Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Automating nut tightening using Machine Learning

**Kevin Wedin*, Christoffer Johnsson*, Magnus Åkerman*,
Åsa Fast-Berglund*, Viktor Bengtsson* and Per-Anders Alveflo****

**Chalmers University of Technology, SE-412 96*
***Volvo Trucks Cooperation*

Abstract: At the Volvo Truck assembly plant the repetitive task of nut tightening is not ideal regarding quality and ergonomic. The solution to both these issues would be to significantly increase the level of automation. However, automating this specific station requires solutions to two specific problems. The first problem is to find and identify what nuts that need to be tightened, since they are not always on the same position for this highly customized product. The second problem is that the automated solution needs to accommodate the working space which is a moving assembly line with human operators. This paper investigates how these two problems ban be solved using machine learning and collaborative robots. A realistic mockup of the assembly station has been created at Stena Industry Innovation Laboratory (SII-Lab) where all the testing has been done.

The problem to identify the nuts to tighten is further complicated by the fact that some nuts are placed backwards for future further assembly which must be avoided. Therefore, the selected solution is to use supervised machine learning for object recognition. This way, the system can be trained to recognize both nuts that need to be tightened and those mounted backwards, and possible other objects needed. Tests have been conducted with different types of CNN (Convolutional Neural Network) algorithms. Results have been very successful, and the test setup has successfully managed to connect the whole task of identifying the correct nuts and move the collaborative robot to that specific position.

Keywords: Machine learning, assembly, collaborative robot.

INTRODUCTION

In the era of industry 4.0, a lot of technology is available in order to achieve higher productivity, improve ergonomics and increase quality. Between 2003 and 2009 forty-nine different technologies were presented in the Gartner hype curve of evolving technologies which have built the foundation towards industry4.0. In 2017, nine developing technologies were presented (Bortolini et al., 2017). Cohen et. al (Cohen et al., 2019) divide the technologies further into software and hardware. Usually there is a mix between these technologies in order to achieve a good result. This paper there will bring up examples of machine learning and object recognition and collaborative robot application.

Machine learning can be described as "a cluster of statistical and programming techniques that give computers the ability to 'learn' from exposure to data, without being explicitly programmed" (Sag, 2019). Object detection is a computer vision technique that tries to solve the problems of both object classification and object localisation. A successful approach for these problems has been to utilise a machine learning approach with Convolutional Neural Networks (CNN) (Krizhevsky et al., 2012).

Collaborative robot application are industrial robots that are designed to work along humans in various levels of interaction (Bauer et al., 2016) i.e. coexistence, synchronized, cooperation, and collaboration. Coexistence means that there is no shared workspace at all. In a synchronized application, the human and robot share the same workspace but never at the same time. Cooperation means that they do work in the same workspace at the same time but not with the same component. A true collaborative application is when robot and human both do work at the same time with the same component.

Collaborative robots are an integral part of future intelligent production systems that allows smaller lot sizes and increased productivity (Rüßmann et al., 2015).

This paper presents the results of the experiment setup and discusses the results in terms of feasibility of implementing a live application.

OBJECT DETECTION USING MACHINE LEARNING

A neural network is a common machine learning approach where the input is propagated through layers of connected neurons. How these neurons are connected are decided by weighted values that are decided though training of the algorithm. Since using every pixel from the images as direct input for fully connected neural networks would be too computational heavy, CNN's consists of two separate parts: feature learning and classification. During feature learning a small part of the image is filtered and simplified and the result consists of several small feature maps. These feature

maps are then flattened and sent though fully connected neural network to classify what the features represent.

There are three different strategies when training machine learning agents: unsupervised learning, reinforcement learning, and supervised learning. In unsupervised learning the agents gradually detect patterns in the input data and forms potentially useful clusters. Reinforcement learning means that the agent is "rewarded" or "punished" depending on output value. An agent receiving supervised learning gets a training set containing input data and corresponding output values. If the output value is part of a finite set, e.g. is an image a dog or a cat, it is a solution to a classification problem. Values that are real numbers, e.g. tomorrows stock market, are solutions to regression problems. (Russell and Norvig, 2013). An early viable method is called R-CNN, which focus on the classification problem by dividing the image into many sub sections (region proposals) (Girshick et al., 2014). Since the number of proposals generated for each image can be very large, this is method is rather computationally heavy, but improvements in methods Fast R-CNN (Girshick, 2015) and Faster R-CNN (Ren et al., 2015) have reduced the computation and training time significantly. It is also possible to approach the object localisation problem as a regression problem, which is the case for the Single Shot Detector (SSD) algorithm (Liu et al., 2016). For a more in-depth view of the evolution of various approaches see (Zhao et al., 2019).

