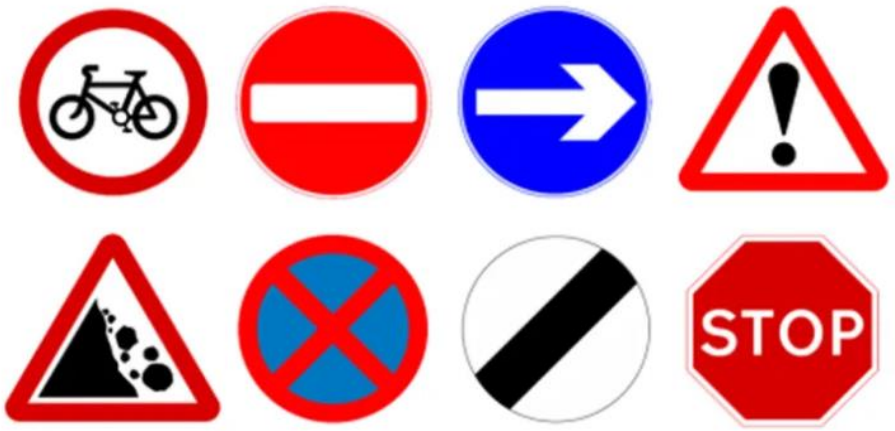
Deep Learning for Large-Scale Traffic-Sign Detection and Recognition

Domen Tabernik and Danijel Skocaj

Course: 1 סמינר במערכות תבוניות-65017

Student: Rakhlevski Ilia

Introduction



Proper management of traffic-sign inventory is an important task in ensuring safety and efficiency of the traffic flow.

Automatic detection and recognition of traffic signs plays a crucial role in management of the traffic-sign inventory. It provides accurate and timely way to manage traffic-sign inventory with a minimal human effort.

Goal of the article

In this paper, the authors address the issue of detecting and recognizing a large number of traffic-sign categories suitable for automating traffic-sign inventory management.
They adopted a convolutional neural network (CNN) approach, the Mask R-CNN [1], to address the full pipeline of detection and recognition with automatic end-to-end learning.
Several improvements were proposed to improve performance. Also, a novel dataset was created.

Existing solutions

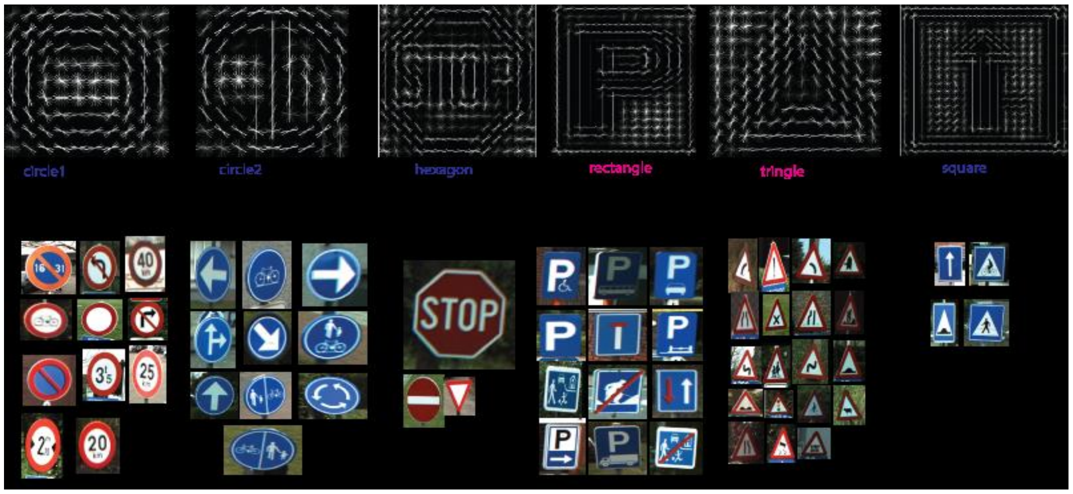
In the computer vision community, the recognition and detection of traffic signs is a well-researched problem and excellent detection and recognition algorithms have already been proposed [2], [3], [4], [5].
But these solutions have been designed only for a small number of categories.
Part of them focused only on traffic-sign recognition (TSR) and ignored the much more complex problem of traffic-sign detection (TSD) where finding accurate location of traffic sign is needed.
Other solutions that do address TSD mostly cover only a subset of traffic-sign categories.

Example of existing solution

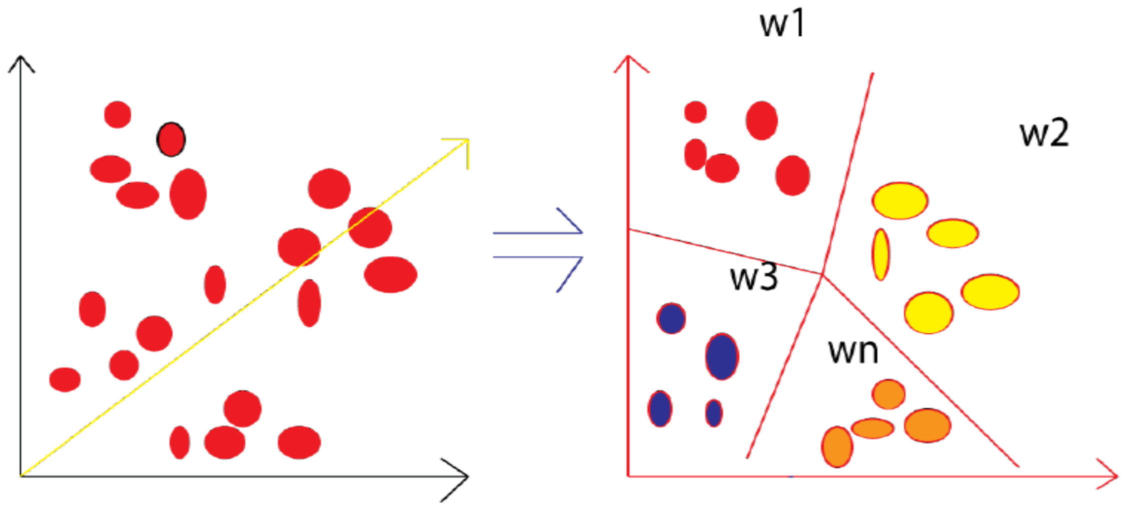
Each of traffic sign category can be assumed very well as combination of one or more topics [2].

For classification of the images, we have used two step method, in first processing we will find the shape of the traffic signs and after that we will classify its actual class.

First step: Shapes Finding using HOG (Histogram Oriented Gradients) [6]:



Second step: Sign Classification using SVM, KNN, k-means:



Method

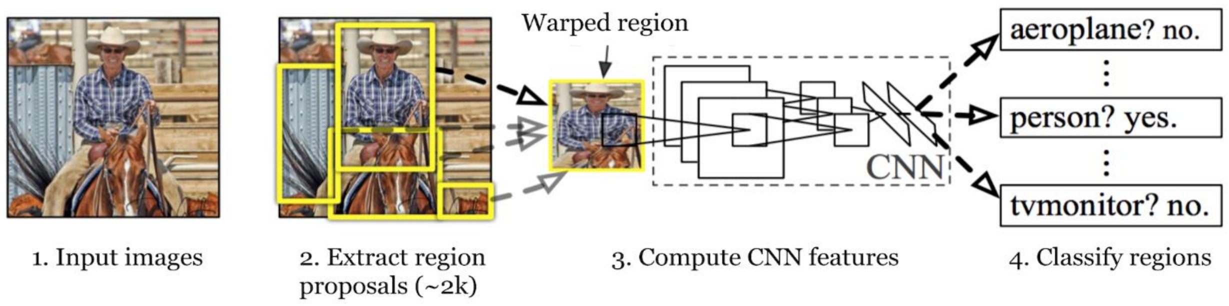
The method is described in the article is based on Mask R-CNN [1]. Mask R-CNN extends Faster R-CNN [7].

