP and precision.But time for training is double, 4 hours versus 2 hours for dataset with one channel

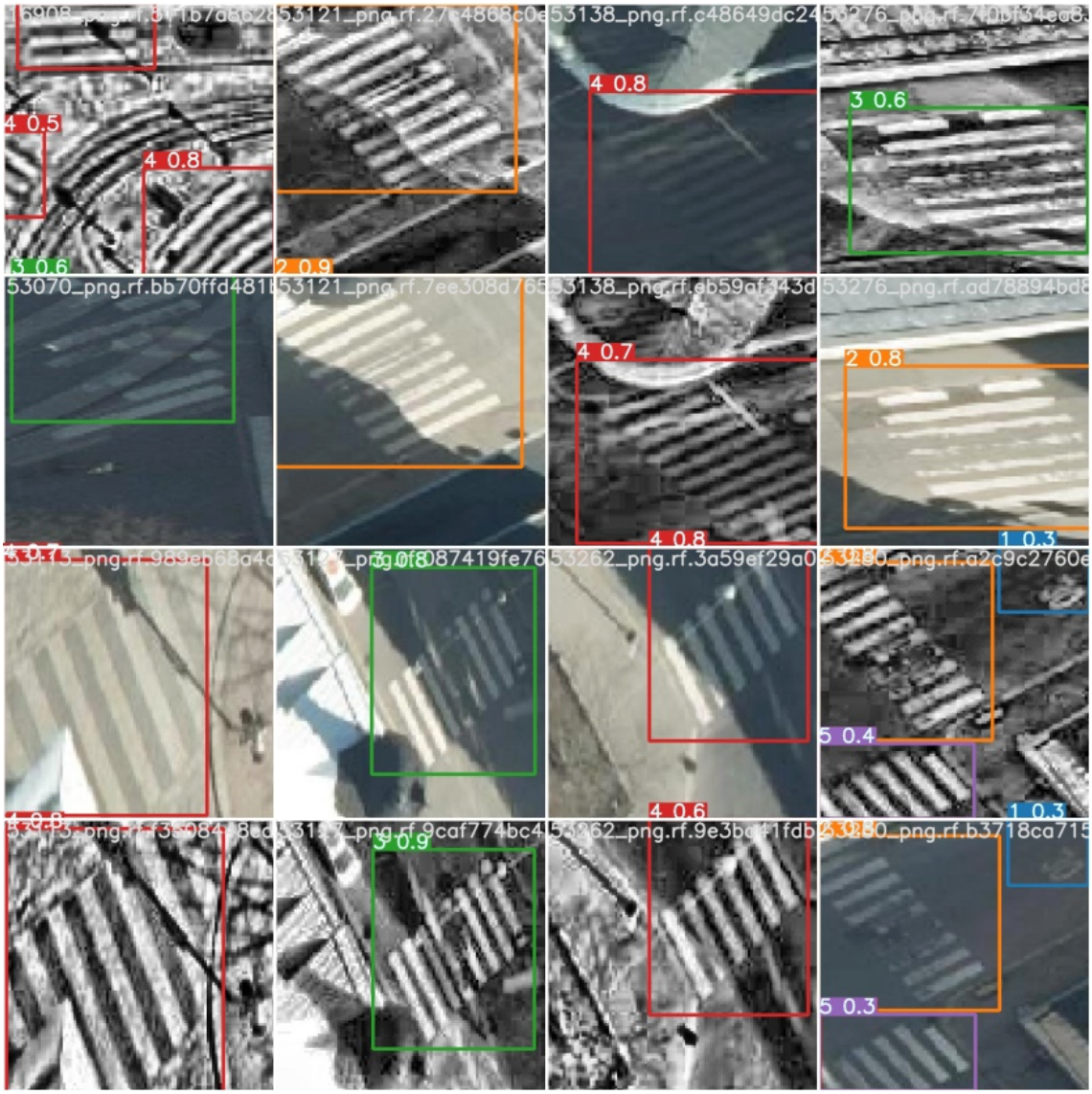

Figure 45: Results of the forth test

information. If the computing power is of concern, it is recommended to use images with one color channel. The recall is the same after 100 epochs. It can mean that it already had converged around 100 epochs.

