

# Processing technical drawings using deep learning

Norberto Carlos Saraiva Martins

Master's degree in Computer Science

Computer Science Department

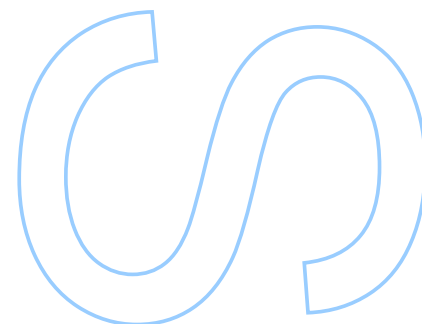
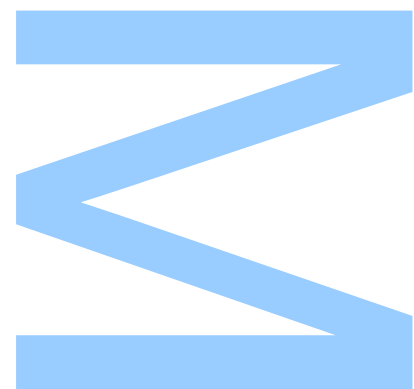
2019

## Supervisor

Miguel Tavares Coimbra, Phd, Faculty of Sciences, University of Porto

## Co-Supervisor

Samuel Moura, Antares, Lda





**U.** PORTO

**FC** FACULDADE DE CIÊNCIAS  
UNIVERSIDADE DO PORTO

All corrections determined by the jury,  
and only those, were incorporated.

The President of the Jury,

Porto, \_\_\_\_ / \_\_\_\_ / \_\_\_\_

**W**

**S**

**Q**



# Acknowledgements

In this work, many were those who supported me and without this support does not would have made it this far. First of all, thank my wife and son for having me supported at all times. I thank Professor Miguel Coimbra and my colleagues at work for all the help provided because without it would not have been possible to have finished this long way.

A sincere thank you to all.



# Resumo

O propósito deste trabalho é analisar métodos de processamento de imagem e deep learning de forma a identificar retângulos, círculos, linhas, e outras figuras geométricas em desenho técnico assim como informação de tolerâncias e de dimensões.

A motivação deste trabalho é reduzir o tempo necessário bem como os erros derivados pela introdução manual por um operador, agravados pela introdução repetitiva. Esses dados estão presentes em imagens de desenho técnico.

Os objetivos são a identificação automática de figuras geométricas e a informação de tolerâncias e as suas dimensões.

Neste trabalho foram identificados os vários desafios e identificados vários algoritmos que puderam ajudar a resolver o problema proposto utilizando métodos de deep learning e quando necessário alguns algoritmos de visão por computador.

Os resultados obtidos foram promissores. Porém, ainda tem muito espaço para melhorias.

Para melhorar o trabalho desenvolvido serão necessárias mais imagens produzidas de forma sintética idênticas às reais ou imagens reais anotadas cedidas pelos clientes.

**Palavras Chave:** aprendizagem de máquina, aprendizagem profunda, detecção de formas, ocr, detecção de texto, reconhecimento de texto





# Abstract

The purpose of this work is to analyze image processing and deep learning methods to identify rectangles, circles, lines, and other geometric figures in technical drawing as well as tolerance and dimension information.

The motivation of this work is to reduce the required time as well as the errors caused by manual input by an operator, compounded by repetitive input. This data is present in technical drawing images.

The objectives are the automatic identification of geometric figures and the information of tolerances and their dimensions.

In this work, we identified the various challenges and identified several algorithms that could help solve the proposed problem using deep learning methods and when necessary some computer vision algorithms.

The results obtained were promising. However, there is still a lot of room for improvement.

To improve the work done, it will be necessary to generate more images, with better quality and that represent better real images. Images provided by customers may also help to improve.

**Keywords:** machine learning, deep learning, shape detection, ocr, text detection, text recognition



# Contents

<b>Acknowledgements</b>	<b>5</b>
<b>Resumo</b>	<b>7</b>
<b>Abstract</b>	<b>9</b>
<b>List of Tables</b>	<b>15</b>
<b>List of Figures</b>	<b>20</b>
<b>1 Introduction</b>	<b>25</b>
1.1 Motivation . . . . .	25
1.2 Technical drawings . . . . .	27
1.3 Objectives and challenges . . . . .	32
1.4 Document structure . . . . .	34
<b>2 Understanding deep learning</b>	<b>35</b>
2.1 Perceptron . . . . .	35
2.2 Activation functions . . . . .	36
2.3 Artificial neural network . . . . .	37

2.4	Training neural networks . . . . .	38
2.4.1	Backpropagation . . . . .	38
2.5	Convolutional Neural Networks . . . . .	39
2.5.1	Transfer learning . . . . .	41
2.5.2	Synthetic data . . . . .	42
<b>3</b>	<b>State of the art</b>	<b>43</b>
3.1	Text detection and recognition . . . . .	43
3.1.1	Text detection . . . . .	44
3.1.2	Character recognition . . . . .	44
3.1.3	Text recognition . . . . .	47
3.1.4	Recurrent Neural Network . . . . .	47
3.1.5	Long Short-Term Memory . . . . .	48
3.1.6	Gated Recurrent Unit . . . . .	49
3.2	Object detection . . . . .	50
3.2.1	Region-based Convolutional Neural Networks . . . . .	51
3.2.2	Fast Region-based Convolutional Neural Networks . . . . .	52
3.2.3	Faster Region-based Convolutional Neural Networks . . . . .	53
3.2.4	Fully Convolutional Network for semantic segmentation . . . . .	53
3.2.5	Mask Region-based Convolutional Neural Networks . . . . .	54
3.2.6	You Only Look Once . . . . .	55
3.2.7	Single Shot Multibox Detection . . . . .	56
3.3	Discussion . . . . .	56

<i>CONTENTS</i>	13
<b>4 Processing technical drawings using deep learning</b>	<b>57</b>
4.1 Computational tools . . . . .	58
4.2 Processing pipeline Overview . . . . .	59
4.3 Image preprocessing . . . . .	60
4.3.1 Color to grayscale conversion . . . . .	60
4.3.2 Perspective Transform . . . . .	60
4.3.3 Adaptive Gaussian Thresholding . . . . .	60
4.4 Text detection . . . . .	63
4.5 Feature Control Frame, Datum and Text detection . . . . .	65
4.6 Text recognition . . . . .	68
4.7 Shape detection . . . . .	69
4.8 Text and shape relationship . . . . .	70
4.9 Results . . . . .	72
<b>5 Conclusions and Future work</b>	<b>87</b>
5.1 Conclusion . . . . .	87
5.2 Future Work . . . . .	89
<b>References</b>	<b>90</b>
<b>A Synthetic images</b>	<b>97</b>
<b>B Shape detection precision matrix</b>	<b>99</b>



# List of Tables

1.1	Geometric tolerance symbols and their usage. . . . .	28
1.2	GD&T modifiers. . . . .	29
4.1	mAP of text detection at IoU=0.5 . . . . .	74
4.2	mAP of shapes detection at IoU=0.5 . . . . .	75
4.3	mAP tested with 1000 images . . . . .	75
4.4	Precision, Recall and F1 Score at different IoU . . . . .	79
4.5	Precision and recall of text recognition with and without line removal . . . . .	83
4.6	Precision and recall of some text recognition without line removal . . . . .	83
4.7	Precision and recall of some text recognition with line removal . . . . .	83
4.8	Most common wrong detections of datum character . . . . .	84
4.9	Precision of GD&T symbols recognition . . . . .	85





# List of Figures

1.1	Example of a technical drawing. . . . .	27
1.2	Example of a Feature Control Frame box. . . . .	29
1.3	How to read a Feature Control Frame. . . . .	30
1.4	Four examples of composite Feature Control Frame. . . . .	30
1.5	Sample of a Datum "A" and an Feature Control Frame referencing the datum "A".	30
1.6	GT&T additional symbols . . . . .	31
1.7	Part with dimension information. . . . .	31
1.8	CAM2 Software with sample part and features to be measure . . . . .	33
2.1	Example of a Perceptron . . . . .	36
2.2	Sigmoid function . . . . .	37
2.3	Hyperbolic Tangent function . . . . .	37
2.4	Rectified Linear Unit (ReLU) function . . . . .	37
2.5	An artificial neural network . . . . .	37
2.6	Backpropagation in an ANN. . . . .	38
2.7	How Backpropagation works in detail. . . . .	39
2.8	Gradient descent algorithm. . . . .	39

2.9	Convolutional Neural Networks. . . . .	40
2.10	Machine learning comparison with deep learning. . . . .	41
2.11	Performance comparison of deep learning with traditional machine learning algorithms. . . . .	41
2.12	Transfer learning comparison . . . . .	42
3.1	A typical CNN for classify handwritten digits. . . . .	45
3.2	Example of a convolution filter. . . . .	46
3.3	Max pooling operation with 2x2 filter and stride 2. . . . .	46
3.4	An RNN unit. . . . .	48
3.5	An LSTM Unit. . . . .	49
3.6	An GRU Unit. . . . .	49
3.7	An example of semantic segmentation. . . . .	50
3.8	Overview of different object localization tasks. . . . .	51
3.9	R-CNN. . . . .	52
3.10	Fast R-CNN architecture. . . . .	52
3.11	Faster R-CNN. . . . .	53
3.12	FCN. . . . .	54
3.13	Mask R-CNN. . . . .	55
3.14	YOLO grid. . . . .	55
3.15	SSD. . . . .	56
4.1	Processing pipeline diagram. . . . .	59
4.2	Result of the original image binarized. . . . .	61

<i>LIST OF FIGURES</i>	19
4.3 Fragment of original image and binarization using global thresholding, adaptive mean thresholding, and adaptive Gaussian thresholding. . . . .	62
4.4 Fragment of original image and binarization using Otsu's thresholding and a pre-processing using median blurring and gaussian blurring. . . . .	62
4.5 Text detection using EAST algorithm. . . . .	63
4.6 An example of houghlines. . . . .	64
4.7 Detecting lines with Hough lines algorithm. . . . .	64
4.8 Example of generated image for training text detection. . . . .	65
4.9 Example of some generated mask images for the previous image. . . . .	66
4.10 An illustration of an FPN. . . . .	66
4.11 Text detection using the trained Mask Region-based Convolutional Neural Networks (Mask R-CNN). . . . .	67
4.12 Text detection in the generated images using EAST algorithm. . . . .	68
4.13 Result image after applying a morphological operation to remove horizontal and vertical lines. . . . .	69
4.14 Text recognition in an FCF after lines removal. . . . .	69
4.15 Example of generated image for training and image with masks. . . . .	70
4.16 Shapes detection using the trained Mask R-CNN. . . . .	71
4.17 Arrows detection using the trained Mask R-CNN. . . . .	71
4.18 Training/Validation loss of text detection. . . . .	72
4.19 Training/Validation loss of shapes and text detection. . . . .	73
4.20 An Illustration of precision, recall and IoU. . . . .	73
4.21 Example of IoU scores. . . . .	74
4.22 Detection on sample image. . . . .	76

4.23	Confusion matrix of one test image. . . . .	77
4.24	Confusion matrix. . . . .	78
4.25	Image with original annotations. . . . .	79
4.26	Detection using the trained Mask R-CNN. . . . .	80
4.27	Detection of some rotated shapes (1). . . . .	80
4.28	Detection of some rotated shapes (2). . . . .	81
4.29	Detection of real world drawing. . . . .	82
4.30	Examples of line removal in GD&T symbols. . . . .	84
4.31	Examples of line removal in FCF. . . . .	85
4.32	Result of FCF text recognition. . . . .	86
4.33	Bug in Tesseract text recognition or an overfitting problem. . . . .	86
A.1	Sample of 4 images used for training. . . . .	98
B.1	Confusion matrix of a sample image (1). . . . .	100
B.2	Confusion matrix of a sample image (2). . . . .	101





# Acronyms

**ANN** Artificial Neural Network. 35, 37

**CAD** Computer-Aided Design. 32, 57

**CNN** Convolutional Neural Network. 44–47, 50, 51, 88

**CPU** Central Processing Unit. 69

**Fast R-CNN** Fast Region-based Convolutional Neural Networks. 52

**Faster R-CNN** Faster Region-based Convolutional Neural Networks. 53

**FCF** Feature Control Frame. 29, 30, 32, 64, 65, 67–69, 85, 86, 88

**FCN** Fully Convolutional Network. 52–54

**FN** False Negative. 74

**FP** False Positive. 74

**GD&T** Geometric Dimensioning and Tolerancing. 27, 28, 30–32, 43, 63, 68, 69, 84–86

**GPU** Graphics Processing Unit. 58, 69

**GRU** Gated Recurrent Unit. 47, 49

**IoU** Intersection over Union. 73, 74, 78, 79

**LSTM** Long Short-Term Memory. 47–49, 58, 68, 88

**mAP** mean Average Precision. 74, 78

**Mask R-CNN** Mask Region-based Convolutional Neural Networks. 19, 54, 66–71, 74, 75, 88, 89

**MSER** Maximally Stable Extremal Regions. 44

**OCR** Optical Character Recognition. 43, 44, 47, 88

**PDF** Portable Document Format. 57, 59

**R-CNN** Region-based Convolutional Neural Networks. 51–54

**ReLU** Rectified Linear Unit. 17, 36, 37

**RNN** Recurrent Neural Network. 47, 48

**ROI** Region of Interest. 52, 54

**RPN** Region Proposal Network. 53

**SSD** Single Shot Multibox Detection. 56

**SWT** Stroke Width Transform. 44

**TP** True Positive. 74

**TPU** Tensor Processing Unit. 58

**YOLO** You Only Look Once. 55, 56, 69



# Chapter 1

## Introduction

---

1.1	Motivation . . . . .	25
1.2	Technical drawings . . . . .	27
1.3	Objectives and challenges . . . . .	32
1.4	Document structure . . . . .	34

---

### 1.1 Motivation

The purpose of this work is to process technical drawings and extract geometric information and tolerances from the image, and then that information is automatically entered into existing software. This software is used by thousands of users, in parts manufacturing companies or metrology consulting companies.

The number of different parts inspected varies widely, from less than a dozen to more than a thousand. These parts can be part of a complex system such as an automobile or aircraft, where each part has a technical drawing with tolerances and dimensions.

This is why a lot of time is spent introducing tolerances and dimensions in metrology software, which entails huge costs as well as more specialized people. An incorrect introduction may result in defectively manufactured parts which later in quality control pass within tolerances. The opposite can also be true where parts within tolerances and dimensions are discarded resulting in garbage and damage due to quality control being input errors.

In the last years, algorithms for image processing based on machine learning and deep learning have been used successfully. Many of these algorithms are not new, but with the increasing processing power of recent computers makes their usage more viable.

These recent algorithms are applied to complex things like autonomous driving, real-time translation of informational panels or products, describing photos, written text recognition.[1]

Three companies have publicly shown that they are also trying to solve this problem, two of them has already made available to their customers' functionality that allows them to solve part of the problem with the help of the user.[2][3] The other was published in university research and also tries to solve the problem partially.[4]

## 1.2 Technical drawings

Figure 1.1 depicts a simple technical drawing that contains typical elements including shapes, Geometric Dimensioning and Tolerancing (GD&T) and dimensions. GD&T is a symbolic language to communicate manufacturing constraints and tolerances. Usually, GD&T information is placed near to shape, however, there are situations in which GD&T tag is associated with several shapes. In Figure 1.1, circles labeled from A1 to A8 share the same GD&T information. The drawing of the part may be placed in various perspectives.

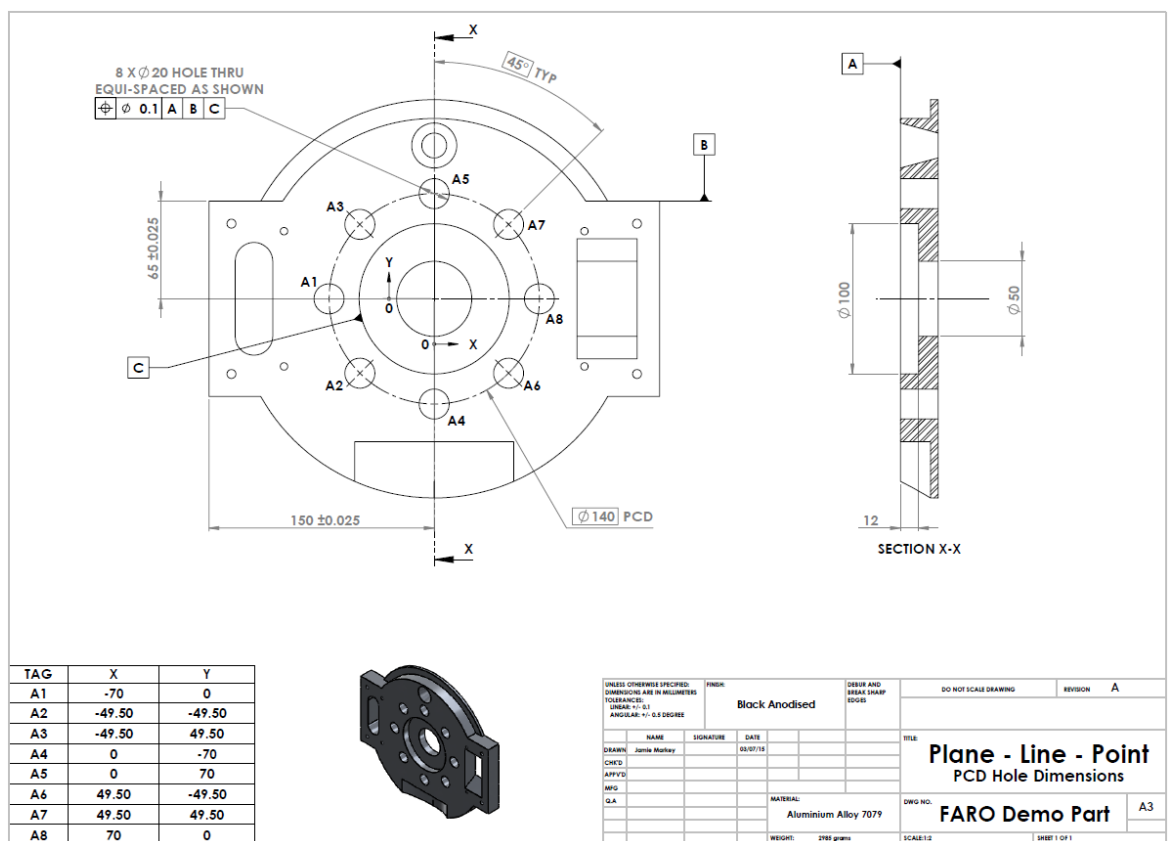


Figure 1.1: Example of a technical drawing.

Source: Provided by FARO Technologies, Inc.

Table 1.1 shows fourteen different geometric tolerance symbols. Being associated with a geometric characteristic, they are grouped by tolerance type. Some of them do not have relation between features. There are special fonts that include these symbols having them slight variations.

Table 1.1: Geometric tolerance symbols and their usage.















Symbol	Geometric Characteristic	Tolerance Type	Control Summary
	Flatness	Form (no relation between features)	Controls form (shape) of surfaces and can also control form of an axis or median plane. Datum reference is not allowed
	Straightness		
	Cylindricity		
	Circularity (Roundness)		
	Perpendicularity	Orientation (no relation between features)	Controls orientation (tilt) of surfaces, axes, or median planes for size and non-size features. Datum reference required
	Parallelism		
	Angularity		
	Position	Location	Locates center points, axes, and median planes for size features. Also controls orientation.
	Profile of a surface		Locates surfaces. Also controls size, form, and orientation of surfaces based on datum references
	Profile of a line		
	Total runout	Runout	Controls surface coaxiality. Also controls form and orientation of surfaces
	Circular runout		
	Concentricity	Location	Locates derived median points of a feature
	Symmetry		

Table 1.2 examples of some GD&T modifiers. The modifiers are only allowed with some geometric characteristics.

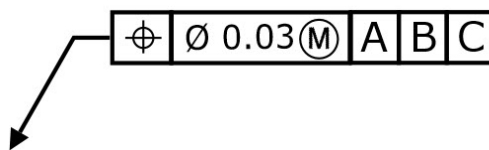
Table 1.2: GD&amp;T modifiers.

Symbol	Meaning
$\textcircled{L}$	LMC - Least Material Condition
$\textcircled{M}$	MMC - Maximum Material Condition
$\textcircled{T}$	Tangent Plane
$\textcircled{P}$	Projected Tolerance Zone
$\textcircled{F}$	Free State
$\textcircled{\varnothing}$	Diameter

The Geometric tolerance symbol is placed inside a Feature Control Frame (FCF). The FCF includes information of:

- Geometric tolerance symbol
- Tolerance zone type and dimensions
- Tolerance zone modifiers
- Datum references (optional, but the maximum references is 3)

Usually, the FCF has an arrow pointing to the related shape. Figure 1.2 shows an example of an FCF and Figure 1.3 shows how to read the same FCF.

Figure 1.2: Example of an FCF box.<sup>1</sup>

In the sample FCF, the information can be read as "This feature has a true position of 30 micron diameter tolerance and in reference to the datum A, B and C with maximum material condition". There are geometric tolerances that do not have an associated datum; such as flatness, straightness, cylindricity or circularity.

<sup>1</sup>Image from: <https://www.gdandtbasics.com/feature-control-frame/>



Figure 1.3: How to read an FCF.<sup>2</sup>

The FCF in a technical drawing may be more complicated as shown in Figure 1.4. Besides, it is also possible to have different geometric tolerance in the same FCF.

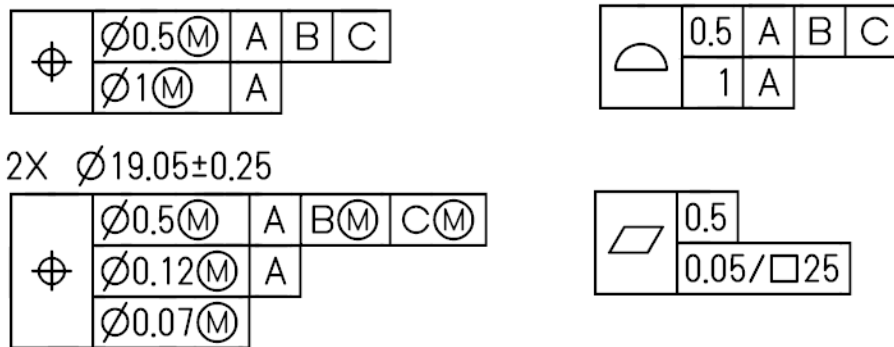


Figure 1.4: Four examples of composite FCF.<sup>3</sup>

A datum is a theoretical (exact) plane, axis or point coordinate that GD&T or dimensional tolerances are referenced to it. It is usually an important functional feature which needs to be controlled during measurement. Each datum is identified by a box containing an identifier such for example "A", and an inverted arrow (or arrows) connected to the shape surface. Figure 1.5 shows a datum and an FCF pointing to a datum.

