Object Classification for Robotic Platforms

Samuel Brandenburg[1], Pedro Machado[1], Pranjali Shinde[2], João Filipe Ferreira[1,3], and T.M. McGinnity[1,4]

[1] Computational Neurosciences and Cognitive Robotics Group, Nottingham Trent University, Clifton Campus NG11 8NS, UK,
samuel.brandenburg2016@my.ntu.ac.uk, {pedro.baptistamachado, joao.ferreira martin.mcginnity}@ntu.ac.uk
[2] INESC TEC, R. Dr. Roberto Frias, Porto, Portugal
pranjali.shinde@inesctec.pt
[3] Inst. of Systems and Robotics, University of Coimbra, Polo II, Coimbra, Portugal,
jfilipe@isr.uc.pt
[4] Intelligent Systems Research Centre, Ulster University, Magee Campus, BT48 7JL, UK,
tm.mcginnity@ulster.ac.uk

Abstract. Computer vision has been revolutionised in recent years by increased research in convolutional neural networks (CNNs); however, many challenges remain to be addressed in order to ensure fast and accurate image processing when applying these techniques to robotics. These challenges consist of handling extreme changes in scale, illumination, noise, and viewing angles of a moving object. The project main contribution is to provide insight on how to properly train a convolutional neural network (CNN), a specific type of DNN, for object tracking in the context of industrial robotics. The proposed solution aims to use a combination of documented approaches to replicate a pick-and-place task with an industrial robot using computer vision feeding a YOLOv3 CNN. Experimental tests, designed to investigate the requirements of training the CNN in this context, were performed using a variety of objects that differed in shape and size in a controlled environment. The general focus was to detect the objects based on their shape; as a result, a suitable and secure grasp could be selected by the robot. The findings in this article reflect the challenges of training the CNN through brute force. It also highlights the different methods of annotating images and the ensuing results obtained after training the neural network.

Keywords: Object classification, training, YOLOv3, CNN, ROS

1 Introduction

Humans can detect target objects amongst distractors nearly instantly once they have caught their attention, even in the presence of extreme changes in scale, illumination, noise, and viewing angle of an object [1]. Adding to this, humans possess self and spatial awareness, abilities which are developed from an early age [2]. Spatial awareness is essential for humans to complete their daily activities,

which may range in difficulty from reaching for a cup of coffee to playing a high-performance sport. Many of these innate capabilities in humans have still to find convincing implementation in state-of-the-art robotics – see, for example, [3, 4, 5]. This research highlights the challenges and recent advancements in detecting objects using an artificial neural network (ANN).

Recent advancements in computer vision have been driven by research on deep neural networks (DNN); however, DNNs require an extensive amount of data and pre-processing. Poor data filtering can miscue results, therefore pre-processing is a crucial procedure for proper training. Therefore, standards such as the cross-industry standard process have been created to ensure to increase the rate of success [6]. This project's main contribution is to provide insight on how to properly train a convolutional neural network (CNN), a specific type of DNN, for object tracking in the context of industrial robotics. The paper structure is as follows: Chapter 2 presents the related work; the methodology is discussed in chapter 3; the results and the critical analysis is done in chapter 4; and the conclusions and future work are discussed in chapter 5.

2 Related work

The enactment of "pick and place" by a robot inherently presents many challenges, many of which at the level of artificial visual perception. Adding to these challenges, the use of ANNs to enhance visual processing involves time-consuming, complex training procedures. Furthermore, many factors, such as insufficient or low-quality training sets, can reduce their performance. For instance, an image fed into an ANN for processing may contain not only the desired object, but also other objects in the background. These distractors may degrade ANN performance if training is not properly performed. Another challenge in using ANNs is the detection of objects using a moving sensor (i.e. by resorting active perception, such as visual servoing). In this case, distortions such as motion blur may result in the detection algorithm failing to recognise the item.