Also, some changes in the architecture and data preparation were made.

R-CNN

R-CNN [8] is short for "Region-based Convolutional Neural Networks". The main idea is composed of two steps.

- First, using Selective Search [9] or some other region proposal algorithm, it identifies a manageable number of bounding-box object region candidates ("region of interest" or "RoI").
- And then it extracts CNN features from each region independently for classification.



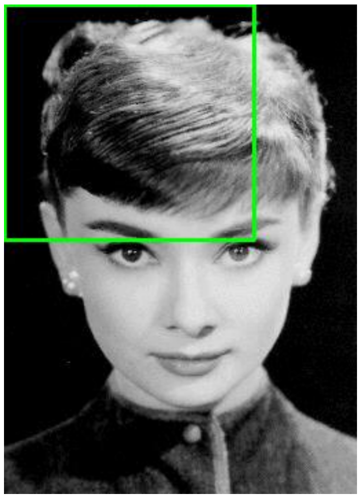
Each image needs to classify *2000* region proposals. So, it takes a lot of time to train the network. It requires *49* seconds to detect the objects in an image on GPU. To store the feature map of the region proposal, lots of disk space is also required.

Region proposal

Region proposals are just smaller parts of the original image, that we think could contain the objects we are searching for. There are different region proposal algorithms we can choose from. Examples:

Sliding window:

A sliding window is a rectangular region of fixed width and height that “slides” across an image. When using this method, you just go over the whole image with different sized rectangles and look at those smaller images in a brute-force-method. The problem is you will have a giant number of smaller images to look at.



Selective Search:

Selective Search [9] is a region proposal algorithm used in object detection. It is designed to be fast with a very high recall. It is based on computing hierarchical grouping of similar regions based on *color*, *texture*, *size* and *shape* compatibility.

Color of the cats is different but textures are same.

Chameleon and grass.

Colors are similar but textures are different.

Car and wheels. There no color or texture similarity. But there is size and shape similarity between wheel and fender (wing).

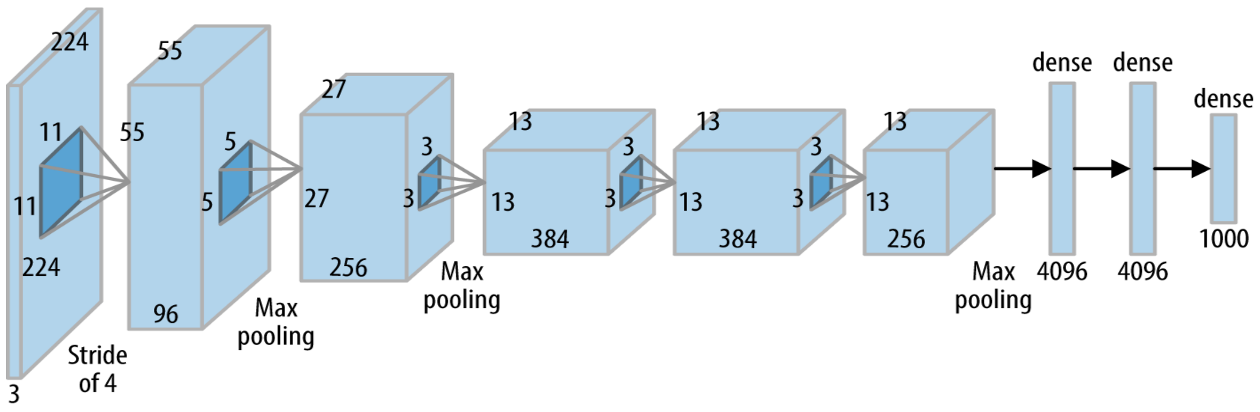
Selective Search algorithm takes performs the following steps:

1. Initialize proposal regions
2. Add all bounding boxes corresponding to segmented parts to the list of regional proposals
3. Group adjacent segments based on similarity
4. Go to step 2

At each iteration, larger segments are formed and added to the list of region proposals. Hence, we create region proposals from smaller segments to larger segments in a bottom-up approach.

CNN

In the next step, we take each region proposal and we will create a feature vector representing this image in a much smaller dimension using a Convolutional Neural Network (CNN). They use the AlexNet architecture [12] as a feature extractor.



SVM

At the previous step we created feature vectors from the image proposals. Now we will need to classify those feature vectors. We want to detect what class of object those feature vectors represent. For this purpose, we use a Support Vector Machine classifier. We have one SVM for each object class and we use them all.

Fast R-CNN

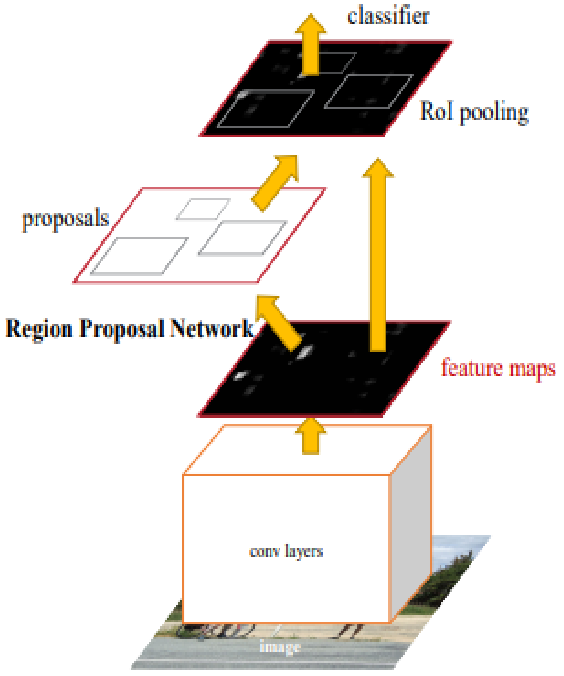
In R-CNN we passed each region proposal one by one in the CNN architecture and selective search generated around *2000* region proposal for an image. So, it is computationally expensive to train and even test the image using R-CNN.

To deal with this problem Fast R-CNN [10] was proposed, it takes the whole image and region proposals as input in its CNN architecture in one forward propagation.

Fast R-CNN drastically improves the training *(8.75 hrs vs 84 hrs)* and detection time from R-CNN. It also improves Mean Average Precision (mAP) [22] marginally as compare to R-CNN. Most of the time taken by Fast R-CNN during detection is a selective search region proposal generation algorithm.

Faster R-CNN

The bottleneck of the architecture Fast R-CNN is Selective Search. Since it needs to generate *2000* proposals per image. In Faster R-CNN [11], it was replaced by the region proposal network.



First of all, we pass the image into the backbone network. This backbone network generates a convolution feature map.

These feature maps are then passed into the region proposal network. The region proposal network takes a feature map and generates the anchors (the center of the sliding window with a unique size and scale).

These anchors are then passed into the classification layer (which classifies that there is an object or not) and the regression layer (which localize the bounding box associated with an object).

Comparison Between R-CNN, Fast R-CNN and Faster R-CNN

In terms of Detection time, Faster R-CNN is faster than both R-CNN and Fast R-CNN. The Faster R-CNN also has better mAP than both the previous ones.