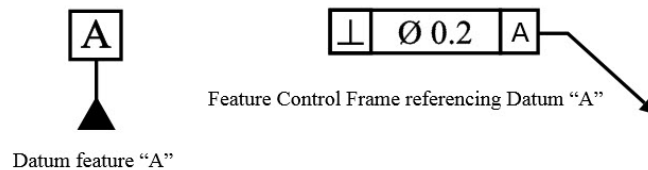


Figure 1.5: Sample of a Datum "A" and an FCF referencing the datum "A".<sup>4</sup>

In more complicated technical drawings, there are symbols similar to shapes, and they can

<sup>2</sup>Image from: <https://www.gdandtbasics.com/feature-control-frame/>

<sup>3</sup>Image from: [http://p-00-kanold.com/?page\\_id=7&cpage=1](http://p-00-kanold.com/?page_id=7&cpage=1)

<sup>4</sup>Image from: <https://www.gdandtbasics.com/datum/>

be inside shapes. In these scenarios, they might be detected incorrectly, resulting in mismatch detection. There are additional symbols in a technical drawing such shown in Figure 1.6.

Symbol	Meaning	Symbol	Meaning
	Dimension Origin	R	Radius
	Counterbore	SR	Spherical Radius
	Countersink	SØ	Spherical Diameter
	Depth	CR	Controlled Radius
	All Around		Statistical Tolerance
	Between		Basic Dimension
	Target Point	(77)	Reference Dimension
	Conical Taper	5X	Places
	Slope		
	Square		

Figure 1.6: GD&T additional symbols.<sup>5</sup>

There are also dimension information such as shown in Figure 1.7 while their appearance can be different.

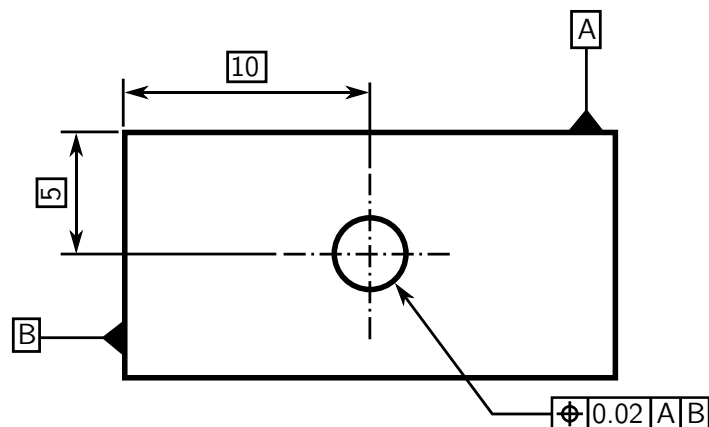


Figure 1.7: Part with dimension information.<sup>6</sup>

<sup>5</sup>Image adapted from: [https://www.sigmetrix.com/wp-content/uploads/2012/04/GDT\\_Symbols\\_Reference\\_Guide2.pdf](https://www.sigmetrix.com/wp-content/uploads/2012/04/GDT_Symbols_Reference_Guide2.pdf)

<sup>6</sup>Image from: [https://en.wikipedia.org/wiki/Geometric\\_dimensioning\\_and\\_tolerancing](https://en.wikipedia.org/wiki/Geometric_dimensioning_and_tolerancing)

### 1.3 Objectives and challenges

The objectives of this work are:

1. Detecting geometry shapes like rectangles, circles, round-slots, lines and other primitives located in a technical drawing.
2. Related information like tolerances and dimensions.

In this work, three main challenges are addressed. The first one is dealing with the detection of various FCF inside a technical drawing. The challenge is then continued by extracting the geometric tolerance symbol and tolerance information. There are several variations of possible combinations of the values inside an FCF. Alternatively, text zones can also be detected. Since there is no standard to define the text angle, it can be placed horizontally, vertically or with preferred orientations.

The second challenge is detecting shapes inside the technical drawing and reporting back the proper location. Shapes can also be inside each other, which complicate the procedure of detection.

The third challenge is to associate a proper relation between detected shapes with the corresponding FCF since there are different ways of this representation. Normally, the Computer-Aided Design (CAD) file of the part in the drawing is often available which can help validate the detection using its projection and find the correspondences with the drawing to analyze. It is also possible to make a CAD file with GD&T information, but this work is for use cases when this information is not present, which according to FARO Technologies vendors and application engineers, only about 10% of CAD files used by customers contain this information.



Figure 1.8 show the CAM2 Measure software with and sample part and the features to be measure. Each of the features has associated tolerances that must be entered manually.

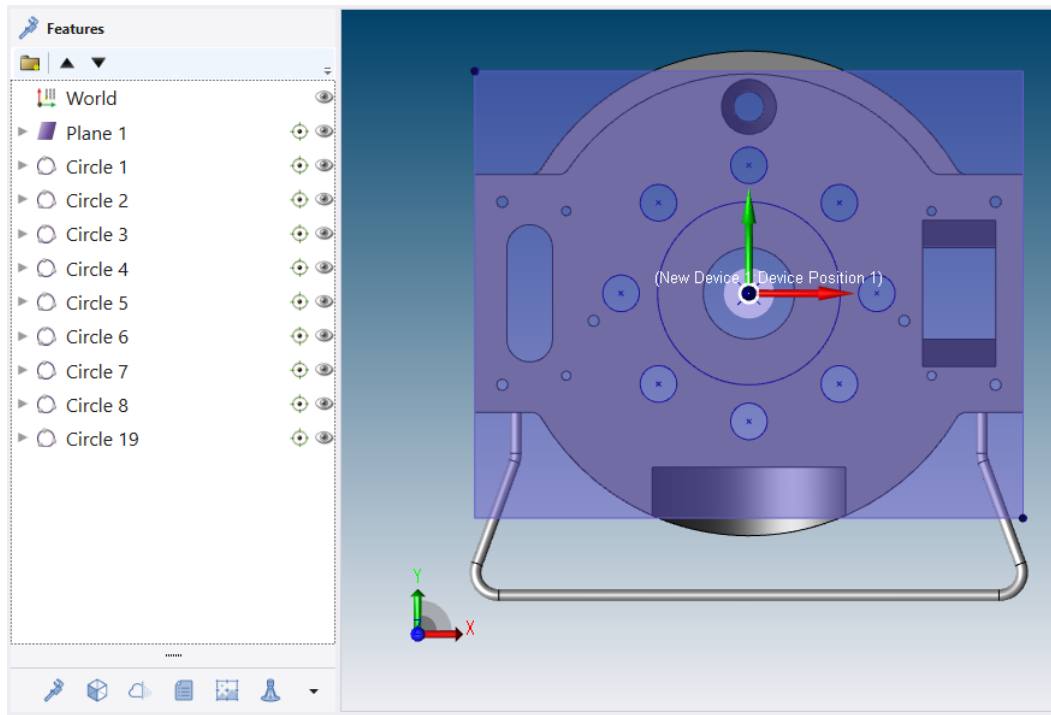


Figure 1.8: CAM2 Software with sample part and features to be measure

## 1.4 Document structure

This document is presented in 5 chapters:

- **Chapter 1 - Introduction:** the project are explained and its motivation;
- **Chapter 2 - Understanding deep learning:** the basic concepts of deep learning are explained;
- **Chapter 3 - State of the art:** the latest techniques to solve the problem are explained;
- **Chapter 4 - Processing technical drawings using deep learning:** the methods and algorithms used to solve the proposed problem are indicated. The tests to validate the model generalization are explained;
- **Chapter 5 - Conclusions and Future work:** a final discussion is presented along with relevant conclusions.

## Chapter 2

# Understanding deep learning

---

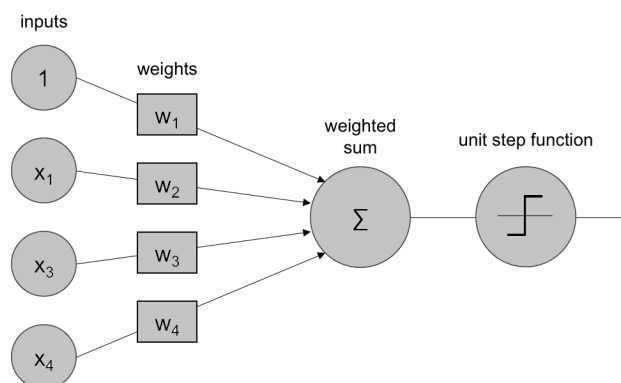
<b>2.1</b>	<b>Perceptron . . . . .</b>	<b>35</b>
<b>2.2</b>	<b>Activation functions . . . . .</b>	<b>36</b>
<b>2.3</b>	<b>Artificial neural network . . . . .</b>	<b>37</b>
<b>2.4</b>	<b>Training neural networks . . . . .</b>	<b>38</b>
<b>2.5</b>	<b>Convolutional Neural Networks . . . . .</b>	<b>39</b>

---

Deep learning is a collection of techniques from Artificial Neural Network (ANN), which is a branch of machine learning. ANNs are inspired by information processing and distributed communication nodes in a biological system such as a human brain. Nodes, which are called artificial neurons, are responsible for the processing, and the flowing of information in the network.

### 2.1 Perceptron

A perceptron is an artificial neuron with several inputs, witch performs a weighted summation to produce an output. The weight of each perceptron is determined during a procedure called training, which depends on training data. A perceptron can only learn simple functions by learning the weights from examples. The training of a perceptron can be carried out through gradient-based methods which are explained later. The output of the perceptron can be obtained by an activation function or a transfer function. Figure 2.1 shows a perceptron.



Source: Adapted from [5].  
Figure 2.1: Example of a Perceptron

## 2.2 Activation functions

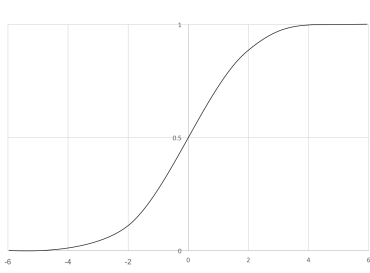
There are several possible activation functions for an artificial neuron. An activation function decides whether a perceptron should 'fire' or not. During training, activation functions are important to adjust the gradients during the training process.

Figure 2.2 depicts a Sigmoid function. It can be considered a smoothed step function, ranging from 0 to 1. It is useful for converting any value to probabilities and can be used for binary classification.[5]

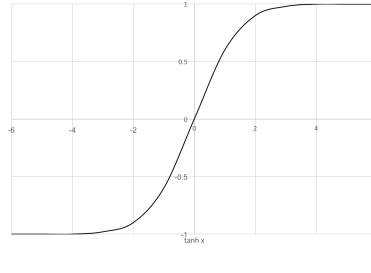
Figure 2.3 shows a Hyperbolic Tangent function which acts similarly a sigmoid, but it maps the input value in a wider range: from -1 to 1. It has been reported that employing a Hyperbolic Tangent function leads to more stable gradients with fewer vanishing problems. [5][6]

Figure 2.4 shows a Rectified Linear Unit (ReLU) function. It maps input  $x$  to  $\max(0, x)$ . The negative inputs are converted to 0. Since the function is simple, it is computationally inexpensive and can be trained faster.[5][7]

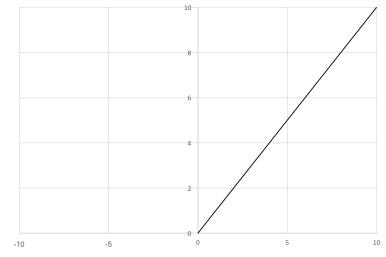
The choice of an activation function is very dependent on the application and its data.



Source: Adapted from Shanmugamani [5].  
Figure 2.2: Sigmoid function



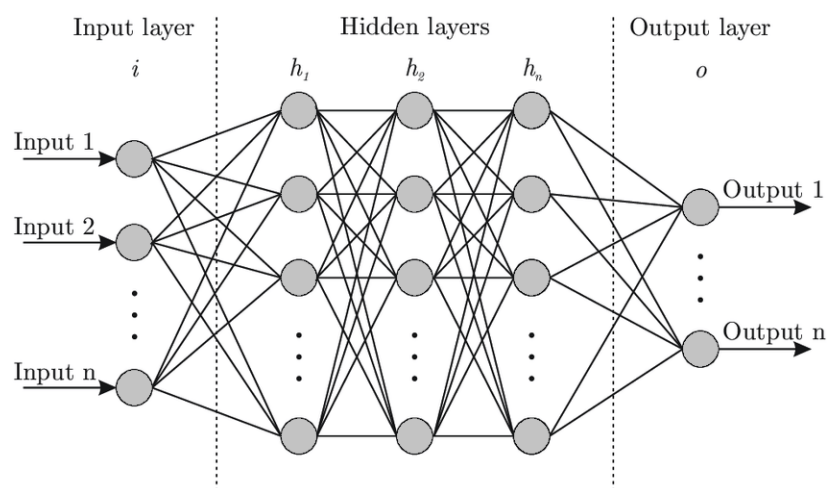
Source: Adapted from Shanmugamani [5].  
Figure 2.3: Hyperbolic Tangent function



Source: Adapted from Shanmugamani [5].  
Figure 2.4: ReLU function

## 2.3 Artificial neural network

An Artificial Neural Network (ANN) is a processing model inspired by biological neural networks. The core component of ANNs is artificial neurons. Each neuron receives inputs from several other neurons, multiplies them by assigned weights, adds them and passes the sum to one or more neurons. Some artificial neurons might apply an activation function to the output before passing it to the next variable.[8][9] Figure 2.5 shows a simple ANN, containing an input layer, hidden layers, and an output layer.



Source: Adapted from Bre et al. [10].  
Figure 2.5: An artificial neural network

## 2.4 Training neural networks

The training process determines the values of these weights. The model values can be initialized randomly in the beginning of the training. The error is then computed using a cost function by subtracting it from the values of the training data (also known as ground truth). Based on the computed cost, weights are tuned in a way to reduce the error in every step. One popular procedure for calculating the weights of each node is called backpropagation [11]. Therefore, a stop criterion of the training process can be defined upon no further decrease is obtained for the error values.

### 2.4.1 Backpropagation

During the training, the weights are updated backwards based on the error calculated. Figure 2.6 shows how backpropagation works at a high level.

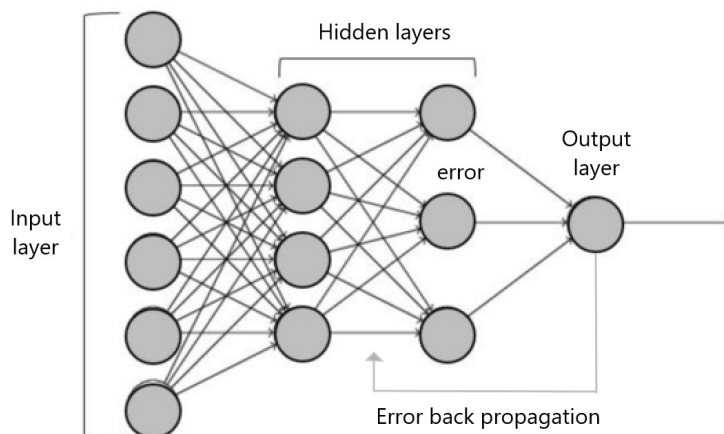
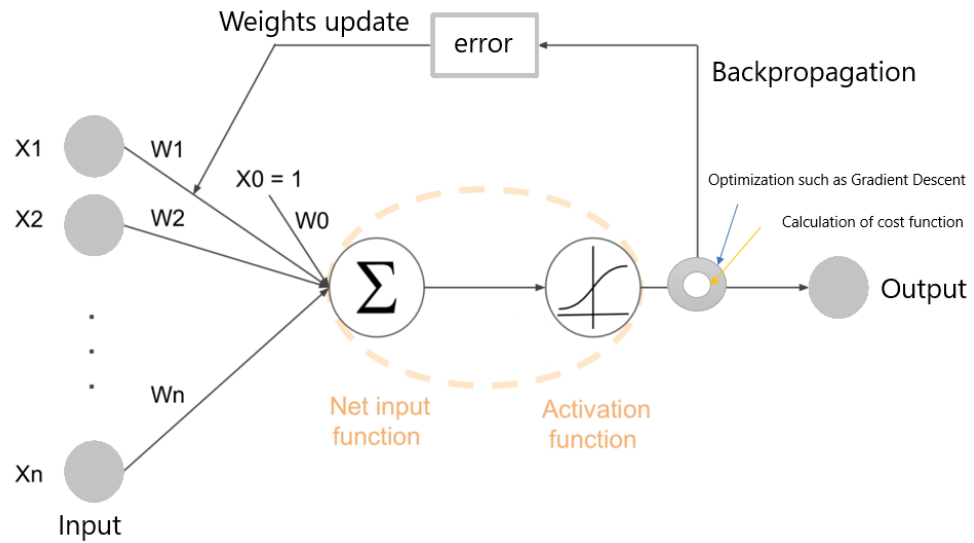
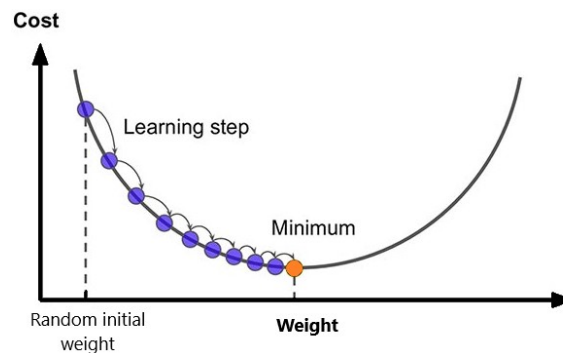


Figure 2.6: Backpropagation in an ANN.

Figure 2.7 shows in detail of backpropagation. It employs the gradient descent algorithm to iteratively update the weights of inputs and minimize the output error. [12]

Figure 2.7: How Backpropagation works in detail.<sup>1</sup>

Gradient descent is widely employed to identify the optimal value by taking large iterative steps when it is far away from the desired solution and small steps when it gets close, assuming that the steepness of the function gradient is a good clue for this distance. One of most used common cost function is Cross-Entropy [13]. Figure 2.8 depicts an example of a gradient descent algorithm.

Figure 2.8: Gradient descent algorithm.<sup>2</sup>

## 2.5 Convolutional Neural Networks

Convolutional Neural Networks is a class of deep neural network that uses multiples layers and progressively extracts higher-level features from raw input. It is most commonly used in image

<sup>1</sup>Image from: <https://datascience.stackexchange.com/questions/44703/how-does-gradient-descent-and-backpropagation->

<sup>2</sup>Image from: <https://saugatbhattarai.com.np/what-is-gradient-descent-in-machine-learning/>

processing. In an image recognition application, the raw input may be a matrix of pixels; the first representational layer may abstract the pixels and encode edges; the second layer may compose and encode arrangements of edges; the third layer may encode a nose and eyes; and the fourth layer may recognize that the image contains a face.

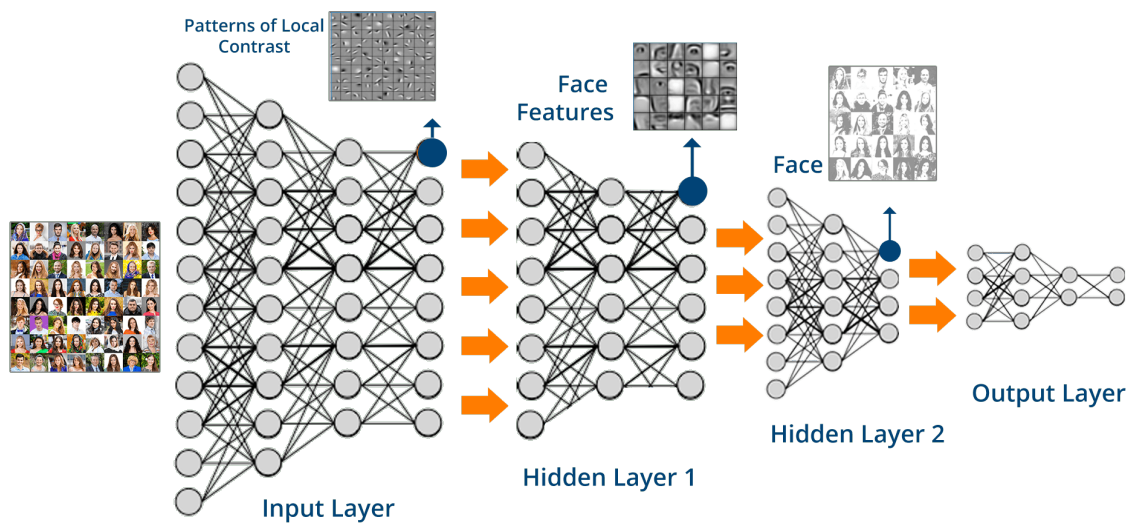


Figure 2.9: Example of a Convolutional Neural Networks.<sup>3</sup>

In traditional machine learning algorithms the feature extraction is done by a human, but in Convolutional Neural Networks the feature extraction is done automatically.[14]

<sup>3</sup>Image from: <https://cdn.edureka.co/blog/wp-content/uploads/2017/05/Deep-Neural-Network-What-is-Deep-Learning-Edureka.png>



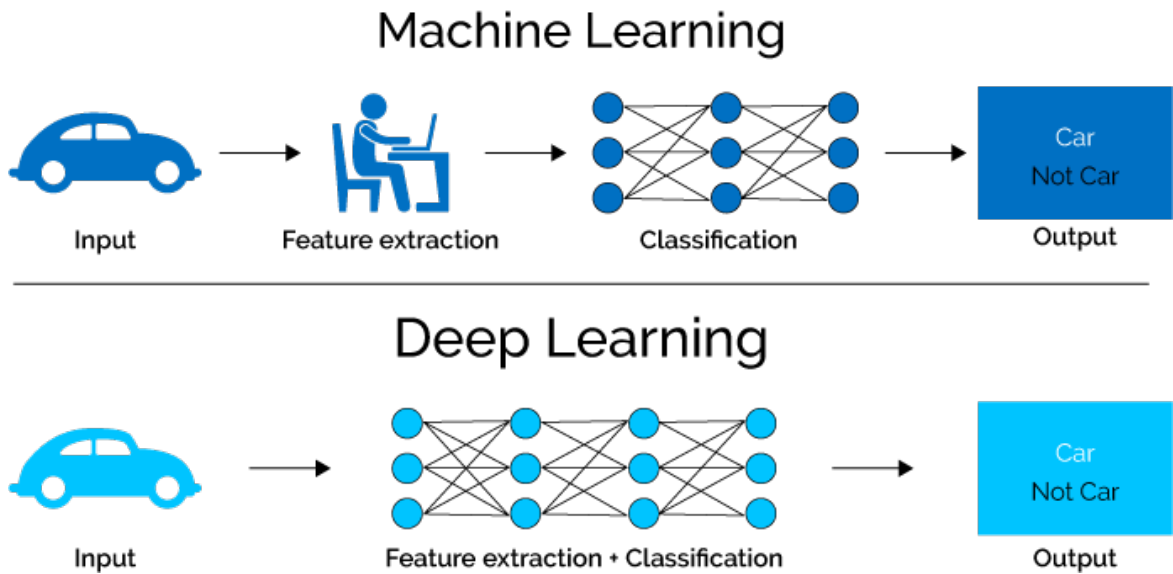


Figure 2.10: Machine learning comparison with deep learning.<sup>4</sup>

Figure 2.11 shows the performance comparison between a traditional machine learning algorithm and a deep learning algorithm. Performance is much higher with increasing if the amount of training data is large.[15]

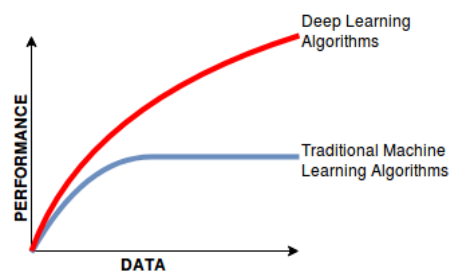


Figure 2.11: Performance comparison of deep learning with traditional machine learning algorithms.<sup>5</sup>

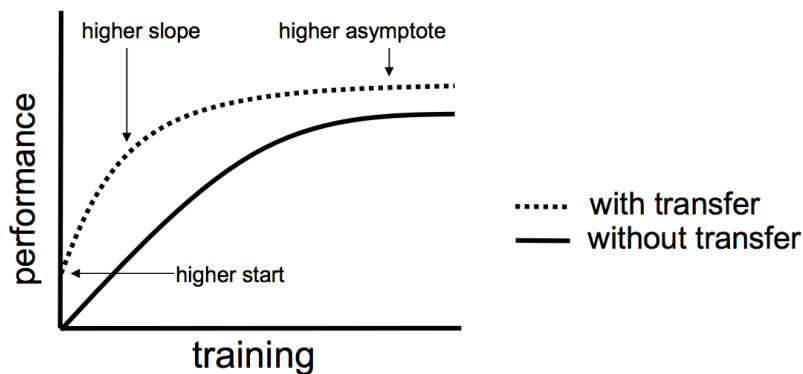
### 2.5.1 Transfer learning

Transfer learning is a machine learning technique to reuse a trained model as a starting point to develop other models to solve different problems. Although starting a model from scratch leads in a specified solution to a problem, it costs a lot of resources, temporally and computationally.