CNN Neural Networks – Convolutional neural networks have proved overtime to be an effective algorithm for recognising visual patterns. The first model for a convolutional neural network was the leNet-5 created by Yann Lecun in 1998 [7]. This network was made up of two convolutional layers, two average pooling layers, two fully connected layers, and a softmax layer [7]. The leNet was not able to classify images, but it proved successful at classifying numbers. However, interest in CNNs was revived In 2012 when AlexNet was created which was much larger than its predecessor [8]. The network contained 5 Convolutional Layers and 3 Fully Connected Layers [8]. Training AlexNet to classify images took five to six days using two GTX 580 3GB GPUs [8]. This breakthrough has led to CNNs being the preferred solutions for image processing; however, real-time requirements make this version of the algorithm unsuitable for computer vision, in particular in robotics.

Faster Region Convolutional Neural Network Algorithm – Faster R-CNN is the current state-of-the-art optimisation method for CNN algorithms, designed to achieve real-time performance. Fast R-CNN and its predecessor used a technique called selective search [9]. Faster R-CNN, however, uses a technique called region proposal networks (RPN), which are much faster. RPN take an image as an input and output sets of anchor boxes of proposed objects; these objects are then associated with a score [10]. Faster R-CNN works by first creating a feature map from the input image. Subsequently, the RPN generates a set of proposed objects together with their score. Like its predecessor, Faster R-CNN still uses the ROI layer to make the proposed regions a fixed length. Once all the regions are of the same size, they are passed to the fully connected layer where SoftMax and linear regression are applied. The image is then classified and the algorithm outputs bounding boxes for the objects [11]. As a consequence of this optimisation process, Faster R-CNN decreases the time it takes to detect an image from 2 seconds to 0.2 seconds [10].

YOLO Algorithm – YOLO, which stands for "you only look once", was first introduced in 2015 by Joseph Redmon et al. [12] as alternative to R-CNN, which had complex pipelines that made it slow and hard to optimise. Unlike R-CNN, YOLO looks at the full image once. It uses a single CNN that predicts multiple bounding boxes and class probabilities for those boxes simultaneously [12]. YOLO predicts the score of an object using logistic regression for each bounding box. The proposed network was composed of 24 convolutional layers, four max-pooling layers[5], and two fully connected layers [12].

There are several benefits associated with YOLO. First of all, as mentioned previously, YOLO is faster compared to other detection methods because it avoids complex pipelines by framing it as a regression problem. YOLO encodes contextual information about classes and their appearance because it views the entire image for training and testing. Furthermore, it outperforms algorithms like fast R-CNN in making fewer background errors – Fast R-CNN mistakes background patches as objects because it does not have the full context of an image. Lastly, YOLO can generalise features it learns from objects, which can then be applied to new items.

At the time YOLO was introduced, it was the fastest general purpose object detection algorithm when compared to R-CNNs, Fast R-CNNs, and two versions of Faster R-CNNs [12]. The first was the VGG-16, which was ten mean Average Precision (mAP) higher but six times slower than YOLO [12]. The other was Zeiler-Fergus Faster R-CNN [12]. This model was much faster than the previous one, but it was not as fast and accurate as YOLO.

Additional developments have been made in the meanwhile to further improve YOLO's performance. In December 2016, YOLO version 2 was launched. This version differs from its predecessor because it uses a classification model

[5] Max pooling is a technique that extracts the most significant features from the convolutional layer.

called Darknet19 [13]. Darknet consists of 19 convolutional layers and 5 max-pooling layers [13]. The purpose of Darknet is to increase the speed and accuracy of classifications. Although the original YOLO algorithm outperformed previous object detectors, it under-performed in accuracy compared to Fast R-CNN by introducing a considerable amount of localisation errors [13]. Another aim was correcting the low recall YOLO produced compared to the region proposal-based method [13]. There were many ways YOLO version 2 addressed these problems. The first was to use batch normalisation, which significantly improved convergence while removing the need for other forms of regularisation. Another improvement made was to add high-resolution classifiers, which provided a 4% increase in mAP [13]. YOLO was susceptible to unstable gradients during training, therefore Anchor boxes reduced mAP slightly from 69.5 to 69.2, but the recall improved from 81% to 88% [13]. In other words, even if accuracy was slightly decreased, it increased the chances of detecting all the ground truth objects.