From the Fig. 45, we see that model performs well. However there are still some zebra classified incorrectly.The above mention problem can be explained by possibility of bias in labelling and unbalanced dataset.

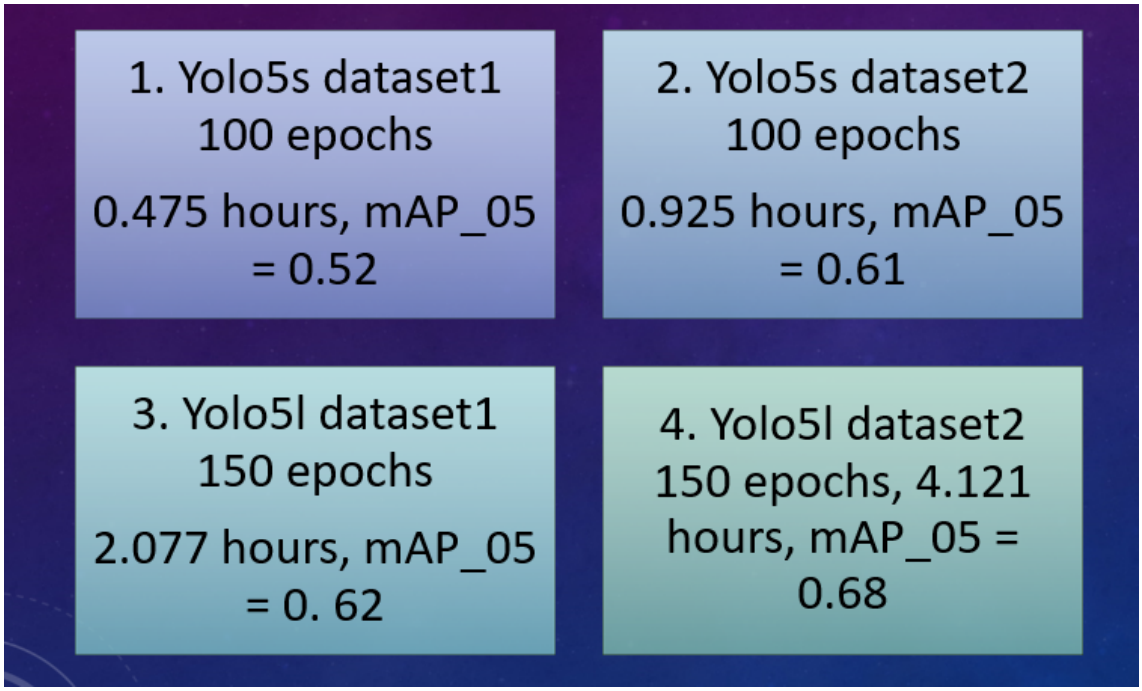

Figure 46: General overview on experiment results with YOLO5

5 Summary and Future Goals

Road marking road classification is crucial in designing smart cities which aim towards increasing the life quality of their citizens. They play essential role in city planning, road infrastructure, public safety and transport flow. Being aware of road markings that need repair work will ease the city planning and prevent dangerous situations.

In this thesis project we researched the feasibility of the application of deep neural network to properly learn features required for classification of pedestrian road markings into 5 categories. We studied traditional and deep learning based methods and preferred deep learning ones as they can scale for the new data and will learn themselves high semantic features for predictions, We investigated the performance of various object detectors and chose Retina Net and YOLO5 for our experiments. Both networks showed good results in other object detection works.

As neural network benefit from larger datasets, we applied image augmentation and used transfer learning. Both networks gave when tested on test data, gave visually good results. However mAP for Retina Net was only 0.31 compared to 0.68

of YOLO5l model with the same training time. Inference results of YOLO5 were also better, 0.017 vs 0.05 s. The inference speed is crucial for real time predictions. Yolo5 applied to dataset with one channel, compared to dataset with 3 channels, take twice less time, maintaining comparable, slightly smaller performance. We showed that despite having small, low-resolution data, with unbalanced classes we can reach reasonable accuracy.

As for future we would like to extend our job by gathering more data. As could be seen from past research, the road markings vary across different regions. The geographical specifications, the illumination, requirements of road markings are different. We have built model robust for Helsinki, however with more data we can extend model to other cities.

