



Evaluation of pre-trained object detection models for the use in the SURE project

Gamze Laitila

BACHELOR'S THESIS
April 2021

ICT Engineering
Software engineering

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in ICT Engineering
Software Engineering

LAITILA, GAMZE:

Evaluation of Pre-trained Object Detection Models for the Use in the SURE Project

Bachelor's thesis 46 pages, appendices 8 pages
April 2021

The main purpose of this study was to investigate as many object detection models as possible in order to evaluate their efficiency and determine a suitable model to be used in the SURE project. This model would have then processed images taken by drones outside, which contain mostly vehicles and crowds of people.

Out of 41 pre-trained object detection models, 17 were selected for evaluation. Over 1000 photographs taken by the drones were received as test data and 240 of them were chosen to be processed. In these photographs, 1754 objects were annotated using an image annotation tool. Detection results and actual annotation results of the images were then compared, and evaluation metrics were calculated using a code written in Python.

As a result of this process, it was found that EfficientDet D7 1536x1536, EfficientDet D6 1280x1280 and EfficientDet D5 1280x1280 were the three top ranking models in terms of accuracy whereas CenterNet Resnet50 V1 FPN Keypoints 512x512, CenterNet Resnet50 V2 Keypoints 512x512 and Faster R-CNN ResNet101 V1 800x1333 were the three top ranking models in terms of image processing speed.

The findings indicate that models that were proven to be the most accurate in international competitions were not necessarily useful for the SURE project. One-stage models could later be investigated, for example YOLO and RetinaNet, but this study suggests the use of the model EfficientDet D6 1280x1280 for efficiency in accuracy and Faster R-CNN Inception ResNet V2 1024x1024 for efficiency in image processing speed.

Key words: object, detection, mAP, evaluation

CONTENTS

1	INTRODUCTION	6
2	ABOUT SURE	7
3	OBJECT DETECTION	8
4	OBJECT DETECTION METHODS	9
4.1	Models under investigation	9
4.2	Two-stage, Neural Network-based Object Detection techniques .	13
4.2.1	CNN	13
4.2.2	R-CNN.....	14
4.2.3	Fast R-CNN	15
4.2.4	Faster R-CNN.....	16
4.2.5	Mask R-CNN.....	18
4.2.6	CenterNet.....	18
4.2.7	EfficientNet.....	20
5	EVALUATION METRICS FOR OBJECT DETECTION.....	21
5.1	Ground truth.....	21
5.2	Intersection over Union	22
5.3	Precision and Recall.....	23
5.4	Mean Average Precision	24
6	ANALYSIS OF THE OBJECT DETECTION MODELS	26
6.1	Work environment	26
6.2	Source of the tested images	27
6.3	Annotation of the images.....	27
6.4	Detection results framed by classes	30
6.5	Evaluation results	32
7	DISCUSSION	35
	REFERENCES	37
	APPENDICES.....	39
	Appendix 1. Complete list of pre-trained models with TensorFlow 2 for COCO 2017 dataset, sorted by mAP results. Marked ones are used in this work.....	39
	Appendix 2. Additional codes made to “code_models” and models\research\object_detection\utils\visualization_utils.py (Tensorflow models repository):.....	40
	Appendix 3. List of object class names in MS COCO dataset.....	42
	Appendix 4. Additional code piece, for renaming ground truth text files for each model.....	43

Appendix 5. Complete chart consisting of image processing times, ranks and mAP scores of each model, sorted by models names in alphabetical order..... 44

ABBREVIATIONS AND TERMS

SURE	Smart urban security and event resilience
TAMK	Tampere University of Applied Sciences
GPU	Graphical Processing Unit
YOLO	You Only Look Once
SSD	Single Shot Detector
R-CNN	Region Based Convolutional Neural Network
Mask R-CNN	Mask Region Based Convolutional Neural Network
Cascade R-CNN	Cascade Region Based Convolutional Neural Network
MS COCO	Microsoft Common Objects in Context
mAP	Mean Average Precision
Faster R-CNN	Faster Region Based Convolutional Neural Network
CNN	Convolutional Neural Network
ReLU	Rectified Linear Unit
RoI	Region of Interest
SVM	Support Vector Machine
IoU	Intersection of Union
TP	True Positive
FP	False Positive
FN	False Negative
AP	Average Precision
code_models	Code that was used for image detection
code_mAP	Code that was used for evaluation of the models

1 INTRODUCTION

With the rapid innovations in image obtaining techniques and digitalized data storage options, people and entities are able to obtain more and more image data. As a result of this, researchers are keen to use these sources for humanity's benefit and look for fields where this number of images can be used. The analysis of the image data obtained, and the extraction of the results have a special importance. With a variety of image processing techniques, results can be drawn from raw image data and used in several applications. For example, daily applications such as face recognition on smartphones or license plate detection of vehicles in traffic are among the most prominent applications in the field of image processing.

In the attempts of developing urban security measures, city of Tampere has initiated a project called SURE (Smart urban security and event resilience), where object detection in images will be used as an observation tool in outdoor events taking place in the city.

Throughout this study, modern object detection methods and models derived from them will be investigated in terms of their accuracy and speed. These models will be evaluated by certain evaluation metrics and the results will be analysed in order to determine a suitable model to be used in the SURE project.

2 ABOUT SURE

Tampere's SURE is an EU-financed project that receives its funds worth of 3.2 million euros from Urban Innovative Actions initiative. Its main purpose is to develop and apply security technologies for cities. Project is to be completed in 3 years and its development started in September 2019.

Partners of the project are Insta Group, Nokia, Securitas, University of Tampere, TAMK (Tampere University of Applied Sciences), Business Tampere and The Baltic Institute of Finland.

One of the project's focus areas is the observation of events in the city where large amounts of people gather. These events and gatherings would be observed with unmanned aerial vehicles, commonly known as drones. The visual data received from these drones will be processed by an object detection algorithm to identify the objects and decipher the happenings throughout the. For example, unexpected movements of the crowd such as pushing, mass movement towards a certain direction or a fast-moving vehicle would be detected by the algorithm and precautions would be taken beforehand in order to prevent security breaches.

3 OBJECT DETECTION

Object detection is a technique used in image processing which lets us detect, identify and track various objects of determined classes (cars, humans, dogs, buildings etc.) in digital images and videos. Object detection is used in many applications simply for image retrieval or in more complex cases like video surveillance.

In the recent years, the amount of unprocessed raw data has increased considerably with the effect of the developments in information storage technologies. This increase led to parallel computing methods and high processing power becoming indispensable for areas such as machine learning and image processing. With all these developments, new libraries that allow parallel calculation on graphic cards have been created. Due to GPUs' (Graphical Processing Unit) high bandwidth, easily programmable registers, and efficiency through thread parallelism, it has become quite easy and convenient to perform arithmetic operations on graphic cards that would normally require a lot of time and high processing power.

As a result, high computation and processing power made it possible to work with multi-layer networks called deep neural networks. Multi-layer neural networks have proven to be the best in the field by getting much better degrees than normal machine learning methods in competitions for classification and detection of objects in images.

4 OBJECT DETECTION METHODS

State-of-the-art object detection methods can be divided into 2 categories: one-stage methods and two stage-methods.

In one-stage methods primary focus is on speed whereas in two-stage methods it is on detection accuracy. Example methods for one-stage methods can be YOLO (You Only Look Once), SSD (Single Shot Detector) and RetinaNet and for two-stage they can be Faster R-CNN (Region Based Convolutional Neural Network), Mask R-CNN and Cascade R-CNN. The most popular benchmark is the MS COCO (Microsoft Common Objects in Context) dataset and the main evaluation metric used is Mean Average Precision (mAP) metric. (Papers With Code n. d.)

During the time of this study, Google and TensorFlow are the leading publishers of object detection models. Every passing year new and more efficient methods and models are developed, and these new models prove themselves in multiple challenges after they have been trained on certain datasets. Dataset in general is a collection of data. This data could be, for example, a combination of several tables which include attributes and values of collected data or like in this case, a database consisting of thousands or millions of images.

Most of above-mentioned challenges are based on MS COCO dataset where there are 80 classes and 328 thousand images or ImageNet dataset where there are 1000 classes and over 14 million images to be detected by the models. Models used in this study are trained on the COCO 2017 dataset with TensorFlow 2.

4.1 Models under investigation

Models that are tested in this study with their full names, including versions and image resolutions are the following:

- CenterNet HourGlass104 Keypoints 512x512
- CenterNet HourGlass104 Keypoints 1024x1024
- CenterNet HourGlass104 512x512
- CenterNet HourGlass104 1024x1024
- CenterNet Resnet50 V1 FPN Keypoints 512x512
- CenterNet Resnet50 V2 Keypoints 512x512
- EfficientDet D2 768x768
- EfficientDet D3 896x896
- EfficientDet D4 1024x1024
- EfficientDet D5 1280x1280
- EfficientDet D6 1280x1280
- EfficientDet D7 1536x1536
- Faster R-CNN Inception ResNet V2 640x640
- Faster R-CNN Inception ResNet V2 1024x1024
- Faster R-CNN ResNet101 V1 800x1333
- Faster R-CNN ResNet101 V1 1024x1024
- Mask R-CNN Inception ResNet V2 1024x1024

At the time of this study, there were 41 pre-trained object detection models available (appendix 1) which were developed using TensorFlow 2 for the COCO 2017-dataset. 17 of these 41 models were chosen for investigation: various versions of Faster R-CNN, EfficientNet and CenterNet and one Mask R-CNN model. These are all two-stage methods. One-stage methods such as YOLO, SSD or RetinaNet could be compared in a later study.

Models seen in appendix 1 were tested and competed at least in the MS COCO 2017 Challenge and got a spot in the comparison charts. Easiest way would have been to choose the one with the best rankings and claim that it can be used in the SURE project. But there are many factors that prevent this, like reverse ratio between accuracy and speed or unexpected results caused by image resolutions.

For example, it can be seen from the list that 'CenterNet Hourglass104 Keypoints 1024x1024' seems to be the most accurate model to determine the

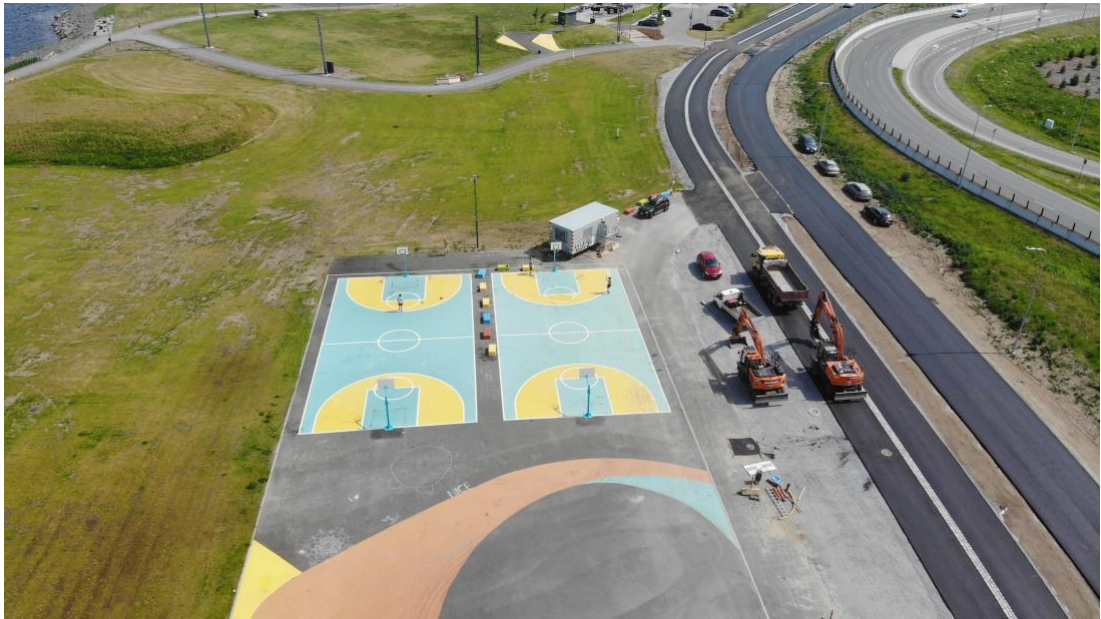
keypoints of the objects although is not as good at boxing them. However, there are way faster models that are not as accurate at keypoints but relatively good at boxing.