YOLO version 3 (YOLOv3) is the latest version that focuses on improvements on object classification [14]. Methods such as Single Shot Detection still outperform YOLOv3 in terms of accuracy; YOLOv3, however, executes three times faster for the same input [14]. Furthermore, YOLOv3 makes a considerable improvement in how well it can detect small objects [14]. Finally, this framework now uses Darknet 53, an enhancement to Darknet19, and increases the scale for feature extraction by increasing the number of convolutional layers from 19 to 53 [14].

3 Methodology

The object classification methodology was designed for robotic grasping applications. In the particular case of the authors, a Sawyer robotic arm[6] equipped with an AR10 hand[7] and state-of-the-art Biotac SP fingertips[8] was target. However, only the thumb, index finger, and middle finger were used with Biotec sensors installed at the end of them. The main task was to grasp objects using 3 of the 5 fingers in the hand using an adaptive grasp controller. It was decided that those items would be of a ball, a Pringles can, a sponge, and a water bottle (Figure 1). These items were chosen because each object varied in shape and size; Therefore, the robotic arm would need to change the grasping method for each of the items. For this to be accomplished, the first objective was to detect the objects. Once the item had been detected, the gripping algorithm could be appropriately adjusted, and the robotic arm and hand guided to perform the grip. The first step

[6] Retrieved from https://www.syntouchinc.com/en/sensor-technology/, last accessed 2019-06-20

[7] Retrieved from https://www.active8robots.com/robots/ar10-robotic-hand/, last accessed 2019-06-20

[8] Retrieved from https://www.active8robots.com/robots/ar10-robotic-hand/, last accessed 2019-06-20

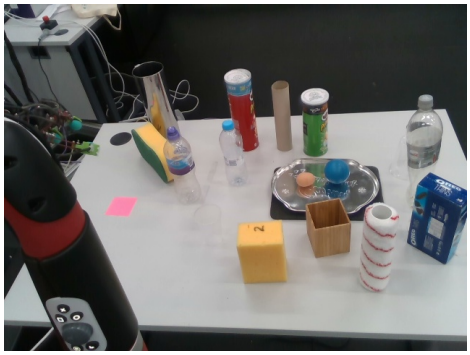

Fig. 1. Objects used for training the YOLOV3

to installing and configuring the YOLOv3[9] and adjust specific convolutional layers to match the number of classes, and filters to the number of classes designed. The selected classes where robotic hand, table marks, sphere, cuboid, prism and cylinder. These classes were extracted from a newly constructed database of images that was taken with an Intel D435 real sense camera. The robotic hand and the table marks are required for navigating the arm toward the target object, and the 3D shape of the objects is required for selecting the pre-grasp and pose of the objects. The aim of this paper in the object detection and therefore the details of the navigational algorithms, pre-grasp and pose estimations are not addressed in the paper. Annotation for training was performed by manually labelling each object in each image in the training data-set using a custom annotation tool developed by the authors[10]. This method of annotating classes makes up the majority of the pre-processing phase. The CNCR annotation tool opens a window browser (see Figure 2) and lets the user import images and classes. The user can then label objects by defining a bounding box. The box parameters are then recorded in a text file. Each row contains five columns: the first indicates the index for the different classes, the second and third columns specify the location of the object in an X and Y grid, respectively, and the last column stores the height and width of the object. Training YOLOV3 can start once a dataset of images and their annotations have been formed; however, preventive measures must be put in place in order to prevent underfitting and overfitting [15]. Overfitting, in particular, prevents the model from generalising to unknown data and thus leading to poor performance. A standard way to prevent overfitting and underfitting is to construct a validation set, i.e. by defining distinct training and test subsets from the images collected in the training phase.

