Detection of Objects in a Tunnel Using Deep Learning

Ritika Agarwal, M.S.

Nima Zahadat, Ph.D.

The George Washington University

ABSTRACT

LiDAR scans capture minute details of the boundaries and the surface of a structure [3]. It generates huge amounts of point cloud information which when processed and analyzed can show interesting features and generate 3D digital models of the structure. A large amount of point cloud data was generated of a LiDAR scan taken of a tunnel.

The overall purpose of this project was to identify the locations of different objects in this tunnel using the point cloud data. The point cloud data was converted into images which were used for object detection. I used the state of the art YOLOv2 algorithm to carry out object detection to identify the objects of interest. My project shows that the algorithm can detect most objects of interest, with confidence going up to 0.80, however it does not necessarily identify all the instances of the objects. Some objects are identified with high confidence, but the generated bounding boxes are not accurate. However, it can still save hours of manual work by using deep learning to locate the objects.

INTRODUCTION

A LiDAR scan was used in a tunnel- Tunnel Mont Royale located in Montreal, Canada of approximately 500metres. The A large amount of point cloud data was generated.

Lidar is a surveying method that measures distance to a target by illuminating the target with pulsed laser light and measuring the reflected pulses with a sensor [4].

Point clouds are a collection of points that represent a 3D shape or feature. Each point has its own set of X, Y and Z coordinates and in some cases additional attributes. We can think about a point cloud as a collection of multiple points. When many points are brought together they start to show some interesting qualities of the feature that they represent [5]. From all this data collected we wanted to identify the different materials and objects in the tunnel.

In order to identify the different objects in the tunnel, the point cloud information was processed and converted into images. It would take hours of tedious manual work of tagging the images to identify all the objects and their locations.

In order to avoid this, we decided to use Deep learning to tag the different objects in the images.

Our process was simple, we converted the point cloud data to images, annotated the images by tagging them which were then ready to be used for object detection. I had the liberty of using whichever algorithm we required but we had time and hardware constraints.

I used the state of the art YOLO algorithm which is implemented by Thrieu in Tensorflow, called the Darkflow.

LITERATURE REVIEW

Object detection is the identification of an object in the image along with its localization and classification. It has wide spread applications and is a critical component for vision based software systems [6] . Right now, there are multiple methods of object detection in images including Faster RCNN, SSD, YOLO.

R-CNN and its variants use region proposals instead of sliding windows to find objects in images. The pipeline for RCNN is complicated and each stage must be tuned independently. It uses selective search to generate the potential bounding boxes and a CNN to extract features. The system is very slow taking more than 40 seconds per image at test time.

Fast and Faster R-CNN focus on speeding up the R-CNN framework by sharing computation and using neural networks to propose regions instead of Selective Search While they offer speed and accuracy improvements over R-CNN, both still fall short of real-time performance.

YOLO has more advanced application in practice. YOLO is a unified object detection model. It's simple to construct and can be trained directly on full images. YOLO's better generalizing representation of object than other models making it ideal for applications that rely on fast, robust object detection. These preeminent and precious advantages make it worthy of being strongly recommended and popularized. The availability of suitable training data could be the vital part in the learning process, to achieve ideal results. [7]

RESEARCH METHODOLOGY

The overall pipeline for the work included the following steps:

1. Transforming the data to images
2. Choosing an algorithm
3. Labeling the data – I used LabelImg go over every image and draw bounding boxes for all objects of interest and tag them with a class and save the annotations.
4. Setup the environment with all the libraries required and downloaded the algorithm – Yolo (Darkflow)
5. Made changes in the code to fit my data.
6. Train the images and then test my data.
7. Visualize my results.

Packages: Tensorflow-gpu, NumPy, matplotlib, Cython

DATA

A LiDAR scan was carried out in the Tunnel Mont Royale located in Montreal, Canada.

The data collected from the LiDAR scan had more than 400 million points. The information collected has the x coordinate (Eastings), y coordinate (Northings), z co-ordinate (Elevation), intensity (alpha channel) and RGB values. In order to query this information and convert them into images, the points were converted to binary, compressed and then stored. The points were divided into clusters which had information including the center of the cluster and Id. This could then be geolocated as a single point for spatial queries. All points within a given range and interval were use and rest of the points were culled to define an image.