To increase the performance of the object detector we either need to collect more data, for example from other cities in Finland or try more data augmentation. Another way is to reduce number of classes, as network performs poorly due to bias in labelling introduced by human error. It was sometimes hard to distinguish between class 3 and class 4. Better set of labelling instructions should be devised.

References

[1] Smartcitiesworld. *https://www.smartcitiesworld.net/news/news/zero-pedestrian-fatalities-in-helsinki-traffic-in-2019-5002*

[2] *https://kartta.hel.fi/?setlanguage=en* 3. painos. Helsinki, RPS-yhtiöt, 2007.

[3] https://medium.com/swlh/yolov5-controversy-is-yolov5-real-20e048bebb08

[4] Rodrigo F. Berriel, Andre Teixeira Lopes, Alberto F. de Souza, and Thiago Oliveira-Santos, N. Deep Learning Based Large-Scale Automatic Satellite Crosswalk Classification. 2. 2017.

[5] K. Fukushima , S. Miyake. Neocognitron: a self-organizing neural network model for a mechanism of visual pattern recognition. In:*Proceedings of the Competition and Cooperation in Neural Nets, 1982*

[6] Dragan Ahmetovic, Roberto Manduchi, James M. Coughlan, Sergio Mascetti. Mind your crossings: Mining GIS imagery for crosswalk localization. *ACM Trans Access Comput.* 2017

[7] Hoeser, Thorsten,Kuenzer, Claudia. Object Detection and Image Segmentation with Deep Learning on Earth Observation Data: A Review-Part I: Evolution and Recent Trends. In: *Remote Sensing.* 2020.12.

[8] Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In: *Proceedings of the NeurIPS.*

[9] Tiemerkintöjen kuntoluokitus. *Toteuttamisvaiheen ohjaus. Tiehallinto.* Helsinki 2014.

[10] https://www.mantapm.com/post/wear-and-tear-on-your-pavement-markings-what-causes-it-and-when-should-you-paint

[11] Tao Wu, Ananth Ranganathan. A Practical System for Road Marking Detection and Recognition.in :*2012 Intelligent Vehicles Symposium.*2012.

[12] Wu Xiongwei, Doyen Sahoo, Steven C. Hoi. Recent advances in deep learning for object detection. In:*Neurocomputing.* 2020. Vol. 396, p 39-64.

[13] Bengio, Y.; Courville, A.; Vincent, P. Representation learning: A review and new perspectives. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 2013. 35, 1798–1828

[14] Xinyi Liu, A. et al. "Automatic pedestrian crossing detection and impairment analysis based on mobile mapping system". In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. IV-2/W4. 2017.

[15] Franke, U.; Heinrich, §. Fast obstacle detection for urban traffic situations. In: *IEEE Trans. Intell. Transp. Syst.* 2002, 3, 173–181.

[16] Z. Li, Z.-X. Cai, J. Xie, and X.-P. Ren. Road markings extraction based on threshold segmentation. In:*Proc. 9th Int. Conf. Fuzzy Syst. Knowl. Discovery.* 2012. pp. 1924-1928.

[17] Fan, Yuhua Sun, Zhonggui Zhao, Guoying. A Coarse-to-Fine Framework for Multiple Pedestrian Crossing Detection. In: *Sensors 2020* 20, 4144.

[18] Riveiro B, Gonzalez-Jorge H, Martınez-Sanchez J, Dıaz-Vilarino L, Arias P. Automatic detection of zebra crossings from mobile LiDAR data. In: *Optics & Laser Technology.* 2015. 70:63–70.

[19] Coughlan, J.; Shen, A fast algorithm for finding crosswalks using figure-ground segmentation. In: *In Proceedings of the 2nd Workshop on Applications of Computer Vision, in Conjunction with ECCV* Graz, Austria. 2006. Vol. 5.

[20] J. Hu, L. Shen, S. Albanie, G. Sun and E. Wu. Squeeze and Excitation Networks. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*

[21] Choi, J.; Lee, J.; Kim, D.; Soprani, G.; Cerri, P.; Broggi, A.; Yi, K. Environment-detection-and-mapping algorithm for autonomous driving in rural or off-road environment. In: textitIEEE Trans. Intell. Transp. Syst. 2012. 13, 974–982.