[9] Retrieved from https://medium.com/@manivannan_data/how-to-train-YOLOv3-to-detect-custom-objects-ccbcafeb13d2, last accessed on 25/04/2019

[10] Available online, https://gitlab.com/CNCR-NTU/CNCR_annotation_tool, last accessed on the 15/06/2019

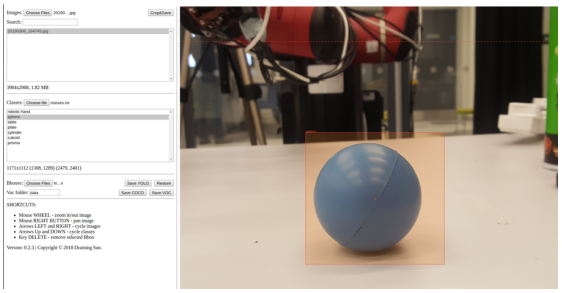

Fig. 2. CNCR annotation tool

The next step of the process was training the model. To train the YOLOV3 model the names of the classes, the configuration file, the data file, and the YOLOv3 architecture were needed, taking approximately two days to conclude the procedure. The YOLOv3 creates weights in increments of 10k steps during this period; however, adjustments were made to have weights created every 2k steps (this value was an empirical value obtained experimentally). The trained weights were then tested on the testing data and the weights that produced better results were selected.

4 Results

During the pre-processing phase, it became apparent that many factors influenced object detection performance when annotating. The majority of this section will highlight the methods used while annotating. Furthermore, this section will convey their results. Over ten training cycles have been conducted during the implementation phase. The overall goal of accomplishing computer vision was to detect and generalise similar objects.

A significant part needed for visual servoing was detecting the parameters of the table. The table was marked on each corner by a strand of red tape. The first objective was producing a stable object detection model. Therefore, detecting the medium size ball was the first goal. Key points that were learned during this objective was the training dataset needed to contain over 200 images. Furthermore, the performance of the model varied depending on the weights used. There were performance issues with each of the weights used. However, 10k weights gave the model more accuracy and stabilised detection. Below displays results of the first test 1.

It took two training sets before obtaining positive results. However, it was found that the ball would not be detected if it was moved too far to the right or left on the table. Another downside was the size of the bounding box on the item. The width of the bounding box was similar to that of the annotations. A smaller box would be ideal. As a result, the next set of annotations contained

Table 1. Accuracy Readings from .5 - 1.0 using different weights

Item	10k weight	12k weight	16k weight	20k weight	30k weight
Ball	.98	.98	.99	.95	.91

small boxes covering the item versus one larger box. It was seen that smaller boxes could reduce the amount of background noise (see

Table 2. Detection with the Background Changing

10k weights percentage	table background(known)	mat background(known)	unknown background
sphere	1.00	.95	N/A
cuboid	.86	.82	N/A
Cylinder	.96	.98	N/A

Additionally, smaller annotation boxes would decrease the size of the bounding box. For instance, if annotation were being done for an object in being grasped by the robotic hand, The boxes outlining the hand would shrink, but increase in number to only contain the hand. This method brought abysmal results because the model became overfitted. The model projected bounding boxes over the whole image. In order to correct this problem changes needed to be made to the labelled files. It was discovered that an object should contain no more than two annotation boxes around it. Creating a limit helped prevent overfitting. It also made the ball detectable again. The previous issue still resided, but it lost the object less than before[11].

The next step was detecting multiple objects at the same time. The can and the ball needed to be detected. Achieving this objective was relatively easy. This was accomplished by increasing the number of images in the training set. It was also important that there was as less background noise as possible. However, more issues appeared when all the items needed to be annotated. The difference was the positioning of the items in the image. Merging annotations occurred very often because the items were too close. Another method would be to highlight the sections that would not overlap the images. The issues with that approach are that it does not get the full object. Therefore, it may degrade results on that one item if partial annotations are continuously being done.