After the point cloud information was converted into images, we had about 3750 images. Each image showed the cross-section of a tunnel flattened out. Each image typically showed features such as the rails, signs, wall stains, cracks, gravel.

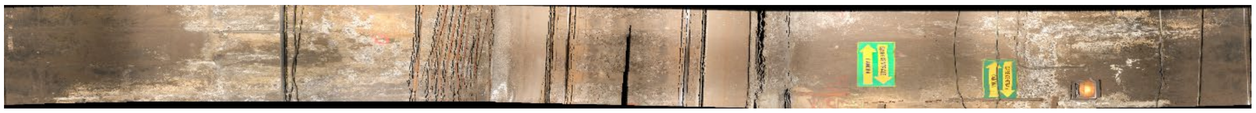


Figure1. Image showing cross-section of the tunnel used in training

DATA ANALYSIS

The first challenge was to decipher the most appropriate algorithm to use for these images to detect the different objects.

There are multiple algorithms for object detection such as Faster-RCNN, YOLO, SSDs. YOLOv2 provides state-of-the-art the best tradeoff between real-time speed and excellent accuracy for object detection than other detection systems across a variety of detection datasets.[7]

I needed an efficient method due to time constraints and hence I chose to go further with the Yolo algorithm. There are three implementations of YOLO

- DarkNet
- DarkFlow
- AlexeyAB

I chose the Darkflow implementation for my project. This works with the TensorFlow library and is faster than the rest. It is ready to use and needs to be downloaded and installed to start detecting 'common' objects.

The next step was to label the data. The data did not have any annotations or labels. I manually labelled images using the 'LabelImg' software. This generated a xml file for each image saved with same file name as the image.

The xml files have the following important information:

1. File name
2. Co-ordinates of the bounding box
3. Class to which the bounding box belongs to.

I paid attention to the following factors while annotating the images

1. Consistency, so that the annotation of the images is consistent, in terms of the definition of the classes, how bounding boxes are placed
2. Accuracy, so that there are as few annotation errors as possible.
3. Exhaustive, so that all object instances are labelled [14].

The annotated dataset consisted of:

Dataset size: 203

- pipes: 1203
- sign: 105
- light: 24
- tracks: 230
- green cables: 17

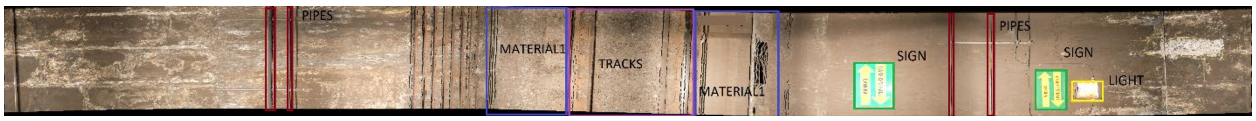


Figure2. Image showing objects of interest

Understanding YOLO

Object detection is the task of identifying objects in an image and drawing bounding boxes around them, i.e. localizing them. [8] When performing standard image classification, given an input image, we present it to our neural network, and we obtain a single class label and perhaps a probability associated with the class label as well.

This class label is meant to characterize the contents of the entire image, or at least the most dominant, visible contents of the image.

Object detection, regardless of whether performed via deep learning or other computer vision techniques, builds on image classification and seeks to localize exactly where in the image each object appears.

When performing object detection, given an input image, we wish to obtain:

- A list of bounding boxes, or the (x, y)-coordinates for each object in an image
- The class label associated with each bounding box
- The probability/confidence score associated with each bounding box and class label [9]

As per the above description out task is to predict the class of an object, the bounding box location co-ordinates and the confidence score.

Each bounding box is described using the parameters:

1. X, Y coordinates represent the center of the box relative to the grid cell
2. Width and Height relative to the whole image.