Another reason for testing so many models is the difference in image resolutions which can be seen at the end of the model names. A model would transform the resolution of the input image into its own, 640x640, 800x1333, 1024x1024 etc. So, detection with higher resolution would last longer yet results would be more accurate.

Last and most important reason for testing these models is the dissimilarity between the images taken by the SURE drone (picture 1; picture 2) and the images in COCO dataset (picture 3) which were used to test the models in getting their rankings.



PICTURE 1. Sample image taken by the SURE drone



PICTURE 2. Sample image taken by the SURE drone



PICTURE 3. Sample images from COCO dataset

As clearly seen in the pictures above, images taken by the drone and images used in testing have totally different concepts and scalings. The SURE drone will be taking the photographs from over tens of meters away in order to capture crowds and places in wider views. While a person's silhouette can easily cover half of a test image, drone image may consist tens or hundreds of persons with way smaller coverage. As a result, a model that has significantly good numbers in the rankings might not perform as well with the SURE drone images.

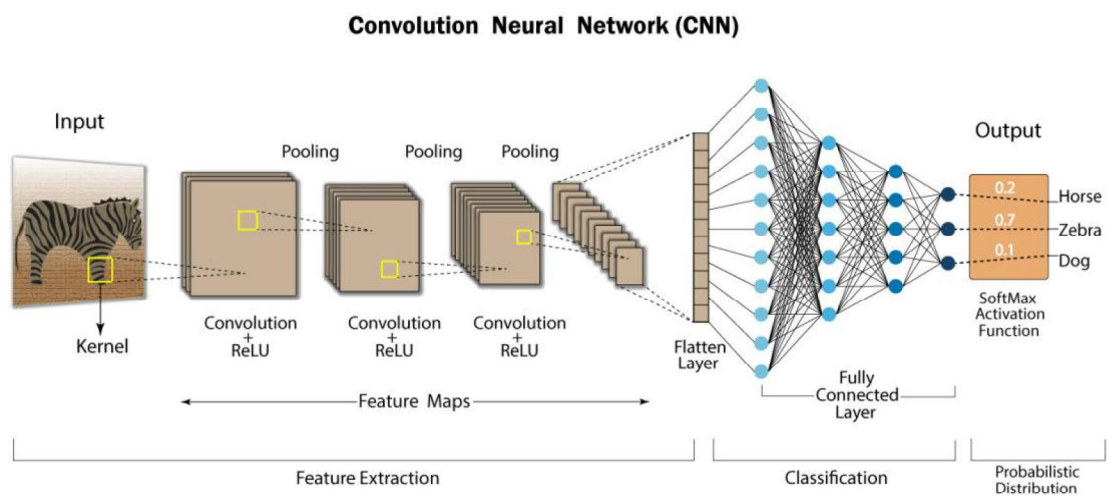
Because of the reasons mentioned above, selection of the pre-trained models started from the top rankings in terms of accuracy. As the list went down, models with the most variety in methods, resolution and speed were additionally selected.

4.2 Two-stage, Neural Network-based Object Detection techniques

Now that it is established which models are under investigation in this study and why, following sections will present relatively brief information about them. In order to set the basics of Faster R-CNN and Mask R-CNN, earlier methods of the R-CNN family (CNN (Convolutional Neural Network), R-CNN and Fast R-CNN) are also explained in the following sections.

4.2.1 CNN

CNN is a deep learning network developed for image and video processing. CNN consists of 4 layers. These are Convolution layer, ReLU (Rectified Linear Unit) layer, Pooling layer and Fully connected layer (picture 4).



PICTURE 4. Architecture of CNN (Swapna 2020)

Convolution layer is the first layer that handles the image in CNN algorithms. Technically images are matrices consisting of pixels with certain values in them. In the convolution layer, a filter smaller than the original image size hovers over the image and tries to capture certain features from these images. Parameters learned in CNN algorithms are the values in these filters. The model constantly updates these values and begins to detect features even better.

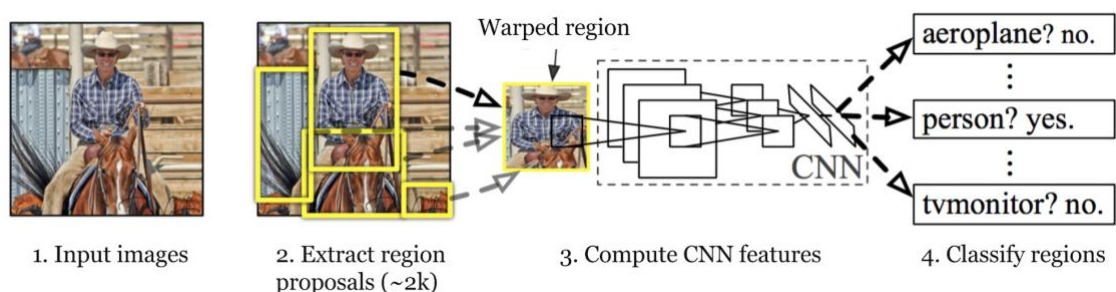
ReLU (Rectified Linear Unit) is a nonlinear function that works as in $f(x) = \max(0, x)$. For example, a result that passes from the convolutional layer might consist of negative numbers caused by a matrix filter. ReLU function takes the negative value, let's say -25, and gives an output of 0. Another ReLU function that takes the value 25, gives 25 as output. ReLU's main purpose is to get rid of negative values, and it has a very important role in CNN.

Like the convolution layer, the pooling layer also aims to reduce the dimensionality. This way the required processing power is reduced, the captured unnecessary features are ignored, and more important features can be focused on.

In the Fully Connected layer the image passes through the convolution layer and the pooling layer several times and is transformed into a flat vector.

4.2.2 R-CNN

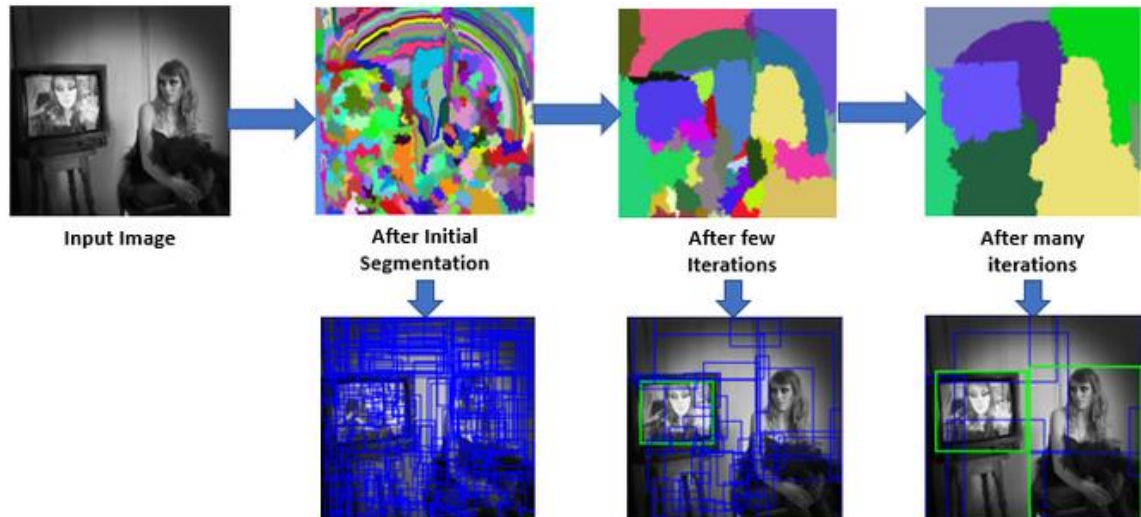
R-CNN (Region-based Convolutional Neural Networks) works in two main steps. As shown in picture 5; first with selective search, it splits the image area into about 2000 "candidate" pieces which are called Region of Interest (RoI). In the second step, system then computes previously mentioned CNN features for each region in order to produce classifications. (Weng 2017.)



PICTURE 5. The architecture of R-CNN (Girshick, Donahue, Darrell & Malik 2014)

As pawangfg (2020) explains, selective search is a method used to determine the regions that need to be captured in images. Small regions are determined

first, which works with the logic of hierarchy from small to large. Then, two similar regions are combined, creating a larger region. This process (picture 6) continues repeatedly and at the end larger regions appear in each iteration.



PICTURE 6. Selective search (pawangfg 2020)

According to Gandhi (2018), there were still problems with R-CNN. First of all, classification of nearly 2000 region proposal was time consuming and real-time implementation was unlikely due to 47 seconds of process cycle per image. Complex multi-stage training pipeline and fixed selective search algorithm prevented the model from learning during the process.

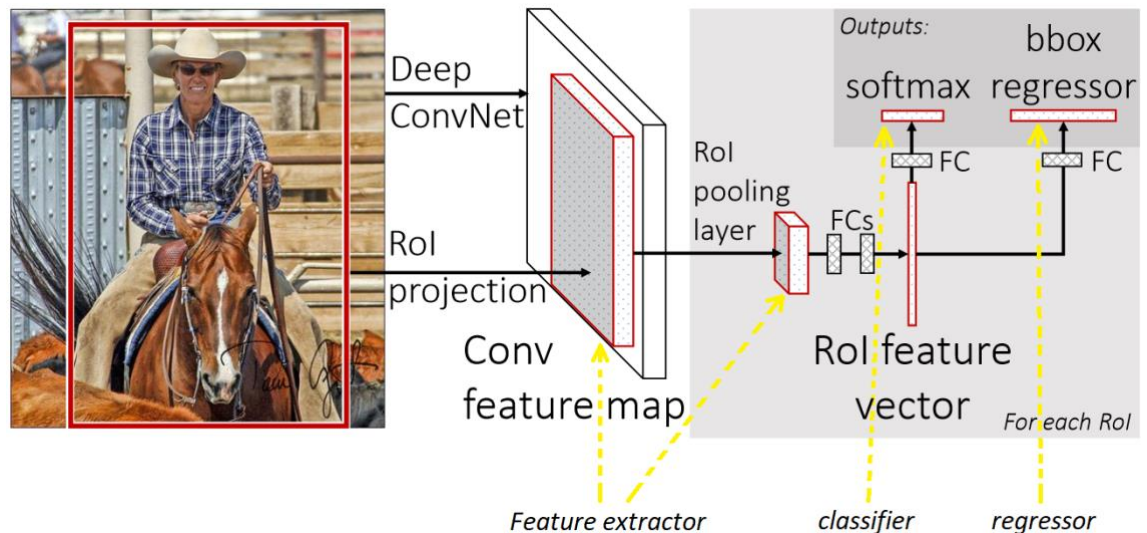
One year after R_CNN, computer scientists developed Fast R-CNN which is about 146 times faster than the R-CNN during the test time. It solves above-mentioned issues efficiently.