	R-CNN	Fast R-CNN	Faster R-CNN
Method For Generating Region Proposals	Selective Search	Selective Search	Region Proposal Network

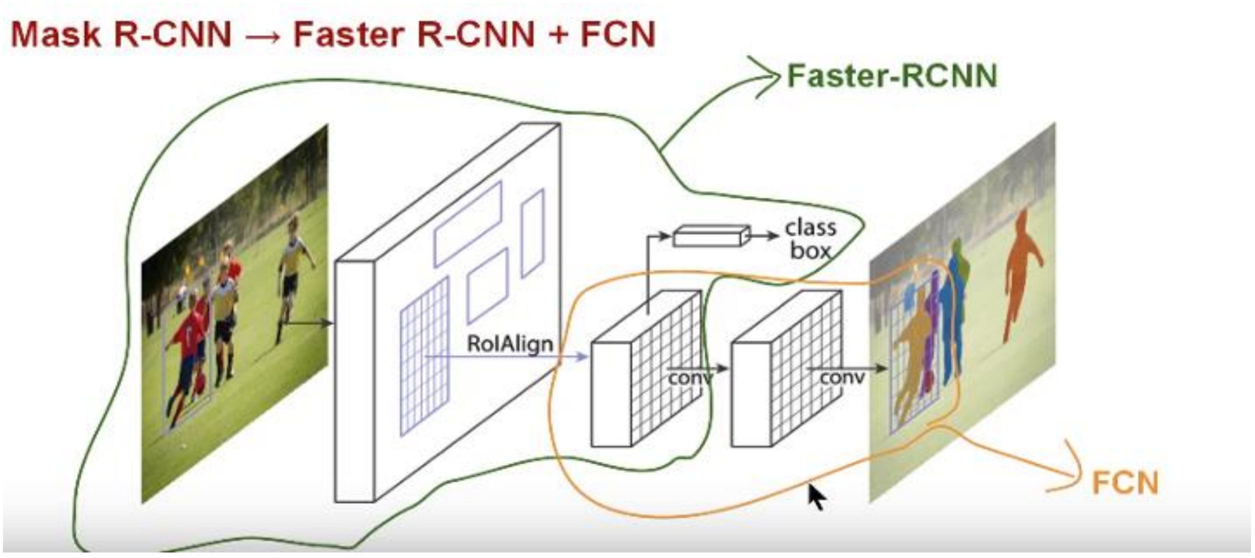
The mAP on Pascal VOC 2007 test dataset(%)	58.5	66.9 (when trained with VOC 2007 only)
		70.0 (when trained with VOC 2007 and 2012 both)
			69.9(when trained with VOC 2007 only)
			73.2 (when trained with VOC 2007 and 2012 both)
			78.8(when trained with VOC 2007 and 2012 and COCO)

The mAP on Pascal VOC 2012 test dataset (%)	53.3	65.7 (when trained with VOC 2012 only) 68.4 (when trained with VOC 2007	67.0(when trained with VOC 2012 only) 70.4 (when trained with VOC 2007

		and 2012 both)	and 2012 both) 75.9(when trained with VOC 2007 and 2012 and COCO)
Detection Time (sec)	~ 49 (with region proposal generation)	~ 2.32(with region proposal generation)	0.2 (with VGG), 0.059 (with ZF)

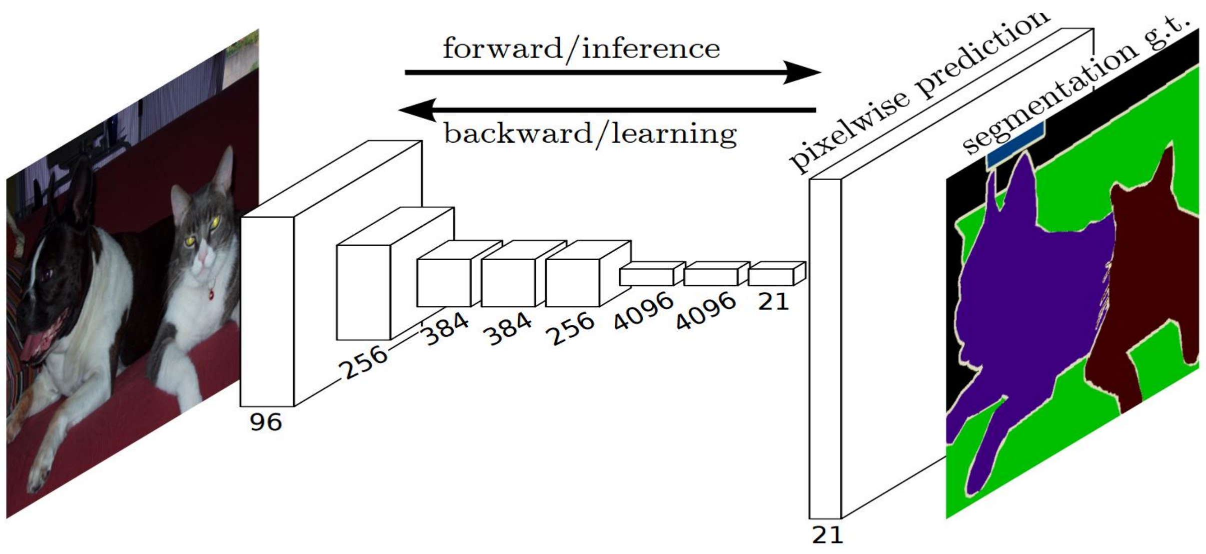
Mask R-CNN

Mask R-CNN [1] extends Faster R-CNN to pixel-level image segmentation. It was added a third branch for predicting an object mask in parallel with the existing branches for classification and localization. The mask branch is a small *Fully Convolution Network* [13] applied to each RoI, predicting a segmentation mask in a pixel-to-pixel manner.



Fully Convolutional Network (FCN)

FCN is a popular algorithm for doing semantic segmentation. This model uses various blocks of convolution and max pool layers to first decompress an image. It then makes a class prediction at this level of granularity. Finally, it uses up sampling and deconvolution layers to resize the image to its original dimensions.



Improvements

The authors extended the architecture with several improvements.

1) Online Hard Example Mining (OHEM):
The core idea of the algorithm is to filter according to the loss of the input samples, filter out the hard example, which indicates the samples that have a great influence on the classification and detection.
In this approach, they replaced random selection of ROIs with the selection based on their classification loss value.
Regions are sorted based on their loss value and only ones with high enough loss are passed to the classification learning module.
This ensures learning on samples on which the network was mistaken the most, i.e., on hard examples.

2) Sample weighting:
   There are often many more background regions, since most traffic signs in images are small and only a few region proposals exist for those traffic signs.
   The learning process will observe background objects more often and will focus on learning the background instead of on the foreground.
   They address this problem with smaller weights for the background regions, which forces the network to learn foreground objects first.

3) Distribution of selected training samples:
   Originally, the Mask R-CNN selects ROIs randomly. This is done separately for foreground and background.
   However, when many small and large objects are present in the image at the same time the random selection introduces imbalance into the learning process.
   The imbalance arises due to large objects having a large number of ROIs that cover it, while small objects having only a small number of ROIs, this causes the larger objects will be observed more often and favored much more than the smaller ones.
   They fixed this by selecting the same number of ROIs for each object present in the image.

4) Adjusting region pass-through during detection:
   They changed the number of ROIs passed from the Region Proposal Network to the classification network during the detection stage.
   The number of regions passed through need to be adjusted due to a large number of small objects that are commonly present in the traffic-sign domain.

Dataset

The dataset that was chosen for this approach is DFG Traffic Sign Dataset [14].

The dataset consists of *200* traffic sign categories captured in Slovenian roads spanning in around *7000* high-resolution images.