At this point, the detection algorithm was performing well on the ball, the Pringles can, and the sponge with a confidence score over 0.7. However, it is important to note that some items may consist of a combination of different shapes. Therefore, the classifying process for such an item must contain different

[11] Available online https://www.youtube.com/watch?v=vdDqMtdyUYU, last accessed on 25/04/2019

classes in an item. This was the case for the bottle. When annotating an item like a bottle, it was important to match each shape with the class that fitted best. Breaking the item into different identifiable shapes assist the process of picking the item up. However, the first attempt at annotating multiple shapes had different classes merging into another. Thus when testing this model using 10k weights, no items were being classified. It was identified that the overlap of labelling boxes substantially increases the number of false positives. Thus the solution was the avoidance of overlapping annotation boxes. This strategy worked, and the detection algorithm began working for all the objects. The next test was to evaluate the generalisation of spheres. This test involved using more than ten spheres with different sizes and colours. However, only two spheres were detected. The training sphere was detected, and another sphere similar in colour but different in size. After receiving abysmal results, further tests were conducted to understand why the generalisation of spheres failed. This test involved using a lid that had a sphere shape with a light blue tint similar to the other two spheres. When the item laid flat, it was not detected, but when it was standing upright, it was detected as a sphere. These results demonstrated that (i) the algorithm recognised objects similar in shape and colour to that it was trained on and (ii) that increasing the number of different samples was crucial to enable the CNN to start generalising patterns to unfamiliar objects. The next step was to create the environment so that visual servoing could be used since all the objects were being detected. This process involved mapping out the boundaries of the table by adding markers to each corner. After training the model to recognise these tags, It was found that the two nearest markers were detected, but not consistently. Furthermore, the furthest two markers failed to be detected. Despite, adding more annotations outlining all the table markers, the problem was not fixed. At the same time, these annotations included the placement of the robotic arm at different positions of the table. It was later discovered that the lighting of the room and shadows created by the arm was the reason behind the inconsistent detection.

After all of the objects were successfully detected, training the CNN to recognise the robotic arm and table marks was next. This is needed for moving the arm from a starting point to the object pick-up place. The process involved mapping out the boundaries of the table by adding markers to each corner. After training the model to recognise these tags, It was found that the two nearest markers were detected, but not consistently. Furthermore, the furthest two markers failed to be detected. Despite, adding more annotations detection of all the table markers was not fixed. At the same time, these annotations included moving the robotic arm across the table when conducting test on its' range of motion. It was later discovered that the lighting of the room and shadows were the reason certain items stopped being detected. This was especially true when the robotic arm was moving since its' arm made a shadow. As mentioned before in the literature review illumination changes make perfect object detection difficult. One method that helped alleviate these issues were annotating objects

while in different illuminations. This method proved it had the potential to work; however, annotating every angle and circumstance was too tedious of a task.

Other interesting facts learned while testing the customised model of YOLOv3 was objects needed to be moved around. If an object stays in the same position for all the images, it will not be detected once it is moved. This was discovered while testing on a set of spheres. The initial training set contained images where the position of the spheres did not move. It was found that annotation files could be automated by copying the parameters in an annotated file. Those parameters were then pasted into new text files that were attached for the remaining training files. However, this method proved unsuccessful. Another aspect that affects the model's performance is when changes are made to the background. This theory was tested by placing a sphere on a black mat. The sphere was detected while on the table, but it was not when moved to the mat. Furthermore, the bounding box for an object would disappear if anything went near the item. This was another challenge to resolve. The robotic requires consistent detection while reaching for the item. If the item is no longer detected than the robotic arm may fail to grasp the item correctly. One way of resolving this issue was including the robot hand near the item and around the item.

The current model was tested using a range of weights to determine which one performed the best. From a visual perspective, 10k weights detect more items and it also was more consistent. Furthermore, it also had the highest accuracy percentage from all the test which was 85%. 30k weights gave the model its' second highest percentage at 83%. Nevertheless, it still struggles to detect the table marker furthermost away on the left-hand side. It also does not detect the robot hand consistently. Whereas, the other weights in the table detected the robot hand consistently. However, their accuracy percentage was also slightly less accurate with the lowest being 20k weights at 77%. Nevertheless, the current model has shown that it is capable of detecting objects it has seen. Another positive