4.2.3 Fast R-CNN

In R-CNN, creation of 2000 different candidate regions and the use of 2000 different CNN networks for these regions costs hugely in terms of the training process. To solve this, Fast R-CNN gets rid of the 2000 CNN models and uses only one model.

The biggest improvement in Fast R-CNN is that it combines CNN, SVM (Support Vector Machine) and Regressor phases that were used in R-CNN. SVM,

“classifier” in picture 7, is an algorithm that finds significant distances between data points in an N-dimensional space where N is the number of features of mentioned data points. This way different types of features are “classified”. Regressor is used to determine more precise coordinates for the bounding box. With this combination, it achieves a tremendous advantage in performance. These phases are summarized and shown in picture 7.



PICTURE 7. Fast R-CNN combined the CNN, classifier, and bounding box regressor into one, single network. (Girshick 2015, edited)

4.2.4 Faster R-CNN

After seeing the imperfections of Fast R-CNN, Ren et al. (2015) developed a better and “faster” version of Fast R-CNN called Faster R-CNN which would lower the number of region proposals, is faster and more accurate than selective search and is able to offer better region selections for overlapping objects. (Weng 2017.)

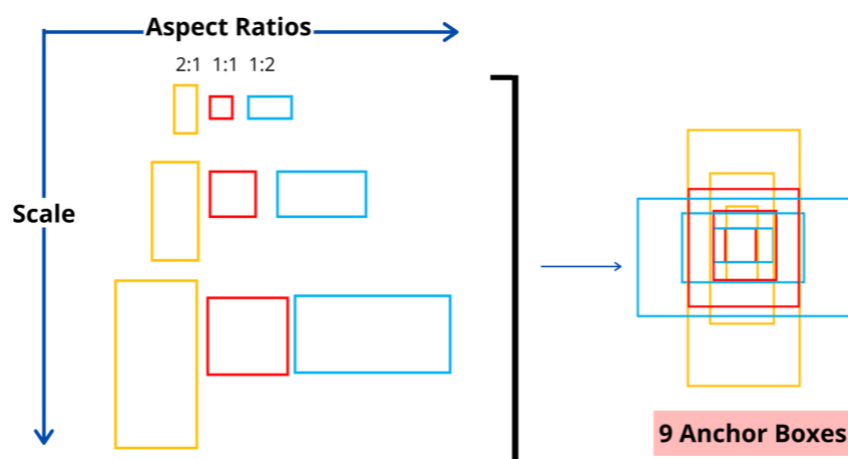
Considering picture 8, on the image on the left, there are a lot of objects overlapping each other. On the right side of the picture, the bounding boxes are drawn for each object. Selective search could be applied here but as a result there would be too many Rols to handle. In this case, Faster R-CNN method offers the use of Anchor Boxes.



PICTURE 8. An image consisting overlapping objects and their bounding boxes (Yelisetty 2020)

Anchor Boxes

There are generally 3 types of boxes where an image could fit. These boxes could be squared, rectangular and wide or rectangular and tall. In addition, these 3 types of boxes could have 3 different sizes: big, small or medium (picture 9). Experimentally, it was found that any object in an image could be detected using one of these 9 boxes. If the overlapping image shown above in picture 8 to be considered, hovering these 9 types of boxes over the image would result in determining the majority of the overlapping objects although not very accurately since the size of the boxes would be fixed. (Yelisetty 2020.)



PICTURE 9. Anchor boxes shown with their aspect ratios and scales (Yelisetty 2020)

4.2.5 Mask R-CNN

Mask R-CNN extends Faster R-CNN to pixel-level image segmentation (Weng 2017). The most significant improvement in Mask R-CNN is that it offers instance segmentation for each detected object. The method is used in applications where there is the need to know where each and every detected object is located. For example, a self-driving car would need to see each car around it in order to make necessary calculations. (ArcGIS Developers n. d.)

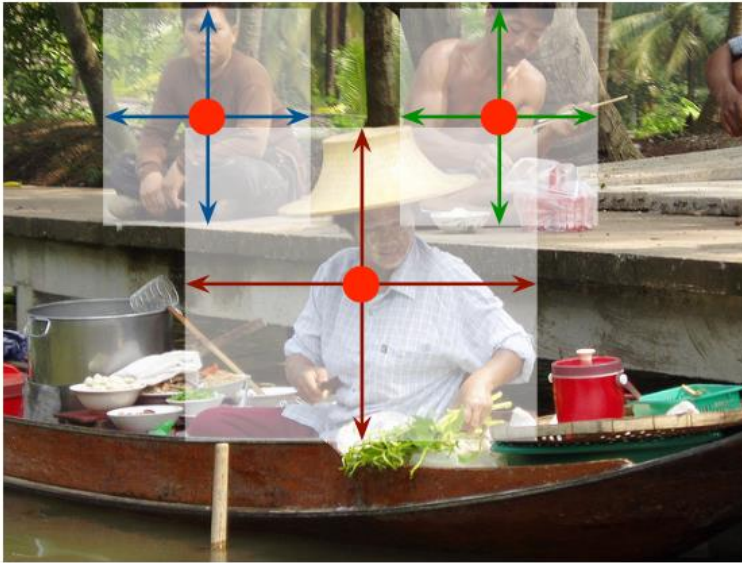
Examples of Mask R-CNN results on the COCO test set are shown in picture 10.



PICTURE 10. Mask R-CNN results on the COCO test set. (He et al. 2017)

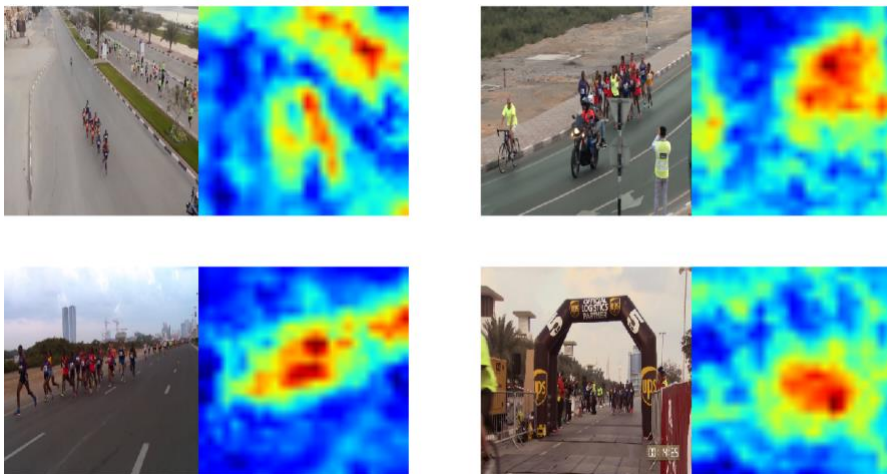
4.2.6 CenterNet

According to Zhou, Wang and Krähenbühl (2019), object detection algorithms where the method is based on window-sliding are wasteful since a large number of locations and dimensions need to be counted and processed. They found a new and more efficient method called CenterNet where objects are represented by a single point at the center of their bounding box as shown in picture 11.



PICTURE 11. Example of modelled objects with the center of their bounding boxes (Zhou et al. 2019)

The main idea is to feed the image to a fully convolutional network which generates a heatmap (picture 12). Peaks of the heatmap address the centerpoints of the bounding boxes, hence the objects. Result image also contains the height and weight of the bounding boxes.



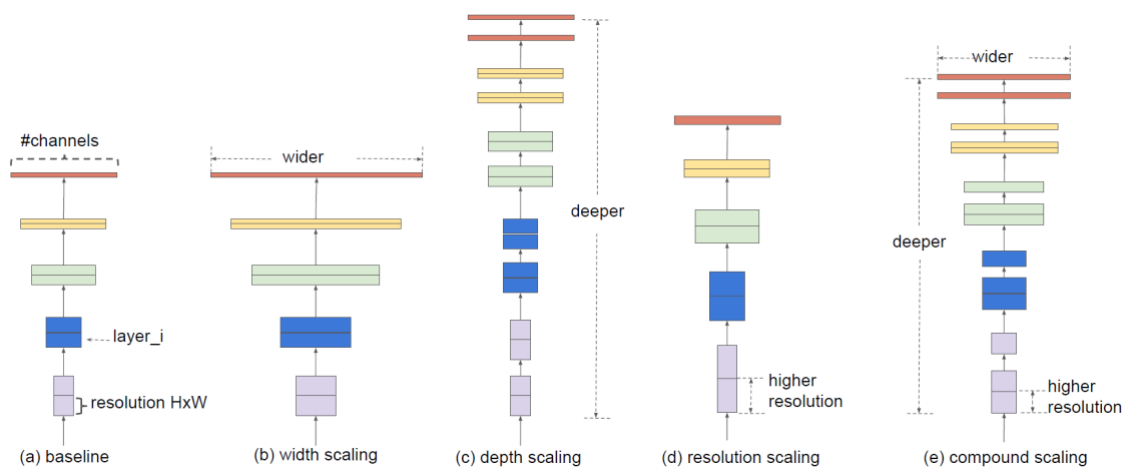
PICTURE 12. Examples of heatmaps: Left parts show the original image and right parts heatmaps (Tzelepi & Tefas 2017)

Zhou (2019) also claims that CenterNet runs at a very high speed due to its simplicity and can easily be adjusted to serve other tasks such as 3D object detection and multi-person human pose estimation.

4.2.7 EfficientNet

In a traditional CNN, factors like depth, width and scaling of the resolution are arbitrary. In order to increase the power of a CNN, more convolution layers are often added to the network e.g. ResNet18, ResNet50, ResNet1202. But the amount of the layers and the performance of the system do not increase at the same rate so after a while this process becomes impractical.

Unlike the conventional practice, EfficientNet uses a compound coefficient to uniformly scale width, depth, and resolution in a principled way (picture 13). The main reason to come up with this compound coefficient, was to prevent the inefficiency caused by the increase in layers and channels, as the size of the images got bigger. (Papers With Code n. d.)