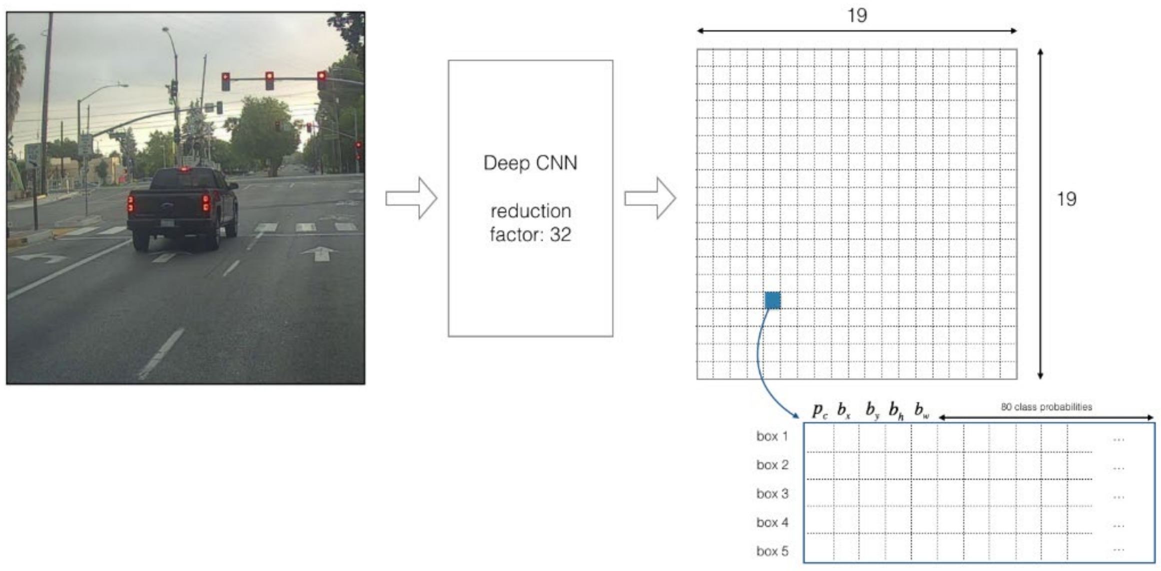


Figure3. Image explaining Yolo algorithm input and output

We can understand the above picture further in the steps below.

1. The image is divided into grid cells (19 by 19)
2. If the center of the object falls into a grid cells, that grid cell is responsible for detecting that object.
3. Each grid cell predicts B bounding boxes (5) and confidence scores for those boxes. Confidence score is the probability that box contains an object and how accurate is the boundary box.
4. The confidence score is given by Probability (Object) ∗ IOU of ground truth predicted boxes.

   If an object is not present the confidence score would be expected to be 0 otherwise it is equal to the intersection over union between the predicted and the ground truth.
5. Each bounding box also predicts C conditional probabilities. These probabilities are restricted to the grid cells containing and object, regardless of the number of objects.
6. At test time, each conditional class probability is multiplied by the individual box confidence prediction. Hence, we get class specific confidence scores for each box. It gives us a prediction tensor of dimension (19*19*(B*(5+Number of classes)).
7. Non-max suppression is used to remove objects which have high intersection over union and low object probability.