To explain a complete background of the machine learning and computer vision concepts touched upon during this experiment is beyond the scope. *Table 1* constitutes a description of needed concepts.

Table 1. Computer vision and machine learning concepts.

Concept	Description
COCO	COCO (http://cocodataset.org/) stands for Common Objects in Context and is an image reference database developed by Microsoft (Lin et al., 2014). It consists of a large set of labelled images with 1.5 million object instances and 80 object categories. It IS used to train and test computer vision applications.
OpenCV	OpenCV (https://opencv.org/) is an open source library for computer vision applications (Bradski and Kaehler, 2008).
Tensorflow	Tensorflow (https://www.tensorflow.org/) is an open-source machine learning platform. It support training and execution of machine learning algorithms in large scale heterogeneous systems (Abadi et al., 2016).
LabelImg	A labelling software that allows the user to put label bounding boxes around objects in pictures to be used for supervised training (Tzutalin, 2015).

INDUSTRIAL CASE

At one of the stations at the Volvo Truck's assembly line the task is to tighten most of the previously entered nuts along the length of the truck frame. Each truck frame has two sides with nuts that need tightening. Some nuts are placed backwards to accommodate future components, these are not to be tightened. Each side contains an average of 200 nuts and around 30 number of inverted nuts. The truck is slowly moving over the assembly station during the entire tact time which is 5 minutes. There are two operators on each side working with tightening tools. They divide the frame into an upper and a lower level to avoid tightening the same nuts.

This task is done by operators using a power tool and leads to two separate issues. One issue regards the manual task of moving and holding on to the tool, which is repetitive and unergonomic. The other issue regards to quality since it is possible for an operator to overlook nuts. Increasing the level of automation can be a solution to both these problems. The ergonomic issues can be solved if the automation level is increased to exclude the human operator from the physical task of moving and holding on to the tool. The quality can be improved if cognitive automation can identify nuts and remember what nuts have been tightened.

EXPERIMENT SETUP

During the summer of 2019 an experiment setup was created at Stena Industry Innovation Lab (SII-Lab) to investigate the possibility to utilise collaborative robots together with computer vision to solve above mentioned issues. It was decided that the computer vision system should be based on machine learning since previous experience with more traditional tools was deemed unavailing. The experiment setup consisted of two parts; 1) Nut detection and 2) Robot application the concepts are tested separately to start with and then an interactive process is done at the end in order to integrate the two concepts, illustrated in (*Fig. 1*).

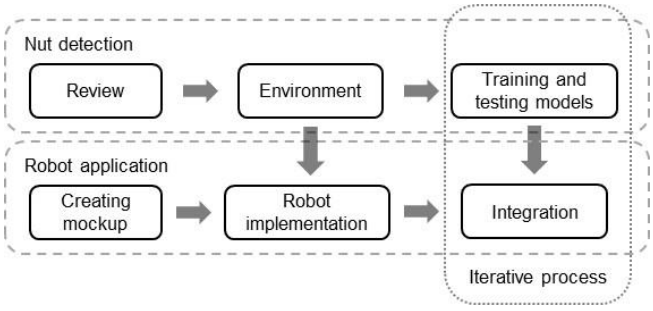

Fig. 1. Visualisation of the process of creating the proof of concept robot application.

4.1 Nut Detection

The nut detection process consisted of three parts; review of the field, setting up an environment, and training and testing of machine learning models, the results were the concepts presented in Table 1. Several Machine learning methods were evaluated. The machine learning methods that have been

tested are Faster R-CNN (Ren et al., 2015) and SSD (Liu et al., 2016).

In terms of the environment, the real tightening tool was not available for the experiment setup so a mockup was created using additive manufacturing. *Fig.* 2 shows the mock-up of the tightening tool that also has the camera integrated. The camera is a regular web-camera from Logitech.