PICTURE 13. Model scaling (Tan & Lee 2019)

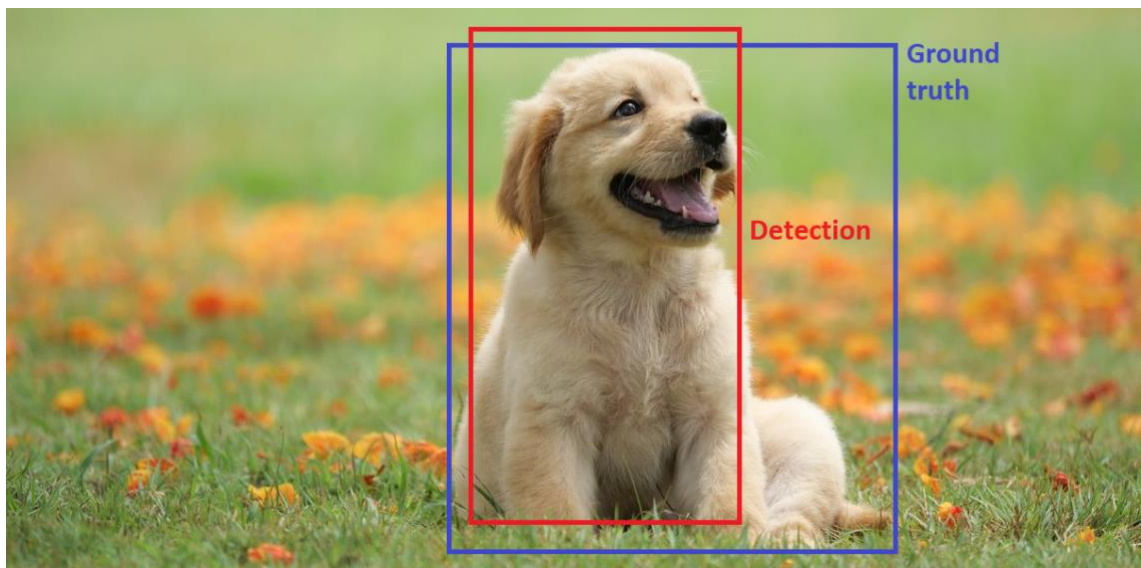
In the above picture;

- (a) is a baseline network example,
- (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution and
- (e) is the proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio. (Tan & Lee 2019.)

5 EVALUATION METRICS FOR OBJECT DETECTION

5.1 Ground truth

Ground truth, in general, means a defined standard by which an algorithm's successful result is evaluated. In object detection, ground truth represents the true state of an object located in an image (picture 14). This state would include the visual bounding box around the object, predicted along with its width, height or centerpoint and an expression to show what class it belongs to. Class here could be a specific object like siberian husky, german shephard, hognose snake, sand viper etc. (classes from ImageNet dataset) or a more generalized one like dog, snake etc. (classes from MS COCO dataset).

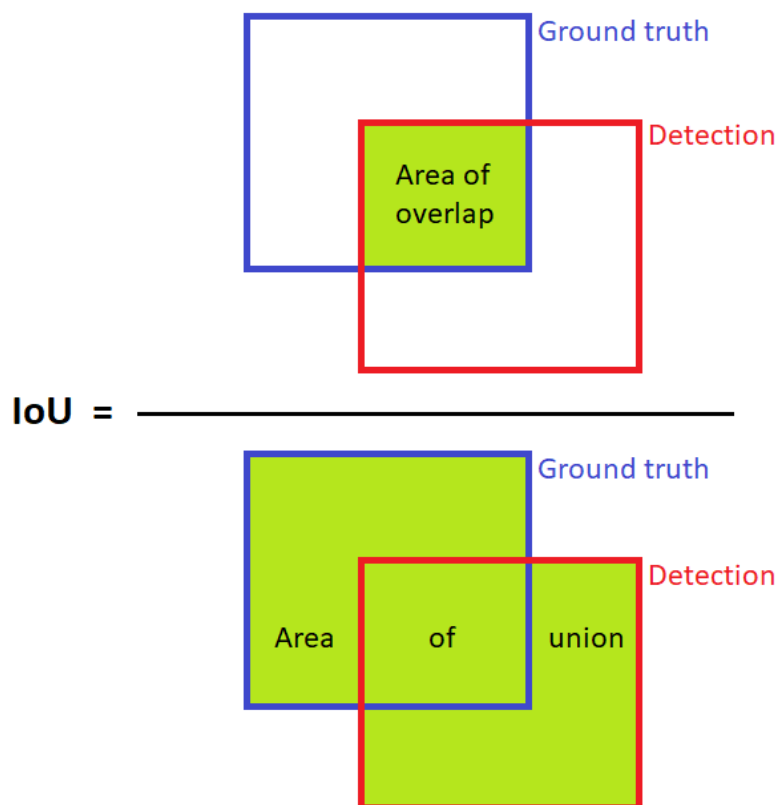


PICTURE 14. Example of ground truth and detection bounding boxes of an object

In this work, the evaluation process required that the ground truth representation would be in text format where one row of the file would consist the information that belongs to one detected object. Detailed explanation of this process can be found in section 6.3.

5.2 Intersection over Union

Intersection over Union (IoU) is the name of the calculation which gives “the overlap divided by the union” of 2 bounding boxes: ground truth bounding box and detection (prediction) bounding box. A simple visual example of IoU is shown in picture 15.

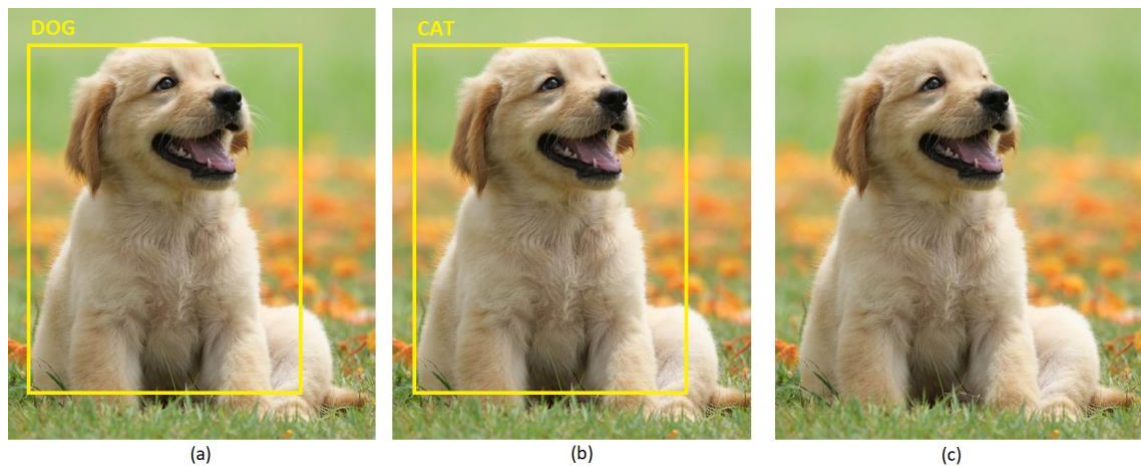


PICTURE 15. Intersection over Union

For most evaluation cases like competitions, an IoU threshold of 0.5 is sufficient. This number means that there is most likely an object inside the ground truth box. IoU is used to determine whether a prediction is positive or negative. For example, if mAP is being calculated for IoU value of 0.5 (mAP@0.5)

- IoU ≥ 0.5 , then **true positive (TP)**: ground truth object is detected with the correct class.
- IoU < 0.5 , then **false positive (FP)**: ground truth object is detected with a wrong class.
- **False negative (FN)**: ground truth object is not detected.

Example cases are shown in picture 16.



PICTURE 16. Example cases of IoU results where (a) is a true positive, (b) a false positive and (c) a false negative.

True positive (TP), False positive (FP) and False negative (FN) concepts are used to calculate precision and recall, which are explained in the next section.

5.3 Precision and Recall

Precision tells in what ratio the object detection model found the correct objects in the image. Or in other words, how many of the positive results are actually positive.

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{\text{total positive results found by the model}}$$

Recall tells in what ratio the model managed to identify those cases that are positive.

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{\text{total number of dog images}}$$

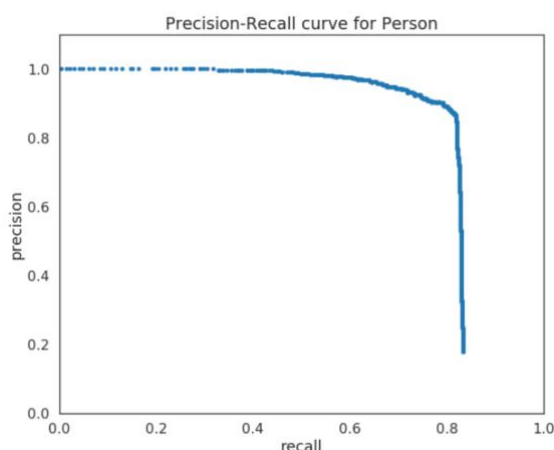
5.4 Mean Average Precision

Mean average precision (mAP) is the most used object detection model evaluation metric which measures the model's ability to correctly determine the objects' bounding boxes for some confidence value. (Nelson 2020.)

Basically, mAP is the average of APs (Average Precision) where AP is found by calculating the area under the precision-recall curve, for specific values of IoUs.

To calculate the AP, for a specific class (say a "person") the precision-recall curve is computed from the model's detection output, by varying the model score threshold that determines what is counted as a model-predicted positive detection of the class. (Arlen 2018.)

An example precision-recall curve may look like in picture 17.



PICTURE 17. Precision-Recall curve for an example classifier. A point on the precision-recall curve is determined by considering all objects above a given model score threshold as a positive prediction, then calculating the resulting precision and recall for that threshold (Arlen 2018)

The final step to calculating the AP score is to take the average value of the precision across all recall values. This becomes the single value summarizing the shape of the precision-recall curve. To do this unambiguously, the AP score is defined as the mean precision at the set of 11 equally spaced recall values, $Recall_i = [0, 0.1, 0.2, \dots, 1.0]$. Thus,

$$AP = \frac{1}{11} \sum_{Recall_i} Precision(Recall_i)$$

The precision at recall i is taken to be the maximum precision measured at a recall exceeding $Recall_i$. (Arlen 2018.)

6 ANALYSIS OF THE OBJECT DETECTION MODELS

6.1 Work environment

The computer used for this work had the following system specifications:

- Processor: Intel® i5-4460
- RAM: 16.0 GB DDR3 @ 1600 MHz.
- 250 GB Crucial MX500 SATA SSD

At first, object detection process was tested by using GPU but the existing GPU (GTX 1060) had a RAM size of 3 GB which proved to be insufficient for the process. During the time of this study, there was a worldwide shortage in GPU production and there was no other way to access another PC having the necessary GPU features. So, the work had to be performed using the existing CPU and RAM combination. This affected the image process time to be a couple of minutes per image whereas this number is tens or a couple of hundreds of milliseconds with GPU process.

Models were run on Python (3.7.9) in Anaconda using many necessary packages one of which was TensorFlow 2.1.0.

The first code, later referred to as “code_models”, that processes the images through the models was obtained from TensorFlow Hub Object Detection Colab (The TensorFlow Hub Authors 2020.). In order to speed up the process, code was mildly edited e.g loops were added to run through image folders so that the model can process one image after another without break. Another edition (appendix 2) was made to extract detection results (class name, confidence percentage, coordinates of the bounding box) and write them in a text file to be used later.

The second source code, later it will be referred to as “code_mAP”, which calculates the mAPs was developed by Cartucho and his colleagues for their paper (Cartucho et al. 2018.). It required that both ground truth (more in section

6.3) and object detection files (more in section 6.4) were in text file format, having the same file names with the following syntaxes:

Ground truth:

```
<class_name> <left> <top> <right> <bottom> [<difficult>]
e.g
tvmonitor 2 10 173 238
book 439 157 556 241
book 437 246 518 351 difficult
pottedplant 272 190 316 259
```

Detection results:

```
<class_name> <confidence> <left> <top> <right> <bottom>
e.g
tvmonitor 0.471781 0 13 174 244
cup 0.414941 274 226 301 265
book 0.460851 429 219 528 247
chair 0.292345 0 199 88 436
book 0.269833 433 260 506 336
```

Ground truth files were created using a graphical image annotation tool named labellmg (Tzutalin 2015, Copyright (c)).