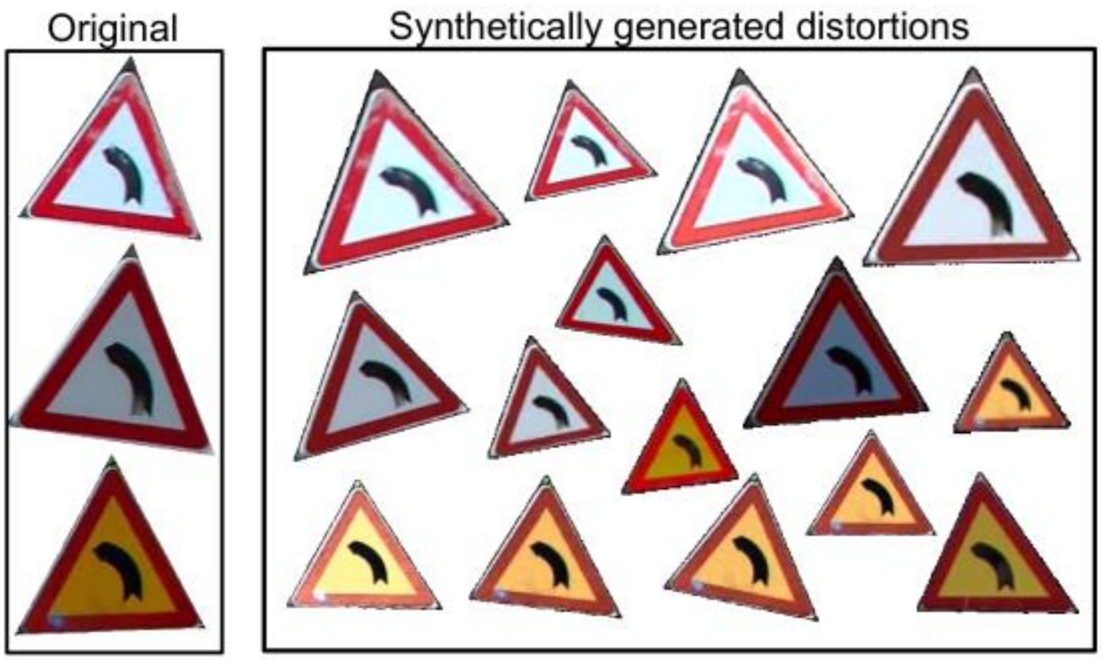
Example:

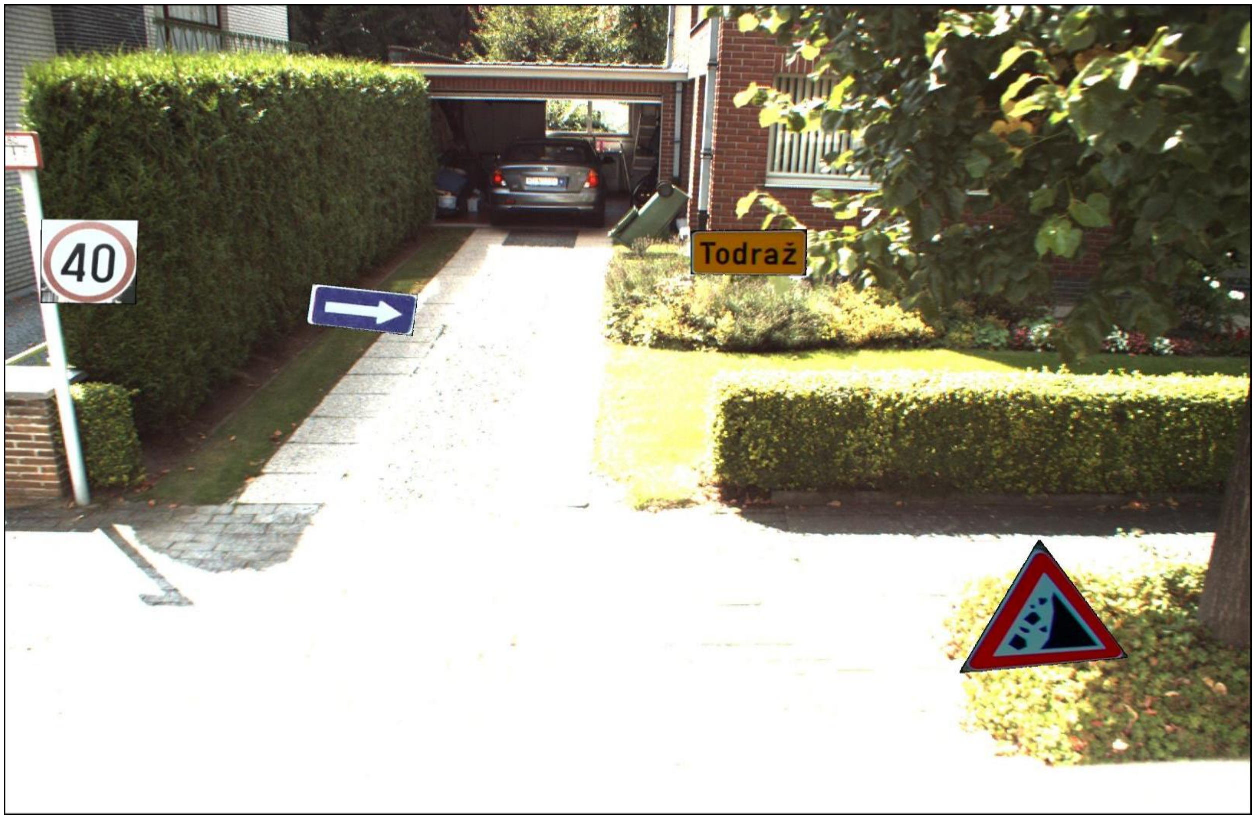
Data augmentation

Data augmentation is a technique to artificially create new training data from existing training data. Due to millions of learnable parameters the system becomes undetermined without a sufficient number of training samples.

They use synthetic traffic-sign instances are created by modifying segmented, real-world training samples.

Examples:





Implementation

- The project is implemented [15] using the Detectron framework [16] that is Facebook AI Research's software system that implements state-of-the-art object detection algorithms.

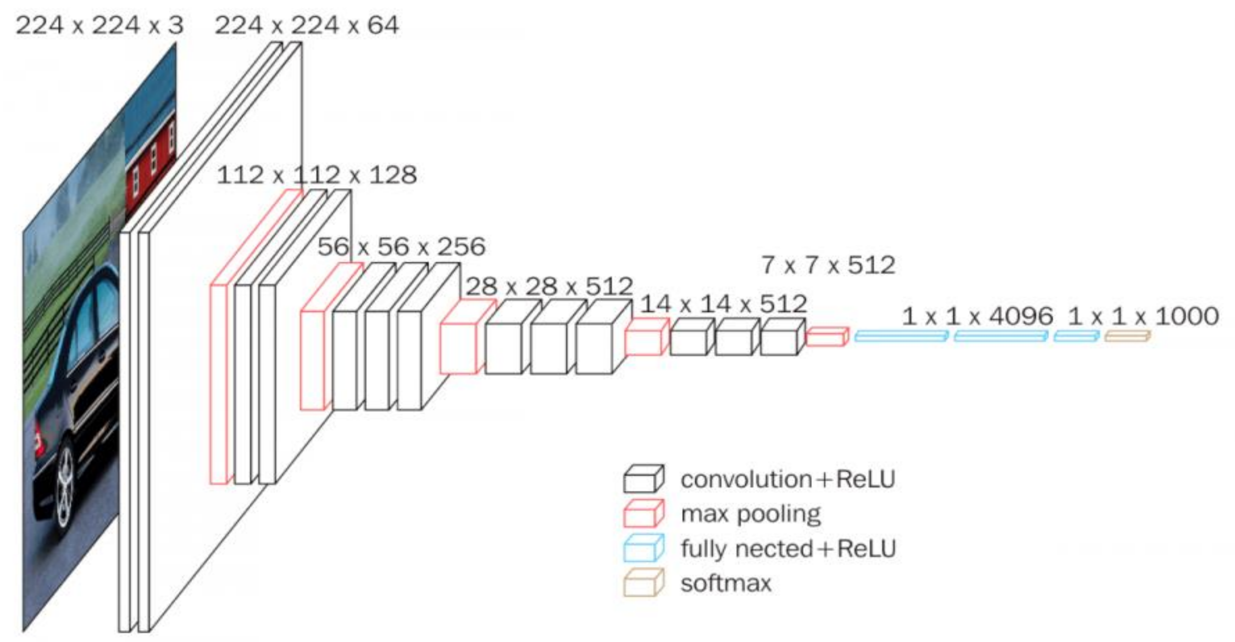
- Using Cafee2 [17] – deep learning framework.

- Programming language is Python.