[22] Li, Yunchong et al. Road Markers Recognition Based on Shape Information.In: *IEEE Intelligent Vehicles Symposium.* 2007.117-122.

[23] Yanbiao Sun, Fan Zhang, Yunlong Gao, and Xianfeng Huang. Extraction and Reconstruction of Zebra Crossings from High Resolution Aerial Images.In: *ISPRS Int. J. Geo-Inf.* 2016.5, 127;

[24] Akinlar C, Topal C. EDLines: A real-time line segment detector with a false detection control. In: *Pattern Recognition Letters.* 2011

[25] D. Herumurti, K. Uchimura, G. Koutaki, and T. Uemura, Urban Road Network extraction based on Zebra Crossing Detection from a very high resolution RGB aerial image and DSM data. In: *Signal- Image Technology & Internet-Based Systems (SITIS), 2013 International Conference on*2013, pp. 79–84.

[26] M. Ghilardi, J. Junior, and I. Manssour, M. Ghilardi, J. Junior, and I. Manssour. Crosswalk localization from low resolution satellite images to assist visually impaired people. 2016.

[27] D. Koester, B. Lunt, and R. Stiefelhagen. Zebra crossing detection from aerial imagery across countries.In: *International Conference on Computers Helping People with Special Needs*2016ȧ27-34

[28] T. M. Hoang, S. H. Nam and K. R. Park. Enhanced Detection and Recognition of Road Markings Based on Adaptive Region of Interest and Deep Learning. In: *IEEE Access*. 2019. Vol. 7, pp. 109817-109832.

[29] Chen, T.; Chen, Z.; Shi, Q.; Huang, X. Road Marking Detection and Classification Using Machine Learning Algorithms. In: *Proceedings of the IEEE Intelligent Vehicles Symposium*. 2015; pp. 617–621

[30] Tang, Zongzhi. A Novel Road Marking Detection and Recognition Technique Using a Camera-based Advanced Driver Assistance System. 2017.

[31] Rafael C. Gonzalez and Richard E. Woods. Digital Image Processing (3rd Edition). Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[32] Tsagkatakis, G.; Aidini, A.; Fotiadou, K.; Giannopoulos, M.; Pentari, A.; Tsakalides, P. Survey of Deep-Learning Approaches for Remote Sensing Observation Enhancement. In: *Sensors 2019*. 19, 3929.

[33] Reichstein, M.; Camps-Valls, G.; Stevens, B.; Jung, M.; Denzler, J.; Carvalhais, N.; Prabhat. Deep learning and process understanding for data-driven Earth system science. In: *Nature*.2019. 566, 195–204.

[34] J. Li, X. Mei, D. Prokhorov, and D. Tao. Deep neural network for structural prediction and lane detection in traffic scene. In: *IEEE Trans.Neural Netw. Learn. Syst.*. 2017. Vol. 28, no. 3, pp. 690-703.

[35] B. He, R. Ai, Y. Yan, and X. Lang. Accurate and robust lane detection based on dual-view convolutional neutral network. In: *Proc. IEEE Intell. Vehicles Symp.*.2016. pp. 1041-1046.

[36] R. G. von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall. LSD: A line segment detector. In: *Image Process. Line*.2012. Vol. 2, pp. 35-55.

[37] Geron Aurelien. Hands-on machine learning with Scikit-Learn, Keras and TensorFlow, 2nd Edition. 2019. p 450

[38] Zeiler, M.D., Fergus, R. Visualizing and Understanding Convolutional Networks. In:*Computer Vision–ECCV*.pp. 818–833.

[39] Simonyan, K.; Zisserman. A. Very Deep Convolutional Networks for Large-Scale Image Recognition.2014. *arXiv:1409.1556.*

[40] LeCun, Y.; Boser, B.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.; Jackel, L.D. Backpropagation Applied to Handwritten Zip Code Recognition. In:*Neural Comput.* 1989, 1, 541–551.

[41] Szegedy, C., Liu,W., Jia, Y., Sermanet, P., Reed, S.; Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A. Going deeper with convolutions. In: *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.2015. pp. 1–9.

[42] Chollet, F. Xception. Deep Learning with Depthwise Separable Convolutions. In: *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.2017. pp. 1800-1807.