6.2 Source of the tested images

Over 1000 photographs taken by the SURE drones were received as test data. Since the focus classes of object detection were persons and vehicles, out of over 1000 images, 240 of them were chosen to be tested through these models. Having a random vehicle in these images was inevitable so chosen images were those where there were as many people at the same time with vehicles, in many angles as possible. 13 of these images had a pixel resolution of 4056x2280 and 227 had 1920x1080.

6.3 Annotation of the images

Ground truth bounding boxes can be drawn by a graphical image annotation tool. Several open-source tools are available on the internet, and in this work Tzutalin's "labellmg" (2015) was used. It was written in Python and uses Qt as graphical

interface. Tool is provided with the original images and a text file that consists a list of the object classes' names (appendix 3). After drawing bounding boxes on an image and saving it, the tool produces an XML file (used for ImageNet dataset) that holds the annotation data such as the annotated object's class and coordinates of the ground truth bounding box.

However, "code_mAP" required a text file with each row only consisting of the detected object's name and coordinates of the bounding box in the order of

- left (starting pixel on x-axis)
- top (starting pixel on y-axis)
- right (ending pixel on x-axis) and
- bottom (max pixel value on y-axis).

So, after the annotation was done, XML files (picture 18) needed to be modified where unneeded columns were removed, and the rest was saved as (for the lack of an option with "space" as a delimiter) a "tab delimited" text file (picture 19).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	folder	filename	path	database	width	height	depth	segmented	name	pose	truncated	difficult	xmin	ymin	xmax	ymax
2	DJI_0142-DJI_0847_57	DJI_0824.jpg	C:\Users\path\DJI_0824.jpg	Unknown	1920	1080	3	0	person	Unspecified	0	0	1288	457	1315	524
3	DJI_0142-DJI_0847_57	DJI_0824.jpg	C:\Users\path\DJI_0824.jpg	Unknown	1920	1080	3	0	person	Unspecified	0	0	949	279	966	328
4	DJI_0142-DJI_0847_57	DJI_0824.jpg	C:\Users\path\DJI_0824.jpg	Unknown	1920	1080	3	0	person	Unspecified	0	0	953	285	973	332
5	DJI_0142-DJI_0847_57	DJI_0824.jpg	C:\Users\path\DJI_0824.jpg	Unknown	1920	1080	3	0	person	Unspecified	0	0	130	185	146	220
6	DJI_0142-DJI_0847_57	DJI_0824.jpg	C:\Users\path\DJI_0824.jpg	Unknown	1920	1080	3	0	person	Unspecified	0	0	286	230	302	269
7	DJI_0142-DJI_0847_57	DJI_0824.jpg	C:\Users\path\DJI_0824.jpg	Unknown	1920	1080	3	0	person	Unspecified	0	0	301	232	318	273
8	DJI_0142-DJI_0847_57	DJI_0824.jpg	C:\Users\path\DJI_0824.jpg	Unknown	1920	1080	3	0	car	Unspecified	0	0	680	333	813	383
9	DJI_0142-DJI_0847_57	DJI_0824.jpg	C:\Users\path\DJI_0824.jpg	Unknown	1920	1080	3	0	car	Unspecified	0	0	425	588	582	679
10	DJI_0142-DJI_0847_57	DJI_0824.jpg	C:\Users\path\DJI_0824.jpg	Unknown	1920	1080	3	0	truck	Unspecified	0	0	907	680	1343	923

PICTURE 18. Example XML file produced by labellingm opened in Excel; needed data were in columns I, M, N, O and P

```

Tiedosto Muokkaa Muotoile Näytä Ohje
person 1288 457 1315 524
person 949 279 966 328
person 953 285 973 332
person 130 185 146 220
person 286 230 302 269
person 301 232 318 273
car 680 333 813 383
car 425 588 582 679
truck 907 680 1343 923
Rivi 9, Sarake 23 100% Windows (CRLF) UTF-8

```

PICTURE 19. Example text file consisting of the necessary data for "code_mAP"

A piece of code was written for this problem which deleted the first row (titles) and changed each tabulator with a space character and was run through all 240 ground truth files.

A total of 1754 objects of 9 classes (car, person, truck, bench, boat, bicycle, bus, dog, motorcycle) were annotated in 240 images with .jpg format as received from the SURE drone. Figure 1 shows totals of the annotations made in these images, sorted by class names.

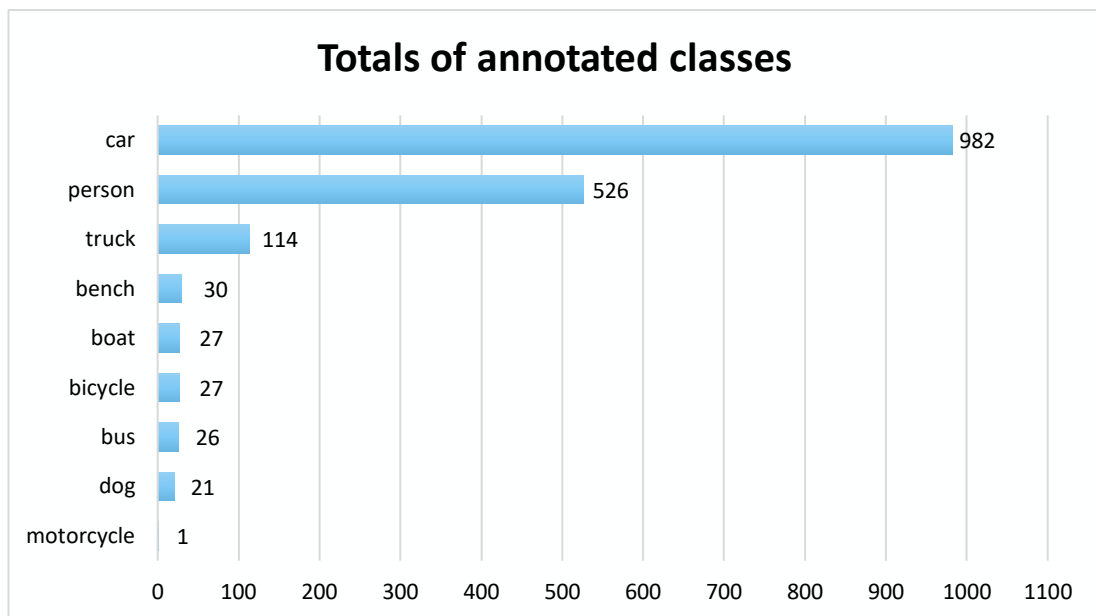
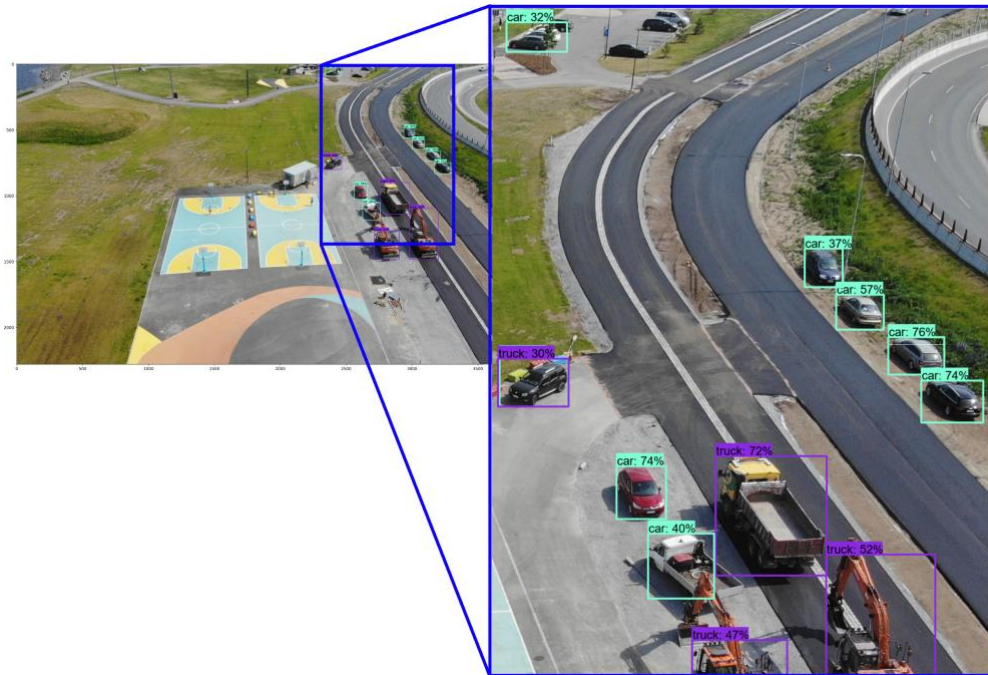


FIGURE 1. Total numbers of annotated objects, sorted by class name

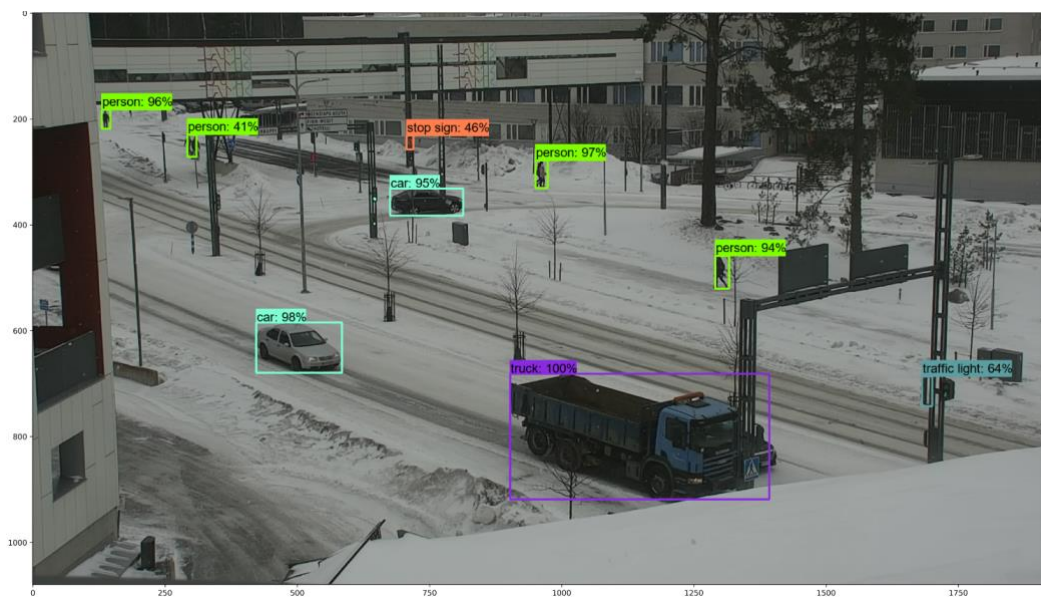
“code_mAP” required that the detection results text files and the ground truth text files to have the same names. Because of this, each model had its own ground truth files created for the same 240 images (copied from original ground truth files and given the model’s name and the name of the image in use), with names coinciding with the corresponding detection result files. This meant there would be 4560 ground truth text files to be handled. Another code piece (appendix 4) was written to speed up this process and keep the filenames organized.

6.4 Detection results framed by classes

At the end of its process, “code_models” would produce a copy of the input image (pictures 20; picture 21) including bounding boxes and confidence values of the objects it has detected in the image.