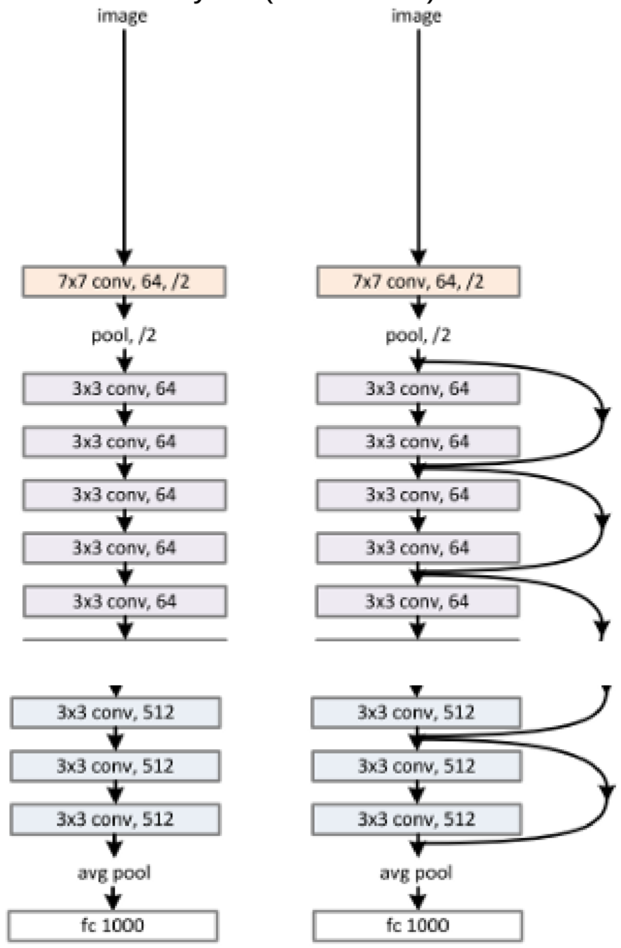
- Implementations were created for both Faster and Mask R-CNN.

- For the Faster R-CNN, they employed the VGG16 [18] network with 13 convolutional layers and 3 fully-connected layers.

  VGG16 – Convolutional Network for Classification and Detection:



- For the Mask R-CNN, they employed a residual network [19] with 50 convolutional layers (ResNet-50):



mAP (mean Average Precision) for Object Detection

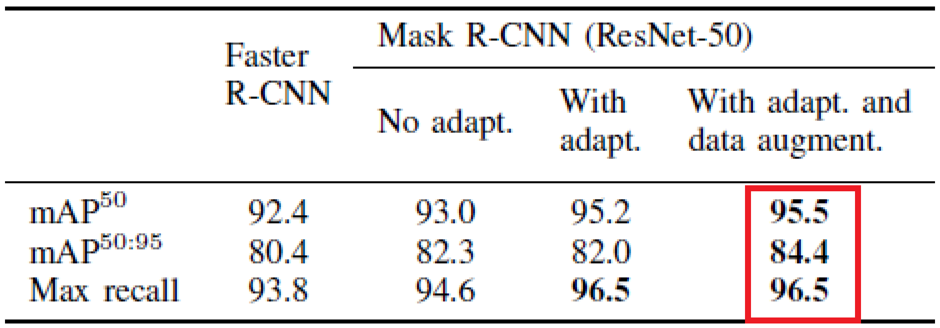
AP (Average precision) [22] is a popular metric in measuring the accuracy of object detectors like Faster R-CNN, SSD, etc. Average precision computes the average precision value for recall value over 0 to 1. They used two variants of the mAP: mAP50, based on the PASCAL visual object challenge [20], and mAP(50:95), based on the COCO challenge [21].

Results

While working on the project they used several databases. In this report they are focused on the DFG Traffic Sign Dataset [14]:

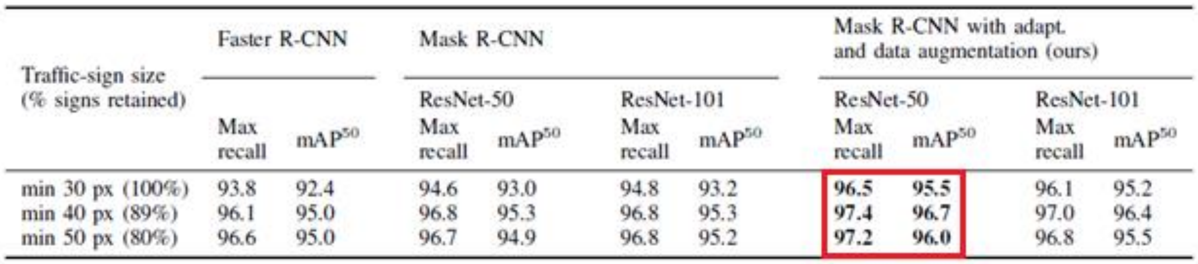
- 200 categories of traffic signs.
- 5254 training and 1703 testing images.
- Using only annotations with at least 30 pixels in size.
- They resized images in all variants of Faster/Mask R-CNN to have image sizes of at least 840 pixels in both width and height.

Results on DFG Traffic Sign Dataset



	Faster R-CNN	Mask R-CNN (ResNet-50)		
		No adapt.	With adapt.	With adapt. and data augment.
mAP^{50}	92.4	93.0	95.2	**95.5**
$mAP^{50:95}$	80.4	82.3	82.0	**84.4**
Max recall	93.8	94.6	**96.5**	**96.5**

Also, they performed evaluation considering different traffic-sign sizes with the results reported in the table:



Traffic-sign size (% signs retained)	Faster R-CNN		Mask R-CNN				Mask R-CNN with adapt. and data augmentation (ours)			
			ResNet-50		ResNet-101		ResNet-50		ResNet-101	
	Max recall	mAP^{50}	Max recall	mAP^{50}	Max recall	mAP^{50}	Max recall	mAP^{50}	Max recall	mAP^{50}
min 30 px (100%)	93.8	92.4	94.6	93.0	94.8	93.2	**96.5**	**95.5**	96.1	95.2
min 40 px (89%)	96.1	95.0	96.8	95.3	96.8	95.3	**97.4**	**96.7**	97.0	96.4
min 50 px (80%)	96.6	95.0	96.7	94.9	96.8	95.2	**97.2**	**96.0**	96.8	95.5

Discussion

Real-time traffic sign detection and recognition application

The proposed solution is applicable to problems requiring the capability of traffic-sign detection such as autonomous driving and advanced driver-assistance systems.

There is a forward-facing camera installed on a vehicle. It captures a video stream. The video stream is a sequence of frames (images). Each frame (or part of them) is processed by the proposed algorithm and the results (selected traffic sign, warnings) are shown on the display.

A possible problem that can occur is performance of Mask R-CNN algorithm. According to the articles that compare this algorithm with YOLO: Real-time object detection system [27], Mask R-CNN is not enough fast:

1) *Poly-YOLO: higher speed, more precise detection and instance segmentation for YOLOv3* [25].

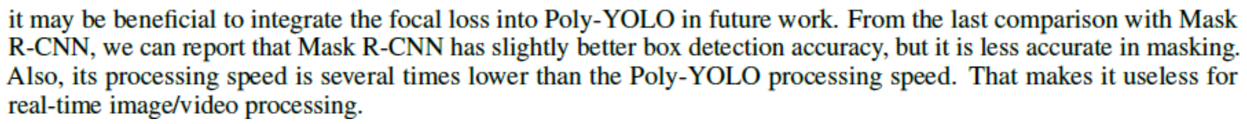
   According to the Table 2 speed of Mask R-CNN processing:

   15.8 FPS for the resolution 448x448 – Simulator dataset

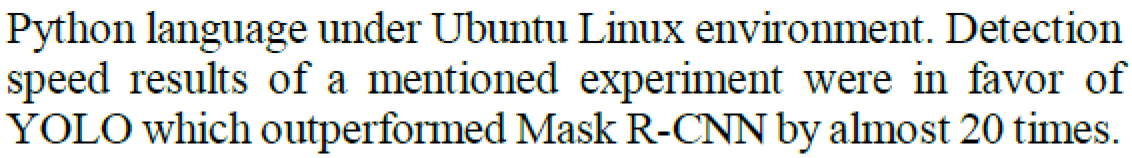
   6.2 FPS for the resolution 1024x1024 – Cityscapes dataset

   7.5 FPS for the resolution 1024x1024 – India driving dataset

   Item *5.3 Results*:


it may be beneficial to integrate the focal loss into Poly-YOLO in future work. From the last comparison with Mask R-CNN, we can report that Mask R-CNN has slightly better box detection accuracy, but it is less accurate in masking. Also, its processing speed is several times lower than the Poly-YOLO processing speed. That makes it useless for real-time image/video processing.

