Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Казанский национальный исследовательский технический университет
им. А.Н. Туполева-КАИ»

Кафедра систем информационной безопасности

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе по дисциплине

«Нейросетевые модели и алгоритмы»

на тему «Обнаружение и распознавание дорожных знаков в видео DVR»

Выполнили:

Базаров Ильмурат, ст. гр. 4167

Назаров Равиль, ст. гр. 4167

Новикова Ксения, ст. гр. 4163

Петров Егор, ст. гр. 4167

Хабибуллоев Мухаммадсодик, ст. гр. 4163

Проверил:

д. т. н, профессор кафедры СИБ
_____________Аникин И.В.

Оценка _________________

«____»____________20__г.

Казань 2021

GERMAN-RUSSIAN INSTITUTE OF ADVANCED TECHNOLOGIES
TU-ILMENAU (GERMANY) AND KNRTU-KAI (RUSSIA)

Research in Computer & System Engineering Program
“Neural Networks” Course

Group Project

Road sign detection and recognition in DVR video

Submitted by
Bazarov Ilmurat
Nazarov Ravil
Novikova Ksenia
Petrov Egor
Khabibulloev Muhammadsodik

Supervised by
D.C.S.. Igor Anikin

Kazan, 2021

Table of contents

Abbreviations

ANN	Artificial Neural Network
CNN	Convolutional Neural Network
R-CNN	Region-based Convolutional Neural Network
mAP	Mean Average Precision

List of Figures

List of Tables

Algorithms and program code

Abstract

Nowadays, many road problems arise from the peculiarities of human thinking. This problem can be solved by automating transport. Unfortunately, this process cannot be done quickly due to a number of unresolved issues. Thus, a transitional stage in solving this problem can be automated systems that facilitate the driver's control. One of the most important functions for this is the ability to recognize road signs. In this paper, we compare the performance of several artificial neural networks for this purpose.

Keywords: Artificial Neural Network, road sign detection, road sign recognition, PyTorch, real-time traffic data.

1. Introduction

1.1. Motivation

Currently, unmanned vehicles, such as Yandex, Tesla and others, are undergoing great development. During driving one of the main components of ensuring safety such unmanned vehicles is real-time recognition of the traffic situation (a type of computer vision). Therefore, systems for automatic rapid recognition of road signs, which can be built into the control system of unmanned vehicles, are becoming relevant.

1.2. Problem statement

It is necessary to investigate the effectiveness of intelligent computer models based on artificial neural networks for automatic recognition of road signs of four main types (prohibitory, danger, mandatory, other) to select those that are most applicable for each type. For the research, neural networks of the following paradigms will be considered: RetinaNet R101, YOLO v4 and Faster R-CNN. These paradigms are special neural networks of computer vision, but their structure and capabilities are different. In this regard, it is necessary to compare them specifically for the task of recognizing road signs.

To achieve this goal the following subtasks have to be done:

Analysis of the possibilities, purpose and methods of implementation of each of the selected networks;

Selection of data sets for analyzing the effectiveness of neural networks;

Software implementation of neural networks of selected paradigms with training on selected data sets

Carrying out computational experiments on recognition of four types of road signs by designed neural networks;

Comparative analysis of the results obtained, selection of the most effective paradigm.

1.3. Thesis structure

The work has the following structure.

Chapter 1 contains the relevance and description of the problem, as well as the statement of the research problem.

Chapter 2 gives the background information about methods of road signs recognition and detection, gives information about different models (RetinaNet R101, YOLOv4 and Faster R-CNN) and comparison them between each other.

Chapter 3 presents the structured literature review process and the findings from the completed process.

Chapter 4 describes details of the dataset and implemented framework.

Chapter 5 contains evaluation part, results for each model and their comparison.

Finally, conclusion and further perspectives of the project are presented in chapter 6.

2. Background

Three deep learning algorithms were selected that are the most popular for the object detection task: Faster R-CNN, YOLOv4, and RetinaNet R101.

2.1. Faster R-CNN

This algorithm is a modification of Fast R-CNN, which, in turn, is a modified R-CNN. The common place for the whole group of algorithms is an attempt to search in a certain limited number of regions [1]. In the latter modification, a separate network is used to obtain region proposals, not a slow and time-consuming selective search. Also, for the last two modifications of the algorithms, it is common to use CNN at the beginning to build a special feature map.