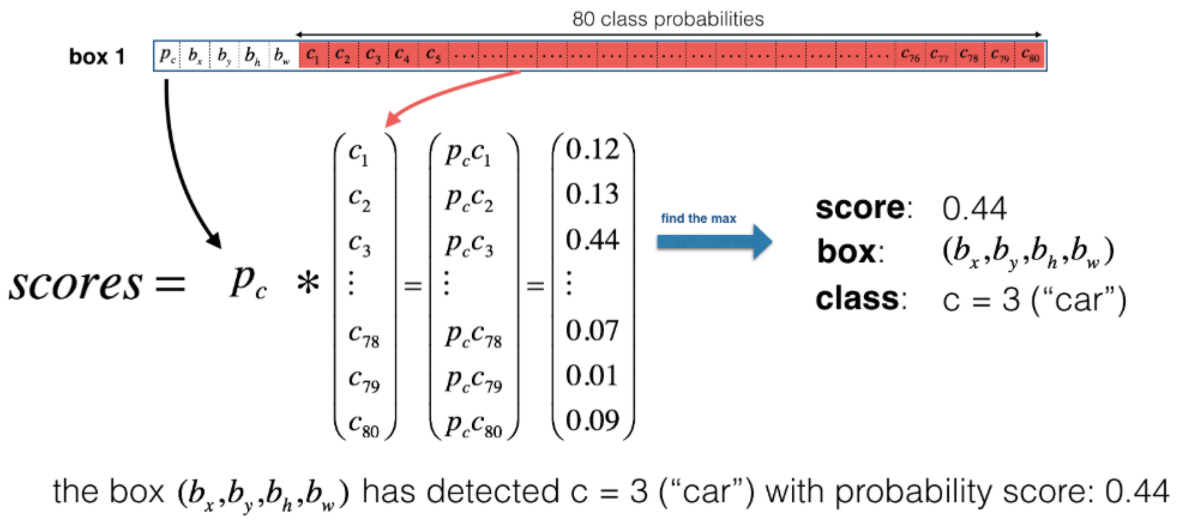

80 class probabilities
box 1
p_c b_x b_y b_h b_w c_1 c_2 c_3 c_4 c_5 … c_{76} c_{77} c_{78} c_{79} c_{80}
$scores = p_c * (c_1, c_2, c_3, \vdots, c_{78}, c_{79}, c_{80}) = (p_c c_1, p_c c_2, p_c c_3, \vdots, p_c c_{78}, p_c c_{79}, p_c c_{80}) = (0.12, 0.13, 0.44, \vdots, 0.07, 0.01, 0.09)$
find the max
score: 0.44
box: (b_x, b_y, b_h, b_w)
class: c = 3 ("car")
the box (b_x, b_y, b_h, b_w) has detected c = 3 ("car") with probability score: 0.44

Figure4. Image explaining how Yolo algorithm calculates the final score for each box

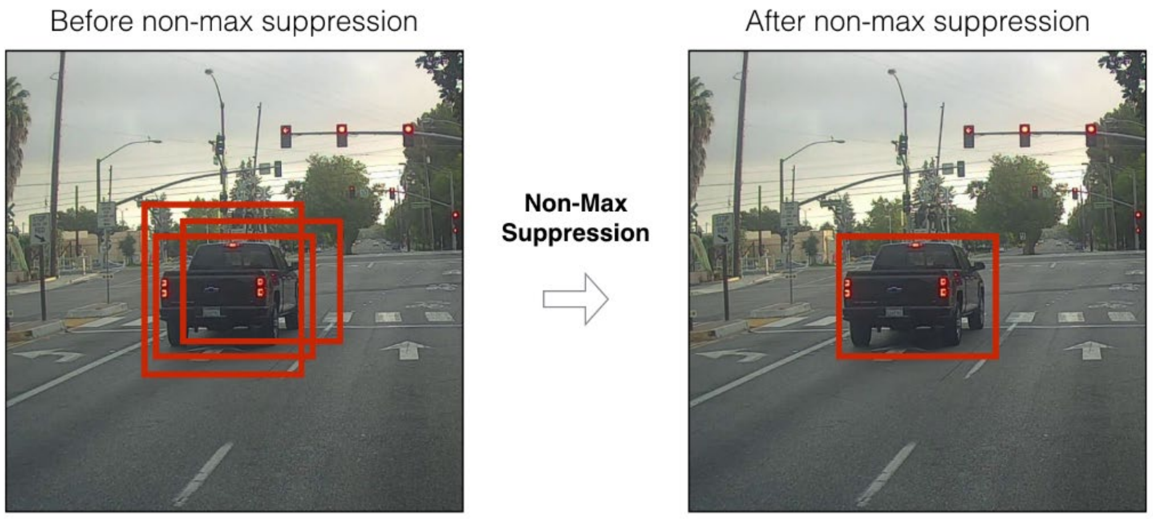


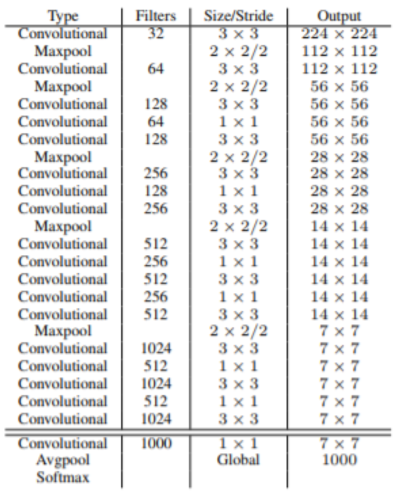
Figure4. Image showing the concept of Non-Max suppression

Using our system, you only look once (YOLO) at an image to predict what objects are present and where they are.

Network Architecture

The model is implemented as a convolutional neural network. The architecture is inspired by GoogleNet.

It has 23 convolution layers followed by three 3 × 3 convolutional layers with 1024 filters each followed by a final 1 × 1 convolutional layer with the number of outputs we need for detection.