2) *Ball detection using YOLO and Mask R-CNN* [26]. Item: *4 Experiment*:


Python language under Ubuntu Linux environment. Detection speed results of a mentioned experiment were in favor of YOLO which outperformed Mask R-CNN by almost 20 times.

Mask R-CNN speed processing may be not enough high to detect/recognize traffic sign when a vehicle drives with very high speed. The traffic sign will be detected on specific location, but the real sign can be already moved to another location.

Possible solution:

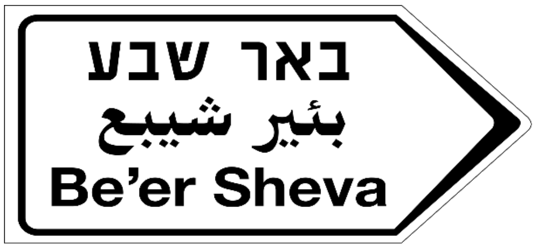
To process 15 frames per second, each second frame (for the stream of 30 fps).

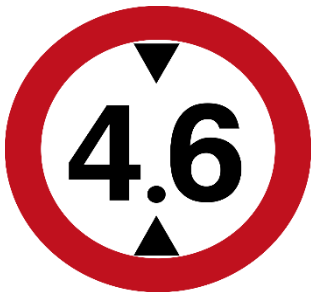
To compress the video frames to relatively small sizes and to send them to the model.

Drawback: small traffic signs may not be detected and recognized.

Traffic signs with text/numbers

Another problem exists: sometimes meaning of the traffic sign depends on the text located on it (the text is not constant):





The architecture described in this article is assigned to detect/recognize non-text traffic signs only (the text can present, but it is constant). If we want to detect/recognize all types of the traffic signs including text we must add extra mechanism to perform it.

Implementation in Israeli context

There is a difference between the traffic signs of Slovenia and the traffic signs of Israel. It must be created a dataset of Israeli traffic signs that will be similar to DFG dataset. The architecture described in the article matches the Israeli non-text traffic signs datasets. But still needs to add a method to detect/recognize a text/numbers on traffic signs.

Traffic Signs Recognition Project

Project: TrafficSignsRecognition

Dataset: DFG Traffic Sign Dataset [14] - traffic signs images in JPEG format.

Programming Language: Python 3

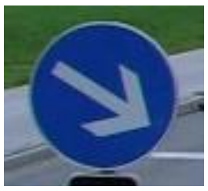
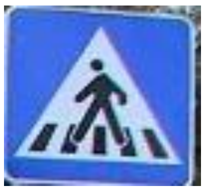
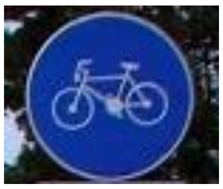
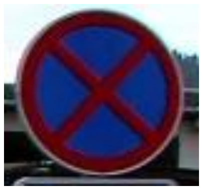
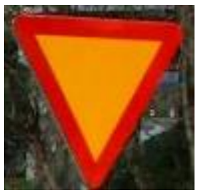
Software: Spyder 3.3.6 / IPython 7.8.0 – development environment.

VGG Image Annotator [23] - an image annotation tool that can be used to define regions in an image and create textual descriptions of those regions.

Technologies/Methods: Mask R-CNN (mrcnn 0.2 [24])

Description: The application is trained to detect/recognize specific traffic signs.

I have selected 5 traffic signs for training/recognition:

I chose 62 images for training, 31 images for validation and 23 images for testing. Each image contains 1-3 traffic signs. Most of signs belong to the selected group (see above). Some images contain traffic signs that are not from the group, for example:

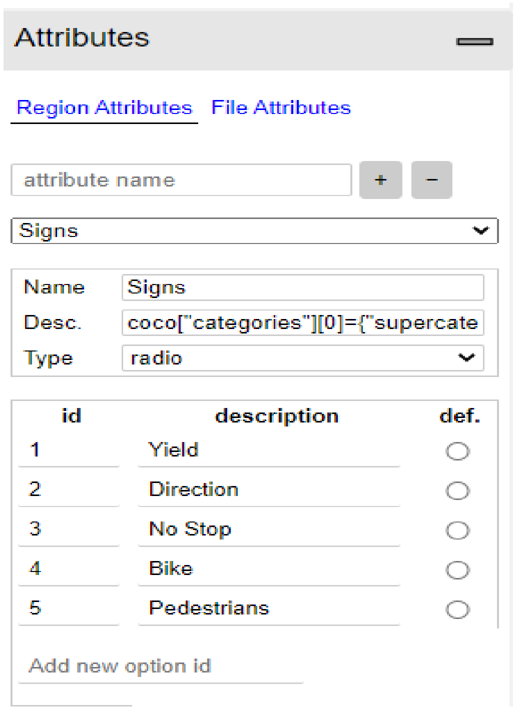
Stages:

1) All the images are 1920X1080. Requirements for image sizes for using the library *mrcnn* (Mask R-CNN): all sizes must be divided into 2 at least 6 times. I chose sizes 768X448. I converted all the images to these sizes, saving height to width ratio (~1.7777777777777). I keep the ratio by adding black pads to height/width:

See the file *resize_images.py* for the code performing resizing.

2) Using *VGG Image Annotator* [23] added annotations to the images in COCO format:

   Create region attributes: 5 classes for 5 traffic signs.