<sup>4</sup>Image from: <https://cdn.edureka.co/blog/wp-content/uploads/2017/05/Deep-Neural-Network-What-is-Deep-Learning-Edu.png>

<sup>5</sup>Image from: <https://www.datacamp.com/community/tutorials/machine-deep-learning>

This technique allows to speed up the training process while it improves the performance of the second model. Also, it can help solve overfitting by using more general data[16]. Figure 2.12 shows an improved performance throughout the training process.



Source: Adapted from Torrey and Shavlik [17].

Figure 2.12: Transfer learning comparison

### 2.5.2 Synthetic data

When there is little data for training, another option is to create a synthetic dataset that can represent data close to the real. On the other hand, there may be a lot of data but it cannot be used for various reasons such as privacy issues[18]. A mix of real data with synthetic data may also be used with some advantages [19].

There are some benefits of using synthetic data such as [20]:

- Protecting privacy
- Enabling exploratory development
- Focusing CS research attention on problems of national priority
- Ensuring reproducibility
- Reducing multiple hypothesis issues
- Avoiding propagation of bias
- Suppressing sensitive signals

# Chapter 3

## State of the art

---

<b>3.1</b>	<b>Text detection and recognition . . . . .</b>	<b>43</b>
<b>3.2</b>	<b>Object detection . . . . .</b>	<b>50</b>
<b>3.3</b>	<b>Discussion . . . . .</b>	<b>56</b>

---

In this chapter, I aimed to analyze several state of the art algorithms which are used to solve the proposed problem. First, I will investigate the state of the art algorithms for text detection and character recognition, followed by the algorithms for detecting and localizing geometrical shapes.

### 3.1 Text detection and recognition

The task of recognizing text is called Optical Character Recognition (OCR) and generally consists of text localization and recognition. Before recognizing the text, it is necessary to first find the zones containing text, then cropping each area, and finally analyze these to recognize the group of characters forming the text. OCR algorithms can also make use of a dictionary to measure the edit distance [21] to known words in order to enhance their accuracy. In this proposed work, the characters to be detected contain special symbols, such as Geometric Dimensioning and Tolerancing (GD&T) symbols, which are not included in typical OCR engines, therefore while these engines can present an appropriate performance for the most common characters, some characters in technical drawings are not recognized correctly.

In the present problem, GD&T symbols may be present alone, so each symbol is not part of a

word, making distance editing techniques not very adequate to improve accuracy.

In classic OCR, there are two methods for character recognition: Matrix matching (also known as template matching), and feature extraction.[22]

Matrix matching is a simpler approach. It compares the character to be detected to a set of known characters. The character set must include the same character in various fonts types. Some preprocessing of the image is required such as binarisation, line removal, layout analysis, line and word detection, character isolation and normalize aspect ratio and scale.[23]

The feature extraction method is more robust since it aims to find features such as open areas, closes shapes, diagonal lines, intersections as a result it is more flexible and able to detect characters from similar fonts.

OCR can also be done using machine learning. A simple algorithm is a Convolutional Neural Network (CNN) which will be explained later.

### 3.1.1 Text detection

Text detection is a task of detecting parts of an image that contains text. The Stroke Width Transform (SWT) and Maximally Stable Extremal Regions (MSER) algorithms can be used to detect text inside the image, and these algorithms only use image processing techniques.

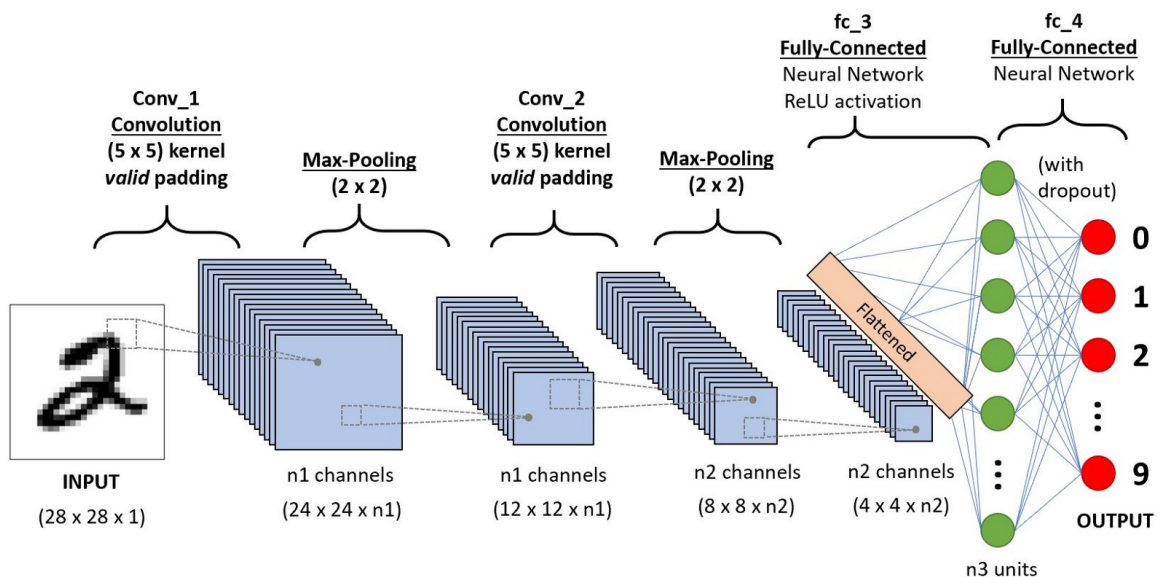
Epshtein et al. [24] describes the implementation of SWT and is capable of detecting text in natural scenes images. Xu-Cheng Yin [25] describes the implementation of MSER and is capable of detecting text in natural scene images, but have some limitations such as can only handle horizontal text due to features and linking strategy. The EAST text detector described by Xinyu Zhou [26] is a deep learning model. It can detect horizontal and non-horizontal text and is very fast.

### 3.1.2 Character recognition

Recent investigations use algorithms based on deep learning to learn the character set and they even allow the recognition of handwriting quite successfully.

One of the simpler deep learning algorithms for image processing is CNN. It is a class of deep neural networks commonly applied to analyzing images and uses relatively little preprocessing compared to traditional image classification algorithms. A CNN algorithm learns filters that in traditional algorithms are hand-engineered, but a simpler CNN only allows image classification. A whole image region is analyzed and the output classifies it into one of the possible categories for which it was trained. For training, a lot of manual classified images is necessary. In some scenarios it can be possible to generate images and the corresponding classification or data annotation which makes it a lot easier. The training process needs much more time for calculation weights of neural network nodes than with conventional ANNs. For accelerating this process, Graphics Processing Units (GPU) are typically used.

Figure 3.1 shows a typical CNN network. The convolution layer of CNN is the first layer and extract features from the input image. It is a mathematical operation that transforms the input image through one or several filters, also known as kernels. The size of the filter can have several sizes but normally is an odd value. The values of kernel determine the type of filter. The output matrix is smaller but can be add some type of the padding to keep the output matrix of the same size.



Source: Adapted from Saha [27].

Figure 3.1: A typical CNN for classify handwritten digits.

Figure 3.2 shows an example of a convolution filter. The algorithm is applied to the whole image and the matrix values around the current pixel are multiplied by the kernel matrix, resulting in this example into a smaller matrix.

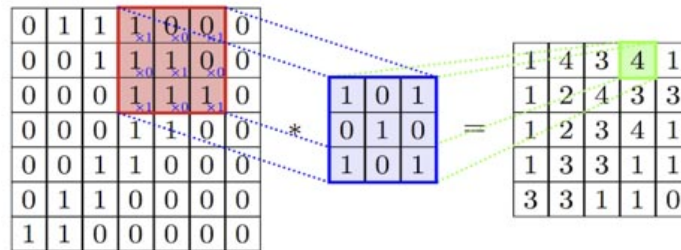
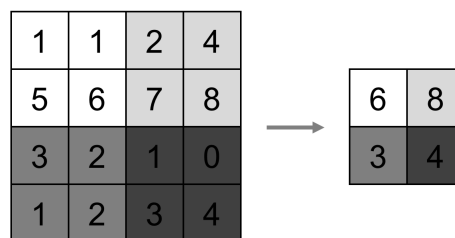


Figure 3.2: Example of a convolution filter.<sup>1</sup>

This resulting matrix of values is sent to the next layer of CNN. In the exemplified CNN, the next layer is a Max-pooling layer.

The pooling layer is not mandatory but commonly used to perform down-sampling to reduce the amount of computational power and also to only send important data to the next layer. There are different kinds of pooling layers. The max pooling only takes the largest value, the average pooling takes the average value from all values, and sum pooling takes the sum of all values. But pooling can cause problems losing positional information. With incremental computer power, there are other ways of down-sampling such as separable and dilated convolutions. The Figure 3.3 shows an example of max pooling operation where four pixels are combined and the highest is sent to the next layer.



Source: Adapted from Shanmugamani [5].

Figure 3.3: Max pooling operation with 2x2 filter and stride 2.

There may be one or more convolution layers connected to the next one. The convolution and pooling layers are part of the feature extraction functional group of the whole CNN.

<sup>1</sup>Image from: <https://insuranalytics.ai/general/different-kinds-convolutional-filters/>

The last layers are the classification layers and consist of fully connected layers. The final layer has as many outputs as possible classifications. Each output produces a probability value. For example, to recognize the numbers from 0 to 9 there are 10 outputs. Usually, when building a CNN several combinations of layers are tested and verified, keeping the ones that obtain better results.

The training process is an optimization iterative process and gradient descent is used to identify the optimal value by taking big steps when the gradient is high and smaller steps when the gradient is low. The most used loss function is the mean squared error.

The training result is a model with weights calculated for each layer of the neural network. Transfer learning can be used to improve the model and reduce the training duration.

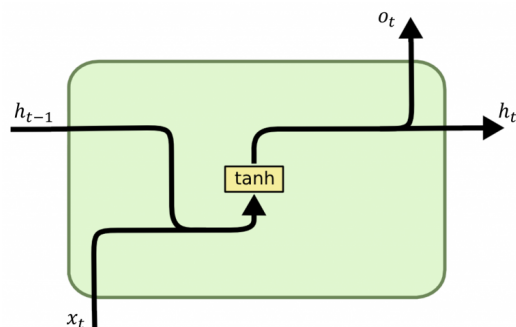
Other CNN based algorithms allow object detection, image segmentation, and object localization. Some of these algorithms are described later.

### 3.1.3 Text recognition

Several papers are referencing Recurrent Neural Network (RNN) [28], Long Short-Term Memory (LSTM)[29][30] and Gated Recurrent Unit (GRU)[31] to perform OCR successfully. They will be described next.

### 3.1.4 Recurrent Neural Network

RNN is a family of neural networks suitable for learning representations of sequential data like text. While each character is not sequential in nature, each character can be interpreted as a sequence of columns and rows. Internally, RNN has hidden states that can process a sequence of data. But RNN has some problems such as error gradient vanishing exponentially quickly.[32] This problem occurs in ANNs trained using gradient descent with backpropagation, and this problem becomes worse as the number of layers in the architecture increases and using activation functions whose derivatives tend to be very close to zero such as the sigmoid function are especially susceptible to the vanishing gradient problem. One way to help solve the problem is to change to an activation function like ReLU.[33] Figure 3.4 shows an RNN unit.

Figure 3.4: RNN Unit.<sup>2</sup>

### 3.1.5 Long Short-Term Memory

LSTM is based on RNN and was discovered in 1997, but in 2007 started to revolutionize some areas such as speech recognition, outperforming traditional models. It has recently been used to perform text recognition and even to perform handwriting recognition.

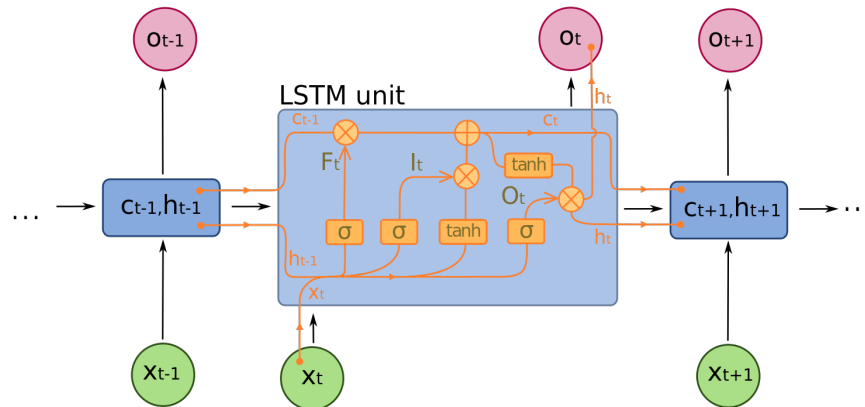
The LSTM is an RNN that has feedback connections. That feedback connection allows the neural network to have persistence and can remember observations over long sequence intervals.

In the LSTM unit, the error is back-propagated from the output layer and the error remains in the LSTM unit cell. This error continuously feeds error back to each LSTM unit's gates until it learns to cut off the value.[34]

Figure 3.5 shows an LSTM unit. A common LSTM unit is composed of a cell, an input gate, an output gate, and a forget gate. The cell is responsible for keeping track of the dependencies between the elements in the input sequence. The input gate controls the extent to which a new value flows into the cell. The forget gate controls the extent to which value remains in the cell. The output gate controls the extent to which value in the cell is used to compute the output activation of the LSTM unit. Commonly, the activation function of LSTM is the logistic function. Each LSTM unit is connected to the next one. The weights of these connections are learned during the training and determine how the gates operate.

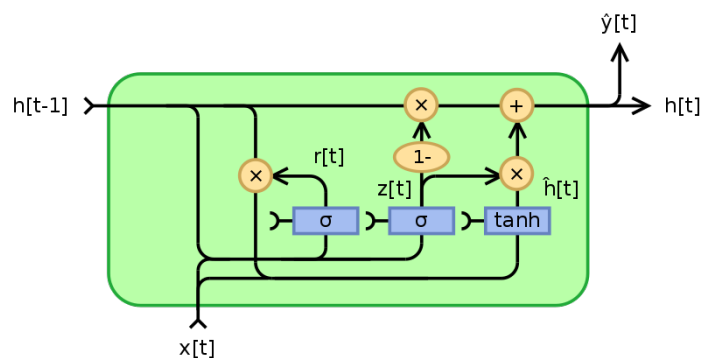
<sup>2</sup>Image from: <https://medium.com/@saurabh.rathor092/simple-rnn-vs-gru-vs-lstm-difference-lies-in-more-flexible-control-5f33e07b1e57>



Figure 3.5: An LSTM Unit.<sup>3</sup>

### 3.1.6 Gated Recurrent Unit

GRU is an LSTM variant that was introduced by [35]. It retains the resisting vanishing gradient properties of LSTM but is internally simpler and faster than LSTM. Figure 3.6 shows an GRU unit. It has only two gates, an update gate and, a reset gate.

Figure 3.6: An GRU Unit.<sup>4</sup>

The update gate ( $r[t]$ ) decides how much of a previous memory to keep around. The reset input defines how to combine new input with the previous value. There is no persistent cell state distinct from the hidden state.

<sup>3</sup>Image from: [https://en.wikipedia.org/wiki/Recurrent\\_neural\\_network](https://en.wikipedia.org/wiki/Recurrent_neural_network)

<sup>4</sup>Image from: [https://en.wikipedia.org/wiki/Recurrent\\_neural\\_network](https://en.wikipedia.org/wiki/Recurrent_neural_network)

## 3.2 Object detection

Semantic segmentation is the task of classifying each pixel in an image into a class label.[36]

Figure 3.7 shows an example of semantic segmentation.

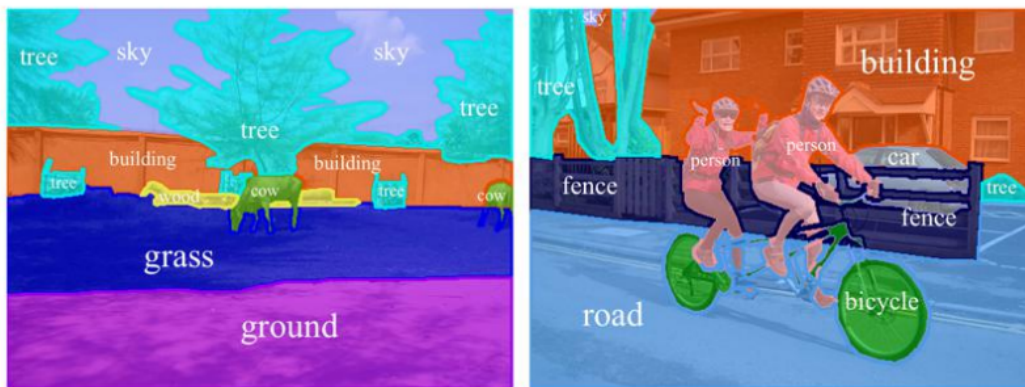


Figure 3.7: An example of semantic segmentation.<sup>5</sup>

Figure 3.8 shows an overview of different object detection tasks and compares them. Detection in this example is the task of verifying if an image contains one or more balloons. A simple CNN is capable of performing this. Semantic segmentation detects all pixels that contain a balloon, but does not distinguish balloons. Object detection knows the location of each balloon but not the exact pixels. Instance segmentation knows the location and the pixels that belong to each balloon.

---

<sup>5</sup>Image from: [http://www.cs.toronto.edu/~tingwuwang/semantic\\_segmentation.pdf](http://www.cs.toronto.edu/~tingwuwang/semantic_segmentation.pdf)

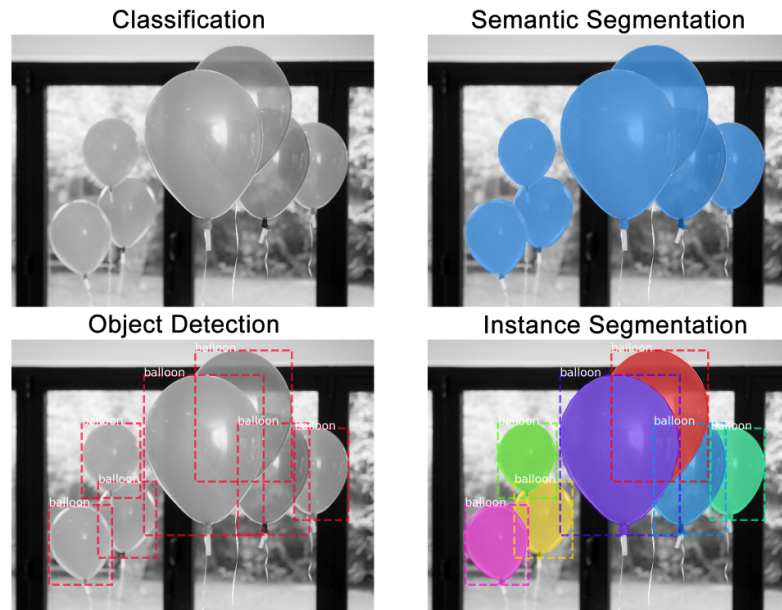


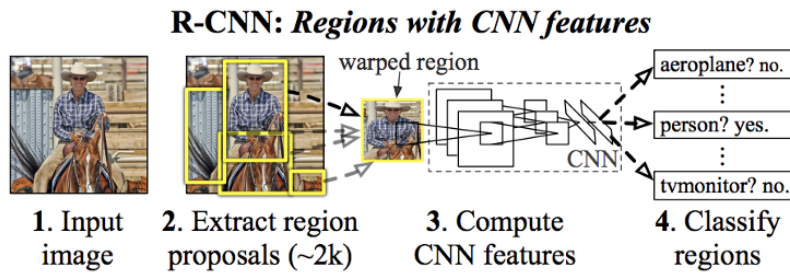
Figure 3.8: Overview of different object localization tasks.<sup>6</sup>

Object detection is the ability to find the presence and position of one or more objects in an image. For detecting multiples objects in the same image, an intuitive method consists of dividing the image into several regions and apply CNN to each region as a separate image. This method requires to have a very large number of regions and requires a huge amount of computational time. In recent years better algorithms have been developed that can analyze an image very quickly and are even fast enough to analyze videos in real-time.

### 3.2.1 Region-based Convolutional Neural Networks

Region-based Convolutional Neural Networks (R-CNN) proposed by Ross Girshick [37] uses a process called selective search to extract boxes (or regions) from the image. It uses image segmentation to split the image into several regions by grouping pixels by texture, color or intensity. These regions produce the final object locations (Region of Interest). Figure 3.9 shows how R-CNN works.

<sup>6</sup>Image from: <https://engineering.matterport.com/splash-of-color-instance-segmentation-with-mask-r-cnn-and-tensorf>



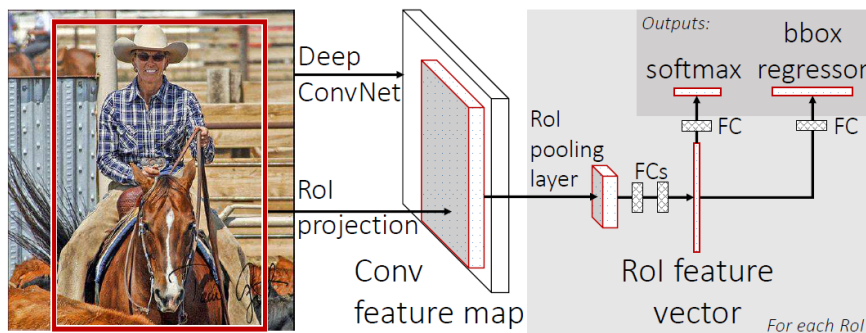
Source: Adapted from [37].  
Figure 3.9: R-CNN.