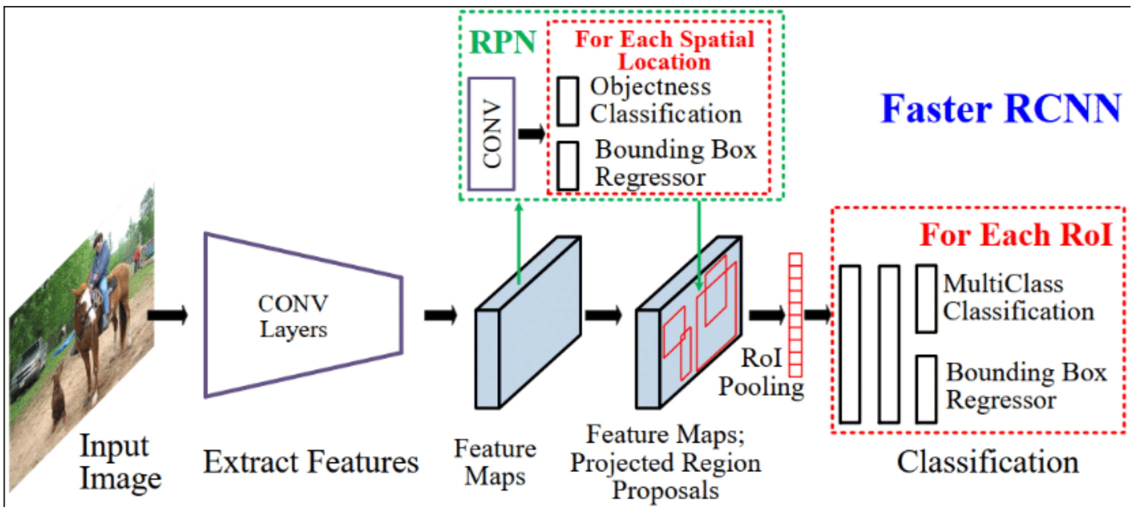


Figure 1. Structure of Faster R-CNN

2.2. YOLO

YOLO architecture accomplishes object detection via a fixed-grid regression [2]. The main feature of the YOLO is the application of CNN not to individual areas of the image, but to the entire image as a whole. A grid is superimposed on the picture and a certain number of bounding boxes are taken, for which the probability of the class is calculated. Those of them for which this probability exceeds a certain threshold value are used to determine the location of the imaged object.

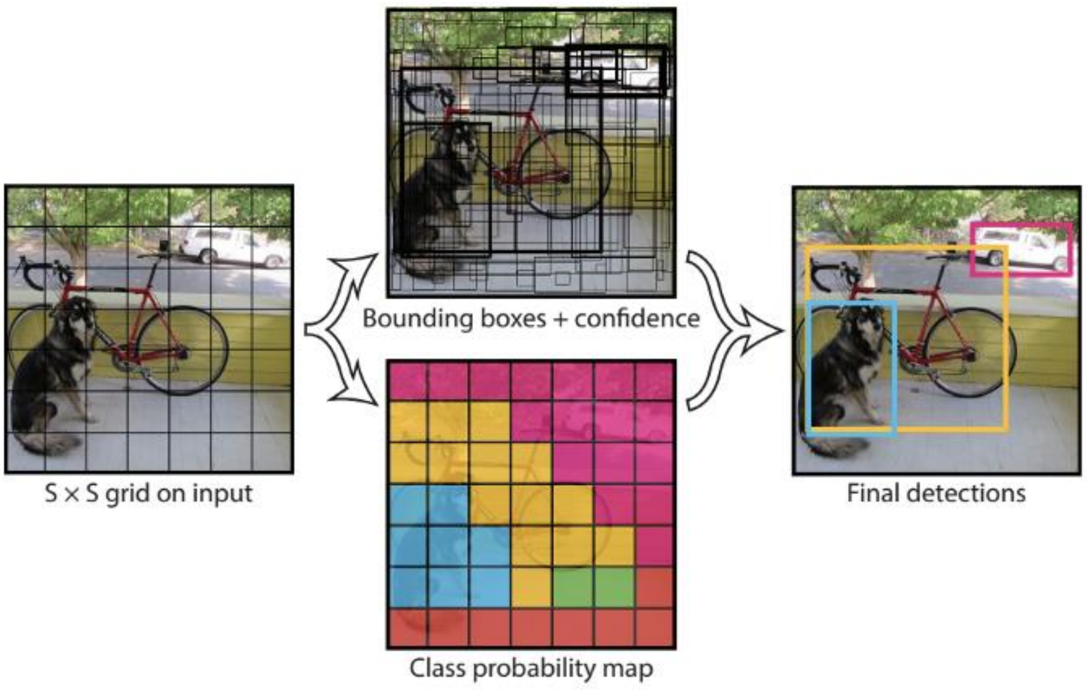


Figure 2. Structure of YOLO

2.3. RetinaNet

RetinaNet uses several networks: Backbone, Feature Pyramid Net, and two dedicated subnets. Backbone is used for feature extraction and is variable - neural networks of various architectures can be used as it. Feature Pyramid Net is a neural network designed to combine features of the upper and lower levels of a neural network. Two specialized subnets, Classification Subnet and Regression Subnet, are used to extract class information from FPNs, solving the classification problem, and coordinate information, solving the regression problem, respectively [3].

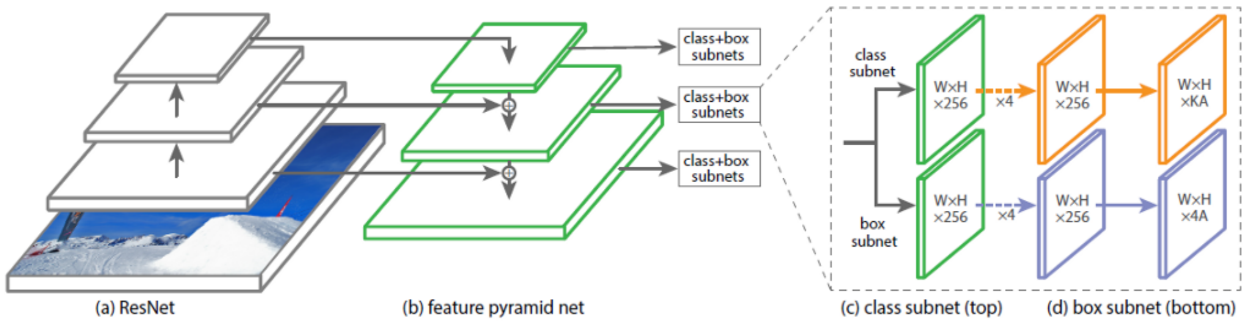


Figure 3. Structure of RetinaNet

3. State of the art

3.1. Recognition of road signs – classification of methods. General description of the process.

According to [4], it is possible to divide the methods of road sign recognition into methods used for detection and methods used for classification. Among the former categories are color-based, shape-based, and learning-based (which includes deep learning methods). Among the latter are those based on features that are given manually and deep learning methods. The process itself, if we combine the classification from [4] and [5], can be divided into the stages of preprocessing, feature extraction and object detection, classification. It should be noted that deep learning methods allow the entire recognition process to be carried out. Because of this fact, and also because deep learning methods are of the greatest interest for this work, they will be separated into a separate section. The rest of the methods will be considered only briefly.