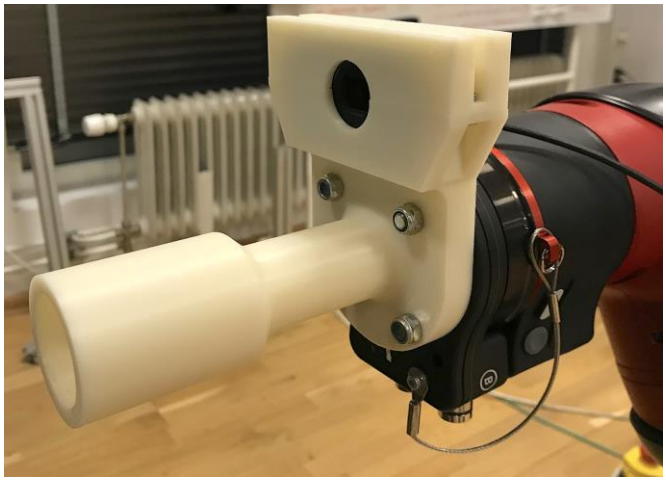

Fig. 2. Mockup of the tightening tool with camera attached to it.

Training and testing were an iterative process, but each iteration is like the other. Supervised training requires labelled images. Therefore, several images of the Truck Frame with nuts and inverted nuts was taken and labelled using LabelImg (Tzutalin, 2015). For Faster R-CNN the resolution of the images was 800x600 and for SSD it was 300x300. The images were all taken perpendicular to the Truck Frame, illustrated in *Fig. 3*.

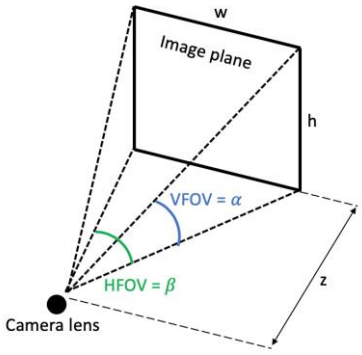

Fig. 3. Positions of the nuts are calculated using the distance to the truck frame and the field of view of the camera.

An important part of the robot application is to translate detected nuts in a two-dimensional image into the corresponding three-dimensional space. This was calculated using the known field of view of the camera and the distance between the camera and the frame. (*Fig. 3*).

4.2 Robot application

The robot application process is also divided into three parts; Creating a mock-up of the nut tightening tool, Implementation of robot programming and then integration of the detection result. needed to be created. After that, with valuable input from how the nut detection environment looks like, a concept of the robot implementation could be created including software, hardware and scope of the application.

The robot that is used for the implementation is a Sawyer from Rethink Robotics. the experiment setup (see *Fig.* 4) has some differences from the real assembly line. The truck frame is smaller and contains fewer other components, but the nuts and bolts are the same. The lighting conditions have not been measured but the experiences are similar. The frame is fixed while on the real assembly line the truck frame is slowly moving on a paced line.

Fig. 4. Robot application with truck frame at SII-Lab.

The ROS platform was used to communicate with the robot. Since ROS is installed on a Linux platform and Windows was used for the Tensorflow application, the software part of the robot application is setup using the client server approach. All the software is crated using the Python programming language.

Interactive process

Then an iterative process of integration took place where the application was improved in parallel to when improved models was trained. The implementations have been done using Tensorflow (Abadi et al., 2016). In the TensorFlow framework, training is setup in configuration files and the most common approach is to utilise a premade Tensorflow sample file. The configuration files "faster_rcnn_inception_v2_pets.config" and "ssd_mobilenet_v2_coco.config" was used for training the algorithms, which are originally optimised for parts of the COCO data set.

Fig. 2 shows the architecture of the robot application. A client handles the communication with the camera, takes pictures using the Open CV platform, and runs those pictures through the object detection algorithm with the help of Tensorflow. The client calculates the real location of the detected nuts and sends those to the server application which tells the robot to move to the position through ROS. The computer hardware used for the client application, that runs the Tensorflow platform, is just a normal computer and it does not utilise the GPU for computer vision tasks.

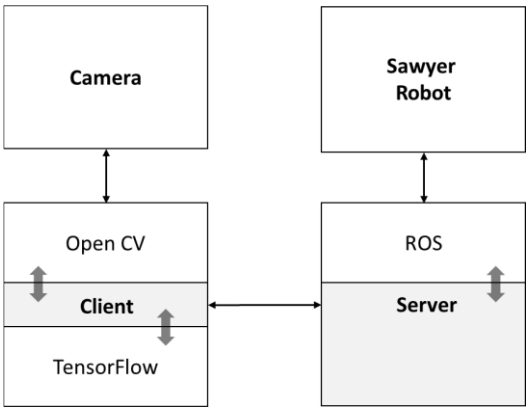

Fig. 2. Schematic picture of the robot application.