Table 3. Illumination impact

labels	table	cylinders	prism	sphere	mislabels
With light	4/4	5/5	3/4	1/1	1
Reduced lighting	3/4	2/5	1/4	0/1	0

result is that the current mode can detect up to 92% of the items. It also can detect the hand while its moving better than it initially could. However, some of the issues mentioned in the related worked section. When the illumination changes, the detection rate drops to 46%. This refers to inconsistent detection and tracking. This experiment exhibited many of the challenges written about in the related work section. In addition to the challenges presented from changes illumination, detection performance drops when viewing objects from different angles. Video footage was taken with multiple cameras from different positions. The main camera was able to detect the items from up above, but only two of

the three side cameras detected anything [12]. This project also revealed how important annotating was and why there is a need for large datasets. This is to say that the large dataset tries to encompass an extensive range of scenarios. If the dataset contains these, the model can then begin to generalise this information to unknown objects.

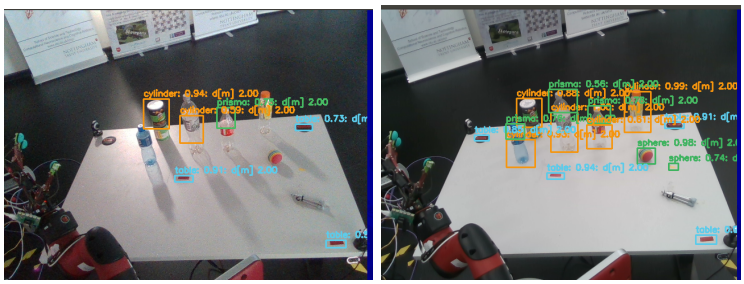

Fig. 3. Object detection with a) reduced lighting and b) normal light

Figure 3 illustrate how much performance is reduced from illumination changes. In the image above it can be seen that most of the objects are detected. However in the second image the performance drops considerably. This is just one example of how variations can impact classification.

5 Conclusions and Future Work

This project proposed a combination of techniques to replicate a pick and place task using a robotic arm. YOLOV3, a state of the art detection algorithm, was the central algorithm used for this work. More specifically, a customised version of YOLOV3 was used, trained on over 2,000 training images. The assessment of this model showed that it could detect up to 92% of the trained objects. However, real-time results are inconsistent; therefore, this model can sometimes achieve a higher percentage rating. Furthermore, detection in different angles along with generalising objects classification was performed. Despitee these achievements, the proposed methods for this project heavily depended on stable object detection and tracking which was not achieved. This was especially needed to implement the intended visual servoing practices. Consistent object detection of the table was to be used to guide the robotic arm to find items using triangulation. Unfortunately, the act of moving robotic arm created many fluctuations that affected the appearance of the object. As a result, detecting and tracking those objects was very inconsistent. It was also discovered that the weight for

[12] Available online https://www.youtube.com/watch?v=IzN3kp7eAuY, last accessed on 25/04/2019

the model made a significant impact on the model performance. Four tests were conducted to discover the optimum amount of weights were needed. This assessment highlighted that the model showed the best results at 10k weights. This change to 10k would increase object detection accuracy to 85%. However, If the number of weights was to decrease the model performance would reduce because it is underfitted. In contrast, if the weights were too high, the model would be overfitted. Both of these issues led to poor performance.

Therefore, at first glance we conclude that, although over 2,000 images were annotated, stable object detection will require many more training samples. However, this is far too time-consuming; therefore, other methods to achieve this objective should be explored in future work. However, the most significant drawback to the proposed method was to require consistent detection of the corners for trajectory planning. A more straightforward solution would have been to collect the coordinate of the location from one triangulation. This would have solved the issue of inconsistent detection. Another potential solution could have been to use object detection to identify if an object is on a designated spot. The robotic arm could then react by reaching for the item. The implementation for this process would be hard coded thus simplifying many of the issues that are associated with visual servoing. Yet another alternative would be to implement one-shot learning on a pre-trained neural network. This could be done using the weights provided by YOLOv3 which would alleviate the tedious task of manual pre-processing thousands of images. Additionally, it may be better to choose a different method of visual servoing. Finally, off-the-shelf open source software such as VISP exists that allows a user to form a spatial map, allowing for even potential improvements to explore in future work.