3.2. Road sign detection methods

Following the steps in the recognition process, let's start with the preprocessing step. Since the resulting image is imperfect, it is necessary to correct it. There are four possible types of correction: sensor correction, noise removal, geometric correction, and color correction. An example of a specific method is image binarization.

For feature extraction and object detection, as described above, there are three categories of methods. Color-based methods can be quite effective for this task, as road signs are usually bright in color. Various color spaces such as RGB, HSV, HSI, YUV can be used. High-Contrast Region Extraction (HCRE) is a popular approach.

Although road signs are usually brightly colored, various factors such as distance and weather conditions can negatively affect the accuracy of the first category methods. Shape-based methods can be more accurate because they work with a grayscale image. On the other hand, they usually require more computational resources and can erroneously recognize characters of the same shape, which, when converted to gray tones, look the same. An example of a specific method is the directional gradient histogram.

Both of the previous categories of methods have common disadvantages, such as problems when working with images with zooms or rotations. Machine learning methods do not have such disadvantages, but they require a lot of annotated data for training. An example of methods of this kind is a cascade of Viola-Jones detectors trained using the AdaBoost algorithm, or a support vector machine.

For the classification based on manually defined features, various methods are used, which can use both the previously mentioned histogram of directional gradients or a cascade of Viola-Jones detectors, and, for example, local binary patterns.

3.3. Deep learning methods for road sign detection

Deep learning techniques can be used for both object detection and classification. However, in some works, deep learning is used for a specific purpose, which will be mentioned in the description of the method.

The most popular use of deep learning for object detection is using a convolutional neural network to extract deep visual features [6]. On the basis of the obtained features, the support vector machine is applied. Unfortunately, the accuracy of this method is 71.42%. But with the modification of this method, the accuracy can be increased to 98.86%. It uses a convolutional neural network with three sets of convolutional layers with max pooling layers in each set and two fully-connected fully connected layer. At the end, before the output, Softmax is used.

When used to classify objects, as a rule, the characteristics must already be extracted. For example, when applying this approach with the CNN architecture LeNet-5, which consists of two sets of convolutional layers with the ReLu activation function and pulling layers. The accuracy of this method is 97%. With such an architecture, it is possible, after extracting features, to use them to train the convolutional neural network step by step, that is, when features from the first stage are transmitted together with features of the second stage, which allows increasing the accuracy to 97.17%.

Also, convolutional neural networks can be used for complete recognition. For example, for a convolutional neural network with three sets of convolutional layers with pulling layers and two fully connected layers, the accuracy was 98.83%. To recognize road signs with inscriptions, two convolutional neural networks were used, one of which detected signs and regions of interest, and the second detected inscriptions. The accuracy of this method has reached 90%.

There are also so-called Fast Branch CNNs, for which it is not necessary to recognize the entire road sign, a partial image or only the shape of the sign may be enough. The architecture used in this model contains three sets of convolutional layers with pulling and normalizing layers. Accuracy - 98, 52%.

4. Development. System design.

This chapter is intended to give information about dataset and the implementation part using PyTorch.

4.1. Dataset

A prepared Traffic Signs Dataset in YOLO format for Detection tasks from the Kaggle website was used in the work [7]. It can be used for training as well as for testing. Dataset consists of images in *.jpg format and *.txt files next to every image that have the same names as images files have. These *.txt files include annotations of bounding boxes of Traffic Sings in the YOLO format:

[Class Number] [center in x] [center in y] [Width] [Height]

For Faster-RCNN and RetinaNet, this dataset was converted into their corresponding format. Roboflow website gives opportunity convert the annotations into COCO format which will be used in our models.

For example, file 00001.txt includes three bounding boxes (each in a new line) that describe three Traffic Signs in 00003.jpg image:

0 0.5540441176470589 0.568125 0.016911764705882352 0.02875

0 0.5536764705882353 0.596875 0.016176470588235296 0.02875

1 0.5536764705882353 0.534375 0.023529411764705882 0.03875

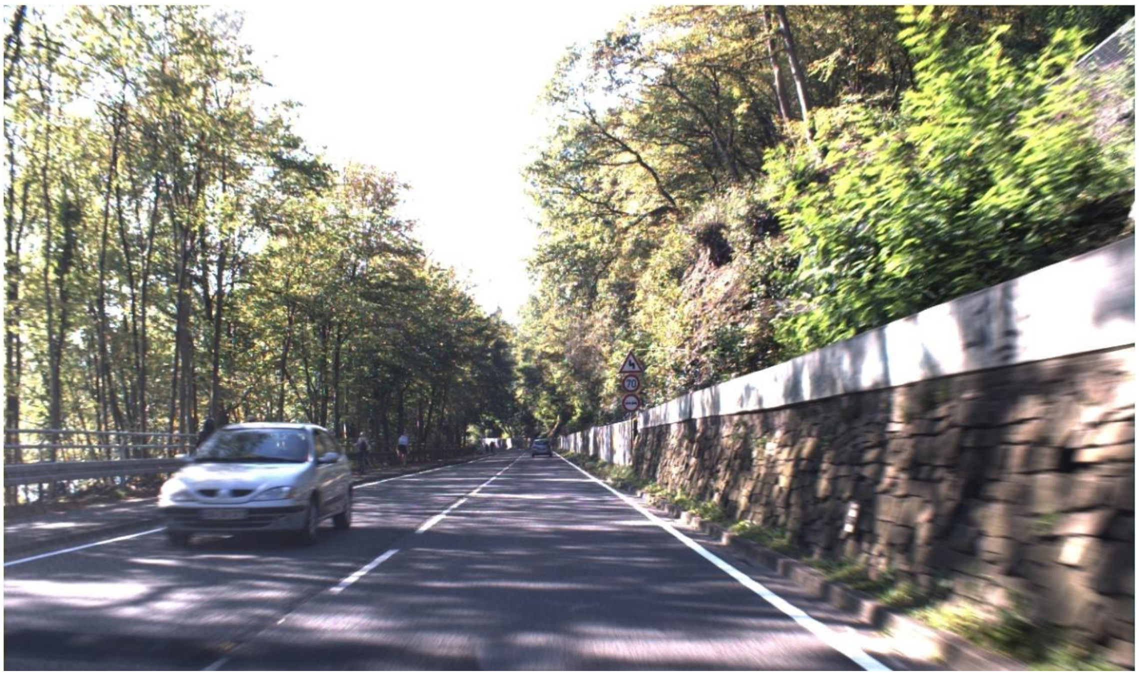


Figure 4. Appearance of file 00003.jpg

Traffic Sins in this Dataset are grouped into four classes: *prohibitory*, *danger*, *mandatory*, *other*

Class 'Prohibitory' consists of following Traffic Signs: speed limit, no overtaking, no traffic both ways, no trucks.

Class 'Danger' consists of following Traffic Sings: priority at next intersection, danger, bend left, bend right, bend, uneven road, slippery road, road narrows, construction, traffic signal, pedestrian crossing, school crossing, cycles crossing, snow, animals.

Class 'Mandatory' consists of following Traffic Sings: go right, go left, go straight, go right or straight, go left or straight, keep right, keep left, roundabout.

Class 'Other' consists of following Traffic Sings: restriction ends, priority road, give way, stop, no entry.

4.2. Implementation

Stack of technologies: Python 3.6, Pytorch, Detectron 2.

Full implementation, results and training graphs are in Github.[8]

RetinaNet R101, YOLOv4 and Faster-RCNN were used for training and making predictions in this work. Dataset is ready to be used in training on our models. The diagram below shows the workflow the implementation.

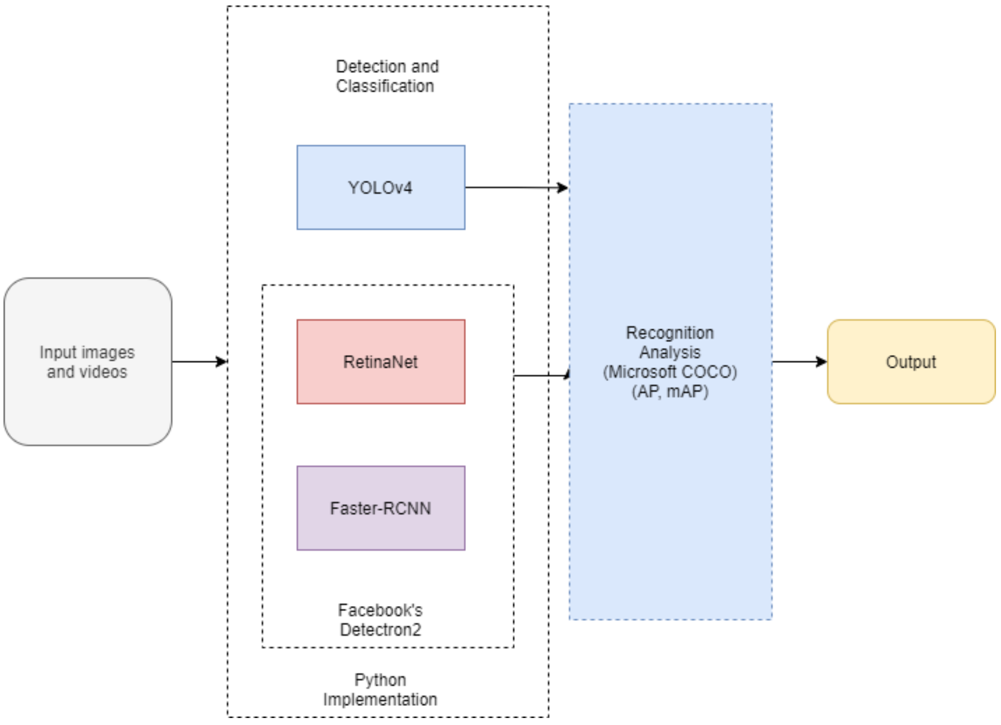
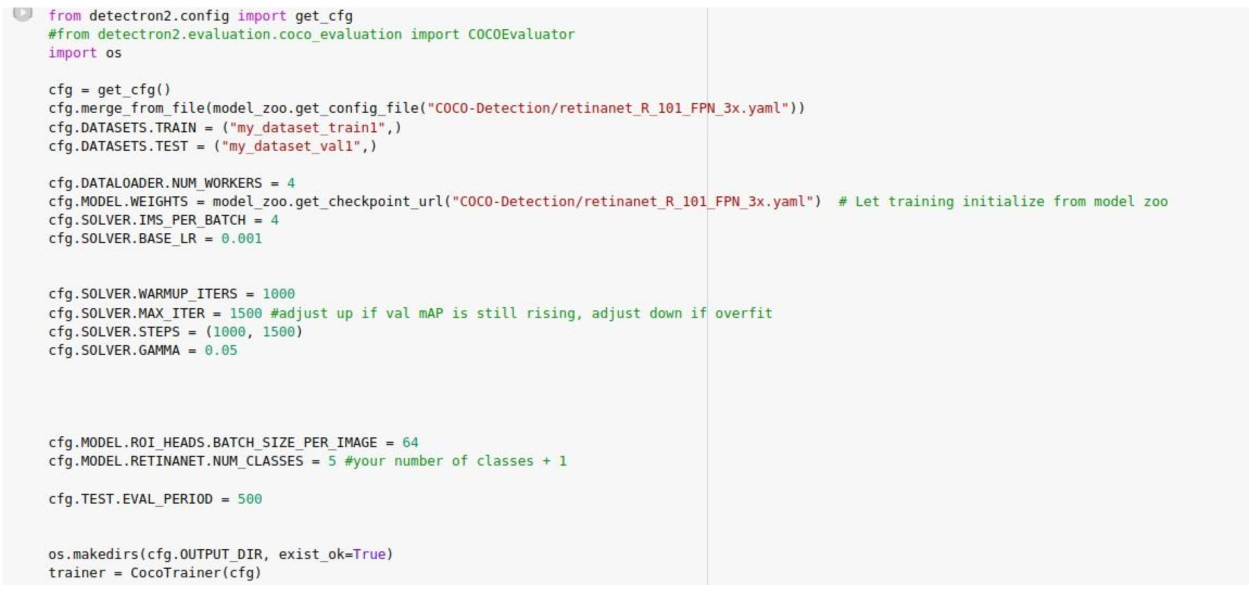