The R-CNN works well but is quite slow because it has to execute all CNN for every single region of the proposal which are about 2000 regions.[38] It has also to train three different models separately without much shared computation. These models are:[38]

- the CNN for image classification and feature extraction
- the top Support-vector machine classifier for identifying target objects
- the regression model for tightening region bounding boxes

### 3.2.2 Fast Region-based Convolutional Neural Networks

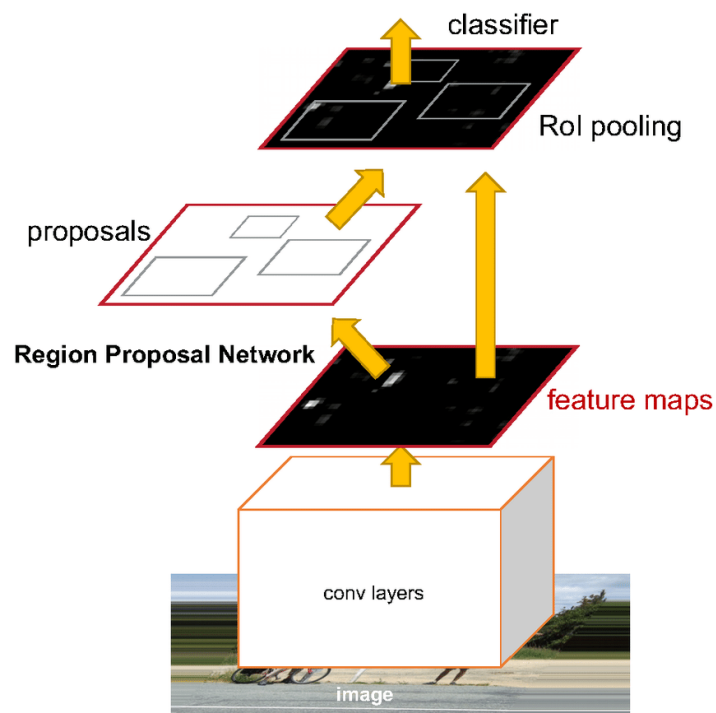
The Fast Region-based Convolutional Neural Networks (Fast R-CNN) proposed by Girshick [39] solves the problems of R-CNN. Figure 3.11 shows a Fast R-CNN architecture. It receives an input image and multiples Region of Interest (ROI) are sent to a Fully Convolutional Network (FCN). Each ROI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers. The output has two output vectors per ROI.



Source: Adapted from [39].  
Figure 3.10: Fast R-CNN architecture.

### 3.2.3 Faster Region-based Convolutional Neural Networks

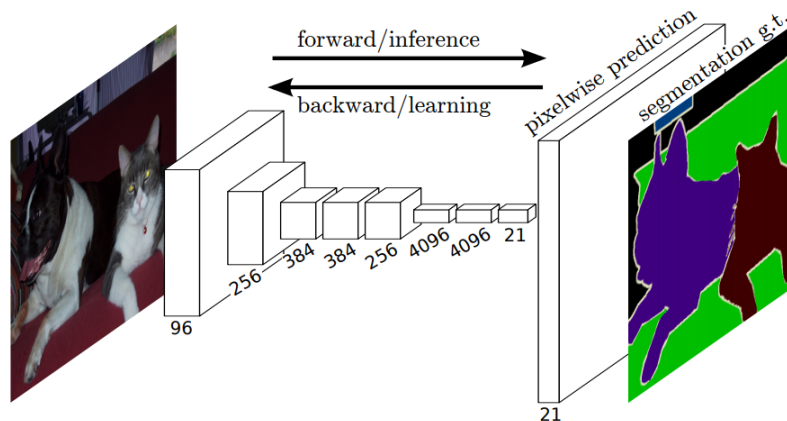
Faster Region-based Convolutional Neural Networks (Faster R-CNN) proposed by Shaoqing Ren [40] uses a Region Proposal Network (RPN) that scans each region and predicts if an object is present or not. But the advantage of RPN is that does not scan the actual image but instead scans the feature map making the process much faster. The Figure 3.11 shows a Faster R-CNN.



Source: Adapted from [40]  
Figure 3.11: Faster R-CNN.

### 3.2.4 Fully Convolutional Network for semantic segmentation

FCN described in Jonathan Long [41] is an extension to Faster R-CNN that allows semantic segmentation. Figure 3.12 shows how an FCN works. This model uses various blocks of convolution layers and max pool layers to first decompress an image to 1/32th of its original size. It then makes a class prediction at this level of granularity. Finally, it uses upsampling and deconvolution layers to resize the image to its original dimensions.[42]



Source: Adapted from [41]

Figure 3.12: FCN.

### 3.2.5 Mask Region-based Convolutional Neural Networks

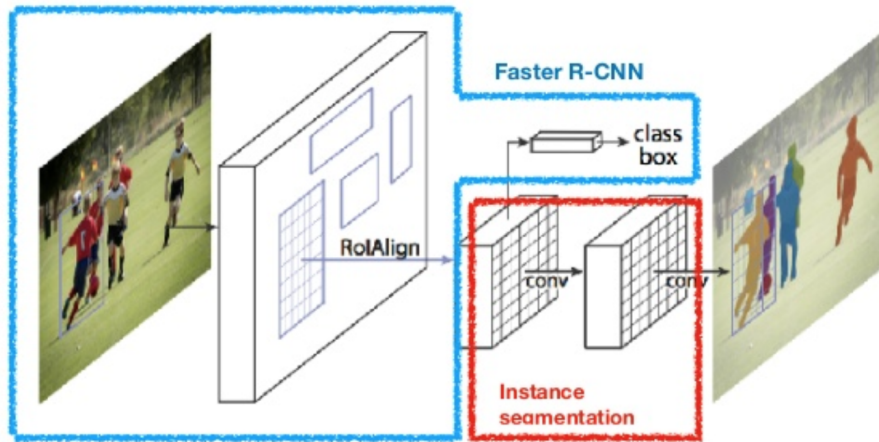
Mask Region-based Convolutional Neural Networks (Mask R-CNN) described in Kaiming He [43] extends Faster R-CNN by adding a branch for predicting segmentation masks on each ROI, in parallel with the existing branch for classification and bounding box regression. The mask branch is a small FCN applied to each ROI, predicting a segmentation mask in a pixel-to-pixel manner. The mask allows pixels segmentation also known as instance segmentation.

It takes the advances of R-CNN and FCN for object detection and semantic segmentation. It is simple to train and adds only a small overhead to the Faster R-CNN. The output gives bounding boxes and segmentation masks on each object. Currently, it is a state-of-the-art framework for image segmentation tasks.

The FCN model can be used for image detection and segmentation.

In the segmentation mask step, the algorithm takes a region of interest as input and generates pixels masks as output. Then these masks are scaled up.

Figure 3.13 shows a mask R-CNN.

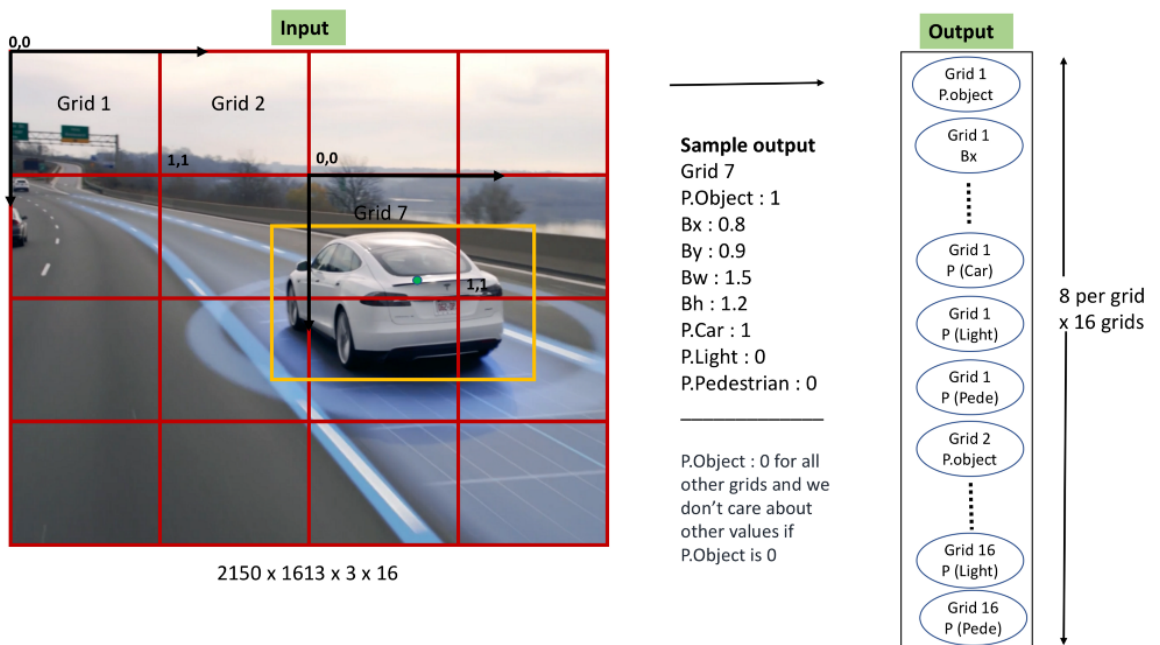


Source: Adapted from [43].  
Figure 3.13: Mask R-CNN.

### 3.2.6 You Only Look Once

You Only Look Once (YOLO) proposed by Joseph Redmon [44] divides the image into a grid of  $S \times S$  size of cells. For each cell predicts bounding boxes and class probabilities.

Figure 3.14 shows a YOLO grid.

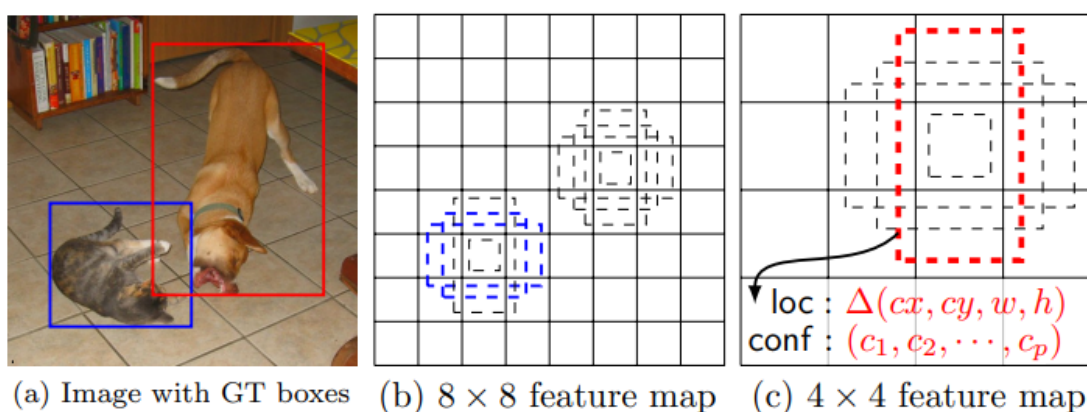


Source: Adapted from [44].  
Figure 3.14: YOLO grid.

This algorithm is faster than previous mentioned R-CNN based algorithms but does not allow us to create a mask around the object. Mask R-CNN is currently the state of the art when we need masks.

### 3.2.7 Single Shot Multibox Detection

Single Shot Multibox Detection (SSD) proposed by Wei Liu [45]. This algorithm only needs to take one single shot to detect multiple object within the image and is much faster compared with two-shot RPN-based approaches. For training it only needs an input image and ground truth boxes for each object. Using convolutions, it evaluate a small set of default boxes of different aspect ratios at each location in several feature maps with different scales. Figure 3.15 shows an SSD with different feature map sizes. This algorithm is faster then YOLOv1 and more accurate.



Source: Adapted from [45].

Figure 3.15: SSD.

## 3.3 Discussion

Initially, the YOLO algorithm was considered for geometry detection as it is considered the state of the art algorithm with regard to the speed of detection. But not allowing the contours of geometry led to the choice of Mask R-CNN.

For text recognition, it became clear that the way forward would be to use LSTM, but for text detection, several approaches may be possible. In addition to the specialized algorithms analyzed, another option will also be to use a Mask R-CNN for this work.



## Chapter 4

# Processing technical drawings using deep learning

---

4.1	Computational tools . . . . .	58
4.2	Processing pipeline Overview . . . . .	59
4.3	Image preprocessing . . . . .	60
4.4	Text detection . . . . .	63
4.5	Feature Control Frame, Datum and Text detection . . . . .	65
4.6	Text recognition . . . . .	68
4.7	Shape detection . . . . .	69
4.8	Text and shape relationship . . . . .	70
4.9	Results . . . . .	72

---

This chapter describes the tools and algorithms used to solve the proposed problem. The input for the processing pipeline is an image obtained from either a vectorial Portable Document Format (PDF) file, a scanned image, or an image taken by camera. Considering the last two cases, in which the input image is obtained via camera or scanner, the image quality and illumination may present a level of variation, therefore, employing image processing techniques can assist as a preprocessing step to improve the quality of the input image.

A Computer-Aided Design (CAD) file containing the drawing of the part can be indicated, and in this case the drawing can be projected from various perspectives to assist the algorithm understanding the shapes present in the image.

Concerning the result of the proposed methodology, outputs should be a list of shapes and related information.

## 4.1 Computational tools

During the experiments here presented, several computational tools were used. The main ones used were:

- **Python:** is an interpreted, high level and general-purpose language. The first release is in 1991 and is currently one of the most used, especially in the areas of machine learning.[46]
- **Jupyter notebooks:** it is an interactive interpreter with support for various languages but is especially focused on python. Google Colab is based on Jupyter notebook and has Graphics Processing Unit (GPU) and Tensor Processing Unit (TPU) that can be used for free, but has a run-time limitation of 12 hours.[47]
- **OpenCV:** is a software library with functions that target real-time computer vision. It is originally developed by Intel. It is cross-platform and free to use under the open-source BSD license. OpenCV integrates with deep learning frameworks like TensorFlow, Torch, and Caffe.[48]
- **TensorFlow:** TensorFlow is a free and open source software library and was developed by Google for internal use. In 2015 it was public released. It is also used for machine learning applications such as neural networks.[49]
- **Keras:** is an open-source neural network library written in Python. It is capable of running on top of TensorFlow and other Machine learning engines. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.[50]
- **Tesseract:** is an optical character recognition engine. The development was started by Hewlett-Packard in 1985. In 2005 was released as open source. The development is also sponsored by Google since 2006. The latest version adds Long Short-Term Memory (LSTM) to the engine.[51]

## 4.2 Processing pipeline Overview

Figure 4.1 shows the processing pipeline for this work. The image is taken from a PDF file, scanned image or taken from a camera and then it is sent to an image preprocessing module that corrects the image perspective and binarizes the pixels values using the adaptive Gaussian thresholding algorithm. The resulting image is sent to the following modules.

The shape detection module and text detection modules can be done in parallel. The detected text areas that match must be merged and sent to the text recognition module. The final module receives the detected shapes and recognized text and tries to match them, through proximity to the text, arrow or image registration.

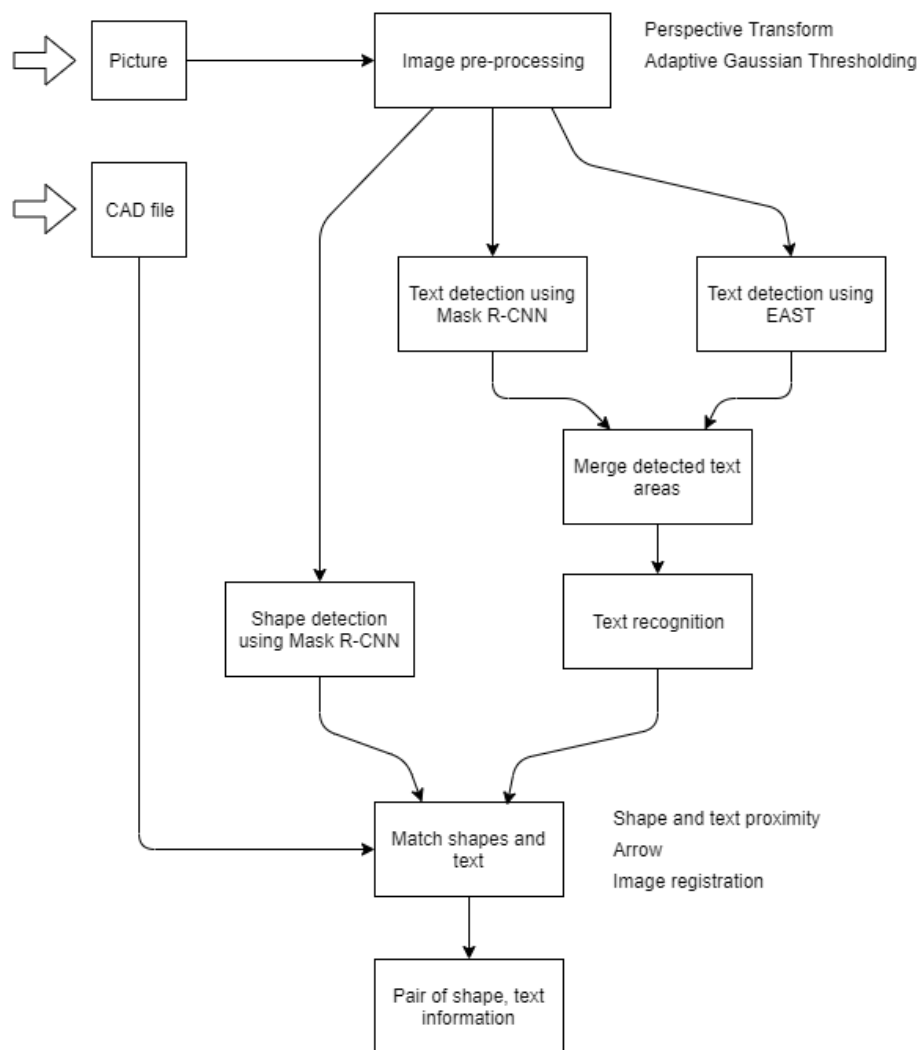


Figure 4.1: Processing pipeline diagram.

## 4.3 Image preprocessing

In this section the algorithms used for image preprocessing are described. In the scenario of the image being obtained through the rasterization of a vector image, these algorithms may not be necessary.

### 4.3.1 Color to grayscale conversion

Usually, the image processing is made with gray scale images when color is not important. In this pipeline this step is mandatory.

### 4.3.2 Perspective Transform

The perspective correction only needs to be made if the image is taken via camera. One way to correct this is to employ Affine Transformation, in which requires 4 points. These points can be annotated manually or automatically determined using a bounding box algorithm such as minimum bounding rectangle.

### 4.3.3 Adaptive Gaussian Thresholding

An image taken from a camera or a scanned image can have some noise caused by lighting and shadows. A rasterized image may contain some intermediate gray pixels that should be white or black. In order to remove noise from the image an algorithm was used that allows the pixels of an image to be converted from a gray scale (0-255) to only 0 or 1. One of the better methods for this is a binary threshold. There are several binary threshold methods but adaptive gaussian thresholding should be the best. In this algorithm, the threshold value is the weighted sum of neighborhood value. The weights are a Gaussian window of some size.

The Figure 4.2 shows a rasterized PDF image that was binarized using the adaptive Gaussian thresholding. The resulting image is improved.

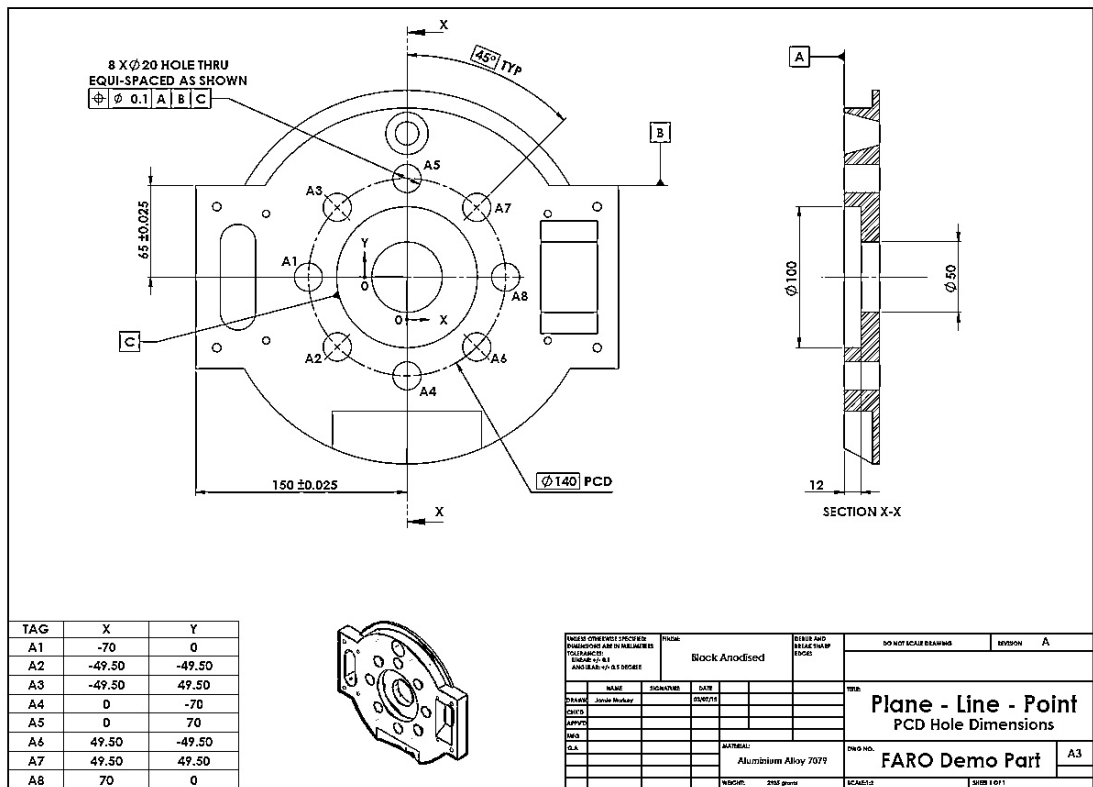


Figure 4.2: Result of the original image binarized.

Figure 4.3 shows a fragment of an image taken from a photo that was binarized using global thresholding, adaptive mean thresholding, and adaptive Gaussian thresholding. In this case the result is not as good as that obtained from a rasterized image. The resulting images contain noise and missing information, but the adaptive gaussian thresholding has the best result.