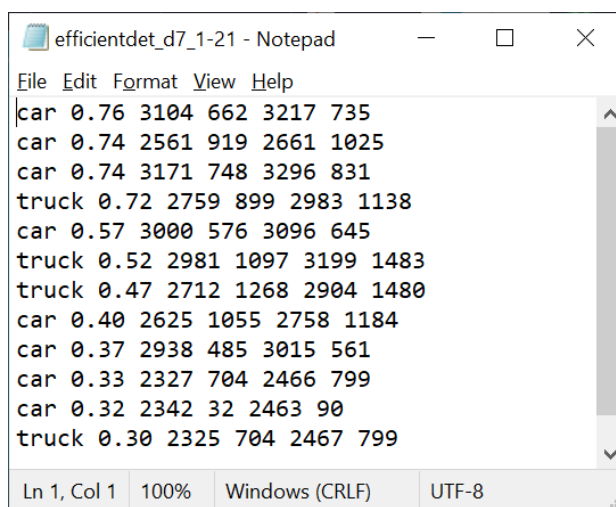


PICTURE 20. Example of a framed image (objects detected by model EfficientDet D7 1536x1536)



PICTURE 21. Example of a framed image (objects detected by model Faster R-CNN Inception ResNet V2 1024x1024)

Detection result text files would also be created at the end of the process (pictures 22; picture 23) using the additional code (appendix 3). As seen in picture 23, sometimes detected classes' names consisted of more than 1 word. In any case, it was enough that class names would match the ones that are in the ground truth files but since the annotations were only for the classes car, person, truck, bench, boat, bicycle, bus, dog and motorcycle which consisted of only 1 word. So there was need to modify only those detection text files that had the 2-word class names; some of them manually and some of them using rows 1211–1229 in the code seen in appendix 2, 2 (2).

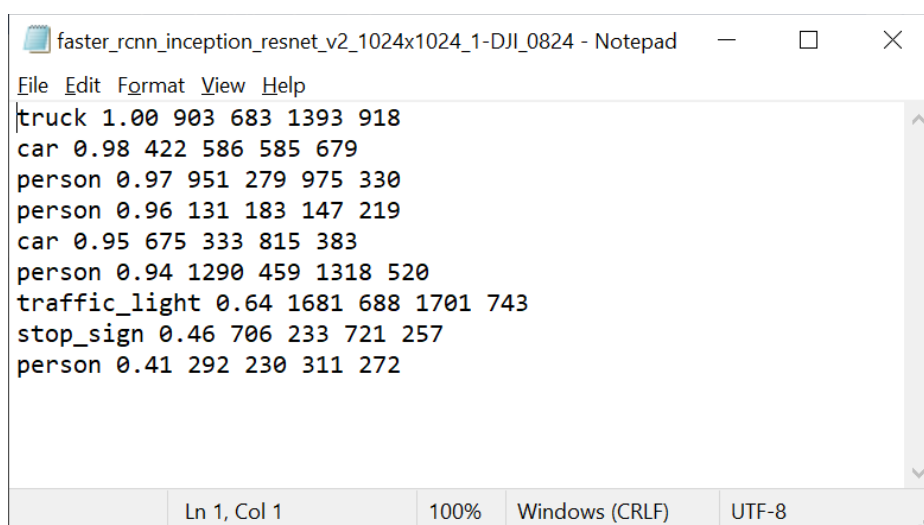


```

efficientdet_d7_1-21 - Notepad
File Edit Format View Help
car 0.76 3104 662 3217 735
car 0.74 2561 919 2661 1025
car 0.74 3171 748 3296 831
truck 0.72 2759 899 2983 1138
car 0.57 3000 576 3096 645
truck 0.52 2981 1097 3199 1483
truck 0.47 2712 1268 2904 1480
car 0.40 2625 1055 2758 1184
car 0.37 2938 485 3015 561
car 0.33 2327 704 2466 799
car 0.32 2342 32 2463 90
truck 0.30 2325 704 2467 799
Ln 1, Col 1 100% Windows (CRLF) UTF-8

```

PICTURE 22. Detection result text file by model EfficientDet D7 1536x1536



```

faster_rcnn_inception_resnet_v2_1024x1024_1-DJI_0824 - Notepad
File Edit Format View Help
truck 1.00 903 683 1393 918
car 0.98 422 586 585 679
person 0.97 951 279 975 330
person 0.96 131 183 147 219
car 0.95 675 333 815 383
person 0.94 1290 459 1318 520
traffic_light 0.64 1681 688 1701 743
stop_sign 0.46 706 233 721 257
person 0.41 292 230 311 272
Ln 1, Col 1 100% Windows (CRLF) UTF-8

```

PICTURE 23. Detection result text file by model Faster R-CNN Inception ResNet V2 1024x1024

“code_model” had, to begin with, a confidence threshold of 30 % to decide whether a detected object was going to be marked on the image or not. In this work, threshold was kept as 0.3, since images taken in the SURE project would be from further away and higher threshold values than 0.3 would not be enough to obtain a healthy view of the whole picture.

It was observed that during a model’s process, sometimes it would detect an object with 2 classes and frame it with both. In these cases, the class with the higher confidence would be saved in the detection results text file.

6.5 Evaluation results

For the sake of a realistic analysis, instead of considering only one evaluation value (mAP@0.5), mAP was calculated within the range of “0.5, 0.95, 0.05”, meaning that 10 APs were calculated with an IoU starting from 0.5 to 0.95, 0.05 being the increment. These 10 APs were then simply averaged to find the value of mAP@(0.5, 0.95, 0.05).

“code_mAP” was provided with coinciding ground truth and detection result files of each model and ran for 10 IoU values as mentioned above. Results for each AP, for each object class were documented per model. Complete chart of these findings, with average image processing times, can be seen in appendix 5.

Since the SURE project’s focus area involves detection of people and vehicles, top 4 models with the most accuracy can be extracted from the complete chart, including these specific results (table 1).

TABLE 1. Four most accurate models with mAP results

Model name	avg_time (min/image)	Class name	Rank	mAP @ (.50, .95, .05) (%)
EfficientDet D7 1536x1536	2,73	bus	3	54,42
		car	2	55,40
		person	2	44,98
		truck	2	30,38
		overall avg.	1	35,43

EfficientDet D6 1280x1280	2,61	bus	1	59,67
		car	1	56,33
		person	1	47,01
		truck	1	31,59
		overall avg.	2	35,20
EfficientDet D5 1280x1280	2,06	bus	2	54,93
		car	3	50,20
		person	5	38,06
		truck	4	26,20
		overall avg.	3	30,40
Faster R-CNN Inception ResNet V2 1024x1024	1,48	bus	5	49,69
		car	8	43,04
		person	6	36,59
		truck	3	26,91
		overall avg.	4	27,41

Three out of six EfficientNet models shared the top three ranks in terms of accuracy. The 4th coming model, Faster R-CNN Inception ResNet V2 1024x1024, is also added to this table because it has relatively high mAP results with a much better speed than the top three models. Comparison between these four models can be observed more clearly in figure 2.

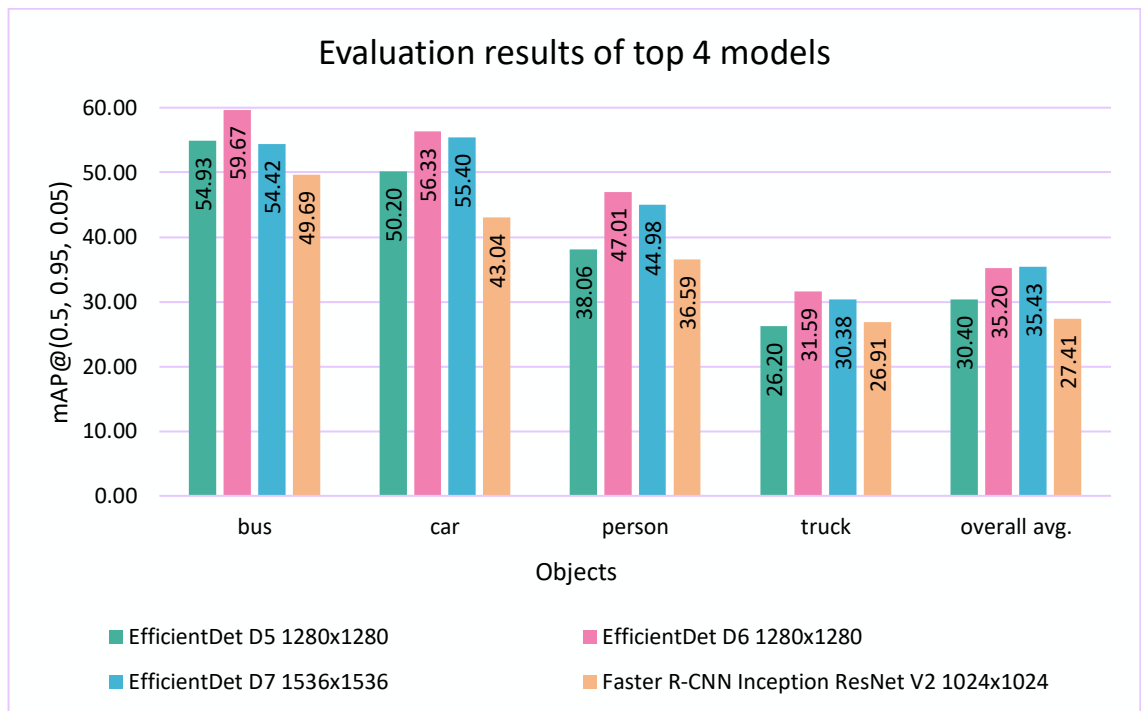


FIGURE 2. Evaluation results of top 4 models

The speed of the models is another metric that should be considered.

Therefore, table 2 is also extracted from the complete chart which shows the top three models that excel in speed.

TABLE 2. Three fastest models with mAP results

Model name	avg_time (min/image)	Class name	Rank	mAP @ (.50, .95, .05) (%)
CenterNet Resnet50 V1 FPN Keypoints 512x512	0,56	bus	15	1,02 %
		car	16	20,78 %
		person	16	17,32 %
		truck	16	2,80 %
		overall avg.	16	6,59 %
CenterNet Resnet50 V2 Keypoints 512x512	0,56	bus	16	1,02 %
		car	17	20,78 %
		person	17	17,32 %
		truck	17	2,80 %
		overall avg.	17	6,59 %
Faster R-CNN ResNet101 V1 800x1333	0,75	bus	8	39,95 %
		car	9	41,82 %
		person	7	35,77 %
		truck	5	24,43 %
		overall avg.	7	24,91 %

Detailed analysis of these results can be found in the discussion section.

7 DISCUSSION

Most of the models that were selected for evaluation were at the top of the COCO 2017 challenge chart, having successful results in terms of accuracy and speed in object detection. Considering the methodology behind all the models, each newer method used in them brought a better and more efficient technic into the field.

As a result of this study, it was found that the following three models were at the top of the list in terms of **accuracy**:

1. EfficientDet D7 1536x1536
2. EfficientDet D6 1280x1280
3. EfficientDet D5 1280x1280

and the following three models were at the top of the list in terms of image processing **speed**:

1. CenterNet Resnet50 V1 FPN Keypoints 512x512
2. CenterNet Resnet50 V2 Keypoints 512x512
3. Faster R-CNN ResNet101 V1 800x1333

CenterNet and EfficientNet models had shared the highest ranks in the challenge chart to begin with. Three out of six EfficientNet models came up as the top three in mAP scores. Although EfficientDet D7 1536x1536 is the 1st only by looking at its mAP@(0.5, 0.95, 0.05) score, EfficientDet D6 1280x1280 is the model that has the highest average mAPs for the specific objects that were focused on: bus, car, person and truck. For this reason, it can be suggested that EfficientDet D6 1280x1280 is more suitable for the SURE project than EfficientDet D7 1536x1536.