Figure 5: Workflow diagram

General idea of the work is that we fed the input image into the object recognition model to detect the traffic signs and classify them according to our given class names in our dataset. YOLOv4 was implemented using its main object detection system. Custom datasets were annotated by the required format, main configurations were configured and the dataset was fed into our object detection system. RetinaNet and Faster-RCNN methods were implemented using Facebook's Detectron2 object detection and segmentation platform. It allows to configure so that segmentation is dropped and only object detection is used (bounding box). After transforming dataset into corresponding format (MS COCO), models were trained. All the obtained weights were saved for further testing in video.

All models use multi-process data loading by setting the argument NUM_WORKERS to 4. Number of iterations are 1000. The value of NUM_CLASSES is number of existing classes plus one. In this case it is 5.

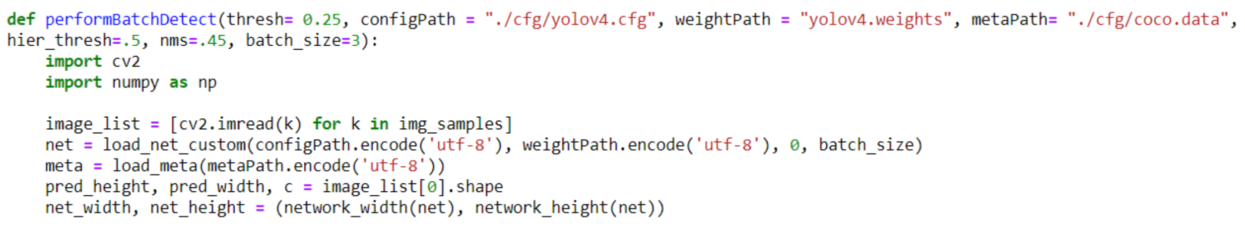
RetinaNet is a one-stage object detection model that uses Feature Pyramid Network (FPN) and utilizes a focal loss function to address class imbalance during training.



```
from detectron2.config import get_cfg
#from detectron2.evaluation.coco_evaluation import COCOEvaluator
import os

cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-Detection/retinanet_R_101_FPN_3x.yaml"))
cfg.DATASETS.TRAIN = ("my_dataset_train1",)
cfg.DATASETS.TEST = ("my_dataset_val1",)

cfg.DATALOADER.NUM_WORKERS = 4
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-Detection/retinanet_R_101_FPN_3x.yaml")  # Let training initialize from model zoo
cfg.SOLVER.IMS_PER_BATCH = 4
cfg.SOLVER.BASE_LR = 0.001


cfg.SOLVER.WARMUP_ITERS = 1000
cfg.SOLVER.MAX_ITER = 1500 #adjust up if val mAP is still rising, adjust down if overfit
cfg.SOLVER.STEPS = (1000, 1500)
cfg.SOLVER.GAMMA = 0.05



cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 64
cfg.MODEL.RETINANET.NUM_CLASSES = 5 #your number of classes + 1

cfg.TEST.EVAL_PERIOD = 500


os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
trainer = CocoTrainer(cfg)
```

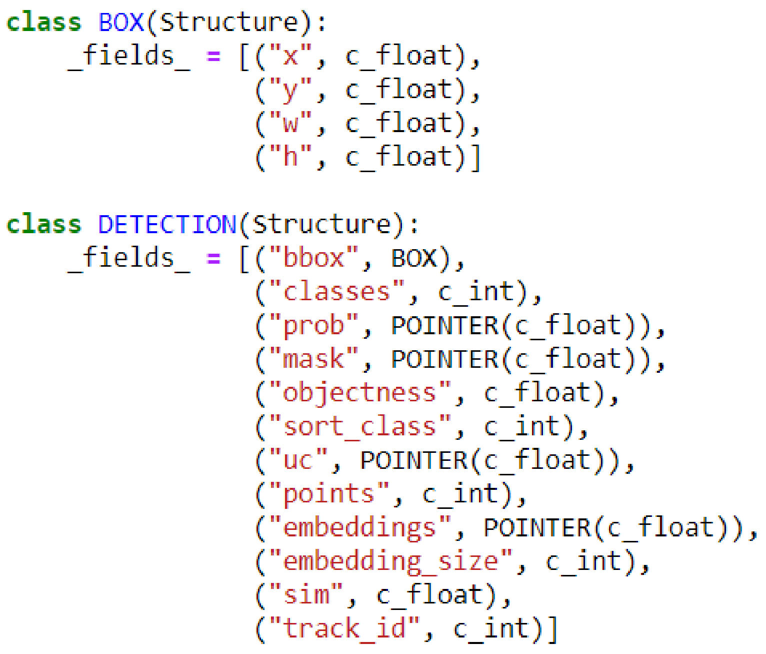
Algorithm 1. Implementation for RetinaNet R101