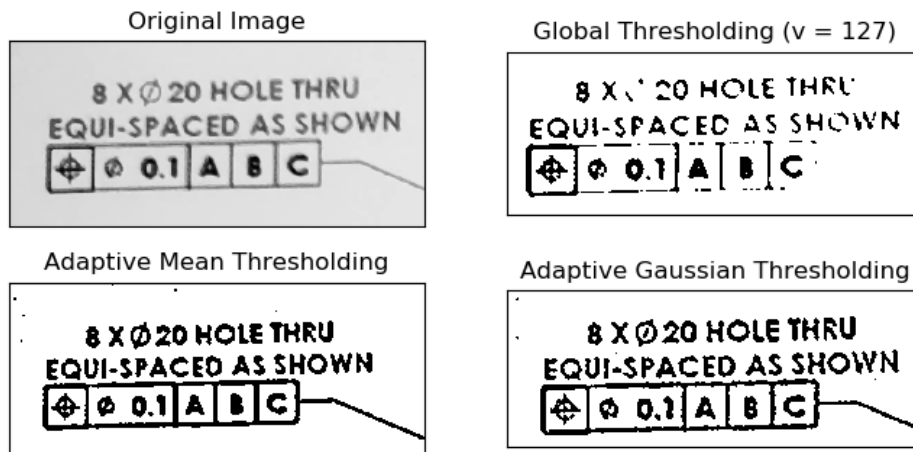


Figure 4.3: Fragment of original image and binarization using global thresholding, adaptive mean thresholding, and adaptive Gaussian thresholding.

Figure 4.4 shows the same fragment of the image but now using the Otsu's thresholding[52] directly from the original image and with two different preprocessing algorithms, median blurring, and Gaussian blurring. The result is not so good comparatively with Adaptive Gaussian Thresholding.

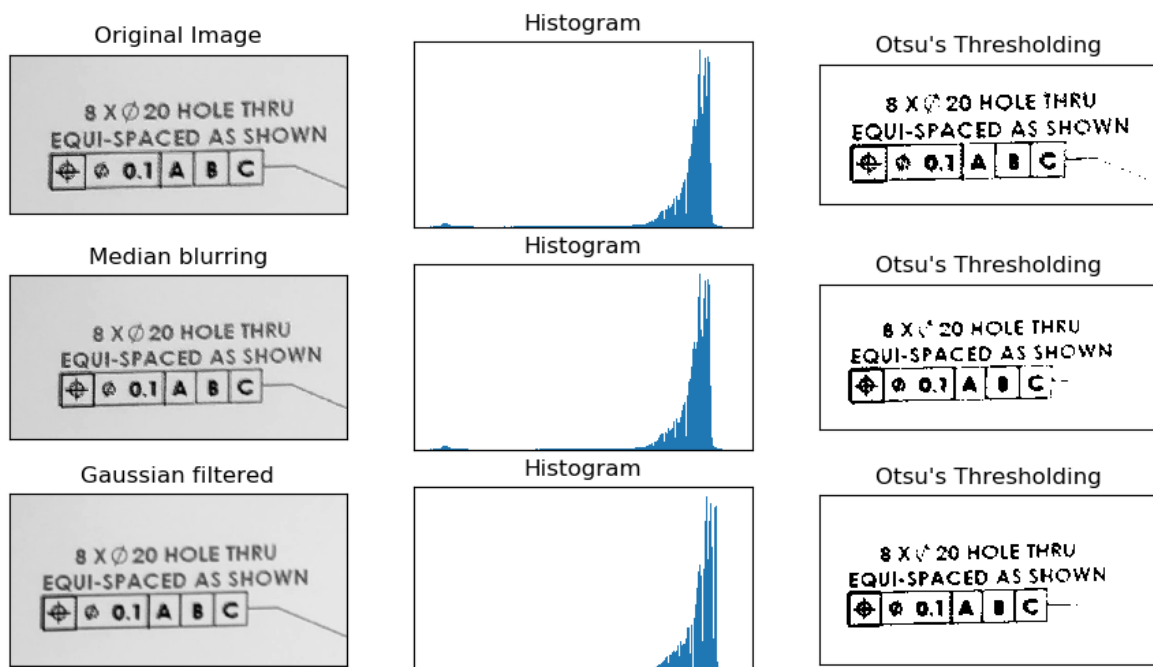


Figure 4.4: Fragment of original image and binarization using Otsu's thresholding and a preprocessing using median blurring and gaussian blurring.

### 4.4 Text detection

In the first attempt tesseract[51] was used to detect and recognize the text in the entire image, but the results were not good.

In the second attempt the EAST algorithm was used to do only detection of text areas. Figure 4.5 shows the detected text using the EAST algorithm. The detected text includes horizontal and non-horizontal text, and was able to detect special symbols. However, not all text is detected.

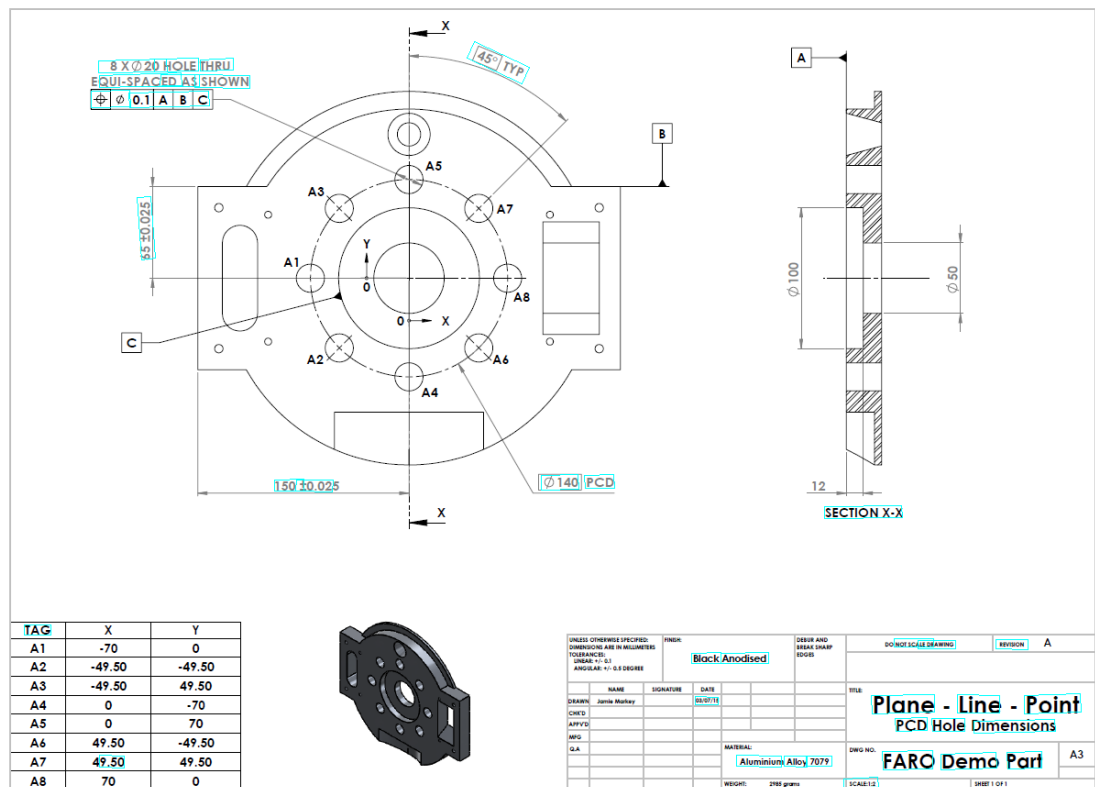


Figure 4.5: Text detection using EAST algorithm.

It has also been tried to use a computer vision algorithm to detect Geometric Dimensioning and Tolerancing (GD&T) and Datum information box lines. Figure 4.7 shows the detected lines with the Hough lines transform algorithm. Hough Transform is a method used in image processing to detect any shapes, if that shape can be represented in a mathematical form and can even detect if the shape is broken or distorted a bit.[53] Figure 4.6 shows an example of houghlines. The points are converted from parametric form to perpendicular distance from origin to the line, and is the

angle formed by this perpendicular line and horizontal axis measured in counter-clockwise.

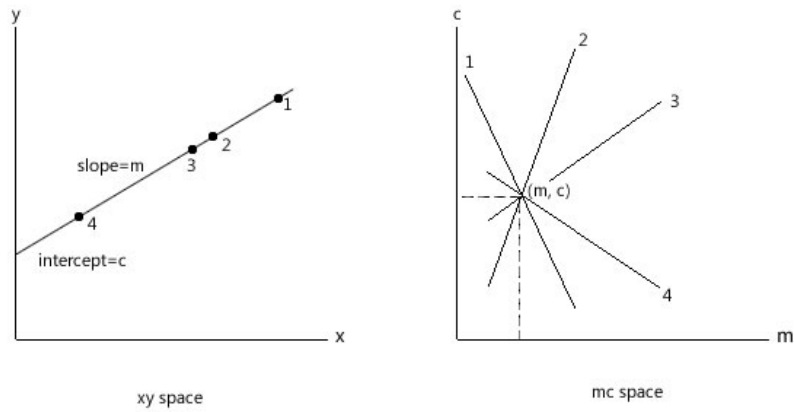


Figure 4.6: An example of Houghlines.<sup>1</sup>

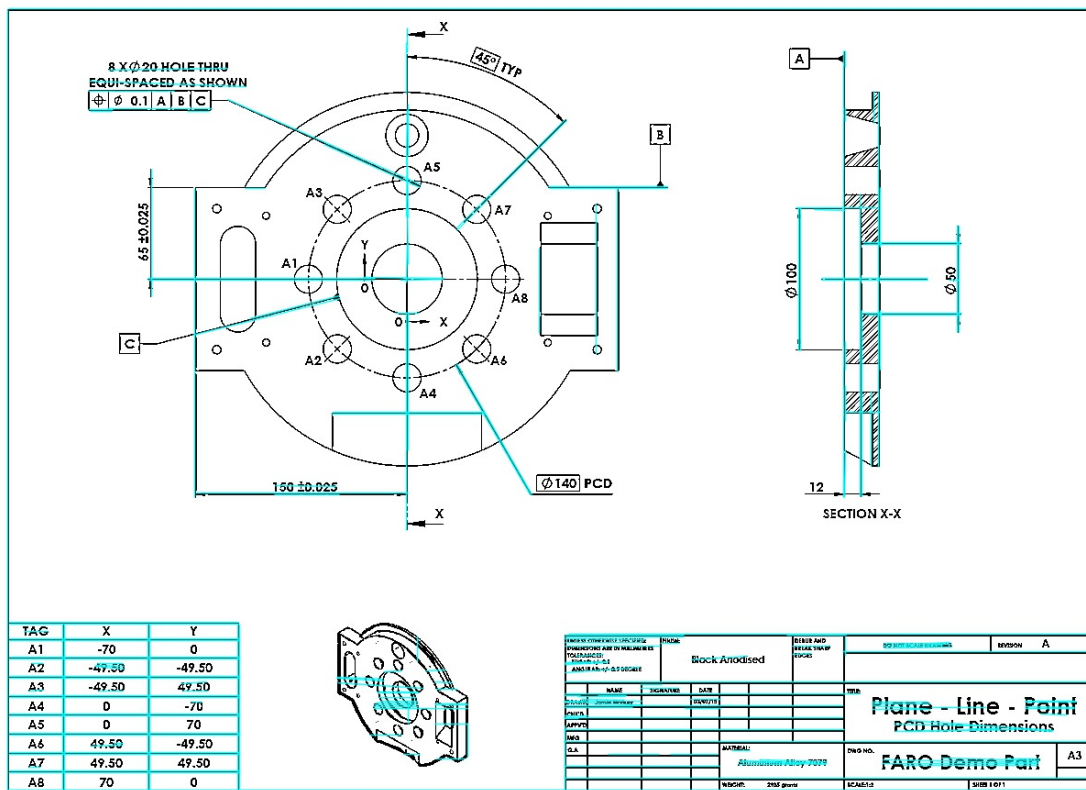


Figure 4.7: Detecting lines with Hough lines algorithm.

Finally, Mask R-CNN trained and tested to detect text. Usually, the text is inside a Feature

<sup>1</sup>Image from: <http://aishack.in/tutorials/hough-transform-basics/>



Control Frame (FCF) or datum box, so the training images must contain text inside FCF, datum box, but also outside.

## 4.5 Feature Control Frame, Datum and Text detection

It is difficult and inadequate to obtain real technical images since they are usually related to industrial secrets. To get around this limitation I decided to generate images containing the text to contemplate the different uses and forms.

Three classes have been defined for this: **fcf**, **datum**, and **text**. Then, a python script was made to generate images with randomly placed text elements inside each image and with different information. Shapes were also placed at random positions but without annotations for that. Figure 4.8 shows an example of a generated image. The python script references the pycairo library to an easy drawing of shapes, but some difficulties were encountered and it was not possible to timely generate images with rotating text and shapes with the corresponding annotations. Each image has a resolution of 512x512 pixels and contains shapes and text. For each texts, image was generated containing the mask of the region where the text is located.

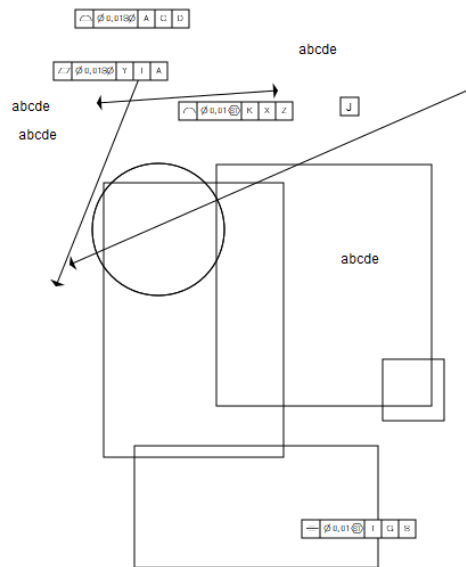


Figure 4.8: Example of generated image for training text detection.

The generated images were separated into 3 groups: train, validation, and test. The train images and annotations serve to train the Mask R-CNN. The validation set of images are used

during the training process to evaluate the approximate error. Figure 4.9 shows the corresponding generated images serving as a mask for each text present in the original image. 8000 images were generated for training, 1000 for validation, and 500 for test.

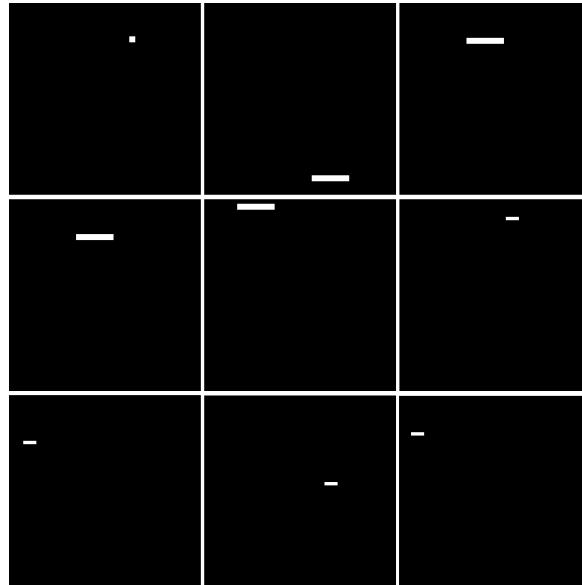


Figure 4.9: Example of some generated mask images for the previous image.

An open-source library implemented by Abdulla [54] was used. This library implements Mask Region-based Convolutional Neural Networks (Mask R-CNN) with small modifications. It is based on Feature Pyramid Network (FPN)[55] and a ResNet101 backbone[56]. Figure 4.10 shows an illustration of an FPN.

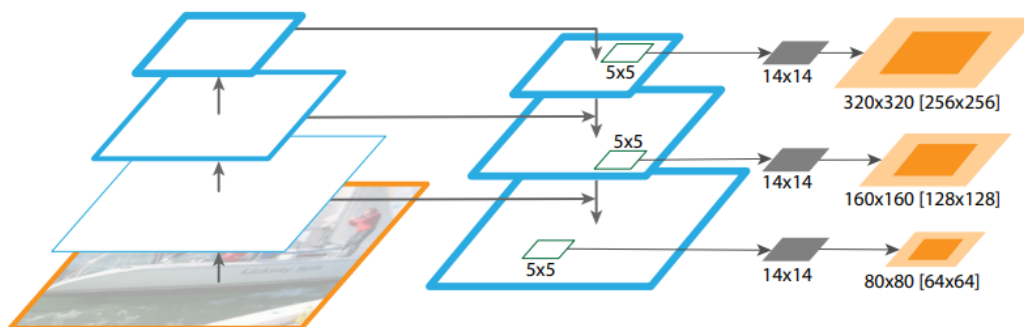


Figure 4.10: An illustration of an FPN.

The annotations were generated automatically using a utility to convert image masks to annotations in format identical to the COCO[57] format. It is in json format and for each image contains a bounding box and a polygon delimiting the object. The COCO format is the most used

for this task. It is sponsored by big companies like Microsoft and Facebook. The COCO project also includes a dataset with:

- Object segmentation
- 330k images (200k of them labeled)
- 1.5 million of object instance

The training process occurred in a total of 75 Epoch and 400 iterations per Epoch. An Epoch is an iteration over the entire dataset.

Figure 4.12 shows the text detection using the trained Mask R-CNN. Near the object is a label that indicates the probability value. Each detected object is contained inside a horizontal rectangle but is also filled with a mask. The mask allows a better localization and orientation of an object. In the generated images all text is horizontal but in real images some text can be non-horizontal and the mask allows to extract the text part.

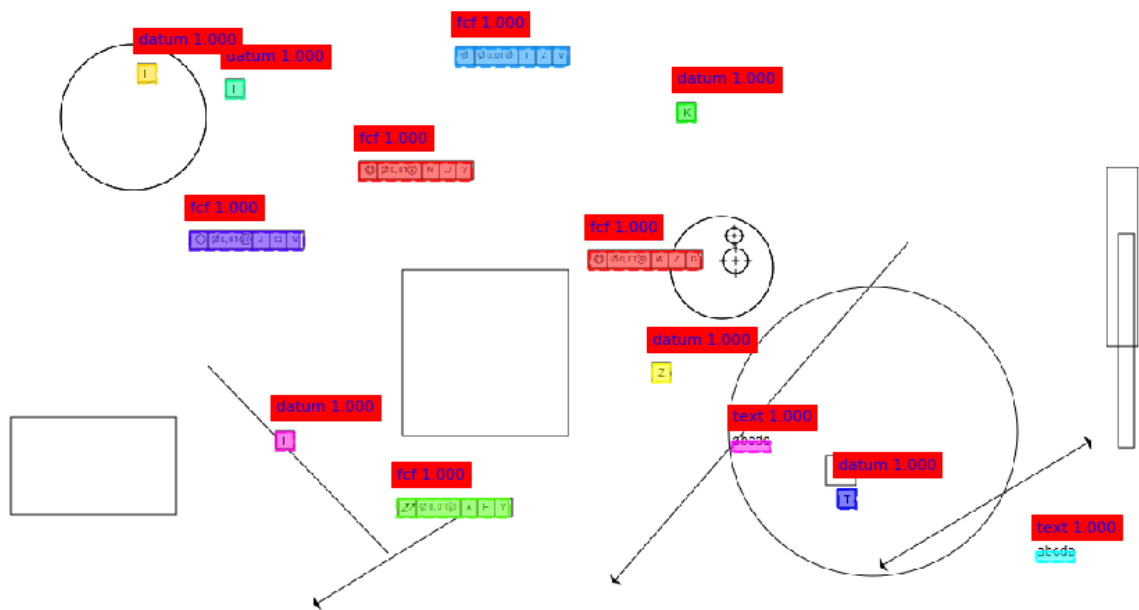


Figure 4.11: Text detection using the trained Mask R-CNN.

Figure 4.12 shows the text detection using EAST algorithm in generated images. We can verify that the algorithm cannot detect text when it is just one character. The FCF are also partially detected.

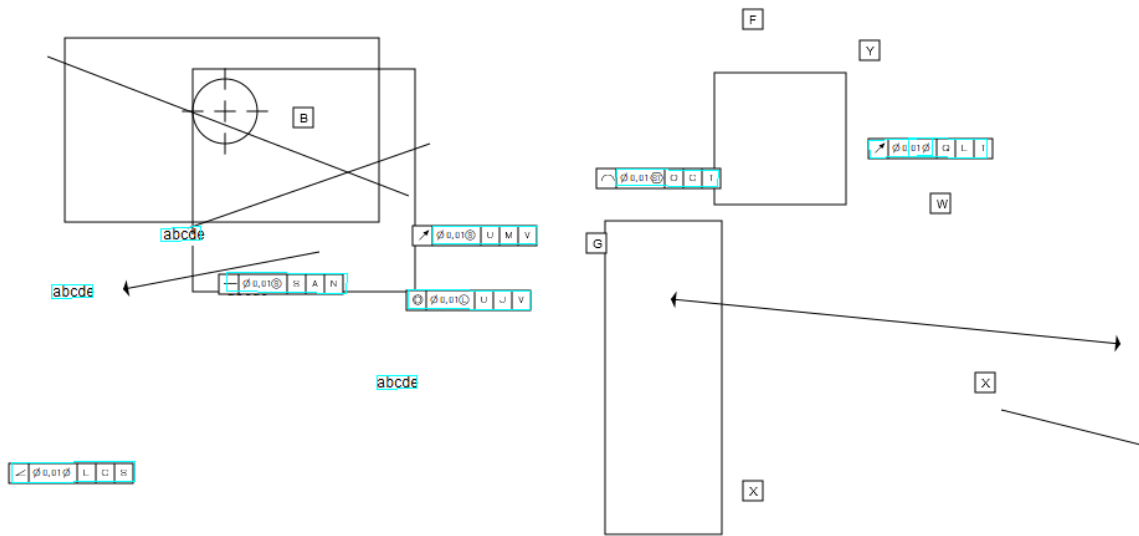


Figure 4.12: Text detection in the generated images using EAST algorithm.

Detection using Mask R-CNN seems to have better results and has the advantage that the detection can classify the text type.

## 4.6 Text recognition

For text recognition was used the latest version of Tesseract. The tesseract 4.0 implements LSTM and is trained with multiple fonts, but does not recognize GD&T symbols, so is necessary to train with images containing these symbols. For generating these images a python script was used to generate symbols with various font sizes. The resulting model must be used in combination with existing models to allow recognition of both symbols and chars. The output can be in the XML format and includes the recognized text and bounding boxes of each detected phase. We are interested only in the recognized text because the image send to tesseract is a text area detected earlier. Even so, there are some issues related to the proximity of lines that sometimes the character is badly recognized. A possible solution is detection lines and subtracting from the original image. This can be done using morphological operations. But in this line removal, parts of the text may also be deleted. Figure 4.13 shows on the left side original areas containing text and on the right side the result image after applying a morphological operation. In one of the FCF,

the vertical lines were not removed and in another FCF the symbol of GD&T was unrecognizable.

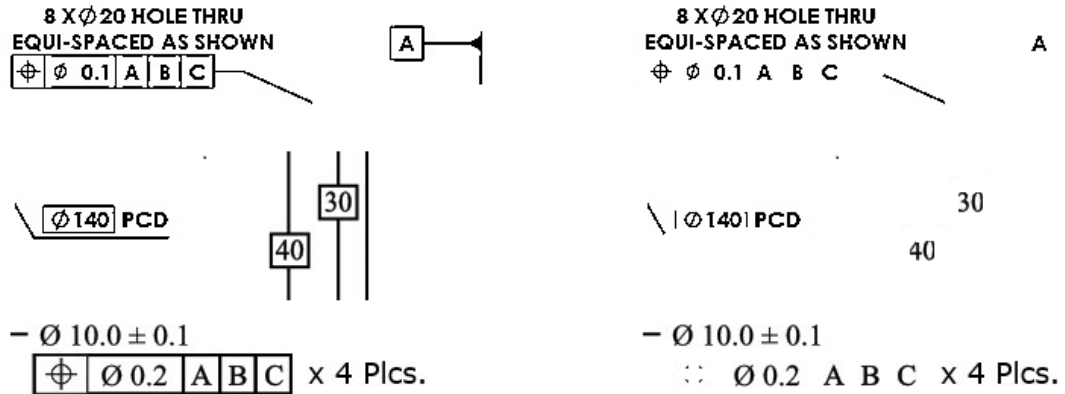


Figure 4.13: Result image after applying a morphological operation to remove horizontal and vertical lines.