Attributes

Region Attributes File Attributes

attribute name + −

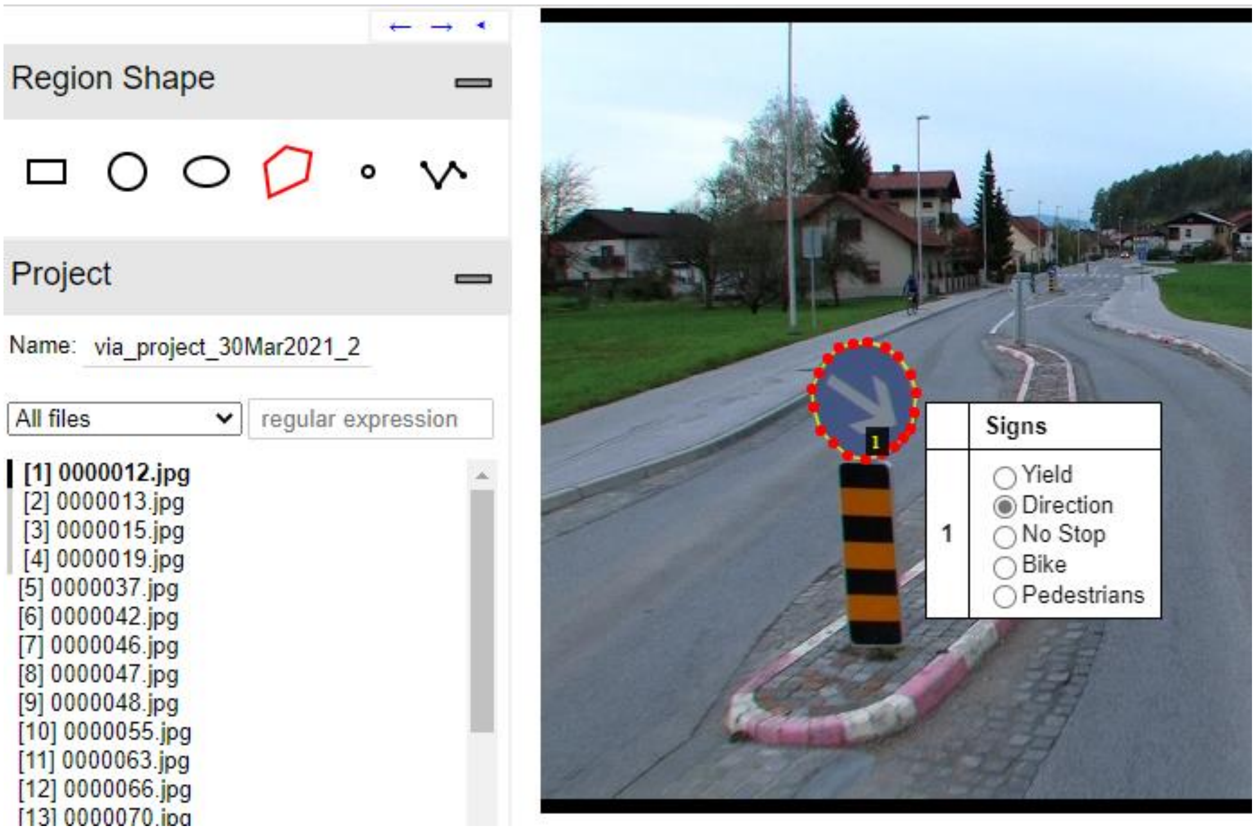
Signs

Name	Signs
Desc.	coco["categories"][0]={"supercate
Type	radio

id	description	def.
1	Yield	◯
2	Direction	◯
3	No Stop	◯
4	Bike	◯
5	Pedestrians	◯

Add new option id

Select region on images:



Region Shape

Project

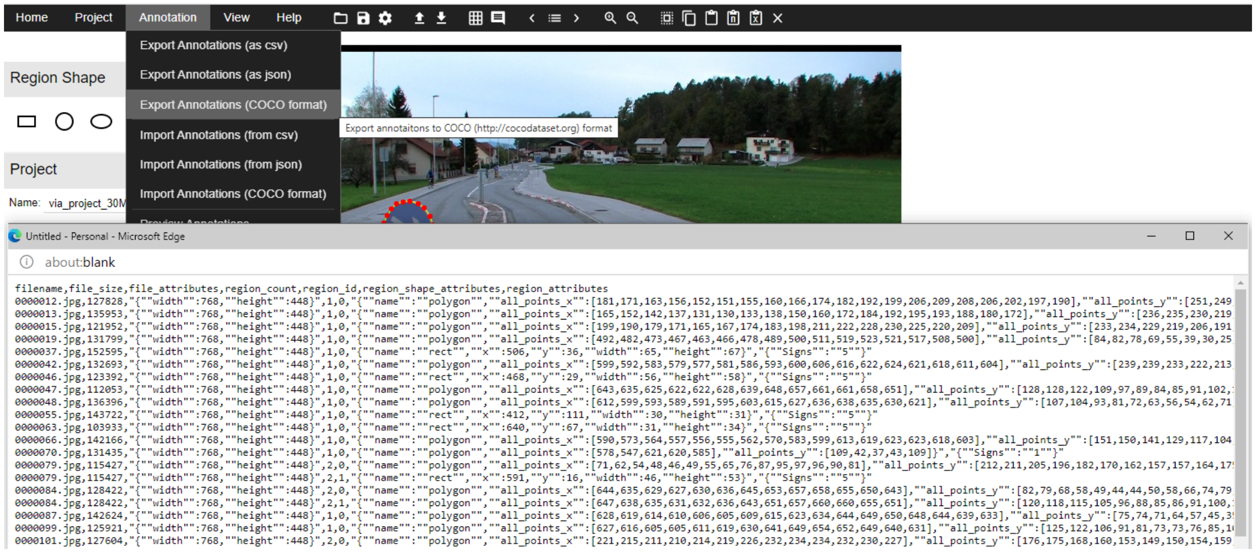
Name: via_project_30Mar2021_2

All files | regular expression

[1] 0000012.jpg
[2] 0000013.jpg
[3] 0000015.jpg
[4] 0000019.jpg
[5] 0000037.jpg
[6] 0000042.jpg
[7] 0000046.jpg
[8] 0000047.jpg
[9] 0000048.jpg
[10] 0000055.jpg
[11] 0000063.jpg
[12] 0000066.jpg
[13] 0000070.jpg

	Signs
1	◯ Yield [✓ Direction] ◯ No Stop ◯ Bike ◯ Pedestrians

Export created annotations in coco annotation json format:


Home
Project
Annotation
View
Help
Export Annotations (as csv)
Export Annotations (as json)
Export Annotations (COCO format)
Export annotaitons to COCO (http://cocodataset.org) format
Import Annotations (from csv)
Import Annotations (from json)
Import Annotations (COCO format)
Region Shape
Project
Name: via_project_30M
Untitled - Personal - Microsoft Edge
about:blank

```
filename,file_size,file_attributes,region_count,region_id,region_shape_attributes,region_attributes
0000012.jpg,127828,"{""width"":768,""height"":448}",1,0,"{""name"":""polygon"",""all_points_x"":[181,171,163,156,152,151,155,160,166,174,182,192,199,206,209,208,206,202,197,190],""all_points_y"":[251,249
0000013.jpg,135953,"{""width"":768,""height"":448}",1,0,"{""name"":""polygon"",""all_points_x"":[165,152,142,137,131,130,133,138,150,160,172,184,192,195,193,188,180,172],""all_points_y"":[236,235,230,219
0000015.jpg,121952,"{""width"":768,""height"":448}",1,0,"{""name"":""polygon"",""all_points_x"":[199,190,179,171,165,167,174,183,198,211,222,228,230,225,220,209],""all_points_y"":[233,234,229,219,206,191
0000019.jpg,131799,"{""width"":768,""height"":448}",1,0,"{""name"":""polygon"",""all_points_x"":[492,482,473,467,463,466,478,489,500,511,519,523,521,517,508,500],""all_points_y"":[84,82,78,69,55,39,30,25
0000037.jpg,152595,"{""width"":768,""height"":448}",1,0,"{""name"":""rect"",""x"":506,""y"":36,""width"":65,""height"":67}","{""Signs"":""5""}"
0000042.jpg,132693,"{""width"":768,""height"":448}",1,0,"{""name"":""polygon"",""all_points_x"":[599,592,583,579,577,581,586,593,600,606,616,622,624,621,618,611,604],""all_points_y"":[239,239,233,222,213
0000046.jpg,123392,"{""width"":768,""height"":448}",1,0,"{""name"":""rect"",""x"":468,""y"":29,""width"":56,""height"":58}","{""Signs"":""5""}"
0000047.jpg,112053,"{""width"":768,""height"":448}",1,0,"{""name"":""polygon"",""all_points_x"":[643,635,625,622,622,628,639,648,657,661,661,658,651],""all_points_y"":[128,128,122,109,97,89,84,85,91,102,
0000048.jpg,136396,"{""width"":768,""height"":448}",1,0,"{""name"":""polygon"",""all_points_x"":[612,599,593,589,591,595,603,615,627,636,638,635,630,621],""all_points_y"":[107,104,93,81,72,63,56,54,62,71
0000055.jpg,143722,"{""width"":768,""height"":448}",1,0,"{""name"":""rect"",""x"":412,""y"":111,""width"":30,""height"":31}","{""Signs"":""5""}"
0000063.jpg,103933,"{""width"":768,""height"":448}",1,0,"{""name"":""rect"",""x"":640,""y"":67,""width"":31,""height"":34}","{""Signs"":""5""}"
0000066.jpg,142166,"{""width"":768,""height"":448}",1,0,"{""name"":""polygon"",""all_points_x"":[590,573,564,557,556,555,562,570,583,599,613,619,623,623,618,603],""all_points_y"":[151,150,141,129,117,104
0000070.jpg,131435,"{""width"":768,""height"":448}",1,0,"{""name"":""polygon"",""all_points_x"":[578,547,621,620,585],""all_points_y"":[109,42,37,43,109]}","{""Signs"":""1""}"
0000079.jpg,115427,"{""width"":768,""height"":448}",2,0,"{""name"":""polygon"",""all_points_x"":[71,62,54,48,46,49,55,65,76,87,95,97,96,90,81],""all_points_y"":[212,211,205,196,182,170,162,157,157,164,17
0000079.jpg,115427,"{""width"":768,""height"":448}",2,1,"{""name"":""rect"",""x"":591,""y"":16,""width"":46,""height"":53}","{""Signs"":""5""}"
0000084.jpg,128422,"{""width"":768,""height"":448}",2,0,"{""name"":""polygon"",""all_points_x"":[644,635,629,627,630,636,645,653,657,658,655,650,643],""all_points_y"":[82,79,68,58,49,44,44,50,58,66,74,79
0000084.jpg,128422,"{""width"":768,""height"":448}",2,1,"{""name"":""polygon"",""all_points_x"":[647,638,635,631,632,636,643,651,657,660,660,655,651],""all_points_y"":[120,118,115,105,96,88,85,86,91,100,
0000087.jpg,142624,"{""width"":768,""height"":448}",1,0,"{""name"":""polygon"",""all_points_x"":[628,619,614,610,606,605,609,615,623,634,644,649,650,648,644,639,633],""all_points_y"":[75,74,71,64,57,45,3
0000099.jpg,125921,"{""width"":768,""height"":448}",1,0,"{""name"":""polygon"",""all_points_x"":[627,616,605,605,611,619,630,641,649,654,652,649,640,631],""all_points_y"":[125,122,106,91,81,73,73,76,85,1
0000101.jpg,127604,"{""width"":768,""height"":448}",2,0,"{""name"":""polygon"",""all_points_x"":[221,215,211,210,214,219,226,232,234,234,232,230,227],""all_points_y"":[176,175,168,160,153,149,150,154,159
```