The main feature of the YOLO architecture is the application of CNN not to individual areas of the image, but to the entire image as a whole. The network divides the image into regions and predicts bounding boxes and probabilities for regions.



```
def performBatchDetect(thresh= 0.25, configPath = "./cfg/yolov4.cfg", weightPath = "yolov4.weights", metaPath= "./cfg/coco.data",
hier_thresh=.5, nms=.45, batch_size=3):
    import cv2
    import numpy as np

    image_list = [cv2.imread(k) for k in img_samples]
    net = load_net_custom(configPath.encode('utf-8'), weightPath.encode('utf-8'), 0, batch_size)
    meta = load_meta(metaPath.encode('utf-8'))
    pred_height, pred_width, c = image_list[0].shape
    net_width, net_height = (network_width(net), network_height(net))
```

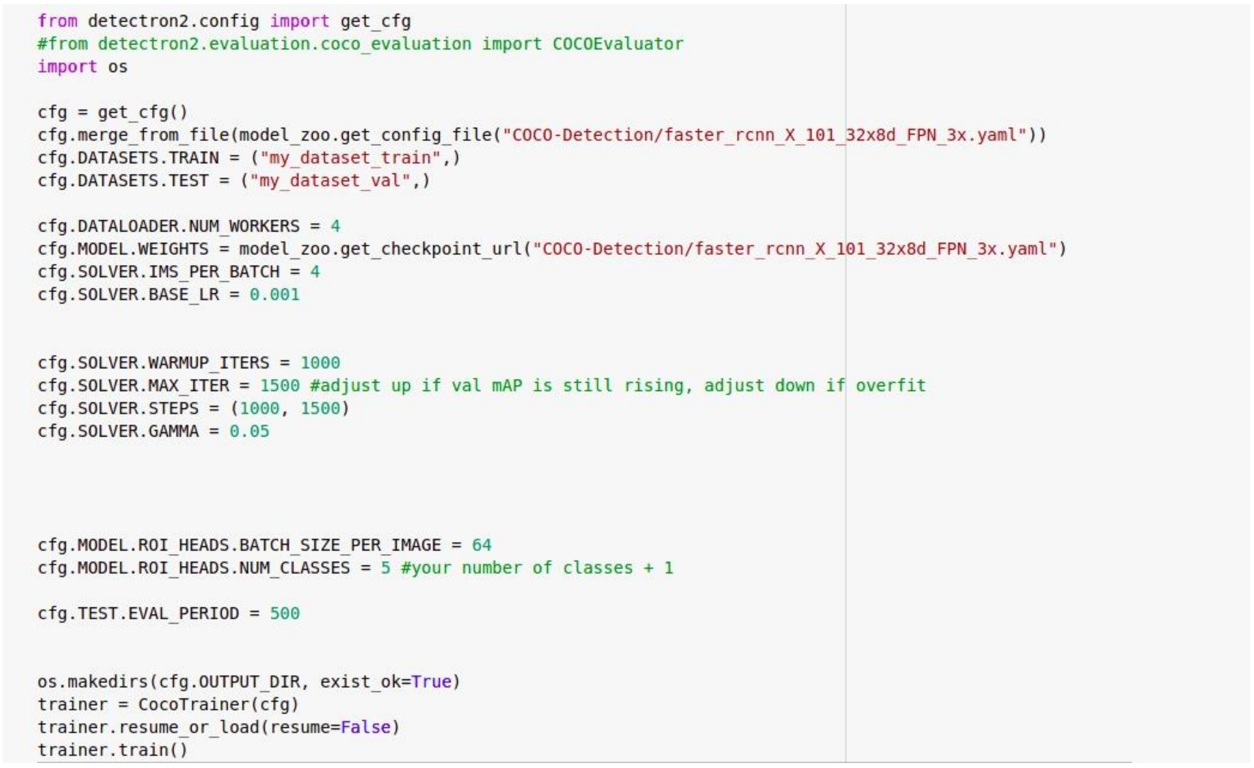
Algorithm 2. Implementation for YOLO v4



```
class BOX(Structure):
    _fields_ = [("x", c_float),
                ("y", c_float),
                ("w", c_float),
                ("h", c_float)]

class DETECTION(Structure):
    _fields_ = [("bbox", BOX),
                ("classes", c_int),
                ("prob", POINTER(c_float)),
                ("mask", POINTER(c_float)),
                ("objectness", c_float),
                ("sort_class", c_int),
                ("uc", POINTER(c_float)),
                ("points", c_int),
                ("embeddings", POINTER(c_float)),
                ("embedding_size", c_int),
                ("sim", c_float),
                ("track_id", c_int)]
```

Algorithm 3. Description of BOX and DETECTION classes (YOLO v4)