Figure 4.14 shows the text recognition in an FCF after lines removal. On the right side of image is a text output of the recognized text.



Figure 4.14: Text recognition in an FCF after lines removal.

## 4.7 Shape detection

For shape detection, several algorithms were analyzed. I started with You Only Look Once (YOLO). Currently, there are several implementations of this algorithm. The official implementation is a C++ library that can run on Central Processing Unit (CPU) and GPU. But before training, we need images with the shapes. As already mentioned, it is difficult to get images of real drawings. Due to this limitation, images containing these shapes have also been generated. But YOLO does not allow us to detect the mask along the shape and in some situations this is an important information. It also has other limitations like detecting small objects.

The best algorithm to solve this problem is again an Mask R-CNN. Several other classes are defined such as rectangles, squares, triangles, lines, round slots. A python script was used to generate the images to train the Mask R-CNN. Figure 4.15 shows a generated image and generated

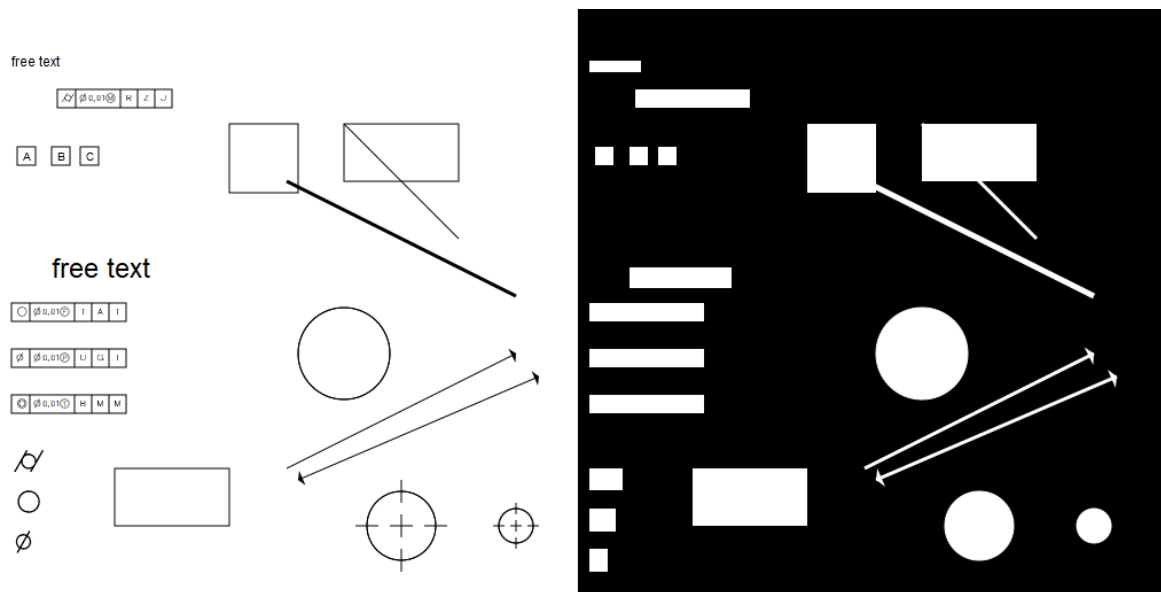


Figure 4.15: Example of generated image for training and image with masks.

image with a mask for each class present in the original image. The masks outline each shape. As with the text, the annotations were generated automatically to MS COCO[57] format. 8000 images were generated for training, 1000 for validation and 500 for test. It is not necessary to start training from scratch because some image features are so common that are already present in the pre-trained models. The training process started with weights of pre-trained COCO[57] dataset. The training process occurred in 75 iterations or epochs.

Figure 4.16 shows the detection using the trained Mask R-CNN. There is a label above each detected object that indicates the class and the probability of belonging to this class. Each detected object is contained inside a horizontal rectangle but is also filled with a mask. The mask allows a better localization and orientation of an object.

## 4.8 Text and shape relationship

In the shapes training set arrows and bidirectional arrows are included. These can help us relate a shape to the corresponding text. Figure 4.17 shows the detection of arrows and their masks. The mask is particularly important in these two shapes. Using the shape of the mask, it is possible to determine the beginning and end of the arrow and its direction. With this information we can relate the text information to the shape that is closest to each end.

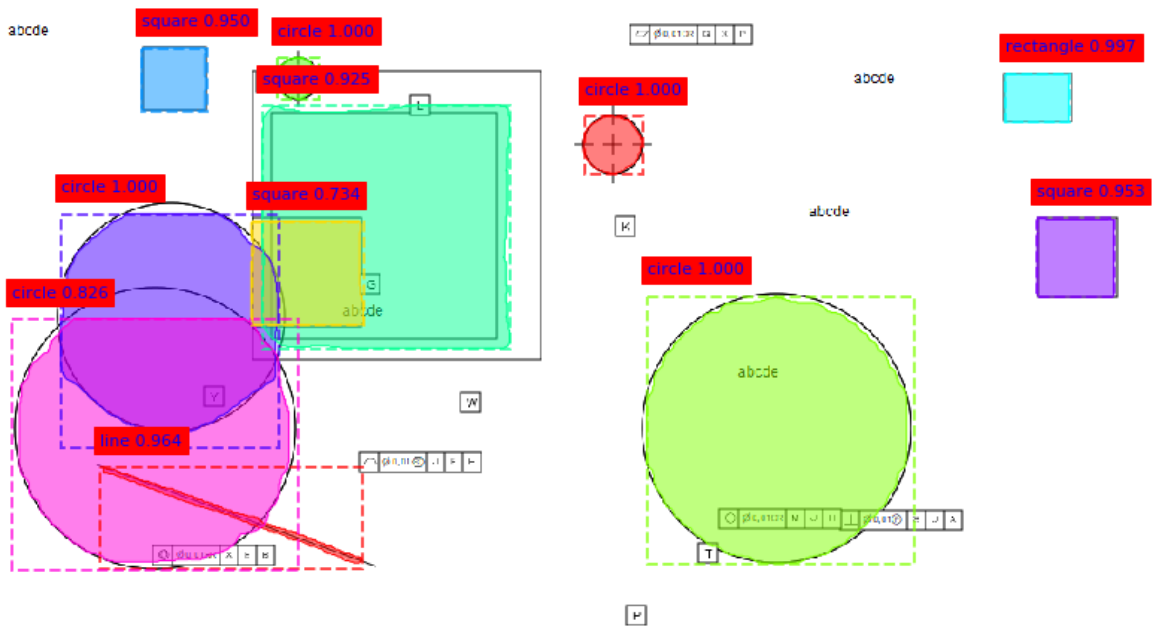


Figure 4.16: Shapes detection using the trained Mask R-CNN.

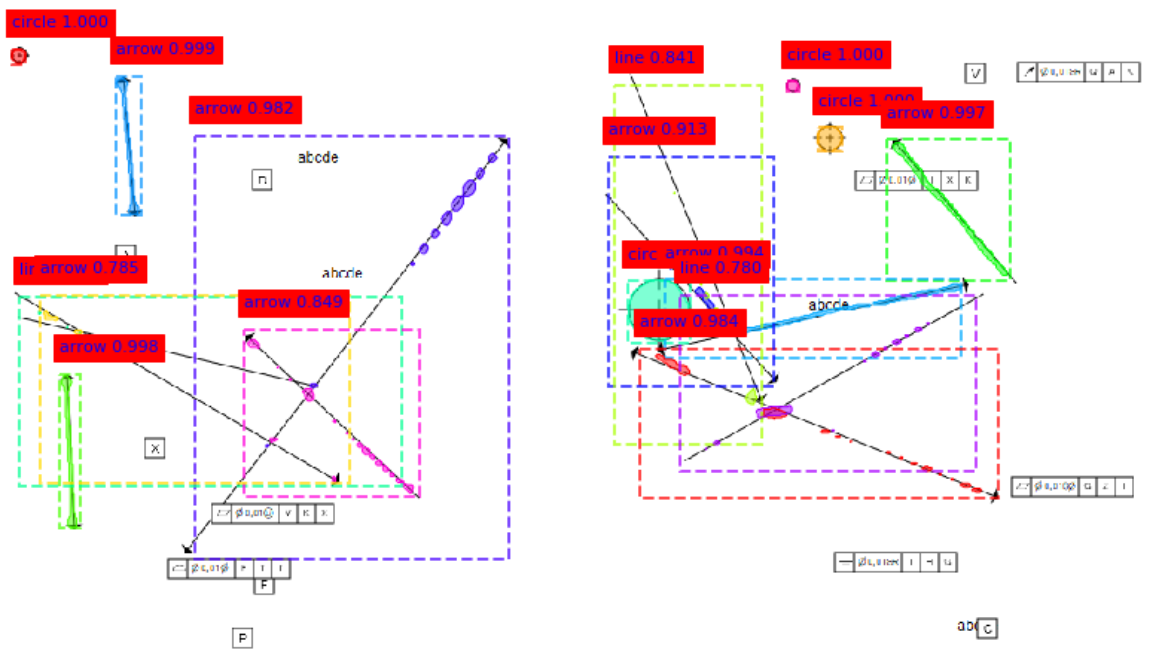


Figure 4.17: Arrows detection using the trained Mask R-CNN.

## 4.9 Results

During the training process were recorded several loss values and allows to verify how the training is going and if it is no longer worth training. They are:

- **rpn class loss:** RPN anchor classifier loss.
- **rpn bbox loss:** RPN bounding box loss graph.
- **mrcnn class loss:** loss for the classifier head of Mask R-CNN.
- **mrcnn bbox loss:** loss for Mask R-CNN bounding box refinement.
- **mrcnn mask loss:** mask binary cross-entropy loss for the masks head.

Figure A.1 shows the chart comparing the training and validation loss along of the 75 epochs of the training the text model. The loss displayed on the chart is the sum of rpn class loss, rpn bbox loss, mrcnn class loss, mrcnn bbox loss and mrcnn mask loss. This training used transfer learning starting from COCO[57] weights. Loss slightly changes after epoch 59, so the 75 epochs were enough.

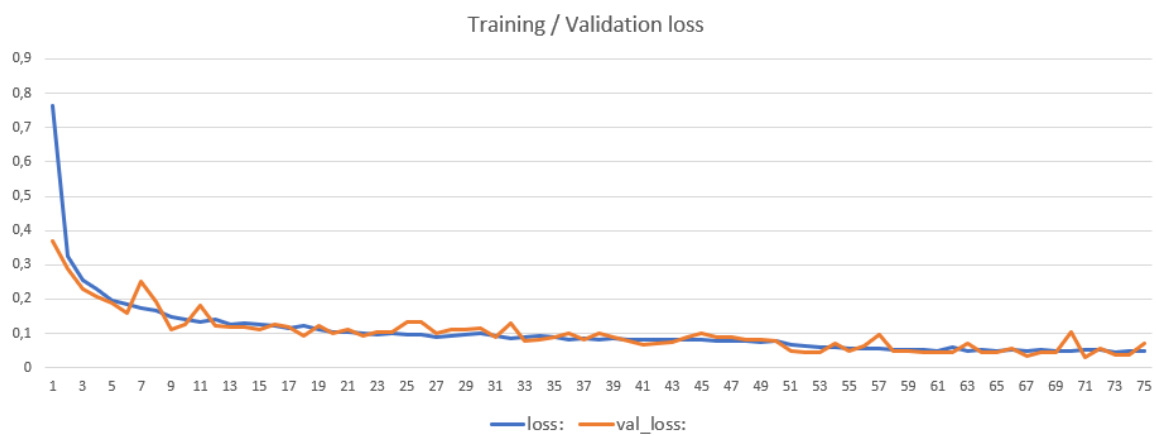


Figure 4.18: Training/Validation loss of text detection.

Figure 4.19 shows the training/validation loss during the training of 75 epochs and annotations for shapes and text. The loss in this training did not go down as much as in training where only text annotations were used. The drop in loss is also not as constant as when trained with text annotations only.



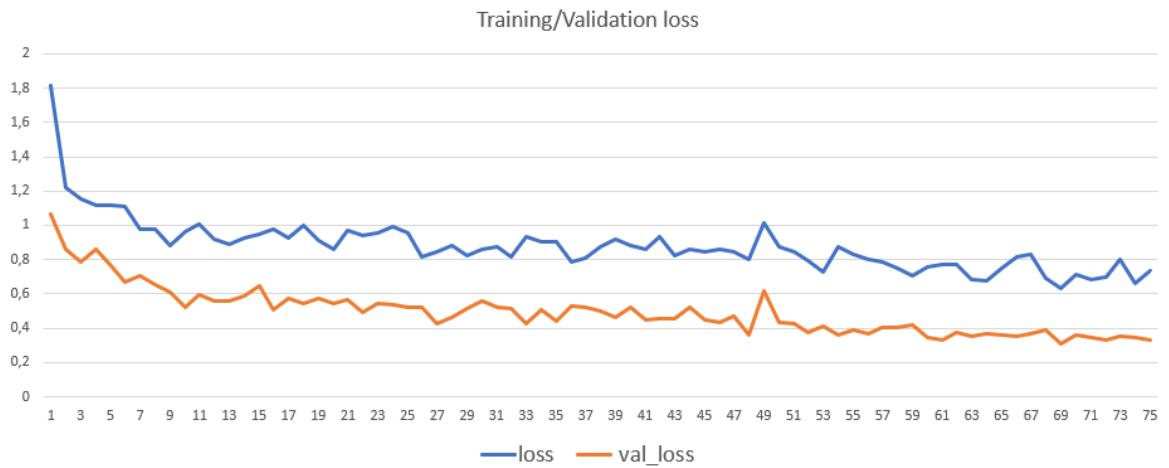


Figure 4.19: Training/Validation loss of shapes and text detection.

The most common metric for evaluating object detection is the Intersection over Union (IoU), also referred to as the Jaccard index. [58][59] It measures the percent of overlap between the object and the detected box. Figure 4.20 shows an illustration of precision, recall, and IoU. In the case of instance segmentation, instead of the bounding box is used the IoU of the object mask.

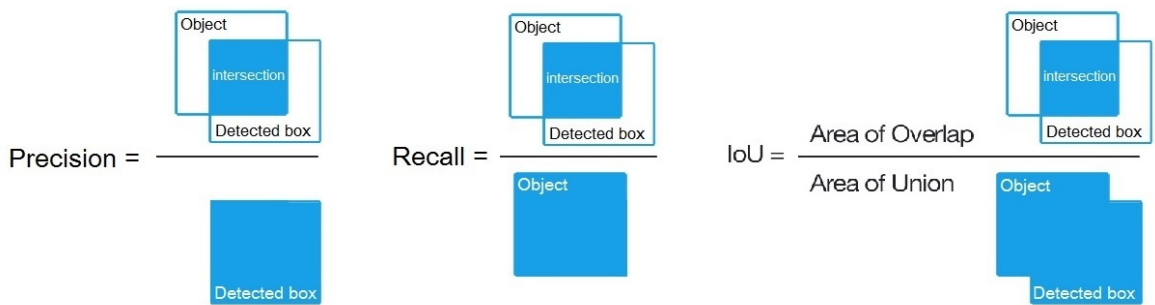
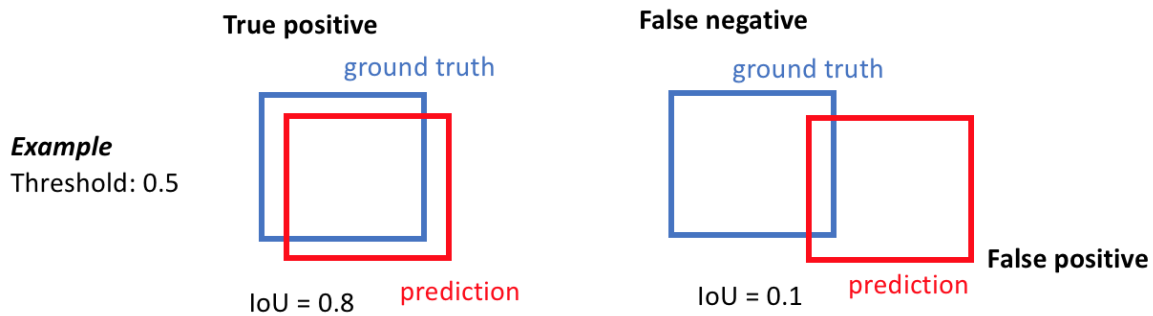


Figure 4.20: An Illustration of precision, recall and IoU.<sup>2</sup>

As a reference, Figure 4.21 shows an example of IoU scores with different overlaps. In this example, only the first image is considered as a true positive.

<sup>2</sup>Image from: <https://github.com/stereolabs/zed-yolo/tree/master/libdarknet>

Figure 4.21: Example of IoU scores.<sup>3</sup>

- A **True Positive (TP)** is observed when the prediction-target pair has an IoU score which exceeds the predefined threshold.
- A **False Positive (FP)** indicates a predicted object mask had no associated ground truth object mask.
- A **False Negative (FN)** indicates a ground truth object mask had no associated predicted object mask.

One of metric to evaluate the Mask R-CNN is mean Average Precision (mAP). [60]

Table 4.1 shows the precision of each text class. The fcf and datum classes has high precision.

Table 4.1: mAP of text detection at IoU=0.5

Class	Total objects	Precision
fcf	437	0.998
datum	497	0.99
text	438	0.75
<b>Average</b>		<b>0.993</b>

Table 4.2 shows the precision of each shape class. Circles have good precision. The squares and rectangles have an acceptable precision, but lines and arrows have low precision. From analyzed images, the detection of lines and arrows is sometimes misidentified due to their similarity. Other times the line or arrow is identified on one edge of a rectangle or square.

<sup>3</sup>Image from: <https://www.jeremyjordan.me/evaluating-image-segmentation-models/>

Table 4.2: mAP of shapes detection at IoU=0.5

Class	Total objects	Precision
circle	434	0.98
square	365	0.82
rectangle	325	0.75
line	336	0.31
arrow	439	0.24
<b>Average</b>		<b>0.62</b>

Table 4.3 shows the precision of a Mask R-CNN trained with all classes. The precision were calculated with iou of 0.5, 0.75 and 0.95.

Table 4.3: mAP tested with 1000 images

Class	Images	IoU=0.5	IoU=0.75	IoU=0.95
datum	962	0.99	0.98	0.56
fcf	943	0.99	0.99	0.89
text	940	0.97	0.84	0.17
square	714	0.80	0.76	0.29
circle	863	0.99	0.97	0.56
line	665	0.48	0.29	0.21
rectangle	685	0.74	0.66	0.002
arrow	894	0.65	0.14	0.03
<b>Average</b>		<b>0.84</b>	<b>0.72</b>	<b>0.36</b>

Figure 4.22 shows a generated image for testing and the detections made by the Mask R-CNN.

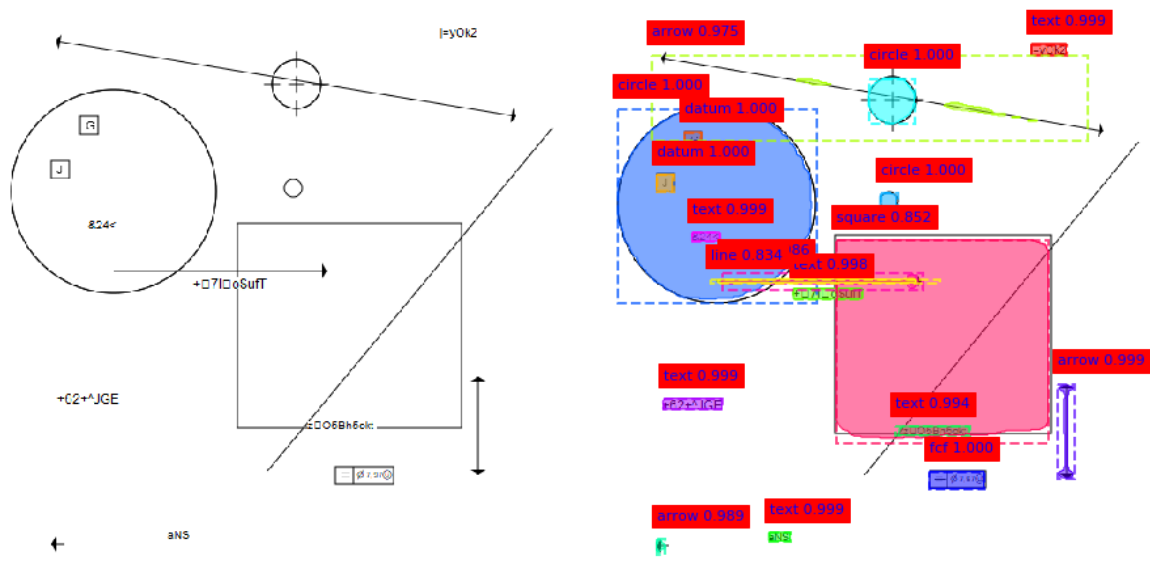


Figure 4.22: Detection on sample image.

Figure 4.23 shows the confusion matrix of the same tested image. We can verify the detection of squares as actually rectangular and the line that is an arrow.