Training of an algorithm prints a result that includes a prediction of how well it performs. However, since the different training procedures did not use the same pictures for training and testing, the results were difficult to compare. Therefore, a separate hit rate test was done. In this test, 10 images were taken with seven nuts, seven inverted nuts, and two empty bolts in each image. This gives a total of 140 objects that we sent through each of the created algorithm. The hit rate is calculated by removing any missed or misidentified object, meaning that finding six or eight objects out of the correct seven result in the same hit rate of 85,7%.

When the algorithm detects nuts and inverted nuts correctly. *Fig. 3*. Shows the result of a correct executed object detection Nuts and inverted nuts are identified and boxed in different colours.

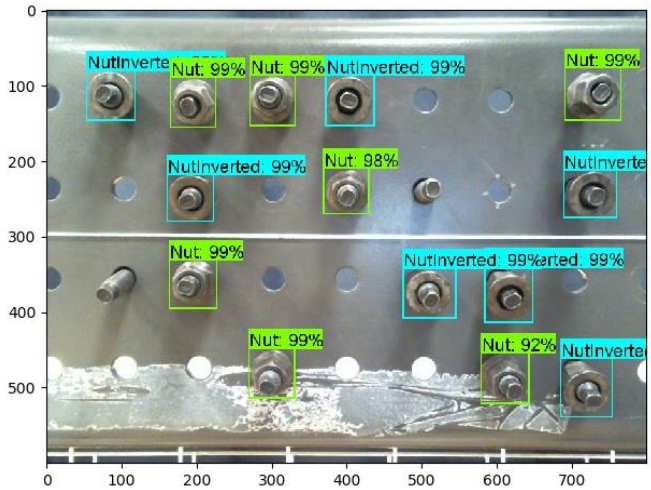

Fig. 3. Result of a correctly executed object detection.

RESULTS

Table 2 shows a summary of the four algorithms that was trained and tested, two using SSD and two using Faster R-CNN.

Table 2. Summary of the different algorithms trained and tested including total number of images and objects.

Test	Algorithm	#images	#Nuts	#Inverted Nuts
1	SSD	119	176	174
2	SSD	141	239	210
3	Faster R-CNN	124	353	128
4	Faster R-CNN	322	928	568

Table 3 shows the result from the hit rate test that tests each algorithm against 140 objects over 10 images. The hit rate varies between 81,4% and 97,1% for the nuts and 67,1% and 95,7% for the inverted nuts. The time it takes to run the SSD algorithm is about half compared to Faster R-CNN.

Table 3. Result from the hit rate test.

Test	Hit Rate Nut	Hit Rate Inverted Nut	Hit Rate Combined	Detection Speed
1	97,1%	37,1%	67,1%	2,02 s
2	81,4%	71,4%	76,4%	1,94 s
3	81,4%	81,4%	81,4%	4,45 s
4	95,7%	95,7%	95,7%	4,49 s

DISCUSSION

The purpose of the experiment setup was to investigate the possibility to automate the nut tightening task using collaborative robots. The results show, despite reduced complexity of the setup, that a successful implementation at the assembly is very attainable. The reasons for that assessment are based on x, y, and z.

Nut Detection using CNN

As can be seen in *Table 3* and *Table 2*, the hit rate of the machine learning algorithms are, not surprisingly, highly dependant on the number of objects that was available during training. The type of algorithm used does play some factor, but this experiment lacks the data to be conclusive. *Table 3* also shows that SSD is faster than Faster R-CNN, again, this was something already known. The trade off between speed and accuracy of machine learning approaches makes choosing the correct approach rely on the specific requirements of the application (Huang et al., 2017).

Test 4 reached the highest accuracy with a total hit rate of 95,7%. This is a significantly higher number than the 35,7% of mean accuracy found in a systematic comparison test

(Huang et al., 2017). That comparison is however based on the COCO data set and tires to detect many objects of various sizes and shapes. It is much easier to optimise an algorithm to only detect specific objects (Jiang and Learned-Miller, 2017). It is at this stage difficult to say what is possible in terms of accuracy but initial tests (see *Table 3*) are promising in terms of quality.

Speed of the object detection is not very relevant for this implementation. The results show a very low detection speed compared to other measures (Huang et al., 2017) and especially for the regression based methods (Liu et al., 2016). The difference is mostly because this test did not include GPU acceleration for any of the heavy calculation that computer vision is. Either way, the seconds that the image processing might add is insignificant compared other aspects such as robot movement and nut tightening tool.