Type	Filters	Size/Stride	Output
Convolutional	32	3 × 3	224 × 224
Maxpool		2 × 2/2	112 × 112
Convolutional	64	3 × 3	112 × 112
Maxpool		2 × 2/2	56 × 56
Convolutional	128	3 × 3	56 × 56
Convolutional	64	1 × 1	56 × 56
Convolutional	128	3 × 3	56 × 56
Maxpool		2 × 2/2	28 × 28
Convolutional	256	3 × 3	28 × 28
Convolutional	128	1 × 1	28 × 28
Convolutional	256	3 × 3	28 × 28
Maxpool		2 × 2/2	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Convolutional	256	1 × 1	14 × 14
Convolutional	512	3 × 3	14 × 14
Maxpool		2 × 2/2	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	512	1 × 1	7 × 7
Convolutional	1024	3 × 3	7 × 7
Convolutional	1000	1 × 1	7 × 7
Avgpool		Global	1000
Softmax			

Figure4. Image showing the network architecture

Experiments

I used an AWS ec2 instance -g3s.xalrge (8GB ram) setup with a DLAMI to train my model.

I used pretrained weights which are available on the official Yolo website. These weights can already detect some features. It reduces the training time required to start detecting objects in my data.

I tried the following optimizers to train my model:

1. RMS PROP
2. ADADELTA
3. ADAGRAD
4. ADAM
5. SGD

I compared the performance of the 5 optimizers and then chose one to complete my training process. Due to large amount of time taken for training, I checked this over 25 iterations.

Training statistics:

Learning rate: 1e-05

Batch size: 6

Epoch number: 25

OPTIMIZER	INTIAL LOSS	LOSS
RMSPROP	233.409	9.174627924582401
ADADELTA	244.28654479980472	227.87544445556
ADAGRAD	233.5775146484375	163.78819392739558
ADAM	258.9695739746094	13.979686690242735
SGD	228.0630340576172	23.744057280813948

As RMS Prop had the lowest loss over 25 iterations I used it for training over all the epochs.

On the final dataset the loss started converging after 2500 iterations.

Final Training statistics:

Learning rate: 1e-05

Batch size: 6

Epoch number: 3027 epochs

Time taken: 50 hours

Optimizer: RMS Prop

KEY FINDINGS

1. The loss reduced to about 4 and did not reduce further. I did not test the accuracy because I did not have labeled data for testing images.
2. However, the detection confidence for the bounding boxes was greater than 0.8. Some objects are identified with high confidence, but the generated bounding boxes are not accurate.
3. Not all instances of the objects are identified with high confidence.
4. The annotations of the images need to be done more carefully and consistently. Increasing the number of images in training will also help the network predict with higher confidence and reduce the loss.
5. There are shadows of some objects such as pipes and conduits. Sometimes the algorithm also tags these shadows as objects. Lights, and signs were predicted and located accurately, probably due to their distinctive features even if the number of instances of these objects were low in the training data.

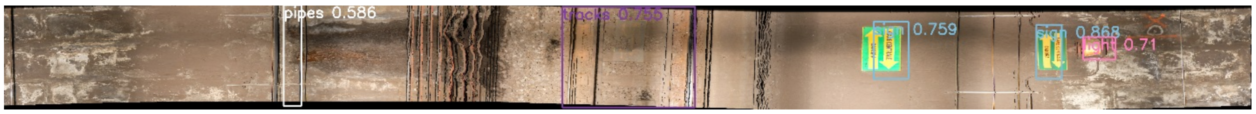


Figure5. Image showing the network prediction

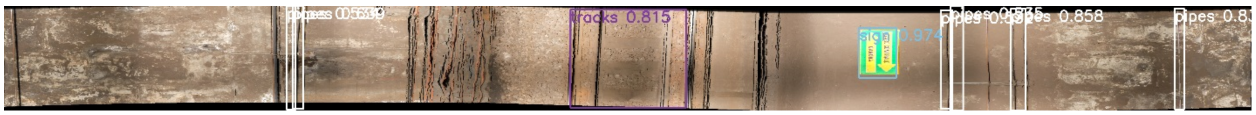


Figure6. Image showing the network prediction

CONCLUSION

Yolo detected the objects with high confidence however some of the objects were misclassified and the bounding boxes were not accurate.

This could be possible due to errors in annotating the images. If this algorithm must be used efficiently, annotation need to be carried out for many more images and with more precision.

Inspite of the errors, using deep learning to tag the images and locate objects will save large amounts of time and manual effort. However, you need a powerful GPU and resources and to get accurate and successful predictions. It is a computationally expensive task.