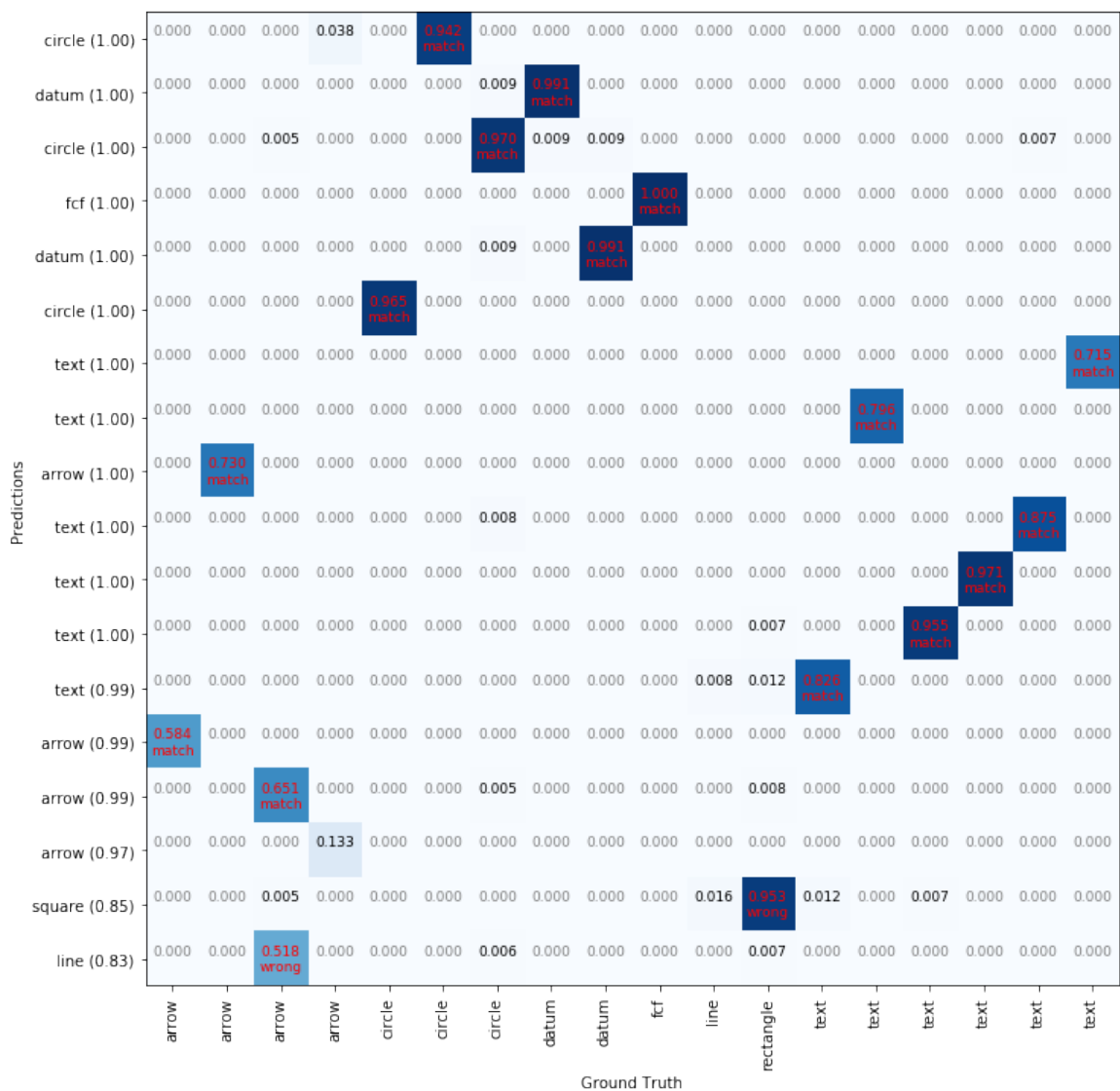


Figure 4.23: Confusion matrix of one test image.

Figure 4.24 shows the confusion matrix of another image. We can verify the detection of an arrow but is a line. After analyzing some images we can verify that the lines are often detected as arrows and the squares as rectangles. These four classes seem to have detection problems due to their similarity. A possible improvement will be to join the rectangle and square classes. On the other hand, the arrows must continue to be distinguished from the lines.



Figure 4.24: Confusion matrix.

Table 4.4 shows the mAP with various IoU starting at 0.5 and ending at 1 with 0.05 increments. These values are calculated with various sets of images.

Table 4.4: Precision, Recall and F1 Score at different IoU

IoU	Precision	Recall	F1 Score
0.50	0.90	0.96	0.93
0.55	0.89	0.96	0.92
0.60	0.87	0.95	0.91
0.65	0.84	0.94	0.89
0.70	0.81	0.93	0.86
0.75	0.77	0.91	0.83
0.80	0.72	0.87	0.79
0.85	0.66	0.81	0.73
0.90	0.56	0.69	0.62
0.95	0.36	0.50	0.42
1.00	0.03	0.23	0.05

The model is trained with annotations like shown in Figure 4.25. The mask of each object to detect is strong along all the objects. This is an ideal scenario.

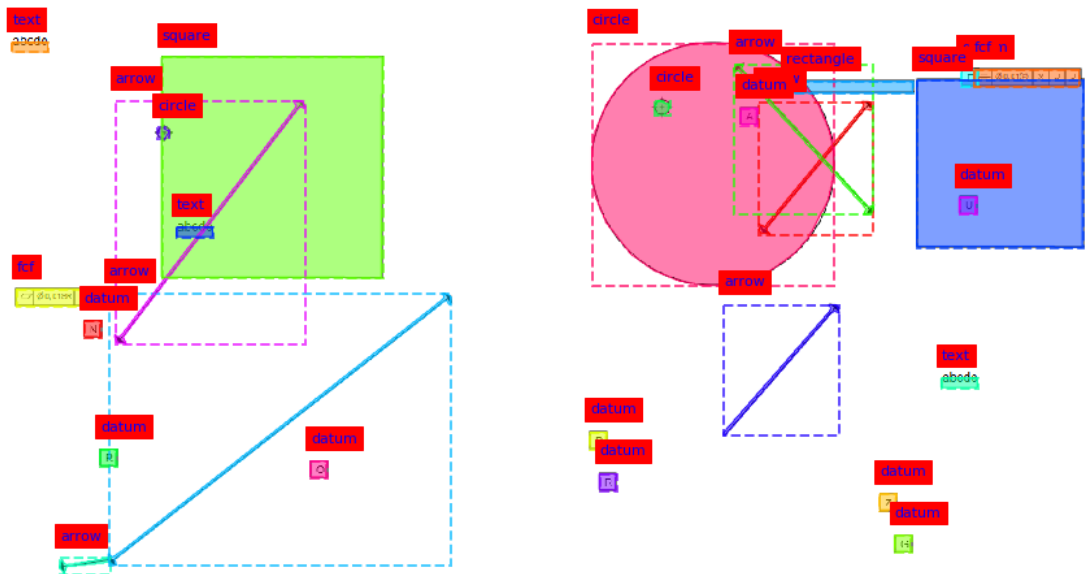


Figure 4.25: Image with original annotations.

Figure 4.26 shows the detection using the trained Mask R-CNN. Near the object is a label that indicates the IoU value. Each detected object is contained inside a horizontal rectangle that matches almost perfectly but is also filled with a mask that no longer corresponds so much with the optimal detection, but still allows a good localization and orientation of an object.

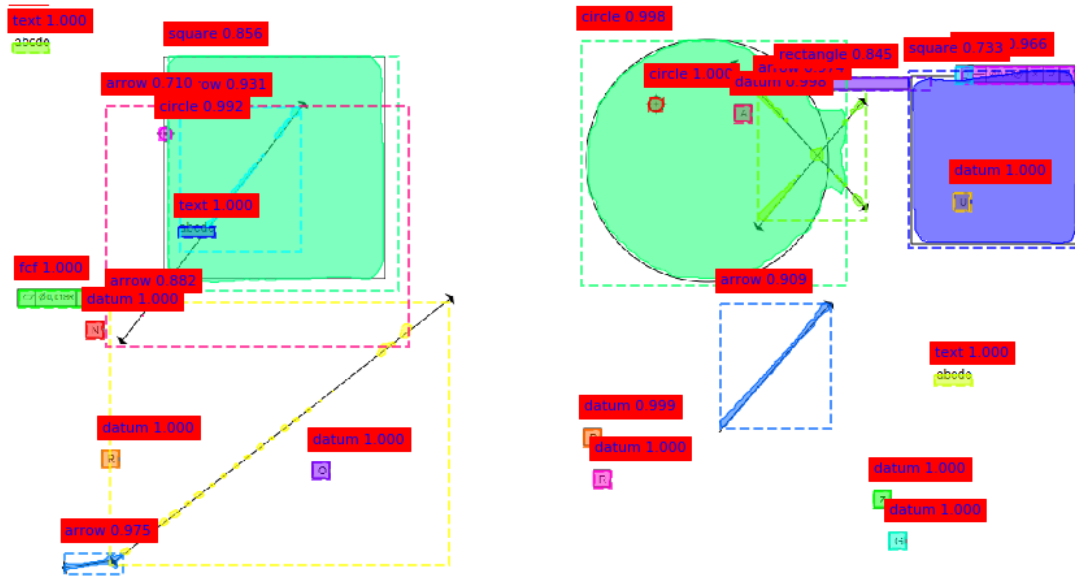


Figure 4.26: Detection using the trained Mask R-CNN.

To test the robustness of the detection model one of the test images was changed to contain some rotated shapes. The training and validation image set has no shape with rotation. Figure 4.27 shows on left the image and on the right the image with detection's made by the model of shapes. Seems to be some detection as lines in some parts of the shapes. It seems to have occurred due to the lack of rotating shapes and the model overfitted as lines.

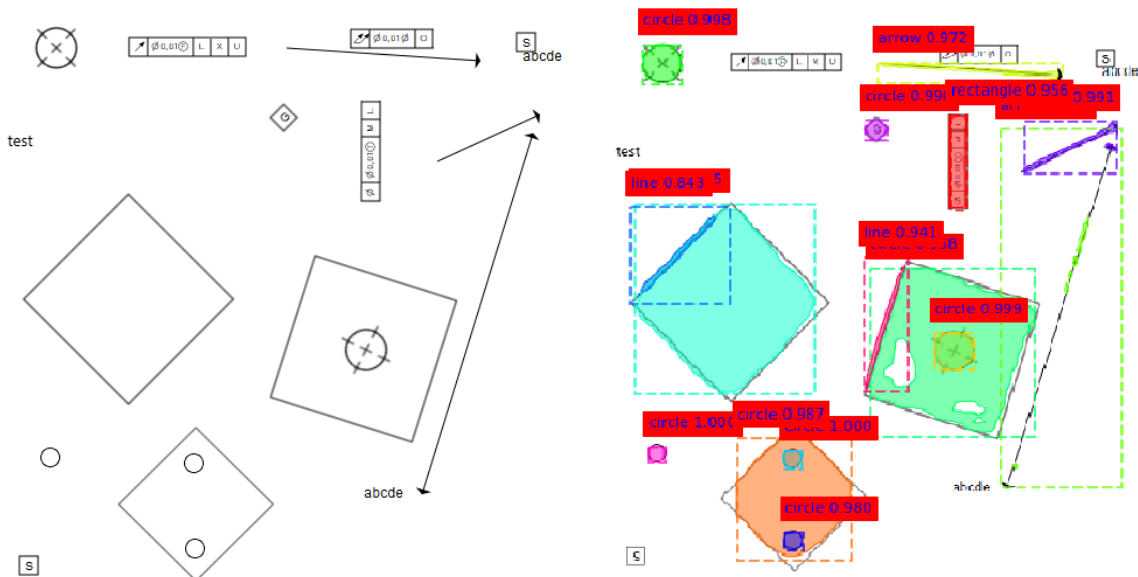


Figure 4.27: Detection of some rotated shapes (1).



Figure 4.28 contains a new shape called round slot and it is detected as a circle. It has also some manually placed circles inside other shapes and is detected correctly.

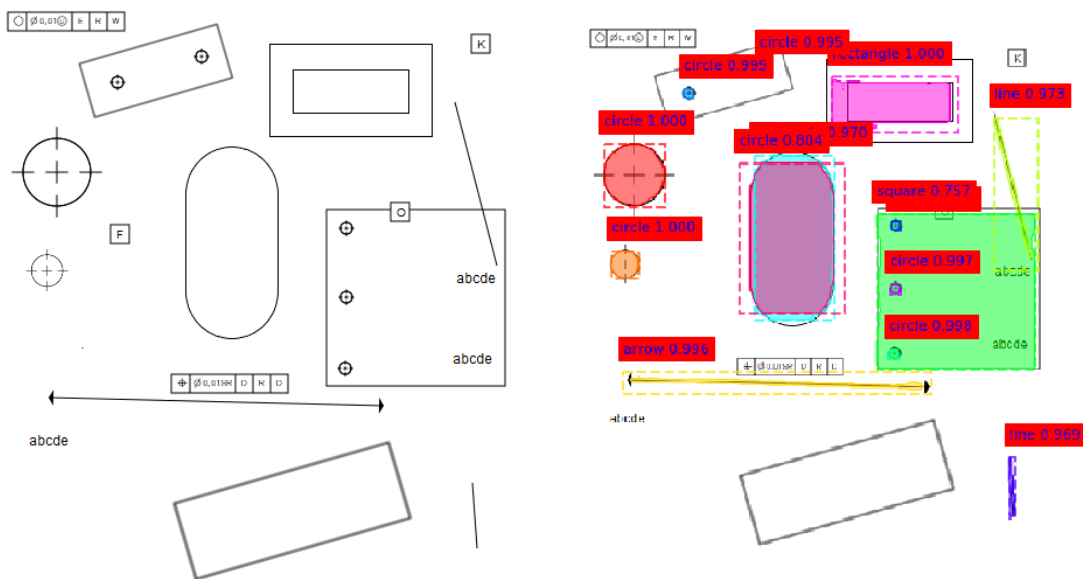


Figure 4.28: Detection of some rotated shapes (2).

Detecting shapes in generated images can be very different from detecting in real images as shown in Figure 4.29. We can verify that text detection is not good, but the text in the training set had little variation and no non-horizontal text was placed, so failed to generalize. Was able to detect shapes within shapes.

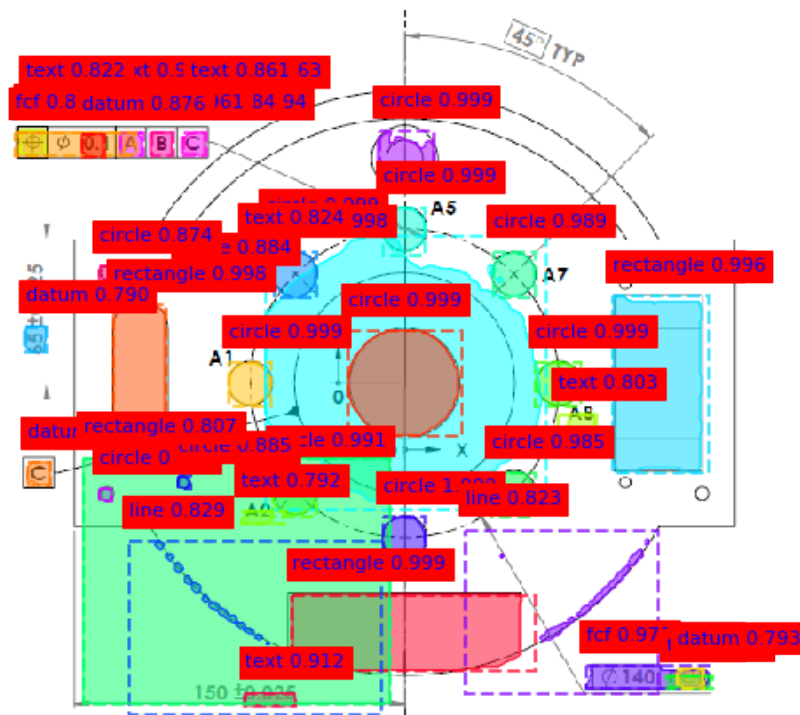


Figure 4.29: Detection of real world drawing.

The most commonly used way to measure text recognition accuracy is by using Levenshtein[61] distance also known as edit distance. Levenshtein distance is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single character edits (insertions, deletions or substitutions) required to change one word into the other.[62] This metric allows measuring the precision with small errors such as an incorrect character. We can also consider the accuracy of binary form for each sentence and just consider an exact match to be correct.

The precision of detection text outside boxes and without special characters is 0.501 for an exact match. But without line removal, precision rises to 0.519. But through data analysis, it was found that detection was not always better without removing lines. Table 4.5 shows the precision and recall, with line removal and without line removal.

Table 4.6 shows the precision and recall, without line removal for some texts and Table 4.7 shows with line removal.

Table 4.5: Precision and recall of text recognition with and without line removal

	Precision	Recall
<b>With line removal</b>	0.74	0.74
<b>Without line removal</b>	0.86	0.85

Table 4.6: Precision and recall of some text recognition without line removal

Text	Prediction	Edit Distance	Precision	Recall
7BqMtDgsM-	7BqMtDgsM-	0	1	1
7rfqv0	7rfqv0	0	1	1
mRI6,Je:,GH:	mRI6,Je:,GH:	0	1	1
U4IC	U4IC	0	1	1
93u-S.u3	93u-S.u3	0	1	1
jku:bm0MN.	jku:bmOMN.	1	0.9	0.9
cEa3H5r3j6	cEa3H5r3j6	0	1	1
nN	nN	0	1	1
zZZ	zZZ	0	1	1
GE	GE	0	1	1
XiwAnGgAlVx	XiwAnGgAl\x	1	0.91	0.91
KmxNnQ5ji.	KmenQSji.	3	0.78	0.7
CtWCt3	CtWCt3	0	1	1
sgYFg:fd9	ngFgÂfd9	3	0.75	0.67
hQrL	hQrL	0	1	1

Table 4.7: Precision and recall of some text recognition with line removal

Text	Prediction	Edit Distance	Precision	Recall
7BqMtDgsM-	7BqMtDgsM-	0	1	1
7rfqv0	ikwmu	6	0	0
mRI6,Je:,GH:	mRI6,Je:,GH:	0	1	1
U4IC	uaxmc	4	0	0
93u-S.u3	93u-S.u3	0	1	1
jku:bm0MN.	jku:bmOMN.	1	0.9	0.9
cEa3H5r3j6	cEa3H5r3j6	0	1	1
nN	nN	0	1	1
zZZ	i	3	0	0
GE	GE	0	1	1
XiwAnGgAlVx	XiwAnGgAlVx	1	0.91	0.91
KmxNnQ5ji.	KmxNnQÂ§ji.	1	0.9	0.9
CtWCt3	CtWCt3	0	1	1
sgYFg:fd9	sgYFg:fd9	0	1	1
hQrL	mkig	4	0	0

The precision of the exact matching of datum is 0.722 in 1000 test images. If we consider correct

also the cases where it failed because the letter is uppercase, lowercase or duplicate characters, then the precision increases to 0.95. Table 4.8 shows the most common wrong detections.

Table 4.8: Most common wrong detections of datum character

Text	Prediction
Z	Zz
C	c
W	WwW
X	Xx
V	Vv
W	Ww
W	w
I	
C	c
X	x
O	c
F	FE
W	Ww
P	p
P	Pp
O	c
C	c
V	Vv
O	oO

Table 4.9 shows the precision of GD&T symbols recognition and total images with that symbol in the dataset with 1000 of total images. Some of the symbols have very low accuracy. Figure 4.30 helps to explain why this happens. Some parts of the symbol are removed making the symbol unrecognizable or even nonexistent. The same symbol is not unrecognizable in all cases. As the images have different sizes, in some cases part of the symbol is detected as a line. To prevent this from happening, the line removal will need to take into account the box size.



Figure 4.30: Examples of line removal in GD&T symbols.

Table 4.9: Precision of GD&T symbols recognition

Symbol	Geometric Characteristic	Precision	Total Images
	Flatness	1.00	81
	Parallelism	1.00	66
	Profile of a line	1.00	80
	Position	0.89	63
	Circularity (Roundness)	0.88	74
	Total runout	0.86	76
	Cylindricity	0.80	83
	Circular runout	0.52	65
	Angularity	0.38	69
	Symmetry	0.29	68
	Straightness	0.28	69
	Perpendicularity	0.27	70
	Profile of a surface	0.11	73
	Concentricity	0.00	64

Figure 4.31 shows some FCF boxes with line removal. Removing lines does not remove anything important to the image except the GD&T symbol as previously shown.

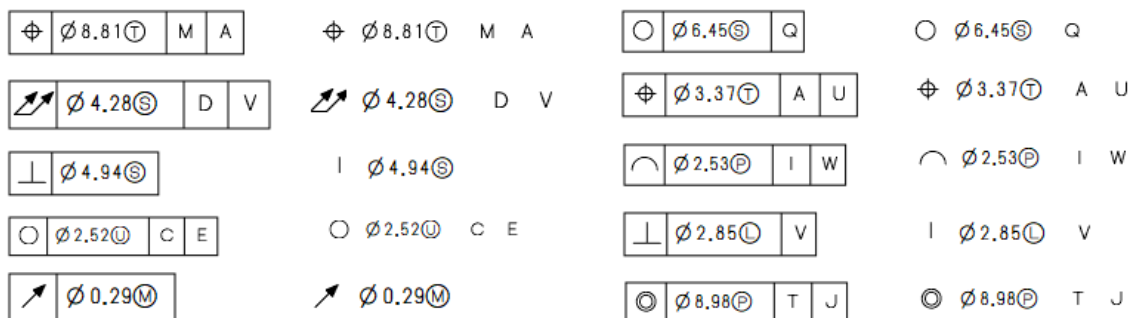


Figure 4.31: Examples of line removal in FCF.

Figure 4.32 shows some FCF boxes with line removal.

⊕uigimⓂMK	oungenⓂK
//uiksiⓂ	@G331®@AU
ⓂukgkkⓂ	∩ingnkⓂWB
ofngnnⓂKN	92.880VV
7G0.23@	ⓂikgnkⓂWU

Figure 4.32: Result of FCF text recognition.

Figure 4.33 shows a strange text recognition. With only one character less the recognition is almost perfect, but with a symbol, in the beginning, the recognition is very bad. If the symbol is placed at the end of the text, the recognition improves but not so good when it did not have the character at the end.

Ⓜ2.67Ⓜ4.56	binisagnkn
2.67Ⓜ4.56	2.670 4.56
2.67Ⓜ4.56 Ⓜ	2.6104.56 LY

Figure 4.33: Bug in Tesseract text recognition or an overfitting problem.

Using the edit distance metric, for FCF text recognition, the precision is 0.1683 and recall is 0.1553. This is a very low precision value, but could not verify why some plain text detections mixed with GD&T symbols are very bad.

## Chapter 5

# Conclusions and Future work

---

5.1 Conclusion . . . . .	87
5.2 Future Work . . . . .	89

---

### 5.1 Conclusion

During this work, I can verify how the CNN based algorithms can learn visual information inside images, but a large amount of images but some type of classification or annotations is required.

There are few technical images available and are not annotated. It is also difficult to get these images from customers because of intellectual property and industrial secrecy reasons. For this reason, was considered a better option to generate synthetic images, but generating good images also takes much development time. With the current generated images was obtained good performance, but the detection model performance could be better if the generated images were more diverse and closer to real images, and with rotated shapes and text.

Training with large numbers of images also requires a high-performance computer, a good graphics card and a lot of memory. Even with good hardware resources, the training process can take many hours or days, so the process of imaging, training and then testing the model can be time-consuming. With the equipment I had available, the faster training process took 7 hours, but with some `glmaskrcnn` settings and data augmentation, it took over 48 hours.

Using the Mask Region-based Convolutional Neural Networks (Mask R-CNN) for shape detection seems to have been a good choice since good results were obtained, although only with synthetic data. It seems that with better and more similar images with real images the result will be even better.

The choice of Long Short-Term Memory (LSTM), implemented by Tesseract, to perform Optical Character Recognition (OCR) does not seem to have been a good choice as it did not give satisfactory results in all cases. I could have used a Convolutional Neural Network (CNN) to perform character classification and recognize character by character, but in this situation, Tesseract had good results. The poor results were obtained when text recognition was applied to entire Feature Control Frame (FCF) data. It was not possible to clarify the reason for this. One way to try to solve this problem is to identify the various parts of the FCF box and perform an OCR on each of them individually.

Detecting arrows that relate text information to shape also had good results. It has failed to detect dimension indicator lines and their dimensions. These indications also differ between different technical drawings and as these indications have some similarities with the arrows already detected, there may be some problems in their correct identification.