Tensorflow (Abadi et al., 2016) was used for the implementation. It was not a difficult choice since it is the most popular open source platform for machine learning applications. However, it was not problem free. A large amount of time went into trying to find the correct versions of the different packages and software needed to create a functional application. This shows that implementing machine learning applications today has ha rather steep initial learning curve. There are also several other frameworks to look into (Shatnawi et al., 2018, Bahrampour et al., 2015).

Robot Application

The speed of the robot is hard to draw credible conclusions around because of the missing time for the tightening of the nuts. When the tool is available for testing the real speed for the session can be evaluated. But with illustration of the robot's movement we can draw the conclusion that the accuracy was good enough, which implies that the robot surrounds the nut with the tool without unwanted collision with the frame or the objects. There is a need to meet 1,5 seconds as average time spent on each nut with one working robot (to meet the maximum amount of 200 nuts per side). So, one solution is either having more robots working alongside, having a faster robot or having collaboration with humans.

Regarding the camera application, the camera's position potentially affected the results. The camera was mounted on the tool of the robot (Fig. 2), and therefore required the robot to move to a certain home position for each new image that needed to be taken by the camera. Thus, potential time losses in the overall speed per nut may occurred. Instead, if a fix camera mounted besides the robot was implemented, it wouldn't require the robot to move to a certain home position the for next image to be taken. Further, the camera wouldn't be limited to be near the robot; a camera implemented at an earlier station at the production line could provide the necessary images for the object detection and coordinates could further be delivered to the robot. However, a such system may require extra sensors to locate the frame in relation to the robot, which this project didn't investigated.

The communication between the robot and the different settings could be more efficient. One of the problems is the hardware without the possibility to utilise a GPU booster. The system in control of the translation of positions are as mentioned in as ROS. It is hard to compare between other systems when no further tests in that regard have been executed. The good thing is that the environment of the robot application shows that it's fully applicable to use ROS in this regard.

This project was performed in a laboratory environment which may entail differences with the real implementation. Additionally, some important aspects will therefore be discussed. First, the frame in the laboratory is fixed compared with the real environment where it moves horizontally on a conveyor. This should be taken into consideration for further implementation, and moreover the conveyor may not have consistent speed, due to e.g. minor stops on the production line, which put certain requirements on the flexibility of the robot application to handle a likewise situation. As a second aspect, the frame at the conveyor may be inconsistent in its placement regarding height and angels in relation to the robots coordinate system. This potential situation could not occur in the laboratory, since the frame was fixed in the laboratory. Therefore, there were no need for calibration of the robot camera in each new object detection session. Additionally, the real implementation would require some sort of sensors to locate the centre of the beam of the frame and adjust the camera to be orthogonal to the beam in that height. On the other hand, a different approach on solving this issue is discussed further down in this chapter.

Finally, a third aspect that needs to be taken into consideration for a real implementation is the potential objects located on the frame that might hinder the camera from partly or entirely capture some nuts, and therefore affect the object detection accuracy. This is something that will be considered when using the pilot testing at Volvo Trucks, since the real environment there have more objects than the ones seen in Figure 5. As Jiang and Learned-Miller (2017) mentioned, it's easier to optimise specific objects and not different objects with various sizes and design. Additionally, with the right amount of training and the same objects comes mounted on the frame, it can be possible to reach acceptable recognition level for automation. It will be harder to achieve if different objects occur on the frame that not been included in the training session.

The project was limited to the use of a camera mounted on the robot, which further was placed perpendicular in a certain distance to the frame for each new image. However, this could have been made differently by using live object detection via video stream, rather than analysis of a single image. A such system would entail for opportunities to let the robot search for nuts on the frame and not be limited to a perpendicular setting. Instead, the robot could systematically screen the frame from one end to the other, and simultaneously tighten the nuts passed by. Further, this could lead to a more flexible and robust system, since the robot wouldn't require calibration and be dependent on the distance and angles to the frame. Nevertheless, this project used

hardware that limited the testing to object detection on images rather than video stream in real time. In addition to that, a real time system as described above should require hardware capable of video stream detection with at least several frames analysed per second.