3) Run the application *MaskRCNN_TrainAndInference.py*.
As a basis model is taken the model *mask_rcnn_coco.h5* - coco trained weights [28]. I performed fine tuning of this model in order to match it to solution of the traffic signs detection/recognition problem.
See *Run Result.html* report for the training/testing results.

<u>Results</u>:

On all the test images all the traffic sign that are defined in the group were recognized correct.

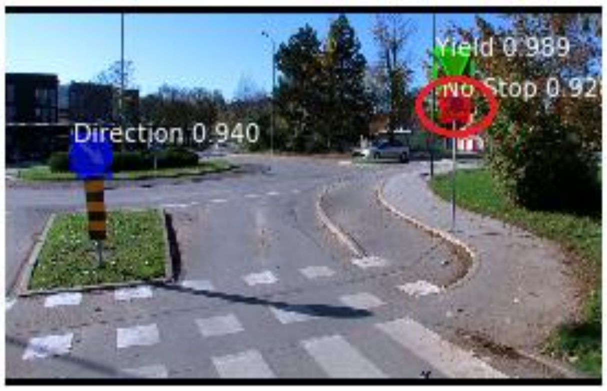
<u>Errors</u>:

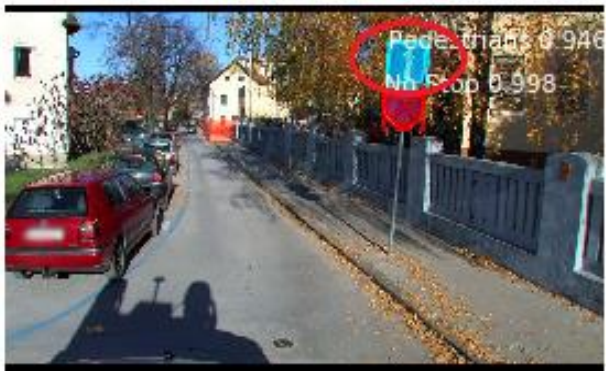
Some traffic sign that are not defined in the group were detected incorrect.

Examples:

Prediction:

Original:



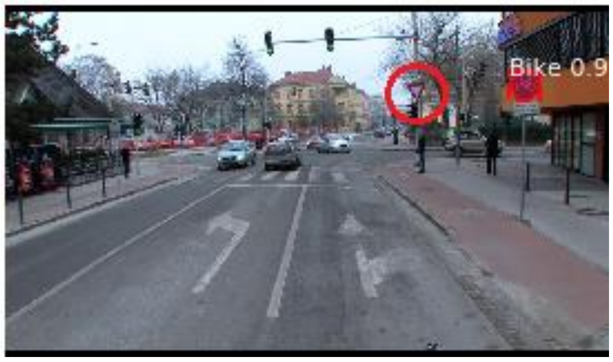


Very small traffic signs are not detected/recognized.

Examples:

Prediction:

Original:



Reason for the errors:

- Only 5 traffic sings were trained to be detected/recognized. So, the application cannot detect/recognize the rest signs.

- Only middle/large signs were trained to be detected/recognized. So, the application cannot detect/recognize the small signs.

References

1. H. Kaiming, G. Gkioxara, P. Dollar, and R. Girshick, “Mask R-CNN,” in International Conference on Computer Vision, 2017, pp. 2961–2969.
2. M. Haloi, “A novel pLSA based Traffic Signs Classification System,” CoRR, vol. abs/1503.0, 2015.
3. Y. Zhu, C. Zhang, D. Zhou, X. Wang, X. Bai, and W. Liu, “Traffic sign detection and recognition using fully convolutional network guided proposals,” Neurocomputing, vol. 214, pp. 758–766, 2016.
4. Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu, “Traffic-Sign Detection and Classification in the Wild,” in CVPR, 2016, pp. 2110–2118.
5. A. Mogelmose, M. M. Trivedi, and T. B. Moeslund, “Vision-Based Traffic Sign Detection and Analysis for Intelligent Driver Assistance Systems: Perspectives and Survey,” Transactions on Intelligent Transportation Systems, vol. 13, no. 4, pp. 1484–1497, 2012.
6. Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)* (Vol. 1, pp. 886-893). Ieee.
7. S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real- Time Object Detection with Region Proposal Networks,” in NIPS, 2015.
8. Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580-587).

9. Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, *104*(2), 154-171.

10. Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 1440-1448).

11. Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*.

12. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*, 1097-1105.

13. Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3431-3440).

14. DFG Traffic Sign Dataset: http://www.vicos.si/Downloads/DFGTSD

15. Detectron for Traffic Signs: https://github.com/skokec/detectron-traffic-signs

16. Detectron framework: https://github.com/facebookresearch/detectron

17. Cafee2: https://github.com/facebookarchive/caffe2

18. K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *in International Conference on Learning Representations*, 2015, pp. 1–14.

19. K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *CVPR*, 2016, pp. 171–180.

20. Everingham, M., Eslami, S. A., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2015). The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, *111*(1), 98-136.

21. Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014, September). Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740-755). Springer, Cham.

22. mAP (mean Average Precision) for Object Detection: https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173

23. VGG Image Annotator: https://www.robots.ox.ac.uk/~vgg/software/via/

24. Mask R-CNN python library (mrcnn 0.2): https://pypi.org/project/mrcnn/

25. Hurtik, P., Molek, V., Hula, J., Vajgl, M., Vlasanek, P., & Nejezchleba, T. (2020). Poly-YOLO: higher speed, more precise detection and instance segmentation for YOLOv3. *arXiv preprint arXiv:2005.13243*.

26. Buric, M., Pobar, M., & Ivasic-Kos, M. (2018, December). Ball detection using YOLO and Mask R-CNN. In *2018 International Conference on Computational Science and Computational Intelligence (CSCI)* (pp. 319-323). IEEE.

27. YOLO: Real Time Object Detection: https://github.com/pjreddie/darknet/wiki/YOLO:-Real-Time-Object-Detection

28. Latest COCO trained weights: https://github.com/matterport/Mask_RCNN/releases/download/v2.0/mask_rcnn_coco.h5