Faster R-CNN makes further progress than Fast R-CNN. Search selective process is replaced by Region Proposal Network (RPN). As the name revealed, RPN is a network to propose regions. For instance, after getting the output feature map from a pre-trained model (VGG-16).



```
from detectron2.config import get_cfg
#from detectron2.evaluation.coco_evaluation import COCOEvaluator
import os

cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-Detection/faster_rcnn_X_101_32x8d_FPN_3x.yaml"))
cfg.DATASETS.TRAIN = ("my_dataset_train",)
cfg.DATASETS.TEST = ("my_dataset_val",)

cfg.DATALOADER.NUM_WORKERS = 4
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-Detection/faster_rcnn_X_101_32x8d_FPN_3x.yaml")
cfg.SOLVER.IMS_PER_BATCH = 4
cfg.SOLVER.BASE_LR = 0.001


cfg.SOLVER.WARMUP_ITERS = 1000
cfg.SOLVER.MAX_ITER = 1500 #adjust up if val mAP is still rising, adjust down if overfit
cfg.SOLVER.STEPS = (1000, 1500)
cfg.SOLVER.GAMMA = 0.05




cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 64
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 5 #your number of classes + 1

cfg.TEST.EVAL_PERIOD = 500


os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
trainer = CocoTrainer(cfg)
trainer.resume_or_load(resume=False)
trainer.train()
```

Algorithm 4. Implementation for Faster R-CNN

5. Evaluation and Experiments

There are 741 images in dataset. For training used 518 images, for validation during training used 148 images and 74 images for test.

5.1. Results for RetinaNet R101

Overall evaluation result for RetinaNet R101 is mean average precision (mAP) = 56.044%

Average precision for each class:

Table 1. Results for RetinaNet R101

Class	0 (prohibitory)	1 (danger)	2 (mandatory)	3(other)
AP	63.602%	60.835%	48.115%	51.625%

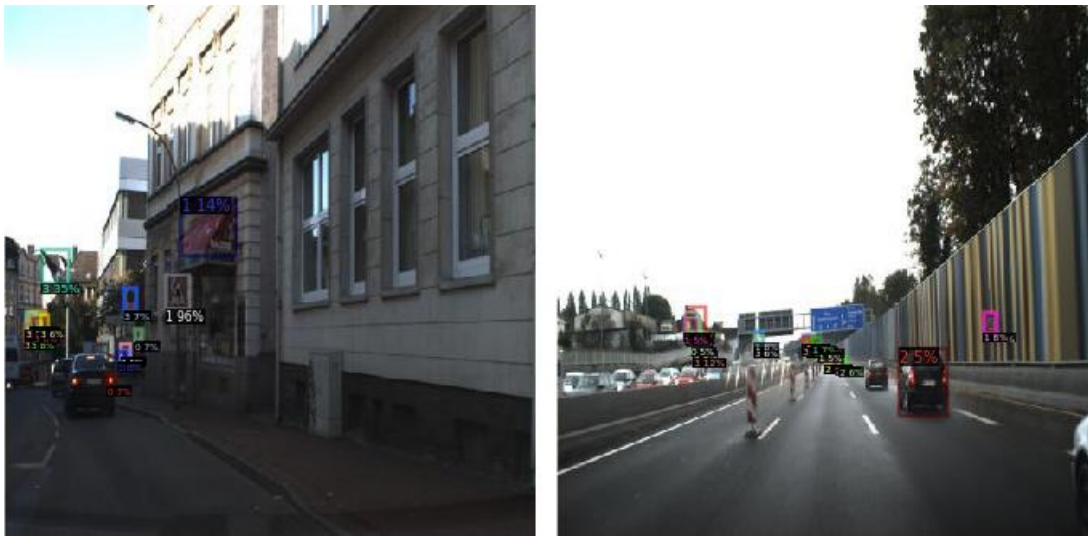


Figure 5. Example of recognized images (RetinaNet R101)

5.2. Results for YOLO v4

Overall evaluation result for YOLO v4 is mean average precision (mAP) = 95.500%

Average precision for each class:

Table 2. Results for YOLO v4

Class	0 (prohibitory)	1 (danger)	2 (mandatory)	3(other)
AP	98.497%	96.085%	92.105%	95.313%

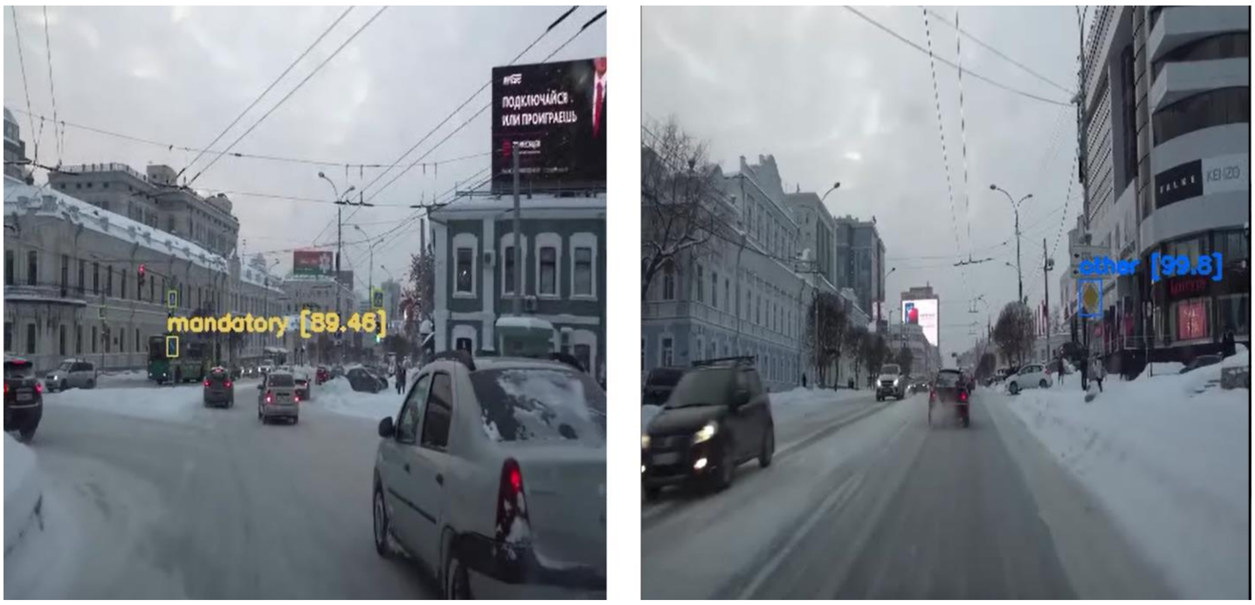


Figure 6. Example of recognized image (YOLO v4)