[43] Tan, M.; Le, Q.V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In: *Proc. Mach. Learn. Res.*2019. 97, 6105–6114.

[44] Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D. Object detection with discriminatively trained part-based models. In: *IEEE Trans. Pattern Anal. Mach. Intell.*2009. 32, 1627–1645.

[45] S. Fidler , R. Mottaghi , A. Yuille , R. Urtasun. Bottom-up segmentation for top–down detection. In: *Proceedings of the CVPR.*2013.

[46] J.R. Uijlings , K.E. Van De Sande , T. Gevers , A.W. Smeulders , Selective search for object recognition. In: *Proceedings of the IJCV.*2013.

[47] Zhao, Zhong-Qiu et al. Object Detection With Deep Learning: A Review. In: IEEE Transactions on Neural Networks and Learning Systems 30.2019. 3212-3232.

[48] Murthy, CB, Hashmi, MF, Bokde, ND Geem. Investigations of object detection in images/videos using various deep learning techniques and embedded platforms-A comprehensive review. In: *Applied Sciences.*2020. Vol. 10. 9, 3280.

[49] K. He , X. Zhang , S. Ren , J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *Proceedings of the ECCV.*2014.

[50] S. Ren , K. He , R. Girshick , J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In: *Proceedings of the NeurIPS.* 2015.

[51] J. Dai , Y. Li , K. He , J. Sun. R-FCN: object detection via region-based fully convolutional networks. In: *Proceedings of the NeurIPS.* 2016.

[52] Lin, T.Y, Dollár, P.,Girshick, R., He, K., Hariharan, B., Belongie, S. Feature pyramid networks for object detection. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition.* 2017; pp. 2117–2125.

[53] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, Y. LeCun. Overfeat: integrated recognition, localization and detection using convolutional networks. 2013. *arXiv: 1312.6229.*

[54] J. Redmon , S. Divvala , R. Girshick , A. Farhadi . You only look once: unified, real-time object detection. In: *Proceedings of the CVPR.* 2016.

[55] W. Liu , D. Anguelov , D. Erhan , C. Szegedy , S. Reed , C.-Y. Fu , A.C. Berg. SSD: Single shot multibox detector. In: *Proceedings of the ECCV.* 2016.

[56] T.-Y. Lin , P. Goyal , R. Girshick , K. He , P. Dollár. Focal loss for dense object detection. In: *Proceedings of the ICCV.* 2017.

[57] J. Redmon , A. Farhadi. Yolo9000: better, faster, stronger. In: *Proceedings of the CVPR*. 2017.

[58] Pryanka Dwivedi. Detect and track people aerially over Stanford campus.

[59] Bochkovskiy Alexey, Chien-Yao Wang, Hong-Yuan Mark Liao. 'YOLOv4: Optimal Speed and Accuracy of Object Detection'.*arXiv:2004.10934.*

[60] Golnaz Ghiasi, Tsung-Yi Lin, Quoc V. Le. DropBlock: A regularization method for convolutional networks. 2018. *https://arxiv.org/abs/1810.12890.*

[61] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, Enhua Wu. Squeeze-and-Excitation Networks. 2019. *https://arxiv.org/pdf/1709.01507.pdf.*

[62] Diganta Misra. Mish:a self-regularized non-monotonic activation function. 2020. *https://arxiv.org/abs/1908.08681.*

[63] Mingxing Tan, Ruoming Pang, Quoc V. Le. Google Research. EfficientDet: Scalable and Efficient Object Detection.2020.*https://arxiv.org/pdf/1911.09070v7.pdf*

[64] Bochkovskiy, Alexey, Chien-Yao Wang, and Hong-Yuan Mark Liao. "YOLOv4: Optimal Speed and Accuracy of Object Detection.2020.*arXiv preprint arXiv:2004.10934*.

[65] YOLO5. *https://github.com/ultralytics/yolov5* .

[66] *https://roboflow.com/*

A Appendix

http://geopython.github.io/OWSLib/installation pip install geopandas

Retina Net

https://github.com/fizyr/keras-retinanet

YOLO5

https://colab.research.google.com/drive/1gDZ2xcTOgR39tGGs-EZ6i3RTs16wmzZQ