References

[1] Mehran Yazdi and Thierry Bouwmans. "New trends on moving object detection in video images captured by a moving camera: A survey". In: *Computer Science Review* 28 (2018), pp. 157–177. ISSN: 15740137. DOI: 10.1016/j.cosrev.2018.03.001. URL: https://doi.org/10.1016/j.cosrev.2018.03.001.

[2] Rabia Jafri, Asmaa Mohammed Aljuhani, and Syed Abid Ali. "A Tangible Interface-based Application for Teaching Tactual Shape Perception and Spatial Awareness Sub-Concepts to Visually Impaired Children". In: *Procedia Manufacturing* 3.Ahfe (2015), pp. 5562–5569. ISSN: 23519789. DOI: 10.1016/j.promfg.2015.07.734. URL: http://dx.doi.org/10.1016/j.promfg.2015.07.734.

[3] Charu C. Aggarwal. "Convolutional Neural Networks". In: *Neural Networks and Deep Learning*. Cham: Springer International Publishing, 2018, pp. 315–371. DOI: 10.1007/978-3-319-94463-0_8. URL: http://link.springer.com/10.1007/978-3-319-94463-0%7B%5C_%7D8.

[4] Danica Kragic et al. "Interactive, Collaborative Robots: Challenges and Opportunities". In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)*. 2018, pp. 18–25.

[5] João Filipe Ferreira and Jorge Dias. "Attentional Mechanisms for Socially Interactive Robots – A Survey". In: *IEEE Transactions on Autonomous Mental Development* 6.2 (June 2014), pp. 110–123. DOI: 10.1109/TAMD.2014.2303072.

[6] Rudiger Wirth and J. Hipp. "CRISP-DM: Towards a standard process model for data mining". In: *Proceedings of the Fourth International Conference on the Practical Application of Knowledge Discovery and Data Mining* 24959 (2000), pp. 29–39. ISSN: 1092-6208. DOI: 10.1.1.198.5133.

[7] C. C.Jay Kuo. "Understanding convolutional neural networks with a mathematical model". In: *Journal of Visual Communication and Image Representation* 41 (2016), pp. 406–413. ISSN: 10959076. DOI: 10.1016/j.jvcir.2016.11.003. arXiv: arXiv:1609.04112v2.

[8] Siddharth Das. *CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more ....* 2017. URL: https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5 (visited on 04/24/2019).

[9] Pulkit Sharma. *A Step-by-Step Introduction to the Basic Object Detection Algorithms (Part 1)*. 2018. URL: https://www.analyticsvidhya.com/blog/2018/10/a-step-by-step-introduction-to-the-basic-object-detection-algorithms-part-1/ (visited on 03/15/2019).

[10] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2017), pp. 1137–1149. ISSN: 01628828. DOI: 10.1109/TPAMI.2016.2577031. arXiv: 1506.01497.

[11] Pulkit Sharma. *A Step-by-Step Introduction to the Basic Object Detection Algorithms (Part 1)*. 2018. URL: https://www.analyticsvidhya.com/blog/2018/10/a-step-by-step-introduction-to-the-basic-object-detection-algorithms-part-1/ (visited on 03/24/2019).

[12] Joseph Redmon et al. In: (2015). ISSN: 01689002. DOI: 10.1109/CVPR.2016.91. arXiv: 1506.02640. URL: http://arxiv.org/abs/1506.02640.

[13] Joseph Redmon and Ali Farhadi. "YOLO9000: Better, faster, stronger". In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* 2017-Janua (2017), pp. 6517–6525. DOI: 10.1109/CVPR.2017.690. arXiv: arXiv:1612.08242v1.

[14] Joseph Redmon and Ali Farhadi. "YOLOv3: An Incremental Improvement". In: (2018). ISSN: 0146-4833. DOI: 10.1109/CVPR.2017.690. arXiv: 1804.02767. URL: http://arxiv.org/abs/1804.02767.

[15] Jason Brownlee. *Overfitting and Underfitting With Machine Learning Algorithms*. 2016. URL: https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/ (visited on 04/22/2019).