Four out of six CenterNet models came as the last four in mAP scores which indicates that their object detection skills were not precise enough for the tiny objects in the test images. Two out of six CenterNet models came as 5th and 6th

with relatively good mAP scores however, they were the top two slowest models.

Faster R-CNN models were expected to stand out from the R-CNN family. Although Mask R-CNN is the best ranking in locating objects in the family, its application area is too specific, it is slower than Faster R-CNN and does not necessarily improve the previous algorithm. In the light of all this, it was not unexpected that a relatively old Faster R-CNN model, Faster R-CNN Inception ResNet V2 1024x1024, would be in the higher ranks (4th) of the list, with a high mAP score and high speed.

Listing the results in terms of the models' speed though, drew a rather disappointing picture. Only one of the fastest models had a decent detection accuracy and this model was also a Faster R-CNN model. Therefore, the decision of choosing the most suitable model can not be based on the speed of the models but primarily on their accuracy.

In conclusion, this work gives two suggestions for the suitable object detection model for the SURE project. First one is EfficientDet D6 1280x1280, if the object detection's primary concern is accuracy. Second one is Faster R-CNN Inception ResNet V2 1024x1024, if the primary concern is speed.

REFERENCES

ArcGIS Developers. n. d. How RetinaNet works? Website article. Read 9.4.2021. <https://developers.arcgis.com/python/guide/how-retinanet-works/>

ArcGIS Developers. n. d. How Mask R-CNN Works? Website article. Read 16.4.2021. <https://developers.arcgis.com/python/guide/how-maskrcnn-works/>

Arlen, T. C. 2018. Understanding the mAP Evaluation Metric for Object Detection. Website article, Medium. Published 1.3.2018. Read 11.4.2021. <https://medium.com/@timothycarlen/understanding-the-map-evaluation-metric-for-object-detection-a07fe6962cf3>

Cartucho, J., Ventura R. & Veloso, M. 2018. Robust Object Recognition Through Symbiotic Deep Learning In Mobile Robots. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2018): 2336-2341. Source code in <https://github.com/Cartucho/mAP>

Gandhi, R. 2018. R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms. Website article, Towards Data Science. Published 9.7.2018. Read 22.3.2021. <https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e>

Girshick, R., Donahue, J., Darrell, T. & Malik, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proc. IEEE Conf. on computer vision and pattern recognition (CVPR), 580-587.

Girshick, R. 2015. Fast R-CNN. In Proc. IEEE Intl. Conf. on computer vision, 1440-1448.

He, Kaiming, Georgia Gkioxari, Piotr Dollár & Ross B. Girshick. 2017. Mask R-CNN. 2017 IEEE International Conference on Computer Vision. 2980-2988.

Lin, T., Goyal, P., Girshick, R.B., He, K., & Dollár, P. 2020. Focal Loss for Dense Object Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 42, 318-327.

Nelson, J. 2020. Evaluating Object Detection Models with mAP by Class. Blog article, Roboflow. Published 3.11.2020. Read 6.4.2021. <https://blog.roboflow.com/mean-average-precision-per-class/>

Papers With Code. n. d. EfficientNet. Website article. Read 9.4.2021. <https://paperswithcode.com/method/efficientnet>

Papers With Code. n. d. Object detection. Website article. Read 12.4.2021. <https://paperswithcode.com/task/object-detection>

Pawangfg. 2020. Selective search for object detection | R-CNN. Geeks for Geeks. Website article. Published 26.2.2020. Read 12.4.2021. <https://www.geeksforgeeks.org/selective-search-for-object-detection-r-cnn/>

Ren, S., He, K., Girshick, R. B., & Sun, J. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39, 1137-1149.

Swapna, K. E. 2020. Convolutional Neural Network | Deep Learning. Developers Breach. Website article. Published 21.8.2020. Read 22.3.2021. <https://developersbreach.com/convolution-neural-network-deep-learning/>

Tan, M. & Le, Q. V. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. Published 2019. ArXiv abs/1905.11946 n. pag.

Tensorflow models repository, <https://github.com/tensorflow/models>

The TensorFlow Hub Authors. 2020. Source code, TensorFlow Hub Object Detection Colab. Read 11.4.2021. https://colab.research.google.com/github/tensorflow/hub/blob/master/examples/colab/tf2_object_detection.ipynb#scrollTo=rOvvWAVTkMR7

Tzelepi, M. & Tefas, A. 2017. Human crowd detection for drone flight safety using convolutional neural networks. 25th European Signal Processing Conference (EUSIPCO), 743-747.

Tzutalin, 2015. Github repository. Read 20.4.2021. <https://github.com/tzutalin/labelImg>

Weng, L. 2017. Object Detection for Dummies Part 3: R-CNN Family. Blog article. Published 31.12.2017. Read 21.3.2021. <https://lilianweng.github.io/lil-log/2017/12/31/object-recognition-for-dummies-part-3.html>

Yelisetty, A. 2020. Understanding Fast R-CNN and Faster R-CNN for Object Detection. Website article, Towards Data Science. Published 13.7.2020. Read 25.03.2021. <https://towardsdatascience.com/understanding-fast-r-cnn-and-faster-r-cnn-for-object-detection-adbb55653d97>

Zhou, X., Wang, D. & Krähenbühl, P. 2019. Objects as Points. ArXiv abs/1904.07850: n. pag.

APPENDICES

Appendix 1. Complete list of pre-trained models with TensorFlow 2 for COCO 2017 dataset, sorted by mAP results. Marked ones are used in this work.

	Model name	Speed (ms)	COCO mAP	Outputs
1	CenterNet HourGlass104 Keypoints 1024x1024	211	42.8/64.5	Boxes/Keypoints
2	CenterNet HourGlass104 Keypoints 512x512	76	40.0/61.4	Boxes/Keypoints
3	Mask R-CNN Inception ResNet V2 1024x1024	301	39.0/34.6	Boxes/Masks
4	CenterNet Resnet50 V1 FPN Keypoints 512x512	30	29.3/50.7	Boxes/Keypoints
5	CenterNet Resnet50 V2 Keypoints 512x512	30	27.6/48.2	Boxes/Keypoints
6	EfficientDet D7 1536x1536	325	51,2	Boxes
7	EfficientDet D6 1280x1280	268	50,5	Boxes
8	EfficientDet D5 1280x1280	222	49,7	Boxes
9	EfficientDet D4 1024x1024	133	48,5	Boxes
10	EfficientDet D3 896x896	95	45,4	Boxes
11	CenterNet HourGlass104 1024x1024	197	44,5	Boxes
12	CenterNet HourGlass104 512x512	70	41,9	Boxes
13	EfficientDet D2 768x768	67	41,8	Boxes
14	CenterNet MobileNetV2 FPN Keypoints 512x512	6	41,7	Keypoints
15	SSD ResNet152 V1 FPN 1024x1024 (RetinaNet152)	111	39,6	Boxes
16	SSD ResNet101 V1 FPN 1024x1024 (RetinaNet101)	104	39,5	Boxes
17	Faster R-CNN Inception ResNet V2 1024x1024	236	38,7	Boxes
18	EfficientDet D1 640x640	54	38,4	Boxes
19	SSD ResNet50 V1 FPN 1024x1024 (RetinaNet50)	87	38,3	Boxes
20	Faster R-CNN Inception ResNet V2 640x640	206	37,7	Boxes
21	Faster R-CNN ResNet152 V1 1024x1024	85	37,6	Boxes
22	Faster R-CNN ResNet152 V1 800x1333	101	37,4	Boxes
23	Faster R-CNN ResNet101 V1 1024x1024	72	37,1	Boxes
24	Faster R-CNN ResNet101 V1 800x1333	77	36,6	Boxes
25	SSD ResNet101 V1 FPN 640x640 (RetinaNet101)	57	35,6	Boxes
26	SSD ResNet152 V1 FPN 640x640 (RetinaNet152)	80	35,4	Boxes
27	SSD ResNet50 V1 FPN 640x640 (RetinaNet50)	46	34,3	Boxes
28	CenterNet Resnet101 V1 FPN 512x512	34	34,2	Boxes
29	EfficientDet D0 512x512	39	33,6	Boxes
30	Faster R-CNN ResNet152 V1 640x640	64	32,4	Boxes
31	Faster R-CNN ResNet101 V1 640x640	55	31,8	Boxes
32	Faster R-CNN ResNet50 V1 800x1333	65	31,6	Boxes
33	CenterNet Resnet50 V1 FPN 512x512	27	31,2	Boxes
34	Faster R-CNN ResNet50 V1 1024x1024	65	31	Boxes
35	CenterNet Resnet50 V2 512x512	27	29,5	Boxes
36	Faster R-CNN ResNet50 V1 640x640	53	29,3	Boxes
37	SSD MobileNet V1 FPN 640x640	48	29,1	Boxes
38	SSD MobileNet V2 FPN Lite 640x640	39	28,2	Boxes
39	CenterNet MobileNetV2 FPN 512x512	6	23,4	Boxes
40	SSD MobileNet V2 FPN Lite 320x320	22	22,2	Boxes
41	SSD MobileNet v2 320x320	19	20,2	Boxes

Appendix 2. Additional codes made to “code_models” and models\research\object_detection\utils\visualization_utils.py (Tensorflow models repository):

- In object_detection_gamze.py file, lines 241&242
- In visualization_utils.py file, lines 1121&1122, 1183, 1210–1229, 1257–1274

```

object_detection_gamze.py × rename_gt.py × visualization_utils.py × mscoco_label_map.pbtxt × main.py ×
227
228     viz_utils.visualize_boxes_and_labels_on_image_array(
229         image_np_with_detections[0],
230         result['detection_boxes'][0],
231         (result['detection_classes'][0] + label_id_offset).astype(int),
232         result['detection_scores'][0],
233         category_index,
234         use_normalized_coordinates=True,
235         max_boxes_to_draw=200,
236         min_score_thresh=threshold,
237         agnostic_mode=False,
238         keypoints=keypoints,
239         keypoint_scores=keypoint_scores,
240         keypoint_edges=COCO17_HUMAN_POSE_KEYPOINTS,
241         model_path=model_path,
242         pic_name=pic_name)

```

```

object_detection_gamze.py* × rename_gt.py × visualization_utils.py ×
1097
1098     def visualize_boxes_and_labels_on_image_array(
1099         image,
1100         boxes,
1101         classes,
1102         scores,
1103         category_index,
1104         instance_masks=None,
1105         instance_boundaries=None,
1106         keypoints=None,
1107         keypoint_scores=None,
1108         keypoint_edges=None,
1109         track_ids=None,
1110         use_normalized_coordinates=False,
1111         max_boxes_to_draw=20,
1112         min_score_thresh=.5,
1113         agnostic_mode=False,
1114         line_thickness=4,
1115         mask_alpha=.4,
1116         groundtruth_box_visualization_color='black',
1117         skip_boxes=False,
1118         skip_scores=False,
1119         skip_labels=False,
1120         skip_track_ids=False,
1121         model_path='x',
1122         pic_name='x'):