CONCLUSIONS

The first step to achieve higher automation for Volvos' nut tightening station is seen as successful. The degree of recognition for both objects reach up to 95,7% with Faster R-CNN (see Table 3), which is acceptable if humans acts as supervisors for the missed nuts. This was achieved with only 322 images, consisting of 928 nuts and 568 inverted nuts (*Table 2*). Further work with this project includes more pictures and better hardware for the training and execution. Also starting with a pilot tests at Volvo Trucks regarding pictures from the real environment and tools to be applied for the tightening of the nuts. As discussed, the environment for the experiment is not fully comparable to the station at Volvo. The primary aim with the experiment was to understand the interactive process between tor robot application and nut detection which have been showed.

Other aspects such as safety between robot and operator needs to be further investigated before the implementation can be tested and implemented in industry. Furthermore, technical and semantic interoperability needs to be tested.

ACKNOWLEDGEMENT

The authors would like to acknowledge the Swedish agency VINNOVA for supporting the national testbed project in which this study has been carried out.

REFERENCES

ABADI, M., BARHAM, P., CHEN, J., CHEN, Z., DAVIS, A., DEAN, J., DEVIN, M., GHEMAWAT, S., IRVING, G., ISARD, M. & KUDLUR, M. 2016. Tensorflow: A system for large-scale machine learning. *12th Symposium on Operating Systems Design and Implementation.*

BAHRAMPOUR, S., RAMAKRISHNAN, N., SCHOTT, L. & SHAH, M. 2015. Comparative study of deep learning software frameworks. *arXiv preprint arXiv:1511.06435*.

BAUER, W., BENDER, M., BRAUN, M., RALLY, P. & SCHOLTZ, O. 2016. Lightweight robots in manual assembly - Best to start simply! *In:* IAO, F.-I. F. A. U. O. (ed.). Stuttgart.

BORTOLINI, M., FERRARI, E., GAMBERI, M., PILATI, F. & FACCIO, M. 2017. Assembly system design in the Industry 4.0 era: a general framework. *IFAC-PapersOnLine,* 50**,** 5700-5705.

BRADSKI, G. & KAEHLER, A. 2008. *Learning OpenCV*.

COHEN, Y., NASERALDIN, H., CHAUDHURI, A. & PILATI, F. 2019. Assembly systems in Industry 4.0 era: a road map to understand Assembly 4.0. *The International Journal of Advanced Manufacturing Technology,* 105**,** 4037-4054.

GIRSHICK, R. 2015. Fast R-CNN. *Proceedings of the IEEE international conference on computer vision***,** 1440-1448.

GIRSHICK, R., DONAHUE, J., DARRELL, T. & MALIK, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition***,** 580-587.

HUANG, J., RATHOD, V., SUN, C., ZHU, M., KORATTIKARA, A., FATHI, A., FISCHER, I., WOJNA, Z., SONG, Y., GUADARRAMA, S. & MURPHY, K. 2017. Speed/accuracy trade-offs for modern convolutional object detectors. *Proceedings of the IEEE conference on computer vision and pattern recognition***,** 7310-7311.

JIANG, H. & LEARNED-MILLER, E. 2017. Face detection with the faster R-CNN. *12th IEEE International Conference on Automatic Face & Gesture Recognition***,** 650-657.

KRIZHEVSKY, A., SUTSKEVER, I. & HINTON, G. E. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems***,** 1097-1105.

LIN, T. Y., MAIRE, M., BELONGIE, S., HAYS, J., PERONA, P., RAMANAN, D., DOLLÁR, P. & ZITNICK, C. L. 2014. Microsoft coco: Common objects in context. *European conference on computer vision***,** 740-755.

LIU, W., ANGUELOV, D., ERHAN, D., SZEGEDY, C., REED, S., FU, C.-Y. & BERG, A. C. SSD: Single Shot Multibox Detector. European conference on computer vision, 2016. Springer, 21-37.

REN, S., HE, K., GIRSHICK, R. & SUN, J. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems***,** 91-99.

RUSSELL, S. & NORVIG, P. 2013. *Artificial Intelligence: Pearson New International Edition: A Modern Approach*, Pearson Education M.U.A.

SAG, M. 2019. The New Legal Landscape for Text Mining and Machine Learning. *Available at SSRN*.

SHATNAWI, A., AL-BDOUR, G., AL-QURRAN, R. & AL-AYYOUB, M. 2018. A comparative study of open source deep learning frameworks. *9th International Conference on Information and Communication Systems***,** 72-77.

TZUTALIN 2015. LabelImg. Git code.

ZHAO, Z. Q., ZHENG, P., XU, S. T. & WU, X. 2019. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*.