It is worth trying different techniques, such as image segmentation for this task.

FUTURE WORK

The objects identified in the images will be plotted back on to a 3D digital model of the tunnel. The algorithm outputs a json file with the coordinates of every bounding box in an image, along with the class name. These coordinates can then be used to identify the objects on the actual 3D model of the image.

This saves hours of manual work of tediously identify every object in the tunnel.

I can also try other algorithms to perform this task and compare the predictions.

Another, task is to implement Mask RCNN to identify different materials in the image. Mask RCNN labels every pixel in an image with a class. I have started this work and below is an example of the same.

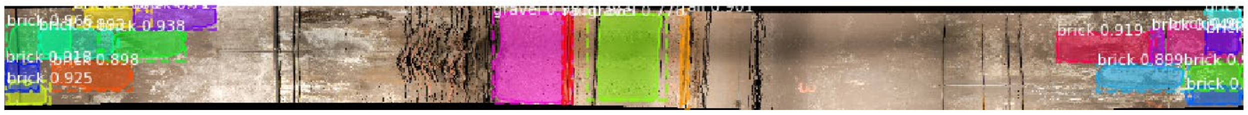


Figure7. Image showing the output generated using Mask RCNN which locates objects such as bricks , gravel and rails.

BIOGRAPHY

Ritika Agarwal is a graduate student in the Data Science Program at The George Washington University. She is working as an intern Data Scientist at AECOM in New York City. She has a background in Civil Engineering and her interests lie in integrating the two domains of Data Science and Civil Engineering.

Dr. Nima Zahadat is a professor of data science, information systems security, and digital forensics. His research focus is on studying the Internet of Things, data mining, information visualization, mobile security, security policy management, and memory forensics. He has been teaching since 2001 and has developed and taught over 100 topics. Dr. Zahadat has also been consultant with the federal government agencies, the US Air Force, Navy, Marines, and the Coast Guard. He enjoys teaching, biking, reading, and writing.

REFERENCES

[1] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. (2015). eprint arXiv:1506.02640.

[2] Joseph Redmon, Ali Farhadi. YOLO9000: Better, Faster, Stronger. (2016). arXiv:1612.08242

[3] Lidar-UK Websites. Taken from http://www.lidar-uk.com/usage-of-lidar/

[4] Wikipedia contributors. (2019, April 25). Lidar. In Wikipedia, The Free Encyclopedia. Retrieved 04:14, April 28, 2019, from https://en.wikipedia.org/w/index.php?title=Lidar&oldid=89411678

[5] robynr. (2015) . Taken from https://knowledge.safe.com/articles/257/what-is-a-point-cloud-what-is-lidar.html

[6] Karanbir Singh Chahal, Kuntal Dey. (2018). A Survey of Modern Object Detection Literature using Deep Learning. arXiv:1808.07256 .

[7] Juan Du. (2018). Understanding of Object Detection Based on CNN Family and YOLO. J. Phys.: Conf. Ser. 1004 012029

[8] Varun Agrawal (2017). Improving Real-Time Object Detection with YOLO. Taken from https://blog.statsbot.co/real-time-object-detection-yolo-cd348527b9b7

[9] Adrian Rosebrock. (2018). A gentle guide to deep learning object detection. Taken from https://www.pyimagesearch.com/2018/05/14/a-gentle-guide-to-deep-learning-object-detection/

[10] Deeplearning.ai. Convolutional Neural Network. From https://www.coursera.org/lecture/convolutional-neural-networks/yolo-algorithm-fF3O0

[11] Park Chansung. (2018) YOLOv2 to detect your own objects using Darkflow. Taken from https://towardsdatascience.com/yolov2-to-detect-your-own-objects-soccer-ball-using-darkflow-a4f98d5ce5bf

[12] Thtrieu. (2017) Taken from https://github.com/thtrieu/darkflow

[13] Tzutalin. LabelImg. Git code (2015). https://github.com/tzutalin/labelImg

[14] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, Andrew Zisserman. 2010. The Pascal Visual Object Classes (VOC) Challenge. International Journal of Computer Vision

APPENDIX A

Variable	Description
YOLO	You Only Look Once Algorithm
IOU	Intersection over Union
C	Number of classes
DLAMI	Deep Learning AMI
B	Number of bounding boxes