With the obtained results, the text area recognition functionality can be implemented in CAM2 software, even if in some case with the help of the user and manually associated with the respective shape (a feature in CAM2 software), which will help to avoid mistakes and speed up the user input process, and also the possibility of detected areas to be corrected manually. With the permission of customers, this data will be sent and stored for future training of the Mask R-CNN.



## 5.2 Future Work

During this work, I just did some research to know the available options. With more images and data validated by the customer, the Mask R-CNN can be trained using the transfer learning to enhance the model performance.

Another possibility that may help in the generation of synthetic but real-based data is to use client files that already have GDT information associated with a shape and project CAD image by relating both with an arrow. But this assumes that customers are willing to upload files with this data and at this time the percentage of customers using this feature will be only 10%.



# References

- [1] Yaron Hadad. 30 amazing applications of deep learning. <http://www.yaronhadad.com/deep-learning-most-amazing-applications/>, 2017.
- [2] Anthony J. Lockwood. Automate inspection processes and reports. <https://www.digitalengineering247.com/article/automate-inspection-processes-reports/>, 2014.
- [3] Using ocr to extract geometric tolerances. <https://help.inspectionxpert.com/help-guides/gdtocr>, 2019.
- [4] Kevin Ma Jonathan O'Hare, James Luther. Optical character recognition reader for manufactured drawings. <https://web.uri.edu/elecomp-capstone/project-details-by-team/2017-2018/hexagon-ocr/>, 2018.
- [5] Rajalingappaa Shanmugamani. *Deep Learning for Computer Vision*. Packt, 2018.
- [6] Abien Fred Agarap. Avoiding the vanishing gradients problem using gradient noise addition. <https://towardsdatascience.com/avoiding-the-vanishing-gradients-problem-96183fd03343>. Accessed: 2019-10-01.
- [7] Jason Brownlee. How to fix the vanishing gradients problem using the relu. <https://machinelearningmastery.com/how-to-fix-vanishing-gradients-using-the-rectified-linear-activation-function/>, . Accessed: 2019-10-01.
- [8] B. Mehlig. Artificial neural networks. *arXiv arXiv:1901.05639v2*, 01 2019.
- [9] Ben Dickson. What are artificial neural networks (ann). <https://bdtechtalks.com/2019/08/05/what-is-artificial-neural-network-ann/>. Accessed: 2019-10-02.

- [10] Facundo Bre, Juan Gimenez, and Víctor Fachinotti. Prediction of wind pressure coefficients on building surfaces using artificial neural networks. *Energy and Buildings*, 158, 11 2017. doi: 10.1016/j.enbuild.2017.11.045.
- [11] Christopher Williams John McGonagle, George Shaikouski. Backpropagation. <https://brilliant.org/wiki/backpropagation/>. Accessed: 2019-10-02.
- [12] Michael A. Nielsen. *Neural Networks and Deep Learning*, chapter How the backpropagation algorithm works. Determination Press, 2015.
- [13] Cross entropy. [https://en.wikipedia.org/wiki/Cross\\_entropy](https://en.wikipedia.org/wiki/Cross_entropy).
- [14] Connor Shorten. Machine learning vs. deep learning. <https://towardsdatascience.com/machine-learning-vs-deep-learning-62137a1c9842>. Accessed: 2019-10-02.
- [15] Mark Johnson. How much data is enough? <http://web.science.mq.edu.au/~mjohnson/papers/Johnson17Power-talk.pdf>, 2017. Accessed: 2019-10-06.
- [16] Scott Martin. What is transfer learning? <https://blogs.nvidia.com/blog/2019/02/07/what-is-transfer-learning/>. Accessed: 2019-10-02.
- [17] Lisa Torrey and Jude W. Shavlik. Chapter 11 transfer learning. 2009.
- [18] Hadi Keivan Ekbatani, Oriol Pujol, and Santi Seguí. Synthetic data generation for deep learning in counting pedestrians. In *Proceedings of the 6th International Conference on Pattern Recognition Applications and Methods, ICPRAM 2017, Porto, Portugal, February 24-26, 2017.*, pages 318–323, 2017. doi: 10.5220/0006119203180323. URL <https://doi.org/10.5220/0006119203180323>.
- [19] Dr. Mohan .H .S Surendra .H. A review of synthetic data generation methods for privacy preserving data publishing. *INTERNATIONAL JOURNAL OF SCIENTIFIC TECHNOLOGY RESEARCH VOLUME 6, ISSUE 03*, 03 2017.
- [20] Bill Howe Luke Rodriguez. In defense of synthetic data. *arXiv arXiv:1905.01351v1*, 05 2019.
- [21] K. Hema Parul Vashist. Character recognition with minimum edit distance method. *International Journal of Science and Research (IJSR)*, 04 2013.
- [22] What's ocr? <http://www.dataid.com/aboutocr.htm>. [Online; accessed 20-July-2010].

- [23] Nicomsoft OCR SDK Tutorials. Optical character recognition (ocr) – how it works. <https://www.nicomsoft.com/optical-character-recognition-ocr-how-it-works/>, 02 2012. [Online; accessed 20-September-2010].
- [24] Boris Epshtein, Eyal Ofek, and Yonatan Wexler. Detecting text in natural scenes with stroke width transform. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2963 – 2970, 07 2010. doi: 10.1109/CVPR.2010.5540041.
- [25] Kaizhu Huang Hong-Wei Hao Xu-Cheng Yin, Xuwang Yin. Robust text detection in natural scene images. *arXiv arXiv:1301.2628v3*, 2013.
- [26] He Wen Yuzhi Wang-Shuchang Zhou Weiran He Jiajun Liang Xinyu Zhou, Cong Yao. East: An efficient and accurate scene text detector. *arXiv arXiv:1704.03155v2*, 2017.
- [27] Sumit Saha. A Comprehensive Guide to Convolutional Neural Networks. <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>, 2018. [Online; accessed 19-July-2010].
- [28] Danny Ho Tong Zhou-Max Q-H. Meng DeLong Zhu, Tingguang Li. A novel ocr-rcnn for elevator button recognition. *IEEE*, 01 2019. doi: 10.1109/IROS.2018.8594071.
- [29] Prem Natarajan Ekraam Sabir, Stephen Rawls. Implicit language model in lstm for ocr. *arXiv arXiv:1805.09441*, 2018.
- [30] Andreas Dengel Martin Jenckel, Saqib Syed Bukhari. Transcription free lstm ocr model evaluation. *IEEE*, 08 2018. doi: 10.1109/ICFHR-2018.2018.00030.
- [31] Jianfeng Wang. Gated recurrent convolution neural network for ocr. <https://papers.nips.cc/paper/6637-gated-recurrent-convolution-neural-network-for-ocr.pdf>, .
- [32] Manish Prasad Thapliyal. Vanishing gradients in recurrent neural networks. <https://medium.com/mlrecipes/vanishing-gradients-in-recurrent-neural-networks-b231c2afde4>.
- [33] Jason Brownlee. How to fix the vanishing gradients problem using the relu. <https://machinelearningmastery.com/>

- how-to-fix-vanishing-gradients-using-the-rectified-linear-activation-function/, .
- [34] Nir Arbel. How lstm networks solve the problem of vanishing gradients. <https://medium.com/datadriveninvestor/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577>.
- [35] Caglar Gulcehre Dzmitry Bahdanau Fethi Bougares Holger Schwenk Yoshua Bengio Kyunghyun Cho, Bart van Merriënboer. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv arXiv:1406.1078v3*, 2014.
- [36] Tingwu Wang. Semantic segmentation. [http://www.cs.toronto.edu/~tingwuwang/semantic\\_segmentation.pdf](http://www.cs.toronto.edu/~tingwuwang/semantic_segmentation.pdf), .
- [37] Trevor Darrell Jitendra Malik Ross Girshick, Jeff Donahue. Rich feature hierarchies for accurate object detection and semantic segmentation tech report (v5). *arXiv arXiv:1311.2524v5*, 2014.
- [38] Edward Ma. How r-cnn works on object detection? <https://medium.com/towards-artificial-intelligence/how-r-cnn-works-on-object-detection-443679b0187c>.
- [39] Ross Girshick. Fast r-cnn. *arXiv arXiv:1504.08083v2*, 2015.
- [40] Ross Girshick Jian Sun Shaoqing Ren, Kaiming He. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv arXiv:1506.01497v3*, 2016.
- [41] Trevor Darrell Jonathan Long, Evan Shelhamer. Fully convolutional networks for semantic segmentation. *arXiv arXiv:1411.04038v2*, 2014.
- [42] Priya Dwivedi. Semantic segmentation — popular architectures. <https://medium.com/towards-artificial-intelligence/how-r-cnn-works-on-object-detection-443679b0187c>.
- [43] Piotr Dollár Ross Girshick Kaiming He, Georgia Gkioxari. Mask r-cnn. *arXiv arXiv:1703.06870v3*, 2018.
- [44] Ross Girshick Ali Farhadi Joseph Redmon, Santosh Divvala. You only look once: Unified, real-time object detection. *arXiv arXiv:1506.02640v5*, 2016.

- [45] Dumitru Erhan Christian Szegedy Scott Reed Cheng-Yang Fu Alexander C. Berg Wei Liu, Dragomir Anguelov. Ssd: Single shot multibox detector. *arXiv arXiv:1512.02325v5*, 2016.
- [46] Python. <https://www.python.org/>.
- [47] Jupyter. <https://jupyter.org/>.
- [48] Opencv. <https://opencv.org/>.
- [49] Tensorflow. <https://www.tensorflow.org/>.
- [50] Keras. <https://keras.io/>.
- [51] Tesseract. <https://github.com/tesseract-ocr/tesseract>.
- [52] Keras. Lecture4:Thresholding.
- [53] Line detection in python with opencv | houghline method. <https://www.geeksforgeeks.org/line-detection-python-opencv-houghline-method/>.
- [54] Waleed Abdulla. Mask r-cnn for object detection and instance segmentation on keras and tensorflow. [https://github.com/matterport/Mask\\_RCNN](https://github.com/matterport/Mask_RCNN), 2017.
- [55] Ross Girshick Kaiming He Bharath Hariharan-Serge Belongie Tsung-Yi Lin, Piotr Dollár. Feature pyramid networks for object detection. *arXiv arXiv:1612.03144v2*, 2017.
- [56] Shaoqing Ren Jian Sun Kaiming He, Xiangyu Zhang. Deep residual learning for image recognition. *arXiv arXiv:1512.03385*, 2015.
- [57] Serge Belongie Lubomir Bourdev Ross Girshick-James Hays Pietro Perona Deva Ramanan C. Lawrence Zitnick Piotr Dollár Tsung-Yi Lin, Michael Maire. Microsoft coco: Common objects in context. *arXiv arXiv:1405.0312v3*, 2015.
- [58] Jeremy Jordan. Evaluating image segmentation models. <https://www.jeremyjordan.me/evaluating-image-segmentation-models/>, 2018.
- [59] Renu Khandelwal. Computer vision: Instance segmentation with mask r-cnn. <https://medium.com/datadriveninvestor/computer-vision-instance-segmentation-with-mask-r-cnn-7983502fcad1>, 2018.

- [60] Jonathan Hui. map (mean average precision) for object detection. [https://medium.com/@jonathan\\_hui/map-mean-average-precision-for-object-detection-45c121a31173](https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173), 2018.
- [61] Romain Karpinski, Devashish Lohani, and Abdel Belaid. Metrics for Complete Evaluation of OCR Performance. In *IPCV'18 - The 22nd Int'l Conf on Image Processing, Computer Vision, & Pattern Recognition*, Las Vegas, United States, July 2018. URL <https://hal.inria.fr/hal-01981731>.
- [62] Levenshtein distance. [https://en.wikipedia.org/wiki/Levenshtein\\_distance](https://en.wikipedia.org/wiki/Levenshtein_distance).





# Appendix A

## Synthetic images

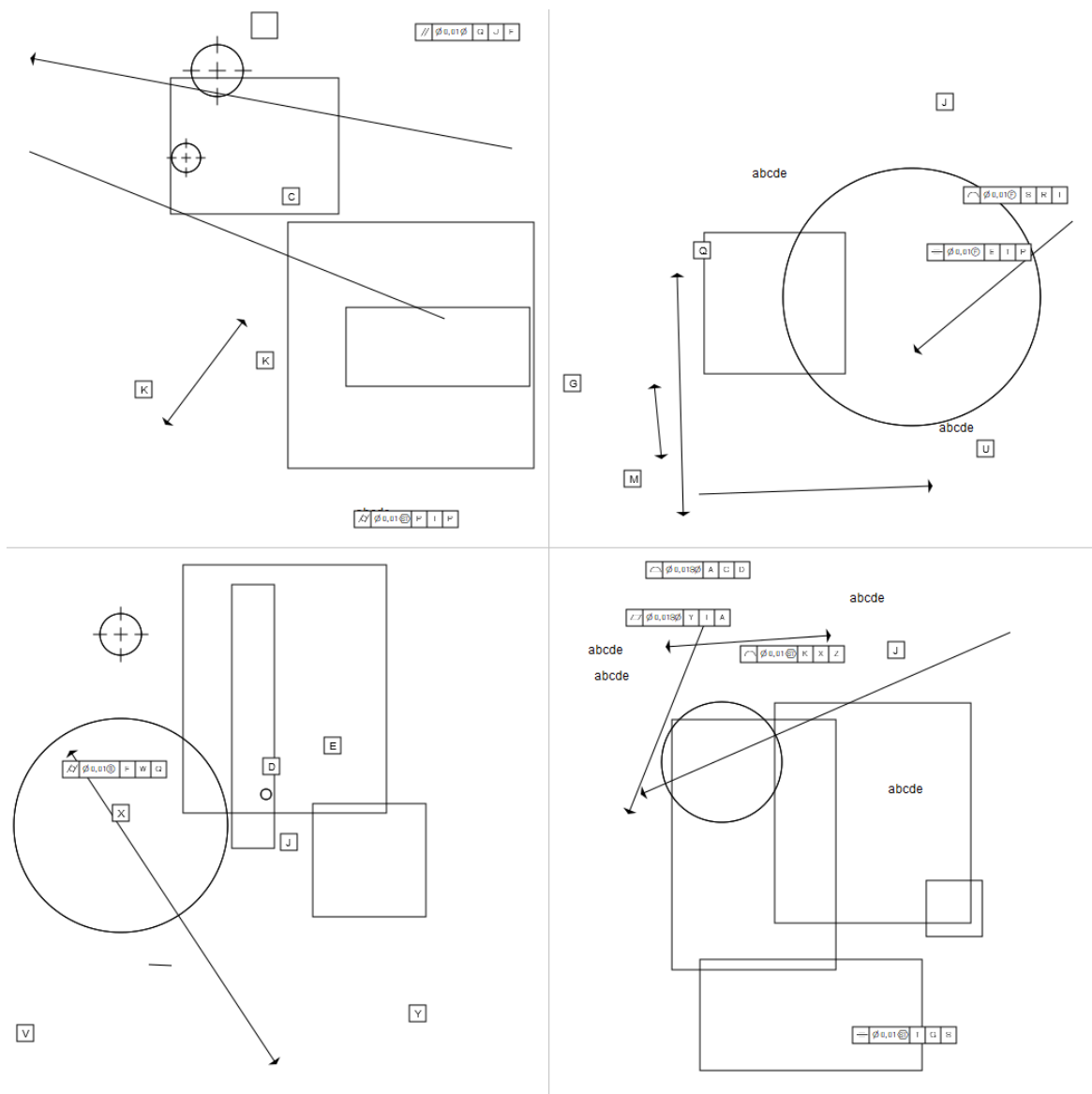


Figure A.1: Sample of 4 images used for training.



## Appendix B

# Shape detection precision matrix

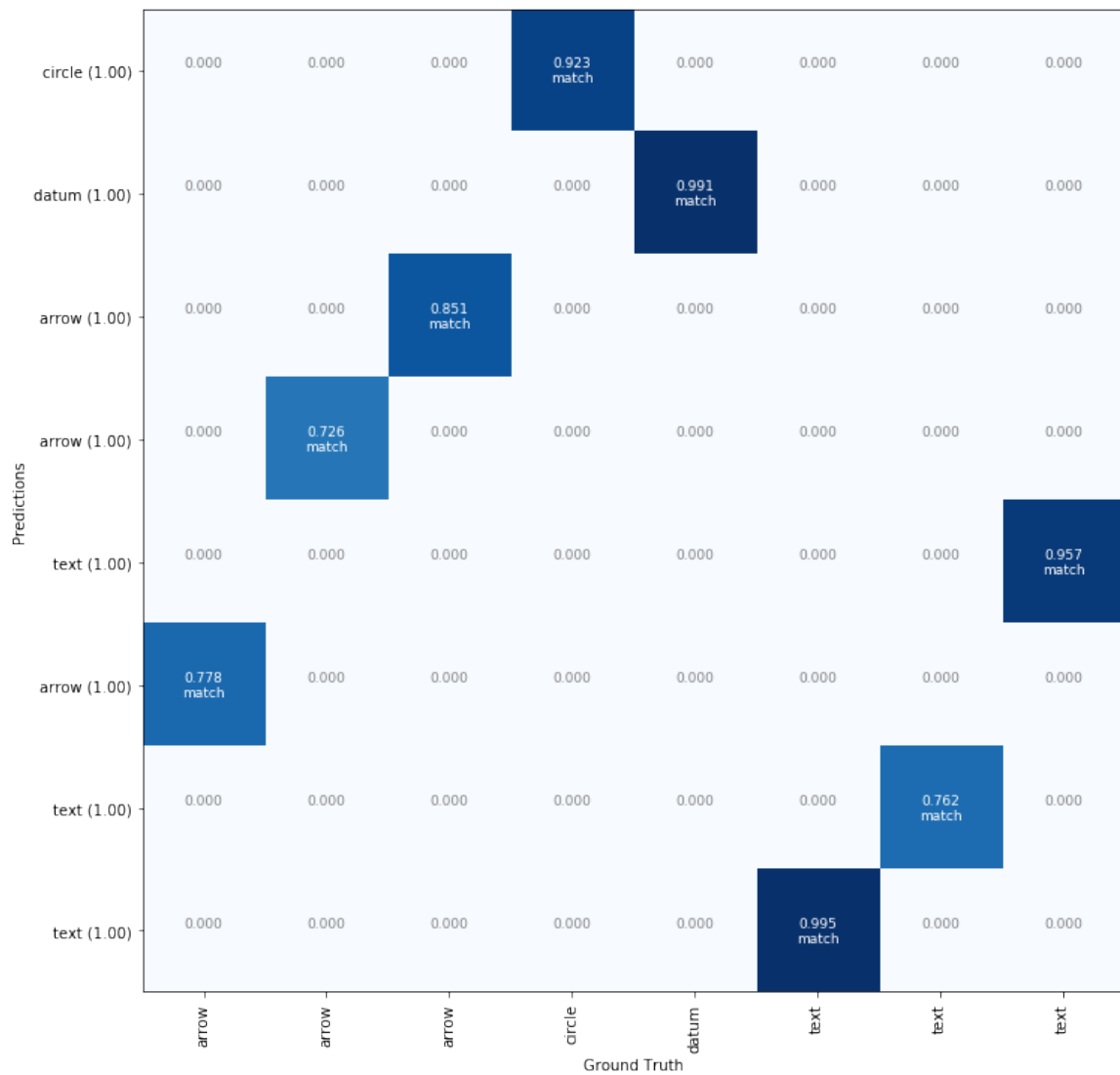


Figure B.1: Confusion matrix of a sample image (1).

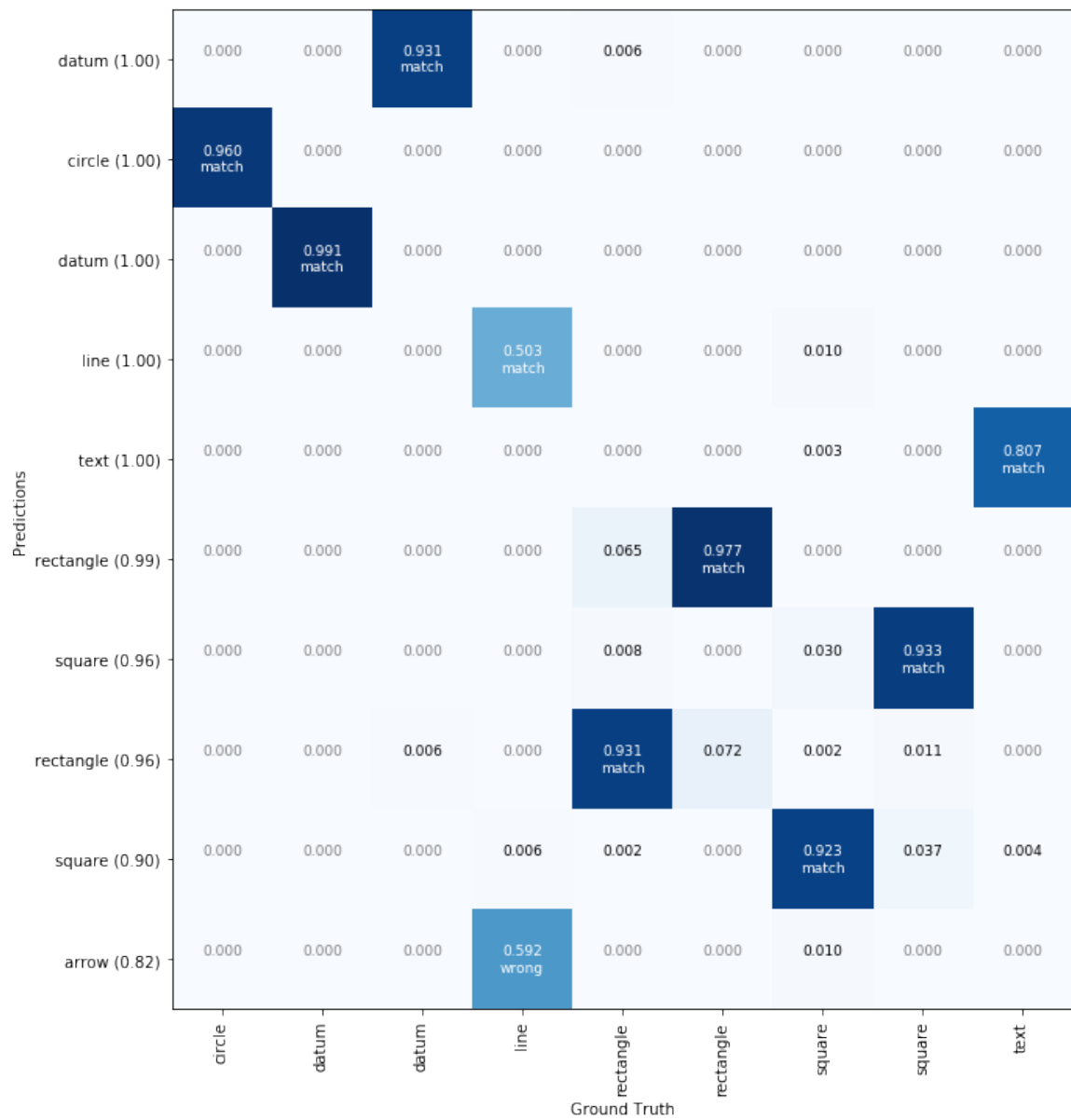


Figure B.2: Confusion matrix of a sample image (2).