```



```

object_detection_gamze.py × rename_gt.py × visualization_utils.py ×
1180     box_to_keypoints_map = collections.defaultdict
1181     box_to_keypoint_scores_map = collections.defaultdict
1182     box_to_track_ids_map = {}
1183     name_and_score = []
1184     all_to_text_file = []
1185     if not max_boxes_to_draw:

```

```

object_detection_gamze.py × rename_gt.py × visualization_utils.py × mscoco_label_map.pbtxt × main.py × remove_first_line_and_tabs.py ×
1207         if not agnostic_mode:
1208             if classes[i] in six.viewkeys(category_index):
1209                 class_name = category_index[classes[i]]['name']
1210                 # Gamze
1211                 if class_name == 'traffic light':
1212                     class_name = 'traffic_light'
1213                 if class_name == 'fire hydrant':
1214                     class_name = 'fire_hydrant'
1215                 if class_name == 'stop sign':
1216                     class_name = 'stop_sign'
1217                 if class_name == 'parking meter':
1218                     class_name = 'parking_meter'
1219                 if class_name == 'sports ball':
1220                     class_name = 'sports_ball'
1221                 if class_name == 'cell phone':
1222                     class_name = 'cell_phone'
1223                 if class_name == 'baseball glove':
1224                     class_name = 'baseball_glove'
1225                 if class_name == 'potted plant':
1226                     class_name = 'potted_plant'
1227                 if class_name == 'dining table':
1228                     class_name = 'dining_table'
1229                 # more 2-word class names if necessary
1230                 class_name_and_score_to_txt = class_name + ' ' + '{score:.2f}'.format(score = scores[i])
1231                 name_and_score.append(class_name_and_score_to_txt)
1232             else:

```

```

object_detection_gamze.py × rename_gt.py × visualization_utils.py × mscoco_label_map.pbtxt × main.py × remove_first_line_and_tabs.py ×
1255
1256     # Draw all boxes onto image.
1257     x = 0
1258     for box, color in box_to_color_map.items():
1259         ymin, xmin, ymax, xmax = box
1260
1261         xmin_to_txt = '{xmn}'.format(xmn = round(xmin*1920))
1262         ymin_to_txt = '{ymn}'.format(ymn = round(ymin*1080))
1263         xmax_to_txt = '{xmx}'.format(xmx = round(xmax*1920))
1264         ymax_to_txt = '{ymx}'.format(ymx = round(ymax*1080))
1265         """
1266         xmin_to_txt = '{xmn}'.format(xmn = round(xmin*4056))
1267         ymin_to_txt = '{ymn}'.format(ymn = round(ymin*2280))
1268         xmax_to_txt = '{xmx}'.format(xmx = round(xmax*4056))
1269         ymax_to_txt = '{ymx}'.format(ymx = round(ymax*2280))
1270         """
1271         coordinates_to_txt = xmin_to_txt + ' ' + ymin_to_txt + ' ' + xmax_to_txt + ' ' + ymax_to_txt
1272         whole_line_to_txt = name_and_score[x] + ' ' + coordinates_to_txt + '\n'
1273         all_to_text_file.append(whole_line_to_txt)
1274         x = x + 1 # end

```

Appendix 3. List of object class names in MS COCO dataset

1-16	17-32	33-48	49-64	65-80
person	dog	sports_ball	sandwich	mouse
bicycle	horse	kite	orange	remote
car	sheep	baseball_bat	broccoli	keyboard
motorcycle	cow	baseball_glove	carrot	cell_phone
airplane	elephant	skateboard	hot_dog	microwave
bus	bear	surfboard	pizza	oven
train	zebra	tennis_racket	donut	toaster
truck	giraffe	bottle	cake	sink
boat	backpack	wine_glass	chair	refrigerator
traffic_light	umbrella	cup	couch	book
fire_hydrant	handbag	fork	potted_plant	clock
stop_sign	tie	knife	bed	vase
parking_meter	suitcase	spoon	dining_table	scissors
bench	frisbee	bowl	toilet	teddy_bear
bird	skis	banana	tv	hair_drier
cat	snowboard	apple	laptop	toothbrush

Appendix 4. Additional code piece, for renaming ground truth text files for each model.

```
import os
import chardet

new_model = 'centernet_hourglass104_512x512_1'

path_gt = 'C:/Users/omistaja/Desktop/ml/ground_truth/DJI_0001-DJI_0138/' #DJI_0142-DJI_0847
path_new = 'C:/Users/omistaja/Desktop/ml/mAP-master/input/z_' + new_model + '/ground-truth/'

files = [os.path.splitext(filename)[0] for filename in os.listdir(path_gt)]

for i in files:
    if os.path.isfile(path_gt + i + '.txt'):
        with open(path_gt + i + '.txt', 'r') as rf:
            with open(path_new + new_model + '-' + i + '.txt', 'w') as wf:
                for line in rf:
                    wf.write(line)
```

Appendix 5. Complete chart consisting of image processing times, ranks and mAP scores of each model, sorted by models names in alphabetical order.

Model name	Avg image process time (min)	Class name	Ranking	mAP@ (.50, .95, .05) (%)	AP ^{IoU=50} (%)	AP ^{IoU=55} (%)	AP ^{IoU=60} (%)	AP ^{IoU=65} (%)	AP ^{IoU=70} (%)	AP ^{IoU=75} (%)	AP ^{IoU=80} (%)	AP ^{IoU=85} (%)	AP ^{IoU=90} (%)	AP ^{IoU=95} (%)
CenterNet HourGlass104 Keypoints 512x512	2,59	bench	12	3,00	3,33	3,33	3,33	3,33	3,33	3,33	3,33	3,33	3,33	0,00
		bicycle	9	17,08	29,22	29,22	29,22	24,04	24,04	17,95	12,30	4,76	0,00	0,00
		boat	12	6,33	17,81	15,86	15,86	6,36	3,81	3,03	0,53	0,00	0,00	0,00
		bus	14	5,29	6,87	6,87	6,87	6,87	6,87	6,87	5,77	2,56	2,56	0,77
		car	13	33,37	47,76	47,40	46,96	46,47	45,19	43,40	35,93	15,91	3,81	0,84
		dog	11	34,71	45,50	45,50	45,50	39,95	37,57	37,57	33,33	33,33	22,45	6,35
		motorcycle	-	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
		person	9	31,60	53,40	51,77	50,42	46,03	39,28	30,07	23,36	13,97	5,97	1,70
		truck	14	6,88	10,82	10,60	10,16	9,95	9,95	9,95	3,56	2,94	0,84	0,00
		mAP	14	15,36	23,86	23,39	23,15	20,33	18,89	16,91	13,12	8,53	4,33	1,07
CenterNet HourGlass104 Keypoints 1024x1024	3,80	bench	3	18,04	25,28	21,39	21,39	21,39	21,39	21,39	16,23	16,23	13,64	2,06
		bicycle	5	23,30	45,61	36,53	36,53	34,14	26,36	23,95	18,15	11,11	0,62	0,00
		boat	10	7,67	16,95	15,61	13,17	10,48	10,48	6,39	3,33	0,26	0,00	0,00
		bus	10	32,57	39,05	39,05	39,05	39,05	39,05	39,05	32,94	29,98	24,65	3,85
		car	4	48,43	65,41	64,91	64,50	64,28	62,52	58,44	50,41	34,13	18,38	1,33
		dog	5	48,02	66,95	66,95	66,95	60,92	56,26	48,51	43,45	36,92	30,92	2,38
		motorcycle	-	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
		person	4	43,87	67,21	66,36	64,59	61,32	57,14	46,34	36,22	24,72	12,15	2,69
		truck	12	11,39	17,01	16,00	16,00	14,25	13,50	13,20	10,85	8,55	4,31	0,25
		mAP	5	25,92	38,16	36,31	35,80	33,98	31,85	28,59	23,51	17,99	11,63	1,39
CenterNet HourGlass104 512x512	1,84	bench	8	7,20	8,67	8,67	8,67	8,67	8,67	8,67	6,67	6,67	6,67	0,00
		bicycle	12	15,64	25,93	25,93	25,93	21,16	17,90	17,90	17,90	3,70	0,00	0,00
		boat	13	5,08	14,20	14,20	12,87	3,56	2,59	2,59	0,37	0,37	0,00	0,00
		bus	17	0,77	0,96	0,96	0,96	0,96	0,96	0,96	0,96	0,96	0,96	0,00
		car	15	31,50	47,06	46,59	45,80	45,09	42,96	37,10	29,47	15,76	4,52	0,66
		dog	13	28,73	41,90	41,90	37,57	30,91	30,91	30,91	30,91	25,62	14,02	2,65
		motorcycle	-	1,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
		person	8	31,64	53,08	51,14	48,35	43,79	39,14	32,18	22,80	15,45	8,11	2,32
		truck	15	6,05	10,45	9,63	9,18	8,40	8,17	8,17	4,28	1,83	0,36	0,05
		mAP	15	14,07	22,47	22,11	21,04	18,06	16,81	15,39	12,60	7,82	3,74	0,63
CenterNet HourGlass104 1024x1024	3,12	bench	4	14,13	15,00	15,00	15,00	15,00	15,00	15,00	15,00	15,00	11,33	10,00
		bicycle	4	26,68	46,14	46,14	46,14	41,36	31,48	31,48	18,82	4,63	0,31	0,31
		boat	5	15,03	38,06	30,90	30,90	19,46	18,50	6,95	5,35	0,19	0,00	0,00
		bus	12	24,99	30,91	30,91	30,91	30,91	30,91	30,91	24,18	24,18	16,09	0,00
		car	7	46,11	64,62	64,27	63,99	63,23	60,80	51,23	42,57	32,34	16,51	1,54
		dog	6	45,27	60,11	60,11	60,11	55,34	55,34	55,34	43,10	39,00	15,87	8,33
		motorcycle	-	2,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
		person	3	44,26	67,59	66,62	64,81	60,53	55,11	47,85	37,95	24,55	14,07	3,48
		truck	11	11,42	17,18	16,55	15,41	15,41	13,29	13,29	10,68	8,22	3,56	0,61
		mAP	6	25,32	37,73	36,72	36,36	33,47	31,16	28,01	21,96	16,46	8,64	2,70
CenterNet Resnet50 V1 FPN Keypoints 512x512	0,56	bench	16	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
		bicycle	16	1,11	3,70	3,70	3,70	0,00	0,00	0,00	0,00	0,00	0,00	0,00
		boat	16	0,56	1,85	1,85	1,85	0,00	0,00	0,00	0,00	0,00	0,00	0,00
		bus	15	1,02	1,28	1,28	1,28	1,28	1,28	1,28	1,28	1,28	0,00	0,00
		car	16	20,78	35,53	35,08	34,02	33,41	31,06	18,28	11,58	6,78	1,91	0,13
		dog	15	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
		motorcycle	-	15,72	23,81	23,81	23,81	23,81	19,05	19,05	19,05	4,76	0,00	0,00
		person	16	17,32	33,91	31,18	27,71	23,30	19,54	15,54	12,37	6,79	2,30	0,51
		truck	16	2,80	4,26	4,26	3,78	3,78	3,64	3,64	3,53	0,61	0,50	0,00
		mAP	16	6,59	11,59	11,24	10,68	9,51	8,28	6,42	5,31	2,25	0,52	0,07