5.3. Results for Faster R-CNN

Overall evaluation result for Faster R-CNN is mean average precision (mAP) = 70.912%

Average precision for each class:

Table 3. Results for Faster R-CNN

Class	0 (prohibitory)	1 (danger)	2 (mandatory)	3(other)
AP	72.030%	71.537%	67.927%	72.153%

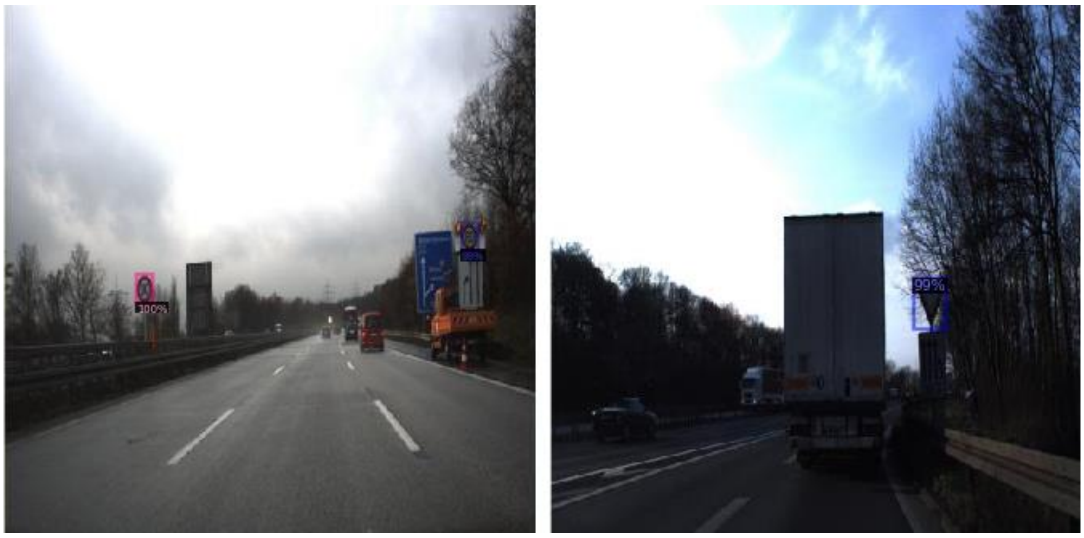


Figure 7. Example of recognized images (Faster R-CNN)

5.4. Comparison of results

Class 0 – ‘prohibitory’, Class 1 – ‘danger’, Class 2 – ‘mandatory’, Class 3 – ‘other’

Table 4. Comparison of results

	RetinaNet R101	YOLO v4	Faster R-CNN
mAP	56.044%	**95.500%**	70.912%
AP for class 0	63.602%	**98.497%**	72.030%
AP for class 1	60.835%	**96.085%**	71.537%
AP for class 2	*48.115%*	***92.105%***	*67.927%*
AP for class 3	51.625%	**95.313%**	72.153%

The best result was shown by the YOLO v4 with a result of mean average precision (mAp) = 95.500%.

The worst was shown by the RetinaNet R101 with a result of mean average precision (mAp) = 56.044%.

The worst result of mean average precision for all neural network models was for class 2 (‘mandatory’).

6. Conclusion and Perspectives

During this work the main software tools for the practical implementation of the models were identified based on a review of existing implementations of artificial neural networks. Also a prepared dataset for road sign recognition was found.

RetinaNet R101, YOLO v4 and Faster R-CNN models were selected and trained on a dataset containing 741 images, 518 of which were used for training.

The best result was shown by the YOLO v4 model with a result of 95%.

In the future, work can be continued in the direction of improving the quality of recognition, as well as expanding the recognized classes. Thus, recognizing not only the categories of signs, but every existing road sign.

References

1. S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards realtime object detection with region proposal networks,” in NIPS, 2015,pp. 91–99.
2. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in CVPR, 2016.
3. T.Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. “Focal loss for dense object detection.” arXiv preprint, arXiv:1708.02002, 2017
4. S. Hussain, M. Abualkibash and S. Tout, "A Survey of Traffic Sign Recognition Systems Based on Convolutional Neural Networks," *2018 IEEE International Conference on Electro/Information Technology (EIT)*, Rochester, MI, 2018, pp. 0570-0573, doi: 10.1109/EIT.2018.8500182.
5. B. Sanyal, R. K. Mohapatra and R. Dash, "Traffic Sign Recognition: A Survey," 2020 International Conference on Artificial Intelligence and Signal Processing (AISP), Amaravati, India, 2020, pp. 1-6, doi: 10.1109/AISP48273.2020.9072976.
6. Saadna, Y., Behloul, A. An overview of traffic sign detection and classification methods. Int J Multimed Info Retr 6, 193–210 (2017)
7. Kaggle Inc.URL: https://www.kaggle.com/valentynsichkar/traffic-signs-dataset-in-yolo-format (date of access to the site: 20.01.2021)
8. Github url with models and files: https://github.com/ilmb/BasicsOf_ANN_CourseWork-GRIAT20-21-